SEVENTH FRAMEWORK PROGRAMME
Theme ICT-1-1.4 (Secure, dependable and trusted infrastructures)

WORLDWIDE OBSERVATORY OF
MALICIOUS BEHAVIORS AND ATTACK THREATS

D12 (D5.1) Root Causes Analysis

Contract No. FP7-ICT-216026-WOMBAT

Workpackage	WP5 - Threats Intelligence
Author	Engin Kirda
Version	2.3
Date of delivery	M21
Actual Date of Delivery	M23
Dissemination level	Public
Responsible	Eurecom

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n°216026.

SEVENTH FRAMEWORK PROGRAMME

Theme ICT-1-1.4 (Secure, dependable and trusted infrastructures)

The WOMBAT Consortium consists of:

France Telecom	Project coordinator	France
Institut Eurecom		France
Technical University Vienna		Austria
Politecnico di Milano		Italy
Vrije Universiteit Amsterdam		The Netherlands
Foundation for Research and Technology		Greece
Hispasec		Spain
Research and Academic Computer Network		Poland
Symantec Ltd.		Ireland
Institute for Infocomm Research		Singapore

Contact information:
Dr. Marc Dacier
2229 Route des Cretes
06560 Sophia Antipolis
France

e-mail: Marc_Dacier@symantec.com
Phone: +33 4 93 00 82 17

Contents

Abstract

This deliverable aims at giving an overview of existing techniques for root cause analysis, and provides some preliminary results with respect to the root cause analysis work performed in the project so far. The deliverable is mainly made up of 6 published peer-reviewed papers and one technical report that has reached a wide-audience.

1 Introduction

Anecdotal evidence indicates the existence of Internet companies and service providers that are under the influence of criminal organizations or knowingly tolerate their activities. Such companies typically control a number of networks with public IP addresses that are abused for a wide range of malicious activities. One such activity is offering bullet-proof hosting, a service that guarantees the availability of hosted resources even when they are found to be malicious or illegal. These hosting services are often used for phishing purposes or for serving exploits and malware. Other malicious activities involve the sending of spam, hosting scam pages, or providing a repository for pirated software and child pornography.

An example of a rogue network that offered bullet-proof hosting was the Russian Business Network (RBN), who made headlines in late 2007 [8, 25]. Various sources alleged that the RBN hosted web sites, exploits, and malware that were responsible for a significant fraction of online scams and phishing. Once publicly exposed, the RBN ceased its operations in St. Petersburg, only to relocate and resume activities in different networks [17]. More recently, a report exposed Atrivo (Intercage), a US-based company that is frequently considered to provide hosting for malicious content [1, 26]. Often referred to as the RBN of the United States, this company is considered to be a "dedicated crime hosting firm whose customer base is composed almost, or perhaps entirely, of criminal gangs" [21]. Again, public outcry quickly lead reputable ISPs to severe their peering relationships with Atrivo, effectively cutting this rogue network off the Internet.

Obviously, rogue networks and bullet-proof hosting providers are only one component of the flourishing underground economy, which is responsible for many of the security problems that Internet users face. Over the last few years, criminals have increasingly leveraged botnets to hide their tracks [15]. Also, large-scale exploitation (such as the recent wave of SQL injection attacks [19] that affected more than half a million web pages) has lead to a situation where malicious content is unwittingly served by many benign, compromised Internet hosts. These hosts are often combined into fast-flux networks to increase the availability of malicious sites and executables [22].

Despite the large numbers of bot-infected machines and compromised servers, rogue networks do play an important role in the underground economy. These networks often house back-end machines (called motherships) that serve scam pages and exploits, while

bots and compromised web pages act as proxies. In this setup, criminals hide their malicious servers behind a layer of bots that can be easily replaced when they are taken down or cleaned up [23]. In addition, the content is located at a central location, which eases management. For example, it is straightforward to check for multiple accesses from the same IP. Often, subsequent accesses to malware pages are redirected to benign sites (such as `msn.com`). This makes life more difficult for human malware analysts, but also foils client-side honeypots that require multiple accesses to the same site to determine malicious pages[1].

Root cause analysis is a class of problem solving methods aimed at identifying the root causes of problems or events. With respect to security and Internet-based attacks, root cause analysis tries to identify which groups, organizations, or machines are behind the attacks and what methods are being used by the attackers.

The partners of WOMBAT have been very active in WP5 which deals with threats intelligence. That is, we have been developing novel techniques and methods to improve the state of the art in root cause analysis research.

This deliverable aims at giving an overview of existing techniques for root cause analysis, and provides some preliminary results with respect to the root cause analysis work performed in the project so far.

As research publications are not only the best approach to disseminate the results of a research project, but also to validate the novelty and quality of the work thanks to peer reviews and competitive selection by experts, we have been very active in publishing papers on the techniques developed within WP5. To provide a good overview of the preliminary results we have achieved, we have decided to include the project papers that have been published in this area. Note that besides the Symantec threats report (that has been published as a whitepaper that is available to a wide audience and received a wide coverage in the media world-wide), all of the papers that we have listed have been published at well-known, peer-reviewed workshops and conferences.

The deliverable is structured as follows. The next section provides a technical survey of root cause analysis in the context of the WOMBAT project. More bibliographic references and detailed revisions of the state of the art are offered in each paper appearing in the following section. Section 3 first briefly discusses the papers that we have included in the deliverable, and then offers a copy of each of them. Finally, Section 3 concludes the deliverable.

[1] For performance reasons, the well-known honeyclient Capture-HPC [36] visits a set of pages in parallel before checking for malicious system modifications. Once an unwanted modification is observed, all pages in a set have to be individually re-visited a second time to determine the ones that are malicious.

2 Technical Survey of Root Cause Analysis

2.1 Introduction

There is currently no universally agreed definition for "attack attribution" in the cyber domain. If one looks at a general definition of the term *attribution* in a dictionary, one will find something similar to: "to explain by indicating a cause" [Merriam-Webster]. However, we note that most previous work related to that field tend to use the term "attribution" as a synonym for *traceback*, which consists in "determining the identity or location of an attacker or an attacker's intermediary" [48].

In the context of a cyber-attack, the obtained identity can refer to a person's name, an account, an alias, or similar information associated with a person or an organization. The location may include physical (geographic) location, or any virtual address such as an IP address or Ethernet address. In other words, *(IP) traceback* is a process that begins with the defending computer and tries to recursively step backwards in the attack path toward the attacker so as to identify her, and to subsequently enable appropriate protection measures (or even counter-attack actions in some cases). The rationales for developing such attribution techniques are mainly due to the untrusting nature of the IP protocol, in which the source IP address is not authenticated and can thus be easily spoofed. For this reason, most existing approaches dealing with IP traceback have been tailored toward (D)DoS attack detection, or to some specific cases of targeted attacks performed by a human attacker who uses stepping stones or intermediaries in order to hide her true identity.

Some typical methods used for IP traceback include packet marking techniques (i.e., marking packets as they traverse routers through the Internet [35, 40, 6]), logging packets or maintaining information on routers situated on the path between the attacker and the defender [39], traceback of active attack flows (that is, by marking data flowing back to the attacker, e.g., using packet watermarking techniques [45]), and reconfiguring network paths during an attack (for instance with controlled flooding of links [10], to determine how this flooding affects the attack stream). An extensive survey of attack attribution techniques used in the context of IP traceback has been done by Wheeler and Larsen, under the umbrella of the US Defense Technical Information Center [48].

We refer to "attack attribution" as something quite different from what is described

here above, both in terms of techniques and objectives. Although tracing back to an ordinary, isolated hacker is an important issue, we are primarily concerned by larger scale attacks that could be mounted by criminal organizations, dissident groups, rogue corporations, and profit-oriented underground organizations. In this context, we aim at developing an effective and systematic method that can help security analysts to determine the root causes of global attack phenomena (which usually involves a large amount of sources), and to easily derive their *modus operandi*. These phenomena can be observed through many different means (e.g., honeypots, IDS's, sandboxes, web crawlers, malware collecting systems, etc). In most cases, we believe that attack phenomena manifest themselves through so-called "attack events", which can be observed with well-placed distributed sensors. Typical examples of attack phenomena that we want to identify can go from malware families that propagate in the Internet through code injection attacks, to zombie armies (or botnets) controlled by the same people and targeting machines in the IP space to recruit new bots, or even to certain client-side threats such as rogue software campaigns run by the same organization, which aims at deploying numerous malicious websites (or compromising legitimate ones) in order to host and sell rogue software.

Attack phenomena are often largely distributed in the Internet, and their lifetime can vary from only a few days to several months. They typically involve a considerable amount of features interacting sometimes in a non-obvious way, which makes them inherently complex to identify. That is, due to their changing nature, the attribution of distinct events having the same root phenomenon can be a challenging task, since several attack features may evolve over time. To address this problem, we developed a method that is based on a novel combination of a graph-based knowledge discovery technique with a multi-criteria decision analysis process.

As noted by Tim Bass in [5], "Next-generation cyberspace intrusion detection (ID) systems will require the fusion of data from myriad heterogeneous distributed network sensors to effectively create cyberspace situational awareness [...] Multisensor data fusion is a multifaceted engineering approach requiring the integration of numerous diverse disciplines such as statistics, artificial intelligence, signal processing, pattern recognition, cognitive theory, detection theory, and decision theory. The art and science of data fusion is directly applicable in cyberspace for intrusion and attack detection". Not surprisingly, our methods are at the crossroads of several active research domains, which we can try to categorize as follows:

i) *investigative and security data mining*, i.e., knowledge discovery and data mining (KDD) techniques that are specifically tailored to problems related to computer security or intelligence analysis;

ii) problems related to *multi criteria decision analysis* (MCDA), and multisensor data fusion;

iii) general techniques for *malicious traffic analyses* on the Internet, with an emphasis on methods that aim to improve the "cyber situational awareness" (Cyber-SA).

In the next paragraphs, we give an overview of some key contributions in each research area.

2.2 Investigative and security data mining

2.2.1 Security data mining

In the last decenny, a considerable research effort has been devoted to applying data mining techniques to security-related problems. However, a large part of this effort has been exclusively focused on the improvement of intrusion detection (ID) systems via data mining techniques, rather than on the discovery of new fundamental insights into the nature of attacks or their underlying root causes, as noted by Julisch in [4]. Furthermore, only a subset of common data mining techniques (e.g., association rules, frequent episode rules or classification algorithms) have been applied to intrusion detection, either on raw network data (such as ADAM [2], MADAM ID [27, 28] and MINDS [18]), or on intrusion alerts streams [16, 24]. A comprehensive survey of DM techniques applied to ID can be found in [3, 9].

All previous approaches aimed at improving alerts classification or intrusion detection capabilities, or at constructing better detection models by (automatically) generating new rules (e.g., using some inductive rule generation mechanism). Our work within the context of WP5 is quite different in many aspects. First, we take advantage of data sets that contain only malicious activities (high and low-interaction honeypots, malware samples, honeyclients, etc). Secondly, we use a graph-based, unsupervised data mining technique to discover *a priori* unknown attack patterns performed by groups or communities of attackers. Then, our objective does not consist in generating new IDS rules to better protect a single network, but instead to determine the root causes of large-scale attack phenomena (e.g., worm, botnet, etc) observed by distributed sensors, and to get insights into their global behavior, i.e., how long do they stay active, what is their average size, their spatial distribution, and more importantly, how do they evolve over time with respect to their origins, or the type of activities they perform?

2.2.2 Crime data mining

There are many similarities between the tasks performed by analysts in computer security and in crime investigations or law-enforcement domains. Several researchers have, thus, explored the possibilities of DM techniques to assist law-enforcement professionals. In [29], McCue provides real-world examples showing how data mining has identified crime trends and helped crime investigators in refining their analysis and decisions. Previous to that work, in [30] Jesus Mena has described and illustrated the usefulness of data mining as an investigative tool by showing how link analysis, text mining, neural networks and other machine learning techniques can be applied to security and crime detection. Finally, a more recent book from Westphal provides even more real-world applications of crime data mining, such as border protection, money laundering, financial crimes or fraud analytics, and it describes also the advantages of using information-sharing protocols and systems in combination with these analytical methods [46].

We observe, however, that most previous work in the crime data mining field has primarily focused on "off the shelf" software implementing traditional data mining techniques (such as clustering, classification based on neural networks and Kohonen maps, or link analysis). Still, Chen et al. have conducted some active research in crime data mining in the context of the COPLINK project [12], using text mining, neural networks and Social Network Analysis (SNA) on different case studies. In our graph-based approach, we can see some similarity with link analysis methods used in crime data mining. However, there are also many differences: for instance, how relationships are created in classical link analysis tools is quite straightforward (usually, using the output of simple comparisons between basic features), whereas we use an aggregation function to combine multiple correlation patterns found in different graphs in order to identify more complex relationships. Furthermore, our approach can be applied to many different types of features vectors, even to statistical distributions.

Finally, there are also some obvious relationships between our graph-based clustering technique and Social Network Analysis (SNA), which has been recognized as an appropriate methodology to uncover previously unknown structural patterns from social or criminal networks. SNA heavily relies on the usage of network graphs and link analysis as key techniques to analyze social communities and networks. Different metrics are used to emphasize the characteristics of a social group and its members, such as centrality, betweenness, closeness, structural cohesion of actors, clustering coefficient, etc [34, 41]. In this context, analysis of *n-cliques*, *n-clans*, *k-plexes*, or more generally "connected component", can reveal interesting subgroups within a network or a community that are strongly connected in the graph representation, i.e., network members sharing many common traits. Probably for those reasons, SNA has been ranked in the

top 5 intelligence analysis methods by K. Wheaton, assistant professor of intelligence studies at Mercyhurst College [47]. Some of our techniques are admittedly inspired by SNA, namely the clique-based clustering of attackers. However, we use a novel, efficient clique algorithm based on dominant sets, and our attribution method can combine multiple graphs (reflecting multi-dimensional features) into a combined graph by using a multicriteria aggregation function, which enables us to model more complex relationships among coalitions of features (e.g., an interdependency between two or more features). As far as we know, this kind of processing is not yet available in traditional SNA techniques.

2.3 Multicriteria decision analysis

2.3.1 MCDA applied to security problems

In our approaches, we have formalized the attribution problem as an application of *multi criteria decision analysis* (MCDA), in which the criteria of concern are given by the links (or distance) values computed during the graph-based clustering (which is performed for each attack feature). That is, we use the distance values between two events as *degrees of evidence* (or fuzzy measures) to decide whether or not they are likely due the same root phenomenon. As such, it can be considered as a classical multi attribute decision making problem where a decision (or an alternative) has to be chosen based on several, eventually conflicting criteria. A combined output is evaluated based on different attributes (or features), which are expressed numerically and can sometimes be obtained as the output of a fuzzy system (e.g., when we need to model vagueness and uncertainty about a given attribute). It is worth noting that MCDA has also been ranked in the top 5 intelligence analysis methods by K. Wheaton [47].

In this formalization, we need, thus, to define an appropriate function that can model a certain decision scheme matching as closely as possible the phenomenons under study. In many MCDA systems, the aggregation process is a sort of averaging function, like a simple weighted means (e.g., Simple Additive Weighting, Weighted Product Method, Analytical Hierarchy Process [52]), or the Ordered Weighted Average (OWA) [49, 7], and Choquet or Sugeno integrals [7]. ELECTRE, TOPSIS and PROMETHEE [20] are three well-known outranking methods that are based on a similar aggregation process. These techniques aim at selecting or ranking different alternatives by using multiple criteria weighted by coefficients. However, one could choose also conjunctive or disjunctive functions (such as *t-norms* and *t-conorms*), or mixed functions (such as *uninorms* and *nullnorms*) to model the aggregation of criteria in more complex systems [7].

We have chosen the Ordered Weighted Average (introduced by Yager [49]) as aggregation function to model more complex relationships among criteria, for example “at

least three" attack criteria to be satisfied in the overall decision function. The power of such an aggregation function lies in the fact that different combinations of criteria may apply to each pair of events. Furthermore, the decision-maker does not need to specify in advance which criteria (or features) must be satisfied to link two events to the same phenomenon.

Despite their great flexibility in combining features or evidences, we note that rather few previous works have used MCDA approaches in order to address security-related problems. However, in [11] the authors consider the problem of discovering anomalies in a large-scale network based on the *data fusion* of heterogeneous monitors. They evaluate the usability of two different approaches for multisensor data fusion: one based on the Dempster-Shafer Theory of Evidence and one based on Principal Component Analysis. The *Dempster-Shafer* theory is a mathematical theory of evidence [37] based on belief functions and plausible reasoning. It allows one to combine evidence from different sources and to obtain a certain degree of belief (represented by a belief function) that takes into account all the available evidence. It can be seen as a generalization of Bayesian inference where probability distributions are replaced by *belief functions*, which can thus provide a certain degree of belief (also referred to as a *mass*). When used as a method for sensor fusion, different degrees of belief are combined using Dempster's rule which is a generalization of the special case of Bayes theorem where events are independent. In our attribution method, we prefer using aggregation functions as described previously, for the greater flexibility they offer in defining how we want to model interactions between different criteria. Moreover, in our case we have absolutely no certainty that events observed by different sensors are always independent.

2.4 Malicious traffic analysis

2.4.1 Research on monitoring darknet traffic

Our research builds also on prior work in malicious traffic analysis, for which the literature in this field is quite significant. For example, in [50], Yegneswaran et. al. have studied the global characteristics and prevalence of Internet intrusions by systematically analyzing a set of firewall logs (from D-Shield) collected from a wide perspective (over four months of data collected from many different networks worldwide). Their study is a general analysis that focused on the issues of volume, distribution (e.g., spatial and temporal), categorization and prevalence of intrusions. Then, in [31] Pang et al. characterize the incessant nonproductive network traffic (which they term Internet *background radiation*) that can be monitored on unused IP subnets when deploying network telescopes or more active responders such as honeypots. They analyzed temporal patterns

and correlated activity within this unsolicited traffic, and they found that probes from worms heavily dominate. More recently, similar research has been conducted by Chen et al. in[13]. While all these previous works provide meaningful results and have much contributed in making advances in malicious traffic analysis, the traffic correlation and analysis techniques used by the authors stay at a fairly basic level. Indeed, they basically break down the components of background radiation by protocol, by application and sometimes by specific exploit, and then apply some statistics across each component. Whereas we apply more elaborated techniques on honeynet traffic, such as graph clustering based on statistical distances, combined with multi-criteria analysis in order to elevate the abstraction level, and to improve the insights into global phenomena.

On our side, we previously developed in [42] an efficient clique-based clustering method to extract groups of correlated attack clusters from a large honeynet dataset, and in [43, 44] we explored two different approaches to combine attack knowledge extracted through these means. Moreover, we have also presented in [32, 33] different signal processing techniques that can be used to extract systematically interesting *attack events* from a large set of honeynet traces.

It would be incomplete to discuss attack attribution without mentioning some active research carried out in Cyber Situational Awareness (or Cyber-SA). We acknowledge the seminal work of Yegneswaran and colleagues in this field, such as in [51] where they explore ways to integrate honeypot data into daily network security monitoring, with the purpose of effectively classifying and summarizing the data to provide ongoing situational awareness on Internet threats. However, their approach aims at providing tactical information, usable for the day to day operations, whereas we are interested in strategic information that reveal long term trends and the modus operandi of the attackers. Closer to our research, Li et. al. have described in [53] a framework for automating the analysis of large-scale botnet probing events and worm outbreaks using different statistical techniques applied to aggregated traffic flows. They also design schemes to extrapolate the global properties of the observed scanning events (e.g., total population and target scope) as inferred from the limited local view of a honeynet. Finally, a first compilation of scientific approaches for Cyber-SA has recently been published in [38], in which a multidisciplinary group of leading researchers (from cyber security, cognitive science, and decision science areas) try to establish the state of the art in cyber situational awareness and to set the course for future research. The goal of this pioneering book is to explore ways to elevate the situation awareness in the Cyber domain. We have contributed to [38] with a chapter on Macroscopic Cyber Situational Awareness, in which we present our extensive data collection infrastructure and illustrate the usefulness of applying a multidimensional analysis to the attack events detected by our honeypots.

Finally, another related project that looks interesting is Cyber-Threat Analytics (Cyber-

TA), founded by SRI International [14]. Cyber-TA is an initiative that gathers several reputed security researchers. It aims at accelerating the ability of organizations to defend against Internet-scale threats by delivering technology that will enable the next-generation of privacy-preserving digital threat analysis centers. According to Cyber-TA, these analysis centers must be fully automatic, scalable to alert volumes and data sources that characterize attack phenomena across millions of IP addresses, and give higher fidelity in their ability to recognize attack commonalities, prioritize, and isolate the most critical threats. However, very few information is available at [14] on which scientific techniques could enable organizations to achieve such goals or to elevate their cyber situation awareness.

2.5 Research on identifying malicious networks

Although much work has been done on studying malicious activity on the Internet (such as phishing, drive-by-download exploits, and malware-based scams), not much focus has been put on automatically identifying the networks and infrastructures used by the attackers. With the novel work we present in this paper, we approach the problem from a different angle and hope to help prevent victims from accessing or receiving traffic from networks that have proven to be malicious in nature.

3 Preliminary Results (Published Papers)

In this section, we first briefly summarize the papers that we have included in the deliverable in order to provide some preliminary results of the ongoing work in WP5. Then, we provide full copies of these papers. Note that each paper has a related work section that throughly describes the existing work in the problem domain that is being addressed.

In the following, we list the papers, sorted by publication date, that have been accepted for publication and that we have included in this deliverable. It is worth pointing out that papers 2 and 5 have each received the best paper award of the conference where they have been presented. Paper 7 is an invited keynote speech (with a paper published in the proceedings) in the most prestigious conference in India.

1. Olivier Thonnard, Marc Dacier Actionable knowledge discovery for threats intelligence support using a multi-dimensional data mining methodology ICDM'08, 8th IEEE International Conference on Data Mining series, December 15-19, 2008, Pisa, Italy , pp 154-163

2. Olivier Thonnard, Wim Mees, Marc Dacier, Addressing the attack attribution problem using knowledge discovery and multi-criteria fuzzy decision-making KDD'09, 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Workshop on CyberSecurity and Intelligence Informatics, June 28th-July 1st, 2009, Paris, France , pp 11-21

3. Thorsten Holz, Markus Engelberth, Felix Freiling, Learning More About the Underground Economy: A Case-Study of Keyloggers and Dropzones, 14th European Symposium on Research in Computer Security (ESORICS 2009), Saint Malo, Brittany, France, September 2009

4. M. Dacier, C. Leita, O. Thonnard. V. H. Pham and E. Kirda, "Assessing Cybercrime Through the Eyes of the WOMBAT", chapter 3 (pp. 103-136) in Cyber Situational Awareness: Issues and Research, Sushil Jajodia, Peng Liu, Vipin Swarup, Cliff Wang, eds., ISBN: 98-1-4419-0139-2, Springer International Series on Advances in Information Security, 2009.

5. Van-Hau Pham, Marc Dacier, Honeypot traces forensics : the observation view point matters, NSS 2009, 3rd International Conference on Network and System Security, October 19-21, 2009, Gold Cost, Australia

6. Brett Stone-Gross, Andreas Moser, Christopher Kruegel, Kevin Almaroth, Engin Kirda, FIRE: FInding Rogue nEtworks, 25th Annual Computer Security Applications Conference (ACSAC), Honolulu, December 7-10, 2009

7. The WOMBAT Attack Attribution method: some results, M. Dacier, V. H. Pham and O. Thonnard, Fifth International Conf. On Information Systems Security (ICISS 2009), 14-18 December 2009,Kokata, India (Invited Keynote talk with paper in the proceedings).

8. The Symantec Public Report on Rogue Security Software, (July 08 - June 09), October 2009

In the following sections, we provide an executive summary of each paper, and briefly discuss its relevance to root cause analysis. Each listed paper covers a specific problem domain in root cause analysis.

3.1 Actionable Knowledge Discovery for Threats Intelligence Support using a Multi-Dimensional Data Mining Methodology

This first paper that was published at ICDM'08. It describes the initial idea of using a multi-dimensional knowledge discovery and data mining (KDD) methodology that aims at discovering actionable knowledge related to Internet threats, taking into account domain expert guidance and the integration of domain-specific intelligence during the data mining process. The objectives are twofold: i) to develop global indicators for assessing the prevalence of certain malicious activities on the Internet, and ii) to get insights into the modus operandi of new emerging attack phenomena, so as to improve our understanding of threats. In this paper, we first present the generic aspects of a domain-driven graph-based KDD methodology, which is based on two main components: a clique-based clustering technique and a concepts synthesis process using cliques' intersections. Then, to evaluate the applicability of the approach to our application domain, we use a large dataset of real-world attack traces collected since 2003.

With respect to root cause analysis, our experimental results show that significant insights can be obtained into the domain of threat intelligence by using this multi-dimensional knowledge discovery method. This is the seminal paper on our attack

attribution method which has been slightly modified and enriched in the following specification.

3.2 Addressing the Attack Attribution Problem using Knowledge Discovery and Multi-criteria Fuzzy Decision-Making

In network traffic monitoring, and more particularly in the realm of threat intelligence, the problem of "attack attribution" refers to the process of effectively attributing new attack events to (un)known phenomena, based on some evidence or traces left on one or several monitoring platforms. Real-world attack phenomena are often largely distributed on the Internet, or can sometimes evolve quite rapidly. This makes them inherently complex and thus difficult to analyze. In general, an analyst must consider many different attack features (or criteria) in order to decide about the plausible root cause of a given attack, or to attribute it to some given phenomenon.

The paper we have listed that was accepted at KDD'09 introduces a global analysis method to address this problem in a systematic way. The approach is based on a novel combination of a knowledge discovery technique with a fuzzy inference system, which mimics the reasoning of an expert by implementing a multi-criteria decision-making process built on top of the previously extracted knowledge. By applying this method on attack traces, we are able to identify large-scale attack phenomena with a high degree of confidence. In most cases, the observed phenomena can be attributed to so-called zombie armies - or botnets, i.e. groups of compromised machines controlled remotely by a same entity.

With respect to root cause analysis, by means of experiments with real-world attack traces, the paper shows how this method can effectively help us to perform a behavioral analysis of those zombie armies from a long-term, strategic viewpoint.

3.3 Learning More About the Underground Economy: A Case-Study of Keyloggers and Dropzones

In this paper that was accepted at ESORICS 2009, we study an active underground economy that trades stolen digital credentials. In particular, we investigate keylogger-based stealing of credentials via dropzones, anonymous collection points of illicitly collected data. Based on the collected data from more than 70 dropzones, we present an empirical study of this phenomenon, giving many first-hand details about the attacks that were

observed during a seven-month period between April and October 2008. We found more than 33 GB of keylogger data, containing stolen information from more than 173,000 victims.

With respect to root cause analysis, analyzing this data set helps us better understand the attacker's motivation and the nature and size of these emerging underground marketplaces.

3.4 Assessing Cybercrime Through the Eyes of the WOMBAT

In this paper that was a book chapter, we offer an early synthesis of the various results that have been obtained when analyzing a large amount of information. However, in order for this chapter to be as self-contained as possible, it starts by re-stating the rationales for this work, as well as by providing a summarized introduction to the data collection infrastructure. It has been produced at the very beginning of 2009 for a presentation in March 2009 at George Mason University at the ARO Cyber Situational Awareness Workshop (Fairfax, Virginia). This led, afterwards, to this book chapter.

3.5 Honeypot traces forensics : the observation view point matters

In this paper, that was accepted to NSS 2009 for publication, we dig deeper in the prerequisites for attack attribution which is the identification of "attack events" in network traces. A method is proposed to identify and group together traces left on low interaction honeypots by machines belonging to the same botnet(s) without having any a priori information about these botnets. In other terms, the paper offers a solution to detect new botnets thanks to very cheap and easily deployable solutions. The approach is validated with several months of data collected with the worldwide distributed Leurre.com system maintained by the WOMBAT project. To distinguish the relevant traces from the other ones, groups are created according to either the platforms, or the countries of origin of the attackers.

With respect to attacks event identification, the paper shows that the choice of one of these two observations view points dramatically influences the results obtained. That is, each one can help in revealing unique botnets. The paper elaborates on why this is the case.

3.6 FIRE: FInding Rogue nEtworks

In this paper that was accepted at ACSAC 2009, we present FIRE, a novel system to identify and expose organizations and ISPs that demonstrate persistent, malicious behavior. The goal is to isolate the networks that are consistently implicated in malicious activity from those that are victims of compromise. To this end, FIRE actively monitors botnet communication channels, drive-by-download servers, and phishing web sites. This data is refined and correlated to quantify the degree of malicious activity for individual organizations.

With respect to root cause analysis, these results can be used to pinpoint and to track the activity of rogue organizations, preventing criminals from establishing strongholds on the Internet. Also, the information can be compiled into a null-routing blacklist to immediately halt traffic from malicious networks.

3.7 The WOMBAT Attack Attribution method: some results

In this paper which was a keynote speech at ICISS 09, we present a new attack attribution method that has been developed within the WOMBAT project together with some initial results. This analytical method aims at identifying large scale attack phenomena composed of IP sources that are linked to the same root cause. All malicious sources involved in a same phenomenon constitute what we call a Misbehaving Cloud (MC). The paper offers an overview of the various steps the method goes through to identify these clouds, providing pointers to external references for more detailed information. Four instances of misbehaving clouds are then described in some more depth to demonstrate the meaningfulness of the concept.

3.8 The Symantec Public Report on Rogue Security Software

The Symantec Report on Rogue Security Softwareis an in-depth analysis of rogue security software programs. This includes an overview of how these programs work and how they affect users, including their risk implications, various distribution methods, and innovative attack vectors. It includes a brief discussion of some of the more noteworthy scams, as well as an analysis of the prevalence of rogue security software globally. It also includes a discussion on a number of servers that Symantec observed hosting these misleading applications. Except where otherwise noted, the period of observation for this report was from July 1, 2008, to June 30, 2009.

With respect to root cause analysis, the Symantec report provides significant insights into the ways the attackers are organized and are operating on the Internet today.

More importantly, its second half presents the application of the WOMBAT attack attribution method to a distinct set of initial attack events than the ones used so far. Indeed, instead of looking at identified attack events, in network traces, we now apply the method to identified malicious websites and show how the very same method can be applied to derive meaningful information on the strategies and modus operandi of the malicious actors behind these sites.

Actionable Knowledge Discovery for Threats Intelligence Support using a Multi-Dimensional Data Mining Methodology

Olivier Thonnard
Royal Military Academy
Polytechnic Faculty
Brussels, Belgium
Olivier.Thonnard@rma.ac.be

Marc Dacier
Symantec Research Labs
Sophia Antipolis, France
marc_dacier@symantec.com

Abstract

This paper describes a multi-dimensional knowledge discovery and data mining (KDD) methodology that aims at discovering actionable knowledge related to Internet threats, taking into account domain expert guidance and the integration of domain-specific intelligence during the data mining process. The objectives are twofold: i) to develop global indicators for assessing the prevalence of certain malicious activities on the Internet, and ii) to get insights into the modus operandi of new emerging attack phenomena, so as to improve our understanding of threats. In this paper, we first present the generic aspects of a domain-driven graph-based KDD methodology, which is based on two main components: a clique-based clustering technique and a concepts synthesis process using cliques' intersections. Then, to evaluate the applicability of this approach to our application domain, we use a large dataset of real-world attack traces collected since 2003. Our experimental results show that significant insights can be obtained into the domain of threat intelligence by using this multi-dimensional knowledge discovery method.

Keywords: Internet threat intelligence, domain-driven data mining, knowledge discovery

1. Introduction

Recently, the security community has been facing what appears to be highly organized and professional malicious activities on the Internet. It has been reported that, motivated by financial profit, today's cybercriminals seem to be building a new and growing underground economy by offering commoditization of activities such as the sale of 0-day exploits and new yet-undetected malware, the sale of compromised hosts, spamming, phishing, etc [12]. For security researchers, this leads to the observation of increasingly coordinated attack activities, which are often related to botnets [37], stealthy multi-headed worms [29] or other sophisticated emerging threats. Client's applications, typically web-browsers and email applications, become also a common infection vector for propagating new malwares that in turn aim at scanning and recruiting more vulnerable machines into zombie armies, which seem to be the preferred weapon of cybercriminals today.

There are several data collection initiatives that offer plausible indicators supporting those claims. However, these data sources are often built in an ad-hoc way to study a specific problem. In fact, the security community seems to lack two important things regarding threats evaluation: *i)* unbiased, meaningful and publicly available data about Internet threats, and *ii)* global threat analysis techniques that can offer real scientific answers to open questions and speculations circulating in the community. Similarly to criminal forensics, the security analyst needs to synthesize different pieces of evidence in order to investigate the root causes of attack phenomena. This is a tedious, lengthy and informal process mostly relying on the analysts expertise. For those reasons, we seek to develop a multi-dimensional knowledge discovery and data mining (KDD) methodology that should help us to improve, in a more systematic way, our understandings of new Internet threats. Our idea consists in *i)* extracting relevant nuggets of knowledge by mining a complex dataset according to different properties considered as relevant by a domain expert; and in *ii)* synthesizing those pieces of knowledge so as to create higher-level concepts describing the underlying phenomena.

The remainder of this paper is organized as follows: in Section 2, we report on related work. In Section 3, we present the theoretical foundations of our method. Section 4 describes our experimental environment. Section 5 presents the lessons learned when applying our technique to a large dataset of real-world attack traces. Finally, we conclude in

Section 6.

2. Related Work

This work is at the crossroads of several domains of expertise. Regarding Internet threats, there are, broadly speaking, three main approaches to monitor, collect and analyze network threats: *i)* low- or high-interaction honeypots [35, 34, 36, 1, 42], which are vulnerable computers intentionally set up as traps to attract and observe attackers on the Internet; *ii)* the so-called Internet telescopes, or darknets [27, 39, 38, 33, 8], which are used in order to monitor all unsolicited traffic directed to unused IP subnets; and *iii)* projects of collecting and sharing firewall and IDS logs gathered from a very large number of heterogeneous sources [11]. This work builds upon a broad experience in this specific security domain [25, 23, 29, 31, 30]. Then, in [32], we investigated the usability of a clique-based technique to group together network traces that share some specific features, namely packet inter-arrival times (IAT's), and more recently in [41] we developed an efficient graph-based clustering method to extract groups of correlated attack time series from an extensive honeynet dataset. We acknowledge the seminal work of Yegneswaran and colleagues on "Internet situational awareness" [44], in which they explore ways to integrate honeypot data into daily network security monitoring, with the purpose of effectively classifying and summarizing the data to provide ongoing situational awareness. Their approach aims at providing tactical information, usable for the day to day operations whereas we are interested in strategic information that reveal long term trends and the modus operandi of the attackers. Another closely related work is *BotMiner* [14], a general botnet detection framework that is independent of botnet C&C protocol and structure, and requires no a priori knowledge of botnets. The authors developed a prototype system that is based on: i) a two-steps clustering (based on X-Means) of C&C communication and activity flows of bots, so as to detect similarity patterns; and ii) the combination of both types of patterns by means of cross-correlation. Our research is different as we do not focus exclusively on the problem of detecting botnets, but instead we aim at understanding the higher-level modus operandi of global attack phenomena (e.g., which "communities" of machines are possibly involved in what type of activities, on which networks they are hosted, etc.).

In the past ten years, a growing number of research projects have applied data mining to various problems in the security field, but almost exclusively in intrusion detection rather than honeynets. Furthermore, most research has focused on the construction and the improvement of operational IDSs via data mining techniques, rather than on the discovery of new and fundamental insights into the nature of attacks [3]. Only a few well-known data mining techniques (e.g., association rules, frequent episode rules or clustering algorithms) have been widely used in intrusion detection, either on raw network data (such as ADAM [2], MADAM ID [21], and [22]), or on intrusion alerts streams [18, 10]. Our work is very different, both in terms of techniques and objectives. We seek to develop a domain-driven knowledge discovery method that could help us to better understand and characterize the modus operandi of Internet threats from a global perspective, rather than focusing on a technique to improve the detection rate of an IDS on one given network.

Finally, some facets of our work are related to some other graph-theoretical data mining techniques, such as the hypergraph model used for clustering of data in a high-dimensional space [15], or the hyperclique pattern discovery approach for mining association patterns [43, 16]. In both cases, all data properties are used together in the graph partitioning algorithm to create hypergraph structures, while in our case we adopt a bottom-up approach by combining different sets of one-dimensional cliques obtained for each property separately.

3. A Methodology for Multi-Dimensional Knowledge Mining

3.1. Overview

This section presents our approach in general terms which will be instantiated according to our concrete application domain requirements in Section 5. The proposed methodology consists of two steps:

1) An unsupervised clique-based clustering of data objects according to well-defined properties. This component aims at finding all groups of highly similar patterns within an object dataset with respect to a single property each time. A domain expert is required to define the possible interesting properties of the dataset. The clusters are formed via the extraction of maximal cliques from a graph.

2) A concepts synthesis process using cliques' intersections, which can be seen as a data fusion process by which different combinations of dataset properties are computed so as to create higher-level concepts.

The clustering in step 1 is not applied directly to the raw datasets but to complex data patterns we derive from it. By "complex pattern", we mean an aggregated, higher-level data structure that already represents a certain abstraction of the dataset. A complex pattern is supposed to carry some semantic regarding the measured phenomena. Statistical distributions, for example the geographical distribution

of a sampled population, or the aggregated time series of a dynamic process, are some examples of complex patterns, as opposed to simple numerical or categorical features such as the weight, the color or a stock value.

Each such pattern is represented as a node in a *graph* where every *edge* represents a *similarity* relationship between two nodes. A graph-based clustering is then performed via the extraction of *cliques*, which are complete subgraphs, for all properties identified as potentially relevant by a domain expert. The idea is to create N sets of cliques where the members of each clique share a highly similar characteristic pattern created along one of the N defined properties.

Following the clustering process, we synthesize the patterns by combining *different* sets of cliques. This leads to the creation of meta-groups, which are termed *concepts*, and where group members have one or more similarity patterns in common. The original cliques are considered as groups of dimension 1. Meta-groups of dimension 2 (resp. 3,..., N) are obtained by combining 2 (resp. 3,..., N) properties. A detailed description of each component of the methodology is provided in the next paragraphs.

3.2. Clique-based Clustering

The first component of our knowledge mining methodology involves a graph-theoretic clustering. Typical clustering tasks involve the following steps [17]: *i)* feature selection and/or extraction, and pattern representation; *ii)* definition of a similarity measure between patterns; *iii)* grouping similar patterns; *iv)* data abstraction (if needed), to provide a compact representation of each cluster; *v)* the assessment of the clusters quality and coherence (if needed).

In any clustering task, we must select certain features characterizing relevant aspects of the dataset, i.e., salient features that may provide meaningful *patterns*. Those patterns are represented with *feature vectors*, which are usually built with formatted data series, or simply arrays of values. There are two key aspects in the clustering process herein presented: *i)* even complex patterns, such as statistical distributions, may be easily used in the clustering algorithm, and *ii)* the types of features used to create different patterns may (and even should) be quite different, introducing thus a certain diversity in the classification. Once the sets of patterns are created, we need to measure the similarity between two patterns. For that purpose, several types of similarity distances are available (e.g., Mahalanobis, Minkowski, Pearson or Spearman correlations, jackknife correlation, etc.). Clearly, the choice of a similarity metric must be carefully determined in consideration of the original data series and the expected properties of the clusters, such as the cluster size, quality, or consistency. In Section 5, we present a few similarity measures we use in practice in our domain-specific application. The following step consists in grouping all patterns that look very similar. There exists a plethora of clustering algorithms for doing this. We use here an unsupervised graph-theoretic approach to formulate the problem, and the clustering is then performed by extracting *maximal weighted cliques* from a graph. To the best of our knowledge, this type of clustering has not been widely covered in previous KDD applications, yet it is in our opinion a convenient and appropriate formulation for solving domain-driven data mining problems, and it has several advantages over other more classical approaches such as K-Means or Bayesian classification, especially when dealing with high-dimensional datasets [15].

A graph is a structure that comprises a set of vertices (or nodes) connected by links called edges, which can be directed or undirected. A *clique* is defined as an induced sub-graph of a (un)directed graph in which the vertices are fully connected. A clique is *maximal* if it is not contained within any other clique.
Hence, finding the largest group of similar elements in a data set can now be transformed into the problem of searching for *complete subgraphs* where the vertices represent the patterns, and the links express the similarity relationships between those vertices. This is a classical NP-complete problem studied in graph-theory, also known as the *maximal clique problem* (MCP) [4]. Because of its NP-hard complexity, many approximate algorithms for solving the MCP have been developed, like local search heuristics, Hopfield network, Ant Colony Optimization, and the heuristic based genetic algorithm, among others.

In this clique-based clustering, we use the *dominant sets* approach of Pavan et al. [28], which proved to be an effective method for finding maximal *weighted* cliques. This means that the weight of every edge is also taken into consideration by the algorithm, as it seeks to discover maximal cliques whose total weight is maximized. This generalization of the MCP is also known as the maximum weight clique problem (MWCP). This approximate method for solving the MWCP aims at finding iteratively dominant sets of maximally similar nodes in the graph. We can show that dominant sets are equivalent to maximum weighted cliques, but finding those dominant sets is far easier to compute. Indeed, this can be done with a continuous optimization technique, which applies replicator dynamics (from evolutionary game theory). As a result, we can solve the problem of extracting dominant sets by simply making a particular temporal expression converge. Let for instance A be a non-negative real-valued $n \times n$ matrix that represents the adjacency matrix of the graph introduced here above, and consider the following dynamical system represented

with its discrete time equation:

$$x_i(t+1) = x_i(t) \cdot \frac{(Ax(t))_i}{x(t)^T Ax(t)}, i = 1, ..., n$$

Starting from an arbitrary initial state, this replicator dynamical system will eventually be attracted by the nearest asymptotically stable point. As it has been proven in [28], this corresponds to a dominant set, hence to a maximum weight clique. In our global knowledge discovery process, for the N identified properties, we apply this clique-based clustering on each edge-weighted graph.

3.3. Concepts Synthesis via Cliques Intersections

The second component of our methodology is similar to a dynamic data fusion process. Starting from all sets of cliques, the idea is to combine k sets out of the N dimensions, with $k = 2, ..., N$, in order to discover actionable knowledge about certain phenomena.

To introduce this concepts synthesis, let us consider some notions used in Formal Concept Analysis (FCA). There is a strong parallel between our KDD method and FCA, since the cliques, and any combination thereof, can be seen as the formal representation of concepts describing a certain phenomenon (or at least some aspect hereof). In FCA [13], a *concept* is defined as the combination of both an object cluster, which comprises all objects that share a common subset of attributes, and a property cluster, which is the set of all properties shared by all the object clusters. Let us consider for example a set of objects $O = \{O_1, \ldots, O_n\}$, and a set of properties $P = \{P_1, P_2, \ldots, P_N\}$ with:

$$P_1 = \{p_{1,1}, \ldots, p_{1,k_1}\}, P_2 = \{p_{2,1}, \ldots, p_{2,k_2}\}, \ldots$$

$$P_N = \{p_{N,1}, \ldots, p_{N,k_n}\}$$

The different subsets of patterns $\{p_{i,j}\}$ correspond to the different feature vectors that are extracted for each property P_i.

A basic example of dataset properties and their associated patterns could be as follows: $P = \{color, shape, nr\_edges\}$, and :

$$P_1 = \{blue, red, yellow\}$$

$$P_2 = \{line, square, circle, ellipse\}$$

$$P_3 = \{0, 1, 2, 3, 4, 5, others\}$$

An example of a dimension 3-concept can be defined as the set of all objects sharing the following (unordered) values for the 3 properties: $\{red, square, 4\}$. With this example, we emphasize also the fact that we do not consider the case of a boolean concept lattice, but we generalize rather to the case of properties characterized by discrete sets of attributes. It is worth noting that, in our domain application, while relevant properties can be defined by a domain expert, the subsets of potential patterns related to those properties (i.e., the $\{p_{i,j}\}$) are completely unknown prior the execution of the clique-based clustering. So, the patterns are discovered via the extraction of cliques along each dataset property. The complete set of concepts is called the concept lattice, and it can be represented with a Hasse diagram. This is illustrated in Figure 1 for a simple case of 10 objects characterized by three properties, each containing one or two different patterns. The boxes filled in light grey can be seen here as the initial cliques (i.e., the dimension-1 concepts), which allowed to extract the *a priori* unknown patterns for each dataset property.

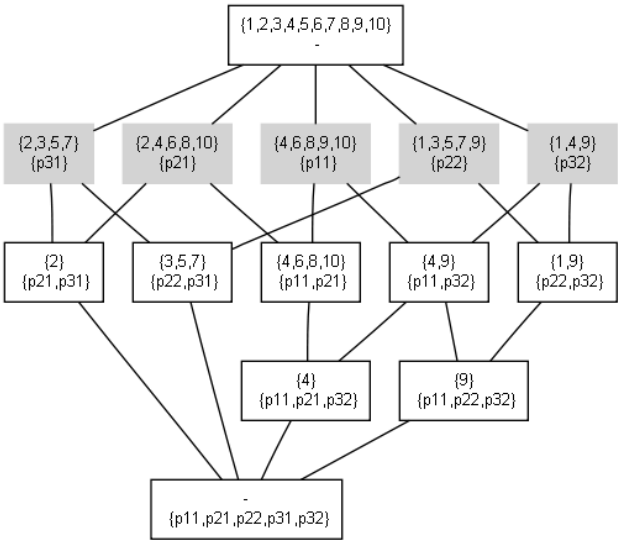


Figure 1. An example of *concept lattice*, represented with a Hasse diagram. In total, 14 concepts have been constructed via the extraction of maximal cliques (in light grey) from the initial dataset containing 10 objects. In each concept, the first line of the label represents the members of the concept (or the *extent*), and the second line is the pattern(s) of the concept (or the *intent*).

Note that there exist many algorithms for generating concept lattices [20], but to the best of our knowledge, our method is the first one that relies on maximal cliques to achieve this goal by discovering a priori unknown intents in a dynamic fashion. Moreover, another advantage is its extensibility. That is, when a practitioner finds a new dataset property to be of interest for the knowledge discovery process, (s)he only needs to include a new set of cliques, inde-

pendently of the existence of previous concepts or cliques. As a result, new combined viewpoints, and thus new formal concepts, are immediately available for assisting the root cause analysis of the phenomena.

4. Honeynet Environment

We describe here the specific dataset we used to validate our multi-dimensional data mining methodology. This unique dataset is made of network attack traces and has been collected in the context of the *Leurre.com Project* [25, 35], a global distributed honeynet. A honeypot is a security resource whose value lies in being probed, attacked, or compromised [40]. Honeypots should have no production value and hence should not see any legitimate traffic or activity. Whatever they capture can then be considered as malicious or at least suspicious. By extension, a network of interconnected honeypots has been termed "honeynet".

Since 2003, a distributed set of identical honeypot platforms, based on honeyd [36], has been deployed in many different countries and on various academic and industrial IP subnets. Recently, a second phase of the project was started with the deployment of high-interaction honeypots based on the ScriptGen [24, 23] technology, in order to enrich the network conversations with the attackers and to intercept code injections, which may lead in some cases to the retrieval of malicious binaries used by the attackers. The *Leurre.com* dataset is publicly available for any researcher under the condition of a Non-Disclosure Agreement that aims at protecting the privacy of the partners involved in the deployment of those honeypot platforms.

A platform runs three virtual honeypots, each one has its own public IP address and they emulate different operating systems (two Windows and one Linux machine) with various common services faking to be open. The collected traffic, including the payloads of the packets, is automatically stored into an Oracle database. The network traces are also enriched with contextual information (geographical location of the attackers, ISP's, domain names, etc). All IP sources are grouped into so-called *attack clusters* [31] built according to the network traces they have left when talking to the honeypot. Each such *cluster* is defined thanks to network characteristics such as the number of virtual machines targeted on a platform by a given IP, the number of packets and bytes sent to each honeypot, the attack duration, the average inter-arrival time between packets, the associated port sequence being probed by the attacker, and the packet payload (when available).

Our work builds upon this notion of *clusters*, as defined in [31], but in the rest of this document, to avoid any ambiguity with our own clique-based clustering technique, we use the expression *attack profile* or simply *attack* instead of *cluster*. In other terms, an *attack profile*, or *attack*, consists of a group of IP addresses that have targeted at least one of the *Leurre.com* platforms and have left very similar network traces when talking to that platform.

In [29], it has been shown that the IPs found in a given *attack profile* could be linked to distinct attack phenomena happening during successive, limited periods of time on each sensor. In the rest of this paper, we use the terms *attack events* to refer to the subset of IPs from a given attack profile on a sensor, and observed within a specific time window identified thanks to the method presented in [29]. Namely, the experiments presented in this paper are based on 351 *attack events* found in a timeframe spanning from September 2006, until June 2008. Those attack events have targeted 36 different sensors, which are located in 20 different countries and spread over 18 subnetworks on the Internet. In this dataset we observed a total of 282,363 distinct sources, distributed over 136 different types of attack profiles.

5. KDD Application for Threats Intelligence Support

5.1. Clique-based Clustering

We first present the different properties that we have selected to cluster the attack events together. We motivate this choice based on domain experience in monitoring malicious traffic. Then, for each dimension, we briefly describe how we have applied the clique-based clustering, more specifically: *i)* which type of patterns (and representation hereof) do we consider, *ii)* how can we measure the similarities between them, and *iii)* what are the results in terms of cliques and what type of insights do they deliver. In the last section, we take advantage of cliques' intersections to synthesize higher-level concepts describing some attack phenomena observed on our sensors.

5.1.1 Geolocalization of Attackers. The geographical location of the attackers can be used to identify attack activities having a specific pattern in terms of originating countries. Such information can be important to identify, for instance, botnets that are located in a limited number of countries. It is also a way to confirm the existence, or not, of so-called safe harbors for the hackers.

Patterns Selection. For every attack event, we generate a feature vector that represents the attacking sources' distribution of all IPs found in that attack event, grouped by country of origin (in absolute values). Concretely, for each attack event, we build an histogram whose elements are labeled with the ISO 3166-1 country codes and we identify how many source IPs belong to each country (Figure 2-Left illustrates such a geographical pattern).

Distance Metric for Frequency Data. To measure how similar two attack events are, with respect to that specific property, we need an appropriate distance metric. In this case, we rely on non-parametric statistical tests to compare those empirical distributions (i.e., the histograms). In our application, for each pair of distributions, we use a combination of three different statistical tests to obtain the distance: first, we compute the maximal p-value according to the Pearson's χ^2 test and the Kolmogorov-Smirnov (KS) test, and secondly we validate this result with the Kullback-Leibler divergence.

χ^2 and *Kolmogorov-Smirnov* are among the most commonly used non-parametric statistical methods for testing the null hypothesis ($H0$) that the frequency distribution of certain observations of a sample is consistent with a particular hypothesized distribution (also called a test of "goodness of fit"). In other words, those tests are used to determine whether two underlying one-dimensional probability distributions differ in a significant way. The output of both tests is a *p-value*, which is compared against a given significance level to decide if the investigator can safely reject the null hypothesis. Low probability values lead to the rejection of $H0$. Inversely, p-values that are largely above the significance level can be interpreted as an indication of a very strong relationship between the two samples, which means that both samples are very likely coming from the same population.

From our observations, the p-values given by both tests (χ^2 and *Kolmogorov-Smirnov*) are usually very close to each other. Still, under certain circumstances, they can also differ substantially. To solve this issue, we validate the significance of the obtained p-values by computing the *Kullback-Leibler* divergence (also known as the relative entropy [19]) between both distributions. When this divergence tends to be large, we set the similarity value to zero, whatever the result of χ^2 or KS might be; otherwise we keep the maximal p-value as measurement as the similarity degree between the two patterns. This technique appears to be, at least for the datasets we deal with, a quite robust and reliable metric for comparing categorical frequency data.

Geographical Cliques. Running the clique-based clustering on the initial dataset of 351 attack events delivers 45 cliques containing between 2 and 23 attack events. In total, 273 attack events (77%) have been classified into those cliques, accounting for 66% of the total volume of sources. The largest cliques contain about twenty different attack events having exactly the same geographical distribution. From those results, we observe that geographical cliques provide good indications of the prevalence of certain countries to be involved in different specific activities (e.g., US, China, Canada, Korea, Taiwan, Italy, France, Germany, Great-Britain, Brasil, Japan and Russia). Moreover, geographical cliques can be useful to identify *communities* of machines used to perform a given type of activity. Surprisingly enough, we observe also groups of targeted attacks coming from rather small or unexpected countries, such as Poland, Hungary, Romania, Pakistan, Argentina or India[1].

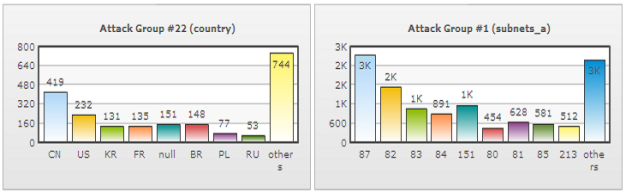


Figure 2. Left: the pattern of a geographical clique of attack events. Right: the pattern of a subnets clique (the labels are the anonymized /8 subnets). All attack events belonging to a same clique have the very same pattern.

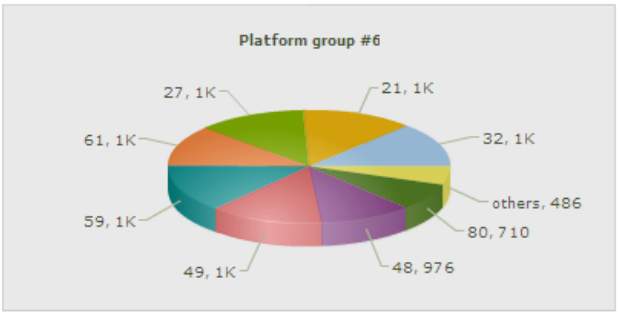


Figure 3. The pattern of a platforms clique of attack events, in which all events have a distribution similar to this one (regarding the targeted platforms in this case).

5.1.2 Netblocks of Origin. The source IP network block is another property that nicely complements the geolocation as described before. Instead of giving insight on possible geostrategic decisions made by the hackers, they can typically reveal some strategies in the propagation model of the malwares. Indeed, attackers' IP subnets can provide a good indication of the spatial "uncleanliness" of certain networks, i.e., the tendency for compromised hosts to stay clustered within unclean networks, especially for zombie machines belonging to botnets as demonstrated in [7]. Previous studies have also demonstrated that some worms show a clear bias in their propagation scheme, such as a tendency

[1] More details about the cliques can be found in an extended technical report available from the Eurecom website (http://www.eurecom.fr/people/dacier.en.htm).

for scanning machines of the same (or nearby) network so as to optimize their propagation [6]. So, for each attack event, we create a feature vector representing the distribution of IP addresses grouped by considering the /8 subnet (which means the first 8-bytes prefix of each IP address). An example of such vector, obtained for a given clique of attack events, is given in Figure 2 (Right). Since we have signed a non-disclosure agreement, we have changed the subnet values while maintaining, as much as possible, the relationships between the discovered netblocks (e.g., consecutive subnets values). As explained in the previous section, we can use the same statistical distances to measure the similarity degree between a pair of subnets histograms.

Subnets Cliques. From the output of the clique-based clustering applied to the subnets dimension, we obtain about 30 cliques; they contain in total 262 attack events (75% of the dataset) accounting for 56% of the total volume of sources. A few cliques are fairly large: about fifty attack events grouped in the same clique and having all the very same subnet distribution. Here also, we observe again many relevant relationships within the attack dataset regarding this dimension. For example, the characteristics of some of those cliques (e.g., the targeted port sequences, etc.), and an additional in-depth analysis, have lead us to conclude that the sources involved in those cliques were apparently members of a larger botnet that has been active during an extended period of time (about three months) in a given Internet region. Those subnets cliques revealed also a sort of dynamism in the affected IP regions (due to new bot infections and computers cleaning) during the lifetime of the botnet, with still some stable clustered IP zones of bot infected machines (i.e., in "unclean networks").

5.1.3 Attack Time Series. Time series analysis can also provide useful information about the underlying attack phenomena [41]. By "attack time series" we mean an aggregated source count for an attack on a given sensor, in a given timeframe. This dataset property can provide indications about synchronized activities targeting different sensors. It can also reveal some typical pattern related to a botnet activity [9]. Finally, discovering synchronized probes on completely different TCP ports (and thus, a priori unrelated attacks) might help to identify multi-headed worms [29], which combine different exploits in a single piece of software.

To include this attack dimension, we have created, for each attack event, a feature vector where each element represents the aggregated source count per day for that specific attack on a given sensor. There are numerous appropriate techniques to compute the similarity between time series, such as singular value decomposition (SVD), piecewise aggregate approximation (PAA), discrete Fourier transform, wavelets, etc. The method we use in this application is an adapted version of SAX (symbolic aggregate approximation) [26]. It falls in the category of PAA techniques which tend to approximate time series by segmenting them into time intervals of equal size and summarizing each of these intervals by its mean value. Each time series (usually of complex shape) is thus replaced by a quantized vector of symbolic values whose shape is by far simpler to process when measuring the similarities among time series. Moreover, SAX provides a lower-bounding distance measure that is easy to interpret, and which can be used in the clustering to decide if two time series are similar or not. More details about the SAX technique can be found in [26], and we refer the interested reader to [41] for a more detailed description of our SAX adaptation.

Cliques of Time Series. The clique-based clustering applied to the 351 events delivered 82 cliques of time series encompassing 92% of the attacking sources, which already confirms the highly organized aspect of activities related to Internet attacks. As noted in [41, 29], we observe only three types of temporal pattern in the resulting cliques: *(i)* a few "voluminous" cliques containing attacks with a continuous activity pattern, primarily due to Messenger Spammers (on UDP ports 1026-1028) and some classical network worms (e.g., Allaple, Slammer, etc.); *(ii)* cliques involving attacks in the form of *sustained bursts*, mainly due to large botnet attack waves or multi-headed worms; and *(iii)* a very large number of small cliques related to *ephemeral attacks* targeting one or a few sensors on the same day, due either to small botnet probes, targeted scan activities or misconfigurations in some rare cases. Figure 4 illustrates the pattern of a clique of the second type, where 21 attack events have targeted 5 different sensors on well-known Windows ports (445T and 139T) for a period of 20 days in December 2006. Even though they target different IP subnets, all those attack events exhibit a quite perfect synchronization. In the light of our detailed analyses, this was attributed to an attack wave of a botnet coming mainly from China, Canada and US.

5.1.4 Targeted Platforms. Apparently, some recent crimeware toolkits are now able to deliver a specific type of malware to different geographical regions [5]. By using this new feature, cybercriminals can thus set up well targeted campaigns by delivering specialized crimeware in specific geographical regions. Indeed, malware may benefit from being adapted to, e.g., the local version of an operating system or application. Therefore, it seems important to look at relationships that may exist between attack events and the platforms they have been observed on.

Attack events are defined per platform. To calculate a feature vector representing the distribution of platforms, we decided to group all strongly correlated attack events within

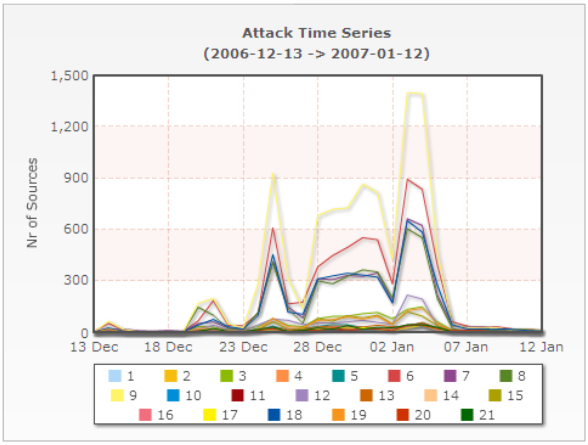


Figure 4. The patterns of an extracted clique of attack time series targeting Windows ports and ICMP (I, I-445T, I-445T-139T). Note the almost perfect synchronization of the attacks on 5 different sensors located in 4 different IP subnets.

its time window of existence, and we then used this group of attack events to create the feature vector representing the proportion of platforms that have been targeted. Fig 3 illustrates the kind of pattern we are looking for in this dimension. Here again, we can use the statistical distance measures introduced previously (in Section 5.1.1) to compare two frequency distributions.

Platforms Cliques. The experiments indicate that 284 events (encompassing 70% of the sources) could be clustered into only 17 cliques for this dimension. According to our in-depth analysis, those cliques helped to discover some nice phenomena too. One such example is related to a large series of ephemeral attack events on high (unusual) TCP ports that were always launched against the very same platform, and where each individual attack event had a quite high intensity but a very short duration (one or two days). A security analyst would most probably disregard such suspicious traffic since each traffic peak is targeting a quite unusual TCP port (on which there is no well-known application running), and unlike worm propagation, those attack events do not sustain an activity over a long period of time. So, without this platform viewpoint, it would be actually very hard to get a global overview of such long-term and stealthy phenomenon. Another finding related to this viewpoint is that several groups of platforms seem to be targeted in a very similar way (thus, by coordinated sources), probably because of their IP proximity, and this hostile traffic represents, in total, a significant volume (70%).

5.2. Concepts Synthesis

So far, we have created N sets of cliques, i.e., one set of cliques for each relevant attack property. As suggested here above, each clique pattern can hold a piece of actionable knowledge about an attack phenomenon, but in some cases the security analyst will have to synthesize different pieces of evidence in order to perform a root causes analysis, and to really understand what happened. Therefore, we can take advantage of all one-dimensional cliques to construct higher-level concepts by simply computing cliques' intersections. Based on the type of phenomenon under scrutiny, the practitioner may include any number of properties in order to create concepts containing more or less semantic meaning. Moreover, those concepts have by construction some relationships with other super- or sub-concepts (from a lower/higher combination level), so a *concept lattice* can be built dynamically to represent the connections between those different concepts describing certain phenomena.

Based on the initial sets of cliques obtained via the clique-based clustering, we have computed the total number of combinations that could, in theory, be derived from those dimension-1 concepts. As this is essentially a combinatorial problem, the theoretical number of combinations grows very quickly, even with as few as 4 dimensions. Indeed, more than 2 millions combinations could exist in theory, based on the number of cliques we have obtained. In practice though, we observe that the number of concepts synthesized at each level is not excessive, and in total we obtain only 0.04% of the theoretical number of combinations. This indicates also that the selected attack properties are not equally distributed and thus seem to carry some meaningful semantics on the observed phenomena, since many cliques' combinations are empty. While the analysis of raw network traces (composed of millions of packets) on each sensor would definitively be impractical, now we observe that the analysis of those concepts can easily provide a more global insight into the real-world phenomena that have caused the attack traffic. To illustrate this, we provide two such examples here under.

Attack concepts analysis - Two case-studies. Table 1 gives an overview of three interesting concepts. The first two concepts belong to the dimension 4 whereas the last one is of dimension 2.

Concepts 1 and 2 justify our initial design choice which was to use attack events, as opposed to attack profiles, for the atomic objects of our datasets. These two concepts are made of nine attack events, each. These nine attack events belong to the same type of attack profile observed over consecutive periods of time. In fact, it is as if we were observing two consecutive waves of the same attack against the same set of targets. The first wave lasts for 30 days in Febru-

Id	Dim.	Nr Events (Extent)	Date (duration)	Patterns (intent)				Nr Sources	Port Sequences
				Time Series	Platforms distri.	Geographical distri	Subnets distri		
1	4	9	2008-01-20 (30 days)	p.m.	32,21,27,61,59, 49,48,80,63	HU,PL,FR,BR	86,84,85,90,83,87,89,203	880	5900T (VNC)
2	4	9	2008-03-11 (7 days)	p.m.	32,21,27,61,59, 49,48,80,63	KR,US,CA,DE	123,89,87,61,90,91,213,222	3,456	5900T (VNC)
3	2	48	2006-09-21 (262 days)	p.m.	50	87,82,83,84,151, 80,81,85,213,others	-	12,305	9763T, 15264T, 29188T, 6134T, 6769T, 7690T 1755T, 50656T, 64264T, 32878T, 64783T, 18462T, 4152T, 25083T, 9661T, 25618T, 28238T, 38009T, 53842T, 64697T, 46030T

Table 1. Some examples of concepts obtained at different semantic levels (the real subnets of origin have been anonymized).

ary while the second starts in March and lasts for only seven days. One could be tempted to consider these two waves as part of a single phenomenon. If that was the case, relaxing the constraint on the time series analysis, i.e. inspecting dimension-3 concepts, would result in having those various attack events ending up in the same concept. This is not what happens. Indeed, the geographical distribution and the subnet distributions also differ between these two concepts. In other terms, even if the modus operandi as well as the targets appeared to be the same, the origins were clearly different. Figuring out if this can be explained by two different botnets being controlled by the same entity or two distinct entities using the same tool lies outside the scope of this paper as it pertains to forensics activities. What matters here is that the approach revealed simply and clearly an important element that can help in the understanding of the observed phenomena.

The second case, illustrated with the concept 3 in Table 1, justifies our motivation to look not only at all dimension-1 and dimension-N concepts, but also to analyze the concepts obtained in between, as there are also phenomena that can emerge at those intermediary semantic levels. There are 48 events involved in concept 3, and all of them have targeted the very same sensor (located in China) at different dates, in the form of very ephemeral spikes of activity, and now they all seem to originate from the same IP netblocks. By raising the dimensional level, this phenomenon would not appear as clearly since the events' time series are not correlated. The events involved in this concept could at first sight appear as Internet noise; but now, a new type of phenomenon clearly emerges. Since it has lasted for at least 262 days, it is quite unlikely that it is due to a pure random process or to background noise. At this stage, there is no obvious reason that could further explain the intent of this fairly large community of machines (12,305 sources), and more viewpoints are probably needed to refine our concept-based root cause analysis. However, those simple examples demonstrate that our technique can effectively highlight stealthier phenomena that would otherwise stay hidden in the Internet background noise.

6. Conclusions

The global analysis of Internet threats is clearly a complex but critical problem, and thus appropriate analysis methods are required in order to effectively get insights into the modus operandi of new emerging attack phenomena. In this work, we have presented a multi-dimensional knowledge discovery and data mining method that can help us to improve our understandings of new Internet threats. Our method consists in *(i)* extracting meaningful nuggets of knowledge by mining a complex dataset according to different properties considered as relevant; and in *(ii)* synthesizing those pieces of knowledge at different dimensional levels, so as to create a *concept lattice* that can best describe real-world phenomena for a domain expert. An experimental validation on real-world attack traces has shown that significant insights can be obtained into the threats intelligence domain thanks to this approach.

The analysis of the concepts has revealed the importance of finding the appropriate association of concepts to understand the underlying phenomena. As future work, we seek to develop algorithms and heuristics that can take advantage of the concept lattice to highlight systematically all relevant attack phenomena found at different dimensional levels. The objective is to further improve the concept-based root cause analysis, and, more importantly, to facilitate the work of the security analyst.

References

[1] P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. Freiling. The nepenthes platform: An efficient approach to collect malware. In *Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2006.

[2] D. Barbarà, J. Couto, S. Jajodia, L. Popyack, and N. Wu. Adam: A testbed for exploring the use of data mining in intrusion detection. In *ACM SIGMOD Record, 30(4)*, pages 15–24, 2001.

[3] D. Barbará and S. Jajodia, editors. *Applications of Data Mining in Computer Security*, volume 6 of *Advances in Information Security*, chapter Data Mining For Intrusion Detection - A Critical Review. Springer, 2002.

[4] I. Bomze, M. Budinich, P. Pardalos, and M. Pelillo. The maximum clique problem. In *Handbook of Combinatorial Optimization*, volume 4. Kluwer Academic Publishers, Boston, MA, 1999.

[5] Finjan Malicious Code Research Center. Web security trends report q1/2008, http://www.finjan.com/content.aspx?id=827, sep 2008.

[6] Z. Chen, L. Gao, and K. Kwiat. Modeling the spread of active worms. In *Proceedings of IEEE INFOCOM*, 2003.

[7] M. P. Collins, T. J. Shimeall, S. Faber, J. Janies, R. Weaver, M. De Shon, and J. Kadane. Using uncleanliness to predict future botnet addresses. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 93–104, New York, NY, USA, 2007. ACM.

[8] SRI International Cyber-TA Honeynet Project. www.cyber-ta.org/honeynet, [july 2008].

[9] D. Dagon, C. Zou, and W. Lee. Modeling botnet propagation using time zones. In *Proceedings of the 13th Annual Network and Distributed System Security Symposium (NDSS'06)*, February 2006.

[10] H. Debar and A. Wespi. Aggregation and correlation of intrusion-detection alerts. In *RAID '00: Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection*, pages 85–103, London, UK, 2001. Springer-Verlag.

[11] DShield. http://www.dshield.org, sep 2008.

[12] J. Franklin, V. Paxson, A. Perrig, and S. Savage. An inquiry into the nature and causes of the wealth of internet miscreants. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 375–388, New York, NY, USA, 2007. ACM.

[13] B. Ganter, G. Stumme, and R. Wille. *Formal Concept Analysis: Foundations and Applications*. Lecture Notes in Artificial Intelligence, no. 3626, Springer-Verlag, 2005.

[14] G. Gu, R. Perdisci, J. Zhang, and W. Lee. BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *Proceedings of the 17th USENIX Security Symposium*, 2008.

[15] E-H. Han, G. Karypis, V. Kumar, , and B. Mobasher. Clustering in a high-dimensional space using hypergraph models. Technical report, Department of Computer Science, University of Minnesota, 1997.

[16] Y. Huang, H. Xiong, W. Wu, and Z. Zhang. A hybrid approach for mining maximal hyperclique patterns. *IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, 0:354–361, 2004.

[17] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall advanced reference series, 1988.

[18] K. Julisch and M. Dacier. Mining intrusion detection alarms for actionable knowledge. In *Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining*, 2002.

[19] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics 22: 79-86.*, 1951.

[20] S. Kuznetsov and S. Obiedkov. Algorithms for the construction of concept lattices and their diagram graphs. In *PKDD '01: Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*, pages 289–300, London, UK, 2001. Springer-Verlag.

[21] W. Lee and S. J. Stolfo. Combining knowledge discovery and knowledge engineering to build IDSs. In *RAID '99: Proceedings of the 3th International Symposium on Recent Advances in Intrusion Detection*, 1999.

[22] W. Lee, S.J. Stolfo, and K.W. Mok. A data mining framework for building intrusion detection models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, pages 120–132, 1999.

[23] C. Leita and M. Dacier. Sgnet: a worldwide deployable framework to support the analysis of malware threat models. In *Proceedings of the 7th European Dependable Computing Conference (EDCC 2008)*, 2008.

[24] C. Leita, K. Mermoud, and M. Dacier. Scriptgen: an automated script generation tool for honeyd. In *Proceedings of the 21st Annual Computer Security Applications Conference*, 2005.

[25] C. Leita, V.H. Pham, O. Thonnard, E. Ramirez-Silva, F. Pouget, E. Kirda, and Dacier M. The Leurre.com Project: Collecting Internet Threats Information Using a Worldwide Distributed Honeynet. In *Proceedings of the WOMBAT Workshop on Information Security Threats Data Collection and Sharing, WISTDCS 2008*. IEEE Computer Society press, April 2008.

[26] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of 8th ACM SIGMOD workshop on Research Issues in data mining and knowledge discovery, California, USA*, 2003.

[27] D. Moore, C. Shannon, G.M. Voelker, and S. Savage. Network telescopes: Technical report. *CAIDA, April*, 2004.

[28] M. Pavan and M. Pelillo. A new graph-theoretic approach to clustering and segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[29] V. Pham, M. Dacier, G. Urvoy Keller, and T. En Najjary. The quest for multi-headed worms. In *DIMVA 2008, 5th Conference on Detection of Intrusions and Malware & Vulnerability Assessment, July, 2008, Paris, France*, Jul 2008.

[30] F. Pouget. *Distributed system of honeypot sensors : Discrimination and correlative analysis of attack processes*. PhD thesis, Ecole Nationale Supérieure des Télécommunications (ENST), Paris., 2006.

[31] F. Pouget and M. Dacier. Honeypot-based forensics. In *AusCERT2004, AusCERT Asia Pacific Information technology Security Conference 2004, 23rd - 27th May 2004, Brisbane, Australia*, 2004.

[32] F. Pouget, M. Dacier, J. Zimmerman, A. Clark, and G. Mohay. Internet attack knowledge discovery via clusters and cliques of attack traces. *Journal of Information Assurance and Security, Volume 1, Issue 1, March*, 2006.

[33] Team Cymru Darknet Project. http://www.cymru.com/darknet/, [july 2008].

[34] The Honeynet Project. http://www.honeynet.org, [july 2008].

[35] The Leurre.com Project. http://www.leurrecom.org, [july 2008].

[36] N. Provos. A virtual honeypot framework. In *Proceedings of the 13th USENIX Security Symposium*, 2004.

[37] M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A multifaceted approach to understanding the botnet phenomenon. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 41–52, New York, NY, USA, 2006. ACM.

[38] J. Riordan, D. Zamboni, and Y. Duponchel. Building and deploying billy goat, a worm-detection system. In *Proceedings of the 18th Annual FIRST Conference*, 2006.

[39] Internet Motion Sensor. http://ims.eecs.umich.edu/, [july 2008].

[40] L. Spitzner. *Honeypots: Tracking Hackers*. Addison-Wesley, 2002.

[41] O. Thonnard and M. Dacier. A framework for attack patterns' discovery in honeynet data. *Journal of Digital Investigation*, 5S:S128–S139, 2008.

[42] T. Werner. Honeytrap. http://honeytrap.mwcollect.org/, [july 2008].

[43] H. Xiong, Pang-Ning Tan, and V. Kumar. Hyperclique pattern discovery. *Data Mining and Knowledge Discovery Journal*, 13(2):219–242, 2006.

[44] V Yegneswaran, P Barford, and V Paxson. Using honeynets for internet situational awareness. In *Fourth ACM Sigcomm Workshop on Hot Topics in Networking (Hotnets IV)*, 2005.

Addressing the Attack Attribution Problem using Knowledge Discovery and Multi-criteria Fuzzy Decision-Making

Olivier Thonnard
Royal Military Academy
Polytechnic Faculty
Brussels, Belgium
olivier.thonnard@rma.ac.be

Wim Mees
Royal Military Academy
Polytechnic Faculty
Brussels, Belgium
wim.mees@rma.ac.be

Marc Dacier
Symantec Research
Sophia Antipolis
France
marc_dacier@symantec.com

ABSTRACT

In network traffic monitoring, and more particularly in the realm of threat intelligence, the problem of "attack attribution" refers to the process of effectively attributing new attack events to (un)-known phenomena, based on some evidence or traces left on one or several monitoring platforms. Real-world attack phenomena are often largely distributed on the Internet, or can sometimes evolve quite rapidly. This makes them inherently complex and thus difficult to analyze. In general, an analyst must consider many different attack features (or criteria) in order to decide about the plausible root cause of a given attack, or to attribute it to some given phenomenon. In this paper, we introduce a global analysis method to address this problem in a systematic way. Our approach is based on a novel combination of a knowledge discovery technique with a fuzzy inference system, which somehow mimics the reasoning of an expert by implementing a multi-criteria decision-making process built on top of the previously extracted knowledge. By applying this method on attack traces, we are able to identify large-scale attack phenomena with a high degree of confidence. In most cases, the observed phenomena can be attributed to so-called *zombie armies* - or botnets, i.e. groups of compromised machines controlled remotely by a same entity. By means of experiments with real-world attack traces, we show how this method can effectively help us to perform a behavioral analysis of those zombie armies from a long-term, strategic viewpoint.

Keywords

Intelligence monitoring and analysis, attack attribution.

1. INTRODUCTION

In the field of threat intelligence, "attack attribution" refers to the process of effectively attributing new attack events to known or unknown phenomena by analyzing the traces they have left on sensors or monitoring platforms deployed on the Internet. The objectives of such a process are twofold: *i)* to get a better understanding of the root causes of the observed attacks; and *ii)* to characterize emerging threats from a global viewpoint by producing a precise analysis of the modus operandi of the attackers on a longer time scale.

In this paper, we introduce a global threat analysis method to address this problem in a systematic way. We present a knowledge mining framework that enables us to identify and characterize large-scale attack phenomena on the Internet, based on network traces collected with very simple and easily deployable sensors. Our approach relies on a novel combination of knowledge discovery (by means of maximum cliques) and a multi-criteria decision-making algorithm that is based on a fuzzy inference system (FIS). Interestingly, a FIS does not need any training prior making inferences. Instead, it takes advantage of the previously extracted knowledge to make sound inferences, so as to attribute incoming attack events to a given phenomenon.

A key aspect of the proposed method is the exploitation of external characteristics of malicious sources, such as their spatial distributions in terms of countries and IP subnets, or the distribution of targeted sensors. We take advantage of these statistical characteristics to group events that seem a priori unrelated, whereas most current techniques used for anomalous traffic correlation rely only on the intrinsic properties of network flows (e.g., protocol characteristics, IDS alerts or signatures, firewall logs, etc) [1, 31].

Our research builds also on prior work in malicious traffic analysis, also referred to as Internet *background radiation* [17, 4]. We acknowledge also the seminal work of Yegneswaran et al. on "Internet situational awareness" [30], in which they explore ways to integrate honeypot data into daily network security monitoring. Their approach aims at providing tactical information, for daily operations, whereas our approach is more focused on strategic information revealing the long-term behaviors of large-scale phenomena. Furthermore, many of these large-scale phenomena are apparently related to the ubiquitous problem of *zombie armies* - or botnets, i.e. groups of compromised machines that are remotely controlled and coordinated by a same entity. Still today, zombie armies and botnets constitute, admittedly, one of the main threats on the Internet, and they are used for different kinds of illegal activities (e.g., bulk spam sending, online fraud, denial of service attack, etc) [3, 18]. While most previous studies related to botnets have focused on un-

CSI-KDD '09, June 28, Paris, France

derstanding their inner working [23, 6, 2], or on techniques for detecting bots at the network-level [8, 9], we are instead more interested in studying the global behaviors of those armies from a strategic viewpoint, i.e.: how long do they stay alive on the Internet, what is their average size, and more importantly, how do they evolve over time with respect to different criteria such as their origins, or the type of activities (or scanning) they perform.

In Section 2, we present the first component of our method, namely the extraction of cliques of attackers. This step aims at discovering knowledge by identifying meaningful correlations in a set of attack events. In Section 3, we present a multi-criteria decision-making algorithm that is based on a fuzzy inference system. The purpose of this second component consists in combining intelligently the previously extracted knowledge, so as to build sequences of attack events that can be very likely attributed to the same global phenomena. Then, in Section 4, we present our experimental results and the kind of findings we can obtain by applying this analysis method to a set of attack events. Finally, we conclude in Section 5 and we suggest some future directions.

2. KNOWLEDGE DISCOVERY IN ATTACK TRACES

2.1 Introduction

We need first to introduce the notion of "attack event". Our dataset is made of network attack traces collected from a distributed set of sensors (e.g., server honeypots), which are deployed in the context of the *Leurre.com Project* [14, 22]. Since honeypots are systems deployed for the sole purpose of being probed or compromised, any network connection that they establish with a remote IP can be considered as malicious, or at least suspicious. We use a classical clustering algorithm to perform a first low-level classification of the traffic. Hence, each IP source observed on a sensor is attributed to a so-called *attack cluster* [21] according to its network characteristics, such as the number of IP addresses targeted on the sensor, the number of packets and bytes sent to each IP, the attack duration, the average inter-arrival time between packets, the associated port sequence being probed, and the packet payload (when available). Therefore, all IP sources belonging to a given attack cluster have left very similar network traces on a given sensor and consequently, they can be considered as having the same *attack profile*. This leads us then to the concept of attack event, which is defined as follows:

> An *attack event* refers to a subset of IP sources having the same attack profile on a given sensor, and whose coordinated activity has been observed within a specific time window.

Fig. 1 illustrates this notion by representing the time series (i.e., the number of sources per day) of three coordinated attack events observed on two different sensors in the same time interval, and targeting three different ports. The identification of those events can be easily automated by using the method presented in [20]. By doing so, we are able to extract interesting events from this spurious, nonproductive traffic collected by our sensors (previously termed "Internet background radiation" in [17]), and we can focus on the most important events that might originate from coordinated phenomena. In the rest of this Section, we show how to take advantage of different characteristics of such attack events to discover knowledge by means of an unsupervised clique-based clustering technique.

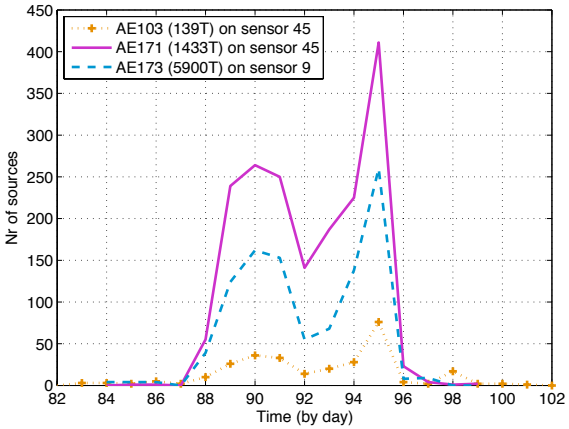


Figure 1: Illustration of 3 attack events observed on 2 different sensors, and targeting 3 different ports.

2.2 Defining Attack Characteristics

In most knowledge discovery applications, the first step consists in selecting certain key characteristics from the dataset, i.e., salient features that may (hopefully) provide meaningful *patterns* [11]. We give here an overview of different attack characteristics we have selected to perform the extraction of knowledge from our set of attack events. In this specific case, we consider these characteristics as useful to analyze the root causes of global phenomena observed on our sensors. However, we do not pretend that they are the only ones that could be used in threat monitoring, and other characteristics might certainly prove even more relevant in the future. For this reason, the framework is built such that other attack features could be easily included when necessary.

So, the two first characteristics retained are related to the *origins* of the attackers, i.e. their spatial distributions. First, the geographical location can be used to identify attack activities having a specific distribution of originating countries. Such information can be important to identify, for instance, botnets that are located in a limited number of countries. It is also a way to confirm the existence, or not, of so-called safe harbors for cybercriminals or hackers. Somehow related to the geographical location, the IP network blocks provide also an interesting viewpoint on the attack phenomena. Indeed, IP subnets can give a good indication of the spatial "uncleanliness" of certain networks, i.e., the tendency for compromised hosts (e.g., zombie machines) to stay clustered within unclean networks [5]. So, for each attack event, we can create a feature vector representing either the distribution of originating countries, or of IP addresses grouped by Class A-subnet (i.e., by /8 prefix).

The next attack characteristic deals with the *targets* of the attackers, namely the distribution of sensors that have been targeted by the sources. Botmasters may indeed send commands at a given time to all zombies to instruct them to start

scanning (or attacking) one or several IP subnets, which of course will create coordinated attack events on specific sensors. Therefore, it seems important to look at relationships that may exist between attack events and the sensors they have been observed on. Since attack events are defined per sensor, we decided to group all strongly correlated attack events that occurred within the same time window of existence (as explained in [20]), and we then use each group of attack events to create the feature vector representing the proportion of sensors that have been targeted.

Besides the origins and the targets, the type of activity performed by the attackers seems also relevant to us. In fact, bot software is often crafted with a certain number of available exploits targeting a reduced set of TCP or UDP ports. In other words, we might think of each botnet having its own *attack capability*, which means that a botmaster will normally issue scan or attack commands only for vulnerabilities that he might exploit to expand his botnet. So, it seems to make sense to take advantage of this feature to look for similarities between the sequences of ports that have been targeted by the sources of the attack events. Let us remind that, in our low-level classification of the network traffic [21], each source is associated to the complete *sequence of ports* that it has targeted on a given sensor for the whole duration of the attack session (e.g., less than 24 hours), which allows us to compute and compare the distributions of port sequences for the observed attack events.

Finally, we have also decided to compute, for each pair of events, the ratio of common IP addresses. We are aware of the fact that, as time passes, some zombie machines of a given botnet might be cured while others may get infected and join the botnet. Additionally, certain ISPs apply a quite dynamic policy of IP address allocation to residential users, which means that bot-infected machines can have different IP addresses when we observe them at different moments. Nevertheless, and according to our domain experience, it is reasonable to expect that if two distinct attack events have a high percentage of IP addresses in common, then the probability that those two events are somehow related to the same global phenomenon is increased (assuming that the time difference between the two events is not too large).

2.3 Extracting Cliques of Attackers

2.3.1 Principles

In our global threat analysis method, we have developed a knowledge discovery component that involves an unsupervised graph-theoretic correlation process. The idea consists in discovering all groups of highly similar attack events (through their corresponding feature vectors) in a reliable and consistent manner, and for each attack characteristic that can bring an interesting viewpoint on the root causes.

In a clustering task, we typically consider the following steps [11]: *i)* feature selection and/or extraction; *ii)* definition of a similarity measure between pairs of patterns; *iii)* grouping similar patterns; *iv)* data abstraction (if needed), to provide a compact representation of each cluster; and *v)* the assessment of the clusters quality and coherence.

In the previous Section, we have already described the attack features that are of interest in this paper; so now we need to measure the similarity between two such input vectors (or distributions, in our case). Clearly, the choice of a similarity metric is very important, as it has an impact on the properties of the final clusters, such as their size, quality, and consistency. To reliably compare the kind of empirical distributions mentioned here above, we have chosen to rely on strong statistical distances. As we do not know the real underlying distribution from which the observed samples were drawn, we use non-parametric statistical tests, such as Pearson's χ^2, to determine whether two one-dimensional probability distributions differ in a significant way (with a significance level of 0.05). The resulting *p-value* is then validated against the Jensen-Shannon divergence (JSD) [15], which derives itself from the Kullback-Leibler divergence [12]. Let p_1 and p_2 be for instance two probability distributions over a discrete space X, then the K-L divergence of p_2 from p_1 is defined as:

$$D_{KL}(p_1||p_2) = \sum_x p_1(x) \log \frac{p_1(x)}{p_2(x)}$$

which is also called the information divergence (or *relative entropy*). D_{KL} is commonly used in information theory to measure the difference between two probability distributions p_1 and p_2, but it is not considered as a *true* metric since it is not symmetric, and does not satisfy the triangle inequality. For this reason, we can also define the Jensen-Shannon divergence as:

$$JS(p_1, p_2) = \frac{D_{KL}(p_1||\bar{p}) + D_{KL}(p_2||\bar{p})}{2}$$

where $\bar{p} = (p_1 + p_2)/2$. In other words, the Jensen-Shannon divergence is the *average* of the KL-divergences to the *average distribution*. The JSD has the following notable properties: it is always bounded and non-negative; $JS(p_1, p_2) = JS(p_2, p_1)$ (symmetric), and $JS(p_1, p_2) = 0$ when $p_1 = p_2$ (idempotent). To be a true metric, the JSD must also satisfy the triangular inequality, which is not true for all cases of (p_1, p_2). Nevertheless, it can be demonstrated that the *square root* of the Jensen-Shannon divergence is a true metric [7], which is what we need for our application.

Finally, we take advantage of those similarity measures to group all attack events whose distributions look very similar. We simply use an unsupervised graph-based approach to formulate the problem: the vertices of the graph represent the patterns (or feature vectors) of all attack events, and the edges express the similarity relationships between those vertices, as calculated with the distance metrics described here above. Then, the clustering is performed by extracting so-called *maximal cliques* from the graph, where a maximal clique is defined as an induced sub-graph in which the vertices are fully connected and it is not contained within any other clique. To perform this unsupervised clustering, we use the *dominant sets* approach of Pavan et al. [19], which proved to be an effective method for finding maximal *weighted* cliques. This means that the weight of every edge (i.e., the relative similarity) is also taken into consideration by the algorithm, as it seeks to discover maximal cliques whose total weight is maximized. This generalization of the MCP is also known as the maximum weight clique problem (MWCP). We refer the interested reader to [27, 26] for a more detailed description of this clique-based clustering technique applied to our honeynet traces.

2.3.2 Some Experimental Clique Results

Our data set comes from a 640-day attack trace obtained

with the *Leurre.com* honeynet in the time period from September 2006 to June 2008. This trace was collected by 36 platforms located in 20 different countries and belonging to 18 different class A-subnets. We have selected only the most prevalent types of activities observed on the sensors, i.e. about 130 distinct attack profiles for which an activity involving a sufficient number of IP sources had been observed at least once on a given day during the whole period. This data set comprises totally 1,195,254 distinct sources, which have sent about 3,423,577 packets to the sensors. By using the technique described in [20], we have extracted 351 attack events that were somehow coordinated on at least two different sensors. This reduced set of attack events still accounts for 282,363 unique sources (23.6 % of the data set), or 741,349 packets (21.5%).

For the set of attack characteristics considered above, we applied our clique-based clustering on those attack events. Table 1 on page 5 presents a high-level overview of the cliques obtained for each attack dimension separately. As we can see, a relatively high volume of sources could be classified into cliques for each dimension. The last colon with the most prevalent patterns gives an indication of which countries or class A-subnets (e.g., originating or targeted IP subnets) are most commonly observed in the cliques that lie in the upper quartile with respect to the number of sources. Interestingly, it seems that many coordinated attack events are coming from a given IP subspace. Regarding the targeted platforms, several cliques involve a single class A-subnet. About the type of activities, we can observe some commonly targeted ports (e.g., Windows ports used for SMB or RPC, or SQL and VNC ports), but also a large number of uncommon high TCP ports that are normally unused on standard (and clean) machines (such as 6769T, 50286T, 9661T, . . .). A non-negligeable volume of sources is also due to UDP spammers targeting Windows Messenger popup service (ports 1026 to 1028/UDP).

2.4 Consolidation of the Knowledge

In order to assess the consistency of the resulting cliques of attack events, it can be useful to see them charted on a two-dimensional map so as to *i)* verify the proximities among clique members (*intra-clique* consistency), and *ii)* understand potential relationships between *different* cliques that are somehow related (i.e. *inter-clique* relationships). Moreover, the statistical distances used to compute those cliques make them intrinsically coherent, which means also that certain cliques of events may be somehow related to each other, although they were separated by the clique algorithm.

Since most of the feature vectors we are dealing with have a high number of variables (e.g., a geographical vector has more than 200 country variables), obviously the structure of such high-dimensional data set cannot be displayed directly on a 2D map. Multidimensional scaling (MDS) is a set of methods that can help to address this problem. MDS is based on dimensionality reduction techniques, which aim at converting a high-dimensional dataset into a two or three-dimensional representation that can be displayed, for example, in a scatter plot. The aim of dimensionality reduction is to preserve as much of the significant structure of the high-dimensional data as possible in the low-dimensional map. As a consequence, MDS allows an analyst to visualize how far observations are from each other for different kinds of similarity measures, which in turn can deliver insights into

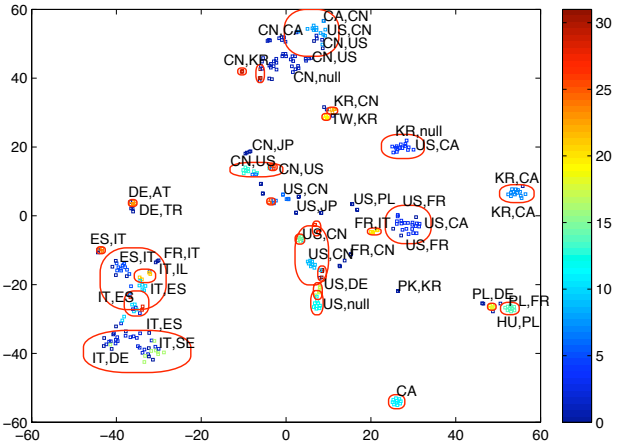


Figure 2: Visualization of geographical cliques of attackers. The coloring refers to the different cliques and the red circles indicate their sizes on the low-D map. The superposed text labels indicate *only the two top attacking countries* for some of the data points.

the underlying structure of the high-dimensional dataset.

Because of the intrinsic non-linearity of real-world data sets, we applied a recent MDS technique called *t-SNE* to visualize each dimension of the data set, and to assess the consistency of the cliques results. t-SNE [28] is a variation of *Stochastic Neighbour Embedding*; it produces significantly better visualizations than other MDS techniques by reducing the tendency to crowd points together in the centre of the map. Moreover, this technique has proven to perform better in retaining both the local and global structure of real, high-dimensional datasets in a single map, in comparison to other non-linear dimensionality reduction techniques such as Sammon mapping, Isomaps or Laplacian Eigenmaps [10]. Stochastic Neighbor Embedding aims at minimizing a cost function that is based on the sum of Kullback-Leibler divergences over all datapoints using a gradient descent method. t-SNE improves further this technique by using an initial Student-t distribution, rather than a Gaussian, to compute the similarity between two points in the low-dimensional space (which tends to alleviate the problem of "crowding" points in the center of the map, see [28] for a detailed explanation).

Figure 2 shows the resulting two-dimensional plot obtained by mapping the geographical vectors on a 2D map using t-SNE. Each datapoint on this map represents the geographical distribution of a given attack event. The coloring refers to the clique membership of each event, as obtained previously by applying the clique-based clustering, and the dotted circles indicate the clique sizes. We could easily verify that two adjacent events on the map have highly similar geographical distributions (even from a statistical viewpoint), while two distant events have clearly nothing in common in terms of originating countries. Quite surprisingly, the resulting mapping is far from being chaotic; it presents a relatively sparse structure with clear datapoint groupings, which means also that most of those attack events present very tight relationships regarding their origins. Due to the

Attack Dimension	Nr of Cliques	Max.size (nr events)	Min.size (nr events)	Volume of sources (%)	Most prevalent patterns found in the cliques[(1)]
Geolocation	31	40	3	84.4	⟨CN,CA,US,FR,TW⟩, ⟨IT,ES,FR,SE,DE,IL⟩, ⟨KR,US,BR,PL,CN,CA⟩ ⟨US,JP,GB,DE,CA,FR,CN,KR⟩, ⟨US,FR,JP,CN,DE,ES,TW⟩, ⟨CA,CN⟩ ⟨PL,DE,ES,HU,FR⟩
IP Subnets (Class A)	25	51	3	91.2	⟨87,82,151,83,84,81,85,213⟩, ⟨222,221,60,218,58,24,124,121,219,82,220⟩ ⟨201,83,200,24,211,218,89,124,61,82,84⟩, ⟨24,60⟩ ⟨83,84,85,80,88⟩, ⟨193,195,201,202,203,216,200,61,24,84,59⟩
Targeted platforms	17	86	2	70.1	⟨202⟩, ⟨88, 192⟩, ⟨195⟩, ⟨193⟩, ⟨194⟩ ⟨129, 134, 139, 150⟩, ⟨24, 213⟩
Port sequences	22	66	4	93.2	⟨I⟩, ⟨1433T⟩, ⟨I-445T⟩, ⟨5900T⟩, ⟨1026U⟩, ⟨135T⟩, ⟨50286T⟩ ⟨I-445T-139T-445T-139T-445T⟩, ⟨6769T⟩, ⟨1028U-1027U-1026U⟩

Table 1: Some experimental clique results obtained from a honeynet dataset collected from Sep 06 until June 08. (1) the given patterns represent the average distributions for the most prevalent cliques, i.e. the ones lying in the upper quartile in terms of number of sources. For the IP subnets (resp. targeted platforms), the numbers refer to the distributions of originating (resp. targeted) class A-subnets.

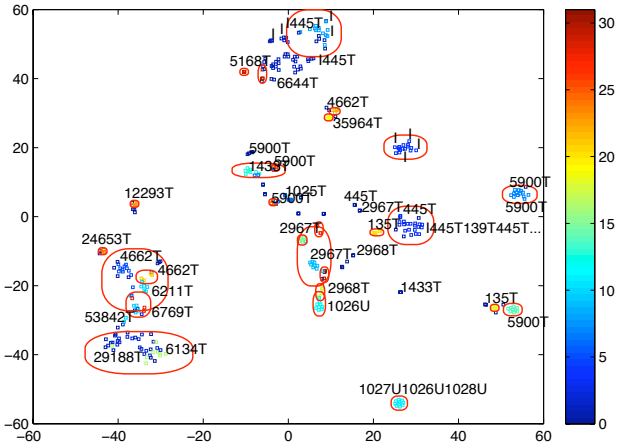


Figure 3: Same visualization of the geographical cliques of attackers as Fig 2, but here the superposed text labels indicate the port sequences targeted by the attackers.

strict statistical distances used to calculate cliques, this kind of correlation can hardly be obtained by chance only.

Similar "semantic mapping" can naturally be obtained for the other dimensions (e.g., subnets, platforms, etc), so as to help assessing the quality of other cliques of attackers. To conclude this Section, Figure 3 shows the same geographical mapping on which the port sequences of several attack events have been superposed on top of the datapoints. This can help to visualize unobvious relationships among different types of activities and their origins, and it leads also to the natural intuition that an intelligent algorithm could potentially leverage the results of this knowledge discovery process, by combining efficiently different sets of cliques.

3. MULTI-CRITERIA DECISION-MAKING

3.1 Requirements and Motivation

The decision-support component of our method shall take advantage of the knowledge obtained via the extraction of cliques, and of the global semantic mappings obtained through dimensionality reduction. The final objective consists in re-constructing *sequences of attack events* that can be attributed with a high confidence to the same root phenomenon in function of multiple criteria. In other words, we want to build an inference engine that takes as input the extracted knowledge to classify incoming attack events into either "known phenomena", or otherwise to identify a new phenomenon when needed (e.g., when we observe the first attack event of a new zombie army). There exists certainly many different classification algorithms that are able to map multiple input features to multiple output classes, even for complex, non-linear mappings, such as Support Vector Machines, Artificial Neural Networks, etc. However, we are confronted to specific constraints that do not allow us to use this type of supervised machine learning techniques. First, we have *a priori* zero-knowledge of the expected output, which means that we can not provide training samples showing the characteristics of the output we are looking for. Secondly, we want to include some domain knowledge to specify which type of combinations we expect to be promising in the root cause identification. Third, the inference system must be flexible enough to allow additional criteria to be used in the future, so as to further improve the inference capabilities. Finally, we favor the "white-box" approach having a transparent reasoning process, which allows an expert to understand the reasons (i.e., the combinations of criteria) for which the system has grouped a given set of events into the same root phenomenon.

Although large-scale phenomena on the Internet are complex and dynamic, our intuition is that two consecutive attack events should be linked to the same root phenomenon if and only if they share at least two different attack characteristics. That is, we want to build a decision-making process that will attribute two attack events to the same phenomenon when the events features are "close enough" for any combination of at least two attack dimensions out of the complete set of criteria: $\{origins, targets, activity, common_{IP}\}$. So, we hypothesize that real-world phenomena may perfectly evolve over time, which means that two consecutive attack events of the same zombie army must not necessarily have all their attributes in common. For example, the bots composition of a zombie army may evolve over time because of the cleaning of infected machines and the recruitment of new bots. From our observation viewpoint, this will translate into a certain shift in the IP subnet distribution of the zombie machines for subsequent attack events of this army

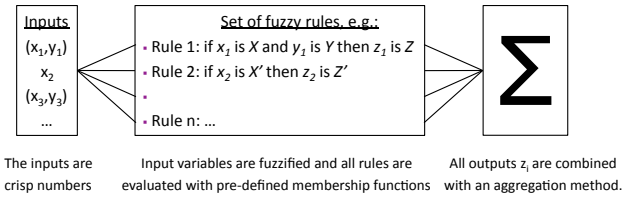


Figure 4: Main components of a Fuzzy System.

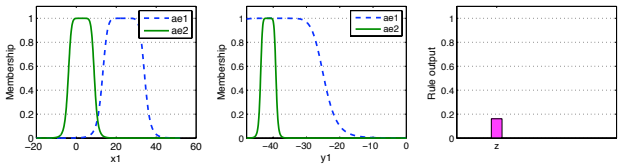


Figure 5: Fuzzy rule evaluation.

(and thus, most probably different cliques w.r.t. the origins). Or, a zombie army may be instructed to scan several consecutive IP subnets in a rather short interval of time, which will lead to the observation of different events having highly similar distributions of originating countries and subnets, but those events will target completely different sensors, and may eventually use different exploits (hence, targeting different port sequences).

On the other hand, we consider that only one correlated attack dimension is not sufficient to link two attack events to the same root cause, since the result might then be due to chance only (e.g., a large proportion of attacks originate from some large or popular countries, certain Windows ports are commonly targeted, etc). However, by combining intelligently several attack viewpoints, we can reduce considerably the probability that two attack events would be attributed to the same root cause whereas they are in fact unrelated.

3.2 Fuzzy Inference Systems

We still need to formally define what is the "relatedness degree" between two attack events, certainly when they do not belong to a same clique but are somehow "close" to each other. Intuitively, attack events characteristics in the real world have unsharp boundaries, and the membership to a given phenomenon can be a matter of degree. For this reason, we have developed a decision-making process that is based on a fuzzy inference system (FIS). The mathematical concepts behind fuzzy reasoning are quite simple and intuitive; in fact, it aims at reproducing the reasoning of a human expert with very simple mathematical functions. Fuzzy inference is thus a convenient way to map an input space to an output space with a flexible and extensible system, and using the codification of common sense and expert knowledge. The mapping then provides a basis from which decisions can be made.

The main components of an inference system are sketched in Fig. 4. To map the input space to the output space, the primary mechanism is a list of if-then statements called rules, which are evaluated in parallel, so the order of the rules is unimportant. Instead of using crisp variables, all inputs are *fuzzified* using membership functions in order to determine the degree to which the input variables belong to each of the appropriate fuzzy sets. If the antecedent of a given rule has more than one part (i.e., multiple 'if' statements), a fuzzy logical operator is applied to obtain one number that represents the result of the antecedent for that rule. For example, the fuzzy OR operator simply selects the maximum of the two values. The results of all rules are then combined and distilled into a single, crisp value that can be used to make a decision. This aggregation process can be done in two different ways. Mamdani's inference [16] expects the output membership functions to be also fuzzy sets. After the aggregation process, there is a fuzzy set for each output variable that needs defuzzification by computing for instance the centroïd of the output function. Whereas in a Sugeno-type inference system [25], the output membership functions are either linear or constant. The general form of a rule in a Sugeno fuzzy model is: if $Input_1$ is x and $Input_2$ is y then Output is $z = a.x + b.y + c$. For a zero-order Sugeno model, the output level z is a constant (a=b=0). The output level z_i of each rule is weighted by the firing strength w_i of the rule. The most common way to calculate the final output of the system is the weighted average of all rule outputs:

$$Final\ output = \frac{\sum_i w_i.z_i}{\sum_i w_i}$$

When it is possible to model a fuzzy system using Sugeno-type inference, the defuzzification and aggregation process is thus greatly simplified and much more efficient than with Mamdani's inferences, which is why we used a Sugeno-type system to model each attack phenomenon.

Concretely, we use the knowledge obtained from the extraction of cliques to build the fuzzy rules that describe the behavior of each phenomenon. The characteristics of new incoming attack events are then used as input to the fuzzy systems that model the phenomena identified so far. In each of those fuzzy systems, the features of the *most recent* attack event shall define the current parameters of the membership function used to evaluate the following simple rules: if x_i is *close* AND if y_i is *close* then z_i is *related*, $\forall i \in \{geo, subnets, targets, portsequence\}$. Fig 5 gives a graphical representation of how such a rule is evaluated for the subnets of origins of two given attack events. Since this characteristic is represented by a 2D mapping, we can see the result of evaluating the relative position of the events according to both dimensions (x, y). Each membership function is maximal within the cliques, then it decreases smoothly to take into account the fuzziness of real-world phenomena. In this case, the antecedents of the rule hold respectively 0.16 and 1.0, which results in an output of 0.16 (since a logical AND in fuzzy logic corresponds to the MIN operator).

So, the membership functions referred to as "is close" in the fuzzy rules are defined by the characteristics of the cliques to which the attack events belong. The calculation of the rule output $z_i \in [0, 1]$ is just the intersection between the two curves, which quantifies the inter-relationship between the cliques (and hence, between the attack events). Similarly, we can evaluate the fuzzy rules for the other dimensions considered in the inference system. For the last dimension, i.e. the common IP's, we use a static membership function whose input is the common IP ratio calculated between the two events. Fig 6 represents this static membership function, where we can see the output Z_{IP} increasing

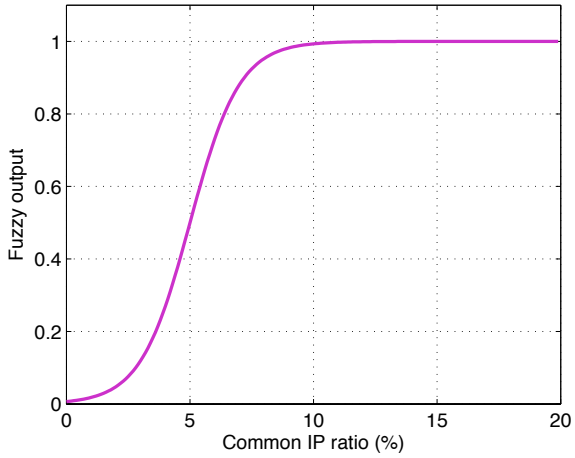


Figure 6: Common IP Membership function.

smoothly as the ratio of common IP addresses increases from 0 to 10%, where Z_{IP} is then maximal. This curve is actually drawn from our knowledge, or domain experience, in monitoring malicious traffic.

Note that, initially, the inference engine has no knowledge, so the first incoming attack event will create the first phenomenon. Then, each time a new event could not be attributed to an existing phenomenon, the inference engine will create a new fuzzy system to model this new emerging phenomenon. The inference engine is thus self-adaptive by design.

3.3 Multi-criteria Decision-making

Having formally defined how to evaluate the output of each rule, for each phenomenon, a last problem remains regarding the weighted average that is used as aggregation function in a classical Sugeno inference system. In fact, it does not allow us to express that certain *combinations of criteria* (or rule outputs) must be somehow prioritized, as previously described in the requirements. We need thus to introduce another type of multi-criteria aggregation function that allows to model more complex requirements such as "most of", or "at least two" criteria to be satisfied in the overall decision function. Yager has introduced in [29] a special type of operator called Ordered Weighted Aggregation (OWA), which allows to include some relationships between multiple criteria in the aggregation process. An OWA operator provides an aggregation function for criteria whose result lies between the classical "and" and "or" operators, which are in fact the two extreme cases. Assume $Z_1, Z_2, \ldots, Z_n$ are n criteria of concern in our multi-criteria problem. For each criteria, $Z_i(x) \in [0,1]$ indicates the degree to which x satisfies that criteria, which corresponds in our case to the rules output of a given fuzzy system. Then, we define a mapping function $F : I^n \rightarrow I$ where $I = [0,1]$ as an OWA operator of dimension n, if associated with F is a weighting vector $W = (W_1, W_2, \ldots, W_n)$ such that

1. $W_i \in [0,1]$
2. $\sum_i W_i = 1$

and where

$$F(z_1, z_2, \ldots, z_n) = W_1.z'_1 + W_2.z'_2 + \ldots + W_n.z'_n$$

with z'_i being the ith largest element in the collection $z_1, ..., z_n$. That is, Z' is an ordered vector composed of the elements of Z put in descending order, which means that the weights W_i are associated with a particular ordered position rather than a particular element. Yager [29] has carefully studied the mathematical foundations of OWA operators, and he demonstrated that such operators have the desired properties such as monotonicity, generalized commutativity, associativity and idempotence. To define the weights W_i to be used, Yager suggests two possible approaches: either to use some learning mechanism with sample data and a regression model, or to give some semantics or meaning to the W_i's by asking a decision-maker to provide directly those values. We selected the latter approach by defining the weighting vector as $W = (0.1, 0.35, 0.35, 0.1, 0.1)$, which translates our intuition about the dynamic behaviors of large-scale phenomena. It can be interpreted as: "at least three criteria must be satisfied, but the first criteria is of less importance compared to the 2nd and 3rd ones". These values were carefully chosen in order to avoid the grouping of unrelated events when, for example, two events are coming from popular countries and targeting common (Windows) ports in the same interval of time, but those events are in reality not related to the same phenomenon. In this worst-case scenario, we can imagine that the ordered vector of criteria (obtained from the evaluation of the fuzzy rules) could be something similar to $Z = (0.3, 0.1, 0, 1, 0)$. That is, we have a high correlation for the targeted port sequences ($z_4 = 1$), and we have then some weak correlation (due to chance) for the geographical origins ($z_1 = 0.3$) and also for the subnets of origins ($z_2 = 0.1$). By applying our weighting vector W to $Z' = (1, 0.3, 0.1, 0, 0)$, we get as final decision value $F = 1 * 0.1 + 0.3 * 0.35 + 0.1 * 0.35 = 0.24$. By considering other scenarios, we can verify that the values of the weighting vector W work as expected, i.e. it minimizes the final output value in these cases. Moreover, these considerations enable us also to fix our decision threshold to an empirical value of about 0.25. That is, when the final output value F lies under this threshold, we will reject the attribution of the attack event under scrutiny to the current phenomenon whose fuzzy system is being evaluated. Finally, when several fuzzy systems provide an output value lying above the threshold, we will obviously chose the highest one to attribute the event; however, this case was rarely observed in our experiments. There exists certainly other alternatives for choosing the W_i's, but according to our experimental results, this choice proved to be very effective in identifying sequences of attack events having the same root cause.

4. BEHAVIORAL ANALYSIS OF GLOBAL PHENOMENA

4.1 Main Characteristics

In this Section, we provide some experimental results obtained by applying our multi-criteria inference method to the same set of attack events we already introduced in Section 2.3 (clique analysis). As already mentioned, these experimental results only aim at validating the applicability and usefulness of the method proposed. They do not pre-

tend to offer a complete view of all possible phenomena observable on the Internet. At the contrary, they show that, even with a limited number of data sources, it is possible to observe and reason about a couple of interesting phenomena. Furthermore, these anecdotal, yet representative, examples show that our method helps in characterizing their root cause, i.e., in addressing the attack attribution issue.

So, over the whole collection period (640 days), we found about 32 global phenomena. In total, 348 attack events (99% of our data set) could be attributed to a given large-scale phenomenon. An in-depth analysis has revealed that most of those phenomena (apart from the noisy network worm W32.Rahack.H [24], also known as W32/Allaple) are quite likely related to *zombie armies*, i.e., groups of compromised machines belonging to the same botnet(s). We conjecture this for the following main reasons: *i)* the apparent coordination of the sources, both in time (i.e., coordinated events on several sensors) and in the distribution of tasks (e.g., scanners versus attackers); *ii)* the short durations of the attack events, typically a few days only, whereas "classical" worms tend to spread over longer, continuous periods of time; *iii)* the absence of known classical network worm spreading on many of the observed port sequences; and *iv)* the source growing rate, which has a sort of exponential shape for worms and is somehow different for botnets [13].

To illustrate the results, Table 2 on page 10 presents an overview of some global phenomena found in our dataset. Thanks to our method, we are able to characterize precisely the behaviors of the identified phenomena or zombie armies. Hence, we found that the largest army had in total 57 attack events comprising 69,884 sources, and could survive for about 112 days. The longest lifetime of a zombie army observed so far was still 586 days. Fig. 7 shows the cumulative distributions (CDF) of the lifetime and size of the identified armies. Those figures reveal some interesting aspects of their global behaviors: according to our observations, at least 20% of the zombie armies had in total more than ten thousand observable[1] sources during their lifetime, and the same proportion of armies could survive on the Internet for at least 250 days. On average, zombie armies have a total size of about 8,500 observed sources, a mean number of 658 sources per event, and their mean survival time is 98 days.

Regarding the origins, we observe some very persistent groups of IP subnets and countries of origin across many different armies. On Fig. 8, we can see the CDF of the sources involved in the zombie armies of Table 2, where the x-axis represents the first byte of the IPv4 address space. It appears clearly that malicious sources involved in those phenomena are highly unevenly distributed and form a relatively small number of tight clusters, which account for a significant number of sources and are thus responsible for a large deal of the observed malicious activities. This is consistent with other prior work on monitoring global malicious activities, in particular with previous studies related to measurements of Internet background radiation [4, 17, 31]. However, we are now able to show that there are still some notable differences in the spatial distributions of those zombie armies with respect to the average distribution over

[1]It is important to note that the sizes of the zombie armies given here only reflect the number of sources we could *observe* on our sensors; the actual sizes of those armies are most probably much larger, even though some churn effects (DHCP, NAT) could also affect these numbers.

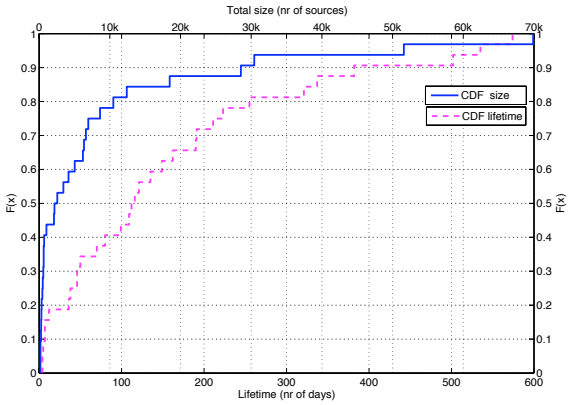


Figure 7: Empirical CDF of the size and lifetime of zombie armies.

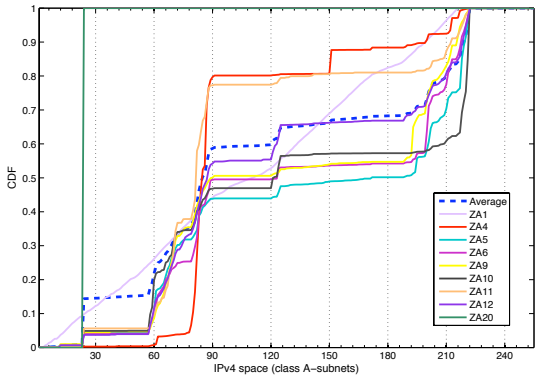


Figure 8: Empirical CDF of sources in IPv4 address space for the 9 zombie armies illustrated in Table 2.

all sources (represented with the blue dashed line). In other words, certain armies of compromised machines can have very different spatial distributions, even though there is a large overlap between "zombie-friendly" IP subnets. Moreover, because of the dynamics of this kind of phenomena, we can even observe very different spatial distributions within a *same army* at different moments of its lifetime. This is a strong advantage of our analysis method that is more precise and enables us to distinguish *individual* phenomena, instead of global trends, and to follow their dynamic behavior over time.

Another interesting observation on Fig. 8 is related to the subnet CDF of ZA1 (uniformly distributed in the IPv4 space, which means randomly chosen source addresses) and ZA20 (a constant distribution coming exclusively from the subnet 24.0.0.0/8). A very likely explanation is that those zombie armies have used spoofed addresses to send UDP spam messages to the Windows Messenger service. So, this indicates that IP spoofing is still possible under the current state of filtering policies implemented by certain ISP's on the Internet.

Finally, in terms of *attack capability*, we observe that about 50% of the armies could target at least two completely dif-

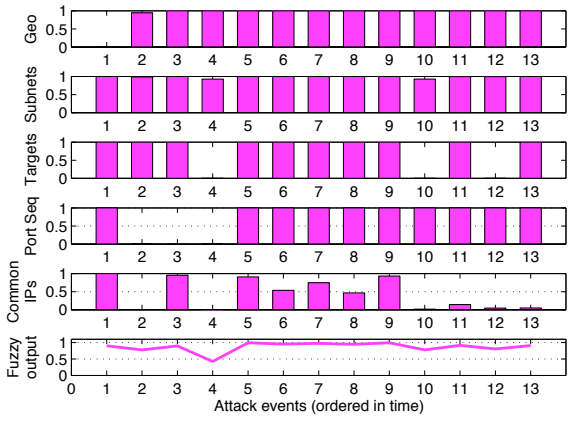


Figure 9: Output of the fuzzy inference system (z_i and $F(z_i)$) modeling the zombie army nr 12.

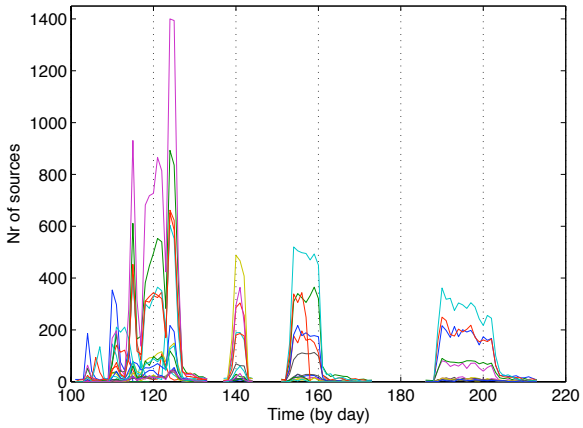


Figure 10: Time series of coordinated attack events for zombie army ZA10 (i.e., nr of sources observed by day).

ferent ports (thus, probably two different exploits, at least), and one army had even an attack capability greater than 10 (ZA4 in Table 2). At this stage, it is unclear why a zombie army would target such a large number of unusual, high TCP ports (12293T, 15264T, etc). A recurrent misconfiguration or P2P phenomenon is thus not excluded; but even in that case, it is very interesting to note that our method was able to attribute all those different events to the same root phenomenon, thanks to the combination of several statistical metrics.

4.2 Some Detailed Examples

In this Section, we further detail two zombie armies to illustrate some typical behaviors we could observe among the identified phenomena, e.g.:

i) a move (or drift) in the origins of certain armies (both geographical and IP blocks) during their lifetime;

ii) a large scan sweep by the same army targeting several consecutive class A-subnets;

iii) within a same army, multiple changes in the port sequences (or exploits) used by zombies to scan or to attack;

iv) a coordination between different armies.

Zombie army 12 (ZA12) is an interesting case in which we can observe the behaviors *ii)* and *iii)*. Fig. 9 represents the output of the fuzzy system modeling this phenomenon. Each bar graph represents the fuzzy output z_i for a given attack dimension, whereas the last plot shows the final aggregated output from which the decision to group those events together was made (i.e., $F(z_i)$). We can clearly see that the targets and the activities of this army have evolved between certain attack events (e.g., when the value of z_i is low). That is, this army has been scanning (at least) four consecutive class A-subnets during its lifetime (still 183 days), while probing at the same time three different ports on these subnetworks.

Then, the largest zombie army observed by the sensors (ZA10) has showed the behaviors *i)* and *iv)*. On Fig. 10, we can see that this army had four waves of activity during which it was randomly scanning 5 different subnets (note the almost perfect coordination among those attack events). When inspecting the subnet distributions of those different attack waves, we could clearly observe a drift in the origins of those sources, quite likely as certain machines were infected by (resp. cleaned from) the bot software. Finally, we found another smaller army (ZA11) that is clearly related to ZA10 (e.g., same temporal behavior, similar activity, same targets); but in this case, a different group of zombie machines, resulting in very different subnet CDF's on Fig. 8), was used to attack only specific IP addresses on our sensors, probably by taking advantage of the results given by the army of scanners (ZA10).

5. CONCLUSIONS

We have introduced a general analysis method to address the complex problem related to "attack attribution". Our approach is based on a novel combination of knowledge discovery and a multi-criteria fuzzy decision-making process. By applying this method, we have showed how apparently unrelated attack events could be attributed to the same global attack phenomenon, or to the same army of zombie machines operating in a coordinated manner. To the best of our knowledge, this is the first formal, systematic and rigorous method that enables us to identify and characterize precisely the behaviors of those large-scale attack phenomena. As future work, we envisage to extend our method to other data sets, such as high-interaction (eventually client) honeypot data, or malware data sets, and to include even more relevant attack features so as to improve further the inference capabilities of the system, and thus also our insights into malicious behaviors observed on the Internet.

Acknowledgments

This work has been partially supported by the European Commission through project FP7-ICT-216026-WOMBAT funded by the 7th framework program. The opinions expressed in this paper are those of the authors and do not necessarily reflect the views of the European Commission.

Id	Nr of events	Total size (nr sources)	Lifetime (nr days)	Targeted sensors (Class A- subnets)	Attack capability	Main origins (countries / subnets)
1	10	18,468	535	24.*,193.*,195.*,213.*	1026U	US,JP,GB,DE,CA,FR,CN,KR,NL,IT 69,128,195,60,81,214,211,132,87,63
4	82	26,962	321	202.*	12293T,15264T,18462T,25083T,25618T,28238T,29188T, 32878T,33018T,38009T,4152T,46030T,4662T,50286T,...	IT,ES,DE,FR,IL,SE,PL 87,82,83,84,151,85,81,88,80
5	13	9,644	131	195.*	135T,139T,1433T,2968T,5900T	CN,US,PL,IN,KR,JP,FR,MX,CA 218,61,222,83,195,221,202,24,219
6	15	51,598	>1 year	> 7 subnets	ICMP (W32.Rahack.H / Allaple)	KR,US,BR,PL,CN,CA,FR,MX,TW 201,83,200,24,211,218,89,124
9	23	11,198	218	192.*,193.*,194.*	2967T,2968T,5900T	US,CN,TW,FR,DE,CA,BR,IT,RU 193,200,24,71,70,213,216,66
10	57	69,884	112	128.*,129.*,134.*,139.*,150.*	I-I445T	CN,CA,US,FR,TW,IT,JP,DE 222,221,60,218,58,24,70,124
11	14	2,636	110	129.*,134.*,139.*,150.*	I-445T-139T-445T-139T-445T	US,FR,CA,TW,IT 82,71,24,70,68,88,87
12	14	27,442	183	192.*,193.*,194.*,195.*	1025T,1433T,2967T	US,JP,CN,FR,TR,DE,KR,GB 218,125,88,222,24,60,220,85,82
20	10	30,435	337	24.*, 129.*, 195.*	1026U,1026U1028U1027U,1027U	CA,CN 24,60

Table 2: Overview of some large-scale phenomena found in a honeynet dataset collected from Sep 06 until Jun 08.

6. REFERENCES

[1] Paul Barford and David Plonka. Characteristics of network traffic flow anomalies. In *In Proceedings of ACM SIGCOMM Internet Measurement Workshop*, 2001.

[2] Paul Barford and Vinod Yegneswaran. *An Inside Look at Botnets*. Advances in Information Security. Springer, 2006.

[3] David Barroso. Botnets - the silent threat. In *European Network and Information Security Agency (ENISA)*, November 2007.

[4] Zesheng Chen, Chuanyi Ji, and Paul Barford. Spatial-temporal characteristics of internet malicious sources. In *Proceedings of INFOCOM*, 2008.

[5] M. P. Collins, T. J. Shimeall, S. Faber, J. Janies, R. Weaver, M. De Shon, and J. Kadane. Using uncleanliness to predict future botnet addresses. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 93–104, New York, NY, USA, 2007. ACM.

[6] Evan Cooke, Farnam Jahanian, and Danny McPherson. The Zombie roundup: Understanding, detecting, and disrupting botnets. In *Proceedings of the Steps to Reducing Unwanted Traffic on the Internet (SRUTI 2005 Workshop)*, Cambridge, MA, July 2005.

[7] B. Fuglede and F. Topsoe. Jensen-shannon divergence and hilbert space embedding. pages 31–, June-2 July 2004.

[8] G. Gu, R. Perdisci, J. Zhang, and W. Lee. BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *Proceedings of the 17th USENIX Security Symposium*, 2008.

[9] Guofei Gu, Junjie Zhang, and Wenke Lee. BotSniffer: Detecting botnet command and control channels in network traffic. In *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08)*, February 2008.

[10] Geoffrey Hinton and Sam Roweis. Stochastic neighbor embedding. In *Advances in Neural Information Processing Systems 15*, volume 15, pages 833–840, 2003.

[11] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall advanced reference series, 1988.

[12] S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics 22: 79-86.*, 1951.

[13] Wenke Lee, Cliff Wang, and David Dagon, editors. *Botnet Detection: Countering the Largest Security Threat*, volume 36 of *Advances in Information Security*. Springer, 2008.

[14] C. Leita, V.H. Pham, O. Thonnard, E. Ramirez-Silva, F. Pouget, E. Kirda, and Dacier M. The Leurre.com Project: Collecting Internet Threats Information Using a Worldwide Distributed Honeynet. In *Proceedings of the WOMBAT Workshop on Information Security Threats Data Collection and Sharing, WISTDCS 2008*. IEEE Computer Society press, April 2008.

[15] J. Lin. Divergence measures based on the shannon entropy. *Information Theory, IEEE Transactions on*, 37(1):145–151, Jan 1991.

[16] E. H. Mamdani and S. Assilian. An experiment in linguistic synthesis with a fuzzy logic controller. *Int. J. Hum.-Comput. Stud.*, 51(2):135–147, 1999.

[17] Ruoming Pang, Vinod Yegneswaran, Paul Barford, Vern Paxson, and Larry Peterson. Characteristics of internet background radiation. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 27–40, New York, NY, USA, 2004. ACM.

[18] Markus Kötter Georg Wicherski Paul Bächer, Thorsten Holz. Know your enemy: Tracking botnets. In *http://www.honeynet.org/papers/bots/*.

[19] M. Pavan and M. Pelillo. A new graph-theoretic approach to clustering and segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2003.

[20] V. Pham, M. Dacier, G. Urvoy Keller, and T. En Najjary. The quest for multi-headed worms. In *DIMVA 2008, 5th Conference on Detection of Intrusions and Malware & Vulnerability Assessment, July, 2008, Paris, France*, Jul 2008.

[21] F. Pouget and M. Dacier. Honeypot-based forensics. In *AusCERT2004, AusCERT Asia Pacific Information technology Security Conference 2004, 23rd - 27th May 2004, Brisbane, Australia*, 2004.

[22] The Leurre.com Project. http://www.leurrecom.org.

[23] M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A multifaceted approach to understanding the botnet phenomenon. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 41–52, New York, NY, USA, 2006. ACM.

[24] Symantec Security Response. W32.rahack.h, [april 2009].

[25] Michio Sugeno. *Industrial Applications of Fuzzy Control*. Elsevier Science Inc., New York, NY, USA, 1985.

[26] Olivier Thonnard and Marc Dacier. A framework for attack patterns' discovery in honeynet data. *DFRWS 2008, 8th Digital Forensics Research Conference, August 11- 13, 2008, Baltimore, USA*, 2008.

[27] Olivier Thonnard and Marc Dacier. Actionable knowledge discovery for threats intelligence support using a multi-dimensional data mining methodology. In *ICDM'08, 8th IEEE International Conference on Data Mining series, December 15-19, 2008, Pisa, Italy*, Dec 2008.

[28] Laurens van der Maaten and Geoffrey Hinton. Visualizing

data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, November 2008.

[29] Ronald R. Yager. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Trans. Syst. Man Cybern.*, 18(1):183–190, 1988.

[30] V Yegneswaran, P Barford, and V Paxson. Using honeynets for internet situational awareness. In *Fourth ACM Sigcomm Workshop on Hot Topics in Networking (Hotnets IV)*, 2005.

[31] Vinod Yegneswaran, Paul Barford, and Johannes Ullrich. Internet intrusions: global characteristics and prevalence. In *SIGMETRICS*, pages 138–147, 2003.

Learning More About the Underground Economy: A Case-Study of Keyloggers and Dropzones

Thorsten Holz[1,2] Markus Engelberth[1] Felix Freiling[1]

[1] Laboratory for Dependable Distributed Systems, University of Mannheim, Germany
<holz,engelberth,freiling@informatik.uni-mannheim.de>
[2] Secure Systems Lab, Vienna University of Technology, Austria

Abstract. We study an active underground economy that trades stolen digital credentials. In particular, we investigate keylogger-based stealing of credentials via *dropzones*, anonymous collection points of illicitly collected data. Based on the collected data from more than 70 dropzones, we present an empirical study of this phenomenon, giving many first-hand details about the attacks that were observed during a seven-month period between April and October 2008. We found more than 33 GB of keylogger data, containing stolen information from more than 173,000 victims. Analyzing this data set helps us better understand the attacker's motivation and the nature and size of these emerging underground marketplaces.

1 Introduction

With the growing digital economy, it comes as no surprise that criminal activities in digital business have lead to a digital underground economy. Because it is such a fast-moving field, tracking and understanding this underground economy is extremely difficult. Martin and Thomas [20] gave a first insight into the economy of trading stolen credit card credentials over open IRC channels. The "blatant manner" in which the trading is performed with "no need to hide" [20] is in fact staggering. A large-scale study of similar forms of online activity was later performed by Franklin et al. [10]. The result of this study is that Internet-based crime is now largely profit-driven and that "the nature of this activity has expanded and evolved to a point where it exceeds the capacity of a closed group" [10]. In other words, digital and classical crime are merging.

In general, it is hard to estimate the real size of the underground economy. This is because the only observable evidence refers to *indirect* effects of underground markets. For example, both previous studies [10,20] did not observe real trading, but only *announcements* of trading and *offers* of stolen credentials in public IRC channels. It is in fact a valid question how much of the offered data really belongs to online scams—rather than being just the result of "poor scum nigerians and romanians try[ing] to make 20$ deals by ripping eachother off" [4].

In this paper, we report on measurements of the *actual kind and amount* of data that is stolen by attackers from compromised machines, i.e., we *directly* observe the goods that can be traded at an underground market. Obviously, this data gives us a much better basis for estimating the size of the underground economy and also helps to understand the attacker's motivation.

It may seem as if direct observations of illicitly traded goods are much harder to obtain than indirect ones. In this paper we show that this must not be the case. In particular, we focus on the newly emerging threat of keyloggers that communicate with the attacker through so-called *dropzones*. A dropzone is a publicly writable directory on a server in the Internet that serves as an exchange point for keylogger data. The attack is visualized in Figure 1. The attacker A first infects victims V_1, V_2 and V_3 with keylogging malware. This malware secretly collects credentials that victims use to authenticate to online services like a bank P_1 or a webmailer P_2. After collecting these credentials, the malware running on a compromised machine sends them to the dropzone, where the attacker can pick them up and start to abuse them [9,30,31,32,34].

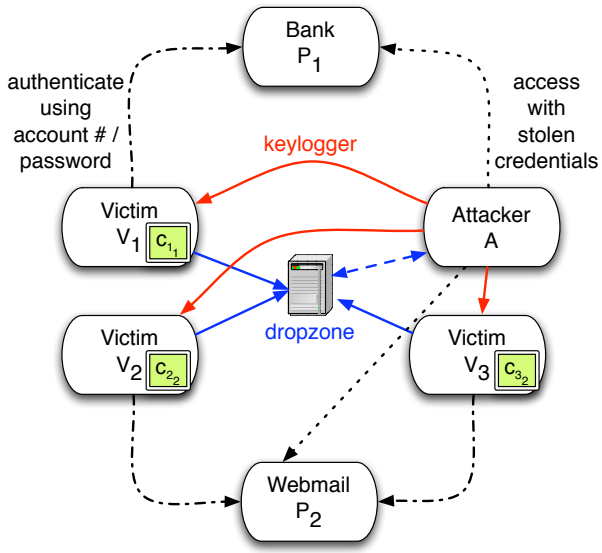


Fig. 1: Schematic overview of keylogger-based attacks using dropzones.

We analyzed these keylogger-based attacks by first collecting keyloggers with different techniques such as honeypots [27] or spamtraps, and then executing them within an instrumented environment [37], thereby extracting the location of the dropzone. By accessing the dropzone directly, we harvested the keylogger data just like the attacker would have done this. We perform our case study using two different classes of keyloggers called *Limbo/Nethell* and *ZeuS/Zbot/Wsnpoem*. We give details of attacks we observed during a seven-month period between April and October 2008. In particular, we were able to harvest a total of 33 GB of keylogger data from more than 70 unique dropzones, resulting in information about stolen credentials from more than 173,000 compromised machines. We present the results of a statistical analysis of this data. To our knowledge, this is the first time that it has been possible to perform such an analysis on *stolen* data on such a large scale. It gives rather credible answers to questions

about the type and the amount of data criminals steal, which allows us to study the underground economy since these stolen credentials are marketable goods. For example, we recovered more than 10,700 stolen online bank account credentials and over 149,000 stolen email passwords, potentially worth several million dollars on the underground market. Our analysis shows that this type of cybercrime is a profitable business, allowing an attacker to potentially earn hundreds or even thousands of dollars per day.

1.1 Related Work

Besides the related work discussed previously, this paper touches on a several related research areas. In the field of phishing prevention and mitigation, there has been some work specific to attacks based on email and fake websites [5,11]. Chandrasekaran et al. [5] generate fake input and investigate a site's response to detect phishing sites. Gajek and Sadeghi [11] use fake credentials to track down phishers. Our work is complementary to this work: we study the actual dropzone and infer from this data more information about the extent and size of the attack.

Recently Kanich et al. studied the *conversion rate* of spam, i.e., the probability that an unsolicited e-mail will ultimately elicit a *sale* [15]. This is another example of a direct observation of the underground economy and provides a different point of view into the market mechanisms behind cybercrime.

The keylogger-based attacks we study in this paper can be stopped using different kinds of techniques, for example multi-factor authentication, biometrics, or special hardware or software. While techniques like SpoofGuard [7], Dynamic Security Skins [8], or Transport Login Protocol [6] can protect against certain forms of these attacks, e.g., classical phishing attacks, they can not stop keylogger-based attacks that we study in this paper. Preventing this kind of attacks is harder since the user machine itself is compromised, which allows the malicious software to steal credentials directly as the victim performs the login procedure. Modern keyloggers also defeat simple tricks to conceal the entered password as proposed by Herley and Florêncio [12]. However, malware prevention methods and systems that protect confidential information can defend against this kind of attacks [21,35].

1.2 Summary of Contributions

To summarize, our work presented in this paper makes the following contributions: We investigate keylogging attacks based on dropzones and provide a detailed analysis of the collected data, giving a first-hand insight into the underground economy of Internet criminals from a unique and novel viewpoint. We believe that our method can be generalized to many other forms of credential-stealing attacks, such as phishing attacks.

We argue that combined with prices from the underground economy, our study gives a more precise estimate of the dangers and potential of the black market than indirect measures performed previously [10,20]. Together with these prior studies, we hope that our results help to relinquish the common mindset we often see with politicians and commercial decision-makers that we do not need to track down and prosecute these criminals because it is too costly. We feel that the sheer size of the underground economy now and in the future will not allow us to neglect it.

Paper Outline. We describe in Section 2 in more detail how keylogging based attacks work and introduce two different families of keyloggers. In Section 3, we introduce our analysis setup and present statistics for the dropzones we studied during the measurement period. We analyze the collected data in Section 4 using five different categories and briefly conclude the paper in Section 5 with an overview of future work.

Data Protection and Privacy Concerns. The nature of data analyzed during this study is very sensitive and often contains personal data of individual victims. We are not in a position to inform each victim about the security breach and therefore decided to hand over the full data set to AusCERT, Australia's National Computer Emergency Response Team. This CERT works together with different banks and other providers to inform the victims. We hope that the data collected during this study can help to recover from the incidents and more damage is prevented.

2 Background: Keylogger-based Attacks

Figure 1 provides a schematic overview of keylogger-based attacks using dropzones. Each victim V_i has a specific credential c_{i_j} to authenticate at provider P_j to use the service. For example, P_1 is an online banking website and V_1 uses his account number and a password to log in. The attacker A uses different techniques to infect each victim V_i with a keylogger. Once the victim V_i is infected, the keylogger starts to record all keystrokes: A defines in advance which keystrokes should be logged and the malware only records these. For example, A can specify that only the login process of an online banking website should be recorded. The malware then observes the values entered in input fields on the website and sends this information to a dropzone. This dropzone is the central collection site for all harvested information. The attacker can access the dropzone, extract the stolen credentials, and use them to impersonate at P_j as V_i.

2.1 Studying the Attack

The practical challenge of our approach is to find a way to access the harvested information so that it can be used for statistical analysis. To study this attack, we use the concept of *honeypots*, i.e., information system resources whose value lies in unauthorized or illicit use of that resource [27]. We play the role of a victim V_i and react on incoming attacks in the same way a legitimate victim would do. For example, we use *spamtraps*, i.e., email accounts used to collect spam, and open email attachments to emulate the infection process of malware that propagates with the help of spam. Furthermore, we also visit links contained in spam mails with client-side honeypots to examine whether or not the spammed URL is malicious and the website tries to install a keylogger via a drive-by download [28,36]. Using these techniques, our honeypot can be infected with a keylogger in an automated way and we obtain information about the attack vector.

After a successful infection, we extract the sample from the honeypot for further analysis. We perform dynamic analysis based on an analysis tool called CWSandbox [37] since static analysis can be defeated by malware using many different techniques [17,24,26]. CWSandbox executes the malware in a controlled environment and

analyzes the behavior of the sample during runtime by observing the system calls issued by the sample. As a result, we obtain an analysis report that includes for example information about changes to the filesystem or the Windows registry, and all network communication generated by the sample during the observation period.

When executing, the keylogger typically first contacts the dropzone to retrieve configuration information. The configuration file commonly includes a list of websites that should be monitored for credentials and similar customization options for the malware. From an attacker's perspective, such a modus operandi is desirable since she does not have to hardcode all configuration options during the attack phase, but can dynamically re-configure which credentials should be stolen after the initial infection. This enables more flexibility since the attacker can configure the infected machines on demand. By executing the keylogger within our analysis environment and closely monitoring its behavior, we can identify the dropzone in an automated way since the keylogger contacts the dropzone at an early stage after starting up.

However, certain families of keylogger already contain all necessary configuration details and do not contact the dropzone: only after keystrokes that represent a credential are observed by these keyloggers, they send the harvested information to the dropzone. In order to study this in a more automated fashion, we need some sort of *user simulation* to actually simulate a victim V. Note that we do not need to generically simulate the full behavior of a user, but only simulate the aspects of user interaction that are *relevant* for keyloggers, e.g., entering credentials in an online banking application or logging into a webmail account. The keylogger then monitors this behavior and sends the collected information to the dropzone, and we have successfully identified the location of a dropzone in an automated way. More information about the actual implementation of user activity simulation is provided in Section 3.1.

2.2 Technical Details of Analyzed Keyloggers

To exemplify a technical realization of the methodology introduced in this paper, we analyzed in detail two different families of keyloggers that are widespread in today's Internet: *Limbo/Nethell* and *ZeuS/Zbot/Wsnpoem*. We provide a short overview of both families in this section. More details and examples are available in a technical report [13].

Limbo/Nethell. This family of malware typically uses malicious websites and drive-by download attacks as attack channel to infect the victims who are lured by social engineering tricks to visit these websites. The malware itself is implemented as a *browser helper object* (BHO), i.e., a plugin for Internet Explorer that can respond to browser events such as navigation, keystrokes, and page loads. With the help of the interface provided by the browser, Limbo can access the Document Object Model (DOM) of the current page and identify sensitive fields which should be monitored for credentials (*form grabbing*). This enables the malware to monitor the content of these fields and defeats simple tricks to conceal the entered password as proposed by Herley and Florêncio [12]. The malware offers a flexible configuration option since the sites to be monitored can be specified during runtime in a configuration file. Upon startup, the malware contacts the dropzone to retrieve the current configuration options from there. Furthermore, this malware has the capability to steal cookies and to extract information

from the Protected Storage (*PStore*). This is a mechanism available in certain versions of Windows which provides applications with an interface to store user data [22] and many applications store credentials like username/password combinations there.

Once a credential is found, the harvested information is sent to the dropzone via a HTTP request to a specific PHP script installed at the dropzone, e.g., `http://example.org/datac.php?userid=21102008_110432_2025612`. This example depicts the initial request right after a successful infection with which the keylogger registers the newly compromised victim. The `userid` parameter encodes the infection date and time, and also a random victim ID. By observing the network communication during the analysis phase, we can automatically determine the network location of the dropzone. The dropzone itself is implemented as a web application that allows the attacker amongst other tasks to browse through all collected information, search for specific credentials, or instruct the victims to download and execute files. We found that these web applications often contain typical configuration errors like for example world-readable directory listings that lead to insecure setups, which we can take advantage of to obtain access to the full data set.

ZeuS/Zbot/Wsnpoem. The attack channel for this family of malware is spam mails that contain a copy of the keylogger as an attachment. The emails use common social engineering tricks, e.g., pretending to be an electronic invoice, in order to trick the victim into opening the attachment. In contrast to Limbo, which uses rather simple techniques to steal credentials, ZeuS is technically more advanced: the malware injects itself into all user space processes and hides its presence. Once it is successfully injected into Internet Explorer, it intercepts HTTP POST requests to observe transmitted credentials. This malware also steals information from cookies and the Protected Storage. All collected information is periodically sent to the dropzone via HTTP requests. The dropzone itself is implemented as a web application and the stolen credentials are either stored in the filesystem or in a database. Again, insecure setups like world-readable directory listings enable the access to the full dropzone data, allowing us to monitor the complete operation of a certain dropzone.

Similar to Limbo, ZeuS can also be dynamically re-configured: after starting up, the malware retrieves the current configuration file from the dropzone. The attacker can for example specify which sites should be monitored (or not be monitored) for credentials. Furthermore, the malware can create screenshot of 50×50 pixels around the mouse pointer taken at every left-click of the mouse for specific sites. This capability is implemented to defeat *visual keyboards*, i.e., instead of entering the sensitive information via the keyboard, they can be entered via mouse clicks. This technique is used by different banks and defeats typical keyloggers. However, by taking a screenshot around the current position of the mouse, an attacker can also obtain these credentials. In addition, the configuration file also specifies for which sites man-in-the-middle attacks should be performed: each time the victim opens such a site, the request is transparently redirected to another machine, which hosts some kind of phishing website that tricks the victim into disclosing even more credentials. Finally, several other configuration options like DNS modification on the victim's machine or update functionality are available.

3 Studying Keylogger-based Attacks

In this section, we introduce the analysis and measurement setup, and present general statistics about the dropzones. The next section then focusses on the results of a systematic study of keylogger-based attacks using keylogger and a dropzone as outlined in the previous sections. All data was collected during a seven-month measurement period between April and October 2008.

3.1 Improving Analysis by Simulating User Behavior

We developed a tool called *SimUser* to simulate the behavior of a victim V_i after an infection with a keylogger. The core of SimUser is based on *AutoIt*, a scripting language designed for automating the Windows GUI and general scripting [1]. It uses a combination of simulated keystrokes, mouse movement, and window/control manipulation in order to automate tasks. We use AutoIt to simulate arbitrary user behavior and implemented SimUser as a frontend to enable efficient generation of user profiles. SimUser itself uses the concept of *behavior templates* that encapsulate an atomic user task, e.g., opening a website and entering a username/password combination in the form fields to log in, or authenticating against an email server and retrieving emails. We implemented 17 behavior templates that cover typical user tasks which require a credential as explained before. These templates can be combined in an arbitrary way to generate a profile that simulates user behavior according to specific needs.

In order to improve our analysis, we execute the keylogger sample for several minutes under the observation of CWSandbox. During the execution, SimUser simulates the behavior of a victim, which browses to several websites and fills out login forms. In the current version, different online banking sites, free webmail providers, as well as social networking sites are visited. Furthermore, CWSandbox was extended to also simulate certain aspects of user activity, e.g., generic clicking on buttons to automatically react on user dialogues. We also store several different credentials in the Windows Protected Storage of the analysis machine as some kind of *honeytoken*. By depositing some credentials in the Protected Storage, we can potentially trigger on more keyloggers.

Simulating user behavior enables us to learn more about the results of a keylogger infection, e.g., we can detect on which sites it triggers and what kind of credentials are stolen. The whole process can be fully automated and we analyzed more than 2,000 keylogger samples with our tools as explained in the next section. Different families of keyloggers can potentially use distinct encodings to transfer the stolen credentials to the dropzone and the dropzone itself uses different techniques to store all stolen information. In order to fully analyze the dropzone and the data contained there, we thus need to manually analyze this communication channel once per family. This knowledge can then be used to extract more information from the dropzone for all samples of this particular family. To provide evidence of the feasibility of this approach, we analyzed two families of keyloggers in detail, as we explain next. Note that even if we cannot fully decode the malware's behavior, we can nevertheless reliably identify the network location of the dropzone based on the information collected during dynamic analysis. This information is already valuable since it can be used for mitigating the dropzone, the simplest approach to stop this whole attack vector.

3.2 Measurement Setup

With the help of CWSandbox, we analyzed more than 2,000 unique Limbo and ZeuS samples collected with different kinds of spamtraps and honeypots, and user submissions at `cwsandbox.org`, in the period between April and October 2008. Based on the generated analysis reports, we detected more than 140 unique Limbo dropzones and 205 unique ZeuS dropzones. To study these dropzones, we implemented a monitoring system that periodically collects information like for example the configuration file.

For 69 Limbo and 4 ZeuS dropzones we were able to *fully* access all logfiles collected at that particular dropzone. This was possible since these dropzones were configured in an insecure way by the attackers, enabling unauthenticated access to all stolen credentials. The remaining dropzones had access controls in place which prevented us from accessing the data. We periodically collected all available data from the open dropzones to study the amount and kind of stolen credentials to get a better understanding of the information stolen by attackers. In total, our monitoring system collected 28 GB of Limbo and 5 GB of ZeuS logfiles during the measurement period.

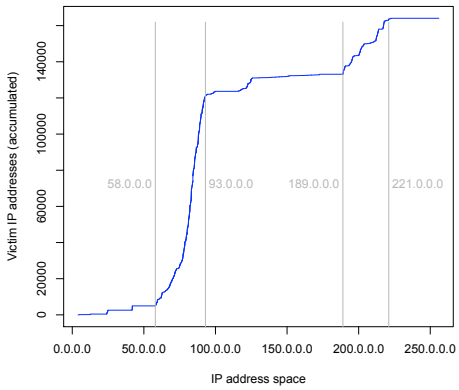
3.3 Analysis of Limbo Victims

To understand the typical victims of keylogger attacks, we performed a statistical analysis of the collected data. The number of unique infected machines and the amount of stolen information per Limbo dropzone for which we had full access is summarized in Table 1. The table is sorted by the number of unique infected machines and contains a detailed overview of the top four dropzones. In total, we collected information about more than 164,000 machines infected with Limbo. Note that an infected machine can potentially be used by many users, compromising the credentials of many victims. Furthermore, the effective number of infected machines might be higher since we might not observe all infected machines during the measurement period. The numbers are thus a lower bound on the actual number of infected machines for a given dropzone. The amount of information collected per dropzone greatly varies since it heavily depends on the configuration of the keylogger (e.g., what kind of credentials should be harvested) and the time we monitored the server. The dropzones themselves are located in many different Autonomous Systems (AS) and no single AS dominates. The country distribution reveals that many dropzones are located in Asia or Russia, but we found also many dropzones located in the United States.

We also examined the *lifetime* for each dropzone and the *infection lifetime* of all victims, i.e., the total time a given machine is infected with Limbo. Each logfile of a dropzone contains records that include a unique victim ID and a timestamp, which indicates when the corresponding harvesting process was started. As the infection lifetime of a victim we define the interval between the timestamp of the last and first record caused by this particular victim. This is the lower bound of the total time of infection since we may not be able to observe all log files from this particular infection and thus underestimate the real infection time. The interval between the last and the first timestamp seen on the whole dropzone is defined as the lifetime of this dropzone. Using these definitions, the average infection time of a victim is about 2 days. This is only a coarse lower bound since we often observe an infected machine only a limited amount of time.

Table 1: Statistical overview of largest Limbo dropzones, sorted according to the total number of infected machines.

Dropzone	# Infected machines	Data amount	AS #	Country	Lifetime in days
webpinkXXX.cn	26,150	1.5 GB	4837	China	36
coXXX-google.cn	12,460	1.2 GB	17464	Malaysia	53
77.XXX.159.202	10,394	503 MB	30968	Russia	99
finXXXonline.com	6,932	438 MB	39823	Estonia	133
Other	108,122	24.4 GB			
Total	164,058	28.0GB			61



(a) Cumulative distribution of IP addresses infected with Limbo.

Country	# Machines	Percentage
Russia	26,700	16,3%
United States	23,704	14,4%
Spain	20,827	12,7%
United Kingdom	19,240	11,7%
Germany	10,633	6,5%
Poland	8,598	5,4%
Australia	6,568	4,0%
Turkey	5,328	3,2%
Brazil	4,369	2,7%
India	3,980	2,4%
Ukraine	2,674	1,6%
Egypt	2,302	1,4%
Italy	1,632	0,9%
Thailand	1,356	0,8%
Other	26,147	16,0%

(b) Distribution of Limbo infections by country.

Fig. 2: Analysis of IP addresses for machines infected with Limbo and their regional distribution.

The maximum lifetime of a Limbo victim we observed was more than 111 days. In contrast, the average lifetime of a dropzones is approximately 61 days.

Figure 2a depicts the cumulative distribution of IP addresses for infected machines based on the more than 164,000 Limbo victims we detected. The distribution is highly non-uniform: The majority of victims are located in the IP address ranges between 58.* – 92.* and 189.* – 220.*. Surprisingly, this is consistent with similar analysis of spam relays and scam hosts [3,29]. It could indicate that these IP ranges are often abused by attackers and that future research should focus on securing especially these ranges.

We determined the geographical location of each victim by using the Geo-IP tool Maxmind [18]. The distribution of Limbo infections by country is shown in Figure 2b. We found a total of 175 different countries and almost one third of the infected machines are located in either Russia or the United States.

3.4 Analysis of ZeuS Victims

We performed a similar analysis for the ZeuS dropzones and the victims infected with this malware. Figure 3a lists the top five countries in which the dropzones are located based on 205 dropzones we identified with our method. Most ZeuS dropzones can be found in North America, Russia, and East Asia — a results that also applies to the Limbo dropzones. We also found that the dropzones are located in many different Autonomous Systems (68 different AS in total), but several AS host a larger percentage of ZeuS dropzones: The three most common AS host 49% of all dropzones, indicating that there are some providers preferred by the attackers. Presumably those providers offer *bullet-proof hosting*, i.e., takedown requests are not handled properly by these providers or the providers even tolerate certain abusive behavior.

Country	# Dropzones	Percentage
United States	34	17%
Russia	29	14%
Netherlands	16	8%
Malaysia	14	7%
China	8	4%

(a) Top countries in which ZeuS dropzones are located.

OS version	# Infected Machines	%
Windows XP SP2	6,629	70.2 %
Windows XP SP0	1,264	13.1 %
Windows XP SP1	1,146	12.1 %
Windows 2000 SP4	285	3.0 %
Other	156	1.6 %

(b) Distribution of operating system for machines infected with ZeuS.

Fig. 3: General statistics for ZeuS dropzones and victims.

The four dropzones we had full access to contained information stolen from about 9,480 infected machines. Based on this data, we can determine the operating system version of each infected machine since the keylogger also extracts this information. Figure 3b provides an overview of the operating system running on the infected machines. The majority of victims is using Windows XP with Service Pack 2, thus they are not on the latest patch level (Service Pack 3 was released on May 6, 2008). A large fraction of machines run on even older version of the operating system. Only a minority of all victims have the latest service pack installed or are running Windows Vista. We also examined the language version of the operating system. Most infected machines have either English (53.8%) or Spanish (20.2%) as language. Consistent to the machines infected with Limbo, the majority of ZeuS infections can be found in the two network ranges 58.* – 92.* (56.9%) and 189.* – 220.* (25.8%).

As explained in Section 2.2, ZeuS can be dynamically re-configured by the attacker via a configuration file. The most frequent configurations are shown in Table 2. Websites that should be logged are listed in the first part of the table and the second part enumerates the websites that should be logged and where a screenshot should be taken. Online banking websites clearly dominate this statistic and indicate that these attacks aim at stealing credentials for bank accounts. Finally, websites where no keystrokes should be recorded are listed at the end of the table. This excluding of websites from the harvesting process is presumably done in order to minimize the logged data.

Table 2: Overview of top four websites a) to be logged, b) to be logged including a screenshot, and c) not to be logged.

	Website	# Appearances (205 dropzones)
a)	https://internetbanking.gad.de/*/portal?bankid=*	183
	https://finanzportal.fiducia.de/*?rzid=*&rzbk=*	177
	https://www.vr-networld-ebanking.de/	176
	https://www.gruposantander.es/*	167
b)	@*/login.osmp.ru/*	94
	@*/atl.osmp.ru/*	94
	@https://*.e-gold.com/*	39
	@https://netteller.tsw.com.au/*/ntv45.asp?wci=entry	29
c)	!http://*myspace.com*	132
	!*.microsoft.com/*	98
	!http://*odnoklassniki.ru/*	80
	!http://vkontakte.ru/*	72

4 Analysis of Stolen Credentials

Based on the data collected by our monitoring system, we analyzed what kind of credentials are stolen by keyloggers. This enables a unique point of view into the underground market since we can study what goods are available for the criminals from a first-hands perspective. We mainly focus on five different areas: online banking, credit cards, online auctions, email passwords, and social networks. At first sight, the last two areas do not seem to be very interesting for an attacker. However, especially these two kinds of stolen credentials can be abused in many ways, e.g., for identity theft, spear phishing, spamming, anonymous mail accounts, and other illicit activities. This is also reflected in the market price for these two types of goods as depicted in Table 3 based on a study by Symantec [33].

Identifying which credentials are stolen among the large number of collected data is a challenge. The key insight is that credentials are typically sent in HTTP POST requests from the victim to the provider. To find credentials, we thus need to pin-point

Table 3: Breakdown of prices for different goods and services available for sale on the underground market according to a study by Symantec [33]. *Percentage* indicates how often these goods are offered.

Goods and services	Percentage	Range of prices
Bank accounts	22%	$10 – $1000
Credit cards	13%	$0.40 – $20
Full identities	9%	$1 – $15
Online auction site accounts	7%	$1 – $8
Email passwords	5%	$4 – $30
Drop (request or offer)	5%	10% – 50% of total drop amount
Proxies	5%	$1.50 – $30

which requests fields are actually relevant and contain sensitive information. We use a trick to identify these fields: when a victim enters his credential via the keyboard, Limbo stores this information together with the current URL. Based on the collected data, we can thus build provider-specific models M_{P_i} that describe which input fields at P_i contain sensitive information. For example, $M_{\text{login.live.com}} = \{\text{login}, \text{passwd}\}$ and $M_{\text{paypal.com}} = \{\text{login\_email}, \text{login\_password}\}$. These models can then be used to search through all collected data to find the credentials, independent of whether the victim entered the information via the keyboard or they were inserted by a program via the Protected Storage. In total, we generated 151,070 provider-specific models. These models cover all domains for which keystrokes were logged by all infected machines. For our analysis, we only used a subset of all provider-specific models that are relevant for the area we analyzed.

We also need to take typing errors into account: if a victim makes a typing error during the authentication process, this attempt is not a valid credential and we must not include it in our statistics. We implement this by keeping track of which credentials are entered by each victim and only counting each attempt to authenticate at a specific provider once. During analysis, we also used methods like pattern matching or heuristics to find specific credentials as we explain below.

4.1 Banking Websites

We used 707 banking models that cover banking sites like Bank of America or Lloyds Bank, and also e-commerce business platforms like PayPal. These models were chosen based on the ZeuS configuration files since this keylogger aims specifically at stealing banking credentials. In total, we found 10,775 unique bank account credentials in all logfiles. Figure 4a provides an overview of the top five banking websites for which we found stolen credentials. The distribution has a long tail: for the majority of banking websites, we found less than 30 credentials.

Banking Website	# Stolen Credentials
PayPal	2,263
Commonwealth Bank	851
HSBC Holding	579
Bank of America	531
Lloyds Bank	447

(a) Overview of top five banking websites for which credentials were stolen.

Credit Card Type	# Stolen Credit Cards
Visa	3,764
MasterCard	1,431
American Express	406
Diners Club	36
Other	45

(b) Overview of stolen credit card information.

Fig. 4: Analysis of stolen banking accounts and credit card data.

ZeuS has the capability to parse the content of specific online banking website to extract additional information from them, e.g., the current account balance. We found 25 unique victims whose account balance was disclosed this way. In total, these 25 bank accounts hold more than $130,000 in checking and savings (mean value is $1,768.45,

average is $5,225). Based on this data, we can speculate that the attackers can potentially access millions of dollars on the more than 10,700 compromised bank accounts we recovered during our analysis.

4.2 Credit Card Data

To find stolen credit card data, the approach with provider-specific models cannot be used since a credit card number can be entered on a site with an arbitrary field name. For example, an American site might use the field name `cc_number` or `cardNumber`, whereas a Spanish site could use `numeroTarjeta`. We thus use a pattern-based approach to identify credit cards and take the syntactic structure of credit card numbers into account: each credit card has a fixed structure (e.g., MasterCard numbers are 16 digits and the first two digits are 51-55) that we can identify. Furthermore, the first six digits of the credit card number are the Issuer Identification Number (IIN) which we can also identify. For each potential credit card number, we also check the validity with the Luhn algorithm [19], a checksum formula used to guard against one digit errors in transmission. Passing the Luhn check is only a necessary condition for card validity and helps us to discard numbers containing typing errors.

With this combination of patterns and heuristics, we found 5,682 valid credit card numbers. Figure 4b provides an overview of the different credit card types we found. To estimate the potential loss due to stolen credit cards we use the median loss amount for credit cards of $223.50 per card as reported in the 2008 Internet Crime Complaint Center's Internet Crime Report [14]. If we assume that all credit cards we detected are abused by the attacker, we obtain an estimated loss of funds of almost $1,270,000.

4.3 Email Passwords

Large portals and free webmail providers like Yahoo!, Google, Windows Live, or AOL are among the most popular websites on the Internet: 18 sites of the Alexa Top 50 belong to this category [2]. Accordingly, we expect that also many credentials are stolen from these kinds of sites. We used 37 provider-specific models that cover the large sites of this category. In total, we found 149,458 full, unique credentials. We detected many instances where the attackers could harvest many distinct webmail credentials from just one infected machine. This could indicate infected system in public places, e.g., schools or Internet cafes, to which many people have access. Figure 5a provides an overview of the distribution for all stolen email credentials.

4.4 Social Networks and Online Trading Platforms

Another category of popular sites are social networks like Facebook and MySpace, or other sites with a social component like YouTube. Of the Alexa Top 50, 14 sites belong to this category. To analyze stolen credentials from social networks, we used 57 provider-specific models to cover common sites in this category. In total, we found 78,359 stolen credentials and Figure 5b provides an overview of the distribution. Such credentials can for example be used by the attacker for spear phishing attacks.

Webmail Provider	# Stolen Credentials
Windows Live	66,540
Yahoo!	27,832
mail.ru	17,599
Rambler	5,379
yandex.ru	5,314
Google	4,783
Other	22,011

(a) Overview of stolen credentials from portals and webmail providers.

Social Network	# Stolen Credentials
Facebook	14,698
hi5	8,310
nasza-klasa.pl	7,107
odnoklassniki.ru	5,732
Bebo	5,029
YouTube	4,007
Other	33,476

(b) Overview of stolen credentials from social networking sites.

Fig. 5: Analysis of stolen credentials from free webmail providers and social networking sites.

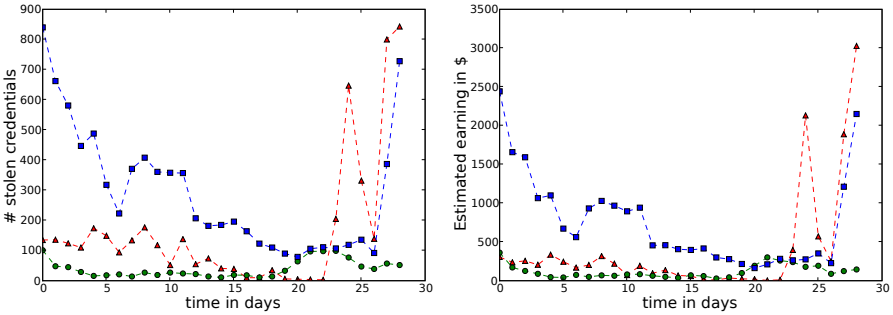
The final type of stolen credentials we analyze are online trading platforms. We used provider-specific models for the big four platforms: eBay, Amazon, Allegro.pl (third biggest platform world-wide, popular in Poland), and Overstock.com. In total, we found 7,105 credentials that were stolen from all victims. Of these, the majority belong to eBay with 5,712 and Allegro.pl with 885. We found another 477 credentials for Amazon and 31 for Overstock.com. This kind of credentials can for example be used for money laundering.

4.5 Underground Market

The analysis of stolen credentials also enables us to estimate the total value of this information on the underground market: each credential is a marketable good that can be sold in dedicated forums or IRC channels [10,20]. If we multiply the number of stolen credentials with the current market price, we obtain an estimate of the overall value of the harvested information. Table 4 summarizes the results of this computation. These results are based on market prices as reported by Symantec [33,34]. Other antivirus vendors performed similar studies and their estimated market prices for these goods are similar, thus these prices reflect – to the best of our knowledge – actual prices paid on the underground market for stolen credentials. These results indicate that the information collected during our measurement period is potentially worth several millions of dollars.

Table 4: Estimation of total value of stolen credentials recovered during measurement period. Underground market prices are based on a study by Symantec [33].

Stolen credentials	Amount	Range of prices	Range of value
Bank accounts	10,775	$10 – $1000	$107,750 – $10,775,000
Credit cards	5,682	$0.40 – $20	$2,272 – $113,640
Full identities / Social Networks	78,359	$1 – $15	$78,359 – $1,175,385
Online auction site accounts	7,105	$1 – $8	$7,105 – $56,840
Email passwords	149,458	$4 – $30	$597,832 – $4,483,740
Total	224,485	n/a	$793,318 – $16,604,605



(a) Number of stolen credentials per day

(b) Estimated daily earning for attackers

Fig. 6: Number of unique stolen credentials and estimated amount of money earned per day due to harvested keylogger data for three Limbo dropzones. Other dropzones have a similar distribution.

Given the fact that we studied just two families of keyloggers and obtained detailed information about only 70 dropzones (from a total of more than 240 dropzones that we detected during our study), we can argue that the overall size of the underground market is considerably larger.

We also studied the estimated revenue of the individual dropzones. For each dropzone, we computed the total number of credentials stolen per day given the five categories examined in this paper. Furthermore, we use the range of prices reported by Symantec [33] to estimate the potential daily earnings of the operator of each dropzone. The results of this analysis are shown exemplarily in Figure 6 for three different Limbo dropzones. These dropzones were chosen since we were able to obtain continuous data for more than four weeks for these sites. However, the distribution for other dropzones is very similar. Figure 6a depicts the number of unique stolen credentials per day. This number varies greatly per day, presumably due to the fact that the malware has a certain rate at which new victims are infected and this rate also varies per day. We also observe that there is a steady stream of fresh credentials that can then be traded at the underground market. On the other hand, Figure 6b provides an overview of the estimated value of stolen credentials for each particular day. We obtain this estimate by multiplying the number of credentials stolen per day with the *lowest* market price according to the study by Symantec [33] (see Figure 3). This conservative assumption leads to a lower bound of the potential daily income of the attackers. The results indicate that an attacker can earn several hundreds of dollars (or even thousands of dollars) per day based on attacks with keyloggers — a seemingly lucrative business.

4.6 Discussion

Besides the five categories discussed in this section, ZeuS and Limbo steal many more credentials and send them back to the attacker. In total, the collected logfiles contain more than three million unique keystroke logs. With the provider-specific models ex-

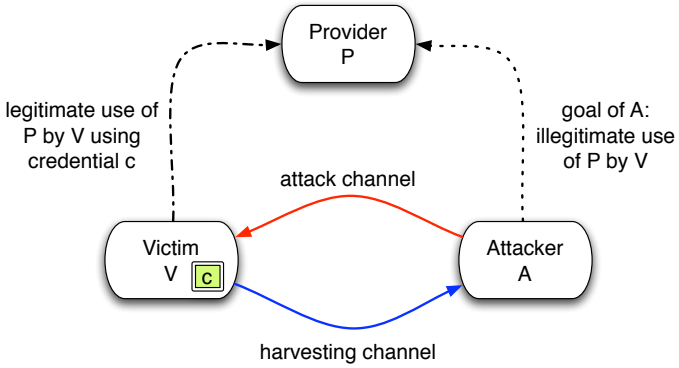


Fig. 7: Structure of attacks susceptible to our method.

amined in the five categories, we only cover the larger types of attacked sites and high-profile targets. Many more types of stolen sensitive information against small websites or e-commerce companies are not covered by our analysis. As part of future work, we plan to extend our analysis and also include an analysis of stolen cookies and the information extracted from the Protected Storage of the infected machines.

5 Conclusion and Future Work

Our user simulation approach is rather ad-hoc and does not allow us to study all aspects of keyloggers. The main limitation is that we do not know exactly on which sites the keylogger becomes active and thus we may miss specific keyloggers. Our empirical results show that keyloggers typically target the main online banking websites and also extract information from the Protected Storage. Nevertheless, we may miss keyloggers that only steal credentials from a very limited set of sites. This limitation could be circumvented by using more powerful malware analysis techniques like multi-path execution [23] or a combination of dynamic and static analysis [16]. Another limitation is that we do not exactly determine which credentials are stolen. Techniques from the area of taint tracking [25,38] can be added to our current system to pinpoint the stolen credentials. Despite these limitation, the ad-hoc approach works in practice and enables us to study keyloggers as we showed in Section 3 and 4.

The approach we took in this paper works for keylogger-based attacks, but it can in fact be generalized to other attacks as well, for example classical phishing. The abstract schema behind the class of attacks that can be analyzed is shown in Figure 7. There, a provider P offers some online service like an online bank or an online trading platform (like eBay or Amazon). The victim V is a registered user of the service provided by P and uses credentials c to authenticate as a legitimate user towards P. The attacker A wants to use P's service by pretending to be V. To do this, A needs V's credentials c. So for a successful attack, there must exist a (possibly indirect) communication channel from V to A over which information about c can flow. We call this channel the *harvesting channel*. Apart from the harvesting channel there also exists another (possibly

indirect) communication channel from A to V. This channel is used by the attacker to initiate or trigger an attack. We call this channel the *attack channel*. The generalization of our approach presented in this paper involves an analysis of the harvesting channel. This is a hard task, which together with more automation is a promising line for future work in this area.

Acknowledgements. We would like to thank Carsten Willems for extending CWSandbox such that certain aspects of user simulation such as generic clicking are directly implemented within the sandbox. Jan Göbel provided valuable feedback on a previous version of this paper that substantially improved its presentation. Frank Boldewin helped in analyzing the ZeuS configuration files and the AusCERT team was very helpful in notifying the victims. This work has been supported by the WOMBAT and FORWARD projects funded by the European Commission.

References

1. AutoIt Script Home Page. Internet: `http://www.autoitscript.com/`, 2009.
2. Alexa, the Web Information Company. Global Top Sites, September 2008. `http://alexa.com/site/ds/top_sites?ts_mode=global`.
3. David S. Anderson, Chris Fleizach, Stefan Savage, and Geoffrey M. Voelker. Spamscatter: Characterizing Internet Scam Hosting Infrastructure. In *USENIX Security Symposium*, 2007.
4. Anonymous. Comment about posting "Good ol' #CCpower" on honeyblog. Internet: `http://honeyblog.org/archives/194-CCpower-Only-Scam.html`, June 2008.
5. Madhusudhanan Chandrasekaran, Ramkumar Chinchani, and Shambhu Upadhyaya. PHONEY: Mimicking User Response to Detect Phishing Attacks. In *Symposium on World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2006.
6. Taehwan Choi, Sooel Son, Mohamed Gouda, and Jorge Cobb. Pharewell to Phishing. In *Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS)*, 2008.
7. Neil Chou, Robert Ledesma, Yuka Teraguchi, and John C. Mitchell. Client-Side Defense Against Web-Based Identity Theft. In *Network and Distributed System Security Symposium (NDSS)*, 2004.
8. Rachna Dhamija and J. D. Tygar. The Battle Against Phishing: Dynamic Security Skins. In *Symposium on Usable Privacy and Security (SOUPS)*, 2005.
9. Finjan. Malicious Page of the Month. `http://www.finjan.com/Content.aspx?id=1367`, April 2008.
10. Jason Franklin, Vern Paxson, Adrian Perrig, and Stefan Savage. An Inquiry Into the Nature and Causes of the Wealth of Internet Miscreants. In *Conference on Computer and Communications Security (CCS)*, 2007.
11. Sebastian Gajek and Ahmad-Reza Sadeghi. A Forensic Framework for Tracing Phishers. In *IFIP WG 9.2, 9.6/11.6, 11.7/FIDIS International Summer School on The Future of Identity in the Information Society*, Karlstad University, Sweden, August 2007.
12. Cormac Herley and Dinei Florencio. How To Login From an Internet Cafe Without Worrying About Keyloggers. In *Symposium on Usable Privacy and Security (SOUPS)*, 2006.
13. Thorsten Holz, Markus Engelberth, and Felix Freiling. Learning More About the Underground Economy: A Case-Study of Keyloggers and Dropzones. Technical Report TR-2008-006, University of Mannheim, 2008.
14. Internet Crime Complaint Center (IC3). 2008 Internet Crime Report, March 2009. `http://www.ic3.gov/media/annualreports.aspx`.

15. Chris Kanich, Christian Kreibich, Kirill Levchenko, Brandon Enright, Geoffrey M. Voelker, Vern Paxson, and Stefan Savage. Spamalytics: An Empirical Analysis of Spam Marketing Conversion. In *Conference on Computer and Communications Security (CCS)*, 2008.
16. Engin Kirda, Christopher Kruegel, Greg Banks, Giovanni Vigna, and Richard Kemmerer. Behavior-based Spyware Detection. In *USENIX Security Symposium*, 2006.
17. Cullen Linn and Saumya Debray. Obfuscation of Executable Code to Improve Resistance to Static Disassembly. In *Conference on Computer and Communications Security (CCS)*, 2003.
18. MaxMind LLC. MaxMind GeoIP. http://www.maxmind.com/app/ip-location, August 2008.
19. Hans P. Luhn. Computer for Verifying Numbers, August 1960. U.S. Patent 2,950,048.
20. Jerry Martin and Rob Thomas. the underground economy: priceless. *USENIX ;login:*, 31(6), December 2006.
21. Jonathan M. McCune, Adrian Perrig, and Michael K. Reiter. Bump in the Ether: A Framework for Securing Sensitive User Input. In *USENIX Annual Technical Conference*, 2006.
22. Microsoft. Protected Storage (Pstore), August 2008. Microsoft Developer Network (MSDN).
23. Andreas Moser, Christopher Kruegel, and Engin Kirda. Exploring Multiple Execution Paths for Malware Analysis. In *IEEE Symposium on Security and Privacy*, 2007.
24. Andreas Moser, Christopher Kruegel, and Engin Kirda. Limits of Static Analysis for Malware Detection. In *Annual Computer Security Applications Conference (ACSAC)*, 2007.
25. James Newsome and Dawn Xiaodong Song. Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation of Exploits on Commodity Software. In *Network and Distributed System Security Symposium (NDSS)*, 2005.
26. Igor V. Popov, Saumya K. Debray, and Gregory R. Andrews. Binary Obfuscation Using Signals. In *USENIX Security Symposium*, 2007.
27. The Honeynet Project. *Know Your Enemy: Learning About Security Threats*. Addison-Wesley Longman, 2nd edition, May 2004.
28. Niels Provos, Panayiotis Mavrommatis, Moheeb A. Rajab, and Fabian Monrose. All Your iFRAMEs Point to Us. In *USENIX Security Symposium*, 2008.
29. Anirudh Ramachandran and Nick Feamster. Understanding the Network-Level Behavior of Spammers. *SIGCOMM Comput. Commun. Rev.*, 36(4):291–302, 2006.
30. SecureWorks. PRG Trojan. `http://www.secureworks.com/research/threats/prgtrojan/`, June 2007.
31. SecureWorks. Coreflood Report. `http://www.secureworks.com/research/threats/coreflood-report/`, August 2008.
32. Mika Stahlberg. The Trojan Money Spinner. In *Virus Bulletin Conference*, 2007.
33. Symantec. Global Internet Security Threat Report: Trends for July – December 07, April 2008.
34. Symantec. Report on the Underground Economy July 07 – June 08, November 2008.
35. XiaoFeng Wang, Zhuowei Li, Ninghui Li, and Jong Youl Cho. PRECIP: Towards Practical and Retrofittable Confidential Information Protection. In *Network and Distributed System Security Symposium (NDSS)*, 2008.
36. Yi-Min Wang, Doug Beck, Xuxian Jiang, Roussi Roussev, Chad Verbowski, Shuo Chen, and Samuel T. King. Automated Web Patrol with Strider HoneyMonkeys: Finding Web Sites That Exploit Browser Vulnerabilities. In *Network and Distributed System Security Symposium (NDSS)*, 2006.
37. Carsten Willems, Thorsten Holz, and Felix Freiling. Toward Automated Dynamic Malware Analysis Using CWSandbox. *IEEE Security & Privacy Magazine*, 5(2):32–39, March 2007.
38. Heng Yin, Dawn Song, Manuel Egele, Christopher Kruegel, and Engin Kirda. Panorama: Capturing System-wide Information Flow for Malware Detection and Analysis. In *Conference on Computer and Communications Security (CCS)*, 2007.

Chapter 5
Assessing Cybercrime Through the Eyes of the WOMBAT

Marc Dacier, Corrado Leita, Olivier Thonnard, Van-Hau Pham and Engin Kirda

5.1 Foreword

The WOMBAT project is a collaborative European funded research project that aims at providing new means to understand the existing and emerging threats that are targeting the Internet economy and the net citizens. The approach carried out by the partners include a data collection effort as well as some sophisticated analysis techniques. In this chapter, we present one of the threats-related data collection system in use by the project, as well as some of the early results obtained when digging into these data sets.

In [21], the authors offer a thorough presentation of one of the data collection infrastructures used within the WOMBAT project to collect threats-related data. The presentation is very detailed, going as far as explaining the database scheme used to represent the vast amount of information they have access to. In the following pages, we wish to offer to the reader an early synthesis of the various results that have been obtained when analyzing this large amount of information. However, in order for this chapter to be as self-contained as possible, we start the presentation by re-stating the rationales for this work, as well as by providing a summarized introduction to the data collection infrastructure. We invite the reader who is already familiar with the WOMBAT project to skip this part and move directly to the presentation of the results.

Marc Dacier
Symantec, Sophia Antipolis, France, e-mail: marc_dacier@symantec.com

Corrado Leita
Symantec, Sophia Antipolis, France, e-mail: corrado_leita@symantec.com

Olivier Thonnard
Eurecom, Sophia Antipolis, France, e-mail: thonnard@eurecom.fr

Van-Hau Pham
Eurecom, Sophia Antipolis, France, e-mail: pham@eurecom.fr

Engin Kirda
Eurecom, Sophia Antipolis, France, e-mail: kirda@eurecom.fr

5.2 Introduction

Understanding the existing and emerging threats on the Internet should help to effectively protect the Internet economy, our information systems and the Internet users. To reach this goal, it is necessary to collect sound measurements about the ongoing attack processes observed worldwide on the Internet. In the last years, the experimental study of Internet threats has gained much attention and many valuable initiatives now exist for monitoring malicious activities or for capturing malware binaries. Important contributions have been made in the field such as: i) the so-called Darknets and Internet telescopes [23, 30, 35], ii) various projects based on the development of low- or high-interaction honeypots [2, 13, 31, 34, 41], and iii) other initiatives aiming at collecting and sharing firewall and IDS logs [14].

The Leurré.com project was initially launched in 2003 and has since then been integrated and further developed within the WOMBAT project. It is based on a worldwide distributed system of honeypots running in more than 30 different countries covering the five continents. The main objective with this infrastructure is to get a more realistic picture of certain classes of threats happening on the Internet by collecting unbiased quantitative data in a long-term perspective. We have decided to keep in one centralized database very precise information concerning a limited number of nodes under close scrutiny. Concretely speaking, we initially deployed identically configured honeypots based on Honeyd [31] on the premises of several partners around the globe. Within WOMBAT, we have improved the infrastructure in a major way by building and deploying new honeypot sensors based on the ScriptGen technology [17, 18, 20]. These new sensors dramatically improve the interaction with the attackers and, hence, enrich our data collection. We record all packets sent to or from these machines, on all platforms, and we store the whole traffic into a database, enriched with some contextual information and with meta-data describing the observed attack sessions. In the next Sections, we present these two data collection infrastructures and, then, offer a synthesis of some of the results obtained by the WOMBAT partners when analyzing the data at their disposal.

This chapter begins with the presentation of the initial data collection infrastructure that is based on the deployment of low-interaction honeypots, for which we give a series of simple examples that reveal the kind of information that such low level traces can provide. Then, we present how we have extended our infrastructure with the SGNET deployment, which has recently been opened to anybody willing to host one of its sensors. Section 5.5 presents how the identification of so-called *attack events* (representing specific activities over limited period of times) enables us to observe the evolution of what we hypothesize to be *armies of zombies*, some of them remaining visible for more than 700 days. Section 5.6 gets deeper into the analysis of the traces, highlighting the usefulness of applying what we call a *multidimensional analysis* to the honeypot events. Section 5.7 provides some insights into the kind of contextual information that SGNET can offer whenever collecting malware. Concrete examples are given that demonstrate the usefulness of such information in discovering new threats and in better understanding the links between the code injection phase, the shellcode injected and the uploaded malware itself.

5.3 Leurre.com v1.0 Honeyd

5.3.1 Historical background

The Institut Eurécom has started collecting attack traces on the Internet in 2003 by means of honeypot responders. The first platform consisted of three high interaction honeypots built on top of the VMware technology (the interested readers in the platform configuration are invited to read

[12] for more information). As shown in [11, 12], these first experiments allowed us to detect some locality in Internet attacks: activities seen in some networks were not observed in others. To validate this assumption, we decided to deploy multiple honeypots in diverse locations. With diversity, we refer both to the geographical location and to the sensor environment (education, government, private sectors, etc). However, the VMware-based solution did not seem to be scalable. First, this solution had a high cost in terms of security maintenance. Second, it required significant hardware resources. In fact, to avoid legal issues we would have needed to ensure that these systems could not be compromised and could not be exploited by attackers as stepping stones to attack other hosts. For those reasons, we have chosen a low-interaction honeypot solution, honeyd [31]. This solution allowed us to deploy low-cost platforms, easy to maintain and with low security risk, hosted by partners on a voluntary basis. The low-cost of the solution allowed us to build a distributed honeynet consisting now of more than 50 sensors distributed all over the world, collecting data on network attacks and representing this information under the form of a relational database accessible to all the parters. Information about the identity of the partners and the observed attackers is protected by a Non-Disclosure Agreement signed by each entity participating to the project. We have developed all the required software to automate the various regular maintenance tasks (new installation, reconfiguration, log collection, backups, etc.) to reduce the maintenance overhead related to the management of such a complex system.

5.3.2 Some technical aspects

We describe here some important technical aspects, including the platform architecture, the logs collection mechanism, the DB uploading mechanism, and the data enrichment mechanism.

Platform architecture: As mentioned before, the main objective is to compare unsolicited network traffic in diverse locations. To make sound comparisons, the platform architecture must be the same everywhere. We tried to make our Honeyd-based solution as similar as possible to the initial VMware setup. We configured Honeyd to simulate 3 virtual hosts running on three different (consecutive) IP addresses. We configured Honeyd's personality engine to emulate the presence of two different configurations, namely two identical virtual machines emulating Windows 2000 SP3, and one machine emulating a Linux Kernel 2.4.20. To the first two configurations (resp. the last) correspond a number of open ports: FTP, Telnet, Web server, Netbios name service, Netbios session service, and Service Message Block (resp. FTP server, SSH server, Web server on ports (80), Proxy (port 8080,8081), remote shell (port 514), LPD Printer service (port 515) and portmapper). We require from each partner hosting the platform a fourth IP address used to access the physical host running Honeyd and perform maintenance tasks. We run tcpdump [36] to capture the complete network traces on each platform. As a security measure, a reverse firewall is set up to protect our system. That is, we accept only incoming connections and drop all the connections that could eventually be initiated from our system (in theory, this should never happen). The access to the host machine is very limited: SSH connections are only allowed in a two-hour daily timeframe and only if it is initiated by our maintenance servers.

Data collection mechanism: An automatized mechanism allows us, on a daily basis, to connect to the platforms through an encrypted connection to collect the tcpdump traces. The script downloads not only the last day's log file but also the eventual older ones that could not have been collected in the previous days due to, for example, a connectivity problem. All the log files are stored on a central server.

Data uploading mechanism: Just after the data retrieval, the log files are then uploaded into a large Oracle database by a set of Perl programs. These programs take tcpdump files as input and parse them in order to create different abstraction levels. The lowest one corresponds to the raw

tcpdump traffic. The higher level is built on the lower ones and has richer semantics. Due to space constraints, we do not present here all the concepts, but instead we will focus only on the most important notions.

1. **Source**: A source corresponds to an IP address that has sent at least one packet to, at least, one platform. Note that, in our Source model, a given IP address can correspond to several distinct sources. That is, an IP remains associated to a given source as long as there is no more than 25 hours between 2 consecutive packets received from that IP. After such a delay, a new source will be assigned to the IP. By grouping packets by sources instead of by IPs, we minimize the risk of gathering packets sent by distinct physical machines that have been assigned the same IP dynamically after 25 hours.
2. **Large_Session**: it's the set of packets which have been exchanged between one Source and a particular honeypot sensor. A Large_Session is characterized by the duration of the attack, the number of packets sent by the Source, the number of virtual machines targeted by the source on that specific platform, ...
3. **Ports sequence**: A ports sequence is a time ordered sequence of ports (without duplicates) a source has contacted on a given virtual machine. For example, if an attacker sends the following packets: ICMP, 135 TCP, 135 TCP, 139 TCP to a given virtual machine, the associated ports sequence will be represented by the string $I|135T|139T$. Each large session can have, at most, three distinct clusters associated to it.
   This is an important feature that allows us to classify the attacks into different classes. In fact, as mentioned in [12], most attack tools are automatized, it is as likely that the same attack tools will leave the same port sequences on different platforms.
4. **Tiny_Session**: A Tiny_Session groups the packets exchanged between one source and one virtual host. A Large_Session is thus composed of up to three Tiny_Sessions, ordered according to the virtual hosts IP addresses.
5. **(Attack) Cluster**: A Cluster is a set of Sources having exhibited the same network fingerprint on a honeypot sensor. We apply a clustering algorithm on the traffic generated by the sources. The first step of this clustering algorithm consists in grouping large sessions into bags. This grouping aims at differentiating between various classes of activity taking into consideration a set of preliminary discriminators, namely the number of targeted virtual hosts and the unsorted list of port sequences hitting them. In order to further refine the bags, a set of continuous parameters is taken into consideration for each large session, namely: its duration, the total number of packets, the average inter arrival time of packets, and the number of packets per tiny session. These parameters can assume any value in the range $[0,\infty]$, but some ranges of their values may be used to define bag subclasses. This is done through a peak picking algorithm that identifies ranges of values considered discriminating for the bag refinement. Large sessions belonging to a bag and sharing the same matching intervals are grouped together in a cluster.
   A very last refinement step is the payload validation. The algorithm considers the concatenation of all the payloads sent by the attacker within a large session ordered according to the arrival time. If it identifies within a cluster multiple groups of large sessions sharing similar payloads, it further refines the cluster according to these groups. In summary, a cluster is by design a set of large sessions that seem to be originating from a similar attack tool.
6. A **Cluster time series** $\Phi_{T,c}$ is a function defined over a period of time T, T being defined as a time interval (in days). That function returns the amount of sources per day associated to a cluster c.
7. An **Observed cluster time series** $\Phi_{T,c,op}$ is a function defined over a period of time T, T being defined as a time interval (in days). That function returns the amount of sources per day associated to a cluster c that can be seen from a given *observation view point op*. The observation view point can either be a specific platform or a specific country of origin. In the first case, $\Phi_{T,c,platform_X}$ returns, per day, the amount of sources belonging to cluster c that have hit $platform_X$. Similarly, in the second case, $\Phi_{T,c,country_X}$ returns, per day, the amount of sources belonging to cluster c that are geographically located in $country_X$. Clearly, we always have: $\Phi_{T,c} = \sum^{\forall i \in countries} \Phi_{T,c,i} = \sum^{\forall x \in platforms} \Phi_{T,c,x}$

Information enrichment

Finally, to enrich the information about each source, we add to it three other dimensions:

1. **Geographical information**: To obtain geographical location such as: organization, ISP, country of a given IP address, we have initially used Netgeo [25], developed in the context of CAIDA Project. It provided a very surprising result which considered Netherlands and Australia as two of the most attacking countries. As a sanity check, we have used Maxmind [22] and we have detected problems with the Netgeo classification. [29] provides a comparison of these two tools. It comes out from this analysis that Netherlands and Australia were not among the top attacking countries anymore when using different sources of information for the geographical location of attacking IP addresses.
2. **OS fingerprint**: To figure out the OS of attacking hosts, we have used passive OS fingerprinting techniques. We take advantage of disco [1] and p0f [42]. It has been shown that p0f is more accurate than disco. Active fingerprinting techniques such as Nmap, Quezo, or Xprobe have not been considered to minimize the risk of alerting the attacker of our investigations.
3. **Domain name:** We also do the reverse DNS lookup to get the domain name of the attacking machine if it is available.

5.3.3 Generic picture

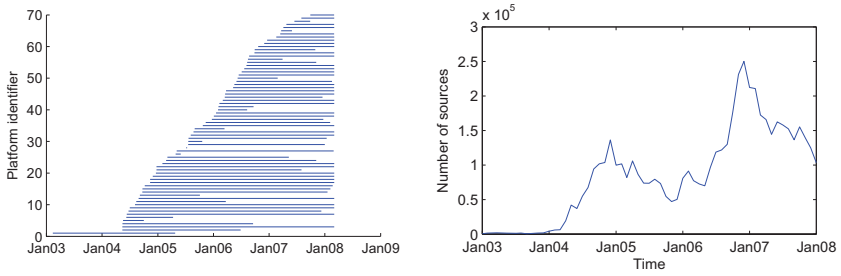


Fig. 5.1 Left: Evolution of platforms, Right: number of sources

Figure 5.1 (left) shows the evolution of platforms. Each curve corresponds to the time life of a platform. As we can see, we started our data collection in January 2003 with one VMware honeypot and we have started to deploy the distributed low interaction honeypots in April 2004. Since then, the number of partners joining us has kept increasing. In total, we have around 50 offical partners and around 20 former partners. These platforms have, in total, covered 37 different /8 networks, locating in 28 different countries in five continents. In total, we have observed 5173624 sources corresponding to 3461745 different IP addresses. Figure 5.1 (right) shows the evolution of the number of sources over time. The variation of the curve is of course influenced by the number of platforms. Note that up to April 2004, the traffic is negligible. After that, the number of sources has increased. It is interesting to observe that the number of sources on the last six months of 2004 is much higher than that of the last six months of 2005 even through, in the second case, we have more platforms. In total, there are 155041 different clusters. Figure 5.2 (left) represents the cumulative distribution function of number of sources per number of cluster. Point (X,Y) on the curve means

that Y*100% of the total amount of clusters contain less than X sources. As we can see, most of clusters are very small. There are, in fact, only 15521 clusters containing more than 10 sources each. Interestingly enough, by querying the database one can find that these clusters, ie. around 10% of the total number of clusters, contain in fact 95% of the observed attacks! In other words, the bulk of the attacks is found in a limited number of clusters whereas a very large number of diverse activities originate from a very limited number of sources. In term of attacking machines' OS, according to p0f, almost all attacking machines are Windows ones. This confirms again the results in [11, 12]. Figure 5.3 shows the top ten attacking countries with US in the head, followed by China and Canada. But the surprising thing is that CS (corresponding to former Serbia and Montenegro) is at the fifth position. The reason is that there is one (and only one!) platform which is heavily attacked by this country. In total, it shows up as one of the most attacking countries. Finally, as an example to show the diversity of the attacks over different platforms, Figure 5.2 (right) shows the distribution of the number of different clusters per platform. Each column represents the number of distinct clusters observed on a platform. We have as many columns as number of platforms. As we can see, the attacks are highly diverse. On some platforms, we observe just small number of clusters, but it is not the case for others.

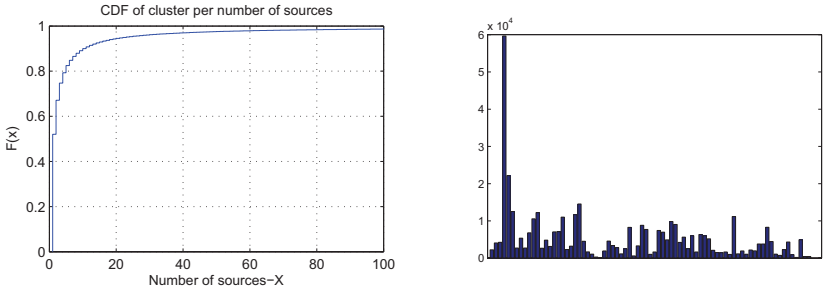


Fig. 5.2 Left:Cumulative distribution function of number of source per cluster; Right:Distribution of number of clusters per platform.

5.3.4 Some illustrative examples

The diversified aspect of real-world datasets, such as honeynet data, makes the task of an analyst rather difficult in selecting and analyzing some appropriate attack characteristics, which may help eventually to make meaningful conclusions about the attack root causes. To illustrate this point, we provide here a series of basic examples of how to analyze various facets of the observed network threats. At this stage, the main ideas we want to convey are: *i)* that several aspects of an attack dataset can potentially deliver meaningful pieces of evidence about attack root causes, and *ii)* that large-scale attack processes manifest themselves through so-called "attack events" on different sensors, which are the basis for the analysis of the underlying root causes.

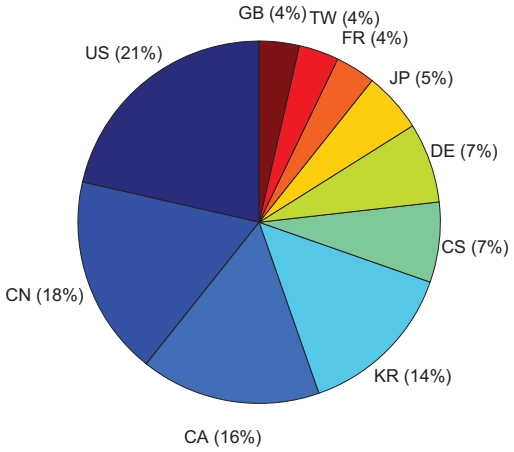


Fig. 5.3 Top ten attacking countries

5.3.4.1 Temporal Evolution of Attack Clusters

Time series analysis can provide valuable information (e.g., trends, abrupt changes, and emerging phenomena) to security practitioners in charge of detecting anomalous behaviors or intrusions in the collected traffic. This first illustration shows a temporal evolution of a given attack cluster, i.e.an aggregated source count of the number of sources belonging to that cluster on a chosen time scale, in this case grouped by day. On Fig 5.4, we can see the evolution of the attack cluster with ID. 17718 in a time period ranging from 1-Dec-06 until 01-Mar-07, either for all platforms together (left plot), or by splitting the time series for each platform separately, so as to analyze the impact of this attack cluster on different platforms.

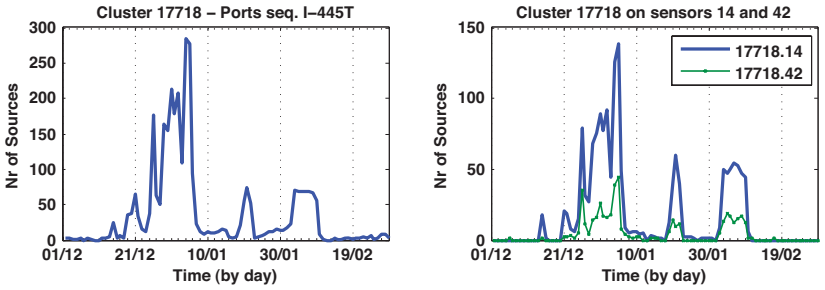


Fig. 5.4 Left: global time evolution of attack cluster 17718, with the sources aggregated by day. Right: time evolution of the same attack cluster for the platforms 14 and 42 separately.

5.3.4.2 Geographical Location of Attackers

Taking back the previous example, we could wonder from which countries the sources belonging to attack cluster 17718 are coming from during the activity period of this attack process. The geographical origins of attackers can be used indeed to identify attack activities having specific patterns in terms of originating countries. Such information can be important to identify, for instance, botnets that are located in a limited number of countries. It is also a way to confirm the existence, or not, of so-called safe harbors for the hackers.

Table 5.1 Geographical distribution for attack cluster 17718 in the time window spanning from 1-Dec-06 until 01-Mar-07.

Country of origin	Nr of Sources	Relative %
CN	1150	35.3
US	378	11.6
CA	255	7.8
FR	236	7.2
unknown	215	6.6
TW	137	4.2
JP	128	3.9
IT	120	3.6
DE	107	3.3
Others	524	16.1

The result of extracting the geographical distribution of cluster 17718 is represented in Table 5.1: the first column indicates the country of origin (represented with its ISO code) and the second column gives the number of sources belonging to that country. The third column indicates the corresponding relative percentage for each country with respect to the total number of sources for this attack process (i.e., 3250 sources in total). With this simple example, we want to show that this kind of aggregated information can in turn be used as input of a correlation process, as it will be demonstrated in Section 5.6.

5.3.4.3 Attackers Subnets Information

The source IP network blocks is a property that nicely complements the geolocation as described before. Instead of giving insight into possible geostrategic decisions made by the attackers, they can typically reveal some strategies in the propagation model of the malware. Indeed, attackers' IP subnets can provide a good indication of the spatial "uncleanliness" of certain networks, i.e., the tendency for compromised hosts to stay clustered within unclean networks, especially for zombie machines belonging to botnets as demonstrated in [8]. Previous studies have also demonstrated that some worms show a clear bias in their propagation scheme, such as a tendency for scanning machines of the same (or nearby) network so as to optimize their propagation [7].

The results of such analysis are presented in Table 5.2[1] in the case of an aggregation of the sources by Class A network blocks, but similar analyses could be performed for other groupings (Class B, C, ...). Again, this kind of feature vector can be used as input for a global correlation process in order to identify attack processes that exhibit similar IP subnets distributions.

[1] To preserve the confidentiality related to the IPs of the attackers, the first byte values have been somehow obfuscated in the table. So these are not the real subnet prefixes, but the eventual proximities among them have been preserved.

Table 5.2 Anonymized distribution of Class A-subnets for attack cluster 17718 in the time window spanning from 1-Dec-06 until 01-Mar-07.

Subnet of Origin (Class A)	Nr of Sources
220.x.x.x	451
56.x.x.x	193
80.x.x.x	168
22.x.x.x	160
217.x.x.x	159
86.x.x.x	123
218.x.x.x	113
69.x.x.x	100
66.x.x.x	91
216.x.x.x	90
Others	1602

5.3.4.4 Targeted Platforms or Subnets

Apparently, some recent crimeware toolkits are now able to deliver a specific type of malware to different geographical regions [5]. By using this new feature, cybercriminals can thus set up well targeted campaigns by delivering specialized crimeware in specific regions, being specific countries or its corresponding IP blocks. Therefore, it seems important to look at the relationships that may exist between attack events and the platforms or subnets they have been observed on. Table 5.3 illustrates this kind of information, where the first column gives the Id. of the platform, and each row of the second column indicates the number of sources belonging to attack cluster 17718 that have targeted the corresponding platform. In the last column, the Class A-subnet of each platform is also given. This last illustration gives yet another example of "viewpoint" that could be used in a global correlation process of attack events.

Table 5.3 Distribution of targeted platforms for attack cluster 17718

Targ.Platform	Nr of Sources	Subnet(A)
14	1552	139
76	871	134
42	431	150
57	70	24
71	67	58
53	42	88
55	42	83
Others	175	-

5.4 Leurre.com v2.0: SGNET

5.4.1 Increasing the level of interaction

We have seen in the previous Section how we have been able to generate valuable dataset with quantitative information on the localization and the evolution of Internet unsolicited traffic. We are able to observe interesting behaviors, most of which are very difficult to justify or to attribute to a specific root cause. It is, indeed, very difficult to link a given observation to a class of activities, and our search for answers in this direction had to deal with a limited amount of information about the final intention of the attacker. The low level of interaction of the Leurré.com honeypots is a limiting factor: when a honeypot receives a client request, it is not able to carry on the network conversation with the attacker, nor to "understand" it.

For instance, in our experience within the Leurré.com project, due to the lack of emulation scripts we have been able to observe only the first request of many interesting activities such as the spread of the Blaster worm [6]. But since Blaster sends its exploit in the second request of its dialog on port 135, we have never been able to observe such a payload. Therefore it becomes very difficult to distinguish Blaster's activity from other activities targeting the same port using solely the payload as a discriminating factor.

Fortunately, experience shows that, even such limited amount of information, a large variety of analyses remain applicable and deliver useful results. In order to increase the amount of available information on attackers, we need to increase the level of interaction with the honeypots. However, in order to keep carrying on our deployment of sensors on a voluntary basis, we need to achieve this objective at the lowest possible cost. This led to the development of the ScriptGen approach.

5.4.2 ScriptGen

The ScriptGen technology [19, 20] was created with the purpose of generating honeypots with a high level of interaction having a limited resource consumption. This is possible by *learning* the behavior of a given network protocol when facing deterministic attack tools. The learnt behavior is represented under the form of a Finite State Machine representing the protocol language. The generated FSM can then be used to respond to clients, emulating the behavior of the real service implementation at a very low cost.

The ScriptGen learning phase is completely protocol agnostic: no knowledge is assumed neither about the structure of the protocol, nor on its semantics. ScriptGen is thus able to replay any deterministic run of a protocol as long as its payload is not encrypted. The ScriptGen learning takes as input a set of samples of network interaction between a client and the real implementation of a server. The core of the learning phase is the Region Analysis algorithm introduced in [20]: taking advantage of bioinformatics alignment algorithms [24], the algorithm exploits the statistical variability of the samples to identify portions of the protocol stream likely to carry a strong semantic meaning and discard the others. In order to build reliable representations of the protocol interaction, it is thus necessary to collect a clean set of samples with enough statistical variability to correctly identify semantically important regions. Figure 5.5 shows an example of semantic abstraction for an excerpt of SMTP FSM.

The properties of the ScriptGen approach allow to perform a completely automated incremental learning of the activities as shown in [19]. ScriptGen-based honeypots are able to detect when a client request falls out of the current FSM knowledge (a 0-day attack or, more exactly, a yet unseen attack) by simply detecting the absence of a matching transition. In such case, the honeypot is thus unable to provide a valid answer to the attacker. We showed in [19] how the honeypot can react to this situation relying on a real host (an *oracle*) and acting as a proxy between the attacker and the

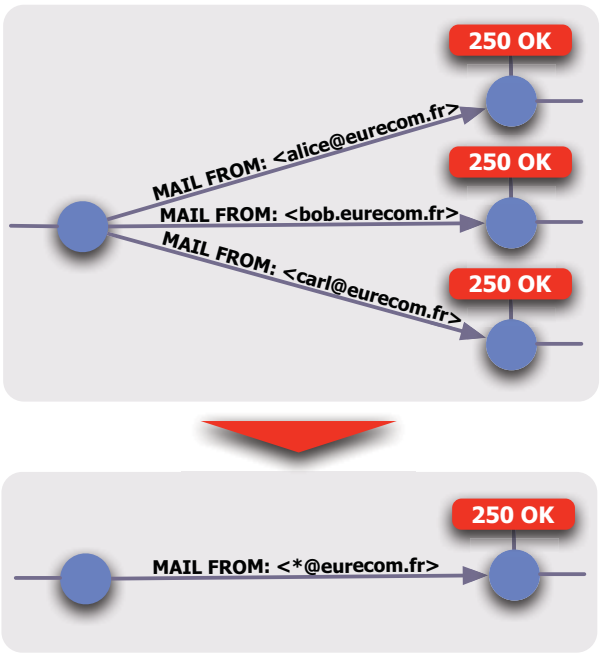


Fig. 5.5 ScriptGen FSM generalization

real host. This allows the honeypot to continue the conversation with the attacker, and to collect a new sample of protocol interaction that can be used to automatically refine the protocol knowledge.

ScriptGen is able to correctly learn and emulate the exploit phase for protocols as complex as NetBIOS [19]. ScriptGen thus allows to build highly interactive honeypots at low cost. The oracles needed to learn new activities can be hosted in a single virtualization farm and contacted by the honeypots through a tunneling system, in a structure similar to Spitzner's *honeyfarm* concept. Differently from classical honeyfarms, access to the real hosts is a rare event resulting from the occurrence of a new kind of attack. As a consequence, systems based on the ScriptGen honeypots potentially have a high degree of scalability.

5.4.3 SGNET: a ScriptGen-based honeypot deployment

We took advantage of this technology to build an experimental honeypot deployment, called SGNET, meant to follow the lines of the Leurré.com deployment but providing a significant improvement in the richness of the collected data.

SGNET and code injections. SGNET is a scalable framework that offers almost the same amount of information than real high interaction systems for a specific class of attacks, namely server-based code injection attacks generated by deterministic scripts. We are aware of the fact that they correspond only to a subset of the possible attack scenarios. However, as of today, they

are considered to be responsible for the creation of large botnets [32] and the preferred propagation mechanisms of a large number of different malware.

The final objective of a code injection attack consists in forcing the execution of executable code on a victim machine exploiting a vulnerable network service. Crandall et al. introduced in [10] the epsilon-gamma-pi model, to describe the content of a code-injection attack as being made of three parts.

Exploit (ε). A set of network bytes being mapped onto data which is used for conditional control flow decisions. This consists in the set of client requests that the attacker needs to perform to lead the vulnerable service to the control flow hijacking step.

Bogus control data (γ). A set of network bytes being mapped onto control data which hijacks the control flow trace and redirects it to someplace else.

Payload (π). A set of network bytes to which the attacker redirects the vulnerable application control flow through the usage of ε and γ.

The payload that can be embedded directly in the network conversation with the vulnerable service (commonly called shellcode) is usually limited to some hundreds of bytes, or even less. It is often difficult to code in this limited amount of space complex behaviors. For this reason it is normally used to force the victim to download from a remote location a larger amount of data: the malware. We extend the original epsilon-gamma-pi model in order to differentiate the shellcode π from the downloaded malware μ.

An attack can be characterized as a tuple $(\varepsilon, \gamma, \pi, \mu)$. In the case of, old, classical worms, it is possible to identify a correlation between the observed exploit, the corresponding injected payload and the uploaded malware (the self-replicating worm itself). Thanks to the correlation between the 4 parameters, retrieving information about a subset of them was enough to characterize and uniquely identify the attack. This situation is changing. Taking advantage of the many freely available tools such as Metasploit [33, 37], even unexperienced users can easily generate shellcodes with personalized behavior and reuse existing exploit code. This allows them to generate new combinations along all the four dimensions, weakening the correlation between them. It is thus important to try to retrieve as much information as possible on all the 4 dimensions of the code injection. We designed SGNET in such a way to delegate to different functional components the 4 dimensions, and combine the information retrieved by these components to have an exact picture of the relationships among them.

The ScriptGen approach is suitable for the learning of the exploit network interaction ε, offering the required level of interactivity with the client required to lead the attacker into sending code injection attacks. For the previously stated reasons, in SGNET we extend this capability with the information provided by other tools in order to retrieve information on the other dimensions of the epsilon-gamma-pi-mu (EGPM) model. We take advantage of the control flow hijack detection capabilities of Argos [28] to detect successful code injection attacks, understand the bogus control data γ and retrieve information about the location of the injected payload π. We take advantage of the shellcode emulation and malware download capabilities of Nepenthes [2] to understand the payload π, emulate its behavior and download the malware sample μ.

When facing an attacker, the SGNET activity evolves through different stages, corresponding to the main phases of a network attack. SGNET distributes these phases to three different functional entities: *sensor*, *sample factory* and *shellcode handler*.

The SGNET sensor corresponds to the interface of the SGNET towards the network. The SGNET deployment aims at monitoring small sets of IPs deployed in multiple locations of the IP space, in order to characterize the heterogeneity of the activities along the Internet as observed in [9, 12]. SGNET sensors are thus low-end hosts meant to be deployed at low cost by different partners willing to join the project and bound to a limited number of IPs. The deployment of the sensors follows the same win-win partnership schema explained before. Taking advantage of the

ScriptGen technology, the sensors are able to handle autonomously the exploit phase ε of attacks falling inside the FSM knowledge with minimal resource requirements on the host.

The SGNET sample factory is the *oracle* entity meant to provide samples of network interaction to refine the knowledge of the exploit phase when facing unknown activities. The sample factory takes advantage of a real host running on a virtual machine and monitors the host state through memory tainting. This is implemented taking advantage of *Argos*, presented by Portokalidis et al. in [28]. Keeping track of the memory locations whose content derives from packets coming from the network, Argos is able to detect the moment in which this data is used in an *illegal* way. Argos was modified in order to allow the integration in the SGNET and load on demand a given honeypot profile with a suitable network configuration (same IP address, gateway, DNS servers, ... as of the sensor sending the request). The profile loading and configuration is fast enough to be instantiated on the fly upon request of a sensor.

The Argos-based sample factories provide information about the presence of code injections (γ) and are able to track down the position in the network stream of the first byte being executed by the guest host, corresponding to the byte B_i of the payload π. We have developed a simple heuristic to identify the injected payload π in the network stream starting from the *hint* given by the sample factory [17]. This allows to embed in the ScriptGen learning additional knowledge, namely the a tag identifying the final state of a successful code injection and information within the preceding transitions that allows to extract from the attacker's protocol stream the payload π.

The final steps of the code injection attack trace are delegated to the SGNET shellcode handler. Every payload π identified by the SGNET interaction is submitted to a shellcode handler. The shellcode handler is implemented reusing part of the functionality of the Nepenthes [2] honeypots. We take advantage of Nepenthes shellcode analyzer to "understand" the payload π and emulate its behavior using Nepenthes download modules. In the context of the SGNET, Nepenthes is thus used as an *oracle* for the payload emulation. Differently from the exploit phase, we do not try to learn the Nepenthes behavior in terms of FSM. We consider the payload emulation a too complex interaction to be represented in terms of a FSM.

SGNET Architecture.

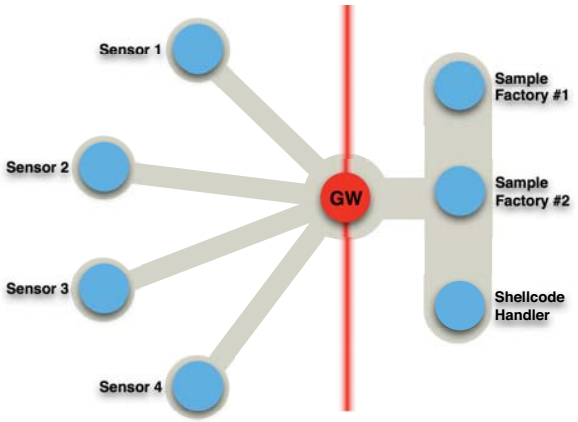


Fig. 5.6 SGNET architecture

The general architecture of the SGNET is presented in Figure 5.6. All the SGNET entities communicate through an ad-hoc HTTP like protocol called Peiros [18]. The Peiros protocol allows

communication under the form of a set of service requests, allowing for instance a sensor to require the instantiation of a sample factory. The sensors, distributed over the IP space and hosted by partners of the project, are connected to a central entity called SGNET gateway, that acts as an application-level proxy for the Peiros protocol. The gateway receives service requests from the sensors and dispatches them to a free internal entity, performing a very simple load balancing. The architecture offers a clean separation between the sensors, relatively simple daemons running over inexpensive hosts, and the internal entities, having a higher complexity and higher resource requirement.

We saw how the ScriptGen learning exploits the variability of the samples to produce "good" refinements of the FSM knowledge. The architecture of Figure 5.6 shows how the SGNET gateway offers a unique standpoint to collect interaction samples: all the tunneled conversations between any sensor and any sample factory flow through the gateway. The gateway becomes thus the best candidate to perform ScriptGen refinements to the current FSM knowledge. Once a new refinement is produced, the gateway takes care of updating the knowledge of all the sensors pushing them the FSM updates. This makes sure that all the sensors online at a given moment share exactly the same knowledge of the protocols.

An important aspect related to the ScriptGen learning is the strict relation between the ScriptGen ability to learn exploits and the configuration of the sample factories. If a service is not installed or activated in the configuration of the virtualized host handled by the sample factory, the SGNET architecture will not be able to observe activities targeting it. It is thus important to carefully configure the sample factories in order to maximize the visibility of malicious activities. We chose to address this problem supporting the assignment of different profiles for the IPs of the SGNET sensors, similarly to what was done on the Leurré.com deployment. Each profile is assigned to a different sample factory configuration, with different services and different OS versions to maximize the visibility on network attacks of our deployment.

The description of the SGNET deployment clearly shows a difference with respect to the original Leurré.com deployment. SGNET is a more complex architecture, that succeeds in raising the level of interaction of the honeypots without raising the resource requirements for the partners hosting the sensors. Taking advantage of the ScriptGen learning, the deployment also allows to minimize the usage of expensive resources such as the sample factories, that are needed only to handle those activities that do not fall *yet* in the FSM knowledge. An important concern for the partner taking advantage of this deployment is the security of the solution. SGNET raises the level of interaction of the honeypots; it is thus important to guarantee that the increased interactivity does not impact the safety of hosting a honeypot platform. The network interaction driven by FSM knowledge is virtually as safe as any low-interaction honeypots: the attacker interacts with a simple daemon performing state machine traversals to provide answers to client requests. When a new activity is handled, the sensor acts as a proxy and the attacker is allowed to interact with a real (and thus vulnerable) host. Two measures are in place to ensure the safety of this process. Firstly, the tunneling system ensures that any outbound packet generated by the sample factory is directed only towards the attacking source (blocking any attempt of exploiting the honeypot as a stepping stone to attack others). Secondly, the memory tainting capabilities of Argos allow us to stop execution as soon as the attacker successfully hijacks the host control flow. This does not include for instance successful password brute-forcing attacks, but this class of attacks can be prevented by a careful configuration of the virtualized host.

5.5 Analysis of Attack Events

5.5.1 Identification of Attack Events

5.5.1.1 Attack Event Definition

An attack event is defined as a set of observed cluster time series exhibiting a particular shape during a limited time interval. This time interval typically lasts a couple of days, but it can also be as short as a single day.

The existence of attack events highlights the coordinated activities of several attacking machines. Note that the set can be a singleton. This is typically the case when the set is a peak of activities on a single day. For illustrative purposes, the top plot of Figure 5.7 represents the attack event 225 which consists of cluster 60332 (targeting port 5900 TCP) attacking seven platforms 5,8, 11, ...,31 from day 393 to day 400. Whereas the bottom plot of Figure 5.7 represents the attack event 14 which consists of activities of cluster 0 on day 307 coming almost only from Spain.

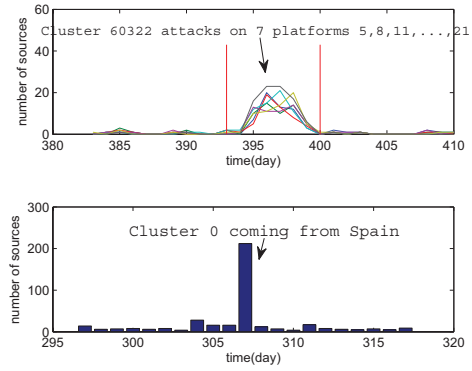


Fig. 5.7 On the top plot, cluster 60232 attacks seven platforms from day 393 to day 400. On the bottom plot, peak of activities of cluster 0 from Spain on day 307.

5.5.1.2 Dataset Description

In order to have a clean dataset for our experiments, we have selected the traces observed on 40 platforms out of the 50 that we had at our disposal. All these 40 platforms have been running for more than 800 days. During this period, note of the platforms has been down for more than 10 times. Furthermore, each one has been up continuously for at least 100 days. All platforms have been up for a minimum of 400 days over that period.

The total amount of sources observed, day by day, on all these 40 platforms can be denoted by the initial time series TS over a period of 800 days.

We can split that time series per country[2] of origin of the sources. This gives us 231 time series TS_X where the i^{th} point of such time series indicates the amount of sources, observed on all platforms, located in country X. We represent by TS_L1 the set of all these Level 1 time series. To reduce the computational cost, we keep only the countries from which we have seen at least 10 sources on at least one day. This enables us to focus on 85 (the set of corresponding countries is called $big_{countries}$), instead of 231, time series. We represent by TS_L1' this refined set of Level 1 time series. Then, we split each of these time series by cluster to produce the final set of time series $\Phi_{[0-800),c_i,country_j} \forall c_i$ and $\forall country_j \in big_{countries}$. The i^{th} point of the time series $\Phi_{[0-800),X,Y}$ indicates the amount of sources originating from country Y that have been observed on day i attacking any of our platforms thanks to the attack defined by means of the cluster X. We represent by TS_L2 the set of all these Level 2 time series. In this case $|TS\_L2|$ is equal to 436,756 which corresponds to 3,284,551 sources. As explained in [27], time series that barely vary in amplitude over the 800 days are meaningless to identify attack events and we can get rid of them. Therefore, we only keep the time series that highlight important variations during the 800 days period. We represent by TS_L2' this refined set of Level 2 time series. In this case $|TS\_L2'|$ is equal to 2,420 which corresponds to 2,330,244 sources. We have done the very same splitting and filtering by looking at the traces on a per platform basis instead of on a per country of origin basis. The corresponding results are given in Table 5.4.

Table 5.4 dataset description: TS: *all sources observed on the period under study*, OVP: *observation view point*, TS_L1: *set of time series at country/platform level*, TS_L1': *set of significant time series in* TS_L1, TS_L2 : *set of all cluster time series*, TS_L2' *set of strongly varying cluster time series*

TS consists of 3,477,976 sources		
OVP	country	platform
$\|TS\_L1\|$	231	40
$\|TS\_L1'\|$	85 (94,4% TS)	40 (100% TS)
$\|TS\_L2\|$	436,756	395,712
$\|TS\_L2'\|$	2,420	2,127
sources	2,330,244 (67% of TS)	2,538,922 (73% of TS)

5.5.1.3 Results on Attack Event Detection

We have applied the techniques presented in [26] to identify the attack events existing in our 2 distinct datasets, namely $TS_{country}$ and $TS_{platform}$. For the time series in $TS_{country}$ (resp. $TS_{platform}$), we have found 592 (resp. 690) attack events which correspond to 574,125 (resp. 578,372) sources. The results are given in Table 5.5

[2] The geographical location is given to us thanks to the Maxmind product, based on the IP address. However, some IPs can not be mapped to any real country and are attached to labels not corresponding to any country, e.g. EU,A1,..

Table 5.5 Result on Attack Event Detection

AE-set-I($TS_{country}$)		AE-set-II($TS_{platform}$)	
No.AEs	No.sources	No.AEs	No.sources
592	574,125	690	578,372

No.AEs: amount of attack events

5.5.2 Armies of Zombies

So far, we have identified what we have called attack events which highlight the existence of coordinated attacks launched by a group of compromised machines, i.e. a zombie army. It would be interesting to see if the very same army manifests itself in more than one attack event. To do this, we propose to compute what we call the *action sets*. An *action set* is a set of attack events that are likely due to same army. In this Section, we show how to build these action sets and what information we can derive from them regarding the size and the lifetime of the zombie armies.

5.5.2.1 Identification of the armies

Similarity Measures: In its simplest form, a zombie army is a classical botnet. It can also be made of several botnets. That is, several groups of machines listening to a distinct *C&C*. This is invisible to us and irrelevant. All that matters is that all the machines do act in a coordinated way. As time passes, it is reasonable to expect members of an army to be cured while others join. Hence, if the same army attacks our honeypots twice over distinct periods of time, one simple way to link the two attack events together is by observing that they have a large amount of IP addresses in common. More formally, we measure the likelihood of two attacks events e_1 and e_2 to be linked to the same zombie army by means of their similarity defined as follows:

$$sim(e_1,e_2) = \begin{cases} max(\frac{|e_1 \cap e_2|}{|e_1|}, \frac{|e_1 \cap e_2|}{|e_2|}) & \text{if } |e_1 \cap e_2| < 200 \\ 1 & \text{otherwise} \end{cases}$$

In which, $|e_1|$ (resp. $|e_2|$) represents the number of distinct IP addresses of attack event e_1 (resp. e_2) and $|e_1 \cap e_2|$ represents the number of IP addresses in common of attack events e_1 and e_2. We conclude that e_1 and e_2 are caused by the same zombie army if and only if $sim(e_1,e_2) > 10\%$. Called P_{e_1,e_2} is the probability that two attack events e_1 and e_2 share n IP addresses in common by chance. We also verify that $|e_1 \cap e_2| > n$, in which the corresponding $P_{e_1,e_2} <= 10^{-9}$.

Action Sets: We now use the $sim()$ function to group together attack events into action sets. To do so, we build a simple graph where the nodes are the attack events. There is an arc between two nodes e_1 and e_2 if and only if $sim(e_1,e_2) > \delta$. All nodes that are connected by at least one path end up in the same action set. In other words, we have as many action sets as we have disconnected graphs made of at least two nodes; singleton sets are not counted as action sets.

We note that our approach is such that we can have an action set made of three attack events e_1, e_2 and e_3 where $sim(e_1,e_2) > \delta$ and $sim(e_2,e_3) > \delta$ but where $sim(e_1,e_3) < \delta$. This is consistent with our intuition that armies can evolve over time in such a way that the machines present in the army can, eventually, be very different from the ones found the first time we have seen the same army in action.

Results: we have identified 40 (resp. 33) zombie armies from AE-set-I (resp. AE-set-II) which have issued a total of 193 (resp. 247) attack events. Figure 5.8 (Left) represents the distribution of attack events per zombie army. Its top (resp. bottom) plot represents the distribution obtained from AE-set-I(resp. AE-set-II). We can see that the largest amount of attack events for an army is 53 (resp. 47) whereas 28 (resp. 20) armies have been observed only two times.

5.5.2.2 Main Characteristics of the Zombie armies

Lifetime of Zombie Army.

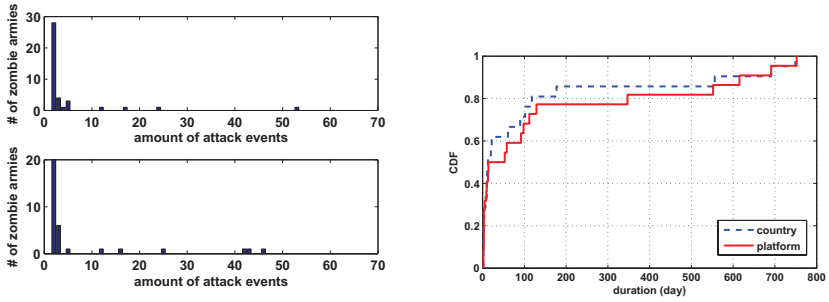


Fig. 5.8 Left: Zombie Army Size. Right: Cumulative Distribution Function (CDF) of the durations of zombie armies.

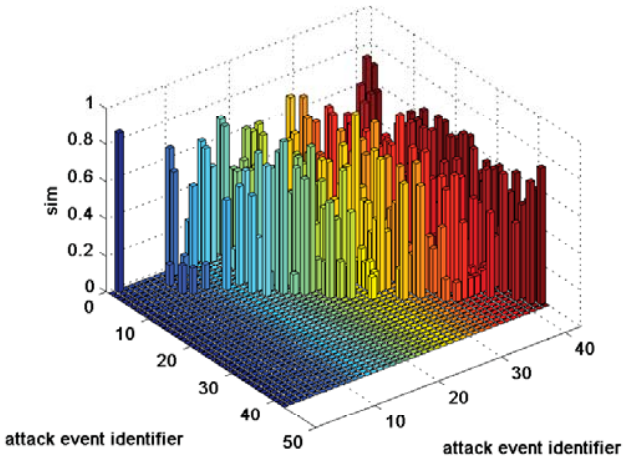
Figure 5.8 (Right) represents the cumulative distribution of minimum lifetime of zombie armies obtained from $TS_{platform}$ and $TS_{country}$ (see Section 5.5.1). According to the plot, around 20% of zombie armies have existed for more than 200 days. In the extreme case, two armies seems to have survived for 700 days! Such result seems to indicate that either *i)* it takes a long time to cure compromised machines or that *ii)* armies are able to stay active for long periods of time, despite the fact that some of their members disappear, by continuously compromising new ones.

Lifetime of Infected Host in Zombie Armies

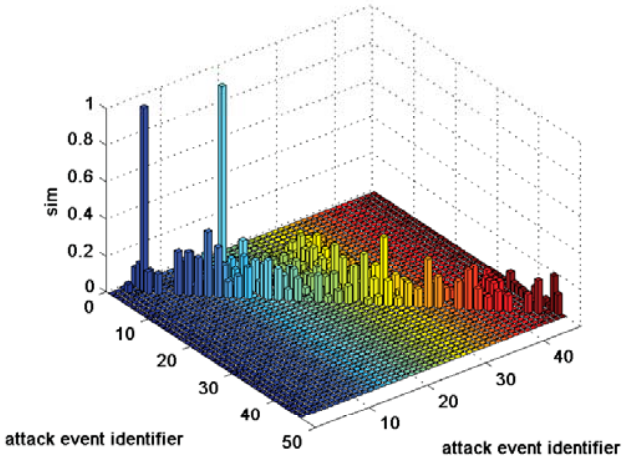
We can classify the armies into two classes as mentioned in the previous Section. For instance, Figure 5.9a represents the similarity matrix of zombie army 33, ZA33. To build this matrix, we first order its 42 attack events according to their occurred time. Then, we represent their similarity relation under an 42×42 similarity matrix $\mathscr{M}$. The cell (i,j) represents the value of $sim()$ of the ordered attack event i^{th} and j^{th}. Since, $\mathscr{M}$ is a symmetric matrix, for the visibility, we represent only half of it.

As one can see, we have a very high similarity measure between almost all the attacks events (i.e., around 60%). This is also true between the very first and the very last attack events. It is important to notice the time interval between the first and the last activities observed from this army is 753 days!

Figure 5.9b represents an opposite case, the zombie army 31, ZA31, consisting of 46 attack events. We proceed as above to build its similarity matrix. One can see that the important values are surrounded around the main diagonal of $\mathscr{M}$. It means that the attack event i^{th} has the same subset of infected machines with only few attack events happening not far from it in terms of time. Another important point to be noticed is that this army changes its attack vectors over time. In fact, it moves from attack against 4662 TCP, to 1025 TCP, then 5900 TCP, 1443 TCP, 2967 TCP, 445 TCP,...And the lifetime of this army is 563 days! It is clear, from these two cases, that the composition of armies evolves over time in different ways. More work remains to be done in order to understand the reasons behind these various strategies.



(a)



(b)

Fig. 5.9 Renewal rate of zombie armies

5.5.3 Impact of Observation View Point

5.5.3.1 Analysis

Table 5.5 highlights the fact that depending on how we decompose the initial set of traces of attacks (i.e the initial time series TS), namely by splitting it by countries of origin of the attackers or by platforms attacked, different attacks events show up. To assess the overlap between attack events detected from different observation view points, we use the *common source ratio, namely csr*, measure as follows:

$$csr(e, AE_{op'}) = \frac{\sum_{\forall e' \in AE_{op'}} |e \cap e'|}{|e|}$$

in which $e \in AE_{op}$ and $|e|$ is the amount of sources in attack event e, AE_{op} is *AE-set-I* and $AE_{op'}$ is *AE-set-II* (or vice versa).

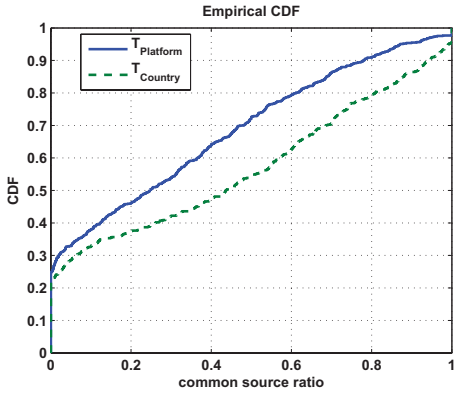


Fig. 5.10 CDF common source ratio.

Figure 5.10 represents the two cumulative distribution functions corresponding to this measure. The point (x, y) on the curve means that there are $y * 100\%$ of attack events obtained thanks to $T_{country}$ (resp $T_{platforms}$) that have less than $x * 100\%$ of sources in common with all attack events obtained thanks to $T_{platforms}$ (resp $T_{country}$). The $T_{country}$ curve represents the cumulative distribution obtained in this first case and the $T_{platforms}$ one represents the CDF obtained when starting from the attacks events obtained with the intial $T_{platforms}$ set of time series. As we can notice, around 23% (resp. 25%) of attack events obtained by starting from the $T_{country}$ (resp. $T_{platform}$) set of time series do not share any sources in common with any attack events obtained when starting the attack even identification process from the $T_{platform}$ (resp. $T_{country}$) set of time series. This corresponds to 136 (16,919 sources) and 171 (75,920 sources) attack events not being detected. In total, there are 288,825 (resp. 293,132) sources present in AE-Set-I (resp. AE-Set-II), but not in AE-Set-II (resp. AE-Set-I).

5.5.3.2 Explanation

There are good reasons why we can not rely on a single viewpoint to detect all attacks events. We elaborate on these reasons in the following discussion.

Split by country: Suppose we have one botnet B made of machines that are located within the set of countries $\{X,Y,Z\}$. Suppose that, from time to time, these machines attack our platforms leaving traces that are also assigned to a cluster C. Suppose also that this cluster C is a very *popular* one, that is, many other machines from all over the world continuously leave traces on our platforms that are assigned to this cluster. As a result, the activities specifically linked to the botnet B are lost in the noise of all other machines leaving traces belonging to C. This is certainly true for the cluster time series (as defined earlier) related to C and this can also be true for the time series obtained by splitting it by platform, $\Phi_{[0-800),C,platform_i} \forall platform_i \in 1..40$.However, by splitting the time series corresponding to cluster C by countries of origins of the sources, then it is quite likely that the time series $\Phi_{[0-800),C,country_i} \forall country_i \in \{X,Y,Z\}$ will be highly correlated during the periods in which the botnet present in these countries will be active against our platforms. This will lead to the identification of one or several attack events.

Split by platform: Similarly, suppose we have a botnet B' made of machines located all over the world. Suppose that, from time to time, these machines attack a specific set of platforms $\{X,Y,Z\}$ leaving traces that are assigned to a cluster C. Suppose also that this cluster C is a very *popular* one, that is, many other machines from all over the world continuously leave traces on all our platforms that are assigned to this cluster. As a result, the activities specifically linked to the botnet B' are lost in the noise of all other machines leaving traces belonging to C. This is certainly true for the cluster time series (as defined earlier) related to C and this can also be true for the time series obtained by splitting it by countries, $\Phi_{[0-800),C,country_i} \forall country_i \in big_{countries}$. However, by splitting the time series corresponding to cluster C by platforms attacked, then it is quite likely that the time series $\Phi_{[0-800),C,platform_i} \forall platform_i \in \{X,Y,Z\}$ will be highly correlated during the periods in which the botnet influences the traces left on the sole platforms concerned by its attack. This will lead to the identification of one or several attack events.

The top plot of Figure 5.11 represents the attack event 79. In this case, we see that the traces due to the cluster 175309 are highly correlated when we group them by platform attacked. In fact, there are 9 platforms involved in this case, accounting for a total of 870 sources. If we group the same set of traces by country of origin of the sources, we end up with the bottom curves of Figure 5.11 where the specific attack event identified previously can barely be seen. This highlights the existence of a botnet made of machines located all over the world that target a specific subset of the Internet.

5.6 Multi-Dimensional Analysis of Attack Events

5.6.1 Methodology

Analogous to criminal forensics, the security analyst needs to synthesize different pieces of evidence in order to investigate the root causes of global attack phenomena on the Internet. This task can be a tedious, lengthy and informal process mostly relying on the analyst's expertise, and involving many different dimensions characteristics of attack events. For those reasons, we seek to develop a multi-dimensional knowledge discovery and data mining methodology that should help us to improve, in a more systematic way, our understandings of emerging Internet threats, so as to achieve a better cyber situational awareness.

Our idea consists of *i)* extracting relevant nuggets of knowledge by mining a dataset of attack events according to different relevant characteristics; and in *ii)* combining systematically those pieces of knowledge so as to create higher-level concepts able to explain more clearly the underly-

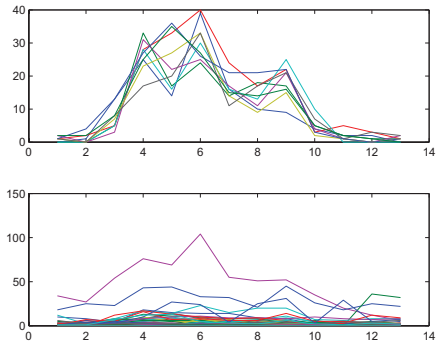


Fig. 5.11 Top plot represents the attack event 79 related to cluster 17309 on 9 platforms. The bottom plot represents the evolution of this cluster by country. Noise of the attacks to other platforms decrease significantly the correlation of observed cluster time series when split by country.

ing phenomena that might be the root cause of the suspicious traffic. Each step is further explained in the next sections.

5.6.2 Clique-based Clustering

5.6.2.1 Principles

The first component of our knowledge mining methodology involves an unsupervised graph-theoretic correlation process which aims at grouping similar "events" (through their corresponding feature vectors) in a reliable and consistent manner.

Typical clustering tasks involve the following steps [16]: *i)* feature selection and/or extraction, and pattern representation; *ii)* definition of a similarity measure between pairs of patterns; *iii)* grouping similar patterns; *iv)* data abstraction (if needed), to provide a compact representation of each cluster; and *v)* the assessment of the clusters quality and coherence (also when needed).

In any clustering, we must select salient features that may provide meaningful *patterns* from the data (e.g., from attack events in our case). Those patterns are represented with *feature vectors*, which are for instance the geographical distributions, subnet distributions, attack time series, etc.

In the second step, we need to measure the similarity between two such defined patterns or input vectors. For that purpose, several types of similarity distances are available (e.g., Pearson correlations, Minkowski, Jackknife, etc.). Clearly, the choice of a similarity metric has an impact on the properties of the clusters, such as their size, quality, or consistency. To reliably compare the empirical distributions mentioned here above, we rely on strong statistical distances that are based on non-parametric statistical tests, such as Pearson's χ^2 and Kolmogorov-Smirnov, whose resulting *p-value* is then validated against the Kullback-Leibler divergence. Those methods are amongst the most commonly used ones to determine whether two underlying one-dimensional probability distributions differ in a significant way.

Finally, we take advantage of those similarity measures to group all pattack events whose patterns look very similar. We simply use an unsupervised graph-theoretic approach to formulate the problem: the vertices of the graph represent the patterns (or feature vectors) of each attack event,

and the edges (or the arcs) express the similarity relationships between those vertices. Then, the clustering is performed by extracting so-called *maximal weighted cliques* (MWC) from the graph, where a maximal *clique* is defined as an induced sub-graph in which the vertices are fully connected and it is not contained within any other clique. Since it is a NP-hard problem [4], several approximate algorithms for solving the MWC problem have been developed. We refer the interested reader to [38, 39] for a more detailed description of this clique-based clustering technique applied to honeynet traces.

5.6.2.2 Some Experimental Results.

We applied our clique-based clustering on a honeynet dataset made of 351 attack events comprising 282,363 IP sources, which were collected on the Internet in a period spanning from Sep 2006 until June 2008. These events were observed on 36 platforms located in 20 different subnets, and belonging to 18 different class A-subnets. In terms of network activities, all sources could be classified in no more than 136 attack clusters

Table 5.6 presents a high-level overview of the cliques obtained for each attack dimension separately. As one can see, a relatively high volume of sources could be classified into cliques in each dimension. The high proportion of correlated sources with respect to the attack time series suggests that a majority of the attack events collected in this dataset were actually coordinated, or at least synchronized, on different honeypots. Among the targeted port sequences, we can observe some commonly targeted ports (e.g., Windows ports used for SMB or RPC, or SQL and VNC ports), but also a large number of uncommon high TCP ports that are normally unused on standard (and clean) machines. A non-negligeable volume of sources is also due to UDP spammers targeting Windows Messenger popup service (ports 1026-1028 UDP).

Table 5.6 Some experimental clique results obtained from a one year-honeynet dataset

Attack Dimension	Nr of Cliques	Volume of sources (%)	Most targeted port sequences
Geolocation	45	66.4	1027U, I, 1433T, 1026U, I445T, 5900T, 1028U, 9763T, I445T80T, 15264T, 29188T, 6134T, 6769T, 1755T, 64264T, 1028U1027U1026U, 32878T, 64783T, 4152T, 25083T, 9661T, 25618T, ...
IP Subnets (Class A)	30	56.0	1027U, I, 1433T, 1026U, I445T, 5900T, 1028U, 9763T, 15264T, 29188T, 6134T, 6769T, 1755T, 50656T, 64264T, 1028U1027U1026U, 32878T, 64783T, 18462T, 4152T, 25083T, 9661T, 25618T, 7690T, ...
Targeted platforms	17	70.1	I, 1433T, I445T, 1025T, 5900T, 1026U, I445T139T445T139T445T, 4662T, 9763T, 1008T, 6211T, I445T80T, 15264T, 29188T, 12293T, 33018T, 6134T, 6769T, 1755T, 2968T, 26912T, 50656T, 64264T, 32878T, ...
Attack time series	82	92.2	135T, I, 1433T, I445T, 5900T, 1026U, I445T139T445T139T445T, I445T80T, 6769T, 1028U1027U1026U, 50286T, 2967T, ...

5.6.2.3 Visualizing Cliques of Attackers.

In order to assess the consistency of the resulting cliques of attack events, it can be useful to see them charted on a two-dimensional map so as to *i)* verify the proximities among clique members (intra-clique consistency), and *ii)* to understand potential relationships between *different* cliques that are somehow related (i.e. inter-clique relationships). Note that a clique can be considered as a stricter definition of a cluster. Moreover, the statistical distances used to compute those cliques make them intrinsically coherent, which means that certain cliques of events may be somehow related to each other, although they were separated by the clique algorithm.

Since most of the feature vectors we are dealing with have a high number of variables (e.g., a geographical vector has more than 200 country variables), obviously the structure of such high-dimensional data set cannot be displayed directly on a 2D map. Multidimensional scaling (MDS) is a set of methods that can help to address this type of problem. MDS is based on dimensionality reduction techniques, which aim at converting a high-dimensional dataset into a two or three-dimensional representation that can be displayed, for example, in a scatter plot. As a result, MDS allows an analyst to visualize how near observations are to each other for many kinds of distance or dissimilarity measures, which in turn can deliver insights into the underlying structure of the high-dimensional dataset.

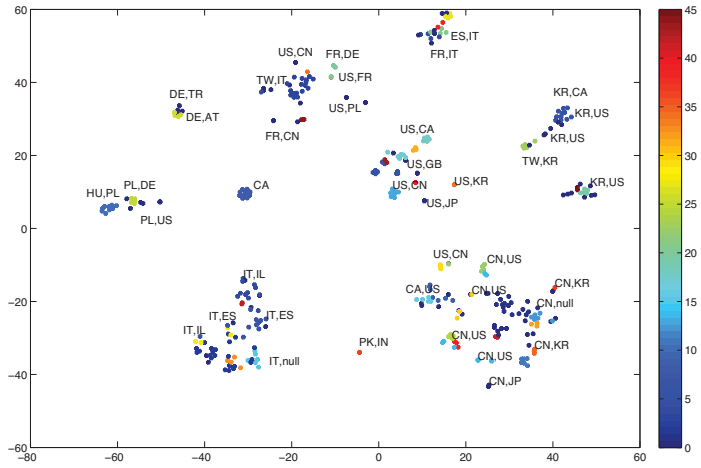


Fig. 5.12 Visualization of geographical cliques of attackers. The coloring refers to the different clique Id's. The superposed text labels show the two top attacking countries for some of the data points.

Because of the intrinsic non-linearity of real-world datasets, and the induced feature vectors, we applied a recent MDS technique called *t-SNE* to visualize each dimension of the dataset and to assess the consistency of the cliques results. t-SNE [40] is a variation of *Stochastic Neighbour Embedding* (SNE); it produces significantly better visualizations than other MDS techniques by reducing the tendency to crowd points together in the centre of the map. Moreover, this technique has proven to perform better in retaining both the local and global structure of real, high-dimensional data in a single map, in comparison to other non linear dimensionality reduction techniques such as Sammon mapping, Isomaps or Laplacian Eigenmaps.

Figure 5.12 shows the resulting two-dimensional plot obtained by mapping the geographical vectors on a 2D map using t-SNE. Each datapoint on the map represents the geographical vector of given attack event, and its coloring refers to its clique membership as obtained previously by applying the clique-based clustering. It can be easily verified that two adjacent events on this map have highly similar geographical distributions (even from a statistical viewpoint), while two distant events have clearly nothing in common in terms of originating countries. Quite surprisingly, the resulting mapping is far from being chaotic; it presents a relatively sparse structure with clear datapoint groupings, which means also that most of those attack events present very tight relationships

regarding their origins. Due to the strict statistical distances used to calculate cliques, this kind of correlation can hardly be obtained by chance only.

Similar "semantic mapping" can naturally be obtained for other dimensions considered (e.g., subnets, platforms, etc), so as to help assessing the quality of other cliques of attackers. To conclude this illustration, on Figure 5.13 we have indicated also some of the port sequences for several geographical cliques of attackers. This can help to visualize unobvious relationships among different types of activities and their origins.

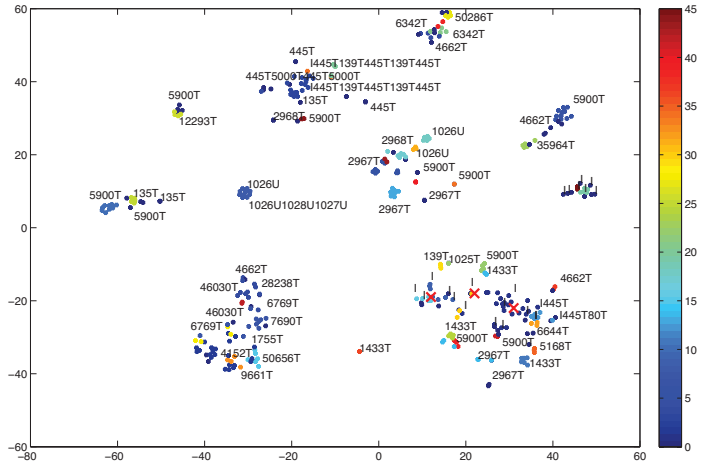


Fig. 5.13 Same visualization of the geographical cliques of attackers as Fig 5.12, but here the superposed text labels indicate the port sequences made by the attackers.

5.6.3 Combining Cliques of Attackers

The second component of our methodology is similar to a dynamic data fusion process. Starting from all sets of cliques, the idea is to combine k sets out of the N attack dimensions, with $k = 2, ..., N$, in order to discover higher-level knowledge about certain phenomena and their root cause (e.g., a set of attack events belonging to a same botnet or worm family, the evolution of a botnet IP location, etc). As such, each clique pattern can hold a piece of interesting knowledge about an attack phenomenon; but in many cases the security analyst will have to synthesize different pieces of evidence in order to perform a root cause analysis, and to really understand what happened. Therefore, we can take advantage of all one-dimensional cliques to construct higher-level concepts by simply combining different sets of cliques. Based on the phenomenon under scrutiny, the practitioner may include any number of dimensions in order to create "concepts" containing more or less semantics. In practice, we observe that the number of concepts obtained by combining any number of dimensions is not excessive. So, while the analysis of raw network traces (composed of thousands of packets) on each sensor would definitively be impractical, now the analysis of a

limited number of combined concepts can provide a better insight into the real-world phenomena that have caused the attack traffic.

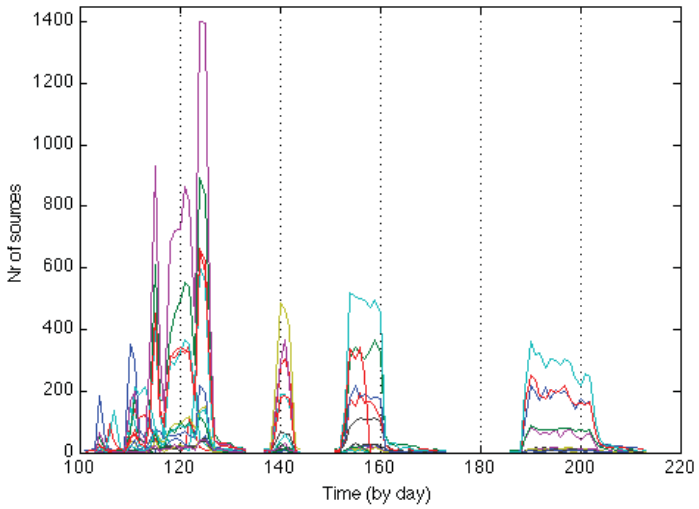


Fig. 5.14 Time series (i.e., nr of distinct sources by day) of a large scale phenomenon related to a botnet activity, made of 67 attack events observed from Dec 06 until April 07.

Illustration of Multi-dimensional analysis.
When we apply the multi-dimensional analysis on the examples given in 5.3.4, now we find out that those attack events were actually involved in a large-scale phenomenon assumed to be related to a botnet activity. This phenomenon has been active in a time period spanning from 1-Dec-06 until 31-Mar-07, during which we observed about 67 attack events that can be grouped into 4 distinct waves (see Fig 5.14) thanks to the temporal dimension. The platform correlations indicate that all attack events have hit exactly the same set of platforms (mostly in Belgium and in UK). Regarding the origins of the attacks (e.g., countries and subnets of origin), our method can clearly highlight two communities of attackers: one large group of "scanners" (performing almost only ICMP scanning on all honeypot IP's), and one smaller group of "attackers" (performing ICMP followed by attacks on Windows ports 445T or 139T). Interestingly, the attackers seem to "know" which honeypots are emulating a Windows machine, as they hit almost exclusively those IP addresses. The last finding deals with the dynamic evolution of the botnet population (in terms of IP blocks) between each botnet attack wave, which can be observed from the mapping of the different cliques of attackers. The scanner community has indeed been split into a few different cliques; but when looking at the geographical mapping (Fig 5.13 - see the regions indicated by the three red crosses in the lower-right part of the map), we can observe that those cliques appear in the same neighborhood.

5.7 Beyond Events Correlation: Exploring the epsilon-gamma-pi-mu space

In the previous Sections we took advantage of correlation techniques to analyze and correlate the available information to infer meaningful facts on the identity and on the behavior of the clients responsible for the observed events. We have left out until now the analysis of the *effects* of these activities on the victims. Many of these activities are likely to be exploitation attempts carried on by self propagating malware. Gathering intelligence on the nature of these activities and studying their structure is an important step towards a better understanding of the Internet threats. As explained in Section 5.4, such analysis requires an increase of the level of interaction, requirement that motivated our efforts in the development of the SGNET deployment.

The SGNET deployment was designed around the phase separation introduced by the epsilon-gamma-pi-mu model: each of the phases of a generic code injection attack is handled by a different entity of the distributed system. The emulation of these phases generates information on the characteristics of the specific instance. For instance, the network interaction involved in the exploit phase ε is associated to a traversal identifier in ScriptGen's FSM models. All the information generated by the different components of the SGNET deployment is collected and stored in the central database. Similarly to the Leurré.com case, this information is then enriched through different analysis tools. For instance, all the malware collected by SGNET is submit to the Anubis sandbox [3] to retrieve information on its behavior. The SGNET dataset puts at our disposal a variety of information on the observed exploits (ε), shellcodes (π) and malware samples (μ).[3]

Following the epsilon-gamma-pi-mu model, we model an attack as a tuple (ε, π, μ), assigning to each dimension a coordinate representative for a given "type" of interaction in the model. The relationship and the correlation among the dimensions of this three-dimensional space offer a perspective over the structure and the amount of code reuse present in nowadays exploitation attempts.

The identification of the interaction "type" is not always a straightforward task, since it needs to cope with the increasing usage of polymorphic techniques in malware and shellcode.

Malware families such as the Allaple one [15] take advantage of polymorphism to generate a new variant of themselves at each propagation attempt. Such a technique ensures each malware sample to completely mutate its binary content at every generation, making its detection much more complex to AV vendors. From our standpoint, the employment of such techniques leads to the proliferation of unique samples (downloaded only once) and makes the problem of attribution of two events to the activity of the same malware type much more complicated. How to define two completely different binaries to be *similar* and thus attribute two different code injections to the activity of the same malware?

The intuition that helped us in solving the problem is that any polymorphic technique can be used by attackers to randomize only *some* of the characteristics of a certain observed event. A polymorphic technique such as that previously mentioned will indeed succeed in randomizing the content of the injected malware (and consequently its MD5 hash), but might not succeed in randomizing other characteristics, such as its structure or its behavior. By looking at a sufficient amount of samples of the same activity type, it will be always possible to identify the invariant characteristics and reduce the activity classification problem to a pattern generation process.

For each of the 29283 code injection attacks observed by the SGNET honeypots in a period of 8 months ranging from January to August 2008, we have considered the set of characteristics described in Table 5.7. For each dimension, we have considered the corresponding characteristics vector, discovered *frequent* patterns and used these patterns to cluster them. In the rest of this work, we will refer to the name e-clusters, p-clusters and m-clusters to refer to the clusters of activities along the epsilon, pi and mu dimensions respectively.

---

[3] While theoretically possible, the prototype deployment did not collect sufficient information on the control flow hijack itself to include the dimension γ in the analysis

Table 5.7 Information taken into account

Exploit	Destination port
	Traversal identifier
	Alerts generated by Snort
Shellcode	Hash of the binary shellcode
	Interaction with the terminal emulator, if any
	Type of malware download (pushed by the attacker or pulled by the victim)
	Protocol used in the malware download
	Host involved in the download (the attacker itself or a central malware repository)
	Port involved in the download
Malware	Hash of the binary sample
	Size of the sample
	Number of created mutexes
	Name of the created processes
	Number of sections declared in the PE header
	Linker version declared in the PE header
	Packer name as detected by the PEiD database
	Number of sections in the PE header marked as both writable and executable

5.7.1 Degrees of freedom

We tried to get a very high level view over the relationship between the three (e,p,m) dimensions by quantifying the number of different combinations witnessed by the honeypot deployment in the 8 months observation period. We have thus counted the number of generated clusters, and the number of combinations over two or three dimensions. In the multi-dimensional case, we have compared the number of combinations witnessed by the honeypots with the maximum achievable number to have a rough estimate of the amount of variability over that combination.

As it possible to see from the table here under, most of the variability is introduced by the combinations of exploits and payloads: 20 different exploits have been combined with 58 different payloads in 107 different ways accounting for approximately 9% of all possible combinations.

e-clusters	20
p-clusters	58
m-clusters	74
(e,p) combinations	107 (9.22%)
(p,m) combinations	186 (4.33%)
(e,p,m) combinations	290 (0.33%)

The previous results seem to suggest a considerable reuse of exploitation code in different malware variants and eventually combined with personalized payloads. Indeed, in the 8 months of observation period, each e-cluster was combined in average with 5.9 different p-clusters and 21.2 different m-clusters. The same payload type was also often used by different malware families: in average, each payload type was used to upload 6.2 different m-clusters.

5.7.2 Interesting cases

It is interesting to look more in depth at different subsets of the epsilon-gamma-pi-mu space to better evaluate the impact of this variability in some practical cases. We have thus focused our attention on three interesting cases, associated to the usage of two specific vulnerability types and to the propagation strategies employed by a specific malware family associated to an m-cluster.

5.7.2.1 ASN.1 vulnerability

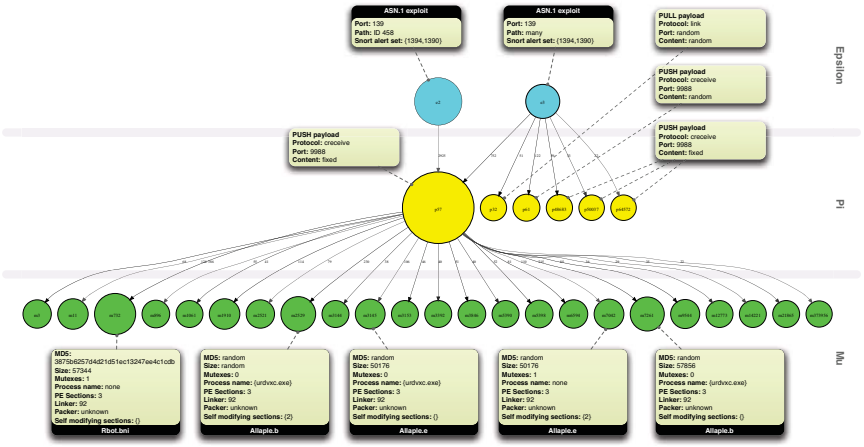


Fig. 5.15 The ASN.1 exploit (port 139)

Figure 5.15 provides an overview of all the observed code injections associated with the exploitation of the ASN.1 vulnerability (MS04-007) on TCP port 139.

In this specific case, there is a very low level of correlation between the first two dimensions. The totality of the *e2* exploits always pushes to the victim a single type of payload (*p57*). The payload involved in these exploits runs a small downloader that binds itself to TCP port 9988 and runs any content received upon connection from the attacker. Such download behavior is easy to identify and block: it is hard to identify a legitimate case in which a host should be allowed to accept inbound connections on a high port, and TCP port 9988 is not associated to any legitimate service. Despite its simplicity and its potentially low success rate, this payload is responsible for pushing to the honeypots a large number of m-clusters. For each of these clusters, we reported in Figure 5.15 the label associated with the malware samples by Kaspersky antivirus.

Many of the m-clusters involved in this propagation strategy are related to different variants of the Allaple worm, previously mentioned as example of polymorphic malware. These different variants are all sharing the same propagation strategy despite differences in the overall behavior of the worm. The same propagation strategy is also used by different malware types that are not directly related to the Allaple worm, such as the m-cluster 732, associated to the Rbot.bni IRC bot.

5.7.2.2 Rbot.bni malware family

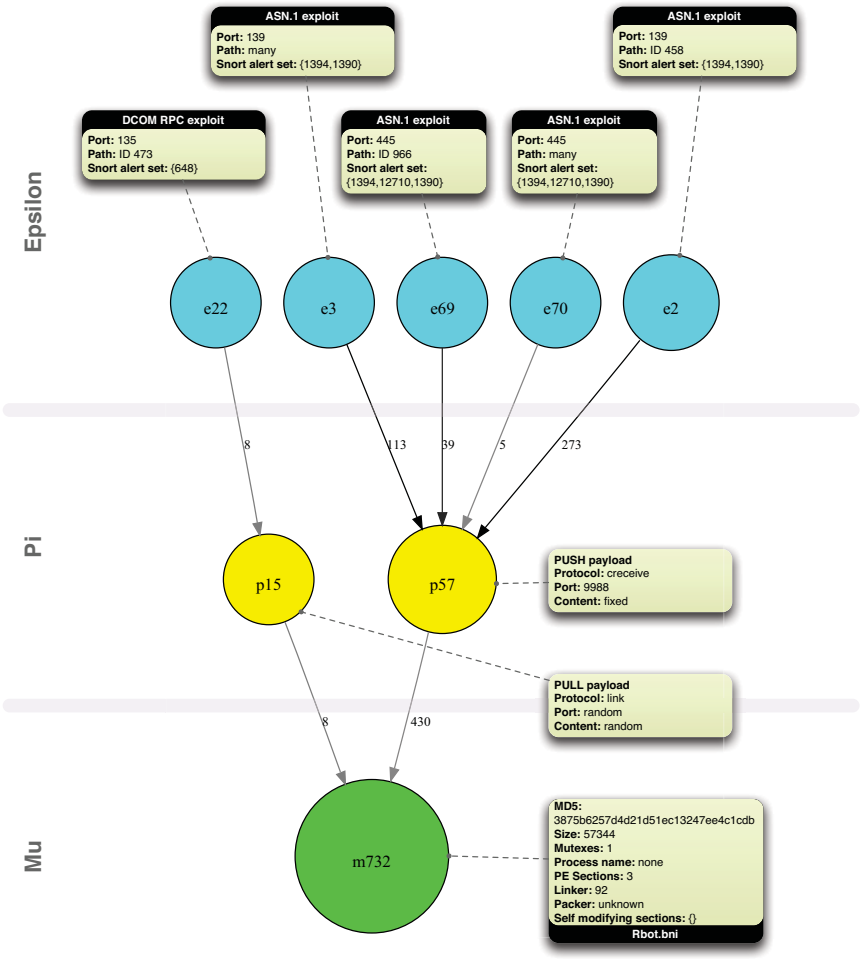


Fig. 5.16 Propagation ability for mu group m732

The propagation strategy used by Rbot.bni is shown in Figure 5.15. Interestingly, while most of the infections associated to it were witnessed through the previously analyzed ASN.1 exploit on port 139, SGNET honeypots observed a more diversified propagation strategy. This malware family was in fact also witnessed exploiting the ASN.1 exploit on port 445 and the DCOM RPC exploit on port 135. While all the ASN.1 exploits took advantage of the payload *p57* previously described, the RPC DCOM exploit (*e22*) took advantage of a completely different download strategy. The

exploits on this vulnerability forced in fact the victim to actively open a connection and download the malware from the attacker on a random port.

5.7.2.3 DCOM RPC vulnerability

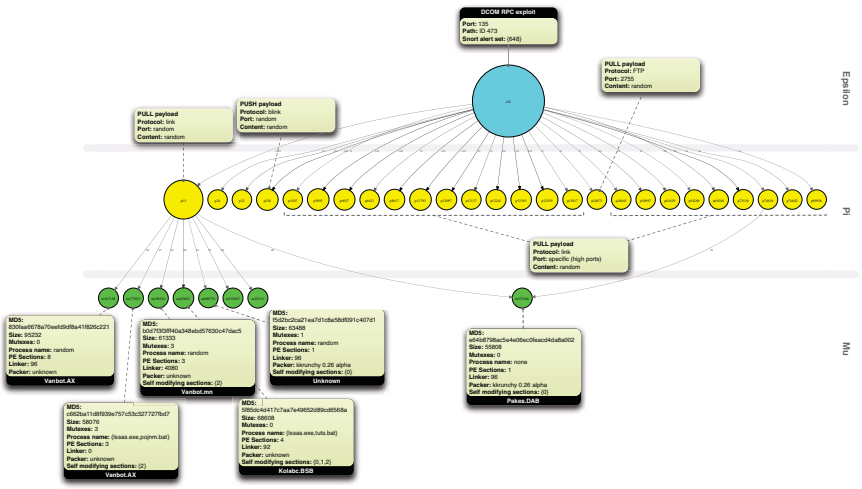


Fig. 5.17 The DCOM RPC exploit

We previously analyzed a case of high correlation between the exploitation type and the corresponding payload. Figure 5.17 takes into consideration a different vulnerability to show a completely different scenario. The vulnerability taken into consideration is the DCOM RPC vulnerability on port 135, used as "secondary" propagation vector by the Rbot.bni malware previously taken into consideration. The difference with Figure 5.15 is striking: in this case, a very high level of variability exists between the exploitation type and the payload involved.

Three different classes of payloads can be identified: a PULL payload (*p24073*) forces the download of malware from the attacker on port 2755; a PUSH payload (*p258*) forces the victim to accept the malware on a random port using the protocol *blink*; finally, there is a proliferation of clusters related to PULL payloads taking advantage of the protocol *link*.

The variability in terms of payloads is also reflected by a variability in terms of malware variants pushed through these combinations of epsilon and pi. While none of the mu clusters in Figure 5.17 correspond to polymorphic malware, all of them are associated with IRC-based C&C channels.

5.8 Conclusions

In this chapter, we have presented in detail Leurré.com, a worldwide distributed system of honeypots running since 2003. We have extensively described its architecture used for collecting meaningful data about emerging attack processes observed at various places on the Internet.

Several examples have been given throughout the text to illustrate the richness of our central data repository and the flexibility of its design, enabling a large diversity of analyses to be carried out on it. It is not the main purpose of this chapter to report on a specific analysis. Other publications have focused on some of these issues and some more work is ongoing. Instead, we have shown by means of simple examples that this database helps in discovering trends in the attacks and in characterizing them quite precisely. Next to this, we have also presented the important improvements we made to our infrastructure by deploying high-interaction ScriptGen sensors, which enable us to collect even more precise and valuable information about malicious activities. In the light of those promising results, we showed that this entire data collection infrastructure holds a great potential in augmenting our threats intelligence capability on the Internet. Being able to conduct in-depth analyses on this huge data collection, in a systematic way, will hopefully help us to make some advances towards the creation of early warning information systems.

So, it is our wish to share the data contained in this database with those interested in carrying some research on it. The authors can be reached by mail to get detailed information on how to join the project in order to gain access to the database.

References

1. ALMODE Security. Home page of disco at at http://www.altmode.com/disco/.
2. P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. Freiling. The Nepenthes Platform: An Efficient Approach to Collect Malware. *Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection (RAID)*, September 2006.
3. U. Bayer, C. Kruegel, and E. Kirda. *TTAnalyze: A Tool for Analyzing Malware*. PhD thesis, Master's Thesis, Technical University of Vienna, 2005.
4. I. Bomze, M. Budinich, P. Pardalos, and M. Pelillo. The maximum clique problem. In *Handbook of Combinatorial Optimization*, volume 4. Kluwer Academic Publishers, Boston, MA, 1999.
5. F. M. C. R. Center. Web security trends report q1/2008, http://www.finjan.com/content.aspx?id=827, sep 2008.
6. CERT. Advisory CA-2003-20 W32/Blaster worm, August 2003.
7. Z. Chen, L. Gao, and K. Kwiat. Modeling the spread of active worms. In *Proceedings of IEEE INFOCOM*, 2003.
8. M. P. Collins, T. J. Shimeall, S. Faber, J. Janies, R. Weaver, M. D. Shon, and J. Kadane. Using uncleanliness to predict future botnet addresses. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 93–104, New York, NY, USA, 2007. ACM.
9. E. Cooke, M. Bailey, Z. M. Mao, D. Watson, F. Jahanian, and D. McPherson. Toward understanding distributed blackhole placement. In *WORM '04: Proceedings of the 2004 ACM workshop on Rapid malcode*, pages 54–64, New York, NY, USA, 2004. ACM Press.
10. J. Crandall, S. Wu, and F. Chong. Experiences using Minos as a tool for capturing and analyzing novel worms for unknown vulnerabilities. *Proceedings of GI SIG SIDAR Conference on Detection of Intrusions and Malware and Vulnerability Assessment (DIMVA)*, 2005.
11. M. Dacier, F. Pouget, and H. Debar. Attack processes found on the internet. In *NATO Symposium IST-041/RSY-013*, Toulouse, France, April 2004.
12. M. Dacier, F. Pouget, and H. Debar. Honeypots, a practical mean to validate malicious fault assumptions. In *Proceedings of the 10th Pacific Ream Dependable Computing Conference (PRDC04)*, Tahiti, February 2004.
13. M. Dacier, F. Pouget, and H. Debar. Leurre.com: On the advantages of deploying a large scale distributed honeypot platform. In *Proceedings of the E-Crime and Computer Conference 2005 (ECCE'05)*, Monaco, March 2005.
14. DShield. Distributed Intrusion Detection System, www.dshield.org, 2007.

15. F-Secure. Malware information pages: Allaple.a, http://www.f-secure.com/v-descs/allaplea.shtml, December 2006.
16. A. Jain and R. Dubes. *Algorithms for Clustering Data*. Prentice-Hall advanced reference series, 1988.
17. C. Leita and M. Dacier. Sgnet: a worldwide deployable framework to support the analysis of malware threat models. In *Proceedings of the 7th European Dependable Computing Conference (EDCC 2008)*, May 2008.
18. C. Leita and M. Dacier. SGNET: Implementation Insights. In *IEEE/IFIP Network Operations and Management Symposium*, April 2008.
19. C. Leita, M. Dacier, and F. Massicotte. Automatic handling of protocol dependencies and reaction to 0-day attacks with ScriptGen based honeypots. In *RAID 2006, 9th International Symposium on Recent Advances in Intrusion Detection, September 20-22, 2006, Hamburg, Germany - Also published as Lecture Notes in Computer Science Volume 4219/2006*, Sep 2006.
20. C. Leita, K. Mermoud, and M. Dacier. Scriptgen: an automated script generation tool for honeyd. In *Proceedings of the 21st Annual Computer Security Applications Conference*, December 2005.
21. C. Leita, V. Pham, . Thonnard, E. Ramirez-Silva, F. Pouget, E. Kirda, and M. Dacier. The Leurre.com Project: Collecting Internet Threats Information using a Worldwide Distributed Honeynet. In *1st WOMBAT open workshop*, April 2008.
22. Maxmind Product. Home page ot the maxmind company at http://www.maxmind.com.
23. D. Moore, C. Shannon, G. Voelker, and S. Savage. Network telescopes: Technical report. *CAIDA, April*, 2004.
24. S. Needleman and C. Wunsch. *A general method applicable to the search for similarities in the amino acid sequence of two proteins*. J Mol Biol. 48(3):443-53, 1970.
25. Netgeo Product. Home page of the netgeo company at http://www.netgeo.com/.
26. V.-H. Pham and M. Dacier. Honeypot traces forensics: The observation view point matters. Technical report, EURECOM, 2009.
27. V.-H. Pham, M. Dacier, G. Urvoy Keller, and T. En Najjary. The quest for multi-headed worms. In *DIMVA 2008, 5th Conference on Detection of Intrusions and Malware & Vulnerability Assessment, July 10-11th, 2008, Paris, France*, Jul 2008.
28. G. Portokalidis, A. Slowinska, and H. Bos. Argos: an emulator for fingerprinting zero-day attacks. *Proc. ACM SIGOPS EUROSYS*, 2006.
29. F. Pouget, M. Dacier, and V. H. Pham. Understanding threats: a prerequisite to enhance survivability of computing systems. In *IISW'04, International Infrastructure Survivability Workshop 2004, in conjunction with the 25th IEEE International Real-Time Systems Symposium (RTSS 04) December 5-8, 2004 Lisbonne, Portugal*, Dec 2004.
30. T. C. D. Project. http://www.cymru.com/darknet/.
31. N. Provos. A virtual honeypot framework. In *Proceedings of the 12th USENIX Security Symposium*, pages 1–14, August 2004.
32. M. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A multifaceted approach to understanding the botnet phenomenon. In *ACM SIGCOMM/USENIX Internet Measurement Conference*, October 2006.
33. E. Ramirez-Silva and M. Dacier. Empirical study of the impact of metasploit-related attacks in 4 years of attack traces. In *12th Annual Asian Computing Conference focusing on computer and network security (ASIAN07)*, December 2007.
34. J. Riordan, D. Zamboni, and Y. Duponchel. Building and deploying billy goat, a worm detection system. In *Proceedings of the 18th Annual FIRST Conference*, 2006.
35. I. M. Sensor. http://ims.eecs.umich.edu/.
36. TCPDUMP Project. Home page of the tcpdump project at http://www.tcpdump.org/.
37. The Metasploit Project. www.metasploit.org, 2007.
38. O. Thonnard and M. Dacier. A framework for attack patterns' discovery in honeynet data. *DFRWS 2008, 8th Digital Forensics Research Conference, August 11- 13, 2008, Baltimore, USA*, 2008.

39. O. Thonnard and M. Dacier. Actionable knowledge discovery for threats intelligence support using a multi-dimensional data mining methodology. In *ICDM'08, 8th IEEE International Conference on Data Mining series, December 15-19, 2008, Pisa, Italy*, Dec 2008.
40. L. van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, November 2008.
41. T. Werner. Honeytrap. http://honeytrap.mwcollect.org/.
42. M. Zalewski. Home page of p0f at http://lcamtuf.coredump.cx/p0f.shtml.

Institut Eurécom

Research Report RR-09-226

HONEYPOT TRACES FORENSICS: THE OBSERVATION VIEW POINT MATTERS

February 12th, 2009

Van-Hau Pham* and Marc Dacier**
*Institute Eurécom, Sophia Antipolis, France
**Symantec Research Labs, Sophia Antipolis, France

Tel : (+33) 4 93 00 81 00
Fax : (+33) 4 93 00 82 00
Email : Van-Hau.Pham@eurecom.fr, Marc_Dacier@symantec.com

[1]Institut Eurécom's research is partially supported by its industrial members: BMW Group Research & Technology - BMW Group Company, Bouygues Télécom, Cisco Systems, France Télécom, Hitachi Europe, SFR, Sharp, STMicroelectronics, Swisscom, Thales.

HONEYPOT TRACES FORENSICS: THE OBSERVATION VIEW POINT MATTERS

Van-Hau Pham and Marc Dacier

Abstract

In this paper, we propose a method to identify and group together traces left on low interaction honeypots by machines belonging to the same botnet(s) without having any a priori information at our disposal regarding these botnets. In other terms, we offer a solution to detect new botnets thanks to very cheap and easily deployable solutions. The approach is validated thanks to several months of data collected with the worldwide distributed Leurré.com system. To distinguish the relevant traces from the other ones, we group them according to either the platforms, i.e. targets hit or the countries of origin of the attackers. We show that the choice of one of these two observations view points dramatically influences the results obtained. Each one reveals unique botnets. We explain why. Last but not least, we show that these botnets remain active during very long periods of times, up to 700 days, even if the traces they left are only visible from time to time.

Contents

List of Figures

1 Introduction

There is a consensus in the security community to say that botnets are today's plague of the Internet. A lot of attention has been paid to detect and eradicate them. Several approaches have been proposed for this purpose. By identifying the so called *Command and Control (C&C)* channels, one can keep track of all IPs connecting to it. The task is more or less complicated, depending on the type of *C&C* (IRC [2, 4, 6, 7, 14, 20], HTTP [3, 5, 23], fast-flux based or not [12, 16, 21], P2P [8, 13, 24, 26], etc.) but, in any case, one needs to have some insight about the channels and the capability to observe all communications on them. Another approach consists in sniffing packets on a network and in recognizing patterns of *bot-like* traffic. This is, for instance, the approach pursued by [9–11] and [22, 25]. The solutions mostly aim at detecting compromised machines in a given network rather than to study the botnets themselves as they only see the bots that exist within the network under study.

In this work, we are interested in finding a very general technique that would enable us to count the amount of various botnets that exist, their size and their lifetime. As opposed to previous work, we are not interested in studying a particular botnet in details or in detecting compromised nodes in a given network. We also do not want to learn the various protocols used by bots to communicate in order to infiltrate the botnets and obtain more precise information about them [20]. By doing so, we certainly will not be able to get as much in depth information about this or that botnet but our hope is to provide insights into the bigger picture of today's (and yesterday's) botnets activities.

Before describing our approach, it is crucial to understand the subtle difference that exists between counting the amount of machines launching a given attack and the amount of machines members of a given botnet. It is very misleading to believe that one can derive the latter from the former. Indeed, it is quite common to see several distinct botnets relying on the same attack vector to compromise more hosts. In such case, the total amount of machines observed using a given attack vector will be greater or equal to the sum of all members of all these botnets (it can be greater as machines not belonging to any botnet may also launch this attack). Clearly, any approach relying on simply summing up counters based on attack vectors characteristics (e.g. ID alerts, firewall logs, AV detection, etc.) is likely to grossly overestimate the size of botnets.

The solution described in the following is generic and simple to deploy widely. It relies on a distributed system of low interaction honeypots. Based on the traces left on these honeypots, we provide a technique that groups together the traces that are likely to have been generated by groups of machines controlled by a similar authority. Since we have no information regarding the *C&C* they obey to, we do not know if these machines are part of a single botnet or if they belong to several botnets that are coordinated. Therefore, to avoid any ambiguity, we write in the following that they are part of a *army of zombies*. An *army of zombies* can be a

single botnet or a group of botnets the actions of which are coordinated during a given time interval.

In this paper, we propose a technique to identify and study the size as well as the lifetime of such *armies of zombies*. We show that armies can stay active for very long periods of time, up to 700 days, even if they manifest themselves only from time to time. The approach does not pretend to be able to identify all *armies of zombies* that could be found in our dataset. At the contrary, we show that, depending on how the dataset is preprocessed, i.e. depending on the observation viewpoint, different armies can be found. Exhaustiveness is not our concern at this stage but, instead, we are interested in offering an approach that could easily be widely adopted and that offers a much better picture of the reality of the problem.

The idea exposed here is similar, in its spirit, to the one presented in the paper coauthored by Allmann et al. [1]. However, instead of " [...] *leveraging the deep understanding of network detectives and the broad understanding of a large number of network witnesses to form a richer understanding of large-scale coordinated attackers*", our approach relies on a diverse yet limited number of low interaction honeypots. They do not need to be neither as smart as the network detectives nor as numerous as the network witnesses proposed in that work. Both approaches are quite complementary.

The reminder of the paper is organised as follows. Section 2 defines the terms used in the paper. Section 3 describes the dataset we have used and what we mean when we refer to the notion of *observation viewpoint*. It also explains why it matters when trying to identify *armies of zombies*. In Section 4, we describe the method itself that we have applied to find these armies, we provide the main characteristics of the results obtained as well as two precise, yet anecdotal, examples of armies detected thanks to our method. Finally, Section 5 concludes the paper.

2 Terminology

In order to avoid any ambiguity, we introduce a few terms that will be used throughout the text.

- **Platform**: A physical machine simulating, thanks to honeyd [19], the presence of three distinct machines. A platform is connected directly to the Internet and collects tcpdump traces that are fed on a daily basis into the centralized Leurré.com's database.

- **Leurré.com**: The Leurré.com project is a distributed system of platforms as defined earlier, deployed in more than 50 different locations in 30 different countries. More detailed information about it can be found in [15]

- **A Source** corresponds to an IP address that has sent at least one packet to, at least, one platform. It is important to understand that a given IP address can correspond to several distinct sources. Indeed, a given IP remains associated

to a given source as long as there is no more than 25 hours between 2 packets received from that IP. After such a delay, a new source identifier will be assigned to the IP. By grouping packets by sources instead of by IPs, we minimize the risk of gathering packets sent by distinct physical machines that have been assigned the same IP dynamically after 25 hours.

- **A Cluster** is made of a group of sources that have left highly similar network traces on all platforms they have been seen on. Clusters have been precisely defined in [18]. They aim at grouping together attackers that are likely launching attacks with the very same attack tool.

- **A Cluster time series** $\Phi_{T,c}$ is a function defined over a period of time T, T being defined as a time interval (in days). That function returns the amount of sources per day associated to a cluster c.

- **An Observed cluster time series** $\Phi_{T,c,op}$ is a function defined over a period of time T, T being defined as a time interval (in days). That function returns the amount of sources per day associated to a cluster c that can be seen from a given *observation view point* op. The observation view point can either be a specific platform or a specific country of origin. In the first case, $\Phi_{T,c,platform_X}$ returns, per day, the amount of sources belonging to cluster c that have hit $platform_X$. Similarly, in the second case, $\Phi_{T,c,country_X}$ returns, per day, the amount of sources belonging to cluster c that are geographically located in $country_X$. Clearly, we always have: $\Phi_{T,c} = \sum^{\forall i \in countries} \Phi_{T,c,i} = \sum^{\forall x \in platforms} \Phi_{T,c,x}$

- **An attack event** is defined as a set of observed cluster time series exhibiting a particular shape during a limited time interval. This time interval typically lasts a couple of days but it can be as short as a single day in the case of observed cluster time series having a one day peak of activities. The existence of attack events highlights the coordinated activities of several attacking machines. It is important to notice that the set can be singleton. This is typically the case when the set is a peak of activities on a single day.

  We denote the attack event i as $e_i = (T_{start}, T_{end}, S_i)$ where the attack event starts at T_{start}, ends at T_{end} and S_i contains a set of observed cluster time series identifiers (c_i, op_i) such that all $\Phi_{[T_{star}-T_{end}),c_i,op_i}$ are strongly correlated to each other $\forall (c_i, op_i) \in S_i$. As an example, the top plot of Figure 1 represents the attack event 225 which consists of cluster 60332 (targeting port 5900 TCP) attacking seven platforms 5,8, 11, ...,31. Each curve represents the amount of sources of that cluster observed from one of these platforms. As we can observe, the attack event start at day 393 and ends at day 400. According to our convention, we have $e_{225} = (393, 400, \{(60232, 5), (60232, 8), ..., (60232, 31)\})$.

  Similarly, the bottom plot of Figure 1 represents the attack event 14 which consists of activities of cluster 0 on day 307 coming almost only from Spain. So, $e_{14} = (307, 307, \{(0, ES)\})$

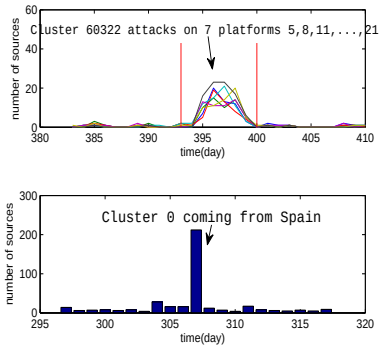


Figure 1: on the top plot, cluster 60232 attacks seven platforms from day 393 to day 400. On the bottom plot, peak of activities of cluster 0 from Spain on day 307

3 IMPACT OF OBSERVATION VIEW POINT

3.1 Dataset Description

In order to have a clean dataset for our experiments, we have selected the traces observed on 40 platforms out of the 50 that we had at our disposal. All these 40 platforms have been running for more than 800 days. During this period, none of them has been down for more than 10 times and each of them has been up continuously for at least 100 days at least once. They all have been up for a minimum of 400 days over that period. The total amount of sources observed, day by day, on all these 40 platforms can be denoted by the initial time series TS over a period of 800 days. We can split that time series per country[1] of origin of the sources. This gives us 231 time series TS_X where the i^{th} point of such time series indicates the amount of sources, observed on all platforms, located in country X. We represent by TS_L1 the set of all these Level 1 time series. To reduce the computational cost, we keep only the countries from which we have seen at least 10 sources on at least one day. This enables us to focus on 85 (the set of corresponding countries is called $big_{countries}$), instead of 231, time series. We represent by TS_L1' this refined set of Level 1 time series. Then, we split each of these time series by cluster to produce the final set of time series $\Phi_{[0-800),c_i,country_j} \forall c_i$ and $\forall country_j \in big_{countries}$. The i^{th} point of the time series $\Phi_{[0-800),X,Y}$ indicates the amount of sources originating from country Y that have been observed on day i attacking any of our platforms thanks to the attack defined by means of the cluster X. We represent by TS_L2 the set of all these Level 2 time series. In this case $|TS\_L2|$ is equal to 436,756 which corresponds to 3,284,551 sources.

[1]The geographical location is given to us thanks to the Maxmind product, based on the IP address. However, some IPs can not be mapped to any real country and are attached to labels not corresponding to any country, e.g. EU,A1,..

As explained in [17], time series that barely vary in amplitude over the 800 days are meaningless to identify attack events and we can get rid of them. Therefore, we only keep the time series that highlight important variations during the 800 days period. We represent by TS_L2' this refined set of Level 2 time series. In this case $|TS\_L2'|$ is equal to 2,420 which corresponds to 2,330,244 sources.

We have done the very same splitting and filtering by looking at the traces on a per platform basis instead of on a per country of origin basis. The corresponding results are given in Table 1.

TS consists of 3,477,976 sources		
OVP	country	platform
$\|TS\_L1\|$	231	40
$\|TS\_L1'\|$	85 (94,4% TS)	40 (100% TS)
$\|TS\_L2\|$	436,756	395,712
$\|TS\_L2'\|$ sources	2,420 2,330,244 (67% of TS)	2,127 2,538,922 (73% of TS)

Table 1: dataset description: TS: *all sources observed on the period under study*, OVP: *observation view point*, TS_L1: *set of time series at country/platform level*, TS_L1': *set of significant time series in* TS_L1, TS_L2 : *set of all cluster time series*, TS_L2' *set of strongly varying cluster time series*

3.2 Attack Event Detection

Having defined the time series we are interested in, we now want to find attack events, that is we now want to identify all time periods during which 2 or more of these observed cluster time series are correlated together.

To do this, in a first step, we fix the time period to a value of L days and we use a sliding window of size L to assess the correlation of all pairs of time series over such sliding window. Therefore, given N time series of length T, we must compute the correlation of N time series for T-L+1 time interval $\{[1, L], [2, L + 1], \ldots [T - L, T]\}$. As a result, we obtain the correlated time intervals for every pair of time series in N. A correlated time interval of two cluster time series is the interval in which two time series are correlated. After this first step, we group together all pairs of cluster time series that are correlated together over the same period of time. Each group of correlated observed cluster time series over a given period of time constitutes what we have defined as an *attack event*.

It is worth noting that this method, which we refer to as $M1$ in the sequel, can not detect attack events made of one observed cluster time series. This is typically the case for peaks of activities occurring on a single day. In such simpler cases, it is much more efficient to apply another, less expensive, algorithm to identify the

attack events. This is what we have done. For the sake of conciseness, we have decided not to include the description of this second method, $M2$, in the paper as it lies outside the scope of the message we are interested to deliver.

In the first case, the techniques used to find strongly correlated time series are classical ones developed within the signal processing community. We refer the interested reader to our previous work [17] where we have covered them in some more detail and have positioned them with respect to the state of the art in this domain. It is worth stressing that, in this earlier publication, the methodology used was very different as well as the results presented. Indeed, in that first work, we have presented a cheap algorithm, based on heuristics, to validate the mere existence of attack events whereas in this work, we have carried out an expensive, brute force approach, to study and analyze the relationships between all attack events one could find in a much larger dataset.

3.3 Impact of Observation View Point

3.3.1 Results on Attack Event Detection

We have applied the attack events identification techniques to our 2 distinct datasets, namely $TS_{country}$ and $TS_{platform}$. For the time series in $TS_{country}$, the first method M1 (resp. second method M2), i.e. the general one, has found 549 (resp. 43) attack events. The total amount of sources found in these attack events is 552,492 for the first method and 21,633 for the second one. Thus, all in all, sources participating to identified attack events account for 574,125 sources (corresponding to 16,5% of all sources contained in our initial dataset). Similarly, when working with the time series found in $TS_{platform}$, we end up with a total of 690 attack events this time, containing 578,372 sources. The results are given in Table 2

Table 2: Result on Attack Event Detection

	AE-set-I($TS_{country}$)		AE-set-II($TS_{platform}$)	
	No.AEs	No.sources	No.AEs	No.sources
M1	549	552,492	564	550,305
M2	43	21,633	126	28,067
Total	592	574,125	690	578,372

No.AEs: amount of attack events
M1,M2: methods represented in Section 3.2

3.3.2 Analysis

The table highlights the fact that depending on how we decompose the initial set of traces of attacks (i.e the initial time series TS), namely by splitting it by countries of origin of the attackers or by platforms attacked, different attacks events show up. To assess the overlap between attack events detected from different observation view points we use the *common source ratio, namely csr*, measure

as follows:

$$csr(e, AE_{op'}) = \frac{\sum_{\forall e' \in AE_{op'}} |e \cap e'|}{|e|}$$

in which $e \in AE_{op}$ and $|e|$ is the amount of sources in attack event e, AE_{op} is $AE_{country}$ and $AE_{op'}$ is $AE_{platforms}$ (or vice versa).

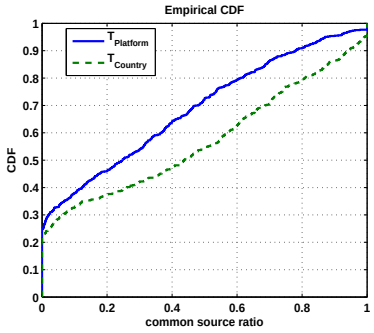


Figure 2: CDF common source ratio

Figure 2 represents the two cumulative distribution functions corresponding to this measure. The point (x, y) on the curve means that there are $y * 100\%$ of attack events obtained thanks to $T_{country}$ (resp $T_{platforms}$) that have less than $x * 100\%$ of sources in common with all attack events obtained thanks to $T_{platforms}$ (resp $T_{country}$). The $T_{country}$ curve represents the cumulative distribution obtained in this first case and the $T_{platforms}$ one represents the CDF obtained when starting from the attacks events obtained with the intial $T_{platforms}$ set of time series. As we can notice, around 23% (resp. 25%) of attack events obtained by starting from the $T_{country}$ (resp. $T_{platform}$) set of time series do not share any sources in common with any attack events obtained when starting the attack even identification process from the $T_{platform}$ (resp. $T_{country}$) set of time series. This corresponds to 136 (16,919 sources) and 171 (75,920 sources) attack events not being detected. In total, there are 288,825 (resp. 293,132) sources present in AE-Set-I (resp. AE-Set-II), but not in AE-Set-II (resp. AE-Set-I). As a final note, there are in total 867,248 sources involved in all the attack events detected from both datasets which correspond to 25% the attacks observed in the period under study. This number is coincidentally comparable with work in [20], in which, with a much more complicated technique, the authors claim that:"*[...] 27% of all malicious connection attempts observed from our distributed darknet can be directly attributed to botnet related spreading activity*".

3.3.3 Explanation

There are good reasons that explain why we can not rely on a single viewpoint to detect all attacks events. They are described here below.

Split by country: Suppose we have one botnet B made of machines that are located within the set of countries $\{X, Y, Z\}$. Suppose that, from time to time, these machines attack our platforms leaving traces that are also assigned to a cluster C. Suppose also that this cluster C is a very *popular* one, that is, many other machines from all over the world continuously leave traces on our platforms that are assigned to this cluster. As a result, the activities specifically linked to the botnet B are lost in the noise of all other machines leaving traces belonging to C. This is certainly true for the cluster time series (as defined earlier) related to C and this can also be true for the time series obtained by splitting it by platform, $\Phi_{[0-800),C,platform_i} \forall platform_i \in 1..40$.However, by splitting the time series corresponding to cluster C by countries of origins of the sources, then it is quite likely that the time series $\Phi_{[0-800),C,country_i} \forall country_i \in \{X, Y, Z\}$ will be highly correlated during the periods in which the botnet present in these countries will be active against our platforms. This will lead to the identification of one or several attack events.

Split by platform: Similarly, suppose we have a botnet B' made of machines located all over the world. Suppose that, from time to time, these machines attack a specific set of platforms $\{X, Y, Z\}$ leaving traces that are assigned to a cluster C. Suppose also that this cluster C is a very *popular* one, that is, many other machines from all over the world continuously leave traces on all our platforms that are assigned to this cluster. As a result, the activities specifically linked to the botnet B' are lost in the noise of all other machines leaving traces belonging to C. This is certainly true for the cluster time series (as defined earlier) related to C and this can also be true for the time series obtained by splitting it by countries, $\Phi_{[0-800),C,country_i} \forall country_i \in big_{countries}$. However, by splitting the time series corresponding to cluster C by platforms attacked, then it is quite likely that the time series $\Phi_{[0-800),C,platform_i} \forall platform_i \in \{X, Y, Z\}$ will be highly correlated during the periods in which the botnet influences the traces left on the sole platforms concerned by its attack. This will lead to the identification of one or several attack events.

The top plot of Figure 3 represents the attack event 79. In this case, we see that the traces due to the cluster 175309 are highly correlated when we group them by platform attacked. In fact, there are 9 platforms involved in this case, accounting for a total of 870 sources. If we group the same set of traces by country of origin of the sources, we end up with the bottom curves of Figure 3 where the specific attack event identified previously can barely be seen. This highlights the existence of a botnet made of machines located all over the world that target a specific subset of the Internet.

4 On the armies of Zombies

So far, we have identified what we have called attack events which highlight the existence of coordinated attacks launched by a group of compromised machines,

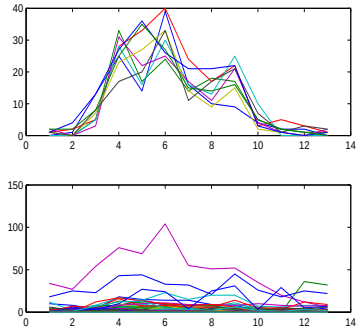


Figure 3: top plot represents the attack event 79 related to cluster 17309 on 9 platforms. The bottom plot represents the evolution of this cluster by country. Noise of the attacks to other platforms decrease significantly the correlation of observed cluster time series when split by country

i.e. a zombie army. It would be interesting to see if the very same army manifests itself in more than one attack event. To do this, we propose to compute what we call the *action sets*. An *action set* is a set of attack events that are likely due to same army. In this Section, we show how to build these action sets and what information we can derive from them regarding the size and the lifetime of the zombie armies.

4.1 Identification of the armies

4.1.1 Similarity Measures

In its simplest form, a zombie army is a classical botnet. It can also be made of several botnets, that is several groups of machines listening to distinct *C&C*. This is invisible to us and irrelevant. All that matters is that all the machines do act in a coordinated way. As time passes, it is reasonable to expect members of an army to be cured while others join. So, if the same army attacks our honeypots twice over distinct periods of time, one simple way to link the two attack events together is by noticing that they have a large amount of IP addresses in common. More formally, we measure the likelihood of two attacks events e_1 and e_2 to be linked to the same zombie army by means of their similarity defined as follows:

$$sim(e_1, e_2) = \begin{cases} max(\frac{|e_1 \cap e_2|}{|e_1|}, \frac{|e_1 \cap e_2|}{|e_2|}) & \text{if } |e_1 \cap e_2| < 200 \\ 1 & \text{otherwise} \end{cases}$$

We will say that e_1 and e_2 are caused by the same zombie army if and only if $sim(e_1, e_2) > \delta$. This only makes sens for *reasonable* values of δ. We address this issue in the coming subsections.

4.1.2 Action Sets

We now use the $sim()$ function to group together attack events into action sets. To do so, we build a simple graph where the nodes are the attack events. There is an arc between two nodes e_1 and e_2 if and only if $sim(e_1, e_2) > \delta$. All nodes that are connected by at least one path end up in the same action set. In other words, we have as many action sets as we have disconnected graphs made of at least two nodes; singleton sets are not counted as action sets.

We note that our approach is such that we can have an action set made of three attack events e_1, e_2 and e_3 where $sim(e_1, e_2) > \delta$ and $sim(e_2, e_3) > \delta$ but where $sim(e_1, e_3) < \delta$. This is consistent with our intuition that armies can evolve over time in such a way that the machines present in the army can, eventually, be very different from the ones found the first time we have seen the same army in action.

4.1.3 Results

To test the sensitivity of the threshold δ, we have computed the amount of action sets for the two datasets for different values of δ. The result is represented in top plot of Figure 4 (the bottom plot represent the corresponding amount of attack events involved in the armies). As we can see, at first, for the value of δ from 1% to 7%, the amount of action sets increases rapidly. Indeed, for very small values of δ all nodes remain connected together but, as δ increases, the initial graph loses arcs and more disconnected graphs appear, i.e. more action sets show up. This creation of action sets reaches a maximum after which action sets start disappearing with a growing δ value. This is due to the fact that some graphs are broken into isolated nodes that are not counting as attack sets anymore. The two curves reach their maximum values almost at the same position (when $\delta = 8\%$). Then they both start decreasing linearly.

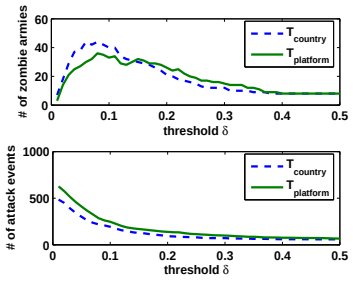


Figure 4: sensitivity check of threshold δ

In the context of this paper, we have arbitrarily chosen to investigate deeper the armies we can find when setting $\delta = 10\%$. We do not pretend that this number is optimal in any sense and, in fact, we do not really care. Indeed, our purpose, at this stage, is just to look at the results for one given value of δ and see if, yes or no, this theory of zombie armies seems to be valid or not, based on the characteristics

of the ones we will find in that particular case. It can very well be that the attack events found in attack sets, as we have built them, have no underlying common cause and that they accidentally share common IPs.

For such value of δ we have identified 40 (resp. 33) zombie armies from AE-set-I (resp. AE-set-II) which have issued a total of 193 (resp. 247) attack events. Figure 5 represents the distribution of attack events per zombie army. Its top (resp. bottom) plot represents the distribution obtained from AE-set-I(resp. AE-set-II). We can see that the largest amount of attack events for an army is 53 (resp. 47) whereas 28 (resp. 20) armies have been observed only two times.

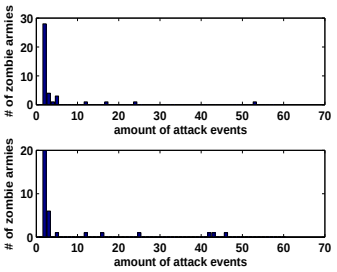


Figure 5: Zombie Army Size

4.2 Main Characteristics of the Zombie armies

In this section, we will analyze the main characteristic of the zombie armies.
Lifetime of Zombie Army Figure 6 represents the cumulative distribution of min-

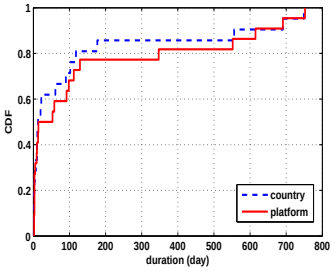
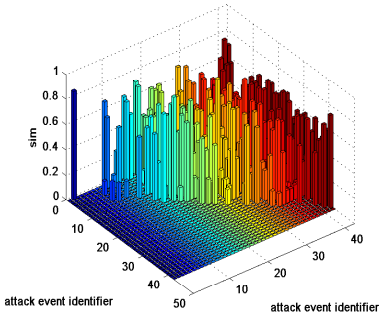


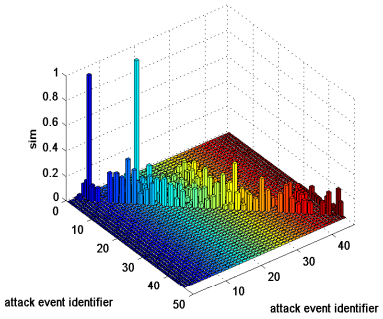
Figure 6: CDF duration

imum lifetime of zombie armies obtained from $TS_{platform}$ and $TS_{country}$ (see Section 4.1.3). According to the plot, around 20% of zombie armies have existed for more than 200 days. In the extreme case, two armies seems to have survived for 700 days! Such result seems to indicate that either i) it takes a long time to cure compromised machines or that ii) armies are able to stay active for long periods of time, despite the fact that some of their members disappear, by continuously compromising new ones.

Lifetime of Infected Host in Zombie Armies In fact, we can classify the armies into two classes as mentioned in the previous Section. For instance, Figure 7a represents the similarity matrix of zombie army 33, ZA33. To build this matrix, we first order its 42 attack events according to their occurred time. Then we represent their similarity relation under an 42×42 similarity matrix $\mathscr{M}$. The cell (i,j) represents the value of $sim()$ of the ordered attack event i^{th} and j^{th}. Since, $\mathscr{M}$ is a symmetric matrix, for the visibility, we represent only half of it. As we can see, we have a very high similarity measure between almost all the attacks events, around 60%. This is also true between the very first and the very last attack events. It is important to notice the time interval between the first and the last activities observed from this army is 753 days!



(a)



(b)

Figure 7: Renewal rate of zombie armies

Figure 7b represents an opposite case, the zombie army 31, ZA31, consisting of 46 attack events. We proceed as above to build its similarity matrix. As we can notice the important values are surrounded around the main diagonal of $\mathscr{M}$. It means that the attack event i^{th} has the same subset of infected machines with only few attack events happening not far from it in terms of time. Another important point to be noticed is that this army changes its attack vectors over time. In fact, it

moves from attack against 4662 TCP, to 1025 TCP, then 5900 TCP, 1443 TCP, 2967 TCP, 445 TCP,...And the lifetime of this army is 563 days! It is clear, from these two cases, that the composition of armies evolves over time in different ways. More work remains to be done in order to understand the reasons behind these various strategies.

Attack Capacity By attack capacity, we refer to the amount of different attacks that a given army is observed lauching over time. The advanced worm, namely multi-headed worm, we have presented in our earlier work [17] is an example of worms that have many attack vectors and use them dynamically. The multi attack vectors allow the worms to have a large chance to propagate, and the varying in activity helps them to have multi attack traces which make it harder for IDS to detect them. This work reinforces the results we have earlier [17]. In fact, in previous work, we were able to detect multi-headed worms by the correlation of attack traces generated by different attack tools within an attack event. In this work, we have some even stronger evidence.Indeed, thanks to the notion of army, we observe several cases in which the same IP address has different behaviors in different attack events attached to a given army. As an example, the two attack events 128 and 131 consist of clusters 1378 and 2666 respectively. They both have 106 IP addresses in common and belong to the zombie army 12. All the attacks of attack event 128 are against port 64783 TCP whereas all the attacks of attack event 131 are against port 6211 TCP. The conclusion is that these 106 attacking machines mentioned earlier have dynamically changed their behavior. Finally, Figure 8 represents the distribution of number of distinct cluster per army. One zombie army has almost 120 clusters, yet not all of them are very different from each other.

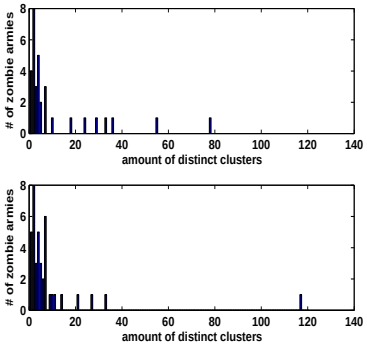


Figure 8: Zombie Army Attack Capacity

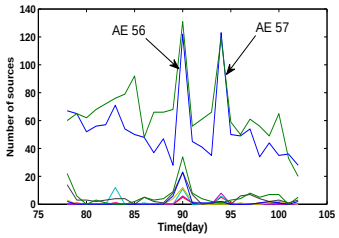
4.3 Illustrated Examples

After having offered a high level overview of the method and main characteristics of the results obtained, we feel it is important to give a couple of concrete,

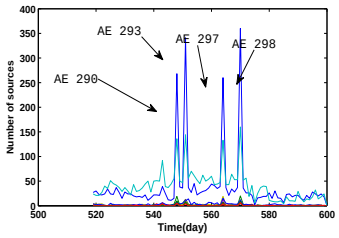
simple, examples of armies we have discovered. This should help the reader in better understanding the reality of two armies as well as what they look like. This is what we do in the next two subsections where we briefly present two representative armies.

4.3.1 Example 1

Zombie army 29, ZA-29, is an interesting example which has only been observed attacking a single platform. However, 16 distinct attack events are linked to that army! Figure 9a presents its two first activities corresponding to the two attack events 56 and 57. Figure 9b represents other four attack events. In each attack event, the army tries a number of distinct clusters such as 13882, 14635, 14647, 56608, 144028, 144044, 149357, 164877, 166477. These clusters try many combinations of Windows ports (135 TCP, 139 TCP, 445 TCP) and Web server (80 TCP). The time interval between the first and the last activities is 616 days !



(a)



(b)

Figure 9: attack events of ZA29

4.3.2 Example 2

Last but not least, the zombie army 33, ZA-33, consisting of 42 attack events (already mentioned in Section 4.2) is an example of a multi-botnets zombie army. In fact, it seems that several botnets do different jobs and from time to time, they do some tasks together. In fact, in one hand, an important set of machines coming from Italy attacks several times one platform in China. As an example, the two top plots of Figure 10 are two examples of these attacks. The attack event 291

consisting of several clusters attack on port 64783T. And always coming from Italy, and targeting the same platform, but attack event 195 tries many clusters on port 9661 TCP. On the other hand, another component of ZA-33 consistently sends ICMP packets only, always coming from Greece and always targeting the same platform also located in Greece (see two plots in the middle of Figure 10). And as an example of coordination of two components of ZA33, the two plots in the bottom of Figure 10 represent two attack events (out of four) coming mostly from these two countries and attacking these two platforms. As a reminder, by design, there are always overlap between the attack events, for instance, attack event 483 share 41 IP address in common with AE 307, whereas 454 and 483 have 47 IP addresses in common.... The interval between the first and the last attack event issued by this zombie army is 753 days.

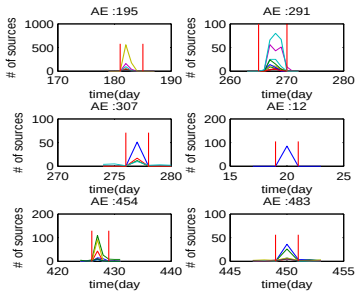


Figure 10: 6 attack events from zombie army 33

5 Conclusion

In this paper, we have addressed the important attack attribution problem. We have shown how low interaction honeypots can be used to track armies of zombies and characterize their lifetime and size. More precisely, this paper offers three main contributions. First of all, we propose a simple technique to identify, in a systematic and automated way, the so-called attack events in a very large dataset of traces. We have implemented and demonstrated experimentally the usefulness of this technique. Secondly, we have shown how, by grouping these attack events, we can identify long living armies of zombies. Here too, we have validated experimentally the soundness of the idea as well as the meaningfulness of the results it produces. Last but not least, we have shown the importance of the selection of the observation viewpoint when trying to group such traces for analysis purposes. Two such viewpoints have been considered in this paper, namely the geolocation of the attackers and the platform attacked. Results of the experiments have highlighted the benefits of considering more than one viewpoint as each of them offers unique insights into the attack processes. Future work needs to be done to consider other

viewpoints as well as the possibility to combine these various viewpoints into a uniformed framework.

References

[1] M. Allman, E. Blanton, V. Paxson, and S. Shenker. Fighting coordinated attackers with cross-organizational information sharing. In *Hotnets 2006*, 2006.

[2] Paul Barford and Vinod Yegneswaran. An inside look at botnets. *Advances in Information Security*, 27:171–191, 2007.

[3] Ken Chiang and Levi Lloyd. A case study of the rustock rootkit and spam bot. In *First Workshop on Hot Topics in Understanding Botnets*, 2007.

[4] Evan Cooke, Farnam Jahanian, and Danny McPherson. The zombie roundup: understanding, detecting, and disrupting botnets. In *SRUTI'05: Proceedings of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet Workshop*, pages 6–6, Berkeley, CA, USA, 2005. USENIX Association.

[5] Neil Daswani and Michael Stoppelman. The anatomy of clickbot.a. In *HotBots'07: Proceedings of the First Workshop on Hot Topics in Understanding Botnets*, pages 11–11, Berkeley, CA, USA, 2007. USENIX Association.

[6] Felix C. Freiling, Thorsten Holz, and Georg Wicherski. Botnet tracking: Exploring a root-cause methodology to prevent distributed denial-of-service attacks. In *Lecture Notes in Computer Science*, pages 319–335. Springer-Verlag GmbH, September 2005.

[7] Jan Goebel and Thorsten Holz. Rishi: Identify bot contaminated hosts by irc nickname evaluation. In *Workshop on Hot Topics in Understanding Botnets 2007*, 2007.

[8] Julian B. Grizzard, Vikram Sharma, Chris Nunnery, Brent ByungHoon Kang, and David Dagon. Peer-to-peer botnets: overview and case study. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 1–1, Berkeley, CA, USA, 2007. USENIX Association.

[9] Guofei Gu, Roberto Perdisci, Junjie Zhang, and Wenke Lee. Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *USENIX Security '08*, 2008.

[10] Guofei Gu, Phillip Porras, Vinod Yegneswaran, Martin Fong, and Wenke Lee. Bothunter: Detecting malware infection through ids-driven dialog correlation. In *Proceedings of the 16th USENIX Security Symposium*, August 2007.

[11] Guofei Gu, Junjie Zhang, and Wenke Lee. Botsniffer: Detecting botnet command and control channels in network traffic. In *the 15th Annual Network and Distributed System Security Symposion*, 2008.

[12] Thorsten Holz, Christian Gorecki, Konrad Rieck, and Felix C. Freiling. Measuring and detecting fast-flux service networks. In *NDSS 2008*, 2008.

[13] Thorsten Holz, Moritz Steiner, Frederic Dahl, Ernst Biersack, and Felix Freiling. Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm. In *LEET'08: Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, pages 1–9, Berkeley, CA, USA, 2008. USENIX Association.

[14] Nicholas Ianelli and Aaron Hackworth. Botnets as a vehicle for online crime. In *18th Annual FIRST Conference*, May 2007.

[15] Corrado Leita, Van Hau Pham, Olivier Thonnard, Eduardo Ramirez Silva, Fabien Pouget, Engin Kirda, and Marc Dacier. The leurre.com project: collecting internet threats information using a worldwide distributed honeynet. In *1st WOMBAT workshop, April 21st-22nd, Amsterdam, The Netherlands*, Apr 2008.

[16] Emanuele Passerini, Roberto Paleari, Lorenzo Martignoni, and Danilo Bruschi. Fluxor: detecting and monitoring fast- flux service networks. In *DIMVA 2008*, 2008.

[17] Van-Hau Pham, Marc Dacier, Guillaume Urvoy Keller, and Taoufik En Najjary. The quest for multi-headed worms. In *DIMVA 2008, 5th Conference on Detection of Intrusions and Malware & Vulnerability Assessment, July 10-11th, 2008, Paris, France*, Jul 2008.

[18] Fabien Pouget and Marc Dacier. Honeypot-based forensics. In *AusCERT2004, AusCERT Asia Pacific Information technology Security Conference 2004, 23rd - 27th May 2004, Brisbane, Australia*, May 2004.

[19] Niels Provos. A virtual honeypot framework. In *Proceedings of the 12th USENIX Security Symposium*, pages 1–14, August 2004.

[20] Moheeb Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis. A multifaceted approach to understanding the botnet phenomenon. In *ACM SIGCOMM/USENIX Internet Measurement Conference*, October 2006.

[21] Anirudh Ramachandran, Nick Feamster, and David Dagon. Revealing botnet membership using dnsbl counter-intelligence. In *SRUTI'06: Proceedings of the 2nd conference on Steps to Reducing Unwanted Traffic on the Internet*, pages 8–8, Berkeley, CA, USA, 2006. USENIX Association.

[22] Günther Starnberger, Christopher Krügel, and Engin Kirda. Overbot - A botnet protocol based on Kademlia. In *SecureComm 2008, 4th International Conference on Security and Privacy in Communication Networks, September 22-25th 2008, Istanbul, Turkey*, Sep 2008.

[23] Joe Stewart. Bobax trojan analysis. http://www.secureworks.com/research/threats/bobax, May 2004.

[24] Joe Stewart. Phatbot trojan analysis. http://www.secureworks.com/research/threats/sinit/, March 2004.

[25] W. Timothy Strayer, Robert Walsh, Carl Livadas, and David Lapsley. Detecting botnets with tight command and control. *Local Computer Networks, Proceedings 2006 31st IEEE Conference on*, pages 195–202, Nov. 2006.

[26] Ping Wang, Sherri Sparks, and Cliff C. Zou. An advanced hybrid peer-to-peer botnet. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 2–2, Berkeley, CA, USA, 2007. USENIX Association.

FIRE: FInding Rogue nEtworks

Brett Stone-Gross, Christopher Kruegel, Kevin Almeroth
University of California, Santa Barbara
{bstone,chris,almeroth}@cs.ucsb.edu

Andreas Moser
Technical University Vienna
andy@iseclab.org

Engin Kirda
Institute Eurecom
kirda@eurecom.fr

Abstract

*For many years, online criminals have been able to conduct their illicit activities by masquerading behind disreputable Internet Service Providers (ISPs). For example, organizations such as the Russian Business Network (RBN), Atrivo (*a.k.a.*, Intercage), McColo, and most recently, the Triple Fiber Network (3FN) operated with impunity, providing a safe haven for Internet criminals for their own financial gain. What primarily sets these ISPs apart from others is the significant longevity of the malicious activities on their networks and the apparent lack of action taken in response to abuse reports. Interestingly, even though the Internet provides a certain degree of anonymity, such ISPs fear public attention. Once exposed, rogue networks often cease their malicious activities quickly, or are de-peered (disconnected) by their upstream providers. As a result, the Internet criminals are forced to relocate their operations.*

In this paper, we present FIRE*, a novel system to identify and expose organizations and ISPs that demonstrate persistent, malicious behavior. The goal is to isolate the networks that are consistently implicated in malicious activity from those that are victims of compromise. To this end,* FIRE *actively monitors botnet communication channels, drive-by-download servers, and phishing web sites. This data is refined and correlated to quantify the degree of malicious activity for individual organizations. We present our results in real-time via the website maliciousnetworks.org. These results can be used to pinpoint and to track the activity of rogue organizations, preventing criminals from establishing strongholds on the Internet. Also, the information can be compiled into a null-routing blacklist to immediately halt traffic from malicious networks.*

1. Introduction

Anecdotal evidence indicates the existence of Internet companies and service providers that are under the influence of criminal organizations or knowingly tolerate their activities. Such companies typically control a number of networks with public IP addresses that are abused for a wide range of malicious activities. One such activity is offering bullet-proof hosting, a service that guarantees the availability of hosted resources even when they are found to be malicious or illegal. These hosting services are often used for phishing purposes or for serving exploits and malware. Other malicious activities involve the sending of spam, hosting scam pages, or providing a repository for pirated software and child pornography.

An example of a rogue network that offered bullet-proof hosting was the Russian Business Network (RBN), who made headlines in late 2007 [5], [16]. Various sources alleged that the RBN hosted web sites, exploits, and malware that were responsible for a significant fraction of online scams and phishing. Once publicly exposed, the RBN ceased its operations in St. Petersburg, only to relocate and resume activities in different networks [10]. More recently, a report exposed Atrivo (Intercage), a US-based company that is frequently considered to provide hosting for malicious content [3], [17]. Often referred to as the RBN of the United States, this company is considered to be a "dedicated crime hosting firm whose customer base is composed almost, or perhaps entirely, of criminal gangs" [13]. Shortly after Atrivo made headlines, two more rogue networks, known as McColo and the Triple Fiber Network (3FN), were discovered to be major hosting providers for malicious content with ties to cybercrime [1], [2], [18]. Again, public outcry quickly lead reputable ISPs to severe their peering relationships with these organizations, cutting them off the Internet.

In this paper, we describe *FIRE*, a system that monitors the Internet for rogue networks. We believe that it is important to expose such networks, for a number of reasons. First, as the examples of the Russian Business Network, Atrivo, McColo, and 3FN demonstrate, criminals fear public attention. As a result of the increased media coverage, all four networks had to cease their immediate activity. In many cases, it is likely that their operations resumed elsewhere. However, it took some time before the miscreants could restructure their setup. Thus, by quickly bringing to light networks that act maliciously, it becomes more difficult for cyber-criminals to establish a home base.

The second advantage of identifying rogue networks is the possibility to generate blacklists that can block all traffic from a netblock, even when certain IPs within this netblock have not yet acted maliciously. This approach prevents criminals from cycling through the available IP space, quickly shifting to a new IP when a current host is blacklisted. Currently, there are manual efforts underway to establish blacklists based on the observation that certain networks are malicious. For example, Spamhaus maintains the *Don't Route Or Peer (DROP)* list, a collection of networks that they consider to be controlled entirely by professional spammers. Spamhaus suggests that traffic from these sources should simply be dropped, and recommends the use of this list by tier-1 ISPs and backbone networks. Another example is the list maintained by EmergingThreats, which identifies netblocks that are thought to belong to the Russian Business Network. While such efforts are beneficial, they are expensive and tedious to maintain. Moreover, these lists are often incomplete and limited in scope (for example, limited to spam operations or the RBN in particular). In contrast, *FIRE* operates in an automated fashion, and we aim to capture a broader range of malicious activity, independent of any *a priori* knowledge of criminal organizations.

To identify rogue networks, we rely on a number of data sources that report the malicious actions of individual hosts. Some of the data feeds are publicly available, such as lists of phishing web pages. The other data originates from our own analysis efforts, such as a list of hosts that provide botnet command and control servers and hosts that are found to exploit browser vulnerabilities. Of course, given the widespread use of botnets and the large number of exploited machines, the fact that a host performs malicious actions is no immediate indication that the corresponding ISP or netblock is malicious. Instead, when a host misbehaves, it is possible that attackers were able to compromise and abuse it for nefarious purposes. Thus, it is necessary to search the data for indicators that allow us to distinguish between hosts under the control of rogue (or grossly negligent) ISPs and infected machines of organizations that make a deliberate effort to keep their network clean.

Based on post-processed information obtained from different data sources, we compute a *malscore* (maliciousness score) for individual ASNs (Autonomous System Number). This score quantifies the amount of recent, malicious activity in a network and serves as an indicator for the likelihood that an ASN is linked to cyber-criminals, or at the least, being very negligent in removing malicious content. Using the malscores, it is easy to identify the worst offenders on the Internet and take appropriate actions (such as increasing the public pressure, breaking peering relationships, or putting their IP address space on a blacklist). Moreover, we can track malicious activity over time.

The main contributions of this paper are as follows:

- We analyze a number of data sources to identify IP addresses of hosts that misbehave in different ways.
- We present techniques to filter these lists for hosts that likely belong to rogue ISPs. In particular, we combine the information from different data sources to compute a *malscore* that quantifies the malicious activities of an autonomous system.
- We show that our system is successful in identifying a number of rogue ISPs and can assist legitimate ISPs in cleaning their networks via our website maliciousnetworks.org.

2. System Overview

The goal of our system is to identify rogue networks. Thus, we first need to concretize what we consider to be a rogue network. Unfortunately, this question is not straightforward to answer. Some service providers are simply lax when it comes to the content that they offer, others are victims of remote exploits, and a few are well-known to blatantly host malicious content. Thus, the fact that a network is the source of unwanted activity does not necessarily qualify it immediately as being malicious.

We consider a rogue network to be a network that is under the control of cyber-criminals or that knowingly profits from cooperating with criminals. Of course, it is difficult to assert such criminal ties without thorough investigations by law enforcement agencies. Thus, we have to redefine our notion of rogue networks based on the activities that are typically associated with such networks. To this end, we consider a rogue network to be one in which significant malicious activity occurs. In addition, this activity lasts for an extended period of time, regardless of abuse complaints. Our logic behind this is that rogue networks provide hosting for malicious content that often remains up for many days (sometimes even months or years). In contrast, malicious activity in other networks tends to be more short-lived due to abuse reporting and honest attempts to undo the damage.

Given our notion of rogue networks, the basic idea to identify such networks is to check for the presence of a large number of long-lived, misbehaving hosts. To this end, we analyze a number of data sources for IP addresses that have exhibited malicious behavior for an extended period of time (the exact extent of this time span depends on the type of data source and is discussed later).

3. Data Collection

In this section, we discuss in more detail the three data sources that we use to identify hosts that likely belong to rogue networks. To this end, we first describe, for each data source, how we obtain the IP addresses of hosts that are *actively* engaged in malicious activity.

3.1. Botnet Command and Control Providers

Despite the emergence of peer-to-peer-based bots, many botnets still rely on centralized command and control (C&C). For this C&C infrastructure, botmasters typically set up IRC servers that provide channels for bots to join, or web servers that can be periodically polled for new commands. The functioning of the complete botnet depends on the availability of these servers. Thus, a botmaster is interested in hosting his C&C infrastructure on a network where it is safe from takedown.

To identify and monitor the networks affiliated with botnet C&C servers, we utilize data collected from Anubis [4]. Anubis executes Windows-based malware binaries in a virtual environment and records file system and registry modifications, process information, and network communications. We are particularly interested in the network traffic (if any) generated by the malware.

IRC-based botnets. When Anubis monitors IRC traffic the corresponding nickname, server, and channel information is logged. To monitor whether IRC C&C channels are active, we use a custom IRC client that leverages the recorded credentials to connect to the IRC server and join the channel. Because we are primarily interested in the longevity of the C&C server, we resolve the C&C server's host name to one or more IP addresses, and then connect to each IP at regular intervals. When the C&C server is not identified by a DNS name but by an IP address, then this address is used directly. A host (an IP address) is considered to be active when our client can join the corresponding C&C channel. Sometimes, transient network problems prevent us from connecting to a host. In such cases, it would be undesirable and premature to declare a host as inactive. Thus, we require that an active C&C channel is unreachable for two days before declaring the corresponding IP address as inactive.

Interestingly, in a number of cases, we observed that a channel (and the corresponding server) was reachable, but no malicious activity was noticeable. This is frequently the case when a bot channel is created on a well-known IRC network (such as undernet or efnet). The reason is that the IRC administrators of these networks quickly ban the botmaster and remove the channel. However, subsequent logins from bots or other users reopen the channel, thus making the channel available and leaving the impression that it is still active. To mitigate this problem, we modify our approach to determine whether a botnet C&C host is active. More precisely, in addition to the requirement that a server is reachable and the appropriate channel exists, we also require that the channel shows bot-related activity. To this end, we introduce heuristics that check the messages and channel topics for well-known IRC bot commands (such as *download*, *update*, *dos*) and signs of encoded or encrypted commands. A channel is considered up only when such indicators are present.

HTTP-based botnets. To identify and monitor web-based botnet C&C servers from samples collected by Anubis, we first require a mechanism to distinguish between legitimate HTTP traffic and traffic related to botnet commands. This is necessary because HTTP traffic sent by a malware sample does not immediately imply a connection to a C&C server (HTTP connections are often used to check for network connectivity or download updates). To identify HTTP C&C traffic, we manually define static, malicious characteristics (signatures) of requests used by well-known botnets. These characteristics include content from the HTTP request path and parameters, HTTP headers and POST data, and the HTTP response from the web server. Such static features are useful even for botnets that use encryption because they frequently send an encryption key, bot identifier, version number, and other parameters to the web server. Thus, the HTTP C&C server must know how to parse the request in a specific format.

As an example of a web-based botnet that we have been monitoring, consider Pushdo/Cutwail, which is believed to be one of the largest, active botnets used for spam. When a Cutwail bot connects to the C&C server, it will often request one or more executables. Although the botnet utilizes encryption, the request path for these binaries contains a predictable semi-static format, such as the prefix `/40E8`. The response from the web server contains one or more executables typically around 100KB. Currently, we are monitoring 24 different types of web-based botnets including Coreflood, Torpig, and Koobface.

3.2. Drive-by-Download Hosting Providers

Our second data source is a list of servers that host malware executables distributed through drive-by-download exploits. Drive-by-downloads are a means of malware distribution where executables are automatically installed on victim machines without user interaction. Typically, the only requirement is for a

user to visit a web page that contains an exploit for her vulnerable browser. In some cases, the exploit and the malware executable is hosted on a compromised host, while in other cases, a compromised web page is only used to redirect the victim to a second machine that performs the exploit (often referred to as a mothership). These mothership servers are frequently located in rogue networks.

There are three data feeds that we use to identify drive-by-download servers. The first feed is through Wepawet [25], a system that checks user-submitted web pages (URLs) for malicious Javascript. In particular, we are interested in cases where malicious script contains shellcode that downloads and executes malware. When malware is discovered, Wepawet records the locations of these binaries and exports them to *FIRE*. The second data feed is through a daily compilation of URLs found in spam mails that are caught in the spam traps of a computer security company and an Internet Service Provider. The third feed is a daily-updated list of "spamvertised" URLs (advertised via spam) provided by Spamcop [23]. So far, after eliminating duplicates, we have recorded more than 1.2 million spamvertised links. Of course, not every URL in a spam email points to a site that launches a drive-by exploit. Instead, these URLs frequently lead to shady businesses such as online pharmacies, casinos, or adult services. To identify those sites and pages that actively perform drive-by-exploits, we use the Capture Honey Pot Client (HPC) [21]. Capture is able to find web-based exploits by opening a potentially malicious web site in a browser on a virtual machine. After visiting a page, the state of the virtual machine is inspected and suspicious changes (i.e., the creation of new files or the spawning of new processes) are recorded, as they indicate that the guest system was compromised by a web-based exploit.

For our analysis, we use a total of eight virtual machines (VMs) dedicated to scanning web pages. All VM images are running Windows XP Professional (Service Pack 2), without any patches installed and automatic updates disabled. To catch recent exploits, we have installed the Flash and Quicktime plug-ins.

When the Capture honey client is compromised by visiting a certain URL, we inspect the network traces recorded from Capture HPC. We are not interested in the server that hosts the web site that contains an exploit. We have observed that those machines are often legitimate web servers that are victims of compromise and, therefore, do not yield much information about malicious networks. Thus, if the malicious binary that is part of an exploit is downloaded from the same server, we ignore that host for our analysis. In the more interesting case, an exploit has been injected into a web page and the associated binary is hosted on a different machine (mothership server that usually serves binaries for many different exploits). Due to the importance of this mothership servers for the criminals behind the exploit, these machines are often located in malicious networks where the chance that it is being shut down is low. Thus, we only consider the IP addresses of those mothership servers for our analysis. Once we have discovered a download server, we revisit it once per day.

3.3. Phish Hosting Providers

The third data source to identify rogue networks is derived from information about servers that host phishing pages. Typically, phishing pages are set up to steal login credentials, credit card numbers, or other personal information. Often, these pages are hosted on compromised servers and are taken down quickly. To mitigate this problem, phishers often resort to hosting their phishing pages directly in networks where there is little or no control of the offered content.

To locate phishing pages, we leverage an XML feed provided by PhishTank [19]. Once a day, this feed provides our system with URLs of phishing pages that are verified by the PhishTank community. Interestingly, all URLs on the PhishTank list are considered to be online. However, our experiments have shown that phishing pages are often taken offline so quickly that the list is already outdated after one day.

To compute the status of phishing IPs, we attempt to download the web page located at a given phishing URL once per day. This is done until either the domain (of the URL) can no longer be resolved, or the site is offline for more than one week. A phishing site is considered offline by our system when the web server is not reachable anymore or when the phishing page has been replaced by another page that is not a phish (usually a HTTP 404 error page or a phishing warning page).

4. Data Analysis

In this section, we discuss our techniques to identify rogue networks and compute their malscores based on the analysis of the individual data sets that we collect.

4.1. Longevity of Malicious IP Addresses

The primary characteristic that distinguishes between rogue and legitimate networks is the longevity of the malicious services. Most legitimate networks are able to clean up illicit content within a matter of days. In contrast, we have observed malicious content that

has been online for the entire monitoring period of more than a year. Figure 1 shows the average uptime of malicious IPs per ASN. It can be seen that the vast majority of networks remove the offending content in less than ten days. However, there were 361 ASNs that had hosts with an average lifespan of more than ten days in our feeds. Also, we discovered that each type of malicious activity displays different behaviors and average uptime.

Since May 2008, we have observed botnet C&C servers on 1,269 IP addresses. Figure 2 displays the uptime of the botnet C&C servers from 0-60 days. Note that we observed C&C servers that were online for more than 60 days, but limited the x-range of the graph to illustrate the rapid decline in botnet C&C servers that are taken down after only a few days, mainly by reputable IRC and web hosting providers.

We have been monitoring 1,161 of drive-by-download servers since August 2008. These servers have a much higher average lifetime than the other sources depicted in Figure 3. In fact, the number of drive-by-download servers that have been online for more than 60 days is 92, or more than 15%. Also, there have been 17 (approximately 3% of all) drive-by-download servers that have been online since the start of our collection.

From July 2008, we recorded 12,149 IP addresses hosting phishing websites. Similar to botnet C&C servers, the majority of phishing websites were online for only a few days. However, we also observed a few phishing sites that were online for more than a year. Figure 4 shows the uptime for the first 60 days for phishing hosts.

As mentioned previously, we use the longevity of malicious services as a distinguishing feature of rogue networks. This insight is supported by the previously-shown data, which demonstrates that a small number of ASNs is responsible for most persistent, malicious activity. To discard IPs that have been active for a short time only, we introduce a threshold δ. IP addresses that are active less than this threshold are not considered rogue and discarded from the subsequent malscore computation. This removes most of the phishing pages that are hosted on free web spaces or hacked machines, and legitimate IRC/web servers that are temporarily abused for botnet communications. As we will explain later in more detail (in Section 5.2), we do not use a threshold-based filter for drive-by-download servers. The reason is that such servers are difficult to set up, and thus, are typically a direct indication for rogue networks. This is also reflected in the uptime graph for drive-by download servers (Figure 3), which is different than the graphs for the other two data sources.

The output of the filtering step (which removes short-lived botnet C&C and phishing IPs) is a list of active, rogue IPs that constitute the input to the malicious score computation process, which is discussed in the next section. In Section 5.2, we will come back to the effects of selecting different values for the threshold δ on malscores and ASN ranks.

4.2. Malscore Computation

Once per day, the data collection process produces three lists $\mathcal{L}_i$ of active, rogue IPs (each derived from a different data source i). In the next step, the goal is to combine this information to expose organizations that act maliciously. For this, we consider an organization to be equivalent with an autonomous system (AS). An autonomous system is a group of a single entity (RFC 1771). Thus, it is a natural choice to perform analysis at the AS-level.

To identify those autonomous systems that are most likely malicious, we first map all IP addresses on the three lists to their corresponding ASN. For this, we query the `whois` database, selecting the most specific entry for an IP address in case multiple autonomous systems announce a particular IP. We are aware that the `whois` data might not be completely accurate. However, even in case of small errors, the database is sufficiently complete and precise to recognize the worst offenders.

A straightforward approach to identify those autonomous systems that are most malicious is to compute, for each AS, the sum of the IPs on the three lists that belong to this AS. While simple, this technique is not desirable because it ignores the size of a network. Clearly, when an AS P controls many more live hosts than AS Q, we can expect that the absolute number of malicious hosts in P are higher than in Q, even though the relative numbers might show the opposite. To avoid this pitfall, we compute the maliciousness score (malscore) $\mathcal{M}_A$ for an AS P as follows:

$$\mathcal{M}_P = \rho_P * \sum_{i=1}^{3} n_i(P) \quad (1)$$

In Equation 1, $n_i(P)$ is the number of IP addresses on list $\mathcal{L}_i$ that belong to the autonomous system P. Moreover, the malscore for each AS is adjusted by a factor ρ, which is indirectly proportional to the number of hosts in a network. That is, ρ decreases for larger networks.

The purpose of ρ is to put into relation the number of incidents with the number of active hosts in an autonomous system. This requires, for each AS, the knowledge of the number of live (active) hosts that

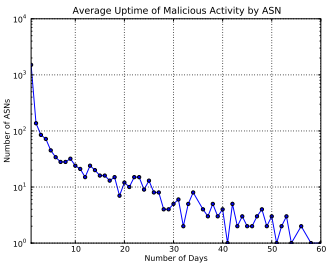


Figure 1: Average IP uptime by ASN.

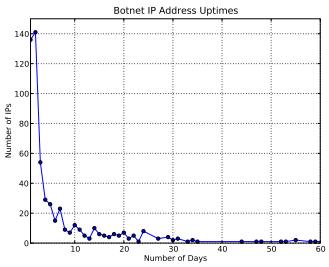


Figure 2: Botnet uptime between 0-60 days.

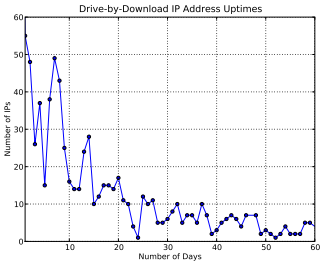


Figure 3: Drive-by uptime between 0-60 days.

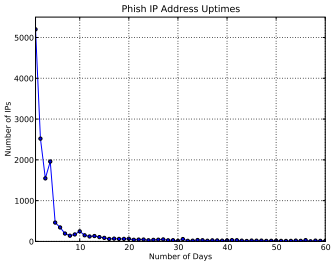


Figure 4: Phishing uptime between 0-60 days.

are operating in the networks of this AS. Clearly, this knowledge is difficult to obtain precisely, and it also can change over the course of several months. Previous work attempted to address this question [20], resorting to the idea of sending ping probes to a well-chosen subset of the IP addresses of a network. While these techniques can discriminate well between completely inactive (dark) regions and used networks, it is still quite difficult to determine the exact number of active hosts. Also, it is possible that networks are configured so that they do not respond to ping requests at all, thereby skewing the results. For these reasons, we decided to estimate the size of a network based on the size of the networks (i.e., the number of IP addresses) that an AS announces as routeable to the global Internet. To determine the size of the address space that an AS announces to the Internet, we leverage data provided by the Cooperative Association for Internet Data Analysis (CAIDA). CAIDA is a collaborative undertaking among organizations in the commercial, government, and research sectors that promotes cooperation in the engineering and maintenance of a robust, scalable, global Internet. In this role, CAIDA makes available a variety of data repositories that provide up-to-date measurements of the Internet infrastructure. One of these data repositories [14] shows a ranking of autonomous systems based on the size of their customer cones (address spaces). This information is compiled from RouteViews BGP tables.

We define $size_p$ as the number of /20 prefixes that an AS P announces. With this, we define ρ as shown in Equation 2 below. As desired, ρ decreases when $size_p$ increases.

$$\rho_p = 2^{-size_p/c}, \quad where \quad c = 4 \tag{2}$$

Of course, we are aware of the fact that the announced address space is not a perfectly reliable indicator for the number of active hosts. For example, there are network telescopes or educational institutions such as MIT that announce huge address ranges while having few or no live hosts. However, such networks are infrequent and, given the shortage of available IPv4 address space, many networks densely populate their available space. On the other hand, masquerading (network address translation - NAT) might result in multiple hosts sharing a singe IP address. Because of the imprecision that is inherent in estimating the number of active hosts, we limit the impact of $size$ on ρ by a parameter c. Empirically, we found that a value of $c = 4$ yields good results. In Section 5.2, we motivate this choice and discuss the influence of different values of c on our results.

5. Evaluation

In this section, we analyze the quality of our results. Moreover, we discuss in more detail the choice of im-

portant system parameters (such as the time threshold δ and size parameter c).

5.1. Analysis Results and Malicious Networks

Table 1 shows a snapshot of our system on June 1st, 2009, listing the ten entries with the largest malscores and the originating country (using the ip2location.com database). For this snapshot, we computed the maliciousness scores for all 417 autonomous systems that control at least one active, rogue IP.

Unfortunately, we do not have ground truth available that would allows us to evaluate the results of our system in a quantitative fashion. In fact, if such information would be available, then there would be no need for our system. Thus, we can only argue qualitatively that our system produces meaningful and interesting insights into the behavior of rogue networks.

Correctness of results. The top ten autonomous systems reported by *FIRE* on June 1st host a large number of persistent, malicious servers. In an attempt to confirm that our results are correct and meaningful, we leveraged a number of third party efforts that attempt to track down certain types of malicious activity on the Internet. More precisely, we first obtained a top-25 list, complied by the ShadowServer Foundation [22], that shows the most malicious networks with regards to botnet activity. Then, we looked at Google's Safe Browsing initiative [15] and extracted the top 150 ASNs, based on the absolute numbers of malicious drive-by servers that Google identified. In addition, we used the top-10 entries provided by ZeusTracker [26], a network that monitors and lists command and control servers for the Zeus botnet. Finally, we searched a number of blogs written by well-known security researchers for references to malicious and rogue ISPs and networks.

For each of our top ten entries, we then tried to find evidence in any of the third party lists that would confirm that a network is known to be rogue, or at least, strongly linked to certain malicious activities. Table 1 shows that we were successful for all ten entries.

In our list, IPNAP-ES (GigeNET) has consistently ranked among the top malicious networks, because it hosts the largest numbers of IRC botnet C&C servers. This is confirmed by the findings of ShadowServer. Some security forums have actually reported botnet activity from IPNAP as early as 2006. The Petersburg Internet Network (PIN), currently ranked second in Table 1, is known to be hosting the Zeus malware kit (also known as Zbot and WSNPoem).

It is also interesting to note that the "Novikov Aleksandr Leonidovich" AS has been linked to the recent Beladen drive-by-download exploit campaign [12], which is believed to be run by the same criminals that operated the Russian Business Network.

Completeness of results. In addition to checking our own top entries and comparing them to information from third parties, we also decided to analyze the top entries that these third parties have listed. This might allow us to find malicious networks that our analysis missed. In many cases, we found that malicious networks in those lists were also identified and prominently listed by *FIRE* (although, of course, not always in the top ten). This is especially true for Google's Safe Browsing list.

For the remaining entries that did not overlap with our results, we found that they mainly fit into two categories. In the first category, we find many large networks that were given an unfair bias in these lists due to the number of compromised hosts on their network. This includes large ISPs such as Cogent. We tagged these large networks with an X in each table to show that they are likely false positives. The second category consists of reputable networks that provide web and IRC hosting services (e.g., EUnet Finland hosts an IRC server for EFnet or FDCservers) with very short-lived malicious servers. That is, these networks just happen to be listed because they were under attack on a certain day, but they drop out quickly once the hosts or services are cleaned up. Thus, we believe that our results clearly show the importance of filtering ASNs by size and IP address longevity to accurately identify rogue networks while removing false positives.

5.2. Sensitivity of Important Parameters

Longevity thresholds. To distinguish between rogue and benign networks, *FIRE* uses thresholds δ based on the longevity of a malicious server. If a malicious host is online/active longer than this threshold, the IP is considered malicious. If a host is taken offline before it reaches the threshold, *FIRE* discards the corresponding IP for the malscore computation. The choices of the thresholds is thus important for the correctness of the analysis. If a threshold is selected too low, many compromised (but benign) hosts would be considered part of malicious networks. If the threshold is chosen too high, true malicious servers will be missed.

To quantify the influence of different thresholds on the results produced by *FIRE*, we introduce a simple distance metric between two rankings (i.e., lists of malicious networks sorted by malscore). This metric works by computing the edit distance between the two rankings A and B; that is, the distance between A and B is the number of insertions and deletions of ASNs that are needed to "convert" the ranking A into B.

Rank	ASN	Name	Country	Score	ShadowServer	Google	ZeusTracker	Blogs
1	AS23522	GigeNET	US	42.4	1	-	-	
2	AS44050	Petersburg Internet Network	UK	28.0	-	-	6	[9]
3	AS3595	Global Net Access	US	18.2	-	23	-	
4	AS41665	National Hosting Provider	ES	16.5	-	104	5	
5	AS8206	JUNIKNET	LV	14.1	-	30	-	
6	AS48031	Novikov Aleksandr Leonidovich	UA	14.0	-	-	-	[12]
7	AS16265	LEASEWEB	NL	13.0	24	14	-	
8	AS27715	LocaWeb Ltda	BR	11.6	-	130	-	
9	AS22576	Layered Technologies	US	11.5	-	64	-	[8]
10	AS16276	OVH OVH	FR	10.6	25	18	-	

Table 1: *FIRE* Top 10 for June 1st, 2009

ShadowServer Botnet C&Cs				Google Safe Browsing			
ASN	Name	FIRE Rank	Large Network	ASN	Name	FIRE Rank	Large Network
AS23522	GigeNET	1		AS4134	Chinanet Backbone No.31	17	X
AS3265	XS4ALL	118	X	AS21844	ThePlanet.com	13	
AS25761	Staminus Comm	-		AS4837	China169 Backbone	90	X
AS30058	FDCservers.net	-		AS36351	SoftLayer Technologies	30	
AS174	Cogent	148	X	AS26496	GoDaddy.com	15	X
AS2108	Croatian Research	-		AS41075	ATW Internet Kft.	23	
AS31800	DALnet	-	X	AS4812	Chinanet-SH-AP Telecom	89	X
AS13301	Unitedcolo.de	86		AS10929	Netelligent Hosting	12	
AS790	EUnet Finland	-		AS28753	Netdirect	11	
AS35908	SWIFT Ventures	68		AS8560	1&1 Internet AG	-	X

Table 2: ShadowServer Botnets / Google Safe Browsing Top 10 for June 1st, 2009

We then add to this value the number of those ASNs that appear in both rankings but that have a different number of rogue IPs.

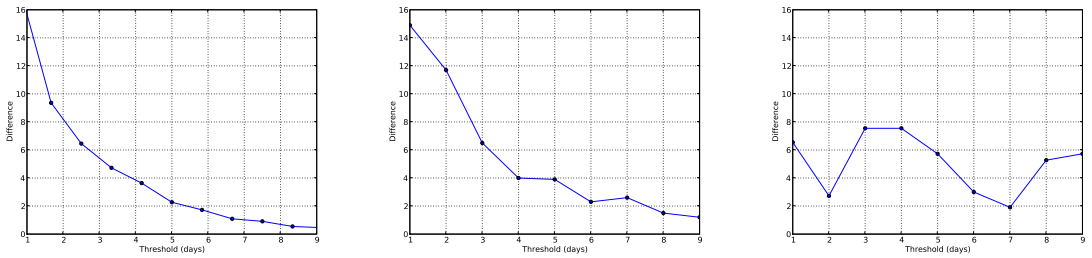
We used our metric to understand the influence of different threshold values on the result. To this end, we first calculated a ranking for a small threshold value. Then, we iteratively increased the threshold by a small value, recalculating the rankings at each step. Finally, we compare the rankings between each pair of subsequent steps. The idea is to see whether rankings eventually "stabilize," or whether they continuously fluctuate, depending on the specific values for δ.

We applied our analysis to all three data sources, ranging the threshold δ from 0 to 9. This was done for each day since January 1st, 2009, and the results were averaged. Figure 5 shows the results. Figures 5a and 5b indicate that for phishing servers and botnet control servers, there is significant fluctuation when threshold values are low. This is a direct result of the fact that these data sources contain many compromised servers that are taken offline after only one or two days by vigilant ISPs. Thus, we select the thresholds in a way that such compromised (but benign) servers are ignored. An ideal threshold value should be chosen high enough that the spikes at the beginning of both graphs are cut off, and the fluctuations around the threshold should be low. Thus, a threshold value that lies to the right of the initial peak in the curve is a good choice. Consequently, *FIRE* uses thresholds of $\delta_{phish} = 3$ and $\delta_{bot} = 4$.

For drive-by-download servers, we did not observe a stabilizing effect over time. On the contrary, Figure 5c shows a constant fluctuation. The reason is that most drive-by-download servers are not taken offline quickly. These servers are typically deployed by professional criminal organizations who do not want to risk that their exploits fail because the mothership server is taken offline. Thus, such servers are predominantly deployed in rogue networks. As a result, we do not take the uptime of drive-by-download servers into account when computing malscores.

Size parameter. As mentioned previously, *FIRE* decreases the malscores of large networks. This is to compensate for the fact that, due to their size, bigger networks are more likely to contain a significant number of rogue IPs. The extent to which the score of larger networks is decreased is influenced by the parameter c.

To show the effect of different choices for the parameter c, we calculated the rankings for varying values of this parameter. Again, we use the metric presented previously to quantify how changes of c influence the rankings. These result are shown in Figure 6. It can be seen that for c values (much) less than 1, the overall rank changes are small. This is due to the fact that, with small values for c, the resulting lists are dominated by ASN size, regardless



(a) Phishing Servers (b) Botnet Servers (c) Download Servers

Figure 5: Ranking changes for varying thresholds.

of the number of incidents. Similarly, for c values much greater than 1, the rankings are dominated by incident count, regardless of the size of a network.

For our analysis, it is thus important to choose a value for c that is located on the right side of the peak shown in the graph, as we want to favor incident count over network size. However, we are interested in a value for c that has some effect and, in particular, reduces the rank of very large networks (such as tier-1 ISPs and backbone networks). This lead to the choice of the threshold $c = 4$ for our malscore computation.

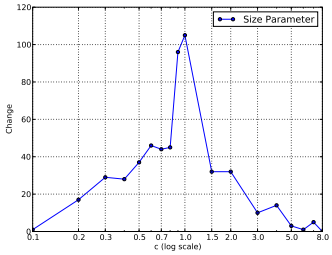


Figure 6: Sensitivity of parameter c.

6. Related Work

The work closest to ours are efforts that attempt to assign a reputation to networks or an individual IP address. In its simplest form, these efforts produce blacklists of IPs that have been observed to perform malicious actions. Most often, such blacklists are used to filter spam mails [23], [24], but there are also blacklists that warn users when they visit potentially harmful web pages [11], [19]. Many of the sites that offer blacklists also compile statistics of the worst offenders, typically by counting the number of incidents in a network. Unfortunately, this technique does not distinguish between compromised, bot-infected machines and hosts in networks that are deliberately malicious. As a result, the worst offenders are typically large networks with many customers. The goal of our work, on the other hand, is to discard the large amounts of compromised machines and identify those (often smaller) networks likely controlled by determined adversaries.

We are aware of two recent papers [6], [7] that look at temporal and spatial properties of attack sources. In [6], the authors study the spatial-temporal characteristics of malicious sources on the Internet, using data from the DShield.org project. The conclusion is that 20% of all IPs are responsible for 80% of the observed attacks. In [7], the authors attempt to find IPs that are clustered (spatial uncleanliness) and persistent (temporal uncleanliness) in sending spam mails, launching network scans, and hosting phishing pages. This work is closest to ours in that the behavior of hosts is used to identify "unclean" (infected) netblocks. The difference to our approach is twofold: First, we attempt to identify networks that are operated by criminals, while their work was focusing on finding bot infections. As a result, the selection of the input data sets (we include drive-by download providers and botnet C&C servers, but do not consider scanning) and the filtering techniques are different. Moreover, we combine results from multiple feeds. Such correlation efforts were not part of the previous paper.

7. Conclusions

In this paper, we presented *FIRE*, a novel system to automatically identify and expose organizations and ISPs that demonstrate persistent, malicious behavior. *FIRE* can help isolate networks that tolerate and aid miscreants in conducting malicious activity on the Internet. It does this by actively monitoring different data sources such as botnet communication channels, drive-by-download servers, and feeds from phishing web sites. Because it is important to distinguish between networks that are knowingly malicious and networks that are victims of compromise, we refine the collected

data and correlate it to deduce the level of maliciousness for the identified networks. Our ultimate aim is to automatically generate results that can be used to pinpoint and track organizations that support Internet miscreants and to help report and prevent criminal activity. Furthermore, the networks we identify can also be used by ISPs as blacklists in order to simply block traffic that is originating from them. Hence, an ISP can enhance the security of its users by not allowing malicious traffic to reach them.

Acknowledgments

The research was supported by the National Science Foundation under grant CNS-0831408 and the WOMBAT project sponsored by the EU commission.

References

[1] J. Armin, G. Bruen, G. Feezel, P. Ferguson, M. Jonkman, and J. McQuaid. McColo - Cyber Crime USA. http://hostexploit.com/downloads/Hostexploit%20Cyber%20Crime%20USA%20v%202.0%201108.pdf, 2008.

[2] J. Armin, P. Ferguson, G. Bruen, G. Feezel, M. Jonkman, and J. McQuaid. McColo - Cyber Crime USA Supplement. http://hostexploit.com/downloads/Hostexploit McColo supplement 111808.pdf, 2008.

[3] J. Armin, J. McQuaid, and M. Jonkman. Atrivo - Cyber Crime USA. http://hostexploit.com/downloads/Atrivowhitepaper082808ac.pdf, 2008.

[4] U. Bayer, C. Kruegel, and E. Kirda. TTAnalyze: A Tool for Analyzing Malware. In *EICAR Conference*, 2006.

[5] D. Bizeul. Russian Business Network Study. http://www.bizeul.org/files/RBN study.pdf, 2007.

[6] Z. Chen, C. Ji, and P. Barford. Spatial Temporal Characteristics of Internet Malicious Sources. In *Infocomm Mini-Conference*, 2008.

[7] M. Collins, T. Shimeall, S. Faber, J. Janies, R. Weaver, and M. D. Shon. Using Uncleanliness to Predict Future Botnet Addresses. In *ACM Internet Measurement Conference (IMC)*, 2007.

[8] D. Danchev. The Malicious ISPs You Rarely See in Any Report. http://ddanchev.blogspot.com/2008/06/malicious-isps-you-rarely-see-in-any.html, 2008.

[9] D. Danchev. GazTransitStroy/GazTranZitStroy Rubbing Shoulders with Petersburg Internet Network LLC. http://ddanchev.blogspot.com/2009/06/gaztransitstroygaztranzitstroy-rubbing.html, 2009.

[10] dn1nj4. The_Shadowserver Foundation: RBN "Rizing". http://www.shadowserver.org/wiki/uploads/Information/RBN Rizing.pdf, 2008.

[11] D. Glosser. DNS-BH - Malware Domain Blocklist. http://malwaredomains.com/, 2008.

[12] D. Goodin. 40,000 sites hit by PC-pwning hack attack. http://www.theregister.co.uk/2009/06/02/beladen mass website infection/, 2009.

[13] V. Hanna. Spamhaus: Cybercrime's U.S. Hosts. http://www.spamhaus.org/news.lasso?article=636, 2008.

[14] B. Huffaker. CAIDA: AS ranking. http://as-rank.caida.org/, 2008.

[15] G. Inc. http://google.com/safebrowsing/diagnostic?site=AS:27715, 2009.

[16] B. Krebs. Taking on the Russian Business Network. http://voices.washingtonpost.com/securityfix/2007/10/taking on the russian business.html, 2007.

[17] B. Krebs. Report Slams U.S. Host as Major Source of Badware. http://voices.washingtonpost.com/securityfix/2008/08/report slams us host as major.html, 2008.

[18] B. Krebs. FTC Sues, Shuts Down N. Calif. Web Hosting Firm. http://voices.washingtonpost.com/securityfix/2009/06/ftc sues shuts down n calif we.html, 2009.

[19] PhishTank. Clearinghouse for phishing data on the Internet. http://www.phishtank.com, 2008.

[20] M. Rajab, F. Monrose, and A. Terzis. Fast and Evasive Attacks: Highlighting the Challenges Ahead. In *International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2006.

[21] C. Seifert. Capture-HPC - Honeypot Client. https://projects.honeynet.org/capture-hpc, 2008.

[22] Shadowserver. ASN Botnet Stats. http://www.shadowserver.org/wiki/pmwiki.php/Stats/ASN, 2009.

[23] SpamCop. Blocking List. http://www.spamcop.net/bl.shtml, 2008.

[24] Spamhaus. Zen: Comprehensive DNSBL. http://www.spamhaus.org/zen/, 2008.

[25] Wepawet. http://wepawet.iseclab.org/, 2009.

[26] ZeuSTracker. https://zeustracker.abuse.ch/statistic.php, 2009.

The WOMBAT Attack Attribution method: some results

Marc Dacier[1], Van-Hau Pham[2], and Olivier Thonnard[3]

[1] Symantec Research
Sophia Antipolis, France
marc_dacier@symantec.com
[2] Institut Eurecom
2229 Route des Crètes,
Sophia Antipolis, France
van-hau.pham@eurecom.fr
[3] Royal Military Academy
Polytechnic Faculty
Brussels, Belgium
olivier.thonnard@rma.ac.be

Abstract. In this paper, we present a new *attack attribution* method that has been developed within the WOMBAT[4] project. We illustrate the method with some real-world results obtained when applying it to almost two years of attack traces collected by low interaction honeypots. This analytical method aims at identifying large scale attack phenomena composed of IP sources that are linked to the same root cause. All malicious sources involved in a same phenomenon constitute what we call a *Misbehaving Cloud* (MC). The paper offers an overview of the various steps the method goes through to identify these clouds, providing pointers to external references for more detailed information. Four instances of misbehaving clouds are then described in some more depth to demonstrate the meaningfulness of the concept.

1 Introduction

There is no real consensus on the definition of "attack attribution" in the cyber domain. Most previous work related to that field tend to use the term "attribution" as a synonym for *traceback*, which consists in "determining the identity or location of an attacker or an attacker's intermediary" [25]. In the context of a cyber-attack, the obtained identity can refer to a person's name, an account, an alias, or similar information associated with a person or an organisation. The location may include physical (geographic) location, or any virtual address such as an IP address or Ethernet address. The rationale for developing such attribution techniques is mainly due to the untrusted nature of the IP protocol, in which the source IP address is not authenticated and can thus be easily spoofed.

[4] Worldwide Observatory of Malicious Behaviors and Threats - http://www.wombat-project.eu

An extensive survey of attack attribution techniques used in the context of IP traceback can be found in [25].

In this paper, we refer to "attack attribution" as something quite different from what is described here above. We are primarily concerned with larger scale attacks. In this context, we aim at developing an analytical method to help security analysts in determining their root causes and in deriving their *modus operandi*. These phenomena can be observed through many different means (e.g., honeypots, IDS's, sandboxes, web crawlers, malware collecting systems, etc). In most cases, we believe that attack phenomena manifest themselves through so-called "attack events", which can be observed with distributed sensors that are deployed in the Internet. Typical examples of attack phenomena that we want to identify vary from worm or malware families that propagate through code injection attacks [9], to established botnets controlled by the same people and targeting machines in the IP space. All malicious sources involved in the same root phenomenon constitute what we call a *Misbehaving Cloud* (MC).

The structure of the paper is as follows: Section 2 describes the experimental environment used to validate the method presented. Section 3 offers a high level overview of the attack attribution method defined within the WOMBAT project and Section 4 gives some more information on the multi criteria fusion approach used in the method. Section 5 discusses a couple of illustrative examples obtained by applying the method on honeynet traces, and Section 6 concludes the paper.

2 Description of the experimental environment

This paper offers an empirical analysis of some attacks collected during two years by a set of low interaction honeypots deployed all over the world by the Leurré.com Project [10]. We refer the interested reader to [8, 19] for an in-depth presentation of the data collection infrastructure. From an analytical viewpoint, our attack attribution method builds upon previous results, namely [18, 4, 16, 24, 17]. For the sake of clarity, we start by introducing some important terms that have been defined in these previous publications.

2.1 Terminology

1. **Platform**: A physical machine running three virtual honeypots, which emulate three distinct machines thanks to *honeyd* [20]. A platform is connected directly to the Internet and collects tcpdump traces that are gathered on a daily basis in a centralized database [10].
2. **Source**: an IP address that has sent at least one packet to, at least, one platform. An IP address remains associated to a given Source as long as no more than 25 hours[5] elapse between two packets sent by that IP. After such

[5] By grouping packets by originating sources instead of by IPs, we minimize the risk of mixing together the activities of two distinct physical machines (as a side effect of the dynamic address allocation implemented by ISP's).

a delay, the IP will be associated to a new source identifier if we observe it again.

3. **Attack**: refers to all packets exchanged between a malicious source and a platform.
4. **Cluster**: all the sources that execute the same attack against any of the platforms constitute an *(attack) Cluster*. In practice, such a cluster groups all malicious sources that have left highly similar network traces on our platforms. How to identify clusters and how those clusters look like are issues that have been explained in other publications [18, 8].

2.2 Honeynet dataset

Machines used in the Leurré.com project are maintained by partners all over the world, on a voluntary basis. Some of these platforms can thus become unavailable. In the context of this paper, we wanted to apply our analytical method on a dataset that would be, as much as possible, unimpacted by these operational issues. Therefore, we have selected a subset of 40 stable platforms from all platforms at our disposal. A total of 3,477,976 attacks have been observed by those platforms. We represent the total number of attacks per day over the whole analysis period (800 days, from Sep 2006 until November 2008), as a time series denoted by TS. Similarly, we can represent, for each platform, the number of attacks observed on it, on a daily basis. This leads to the definition of 40 distinct attack time series (each made of 800 points), denoted by TS_X where X represents a platform identifier.

We can go even further in splitting our time series in order to represent which type of attack was observed on which platform. To do this, we split each TS_X into as many time series as there are *attack clusters*, as defined before. These newly obtained time series are represented by $\Phi_{[0-800),c_i,p_j} \forall \,\text{cluster}\, c_i$ and $\forall \,\text{platform}\, p_j$. That is, the i^{th} point of the time series $\Phi_{[0-800),X,Y}$ represents the amount of sources attacking, on day i, the platform Y by means of the attack defined by the cluster identifier X. We represent by TS_L2 the set of all these observed cluster time series (in total, 395,712 time series).

In [17], it has been shown that a large fraction of these time series barely vary in amplitude on a daily basis. This continuous, low-intensity activity is also referred to as the Internet *background radiation* [13]. In this paper, we do not consider those flat curves, and we instead focus on time series that show some significant variations over time, indicating the existence of some ephemeral phenomena. To automatically identify these time series of interest, we have applied the method presented in [17], which finally gives a subset of time series denoted by TS_L2'. In our dataset, TS_L2' contains now only 2,127 distinct time series. However, they still comprise a total of 2,538,922 malicious sources. TS_L2' represents the set of time series we have used for this analysis.

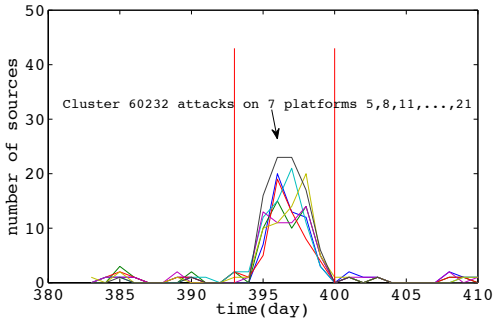


Fig. 1. An example of $\mathcal{M}$-event, composed of seven μ-events (on seven different platforms) that are correlated in the same time interval. Cluster 60332 corresponds to a malicious activity on the VNC port (5900/TCP).

3 Overview of WOMBAT attribution method

The WOMBAT attack attribution method is made of two distinct steps. In the very first one, we identify periods of time where some of the time series from TS_L2' exhibit a pattern that indicate that a specific phenomenon worth of interest is happening. We call a *micro attack event* such period of time for a given time series from TS_L2'. Moreover, we call *macro attack event* a group of micro attack events that are correlated during the same period of time.

The second step of the method consists in characterizing each of these *micro attack events* and in trying to establish connections between them. All micro attack events that share enough features constitute what we call a *Misbehaving Cloud* (MC). We hypothesize that all malicious sources involved in a Misbehaving Cloud have a common root cause. By identifying them and studying their global behavior, we hope to get a better insight into the modus operandi and the strategies of those responsible for them.

We further detail the two steps of the method in the next subsections.

3.1 Step 1: Micro and Macro attack events identification

Definition (μ-event): A *micro attack event* (or μ-event) is defined by a tuple $(\mathcal{T}, \mathcal{C}_i)$ where $\mathcal{T}$ represents a limited period of time (typically a few days) during which a significant attack activity is observed, and $\mathcal{C}_i$ represents the time series corresponding to cluster $\mathcal{C}$ observed on the platform i.
Definition ($\mathcal{M}$-event): A set of micro attack events observed over the same period of time, and during which the corresponding time series are strongly correlated is defined as a *macro attack event* (or $\mathcal{M}$-event).

Figure 1 illustrates this concept by representing a $\mathcal{M}$-event composed of seven μ-events that are correlated in the same time interval.

Identification of μ-events. The micro attack event identification relies mostly on some well-known signal processing techniques. The goal is to segment the time series into periods of interest. Such periods are characterized by some *intense period* of activities isolated by periods of very stable or non existent activities. Several techniques exist to detect abrupt changes in a signal [1]. In this paper, the method we have used is the one that has been precisely presented in [15].

Identification of $\mathcal{M}$-event. Once we have identified all μ-events of interest in our dataset, we need to identify all those that are strongly correlated over the same period of time, which form thus a $\mathcal{M}$-event. The problem is not as trivial as it may sound, because *i)* μ-events may have overlapping periods, and *ii)* within a given period of time, several distinct phenomena may have taken place. Here too, we have presented and compared various approaches and we refer the interested reader to [17, 15] for an in-depth explanation of the algorithms used.

3.2 Step 2: Multi criteria fusion of attack events features

The purpose of this second step consists in deciding whether several distinct μ-events are likely due to a same root phenomenon (i.e., the same Misbehaving Cloud), on the basis of different characteristics derived from the network traffic generated by malicious sources involved in such events.

Our approach is based on three components:

1. Attack Feature Selection: we determine which *attack features* we want to include in the fusion process, and we thus characterize each μ-event according to this set of features;
2. Graph-based Clustering: a graph of μ-events is created regarding each feature, based on an appropriate distance for measuring pairwise similarities. Fully connected components can then be identified within each graph;
3. Multi criteria fusion: the different graphs are then combined using an *agregation function* that models some dynamic behavior.

This approach is mostly unsupervised, i.e., it does not rely on a preliminary training phase to attribute μ-events to larger scale phenomena. In the next Section, we describe the three steps of this method.

4 On the Multi criteria fusion approach

4.1 Attack Features Selection

In most clustering tasks, the very first step consists in selecting some key characteristics from the dataset, i.e., salient features that may reveal meaningful *patterns* [6]. In this analysis, we have selected some *features* that we consider useful to analyze the behavior of global phenomena.

One of the key features used in this attribution technique is the spatial distributions of malicious sources involved in μ-events, in terms of originating countries and IP blocks. Looking at these statistical characteristics may reveal attack

activities having a specific distribution of originating countries or IP networks, which can help for instance to confirm the existence of "unclean networks" [3]. In practice, for each μ-event, we create a feature vector representing the distribution of countries of sources (as a result of the IP to geolocation mapping), or a vector representing the distribution of IP addresses (grouped by their Class A-prefix, to limit the vector's size).

We have also selected an attack characteristic related to the *targeted platforms*. Looking at which specific platform has observed a μ-event is certainly a pertinent feature. At the same time, we combine this information with the $\mathcal{M}$-event identification, since (by definition) $\mathcal{M}$-events are composed of μ-events that are strongly correlated in time (which indicates a certain degree of coordination among them).

Besides the origins and the targets, the *type of activity* performed by the attackers seems also relevant. In fact, worm or bot software is often crafted with a certain number of available exploits targeting a given set of TCP or UDP ports. So, it makes sense to take advantage of similarities between the *sequences of ports* that have been probed or exploited by malicious sources.

Finally, we have decided to compute, for each pair of μ-events, the ratio of common IP addresses. We are aware of the fact that, as time passes, some machines of a given botnet (or misbehaving cloud) might be cured while others may get infected (and thus join the cloud). Additionally, certain ISPs apply a quite dynamic policy of IP allocation for residential users, which means that infected machines can have different IP addresses when we observe them at different moments. Nevertheless, considering the huge size of the IP space, it is still reasonable to expect that two μ-events are probably related to the same root phenomenon when they have a high percentage of IP addresses in common.

To summarize, and to provide a short-hand notation in the rest of this paper, for each μ-event we define a set of features that we denote by:

$$F = \{F_i\}\,,\ i \in \{geo, sub, targ, ps, cip\}$$

where:

$$\begin{cases} geo = \text{geolocation, as a result of mapping IP addresses to countries;} \\ sub = \text{distribution of sources IP addresses (grouped by Class A-subnet);} \\ targ = \text{targeted platforms} + \text{degree of coordination } (\mathcal{M}\text{-event membership);} \\ ps = \text{port sequences probed or targeted by malicious sources;} \\ cip = \text{feature representing the ratio of common IP addresses among sources;} \end{cases}$$

4.2 Graph-based Clustering

The second component of our attribution method implements an unsupervised clustering technique that aims at discovering groups of strongly connected μ-events, when these are represented within a graph. In [22, 23], we have given a detailed description of this graph-based clustering technique. However, to make this paper as self-contained as possible, we briefly describe the high-level principles of this technique.

As defined by Jain and Dubes in [6], many typical clustering tasks involve the following steps:

i) feature selection and/or extraction (as described in the previous Subsection);
ii) definition of an appropriate distance for measuring the similarities between pairs of elements with respect to a given feature;
iii) application of a grouping algorithm, such as the classical hierarchical clustering or K-means algorithm;
iv) data abstraction (if needed), to provide a compact representation of each cluster;
v) optionally, the assessment of the clusters quality and coherence, e.g. by means of validity indices.

Steps (iv) and (v), while important, lie outside the scope of this paper. Instead, we will simply use four anecdotal examples to intuitively demonstrate the quality, i.e., the meaningfulness, of the groups created by the method. Steps (ii) and (iii) are described here after.

Choosing a distance function How to measure *pairwise similarities* between two feature vectors is obviously an important step, since it will have an impact on the coherence and the quality of the resulting clusters.

When we have to deal with observations that are in the form of probability distributions (or frequencies), like in the case of features F_{geo} and F_{sub}, we need to rely on statistical distances. One commonly used technique is the Kullback-Leibler divergence [7]. Let p_1 and p_2 be for instance two probability distributions over a discrete space X, then the K-L divergence of p_2 from p_1 is defined as:

$$D_{KL}(p_1||p_2) = \sum_x p_1(x) \log \frac{p_1(x)}{p_2(x)} \tag{1}$$

which is also called the information divergence (or *relative entropy*). Because D_{KL} is not considered as a true metric, it is usually better to use instead the Jensen-Shannon divergence (JSD) [11], defined as:

$$JS(p_1, p_2) = \frac{D_{KL}(p_1||\bar{p}) + D_{KL}(p_2||\bar{p})}{2} \tag{2}$$

where $\bar{p} = (p_1 + p_2)/2$. In other words, the Jensen-Shannon divergence is the *average* of the KL-divergences to the *average distribution*.

Finally, to transform pairwise distances d_{ij} to similarity weights sim_{ij}, we still have to define a mapping function. Previous studies found that the similarity between stimuli decay exponentially with some power of the perceptual measure distance [21]. As customary, we can thus use the following functional form to do this transformation:

$$sim(i,j) = exp(\frac{-{d_{ij}}^2}{\sigma^2}) \tag{3}$$

where σ is a positive real number that affects the decreasing rate of w.

Measuring pairwise similarities for the other considered features ($F_{targ}, F_{ps}, F_{cip}$) is more straightforward. In those cases, we can use simpler distance functions, such as the *Jaccard similarity coefficient*. Let s_1 and s_2 be two sample sets (for instance with F_{ps}, s_1 and s_2 are sets of ports that have been probed by sources of two μ-events), then the Jaccard coefficent is defined as the size of the intersection divided by the size of the union of the sample sets, i.e.:

$$sim(i,j) = \frac{|s_1 \bigcap s_2|}{|s_1 \bigcup s_2|}$$

The Jaccard similarity coefficient can also be used to compute the ratio of common IP addresses between attack events (F_{cip}). Regarding F_{targ}, a simple weighted means is used to combine two scores: *i)* one score in $[0, 1]$ as given by the simple comparison of the two targeted platforms, and *ii)* another score (also in $[0, 1]$) indicating whether two μ-events belong to the same $\mathcal{M}$-event (indicating a time coordination).

Grouping algorithm In this step, we formulate the problem of clustering μ-events using a graph-based approach. The vertices (or nodes) of the graph represent the patterns (or feature vectors) of the μ-events, and the edges (or links) express the similarities between μ-events, as calculated with the distance metrics described before. Then, we can extract so-called *maximal cliques* from the graph, where a maximal clique is defined as an induced subgraph in which the vertices are fully connected and it is not contained within any other clique. To do this, we use the *dominant sets* approach of Pavan et al. [14], which proved to be an effective method for finding maximal *weighted* cliques. This means that the weight of every edge (i.e., the relative similarity value) is also considered by the algorithm, as it seeks to discover maximal cliques whose total weight is maximized.

By repeating this process, we can thus create an undirected edge-weighted graph G_i for each attack feature F_i, in which the edges are similarity weights $\in [0, 1]$ that can be seen as *relatedness degrees* between μ-events (where a zero value indicates totally unrelated events). Then, the clique algorithm extracts one set of cliques per feature, which reveals the cohesions among μ-events regarding each F_i.

4.3 Multi-Criteria Aggregation

Definition (Aggregation function). An aggregation function is formally defined as a function of n arguments ($n > 1$) that maps the (n-dimensional) unit cube onto the unit interval: $f : [0, 1]^n \longrightarrow [0, 1]$, with the following properties [2]:

(i) $f(\underbrace{0, 0, \ldots, 0}_{n\text{-}times}) = 0$ and $f(\underbrace{1, 1, \ldots, 1}_{n\text{-}times}) = 1$

(ii) $x_i \leq y_i$ for all $i \in \{1, \ldots, n\}$ implies $f(x_1, \ldots, x_n) \leq f(y_1, \ldots, y_n)$

Aggregation functions are used in many prototypical situations where we have several criteria of concern, with respect to which we assess different options. The objective consists in calculating a combined score for each option, and this combined output forms then a basis from which decisions can be made. For example, aggregation functions are largely used in problems of *multi criteria decision analysis* (MCDA), in which an alternative has to be chosen based on several, sometimes conflicting criteria. Usually, the alternatives are evaluated from different attributes (or features) that are expressed with numerical values representing a degree of preference, or a degree of membership.

In our application, we have n different attack features given by the F_i's, and thus a vector of criteria $\mathbf{x} \in [0,1]^n$ can be constructed from the similarity weights, i.e., $x_i = A_i(j,k)$, with A_i being the similarity matrix of graph G_i corresponding to attack feature F_i. Our approach consists in combining the n values of each criteria vector $\mathbf{x}$ (which reflect the set of all relationships between a pair of μ-events), in order to build an aggregated graph $G' = \sum G_i$ from which we can then extract the connected components. A straightforward but rather simplistic approach would consist in combining the criteria using a simple arithmetic mean, or by assigning different weights to each criteria (weighted mean). However, this does not allow us to model more complex behaviors, such as "most of", or "at least two" criteria to be satisfied in the overall decision function. Yager has introduced in [26] a type of operator called *Ordered Weighted Averaging* (OWA), which allows to include certain relationships between multiple criteria in the aggregation process. An OWA aggregation operator differs from a classical weighted means in that the weights are not associated with particular inputs, but rather with their *magnitude*. As a result, OWA can emphasize the largest, smallest or mid-range values. It has become very popular in the research community working on fuzzy sets.

Definition (OWA). For a given weighting vector $\mathbf{w}$, $w_i \geq 0$, $\sum w_i = 1$, the OWA aggregation function is defined by:

$$OWA_w(\mathbf{x}) = \sum_{i=1}^{n} w_i x_{\searrow(i)} = < \mathbf{w}, \mathbf{x}_{\searrow} > \tag{4}$$

where we use the notation $\mathbf{x}_{\searrow}$ to represent the vector obtained from $\mathbf{x}$ by arranging its components in decreasing order: $x_{(1)} \geq x_{(2)} \geq \ldots \geq x_{(n)}$.

It is easy to see that for any weighting vector w, the result of OWA lies between the classical **and** (=min) and **or** (=max) operators, which are in fact the two extreme cases when $\mathbf{w} = (0, 0, \ldots, 1)$ (then $OWA_w(\mathbf{x}) = min(\mathbf{x})$) or when $\mathbf{w} = (1, 0, \ldots, 0)$ (then $OWA_w(\mathbf{x}) = max(\mathbf{x})$). Another special case is when all weights $w_i = \frac{1}{n}$, which results in obtaining the classical arithmetic mean.

To define the weights w_i to be used in OWA, Yager suggests two possible approaches: either to use some learning mechanism with sample data and a regression model (i.e., fitting weights by using training data and minimizing the least-square residual error), or to give some semantics to the w_i's by asking an expert

to provide directly those values, based on domain knowledge. We selected the latter approach by defining the weighting vector as $\mathbf{w} = (0.1, 0.35, 0.35, 0.1, 0.1)$, which translates our intuition about the dynamic behavior of large-scale attack phenomena. It can be interpreted as: *at least three criteria must be satisfied, but the first criteria is of less importance compared to the* 2^{nd} *and* 3^{rd} *ones* (because only one correlated feature between two μ-events might be due to chance only).

These weights must be carefully chosen in order to avoid an unfortunate linkage between μ-events when, for example, two events involve IP sources originating from popular countries and targeting common (Windows) ports in the same interval of time (but in reality, those events are not due to the same phenomenon). By considering different worst-case scenarios, we verified that the values of the weighting vector $\mathbf{w}$ work as expected, i.e., that it minimizes the final output value in such undesirable cases. Moreover, these considerations enable us to fix our decision threshold to an empirical value of about 0.25, which has been also validated by a sensibility analysis. In other words, all combined values that are under this threshold will be set to zero, leading to the removal of corresponding edges in the aggregated graph G'.

Finally, we can easily identify misbehaving clouds by extracting the connected components (or subgraphs) from G'. As a result, for any subset of events of a given MC, we will find a sufficient number of evidences that explain why those events have been linked together by the multi criteria aggregation process.

5 Experimental Results

5.1 Overview

When applying the technique described in Section 3.1 to the dataset described in Section 2.2, we obtain 690 $\mathcal{M}$-events which consist of 2454 μ-events. We use these μ-events as input for the multi-criteria fusion approach (Section 4), and we consequently identify 83 Misbehaving Clouds (MCs), which correspond to 1607 μ-events, and 506,835 attacking sources. The phenomena involve almost all common services such as NetBios (ports 139/TCP, 445/TCP), Windows DCOM Service (port 135/TCP), Virtual Network Computing (port 5900/TCP), Microsoft SQL Server (port 1433/TCP), Windows Messenger Service (ports 1025-1028/UDP), Symantec Agent (port 2967/TCP), and some others. Figure 7a shows the distribution of μ-events per MC. As we can see, in most cases, the MCs contain few μ-events. However, around 20% of MCs contain more than 15 μ-events, and some even contain up to 300 events. Figure 7b represent the CDF of the MCs lifetime. Such lifetime is defined as the time interval, in days, between the very first and the very last attack event of a given MC. As showed in Figure 7b, 67% of MCs exist during less than 50 days but around 22% of them last for more than 200 days.

Figure 7c represents the CDF of the number of platforms targeted by MC. As showed in the Figure, in 94% of the cases, the MCs are seen on less than 10 platforms.

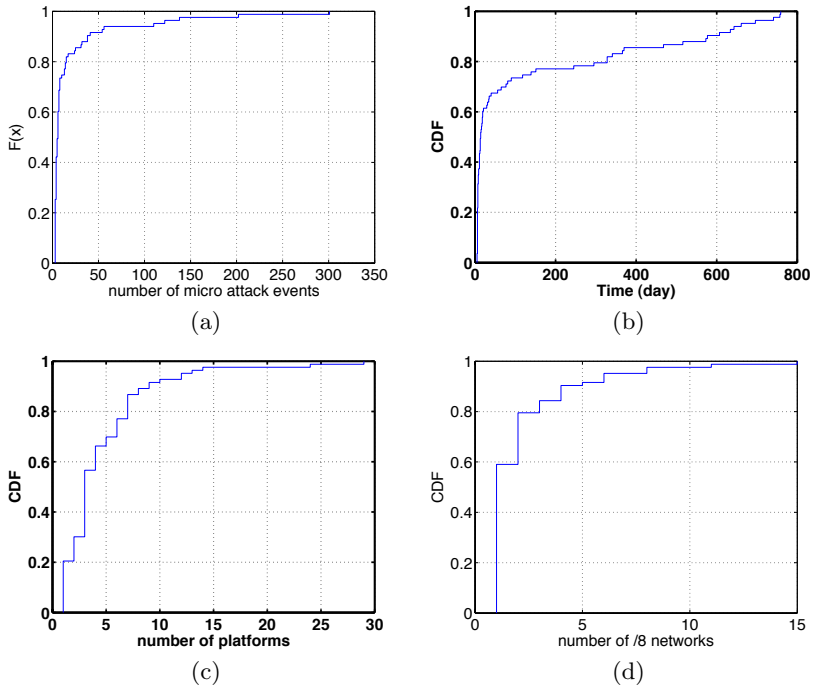


Fig. 2. Some global characteristics of the obtained MCs

These various characteristics suggest that the root causes behind the existence of these MCs are fairly stable, localised attack processes. In other words, different places of the world do observe different kind of attackers but their modus operandi remain stable over a long period of time. We are, apparently, not that good at stopping them from misbehaving.

5.2 Case Studies

It is certainly not our intention to detail extensively the behavior and characteristics of every MC that has been found in our 2-year data set. Instead, in this Section, we detail only four MCs, which, although anecdotal, still reflect the kind of findings that our method can provide automatically. Table 1 provides some high-level characteristics of these four MCs phenomena under study. Each MC is analyzed in some detail in the following pages.

***MC*2: Worm-behaving cloud.** $MC2$ consists of 122 μ-attack events. These μ-events exhibit a shape which is fairly similar to the one left by a typical worm: its trace exists for several days, it has a small amplitude at the beginning but grows quickly, exhibits important drops that can correspond to subnets being

Table 1. High-level characteristics of four *MCs* under study. The colon *Root cause* refers to the presumed type of phenomenon, based on the results of the attack attribution method.

MC Id Nr	Events Nr	Sources	Duration	Root cause	Targeted ports
2	122	45,261	741	Worm-behaving cloud	1433T (MSSQL), 1025T (RPC), 139T (Netbios), 5900T (VNC), 2967T (Symantec)
3	56	48,007	634	UDP spammers (botnet)	1026U (Windows Messenger)
10	138	26,243	573	P2P	Unusual ephemeral ports (TCP)
20	110	195,018	696	UDP spammers (botnet)	1026U, 1027U, 1028U

cured or blacklisted, and it eventually dies slowly (see [15] for a more formal description of this class of phenomena).

The interesting thing with $MC2$ is that it is made of a sequence of *worm-like* shaped μ-events. The lifetime of this MC is 741 days! It is composed of μ-events that have targeted a number of distinct services, including 1025T, 139T, 1433T, 2967T and 5900T. The results of the multi-criteria fusion algorithm indicate that those μ-events have been grouped together mainly because of the following three features: geographical location, targeted platform, and ports sequence. Moreover, a detailed analysis reveals that an important amount of IP addresses is shared by many μ-events composing this MC.

To illustrate the kinds of μ-events found in this MC, Figures 3a and 3b represent four μ-events time series. Figure 3a represents two of them, namely e626 and e628, consisting of activities against Microsoft SQL Server (1433/TCP). Whereas Figure 3b represents the other two, namely e250 and e251, consisting of activities against a Symantec Service (2967/TCP). Figure 3c zooms on these last two μ-events from day 100 to day 150. We can observe the slow increase of the two curves that are typical of worm-related attacks [15, 27].

The two μ-events on the left (resp. middle) share 528 (resp. 1754) common IP addresses with each other. Given these elements, we are tempted to believe that e626 and e628 (resp. e250 and e251) are generated by the same worm, called $WORM\_A$ (resp. called $WORM\_B$). Both worms, $WORM\_A$ and $WORM\_B$, target the same two platforms: 25 and 64. Furthermore, we found that these four μ-events share an important amount of common compromised machines. This could indicate that both worms, before having contacted our honeypots, had contaminated a relatively similar population of machines. A plausible explanation could be that both had been launched from the same initial set of machines and that they were using the same, or similar, code to choose their targets.

From the attack vector point of view, these two worms have nothing in common since they use very different types of exploits. Furthermore, they have been active in different periods of time. However, the analysis reveals that they exhibit a very similar pattern both in terms of propagation strategy and in terms of success rates. Thus, even if the infection vector differs between the two, the starting point of the infection as well as the code responsible for the propagation are, as explained, quite likely very similar. This reasoning can be generalized to all 122 μ-events, revealing the high probability that all these different attack

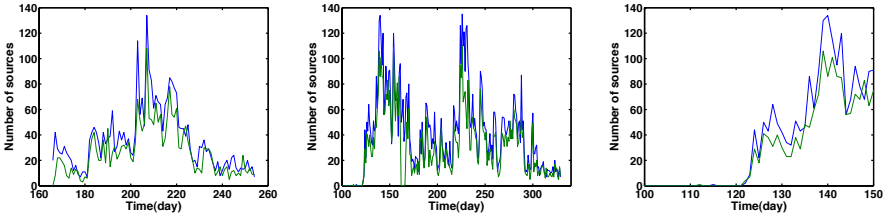


Fig. 3. Attack time series (nr of sources by day) of some μ-events from $MC2$, targeting (a) MS SQL Server (1433/TCP), (b) Symantec agent (2967/TCP). Fig. (c) is a zoom on (b).

phenomena have some common root cause(s). This does not, per se, mean that all these attacks are due to the very same person or organisation -even if this is likely- but it indicates that the same core piece of code has probably been reused, from a very similar starting point to launch a number of distinct attacks. This reveals some aspect of the modus operandi of those who have launched these attacks and this is an important piece of information for those who are in charge of identifying these misbehaving groups and their tactics.

***MC*3 and *MC*20: Windows Messenger Spammer.** In this other case study, we look at two distinct MCs: $MC3$ and $MC20$. Both are made of μ-events that have exclusively tried to send spam to innocent victims thanks to the Windows Messenger service, using UDP packets. Both MCs have been observed over a large period of time, more than 600 days in both cases. Even if they, conceptually, look similar, there are important differences between $MC3$ and $MC20$. First, the targeted ports are not identical: in $MC3$, UDP packets are being sent to three different UDP ports, namely 1026, 1027 and 1028, while in $MC20$ packets are sent exclusively to the 1026 UDP port. Then, as illustrated in Fig.4 where we can see the cumulative distribution (CDF) of sources IP addresses (grouped by /8 blocks of addresses), we observe that $MC3$ is uniformly distributed in the IPv4 space. This result is absurde since large portions of the IPv4 space can not be allocated to individual machines (multicast, bogons, unassigned, etc.) and, in all these regions, it is impossible to find compromised machines sending spams. If we find these IPs in packets hitting our honeypots, it clearly means that these are spoofed IP addresses. Furthermore, the uniform distribution of all the IP addresses in that MC leads us to believe that all other IPs are also spoofed. On the other hand, $MC20$ has a constant distribution pointing exclusively to a single /8 block owned by an ISP located in Canada[6]. A likely explanation is that those spammers have also used spoofed addresses

[6] Actually, a closer inspection of sources IP addresses reveals they were randomly chosen from only two distinct /16 blocks from this same /8 IP subnet.

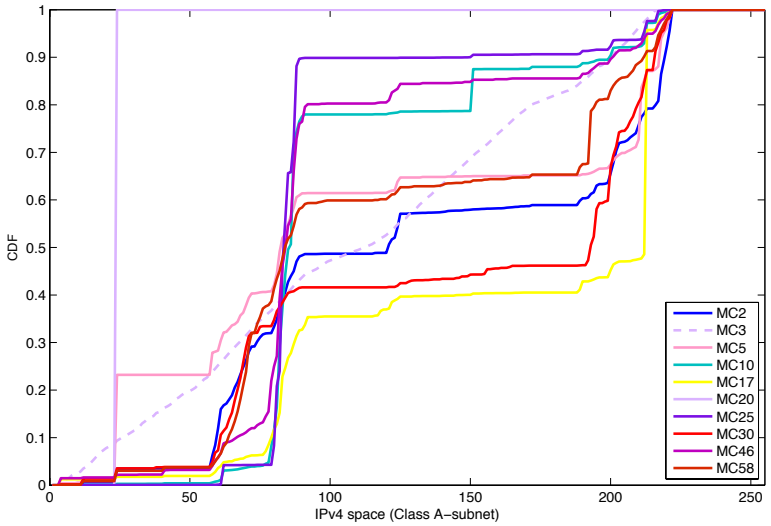


Fig. 4. CDF's of originating IP subnet distributions for the largest phenomena.

to send UDP messages to the Windows Messenger service, and they have been able to do so for 600 days without being disturbed!

To further validate these results, we also looked at the payloads of the UDP packets by computing a hash for each packet payload. What we discovered is quite surprising: all payloads sent by the sources have exactly the same message template, but the template was different for the two clouds. Fig.5 and Fig.6 show the two different templates used by spammers of $MC3$ and $MC20$ respectively. Regarding $MC3$, we also observe many alternate URL's, such as: 32sys.com, Fix64.com, Key32.com, Reg64.com, Regsys32.com, Scan32.com, etc, whereas spammers in $MC20$ use apparently almost[7] always the same URL (www.registrycleanerxp.com).

This knowledge has been derived from the observation of the MCs automatically built by our method. This illustrates the richness and meaningfulness of the analyses that can be performed. At this point, there are still two questions left unanswered when we look at those two UDP spam phenomena:

i) Do all those UDP packets really use spoofed IP addresses, and how were they sent (e.g., from a single machine in the Internet or from a very large botnet)?
ii) Could it be that those two phenomena have in fact the same root cause, i.e., the same (group of) people running in parallel two different spam campaigns?

[7] For $MC20$, only a few instances of spam messages were observed with a different URL: nowfixpc.com

```
SYSTEM  ALERT  - STOP! WINDOWS REQUIRES IMMEDIATE ATTENTION.
Windows has found CRITICAL SYSTEM ERRORS.

To fix the errors please do the following:
1. Download Registry Cleaner from: http://www.wfix32.com
2. Install Registry Cleaner
3. Run Registry Cleaner
4. Reboot your computer
FAILURE TO ACT NOW MAY LEAD TO DATA LOSS AND CORRUPTION!
```

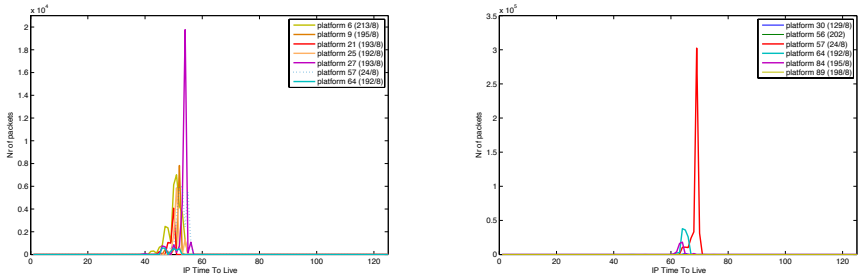
Fig. 5. Spam template used in $MC3$.

```
Local System User
CRITICAL ERROR MESSAGE! - REGISTRY DAMAGED AND CORRUPTED.

To FIX this problem:
Open Internet Explorer and type:  www.registrycleanerxp.com
Once you load the web page, close this message window

After you install the cleaner program
you will not receive any more reminders or pop-ups like this.

VISIT www.registrycleanerxp.com IMMEDIATELY!
```

Fig. 6. Spam template used in $MC20$.

To answer the first question, we have extracted from the UDP packets the Time To Live (TTL) value of their IP headers. We have computed the distributions of these TTL values for both phenomena, grouped by targeted platform. The results, illustrated in Fig.7, seems to confirm our intuition about spoofed UDP packets, since these TTL distributions are too narrow to originate from a real population of physical machines. In both cases ($MC3$ and $MC20$), we observe that the TTL distributions have a width of about 5 hops, whereas TTL distributions for non-spoofed packets are normally much larger, certainly when sources are largely distributed. As a sanity check, we retrieved the TTL distributions for another phenomenon, which has been validated as a botnet of machines. As one can see in Fig.8, the TTL distributions are much larger (around 20 hops) than for spoofed UDP packets. Another finding visible in Fig.7 is the unusual initial value used for TTL's, which also indicates that those packets were probably forged using raw sockets, instead of using the TCP/IP protocol stack of the operating system.

Finally, trying to answer the last question (same root cause or not), we looked at one additional feature of the attacks. We generated a distribution of sources by grouping them based on the *day and hour of the week* they have been observed by our platforms (using the same universal time reference, which is GMT+1 in this case). As one can see in Fig.9, the result is very intriguing: although there



(a) (b)

Fig. 7. TTL distribution of UDP packets for $MC3$ (a) and $MC20$ (b) (grouped by targeted platform)

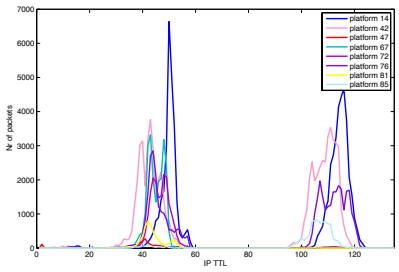


Fig. 8. TTL distribution of TCP packets for a phenomenon (MC28) attributed to a botnet targeting ports 445T and 139T (grouped by targeted platform).

is no privileged day or time interval in the week on which we observe a specific pattern, the UDP traffic created by $MC3$ (in dashed) and $MC20$ (in green) look apparently synchronized. Since both phenomena have lasted more than 600 days, it is quite unlikely that such correlation could be due to chance only. So, while we have no true evidence to verify this, we can reasonably assume that both phenomena have been orchestrated by the same people, or at least using the same software tool and sets of compromised machines.

***MC*10: P2P aberrations** $MC10$ is a very interesting, yet intriguing, cloud. Our technique has grouped together 138 μ-events that have been observed over a period of 573 days. All these events share a number of common characteristics that we have some difficulty to explain:

1. The vast majority of these μ-events target a single platform, located in China. A very few μ-events have also hit another platform in Spain.
2. The vast majority of these μ-events originate from Italy and Spain only.

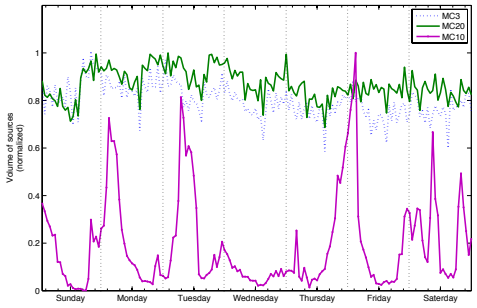


Fig. 9. Distribution of malicious sources grouped by weekdays. For each MC, a data point represents the accumulated number of sources observed for a given day and hour of the week.

3. All these μ-events exist during a single day.
4. All these μ-events target a single high TCP port number, most of them not being assigned to any particular protocol (e.g. 10589T, 15264T, 1755T, 18462T, 25618T, 29188T, 30491T, 38009T, 4152T, 46030T, 4662T, 50656T, 53842T, 6134T, 6211T, 64264T, 64783T, 6769T, 7690T)
5. these μ-events share a substantial amount of source addresses between them.
6. A number of high port numbers correspond to port numbers used by well known P2P applications (e.g., 4662/TCP, used by eDonkey P2P network).

This last remark leads us to hypothesize that this extremely weird type of attack traces may have something to do with P2P traffic aberrations. It can be a misconfiguration error or, possibly, the side effect of a deliberate attack against these P2P networks, as explained in [12, 5], in which authors argued that it is possible to use P2P networks to generate DDoS attacks against any arbitrary victim.

Also, Figure 9 highlights the fact that these 138 μ-events are not randomly distributed over the hours of the week but that, instead, they seem to exist on a limited number of recurrent moments.

All these elements tend to demonstrate the meaningfulness of grouping all these, apparently different, attack events. Even if we are not able, at this stage, to provide a convincing explanation related to their existence, our method has, at least, the merit of having highlighted the existence of these, so far, unknown phenomena.

It is our hope that other teams will build upon this fundational result to help all of us to better understand these numerous threats our approach has identified.

6 Conclusions

In this document, we have presented the WOMBAT attack attribution method. We have explained its motivations, its principles, the various steps it was made of, as well as some of the interesting results it had delivered so far. We have applied that technique to 2 years of attack traces captured on 40 low interaction honeypots located all over the world. It is worth noting that the method could as easily be applied on completely different threats-related events. In fact, the interim Symantec report published mid October 2009 on the analysis of rogue AV web sites offers results of the application of this very same method to the problem of understanding the modus operandi of malicious users setting up rogue AV campaigns.

It is our hope that people will be interested in trying to understand the rationales behind the *Misbehaving Clouds* we have identified. We are eager to share as much information as possible with such interested parties. Similarly, we are looking forward in having other opportunities to apply this method to other security datasets that future partners would be willing to share with us.

References

1. Michele Basseville and Igor V. Nikiforov. *Detection of Abrupt Changes:Theory and Application*. Prentice Hall, 1993.
2. G. Beliakov, A. Pradera, and T. Calvo. *Aggregation Functions: A Guide for Practitioners*. Springer, Berlin, New York, 2007.
3. M. P. Collins, T. J. Shimeall, S. Faber, J. Janies, R. Weaver, M. De Shon, and J. Kadane. Using uncleanliness to predict future botnet addresses. In *IMC '07: Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 93–104, New York, NY, USA, 2007. ACM.
4. Marc Dacier, Fabien Pouget, and Hervé Debar. Attack processes found on the internet. In *NATO Symposium IST-041/RSY-013*, Toulouse, France, April 2004.
5. Karim El Defrawy, Minas Gjoka, and Athina Markopoulou. Bottorrent: misusing bittorrent to launch ddos attacks. In *SRUTI'07: Proceedings of the 3rd USENIX workshop on Steps to reducing unwanted traffic on the internet*, pages 1–6, Berkeley, CA, USA, 2007. USENIX Association.
6. A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall advanced reference series, 1988.
7. S. Kullback and R. A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics 22: 79-86.*, 1951.
8. C. Leita, V. H. Pham, O. Thonnard, E. Ramirez Silva, F. Pouget, E. Kirda, and M. Dacier. The leurre.com project: collecting internet threats information using a worldwide distributed honeynet. In *1st WOMBAT workshop, April 21st-22nd, Amsterdam, The Netherlands*, Apr 2008.
9. Corrado Leita and Marc Dacier. Sgnet: a worldwide deployable framework to support the analysis of malware threat models. In *Proceedings of the 7th European Dependable Computing Conference (EDCC 2008)*, May 2008.
10. Leurre.com, Eurecom Honeypot Project. http://www.leurrecom.org/, [[s]ep 2009].
11. J. Lin. Divergence measures based on the shannon entropy. *Information Theory, IEEE Transactions on*, 37(1):145–151, Jan 1991.

12. Naoum Naoumov and Keith Ross. Exploiting p2p systems for ddos attacks. In *InfoScale '06: Proceedings of the 1st international conference on Scalable information systems*, page 47, New York, NY, USA, 2006. ACM.
13. Ruoming Pang, Vinod Yegneswaran, Paul Barford, Vern Paxson, and Larry Peterson. Characteristics of Internet Background Radiation. In *Proceedings of the 4th ACM SIGCOMM conference on the Internet Measurement*, 2004.
14. M. Pavan and M. Pelillo. A new graph-theoretic approach to clustering and segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2003.
15. Van-Hau Pham. *Honeypot traces forensics by means of attack event identification.* PhD thesis, TELECOM ParisTech, 2009.
16. Van-Hau Pham and Marc Dacier. Honeypot traces forensics : the observation view point matters. In *NSS 2009, 3rd International Conference on Network and System Security, October 19-21, 2009, Gold Coast, Australia*, Dec 2009.
17. Van-Hau Pham, Marc Dacier, Guillaume Urvoy Keller, and Taoufik En Najjary. The quest for multi-headed worms. In *DIMVA 2008, 5th Conference on Detection of Intrusions and Malware & Vulnerability Assessment, July 10-11th, 2008, Paris, France*, Jul 2008.
18. Fabien Pouget, Marc Dacier, and Hervé Debar. Honeypot-based forensics. In *Proceedings of AusCERT Asia Pacific Information Technology Security Conference 2004*, Brisbane, Australia, May 2004.
19. Fabien Pouget, Marc Dacier, and Van Hau Pham. Leurre.com: on the advantages of deploying a large scale distributed honeypot platform. In *ECCE'05, E-Crime and Computer Conference, 29-30th March 2005, Monaco*, Mar 2005.
20. Niels Provos. A virtual honeypot framework. In *Proceedings of the 12th USENIX Security Symposium*, pages 1–14, August 2004.
21. Roger N. Shepard. Multidimensional scaling, tree fitting, and clustering. *Science*, 210:390–398, 1980.
22. Olivier Thonnard and Marc Dacier. A framework for attack patterns' discovery in honeynet data. *DFRWS 2008, 8th Digital Forensics Research Conference, August 11- 13, 2008, Baltimore, USA*, 2008.
23. Olivier Thonnard and Marc Dacier. Actionable knowledge discovery for threats intelligence support using a multi-dimensional data mining methodology. In *ICDM'08, 8th IEEE International Conference on Data Mining series, December 15-19, 2008, Pisa, Italy*, Dec 2008.
24. Olivier Thonnard, Wim Mees, and Marc Dacier. Addressing the attack attribution problem using knowledge discovery and multi-criteria fuzzy decision-making. In *KDD'09, 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Workshop on CyberSecurity and Intelligence Informatics, June 28th - July 1st, 2009, Paris, France*, Dec 2009.
25. D. Wheeler and G. Larsen. Techniques for Cyber Attack Attribution. *Institute for Defense Analyses, Oct 2003*, 2008.
26. Ronald R. Yager. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Trans. Syst. Man Cybern.*, 18(1):183–190, 1988.
27. Vinod Yegneswaran, Paul Barford, and Vern Paxson. Using honeynets for internet situational awareness. In *Fourth ACM Sigcomm Workshop on Hot Topics in Networking (Hotnets IV)*, 2005.

REPORT: SYMANTEC ENTERPRISE SECURITY

Symantec Report on Rogue Security Software

July 08 – June 09

Published October 2009

Confidence in a connected world.

symantec™

White Paper: Symantec Enterprise Security

Symantec Report on Rogue Security Software

July 08 – June 09

Contents

Symantec Report on Rogue Security Software
July 08 – June 09

Introduction

The *Symantec Report on Rogue Security Software* is an in-depth analysis of rogue security software programs. This includes an overview of how these programs work and how they affect users, including their risk implications, various distribution methods, and innovative attack vectors. It includes a brief discussion of some of the more noteworthy scams, as well as an analysis of the prevalence of rogue security software globally. It also includes a discussion on a number of servers that Symantec observed hosting these misleading applications. Except where otherwise noted, the period of observation for this report was from July 1, 2008, to June 30, 2009.

Symantec has established some of the most comprehensive sources of Internet threat data in the world through the Symantec™ Global Intelligence Network. More than 240,000 sensors in over 200 countries monitor attack activity through a combination of Symantec products and services such as Symantec DeepSight™ Threat Management System, Symantec Managed Security Services and Norton™ consumer products, as well as additional third-party data sources.

Symantec also gathers malicious code intelligence from more than 130 million client, server, and gateway systems that have deployed its antivirus products. Additionally, Symantec's distributed honeypot network collects data from around the globe, capturing previously unseen threats and attacks and providing valuable insight into attacker methods.

Spam and phishing data is captured through a variety of sources including the Symantec Probe Network, a system of more than 2.5 million decoy accounts; MessageLabs™ Intelligence, a respected source of data and analysis for messaging security issues, trends and statistics; and other Symantec technologies. Data is collected in more than 86 countries. Over 8 billion email messages and over 1 billion Web requests are processed per day across 16 major data centers. These resources give Symantec's analysts unparalleled sources of data with which to identify, analyze, and provide informed commentary on emerging trends in attacks, malicious code activity, phishing, and spam.

NOTE: Symantec advises against visiting the websites of the rogue security applications discussed in this report because these sites may be unsafe and could potentially harm your computer.

Also, rogue security applications are often marketed by different distributors under slightly different spellings. For example, AntiVirus XP 2008 may appear as AntiVirusXP 2008, AntivirusXP 2008, etc. Symantec uses what it considers to be a common variation for this report.

Overview of Rogue Security Software

A rogue security software program is a type of misleading application (also known as scareware) that pretends to be legitimate security software, such as an antivirus scanner or registry cleaner, but which actually provides the user with little or no protection whatsoever and, in some cases, can actually facilitate the installation of malicious code that it purports to protect against. There are two prevalent ways in which rogue security software can be installed on a user's computer: either it is downloaded and installed manually by a user after he or she has been tricked into believing that the software is legitimate; or it is unknowingly installed onto a user's computer, such as when a user visits a malicious website designed to automatically download and install illegitimate applications.

Profit is a primary motivation for creators and distributors of rogue security software scams. A common approach is to try to trick users into believing that these rogue security applications are valid and to get users to download and install the programs and to pay for them. Techniques used to entrap users often rely on fear tactics and other social engineering tricks that are distributed through means such as links in spam, pop-up and banner advertisements on websites and instant messaging programs, postings on forums and social networking sites, and sponsored or falsely promoted search engine results.[1] Attackers also market rogue security software with claims that the programs can remove unwanted applications such as spyware or adware. Not only do these scams cheat users out of money—advertised costs for these products range from $30 to $100 (all currency U.S.) and some even try to sell multi-year licenses—but the personal and credit card information that users provide to register these fake products could also be used in additional fraud.[2]

Once installed on a user's computer—and to induce payment—rogue security applications often deliberately misrepresent the computer's security status or performance, displaying fake or exaggerated claims of security threats even if the computer has not been compromised. These applications use continuous pop-up displays, taskbar notification icons, and other alerts to indicate that the user needs to purchase a full version or register for an annual subscription of the program in order to remove the reported threats and clean the computer (figure 1).[3] Some rogue security applications may even install additional threats onto the compromised computer while simultaneously producing reports that it is clean.

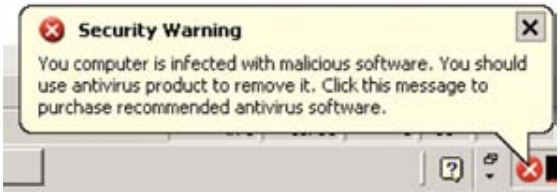


Figure 1. Rogue security software taskbar notification alert

Courtesy: Symantec Corporation

To fool potential victims, rogue security software programs are designed to appear as legitimate as possible. This includes using realistic-sounding names such as VirusRemover2008,[4]AntiVirusGold,[5] or SystemGuard2009,[6] or names that mimic existing legitimate security software, such as "Nortel."[7] Most rogue security programs also have fully developed websites that include the ability to download and purchase the software, with some actually using legitimate online payment services to process credit card transactions from successful scams. Some scams even return an email message to the victim with a receipt for purchase that includes a serial number and a valid, functioning customer service phone number. The advertisements, pop-up windows, and notification icons used to market these scams are also all designed to mimic

legitimate antivirus software programs, often using the same fonts, colors, and layouts as trusted security software vendors (figure 2).

Figure 2. Security warning mimicking a legitimate vendor

Courtesy: Symantec

Rogue security software programs are often rebranded or cloned versions of previously developed programs. Cloning is often done because the original version of the rogue security application has been discovered or detected by legitimate security vendors. Cloning is therefore fuelled by the hope that one or more of the clones will escape detection.[8] This process sometimes involves nothing more than changing out the name, logos, and images of a program in an attempt to give it a new identity while the program itself remains unchanged. One program may be rebranded multiples times.

Another reason for cloning programs is to minimize the impact of credit card chargebacks and payment reversals.[9] Major credit card companies fine issuing banks and credit card payment processors for retaining merchants with high chargebacks.[10] Usually, the payment processing company simply ceases conducting business with such merchants or else passes the cost of the fines onto them. By rebranding the applications and registering using a different name, rogue security software creators and distributors—the merchants in this case—can circumvent these issues. As well, many users might not recognize the rebranded application as false.

Examples of rebranded rogue security software programs include AntiVirus 2009,[11] which is a clone of Antivirus 2008,[12] and AntiVirus XP 2008,[13] which is a clone of Malware Protector 2008 (figure 3).[14] The latter program is also part of a family of rogue security software clones that includes AdvancedXPFixer[15] and WinIFixer.[16]

1-http://www.messagelabs.com/mlireport/MLIReport_Annual_2008_FINAL.pdf: pp. 31, 35
2-http://www.symantec.com/connect/blogs/misleading-applications-show-me-money
3-Ibid.
4-http://www.symantec.com/security_response/writeup.jsp?docid=2008-072217-2258-99
5-http://www.symantec.com/security_response/writeup.jsp?docid=2006-032415-1558-99
6-http://www.symantec.com/business/security_response/writeup.jsp?docid=2009-031311-4206-99
7-http://www.symantec.com/connect/blogs/nort-what-av (Please note that the spoofed site has no association at all with Nortel Networks™.)
8-http://www.symantec.com/connect/blogs/cloning-profit
9-A credit card chargeback is when the consumer's issuing bank returns the funds back to the consumer, and the payment to the merchant is reversed. This usually occurs when the consumer files a complaint regarding the charge with the issuing bank.
10-http://www.corporate.visa.com/pd/rules/pdf/visa-international-operating-regulations.pdf: Table 1-9
11-http://www.symantec.com/business/security_response/writeup.jsp?docid=2008-082521-2037-99
12-http://www.symantec.com/business/security_response/writeup.jsp?docid=2008-050906-3727-99
13-http://www.symantec.com/security_response/writeup.jsp?docid=2008-071613-4343-99
14-http://www.symantec.com/security_response/writeup.jsp?docid=2008-060420-4214-99
15-http://www.symantec.com/security_response/writeup.jsp?docid=2008-052212-0934-99
16-http://www.symantec.com/security_response/writeup.jsp?docid=2008-030406-0943-99

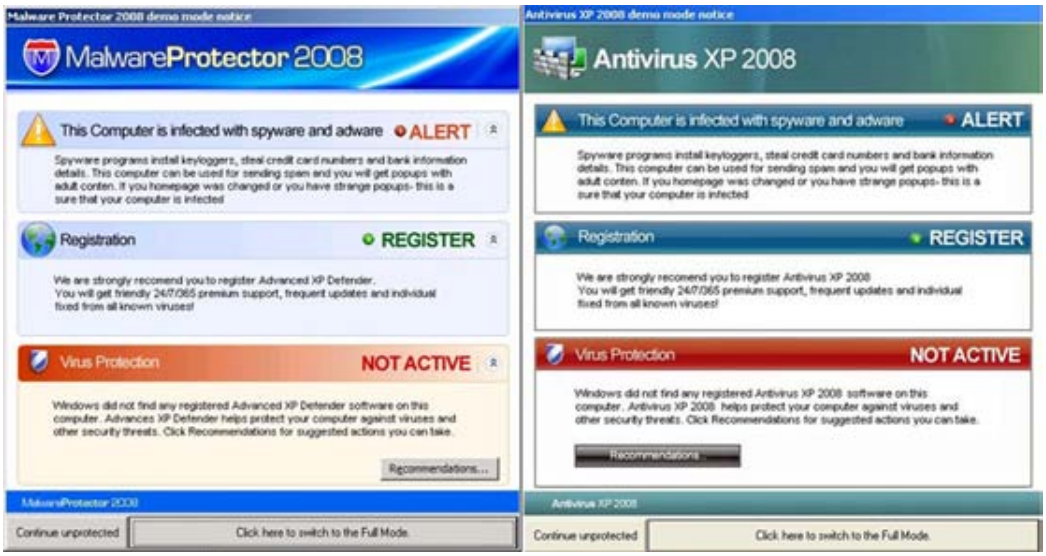


Figure 3. Malware Protector 2008 and its clone, AntiVirus XP 2008

Courtesy: Symantec

Risks

One major risk associated with installing rogue security software programs is that the user may be given a false sense of security with the belief that the application is genuine and that his or her computer is protected from malicious code threats. This is because rogue security applications frequently report that malicious threats have been removed and that the computer is clean and fully protected when, in reality, the opposite is often true and the misleading application is providing little or no protection from threats at all. These programs may actually increase the danger of the user's computer being compromised. This is because some rogue security software programs instruct the user to lower existing security settings in order to advance the registration process, such as switching off firewall settings and/or disabling existing (and legitimate) antivirus programs (figure 4). Also, once installed, the false application may prevent the computer from accessing legitimate security vendor websites, thus obstructing the user's ability to research how to remove the misleading software.

Figure 4. Registration pop-up display for AntiVirus 2009

Courtesy: Symantec

In other instances, a computer may have already been compromised with malicious code or may be at risk of attack from additional threats. This is because some rogue security applications are designed to install additional threats (even while continuing to report that the compromised computer is clean). For example, some applications will launch pop-up windows that, if any of the options presented are clicked, will download malicious code to the victim's computer.[17] This will occur even if the option chosen is the close window "X" or the negative response option.

Another potential risk involved with rogue security software is that the scam perpetrators will use the personal information gained from the victim to commit fraud and/or identity theft. Thus, not only can these programs cheat the user out of money, but the personal details and credit card information that are provided during the purchase (figure 5) can be used in additional fraud or else sold on black market forums, where credit card data is advertised for as much as $30 per card.[18]

17-http://www.messagelabs.com/mlireport/MLIReport_Annual_2008_FINAL.pdf : pp. 31, 35
18-Underground economy servers are black market forums for the promotion and trade of stolen information and services, such as credit card numbers and bank accounts. See the Symantec Report on the Underground Economy, http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_underground_economy_report_11-2008-14525717.en-us.pdf

Figure 5. AntiVirus 2009 payment page (with option for "Premium Support" and "upgrade to fileShredder")
Courtesy: Symantec

Some versions of rogue security software include keystroke loggers as well as backdoor functionality, allowing potential access to personal information and other stored information on the user's computer such as stored passwords and other sensitive information. For example, figure 6 shows the administrative interface to the Bakasoftware back-end management system. This administrative tool allows the Bakasoftware administrator to load new and additional software (for example, "cosma bot") on a computer already compromised with rogue security software.

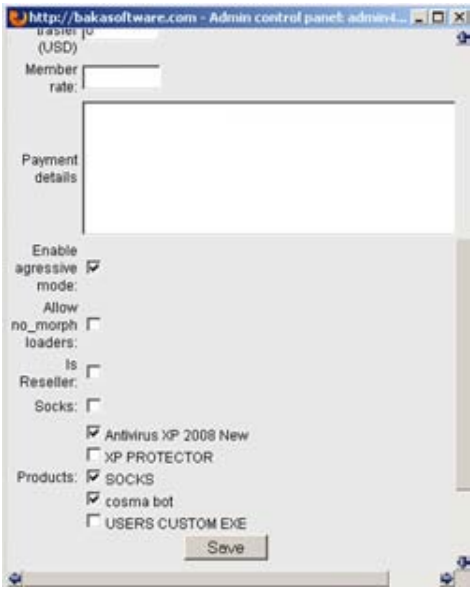


Figure 6. Bakasoftware administrative control panel

Courtesy: Symantec

Just as legitimate security software needs to contact a manufacturer's servers to obtain signature updates and other functions, the rogue security software may also contact the scam perpetrator's servers for updates and added functionality. In this case, though, the update results in the further compromise of the user's computer. In this way, rogue security software could represent a greater risk than expected if it is possible for a computer compromised with rogue security software to be used in a larger bot network that is maintained by structured updates from control servers.

Advertising methods

Attackers use many methods to tempt users into downloading and installing rogue security software programs. Along with employing a number of standard methods similar to legitimate Internet advertising campaigns, scam perpetrators also employ fear tactics and other social engineering techniques to sell their products. This section discusses some of the main advertising methods used to market rogue security software programs.

Spam

Spam is an easy way to advertise rogue security software programs because it is relatively quick and inexpensive to send a large number of email messages, especially if a spammer uses a botnet to do the work. For example, in 2008, spam for AntiVirus XP 2008 was sent out from botnets such as Peacomm,[19] Srizbi,[20] Rustock,[21] and Ozdok[22].[23] Email addresses suitable for spam are inexpensive, costing as little as $0.33/MB (with one MB containing as many as 40,000 email addresses).[24]

Some spam is sent with executable file attachments that, if opened, will install the rogue security software program. Because many security software programs and upstream providers now guard against this with spam filters that flag email containing suspicious attachments, spam distributors instead send email with messages that are worded to lure users into following a link to the associated website for the fraudulent program.

19-http://www.symantec.com/security_response/writeup.jsp?docid=2007-011917-1403-99

Advertisements on websites

Rogue security software programs are advertised on a variety of both malicious and legitimate websites, including blogs, forums, social networking sites, and adult sites.[25] These advertisements typically prey on users' fears of malicious code, with claims such as, "If this ad is flashing, your computer may be at risk or infected," and will urge users to follow a link that will provide the software to remove the threats.

Link spamming packages (also known as auto-submitters) are also often used to place links pointing to rogue security application websites. One example is the Xrumer software package. Xrumer can bypass CATCHA protections, automatically register and confirm email activation requests, and is capable of quickly spamming large numbers of websites.[26] By using such tools, scam distributors can increase their search engine rankings and place links on thousands of websites to drive victims to a rogue security application website.

To increase exposure and add an air of legitimacy, scam distributors also place Web banner advertisements on major Internet advertising networks and with advertising brokers of legitimate sites.[27] This is possible because administrators of legitimate websites often link to feed services that control the dispersal of these advertisements and the administrators usually have no control what content is displayed in the advertisements.[28] Moreover, the feed service distributors may not be able to control content either, because they are often a middle ground between feed subscribers and the actual advertisers. If an advertiser pays the distributor to display advertisements, the distributor may have very little control over the data displayed in the advertisements. This makes mitigating deceptive or malicious advertisements very difficult. Tracking down the original source of the malicious or deceptive content can also be very challenging.

Search engine results seeding

Another method of advertising rogue security software programs is to seed search engine results by capitalizing on popular news items, events, or celebrities.[29] Scam creators use a range of black hat search engine optimization (SEO) techniques to effectively poison search engine results and increase the ranking of their scam sites whenever any topical news event is searched.[30] For example, the Downadup[31] worm (also known as Conficker) emerged and spread rapidly in the latter months of 2008, with well over a million individual computers affected by the end of that year.[32] To play on consumers' fears of the worm, scam perpetrators created website pages full of terms such as "remove virus" or "free anti-virus," etc. This increased the keyword count of the pages, thus making them seem more relevant to search engine relevancy algorithms.[33]

Browser helper objects

Another method recently observed by Symantec for advertising rogue security applications was used in the promotion of AntiVirus 2009, one of the most widely reported of these programs during this reporting period.[34] In this approach, once AntiVirus 2009 is installed on a computer, it creates a browser helper object (BHO) that modifies all pages from a search engine by adding a fake "security tip" that appears to originate from the search engine company, complete with legitimate logos (figure 7).[35] In reality, this tip service is non-existent. The purpose of the tip on the Web page is to entice the user of the compromised computer to click on the link to "activate" Antivirus 2009.

20-http://www.symantec.com/security_response/writeup.jsp?docid=2007-062007-0946-99
21-http://www.symantec.com/security_response/writeup.jsp?docid=2006-011309-5412-99
22-http://www.symantec.com/security_response/writeup.jsp?docid=2008-021215-0628-99
23-http://www.messagelabs.com/mlireport/MLIReport_Annual_2008_FINAL.pdf : p. 31
24-http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_internet_security_threat_report_xiv_04-2009.en-us.pdf : p. 82

Figure 7. Fake tip page

Courtesy: Symantec

Installation techniques

Rogue security software programs can get onto a user's computer either by being manually installed by the user, who has been fooled into thinking that he or she is downloading a legitimate program; or it is unknowingly downloaded and installed by the user without his or her consent or knowledge. This section will discuss delivery methods and strategies used by scam distributors. (For best practices to safeguard against these threats, please see Appendix A of this report.)

Emailed executable files

One of the simplest ways to distribute rogue security software programs is through executable files attached to spam. The malicious attachments are typically disguised as executable files with false file extensions, such as music, media, or compressed (that is, .zip) files. If opened, these attachments will instead either install a rogue security software program directly, or else will load malicious code onto the computer that subsequently installs the rogue software. As mentioned, many security software programs and ISPs now have extensive safeguards to protect against potentially malicious attachments.

Malicious code

Rogue security software programs can be installed onto a user's computer by malicious code such as staged downloaders. Staged downloaders are threats that, once on a computer, will download and install other malicious code. This is typically done without the user's knowledge or consent. One of the more popular methods of getting malicious code onto a victim's computer is through drive-by download attacks. Drive-by downloads occur when a user visits a malicious website or a legitimate website that has been compromised and malicious code is downloaded onto the user's computer without the user's interaction or authorization. The attacks attempt to gain access to a user's system by exploiting vulnerabilities in browsers, browser plug-ins and applications, or desktop applications. The download is typically an executable file containing malicious code that then attempts to download additional threats, such as rogue security software programs. Because the user is usually oblivious to these occurrences, such attacks can be difficult to mitigate. Drive-by downloads

25-http://www.symantec.com/norton/theme.jsp?themeid=mislead
26-http://blog.washingtonpost.com/securityfix/2007/01/scary_blogspam_automation_tool_1.html
27-An advertising network is a distributor of advertisements to websites that want to host them; they typically have a large inventory of advertisements that get displayed each time a Web page is loaded or refreshed; the website often will not have control over the content of these advertisements.
28-See http://www.eweek.com/c/a/Security/DoubleClick-Serves-Up-Vast-Malware-Blitz/ and http://www.theregister.co.uk/2008/02/21/itv_scareware_peril/
29-http://www.symantec.com/connect/blogs/misleading-applications-show-me-money-part-2
30-SEO is a process for making websites more popular in search engine results; black hat SEO uses search optimization techniques that are considered unethical by the mainstream SEO community, which may include spamming and other questionable practices.
31-http://www.symantec.com/security_response/writeup.jsp?docid=2009-040823-4919-99
32-http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the_downadup_codex_ed2.pdf
33-http://www.symantec.com/connect/blogs/downadup-related-search-indexes-poisoned-fake-av-sites
34-http://www.symantec.com/security_response/writeup.jsp?docid=2008-082521-2037-99
35-http://www.symantec.com/connect/blogs/misleading-applications-start-tipping

are becoming an increasingly dominant vector of attack, as discussed in Volume 14 of the *Symantec Internet Security Threat Report,* especially since such attacks can be launched from both legitimate and malicious websites.[36]

A specific example of malicious code associated with rogue security software is the Zlob Trojan.[37]First identified in 2005, Zlob was the third most common staged downloader component observed by Symantec in 2008.[38] One of its primary attack vectors to get onto a user's computer is disguised as a video codec installer. A video codec is a type of software that supports the compression (or decompression) of digital video. Because additional codecs are often required to play a specific video format, depending on how the video in question was created, users may be more likely to trust such prompts and download the files. This type of Web-based attack follows a trend of attackers inserting malicious code into legitimate high-traffic websites where users are likely to be more trusting of the content, rather than attempting to lure users to visit specifically designed, malicious sites.[39]

Once embedded onto a compromised computer, one particular function of Zlob is to display fake security alerts and pop-ups claiming that the computer is infected with spyware. If a user clicks on the alert, Zlob will redirect the user's Web browser to a website containing malicious code, at which point the computer will be attacked further. The top three reported rogue security applications observed by Symantec during this reporting period (discussed below in "Top reported rogue security software") were all distributed in part by Zlob, as were a number of others, including PrivacyCenter,[40] Malware Defender 2009,[41] VirusProtectPro,[42] and IE Defender.[43]

IE Defender is worth noting further because, once installed on a computer, the program performs a scan that automatically detects the presence of malicious code, including Zlob. Thus, IE Defender prompts the user to pay for a full license of itself in order to remove Zlob, which is responsible for IE Defender being installed on the user's computer in the first place.

Another example of malicious code associated with rogue security software is the Vundo Trojan, which is a component of an adware program that exploits a browser vulnerability.[44] Vundo was the top-ranked malicious code sample observed by Symantec globally in both 2007 and 2008.[45] It typically infects computers through links to malicious websites from spam or email attachments that, in reality, also contain the malicious code. The compromise may also occur via a drive-by download, as described above.[46] As a staged downloader, once Vundo is installed on a computer, it attempts to contact certain IP addresses to download additional components, including the adware downloader component of the Trojan that, once executed, is used to display pop-up advertisements.

Rogue security software website downloads

Websites created to market rogue security software programs are designed to look as legitimate as possible so that users will be convinced that the products are authentic and will download them. As such, they often include the logos and formatting typical of the websites of legitimate security vendors, testimonials from satisfied customers, and other seemingly genuine techniques. One rogue security application site, for Green Antivirus 2009,[47] even claims to be the "world's first antivirus that cares about the environment," pledging that "$2 from every sale will be sent on saving green forests in Amazonia" [*sic*].[48] To trick users into downloading their products, some rogue security websites offer free trials or free system scans. In fact, MessageLabs Intelligence observed that, of the most frequent rogue security applications blocked through MessageLabs Web Security Service (WSS), 95 percent contained the generic "freescan.php" filename.

36-http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_internet_security_threat_report_xiv_04-2009.en-us.pdf : p. 52
37-http://www.symantec.com/security_response/writeup.jsp?docid=2005-042316-2917-99

Many rogue security software websites are associated with more than one domain name so that, if one server is taken offline to evade detection by authorities or shutdown by upstream ISPs, redundancies exist to keep the scam running. In Symantec's research on servers hosting rogue security software, discussed further below in this report, over 194,000 domain names were observed associated with these false applications over a two-month period.

Rogue security software distributors

The creators of rogue security software often use an affiliate-based, pay-per-install model to distribute their misleading applications. Users who wish to participate in a rogue security software scam can register as an affiliate on a distribution site, such as TrafficConverter.biz, where they can obtain the appropriate files and links to market the scam.[49] Typically, these websites offer free registration and the affiliates then carry out all of the marketing for the product. The main purpose of these distribution websites is to recruit affiliates to sell the rogue security software programs.

The creators of the distribution websites provide affiliates with the support and the tools required to distribute and market the scams, such as fake codec links, fake scanner links, and malicious code executable files. They may also provide affiliates with promotional and marketing materials, as well as obfuscation tools such as packers and binders (used to create versions of the code in order to evade detection by legitimate security software).

Another evasive maneuver is the use of polymorphic techniques. Polymorphic obfuscation modifies program code, as often as every five minutes, to alter the digital signatures of the code while keeping the underlying functionality intact. This makes polymorphic threats difficult to detect since they are constantly changing. These services and tools are usually provided to the scam distributors for free or for a nominal fee.

Affiliates are paid a predetermined amount for every successful installation, ranging from $0.01 to $0.55.[50] This per-installation payment is dependent on the type of installation and the distribution site, with malicious code installations returning the highest commission. The price is also dependent on the country of the computer on which the rogue security software program has been installed. For example, one distribution site paid $0.55 per installation on computers in the United States, but only $0.05 per installation on computers in Mexico (table 1).[51] The site also gave installation incentives to affiliates through additional bonuses, such as a 10 percent bonus for more than 500 installations per day and a 20 percent bonus for over 2,500 installations per day. The per-installation price variations from country to country may depend on the likelihood of a user in that country paying for either a subscription to, or a fully registered version of the rogue security software. Basically, the higher the percentage of users in a certain country that pays, the higher the per-installation payment.

38-http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_internet_security_threat_report_xiv_04-2009.en-us.pdf : p. 62
39-Ibid: p. 31
40-http://www.symantec.com/business/security_response/writeup.jsp?docid=2009-050702-2910-99
41-http://www.symantec.com/business/security_response/writeup.jsp?docid=2009-033012-2224-99
42-http://www.symantec.com/business/security_response/writeup.jsp?docid=2007-070323-1203-99
43-http://www.symantec.com/security_response/writeup.jsp?docid=2007-111420-0754-99
44-http://www.symantec.com/security_response/writeup.jsp?docid=2004-112111-3912-99
45-http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_internet_security_threat_report_xiv_04-2009.en-us.pdf : p. 60
46-http://www.securityfocus.com/bid/11515
47-http://safeweb.norton.com/report/show?name=green-av-pro.com
48-http://www.411-spyware.com/tag/green-anti-virus-2009
49-http://voices.washingtonpost.com/securityfix/2009/03/obscene_profits_fuel_rogue_ant.html
50-http://www.symantec.com/connect/blogs/misleading-applications-show-me-money-part-3
51-Ibid.

Country	Region	Per-installation Price
United States	NAM	$0.55
United Kingdom	EMEA	$0.52
Canada	NAM	$0.52
Australia	APJ	$0.50
Spain	EMEA	$0.16
Ireland	EMEA	$0.16
France	EMEA	$0.16
Italy	EMEA	$0.16
Germany	EMEA	$0.12
Belgium	EMEA	$0.12
Netherlands	EMEA	$0.12
Denmark	EMEA	$0.10
Norway	EMEA	$0.05
Mexico	LAM	$0.05
Other countries	N/A	$0.01

Table 1. Examples of per-installation prices for rogue security software, by country[52]

Source: Symantec

In the case of TrafficConverter.biz, the website was associated with the Downadup worm as a URL from which Downadup attempted to download its payload.[53] The site was shut down in November 2008 before the worm could download the unknown payload. TrafficConverter.biz and other reincarnations of the website paid affiliates $30 per sale of their rogue security software programs, such as XP Antivirus.[54] The site purported to have at least 500 active affiliates, with top affiliates earning as much as $332,000 in a month for installing and selling security risks—including rogue security software programs—onto users' computers.[55] The top 10 earning affiliates purportedly each earned $23,000 per week, on average. The website even kept statistics on their top sellers, including listing percentages on the conversion of installations-to-sales per day (figure 8). In addition, the website offered "VIP-points" contests to top-selling affiliates, complete with prizes such as electronics and a luxury car (figure 9).

The successful webmaster:

Date	Antispyware							Refs	Total
	U	I	U/I	I/S	U/S	S	Earn		
15.01.2008	36109	3794	10	53	502	72	$2304.00	$0.00	$2304.00
16.01.2008	34634	3972	9	46	398	87	$2784.00	$0.00	$2784.00
17.01.2008	47484	5543	9	55	475	100	$3200.00	$0.00	$3200.00
18.01.2008	54756	5748	10	55	527	104	$3328.00	$0.00	$3328.00
19.01.2008	70018	6630	11	55	583	120	$3840.00	$0.00	$3840.00
20.01.2008	71238	6744	11	77	810	88	$2816.00	$0.00	$2816.00
21.01.2008	77558	6562	12	49	575	135	$4320.00	$0.00	$4320.00
Total:	391797	38993	10	55	555	706	$22592.00	$0.00	$22592.00

Uniques - U
Sales - S
Installs - I

Figure 8. TrafficConverter.biz sample earnings per day

Courtesy: Symantec

52-NAM = North America, EMEA = Europe, the Middle East, and Africa, APJ = Asia-Pacific/Japan, LAM = Latin America
53-http://www.symantec.com/connect/blogs/downadup-motivations
54-http://www.symantec.com/security_response/writeup.jsp?docid=2007-101010-0713-99

Figure 9. TrafficConverter.biz website with contest announcement

Courtesy: Symantec

Dogma Software was yet another rogue affiliate program that offered incentives to install their scareware on victim computers. The Dogma affiliate program claims to be "cleaning software" and offers up to $30 per installation (figure 10).

55-http://voices.washingtonpost.com/securityfix/2009/03/obscene_profits_fuel_rogue_ant.html

Figure 10. Dogma Software website

Courtesy: Symantec

These affiliate "master sites" such as Bakasoftware, TrafficConverter and Dogma Software seem to be the drivers for the associated domain names, websites, and malicious advertising behind many rogue security software scams. Without the affiliate commission payouts and back-end billing systems in place, there would likely be fewer scams perpetuated. Many in the security community have realized this and have refocused their efforts on identifying and shutting down the scam creators instead of trying to track down and identify the myriad domain names used to offer rogue security software.

Legal actions and noteworthy scam convictions

Attackers who create and distribute rogue security software programs can make a significant amount of money through these scams. They can also use the credit card information obtained from the victims to commit further fraud or to sell the data on black market forums.[56] This section will discuss several notable scams and the actions that government organizations have taken to combat perpetrators of rogue software security scams.

Legal actions taken against this type of scam include charges of fraud, deceptive advertising, misrepresentation, and in some cases, spam distribution (in cases where the software itself may not be illegal). For example, in 2006, the Attorney General for Washington State obtained a $1 million settlement from a New York-based company through a combination of

56-http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_internet_security_threat_report_xiv_04-2009.en-us.pdf : p. 83

the state's 2005 Computer Spyware Act, federal and state spam laws, and the U.S. Consumer Protection Act.[57] The company fined was distributing the rogue security software program, Spyware Cleaner.[58] The state sued the company for marketing software that falsely made claims of threats on users' computers.

The Attorney General for Washington State has also filed lawsuits against a Texas-based company and its owner for misrepresentation of Registry Cleaner XP.[59] The lawsuit has asked for restitution for the victims of the scam, fines for the defendants, and recovery for damages for each violation.[60]

Under the Washington State Computer Spyware Act it is illegal to persuade a user to download software under the guise that it is necessary for the safe operation of his or her computer. In addition to requesting that the rogue security software creators and distributors cease all operations, the state also asks for monetary compensation to be provided for all victims of these scams.

In another case, in 2008 the head of a South Korean-based computer security company was charged with fraud by the Seoul Metropolitan Police Agency for the distribution of the rogue security software program Doctor Virus to over four million users.[61] The company is alleged to have made over $9.8 million over the course of three years in the scam.

In June 2009, a U.S.-based defendant and his company were required to pay more than $1.9 million to settle fraud charges with the Federal Trade Commission stemming from a rogue security software scam.[62] The defendants used deceptive advertising to mislead more than 1 million people into purchasing their rogue security applications, including such titles as WinFixer,[63] WinAntivirus, DriveCleaner,[64] XP Antivirus 2008 and ErrorSafe.[65]

The defendants placed advertisements for their rogue security software program on popular legitimate websites and on a major Internet advertising network and with brokers.[66] After receiving complaints that the banner ads contained code that would automatically install malicious software, the advertising network stopped placing advertisements for all security products. To bypass this, the operators created advertisements for legitimate companies, including a charity, and these advertisements were displayed for an IP address range associated with the advertising network company. For all other IP addresses outside of the range, it displayed the advertisement for the rogue security software program that contained code that automatically performed fake scans on the users' computers. The scan would report threats of spyware and illegal pornography, and then urge users to download and install the rogue security software program so that it could perform a more detailed scan. This second scan would also report that the computer was infected by the same threats as the first scan. Users were then directed to purchase a full copy for $39.95 to "fix" these false threats. In reality, no computer scans were conducted at any point and the threats that they detected were false and non-existent.

The settlement amount of $1.9 million represented the total gross revenue that the company realized from the scam. Moreover, the court order prohibited the defendants from engaging in deceptive advertising tactics and installing programs on consumers' computers.

In addition to government actions, some companies have also been effective in taking actions against rogue security software distributors and hosts. In August 2009, a Latvian ISP associated with rogue security software programs and the hosting of malicious activities (such as websites responsible for Web-based attacks and phishing sites) was taken offline after being disconnected by its upstream provider.[67] The ISP allowed customers to remain online even after they were

57-http://www.atg.wa.gov/pressrelease.aspx?&id=5926
58-http://www.symantec.com/security_response/writeup.jsp?docid=2006-041017-1914-99
59-http://news.bbc.co.uk/2/hi/technology/7645420.stm
60-http://www.pcworld.com/businesscenter/article/151640/washington_state_pursues_scareware_distributors.html

linked to malicious activities. As such, following complaints from Internet security researchers, the main provider informed the upstream provider to cease operations with the ISP or face sanctions.

61-http://www.theregister.co.uk/2008/03/04/south_korea_scareware_fraud_charges/
62-http://www.ftc.gov/opa/2009/06/winsoftware.shtm
63-http://www.symantec.com/business/security_response/writeup.jsp?docid=2005-120121-2151-99
64-http://www.symantec.com/business/security_response/writeup.jsp?docid=2006-062217-0726-99
65-http://www.symantec.com/business/security_response/writeup.jsp?docid=2006-012017-0346-99
66-http://www.ftc.gov/os/caselist/0723137/081202innovativemrktgcmplt.pdf
67-http://www.messagelabs.co.uk/download.get?filename=MLIReport_2009.08_Aug_FINAL.pdf

Prevalence of Rogue Security Software

To date, Symantec has detected over 250 distinct rogue security software programs. The following discussions are based on the top reported rogue security software programs that Symantec observed between July 1, 2008, and June 30, 2009. Of the top 50 most reported rogue security software programs that have been analyzed for this report, 38 of the programs were detected prior to July 1, 2008. The continued prevalence of these programs emphasizes the ongoing threat they pose to potential victims despite efforts to shut them down and raise public awareness. Each consumer report is considered to be an attempted and potentially successful scam. For example, during the period of this report, Symantec received reports of 43 million rogue security software installation attempts from the 250+ distinct samples. The results have been analyzed to provide insight into how certain aspects of the programs, such as advertising methods and regional distribution, may contribute to their prevalence.[67]

Top reported rogue security software

This section will discuss the top five of the most reported rogue security software programs observed by Symantec during this reporting period (table 2). The intention is to provide insight into methods of distribution of rogue security software for prevalence, examine related applications, discuss incidents related to the applications, and to highlight malicious activity originating from sites hosting the rogue security applications.

Rank	Name
1	Spyware Guard 2008
2	AntiVirus 2008
3	AntiVirus 2009
4	Spyware Secure
5	XPAntivirus
6	WinFixer
7	SafeStrip
8	Error Repair
9	Internet Antivirus
10	DriveCleaner

Table 2. Top reported rogue security software

Source: Symantec

Spyware Guard 2008

Spyware Guard 2008[68] was the most prevalent rogue security application that Symantec observed during this reporting period. First detected in October 2008, Spyware Guard 2008 uses deceptive Web advertisements that inform users that they have supposedly been exposed to malicious code threats. The advertisements advise users to "turn on protection," which will instead download and install the program if chosen. The downloaded program presents itself as a trial version that scans for and reports various threats (figure 11). After reporting false or exaggerated scan results, the software then asks the user to register and pay for a software license, purportedly enabling the removal of the reported threats. The website for Spyware Guard 2008 offers three different licenses, with costs marked at $49.95, $69.95, and $89.95.

Another distribution technique used by Spyware Guard 2008 is to inject links in innocuous search results for domains that redirect to websites for the rogue application.[69]

Figure 11. Spyware Guard 2008 fake scan results screen

Courtesy: Symantec

Spyware Guard 2008 was created by Pandora Software,[70] which has been identified as being responsible for a number of other rogue security applications, such as AntiVirus XP 2008, EasySpywareCleaner,[71] InfeStop,[72] Malware Protector 2008, SpyRid,[73] and WinIFixer. Pandora Software is believed to be associated with Bakasoftware, an affiliate network based in Russia.[74]Bakasoftware provides various services for its affiliates, including a range of installation methods to aid in scam distributions such as ActiveX controls, fake codecs, and fake online scanners. A list of earnings for Bakasoftware affiliates was published for a one-week period and the top earners purportedly made between $58,000 and $158,000.[75] Pandora is also reputed to act as a payment processor for purchases of misleading applications.[76]

Symantec also observed some unusual behavior on the part of Spyware Guard 2008 in that it was directing users to purchase legitimate software titles (figure 12).[77] This is also a scam, however, because the Web-based storefront is fraudulent and the software, if purchased, is never shipped to the victim. Symantec speculates that this may have been an attempt to gather credit card information. An additional possibility is that the scammers intended to sell pirated software, or did so for a short period, but subsequently stopped shipping the goods.

68-http://www.symantec.com/security_response/writeup.jsp?docid=2008-100114-4845-99
69-http://community.ca.com/blogs/securityadvisor/archive/2009/01/09/unabated-fraud-spyware-guard-2008.aspx
70-Note: The Pandora Software company mentioned in this report is solely affiliated with the distribution, publishing, and/or payment processing of misleading applications such rogue security software and is in no away affiliated with similarly named companies.
71-http://www.symantec.com/security_response/writeup.jsp?docid=2008-022916-2526-99
72-http://www.symantec.com/security_response/writeup.jsp?docid=2008-022916-3210-99
73-http://www.symantec.com/security_response/writeup.jsp?docid=2008-012117-0229-99
74-http://www.secureworks.com/research/threats/rogue-antivirus-part-2/
75-http://www.nytimes.com/2008/10/30/technology/internet/30virus.html?_r=1
76-http://ddanchev.blogspot.com/2009/06/diverse-portfolio-of-fake-security.html
77-http://www.symantec.com/connect/blogs/misleading-applications-supposedly-reselling-popular-software-titles

Figure 12. Spyware Guard 2008 advertising legitimate software

Courtesy: Symantec

Spyware Guard 2008 is not hosted on as many domains as has been observed with other samples (Symantec has observed four domains hosting Spyware Guard 2008 executables); however, other distribution methods have been noted. In particular, it was distributed by the Downadup.E worm (a variant of the original Downadup.C).[78] Additionally, Downadup.E was also observed to be distributing variants of Spyware Guard 2008.[79]

AntiVirus 2008 and AntiVirus 2009

AntiVirus 2008[80] was the second most reported rogue security application observed by Symantec during this reporting period, while AntiVirus 2009 was the third most reported. Because they are nearly identical variants from the same source, they will be addressed together here and referred to as AntiVirus 200X for the sake of discussion.

Antivirus 200X is designed to get installed on target computers a number of ways, including intentional downloads, misleading Web advertisements, drive-by downloads, and installation through malicious code. Once installed on a user's computer, AntiVirus 200X then performs a pseudo-scan of the system and falsely reports the discovery of numerous security threats (figure 13). The reported threats range from adware applications and spyware, to Trojans and viruses. AntiVirus 200X even reports the detection of rogue security software.

78-http://blogs.zdnet.com/hardware/?p=4131
79-Ibid.
80-http://www.symantec.com/security_response/writeup.jsp?docid=2008-050906-3727-99

Figure 13. Antivirus 2009 scan result page

Courtesy: Symantec

Upon completion of the mock scan, the user is presented with options to deal with these threats, including to "Remove all threats now" or to "Continue unprotected." Selecting the threat removal option will result in the user being presented with a prompt to purchase and to enter a registration key to fully activate and unlock the threat removal features; choosing not to pay will result in AntiVirus 200X continually bombarding the computer desktop with alarmist messages (figure 14).

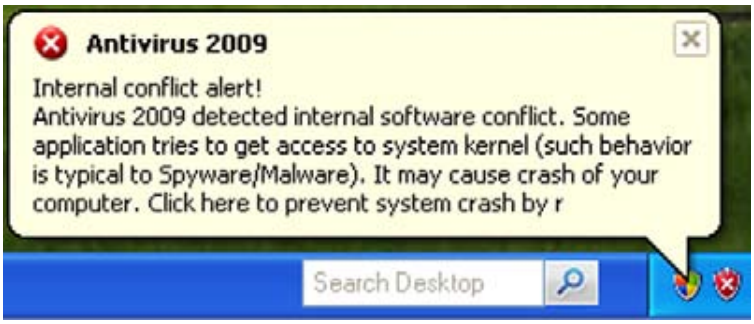


Figure 14. AntiVirus 2009 taskbar alert

Courtesy: Symantec

Furthermore, AntiVirus 200X incorporates a window that closely mimics the legitimate Microsoft®Windows® Security Center service (figure 15). When the software is unregistered, the false security center lists virus protection as "not found," even if there actually is a legitimate security application enabled, and explains that AntiVirus 200X is not fully enabled. It also presents a link for the user to click in order to purchase a license.

Figure 15. AntiVirus 200X Security Center (left) vs. Microsoft Windows Security Center (right)

Courtesy: Symantec

In addition to the described methods used by AntiVirus 200X to appear legitimate, the application will prompt unregistered users that a new database of threat signatures should be downloaded to update the software (figure 16). Choosing to update the program presents the previously described registration window.

Figure 16. AntiVirus 2009 software update alert

Courtesy: Symantec

AntiVirus 2008 was identified in May 2008, while Antivirus 2009 was detected just two months later, in July. Efforts by legitimate security firms to raise awareness and reduce the number of potential victims of the original program may have been cause for the scam authors to release a rebranded version. The rebranding may also have been an attempt to seem as though an upgraded version was available. This may suggest that the scam authors actively monitor the success of their scams and modify them accordingly. This level of involvement may be a contributing factor in the relative success of the scams as well.

Symantec has observed 218 unique domains hosting AntiVirus 2008 executables. Sites hosting AntiVirus 2008 were also observed to be hosting these other threats and rogue applications:

- AntiVirus 2009
- Bloodhound.Exploit.196[81]
- Downloader.Psyme[82]
- InternetAntivirus[83]
- SecureExpertCleaner[84]
- Trojan.Fakeavalert
- WinFixer[85]

A number of the threats detected on sites hosting AntiVirus 2008 are noteworthy because of their involvement in malicious activity. Bloodhound.Exploit.196 is a Symantec heuristic signature that detects exploits for a series of vulnerabilities in Adobe® Acrobat® and Adobe Reader®. The first series of vulnerabilities was discovered in February, 2008.[86] The second series of vulnerabilities was discovered in May, 2009.[87] (Both series have since been patched.) Downloader.Psyme is a downloader that attempts to transfer various malicious executables to the affected computer. InternetAntivirus, SecureExpertCleaner, and WinReanimator are other rogue security applications. The sites hosting AntiVirus 200X have also been observed to be distributing other forms of malicious code. Thus, in addition to the risk posed by the rogue security applications, visitors to these sites could be exposed to exploitation by client-side vulnerabilities or be the target of drive-by downloads.

One of the threats identified on sites hosting AntiVirus 2008 is the Trojan Fakeavalert.[88]FakeAvalert was discovered in October, 2007. Once on a victim's computer, it produces prompts with false alerts about the security status of the compromised computer and prompts the user to run a full scan. If the user authorizes the scan, Fakeavalert launches the user's browser and directs it to a site that tells the user that his or her computer is "infected," along with containing a "Fix now" button that, if clicked, will prompt a download of the rogue security software program, AVSystemCare.[89]

Some characteristics of AVSystemCare that make it appear legitimate are worth noting. This includes the presence of an installation wizard and an End-User License Agreement (EULA), to which the user actually must agree to in order to proceed with the installation. Symantec has observed over 100 clones of this program, with names such as Antispywaresuite, Antiworm2008, and so on. In addition to disabling access to websites of legitimate security vendors, AVSystemCare also disables access to adware sites, which may be an attempt by it to obstruct access to its competitors.

Symantec has observed 179 unique domains hosting AntiVirus 2009 executables. Sites hosting AntiVirus 2009 have also been observed to host the following threats and rogue applications:

- AntiVirus 2008
- Bloodhound.Exploit.196
- Bloodhound.Exploit.213[90]
- IEDefender
- Trojan.Blusod[91]
- Trojan.Fakeavalert
- Trojan.Virantix[92]
- Trojan.Virantix.C[93]

81-http://www.symantec.com/security_response/writeup.jsp?docid=2008-080702-2357-99

In many cases, other misleading applications and threats may be hosted together. This may indicate that the website has been used to launch various attacks and scams. In some cases, malicious software and exploits are hosted on the same website for the purpose of distributing scams. Some of the threats and rogue applications that have been hosted on the same sites as AntiVirus 2009 are worth noting further: Bloodhound.Exploit.213 is a Symantec heuristic signature that detects exploits for a vulnerability in Adobe Acrobat;[94] Trojan.Blusod displays a "blue screen of death" screensaver and false warnings about security threats on the computer and also attempts to download a variant of Zlob from malicious sites; the Trojans Virantix and Virantix.C display false security warnings and also attempt to download additional software to affected computers; Virantix.C also attempts to install the WinReanimator rogue security application on computers.

Spyware Secure

Spyware Secure[95] was the fourth most prevalent rogue security application that Symantec observed during this reporting period. Spyware Secure has been distributed mainly through a single domain that hosts installation executables. Symantec first discovered Spyware Secure in September, 2007. The length of time that the scam has been distributed, in addition to the fact that the main site hosting the executables is still operational, may be contributing factors to the prevalence of this sample.

Spyware Secure is a good example of a scam that tries to socially engineer users into downloading a rogue security application by convincing them that their computers are not protected from, as the ad reads, "spywares" (figure 17). The interface cites statistics from a legitimate security software company in an attempt to scare users into installing the program. It also list common occurrences that many computer or Internet users are likely to encounter such as occasional crashes, slow navigation, and unwanted pop-ups.

Figure 17. SpywareSecure registration screen

Courtesy: Symantec

82-http://www.symantec.com/security_response/writeup.jsp?docid=2004-040112-5204-99
83-http://www.symantec.com/security_response/writeup.jsp?docid=2008-081212-1113-99
84-http://www.symantec.com/security_response/writeup.jsp?docid=2008-072807-2626-99
85-http://www.symantec.com/security_response/writeup.jsp?docid=2005-120121-2151-99
86-http://www.securityfocus.com/bid/27641
87-http://www.securityfocus.com/bid/34169
88-http://www.symantec.com/security_response/writeup.jsp?docid=2007-101013-3606-99
89-http://www.symantec.com/security_response/writeup.jsp?docid=2007-061509-3222-99
90-http://www.symantec.com/security_response/writeup.jsp?docid=2008-110718-2219-99
91-http://www.symantec.com/security_response/writeup.jsp?docid=2008-062711-5534-99
92-http://www.symantec.com/security_response/writeup.jsp?docid=2007-073011-3204-99
93-http://www.symantec.com/security_response/writeup.jsp?docid=2008-050916-1055-99
94-http://www.securityfocus.com/bid/30035
95-http://www.symantec.com/security_response/writeup.jsp?docid=2007-091719-0351-99

Once a rogue application becomes prevalent, there is also a risk that scam distributors may capitalize on its popularity to advertise other scams that purport to remove the now widespread application. For example, searches for Spyware Secure return a sponsored link that advertises applications that claim to remove the threat (figure 18).

Figure 18. SpywareSecure search results

Courtesy: Symantec

Similar cases have been reported where scam distributors have advertised software that purports to remove rogue security software offered by competitors.[96] Some scams even purport to remove rebranded versions of the same program.[97] This demonstrates competition between scam authors and that they may not be concerned with creating the illusion of a trustworthy brand identity, but instead are attempting to capitalize on the confusion resulting from the distribution of multiple rogue products with similar names and interfaces.

As individual rogue applications are deemed untrustworthy, new versions are often cloned by the same developers and distributed with the promise of removing the old versions. By disassociating themselves from other rogue applications, the scam authors can create confusion and make it difficult to discern which security software programs are authentic. Furthermore, cautious users may be led to distrust advertisements for security applications in general due to the prevalence of false and malicious advertising. This could adversely affect the ability of new, legitimate security software products to establish a trustworthy brand in the marketplace.

XP Antivirus

XP AntiVirus was the fifth most observed rogue security application by Symantec during this reporting period. XP AntiVirus was, at one point, distributed by the Russian Business Network (RBN),[98] and was also one of the rogue security applications targeted by the FTC complaint against Innovative Marketing, Inc. and ByteHosting Internet Services, LLC.[99] These companies were also responsible for distributing other rogue applications including WinAntivirus, DriveCleaner, ErrorSafe, and WinFixer. WinFixer and ErrorSafe are noteworthy because of an incident where they were distributed through banner advertisements in Windows Live™ Messenger.[100]

96-http://ddanchev.blogspot.com/2008/11/diverse-portfolio-of-fake-security_12.html
97-http://ddanchev.blogspot.com/2009/04/diverse-portfolio-of-fake-security_16.html
98-http://ddanchev.blogspot.com/2008/03/rogue-rbn-software-pushed-through.html
99-http://ftc.gov/opa/2008/12/winsoftware.shtm
100-http://msmvps.com/blogs/spywaresucks/archive/2007/02/18/591493.aspx

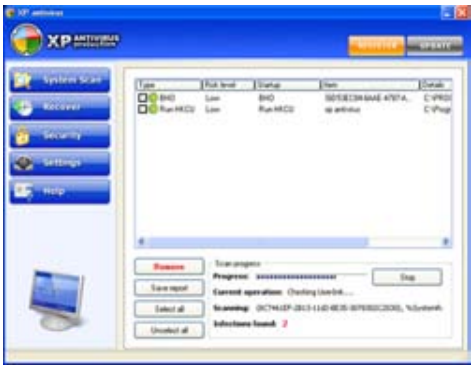


Figure 19. XP AntiVirus interface

Courtesy: Symantec

XP AntiVirus was observed by Symantec to be hosted on 73 unique domains. Sites hosting XP Antivirus have also been observed to host the following threats and rogue applications:

- AntiVirus 2008
- AntiVirus XP 2008
- Trojan.Fakeavalert
- Trojan.Galapoper.A[101]
- Trojan.Zlob

A few of the rogue applications and threats listed above are worthy of discussion. AntiVirus XP 2008 was implicated in an incident where search engine advertisements were poisoned with a number of client-side exploits to install AntiVirus XP 2008.[102] Trojan.Zlob was also found on sites that were hosting XP AntiVirus.

Many of the samples discussed here are hosted on sites that website reputation services have flagged as having a reputation for malicious activity.[103] While this malicious activity is not necessarily directly associated with rogue security applications, it is likely that scam distributors are reusing these domains for various rogue software and malicious code distribution operations. This may be to extract the maximum value from the domains under their control. Exploits targeting client-side vulnerabilities are also present on some sites, which aid in drive-by downloads of malicious software and rogue security applications. In particular, browser plug-in vulnerabilities are often exploited in such attacks. These vulnerabilities are a potent means of distributing rogue security software due to the large number of users affected. Symantec discusses the prevalence of browser plug-in vulnerabilities in Volume 14 of the *Symantec Internet Security Threat Report*.[104]

Additional noteworthy rogue security software samples

As well as the discussion above on the most widely reported rogue security samples observed by Symantec, there are two other examples worth additional mention that Symantec observed during this reporting period.

101-http://www.symantec.com/security_response/writeup.jsp?docid=2006-042013-1813-99
102-http://sunbeltblog.blogspot.com/2008/08/xp-antivirus-2008-now-with-sploits.html
103-http://safeweb.norton.com/
104-http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_internet_security_threat_report_xiv_04-2009.en-us.pdf : p. 40

FileFix Professional

FileFix Professional[105] is a rogue security application that is installed by the Trojan Xrupter.[106]Xrupter is a malicious executable that is installed by Vundo Trojan variants.[107] The rogue security application works in tandem with Xrupter. When Xrupter is installed on a victim's computer, it encrypts personal documents. The Trojan then displays warnings to the user about corrupt documents with a button to repair them (figure 20).

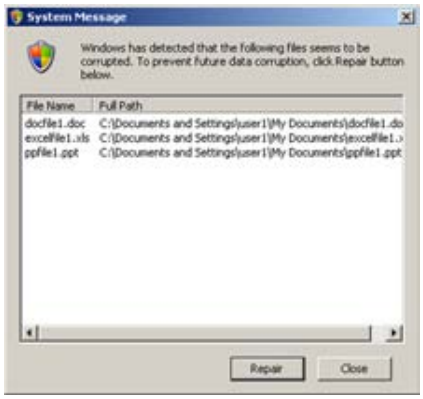


Figure 20. Trojan.Xrupter results detecting corrupted files

Courtesy: Symantec

When the "Repair" button is clicked, the user is directed to obtain FileFix Professional (figure 21). However, if the user opts to obtain FileFix Professional, a demo version is instead presented and the user must pay to register for a full version in order to recover the files. Instead of attempting to sway the user with false security alerts, this variation of the rogue security software business model attempts to extort money from affected users in return for decrypting their documents, which were initially encrypted when Xrupter was installed.

Figure 21. FileFix Professional

Courtesy: Symantec

The connection to the Vundo Trojan is noteworthy. Once computers are affected by Vundo, a number of misleading applications and threats may be installed. Vundo itself has been distributed by other malicious code samples. In February

105-http://www.symantec.com/business/security_response/writeup.jsp?docid=2009-032209-4419-99

2009, Symantec observed a spike of Vundo infections as a result of the W32.Ackantta.B@mm mass-mailing worm.[108] These multiple layers of misdirection help Vundo variants, related threats, and misleading applications evade detection. Vundo variants have also been detected exploiting vulnerabilities as a means of propagating, such as a vulnerability in Microsoft Internet Explorer®.[109]

Malicious software such as the Vundo and Zlob Trojans that are used to distribute rogue security software are effectively acting as affiliates. This implies that their revenue generation model is similar to other affiliate programs, whereby commissions are generated on a per-install basis. As noted earlier, Vundo was listed as the most prevalent malicious code sample for 2007 and 2008 in Volume 14 of the *Symantec Internet Security Threat Report*.[110] One of the reasons Zlob and Vundo were originally created was to download and install adware onto users' computers, likely earning money for the creators through adware affiliate programs. Legislative measures have reduced the profitability of adware scams and may have led to the modification of these Trojans for rogue security software scams instead. This may have contributed to the success of numerous misleading applications that have been associated with Zlob and Vundo. Through these methods, it is possible for malicious code authors to monetize their creations.

Mac OS X rogue security applications

Rogue security applications have not been limited to Microsoft Windows operating systems. In January, 2008, a rogue security application targeting Mac OS® X users named MacSweeper[111] was discovered (figure 22). Symantec believes that MacSweeper is a Mac OS X clone of the MalwareAlarm Scanner rogue security application that runs on Microsoft Windows.[112]

Figure 22. MacSweeper "scan" results page

Courtesy: Symantec

A further variant was released for Mac OS X entitled iMunizator.[113] When run, iMunizator flags a number of safe system binaries as problematic and prompts the user to pay a licensing fee to fix the problems on the computer. iMunizator is a fairly simple rogue security application that uses UNIX command-line utilities to find random files on the computer that it

106-http://www.symantec.com/business/security_response/writeup.jsp?docid=2009-032207-0838-99
107-http://www.symantec.com/connect/blogs/offer-too-good-refuse-courtesy-vundo

will flag as problematic. This is in contrast to many rogue security software applications that purport to remove specific well-known security risks and malicious code.

These Mac OS X samples lack the degree of social engineering and functionality demonstrated in other prevalent samples targeting Microsoft Windows users. It is apparent that scam developers are experimenting with the Mac OS X platform, but that the observed samples lack the sophistication of those targeting Microsoft Windows users, which have generated far more success and revenue.

Innovations such as encrypting the user's data in exchange for a ransom payment and targeting Mac OS X users have not resulted in rogue security applications that are highly prevalent. Neither FileFix Professional nor MacSweeper/iMunizator rank among the top reported samples observed by Symantec. While this may be a matter of distribution, it is also likely that conventional tactics are profitable enough that novel techniques are not required to increase the revenue of scammers.

Top rogue security software by region

For this measurement, Symantec analyzed the regional distribution of the consumer reports between July 1, 2008, and June 30, 2009 of the top 50 rogue security software programs (figure 23). During this period, 61 percent of rogue security software scams observed by Symantec were attempted on users in the North America (NAM) region, 31 percent occurred in the Europe, the Middle East, and Africa (EMEA) region, six percent occurred in the Asia-Pacific/Japan (APJ) region, and two percent occurred in the Latin America (LAM) region.

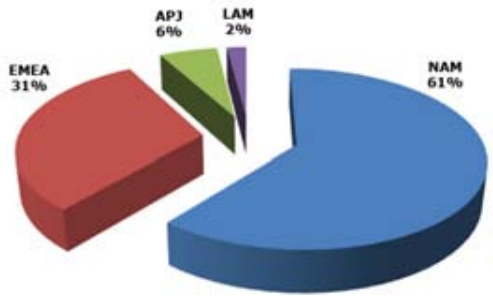


Figure 23. Percentage of rogue security software distribution, by region
Courtesy: Symantec

The variance in the percentages of reported scams between each region suggests that regional boundaries affect the distribution of rogue security software. This may be related to the amount of malicious activity in general that affects these regions. As discussed in Volume 14 the *Symantec Internet Security Threat Report,* the majority of malicious activity globally is detected in the NAM and EMEA regions.[114] Considering that rogue security software is often installed on computers by malicious code or through drive-by download attacks, the prevalence of malicious activity in NAM and EMEA may be a contributing factor in the distribution of rogue security software programs.

108-http://www.symantec.com/security_response/writeup.jsp?docid=2009-022520-1425-99
109-http://www.securityfocus.com/bid/11515
110-http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_internet_security_threat_report_xiv_04-2009.en-us.pdf : p. 60
111-http://www.symantec.com/security_response/writeup.jsp?docid=2008-011613-5206-99
112-http://www.symantec.com/connect/blogs/attack-clones-ii
113-http://www.symantec.com/connect/blogs/cloning-shop-mac-users-now-open
114-http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_internet_security_threat_report_xiv_04-2009.en-us.pdf : p. 17 and 31

An additional factor contributing to the prominence of NAM and EMEA in this measurement may be the regional difference in per-installation prices paid for affiliate distribution, as discussed earlier in this report. For example, the price-per-install for North America is as much as 10 times that of the price-per-install for Latin America, which would likely incline scam distributors toward distributing these programs where the returns will be greater.

The overwhelming number of attempted rogue security software scams reported in North America may also be due to the majority of programs being created in English, the primary language for the majority of people in the region. Although some programs target other languages—such as CodeClean,[115] which targets Korean users (figure 24)—the majority of the programs that Symantec observed during this reporting period have been developed and distributed in English.

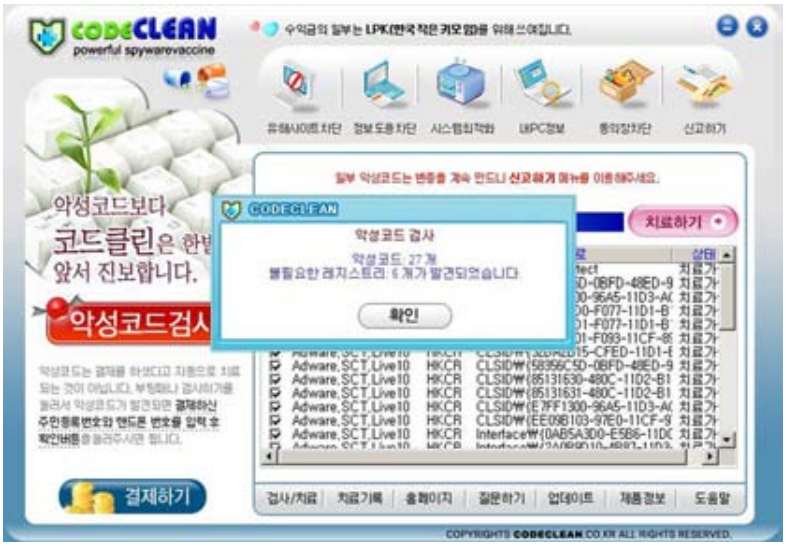


Figure 24. CodeClean rogue security application that targets Korean language users

Courtesy: Symantec

Top rogue security software installation methods

There are two main ways that rogue security software programs can get onto a user's computer, as described earlier in this report. This is either through an intentional download, where the user is persuaded to download and install the program, or via an unintentional download, where the download occurs without the user's permission or knowledge. This section will examine the prevalence of the distribution methods used by the top 50 rogue security software programs observed by Symantec during the period of this report. It is worth noting that distribution methods are not mutually exclusive and that, in nearly 70 percent of reports for the top 50 programs, both distribution methods were employed.

The most common distribution method observed by Symantec during this reporting period was intentional downloads, which were employed in 93 percent of the attempts of the top 50 rogue security software scams. One reason that this method of distribution is popular may be because many users are suspicious of unauthorized installation procedures or programs that appear on their computers without their interaction.

Legal implications could also be a factor that makes intentional downloads a popular distribution method. Downloading and installing a program onto a computer without the user's consent is illegal in some countries. However, if a program is

115-http://www.symantec.com/security_response/writeup.jsp?docid=2007-013111-5717-99

downloaded and installed intentionally, the onus could fall on the user and not the scam distributor. Scam perpetrators operating where such restrictions exist may opt to reduce legal liability as much as possible and rely on intentional downloads. Some rogue security software programs implement EULAs that must be accepted during installation; by accepting a EULA, the consumer may potentially be releasing the scam distributor from legal implications.

Unintentional downloads were employed in 76 percent of the attempts in the top 50 rogue security software scams observed by Symantec during this reporting period. As noted earlier, an unintentional download occurs when malicious code is downloaded onto a computer without the interaction or knowledge of the victims, such as via drive-by download attacks, or when users have been duped into downloading and installing what they think are legitimate applications, such as missing video codecs. These downloads often contain staged downloaders that, once the user's computer is compromised, download and install additional programs such as rogue security applications.

The lower percentages for unintentional downloads compared to intentional downloads as a distribution method may also be a reflection of the relative skill levels of some scam authors or distributors. The development of the code required for intrusive distribution might require a deeper technical ability than some of these people are able to learn or care to use. Although scam distributors may pay malicious code developers per install to distribute rogue security software, some of them might not have the desire or necessary contacts to do so. Additionally, some scammers may be effective at using other means to lure in users, such as social engineering skills, and thus do not require the technical demands of programming code.

Additionally, some malicious code authors may have been slow to realize the revenue generating potential of rogue security software scams. With Trojans such as Zlob and Vundo being successful and effective affiliates for rogue security software, there may be an increase in malicious code as a distribution method in the future as other authors realize the earning potential from these scams.

Top rogue security software advertising methods

Scam distributors use many methods to tempt users into downloading and installing rogue security software. This section examines the prevalence of certain advertising methods used in the top 50 rogue security software programs that Symantec observed during this reporting period.

The most common advertising method used by the top 50 rogue security software programs that Symantec observed during this reporting period was through dedicated websites, which were used in 93 percent of scams. It should be noted that although the percentage of advertising using scam websites is the same as the percentage of distribution by intentional downloading, discussed in "Top rogue security software installation methods," above (with both being 93 percent), the results are coincidental. While this method of advertising is closely related to distribution by intentional downloads (that is, if a website exists, the program can most likely be downloaded there), the ability to also download programs from third-party hosts means that a particular scam does not necessarily require a website in order to be intentionally downloaded. Also, some websites dedicated to rogue security software act solely as a launching point for drive-by download attacks, forgoing the use of distribution by intentional download altogether.

The second most common advertising method for rogue security software observed by Symantec during this reporting period was Web advertising, which was used in 52 percent of the attempted rogue security software scams. While this may

suggest that Web advertisements are not as effective as dedicated websites for promoting rogue security software, more Web advertisements were observed for the top 10 programs than in the remaining 40 of the top 50 programs combined. This may indicate that well-deployed Web advertisements can be a very effective method of distributing rogue security software.

Although the reverse is not true, nearly all of the programs that use Web advertisements also use malicious code and drive-by downloads (or both) as a distribution method. For example, the WinFixer scam—the sixth most reported scam observed by Symantec during this reporting period—uses both a website and Web advertisements in addition to being distributed by malicious code, including by the Vundo Trojan, and by both intentional and drive-by downloads. This may indicate that Web advertisements are more effective as launch points for intrusive distribution tactics than they are for luring intentional downloads. This may also explain why the percentage of rogue security software programs that use Web advertisements is similar to the distribution method percentages of malicious code and drive-by downloads.

Analysis of Rogue Security Software Distribution

This section of the *Symantec Report on Rogue Security Software* will expand on the overview of this topic earlier in this report. It will discuss specific examples of how rogue security software applications are distributed, presenting more information about specific incidents and insight into the infrastructure of rogue security software distribution.

Given that profit is a motive behind most rogue security software scams, the success of these scams depends on convincing consumers to purchase the fake software. To do so, scam creators try to convince users of exaggerated or non-existent threats on their computers and that the fake security software is a valid solution. As such, scam software often mimics the appearance of legitimate security software. A common tactic is to present an interface that is similar to the Microsoft Windows Security Center, as is shown in the discussion on AntiVirus 2008 and AntiVirus 2009, above (figure 15).[116] The Security Center has been a feature of Windows since the release of XP Service Pack 2, with minor changes to the interface in Windows Vista®; users are likely to be familiar with this interface and might consider the false applications that mimic its appearance to be the real thing.

As noted, other scam software may mimic the appearance of well-known, genuine security software. To facilitate this, scam authors create user-interface templates that can be reused and modified to create new variations of the scam. The templates enable the customization of various aspects of the scam, such as the title of the rogue application, the text to display, and the appearance of the interface. This helps scam creators to easily re-brand rogue applications once they are identified and exposed as scams. Templates also often incorporate social engineering tactics to scare users. In one example, a fake "blue screen of death" interface is presented that urges the user to solve this critical issue by installing a rogue security application named SystemSecurity.[117] Templates also allow for easy localization of scams for distribution in new markets. For example, the fake "blue screen of death" template has also been observed localized into Arabic.[118]

Making the rogue software modular and comprised of re-usable components to perpetrate different variations of scams reduces the time required to develop and deploy new scams. Additionally, it allows different skills to be outsourced, such as the design of templates and social engineering angles. Symantec observed similar behavior with phishing scams in its study of the underground economy.[119] It was observed that different individuals and groups may develop modular components of phishing methods such as scam letters and phishing website templates, which may then be sold as part of a customized package to scam distributors. This tactic is also used by websites designed to deliver malicious code.[120] The same principle can be applied not only to the applications themselves, but also to the websites that distribute the applications.

Scam distributors also attempt to have the websites for their rogue security applications appear at the top of search engine results to increase the chances of being noticed—and considered genuine—by users. If these websites can appear among legitimate websites in search results for malicious code and security-related search queries, it may be more difficult for users to distinguish legitimate sites from those that are malicious. For example, in March 2009, distributors of rogue security applications employed this tactic by injecting links to their software in Downadup-related search results.[121] In the same month, scam distributors also manipulated search results for a number of keywords related to antivirus and desktop applications.[122]

116-http://www.microsoft.com/windowsxp/using/security/internet/sp2_wscintro.mspx
117-http://blogs.zdnet.com/security/?p=3912
118-http://ddanchev.blogspot.com/2009/08/scareware-template-localized-to-arabic.html
119-http://eval.symantec.com/mktginfo/enterprise/white_papers/b-whitepaper_underground_economy_report_11-2008-14525717.en-us.pdf : p. 30
120-http://ddanchev.blogspot.com/2008/07/template-ization-of-malware-serving.html

Scam distributors also capitalize upon interest in current events to lure users into visiting websites that host rogue security software. For example, in May, 2009, one scam attempted to exploit public interest in the H1N1 virus outbreak as a means to distribute rogue security software.[123] Symantec also observed malicious code authors exploiting interest in the H1N1 virus by developing and distributing a PDF with FAQs on the flu that also included a payload of malicious code.[124] Spam distributors were also observed exploiting the headlines about H1N1.[125] This demonstrates that rogue security software scam perpetrators are willing to use the similar social engineering tactics employed by spammers and malicious code authors.

Search engines are a common means of distributing rogue security software. Black hat SEO operations are conducted to push sites that host rogue security applications to the top of search engine indexes.[126] A common black hat SEO tactic involves planting links to rogue security software websites on legitimate websites, such as blogging services, wikis, forums, and social networking sites. This tactic exploits search engine indexing algorithms that determine the relevancy of a website by the number of links that point to it. This process is typically automated by software that can visit these various Internet locations and add content. Because this activity is considered a form of spam, many websites implement measures such as CAPTCHA schemes to prevent content from being added automatically.[127] CAPTCHA schemes are used to ensure that human users, and not automated systems, are adding the content. This in turn has resulted in a number of efforts to bypass CAPTCHA that range from exploiting weaknesses in CAPTCHA algorithms to outsourcing the task of manually solving CAPTCHA challenges.[128]

Other black hat SEO tactics include link farming, keyword stuffing, and cloaking: link farming is an SEO tactic used to increase search rankings by having a large group of websites include reciprocal links to each other; keyword stuffing involves placing long lists of often irrelevant keywords into Web page content; cloaking involves creating website content specifically for search engine website crawlers and which is different than the content accessible to users, which may cause search engines to index the site based on misleading content and potentially improve search rankings. Black hat SEO campaigns have also been known to exploit vulnerabilities in websites such as cross-site scripting.[129] In one reported example, vulnerabilities in a popular blogging platform were exploited to promote rogue security software.[130] Scam distributors also purchase keywords from search engines in order to boost the ranking of their scam websites and so that the websites will appear as valid, "sponsored" results.[131]

Rogue security software distributors use these black hat SEO tactics in combination with other techniques such as typo-squatting. Typo-squatting involves hosting sites with domain names that are similar to sites the scan authors are trying to spoof. Mistyping a URL may lead a user to the spoofed site instead of the legitimate website the user is trying to reach.

Malicious or false search engines have also been employed. To get users to use the illegitimate search engine, they are enticed to search for a special file, usually a topical video or the like. When the user searches with one of these fake search engines, the results instead mislead the user to websites that host malicious code and rogue security software.[132]

Affiliate networks can provide the scam developers with the talent and resources necessary to distribute their software using the tactics discussed above. In turn they may rely on resources in the underground economy to launch spam and black hat SEO campaigns. This may include purchasing lists of email addresses in bulk, spam proxies, credit cards to register domains in bulk, etc. When activities such as the development and distribution of rogue security software are

121-http://www.symantec.com/connect/blogs/downadup-related-search-indexes-poisoned-fake-av-sites
122-http://www.symantec.com/connect/blogs/yahoo-sponsored-search-results-leads-misleading-websites
123-http://ddanchev.blogspot.com/2009/05/dissecting-swine-flu-black-seo-campaign.html
124-http://www.symantec.com/connect/blogs/malicious-code-authors-jump-swine-flu-bandwagon

monetized and begin to generate revenue, the demand for other products and services in the underground economy increases as well.

Internet advertising networks have been used as a means to distribute scams. The legitimate appearance of rogue security applications may allow scam distributors to penetrate Internet advertising networks. The advertisements are likely to remain on the networks until the software being advertised is exposed as fraudulent. Additionally, scam distributors have also employed "malvertising" tactics.[133] In one observed attack, malicious advertisements were found to be exploiting a client-side vulnerability.[134] The advertisement redirected users to a site that exploited a vulnerability in Adobe Reader (since patched) via a malicious PDF document. Upon exploitation, the rogue security application Anti Virus 1 was installed. The attack also changed the system "hosts" file to redirect users to a site advertising further rogue applications.[135] In another attack, a malicious Flash advertisement that exploited a client-side vulnerability was distributed through an advertising network to a number of high-profile websites.[136] In one additional example, the advertising network for a news site was serving advertisements that prompted users to install rogue applications.[137]

Such attacks damage the reputation of not only the advertising networks, but potentially of the websites that circulate the malicious advertisements. In addition to the negative press surrounding such incidents, website reputation services may flag these sites as disreputable or suspect. Some browsers and security software will check website reputation databases before letting users browse to a website, thus potentially affecting legitimate traffic to flagged sites. Additionally, advertising revenue could be lost as users begin to distrust the advertising networks and implement security measures to block their advertisements.

In order to collect registration and/or subscription fees from consumers who have purchased rogue security software, scam perpetrators need online payment processing services. Since the payment services used are often legitimate, there is a constant threat that the payment service provider will discover that its service is being used for fraud. This is one reason why rogue applications are often re-branded, to avoid credit card chargebacks and payment reversals that may ultimately draw attention to the scam. However, rogue payment processors have also been established to serve affiliate networks who distribute rogue security software.[138] Due to their illicit nature, these rogue payment processing services run the risk of being shut down once their activities are discovered and are often short-lived.

In order to further evade discovery, scam payment processing often occurs through a number of gateway websites registered under different domain names that will redirect to the actual payment processor for the scam.[139] The domains are registered under a variety of email addresses to make it appear as though multiple individuals own the domains. Scammers can acquire email addresses by means such as purchasing them in bulk in the underground economy or by the automated generation of email accounts through popular Web-based email services. Similar approaches are used to register domain names for hosting the scam software, as is discussed next in "Analysis of rogue security software servers." Distributors of rogue security software may register domains with domain registrars in places where enforcement is perceived to be weak or where anonymous registration services are offered.[140] Rogue ISPs such as the RBN have also been

125-http://www.symantec.com/connect/blogs/swine-flu-outbreak-headlines-used-spammer-s-gain
126-SEO (Search Engine Optimization) is a process for making websites more popular in search engine results. Black hat SEO uses search optimization techniques that are considered unethical by the mainstream SEO community, which may include spamming and other questionable practices. For an overview of SEO techniques and guidelines, please see: http://www.google.com/support/webmasters/bin/answer.py?hl=en&answer=35291
127-CAPTCHA stands for "Completely Automated Public Turing test to tell Computers and Humans Apart". CAPTCHA schemes often take the form of an image containing characters that must be entered before the user can perform an action such as creating an account or submitting content on a website.
128-http://ddanchev.blogspot.com/2008/08/exposing-indias-captcha-solving-economy.html
129-http://ha.ckers.org/blog/20060608/xss-redirects-and-seo/
130-http://pandalabs.pandasecurity.com/archive/New-Blackhat-SEO-attack-exploits-vulnerabilities-in-Wordpress-to-distribute-rogue-antivirus-software.aspx
131-http://blogs.zdnet.com/security/?p=1995
132-http://www.computerweekly.com/Articles/2009/05/07/235935/cybercrooks-develop-own-search-engines-to-burgle-users.htm
133-Malvertising is a term to describe the practice of malicious advertising which includes tactics such as obfuscating malicious content in Flash advertisements or embedded exploit code into advertising content
134-http://www.eweek.com/c/a/Security/Attackers-Infect-Ads-With-Old-Adobe-Vulnerability-Exploit/

involved in various aspects of scam development and distribution. This includes hosting domains that distribute rogue security applications.[141]

Scammers also benefit by phishing personal information from users who register rogue applications. Information such as email addresses, credit card details, and payment processing credentials can be used for further fraudulent activities or sold in the underground economy. In this manner, a single scam can be used to generate revenue in different ways. Furthermore, fraudulent activities such as credit card and payment processing fraud can help to finance the startup costs of additional scams.

135-The system hosts file maps IP addresses to hostnames. The system hosts file is often consulted before domain name server lookups to resolve a hostname. This means that mappings in the hosts file often take precedence over DNS lookups; malicious code often employs the tactic of changing hostname to IP address mappings so that users are redirected to malicious sites or blocked from visiting sites where security updates and security software are available.
136-http://blogs.zdnet.com/security/?p=1815
137-http://blogs.zdnet.com/security/?p=3140
138-http://ddanchev.blogspot.com/2009/01/diverse-portfolio-of-fake-security.html
139-http://ddanchev.blogspot.com/2009/07/diverse-portfolio-of-fake-security.html
140-http://voices.washingtonpost.com/securityfix/2008/09/estdomains.html
141-http://rbnexploit.blogspot.com/2007/10/rbn-top-20-fake-anti-spyware-and-anti.html

Analysis of Rogue Security Software Servers

In this section, Symantec conducted a geographic analysis of servers hosting rogue security software. This analysis is not meant to represent all rogue security software servers; instead the goal was to identify any emerging patterns in the way these servers are created, managed, and interconnected with each other. The data was collected in a two-month period over July and August 2009.

For this measurement, Symantec analyzed 6,500 DNS entries pointing to 4,305 distinct IP addresses hosting rogue security software servers.[142] At least 45 percent of these domains were registered through just 29 out of several hundred existing domain registrars. This may indicate that rogue security software distributors are choosing specific registrars, possibly because of perceived lax security or oversight of the registration of names.

The DNS entries resolving to these IP addresses were first identified by monitoring DNS activity across the servers. From this, an additional 187,514 DNS entries associated with rogue security applications were observed, for a total of 194,014 domain names. In total, 2,677 Web servers hosting domains (as identified by their unique IP addresses) were identified as dedicated to serving only rogue security software, an additional 118 Web servers hosted rogue security software along with domains that served malicious code, and the remaining 1,510 IP addresses served malicious code along with innocuous domains.

Of the servers hosting rogue security software that Symantec geographically located, 53 percent were in the United States, far more than any other country (table 3 and figure 25). The high ranking of the United States may be due to the methods for identifying rogue security software sites, which more easily identified English-language sites than sites marketing scams in other languages. Germany ranked second in this survey, accounting for 11 percent of the total servers hosting rogue security software identified by Symantec. This ranking may be due to Germany being the top country in EMEA for broadband subscribers and a major broadband connection hub.

Rank	Country	Percentage
1	United States	53%
2	Germany	11%
3	Ukraine	5%
4	Canada	5%
5	United Kingdom	3%
6	China	3%
7	Turkey	3%
8	Netherlands	2%
9	Italy	2%
10	Russia	1%

Table 3. Servers hosting rogue security software, by country

Source: Symantec

142-The Domain Name System (DNS) is a hierarchical naming system for computers, services, or any resource connected to the Internet or a private network.

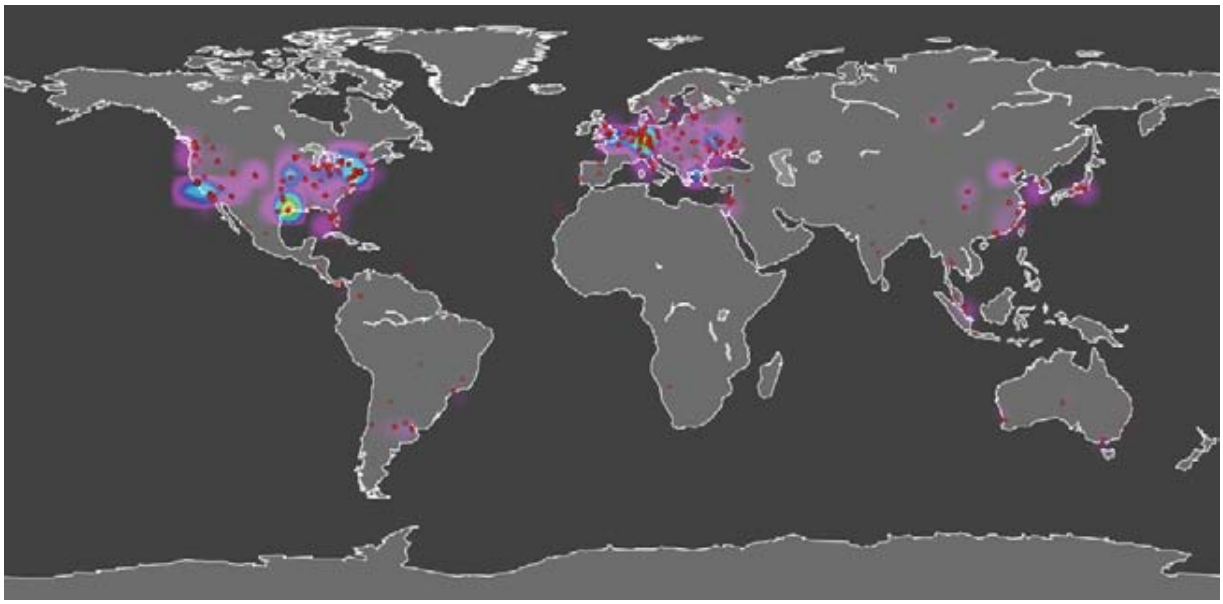


Figure 25. Global distribution of rogue security software servers[143]

Source: Symantec

After analyzing the distribution of the servers hosting rogue security software and their corresponding DNS servers, there appears to be a high degree of correlation between the two (figure 26). As such, it is likely that distributors of rogue security software are not using botnets as part of their hosting infrastructure, although some malicious code, such as Downadup, attempts to download rogue security software onto compromised computers.[144] Since botnets can be easily operated from home computers, the use of botnets as rogue security software servers would likely have resulted in a more even distribution of server IP addresses across the entire address space, instead of the concentration that was observed. This correlation of servers indicates that many rogue security software distributors are likely just using commercial Web server hosting providers.

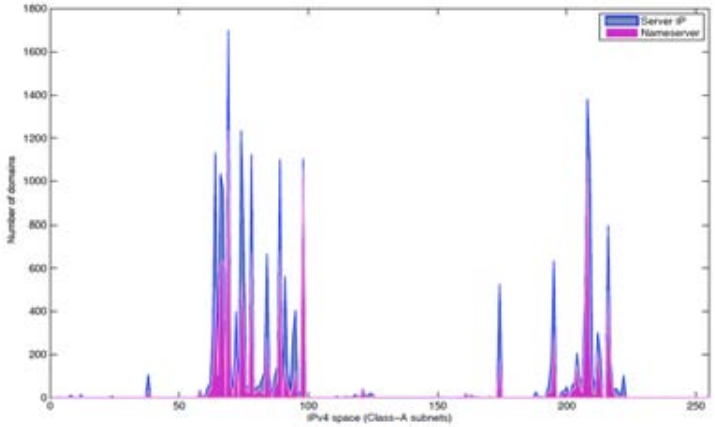


Figure 26. Distribution of rogue security software server IP addresses and their DNS servers

Source: Symantec

143-Each red dot represents a distinct server, while the different gradients on the background underline the areas with highest density of deployed servers.
144-Downadup is associated with rogue security distribution scams such as TrafficConverter.biz, as discussed above.

To determine the relationship between servers (IP addresses) and domain names for rogue security software, a subset of the total analyzed domains has been graphically represented as clusters (figure 27). This subset represents 174 servers that were hosting a total of 30,632 distinct domain names.[145] The relationship between domains (dots in the figure) that were associated with servers is represented by the connecting lines. Clusters are formed when one server has multiple domains associated with it.

Of this observed domain set, those that hosted rogue security software accounted for 15 percent of the total (shown as red in the figure). Nine percent of the total domains were observed to host malware such as malicious executables, scripts, and documents, but may not be hosting rogue security software (shown as orange), and domains that are not malicious accounted for 76 percent of the total observed servers (shown as green).

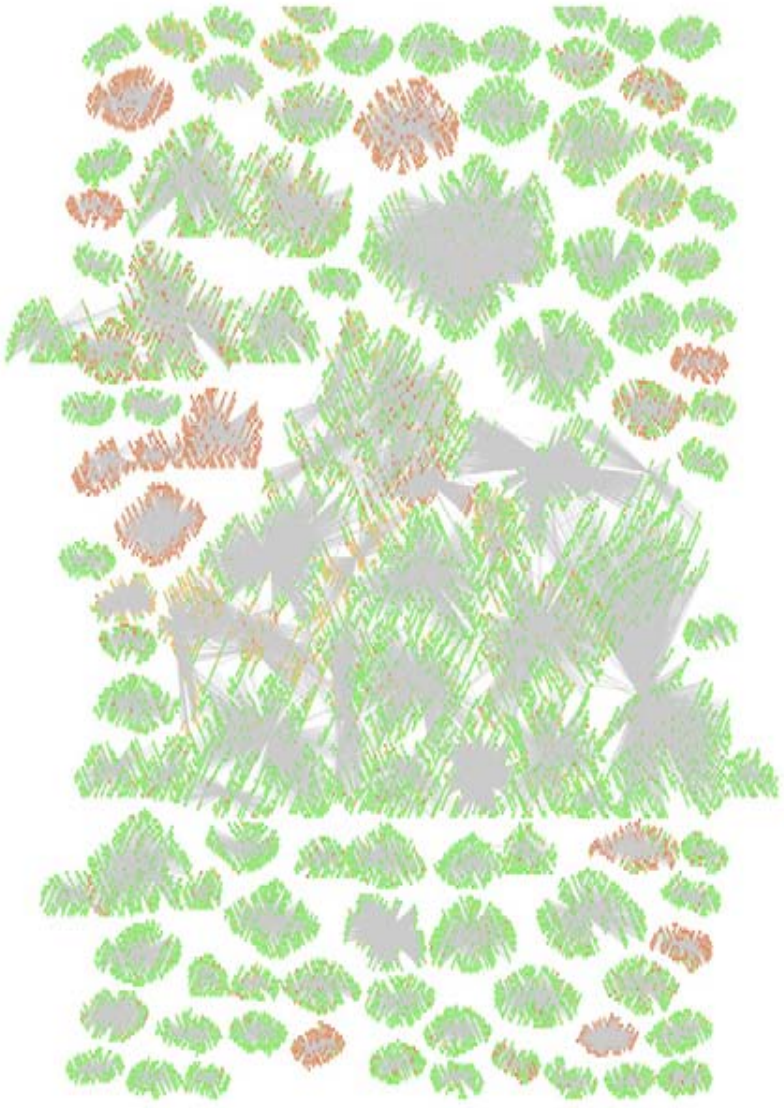


Figure 27. Observed servers and domain name cluster relationships

Source: Symantec

145-For representation purposes, only servers that were observed hosting at least 100 distinct domains are shown in the figure; although the figure does not show all domains, all were used in the analysis.

While most domain names are linked to a single Web server (shown as an isolated cluster), some rogue security software networks span multiple Web servers. Also, some domains were observed as being hosted on more than one server, which may be an attempt to reduce the effectiveness of mitigation measures such as IP blocking or blacklisting servers.

Figure 28, below, highlights the domain clusters that hosted rogue security software. In other words, non-malicious servers (the green in figure 27, above) and servers hosting malicious code (orange, above) have been removed to show just the rogue security software domain clusters.[146] The figure represents 416 servers (IP addresses) hosting 9,964 rogue security software domains (shown in red). Their relationship is shown by a connecting line.

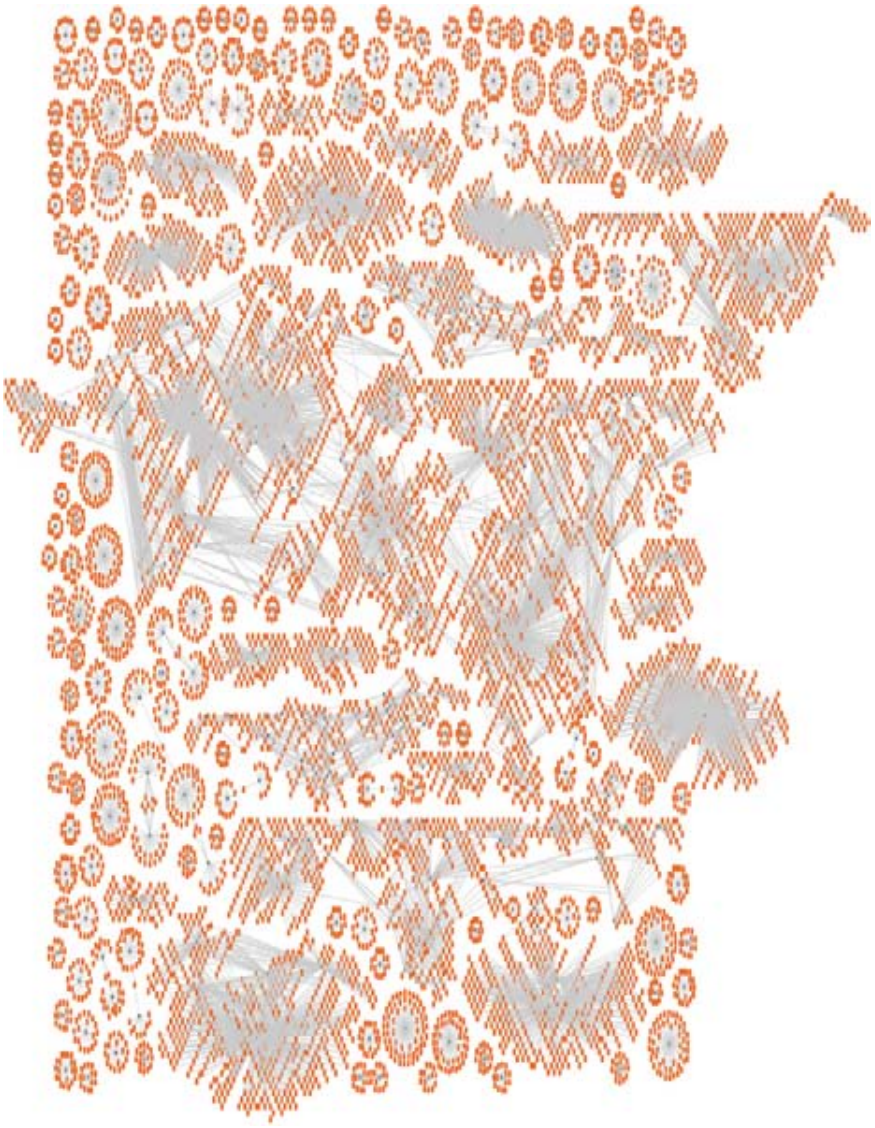


Figure 28. Observed servers and domain name clusters hosting only rogue security software
Source: Symantec

Although a majority of the servers are not malicious, Symantec did observe a number of highly malicious servers. Of the observed rogue security software domains, 26 percent of the total served malicious content of various types (table 4). In

addition, 13 percent of the domains attempted to use browser exploits, one percent attempted to perform drive-by downloads, which seek to infect client computers by forcing them to download and execute malware, without requiring further action (such as a confirmation prompt) by the user, and less than one percent led to the installation of spyware on the user's computer. (Note that a given Web server could belong to several of these categories.)

Rank	Type of Activity	Percentage
1	Infected system with malicious code	26%
2	Attempted to exploit browser vulnerability	13%
3	Attempted a drive-by download	1%
4	Installed spyware	< 1%

Table 4. Percentage of rogue security software domains serving malicious activity, by type
Source: Symantec

Two specific clusters of rogue security software servers from figure 28 were analyzed in detail (figures 29 and 30). Although the two clusters initially appear to be distinct, they have a number of similarities:

- Both clusters use the exact same domain naming scheme (except that one uses “spyware” while the other uses “virus”)
- All of the domains in each cluster use the same registrar and are serviced by the same two ISPs
- All domains within each cluster were registered in a single day and became active (serving software) at nearly the same time
- The email addresses of all domain registrants are in “.ru” domains;
- The servers were on consecutive IP addresses
- The content of these sites was identical, with the exception of one differing image

These similarities strongly suggest that the task of registering, creating, and hosting these rogue security software domains was automated and that the same entity may be responsible for both clusters. Also worth noting is that both clusters are split between two different ISPs, suggesting an attempt to provide some level of redundancy in case a cluster is taken offline by the ISP.

146-As with figure 27, for graphical clarity in figure 28, only rogue security software domain clusters containing at least 10 observed domains are shown.

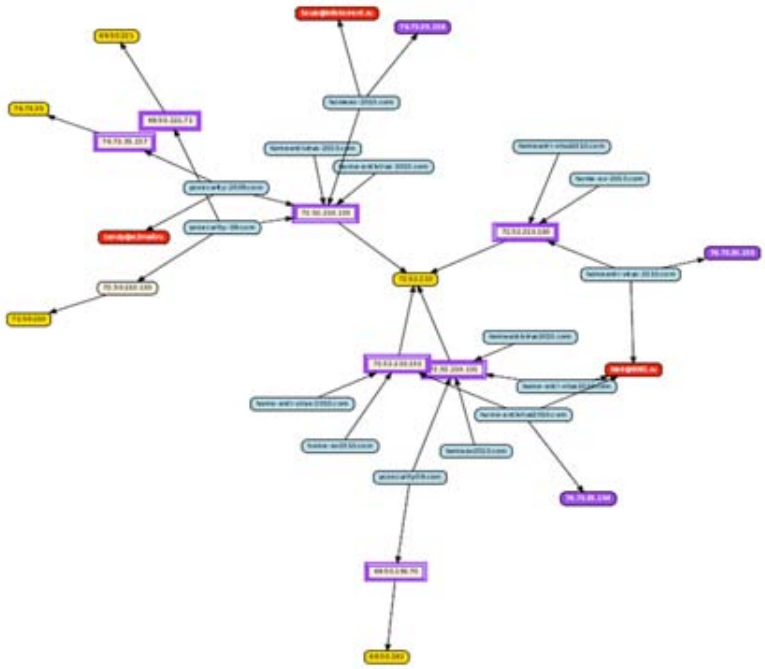


Figure 29. Example cluster 1[147]*Source: Symantec*

147-DNS domains are shown in light blue, DNS servers in purple, the Web server /24 subnets in yellow, and the email address of the registrant in red. Double-edged purple boxes indicate servers with co-located DNS and Web servers.

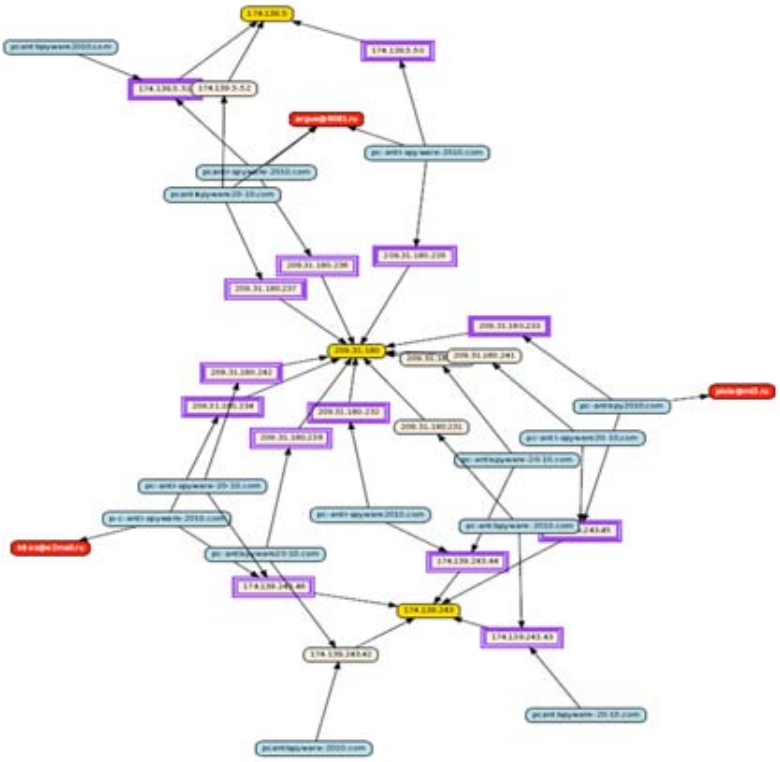


Figure 30. Example cluster 2[148] *Source: Symantec*

A commonly observed characteristic of rogue security software operations was that domain names are registered in large groups within a span of a few days. Symantec observed one site that registered 310 .cn top-level domain names in three days (represented in Figure 31), The 310 domain names (in blue) point to 13 IP addresses residing in five subnets (yellow) and were registered by a number of Web-based email addresses (red) in three days (purple). The prevalent use of popular Web-based email accounts to register these domains is assumed to be because these email services are easily anonymized. These registrants also make use of domain registration services that can either protect registrant privacy or ones that do not verify identities and email addresses.

148-DNS domains are shown in light blue, DNS servers in purple, the Web server /24 subnets in yellow, and the email address of the registrant in red. Double-edged purple boxes indicate servers with co-located DNS and Web servers.

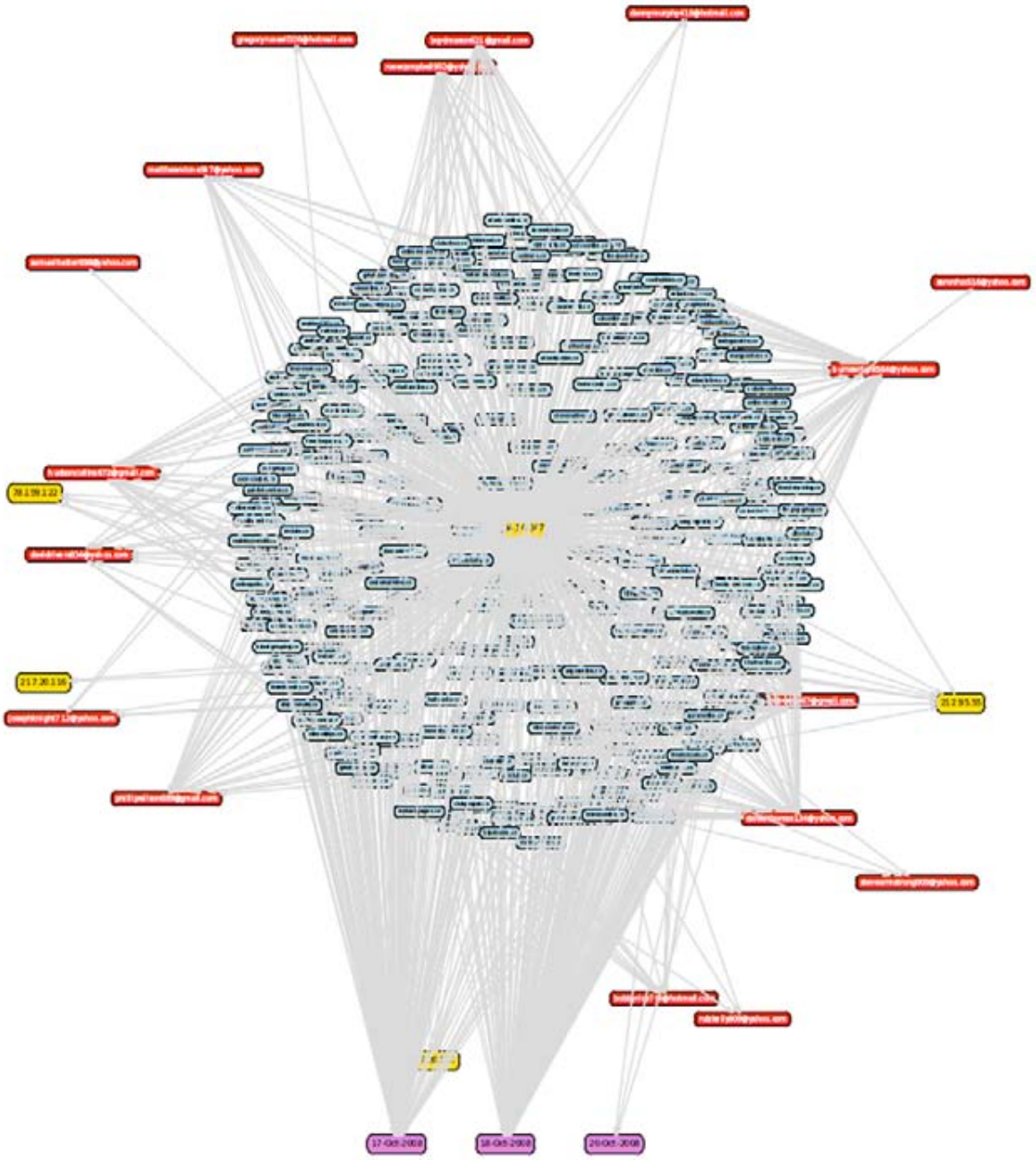


Figure 31. Cluster of 310 domain names registered within three days

Source: Symantec

In another example, 750 .cn top-level domain names (resolving to 135 IP addresses in 14 subnets) were registered on eight specific dates over a span of eight months (figure 32). It should be noted that the .cn top-level domain has no registration restrictions and non-Chinese based operators can register a domain name. For example, of the 750 domains registered in the second example, the majority of the IP addresses of the hosting servers (pointed to by these domains)

were hosted in the United States, Germany, and Belarus. No servers could be identified as being located in China (.cn is the domain designation for China).

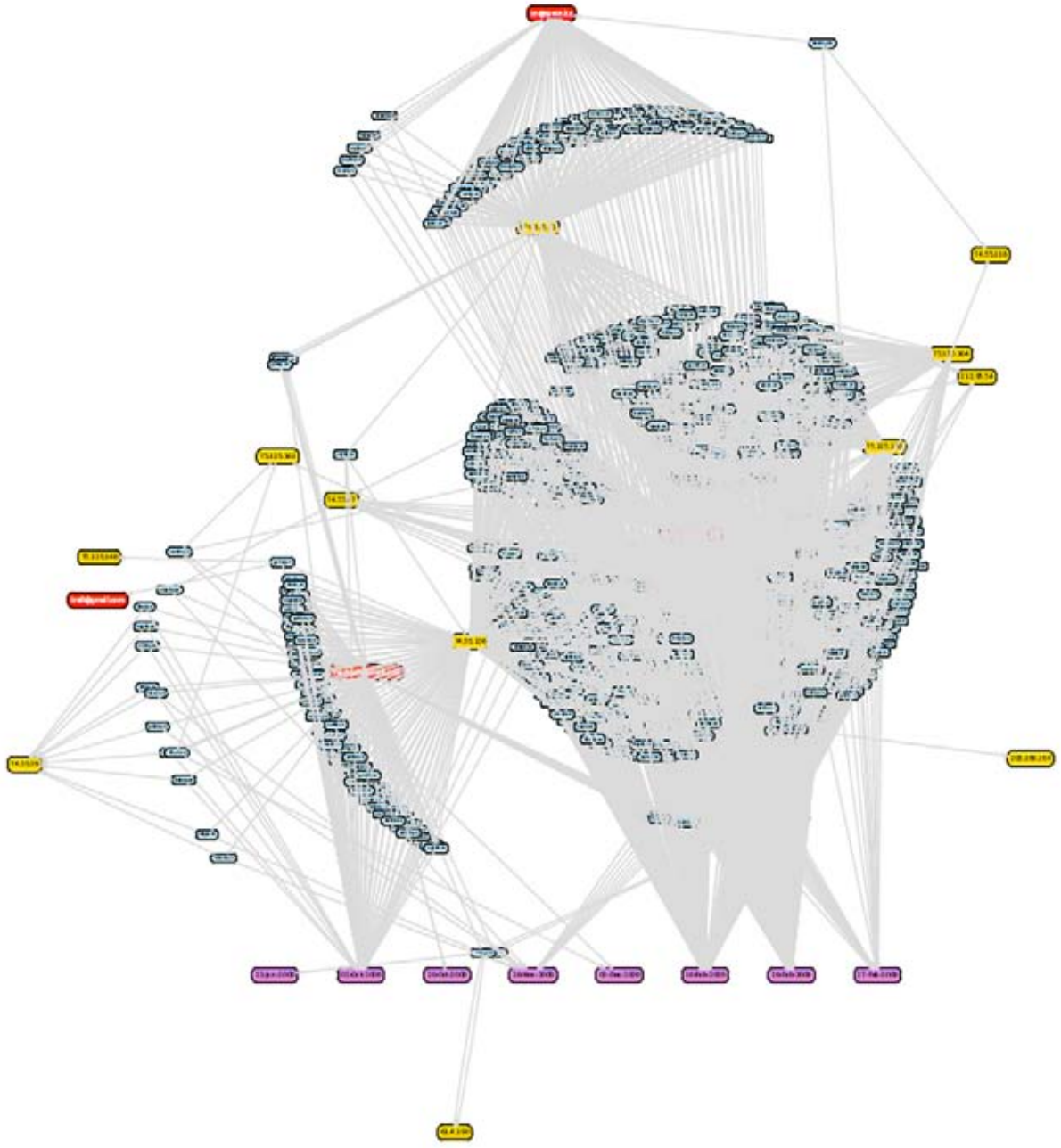


Figure 32. Cluster of 750 domain names

Source: Symantec

Appendix A: Protection and Mitigation

There are a number of general measures that enterprises, administrators, and end users can employ to protect against fraud-related activities such as rogue security software scams.

Enterprise

Administrators should update antivirus definitions regularly and ensure that all desktop, laptop, and server computers are updated with all necessary security patches from their operating system vendor. Also, computers should use the latest protection from spyware and other security risks, such as Norton Internet Security. As compromised computers can be a threat to other systems, Symantec also recommends that enterprises notify their ISPs of any potentially malicious activity, such as bots. Symantec recommends that organizations perform both ingress and egress filtering on all network traffic to ensure that malicious activity and unauthorized communications are not taking place. Organizations should also filter out potentially malicious email attachments to reduce exposure to enterprises and end users.

Organizations should monitor all network-connected computers for signs of malicious activity including bot activity and potential security breaches, ensuring that any infected computers are removed from the network and disinfected as soon as possible. Organizations should employ defense-in-depth strategies, including the deployment of antivirus software and a firewall.[149]

To protect against potential rogue security software scam activity, organizations should educate their end users about these scams. They should keep their employees notified of the latest scams and how to avoid falling victim to them, as well as provide a means to report suspected malicious rogue security software websites. By creating and enforcing policies that identify and restrict applications that can access the network, organizations can minimize the effect of malicious activity, and hence, minimize the effect on day-to-day operations.

Administrators can use a number of measures to protect against the effects of vulnerabilities. They should employ a good asset management system to track what assets are deployed on the network and to determine which ones may be affected by the discovery of new vulnerabilities. Vulnerability management technologies should also be used to detect known vulnerabilities in deployed assets. Administrators should monitor vulnerability mailing lists and security websites to keep abreast of new vulnerabilities in Web applications.

Website maintainers can reduce their exposure to site-specific vulnerabilities by conducting a security audit for common vulnerabilities affecting their sites. Web application code should be audited prior to being released to production systems. When developing Web applications, organizations should investigate the availability and applicability of secure libraries to perform validation of user-supplied input. Secure development practices and threat modeling should also be employed when developing Web-based applications. Web-application firewalls may also detect and prevent exploitation of Web-based vulnerabilities on production sites.

To protect against successful exploitation of Web browser vulnerabilities, Symantec advises users and administrators to upgrade all browsers to the latest, patched versions. Symantec recommends that organizations educate users to be extremely cautious about visiting unknown or untrusted websites and viewing or following links in unsolicited emails. Administrators should also deploy Web proxies in order to block potentially malicious script code. While attacks are likely

to originate from websites that are trusted as well as those that are not, Web browser security features can help reduce exposure to browser plug-in exploits, as can whitelisting. Specifically, administrators and end users should actively maintain a whitelist of trusted websites, and should disable individual plug-ins and scripting capabilities for all other sites. This will not prevent exploitation attempts from whitelisted sites, but may aid in preventing exploits from all other sites. Only plug-ins that have been audited and certified should be installed on workstations throughout the organization.

Symantec recommends that certain best security practices always be followed to protect against malicious code infection. Administrators should keep patch levels up to date, especially on computers that host public services and applications-such as HTTP, FTP, SMTP, and DNS servers-and that are accessible through a firewall or placed in a DMZ. Email servers should be configured to only allow file attachment types that are required for business needs and to block email that appears to come from within the company, but that actually originates from external sources. Additionally, Symantec recommends that ingress and egress filtering be put in place on perimeter devices to prevent unwanted activity.

Administrators should ensure that all email attachments are scanned at the gateway to limit the propagation of email-borne threats. Additionally, all executable files originating from external sources, such as email attachments or downloaded from websites should be treated as suspicious. All executable files should be checked by antivirus scanners using the most current definitions.

Enterprises should take measures to prevent P2P clients from being installed on any computers on the network. They should also block any ports used by these applications at the network boundary. End users who download files from P2P networks should scan all such files with a regularly updated antivirus product.

End user

In addition to the protection and mitigation measures recommended for enterprises, end users could also take more security precautions when conducting Internet activities to ensure that their computer will not be compromised and their information will not be compromised and used for identity fraud. Users should also avoid following links from emails, as these may be links to spoofed or malicious websites. Instead, they should manually type in the URL of the website. Symantec also advises that users never view, open, or execute any email attachment unless the attachment is expected and comes from a known and trusted source, and unless the purpose of the attachment is known. Also, users should be suspicious by an email that is not directly addressed to their email address.

Users should be cautious of pop-up displays and banner advertisements that mimic legitimate displays or try to promote security products. Also, users should not accept or open suspicious error displays from within their Web browser as these are often methods rogue security software scams use to lure users into downloading and installing their fake product. In addition, users should only purchase security software from reputable and trusted sources and only download applications directly from the vendor's website or legitimate partners.

Individual Web users should also exercise caution when browsing the Web. Since malicious attacks can result in hijacking of open sessions, users should make sure to log out of websites when their session is complete. Users should also be wary of visiting untrusted or unfamiliar sites. Scripting and active content can also be disabled when casually browsing the Web.

Users should regularly review credit card and other financial information as this can provide information on any irregular activities. For further information, the Internet Fraud Complaint Center (IFCC) has also released a set of guidelines on how to avoid Internet-related scams.[149] Additionally, network administrators can review Web proxy logs to determine if any users have visited known phishing sites.

149-Defense-in-depth emphasizes multiple, overlapping, and mutually supportive defensive systems to guard against single-point failures in any specific technology or protection methodology. Defense-in-depth should include the deployment of antivirus, firewalls, and intrusion detection systems, among other security measures.

Appendix B: Methodologies

Top reported rogue security software

This metric will determine the most prevalent rogue security software programs based on the number of consumer reports for each rogue security software program observed during the reporting period. The top five applications will be discussed, including analysis of their affects and features. This will provide insight into which rogue security software scams have been the most successful and may indicate prevailing attributes that will continue to be employed or enhanced in future scams.

Top rogue security software by region

Using the top 50 rogue security software programs, as determined by the number of consumer reports per program, this metric will discuss the geographic location of rogue security software reports. The percentage of reports in each of the regions (NAM, LAM, EMEA, and APJ) will be examined to determine whether or not geographic boundaries affect the distribution of software and to provide insight about whether or not these scams are tailored for specific regions or languages.

Rogue security software distribution methods

Using the top 50 rogue security software programs, as determined by the number of consumer reports per program, this metric will discuss how rogue security software gets onto a user's system. Information about each of the top 50 programs will be analyzed to determine which distribution methods each program uses. The resulting data will be combined with the number of consumer reports to determine the prevalence of each distribution method during the reporting period. Distribution methods will include intentional downloads and unintentional downloads.

Rogue security software advertising methods

Using the top 50 rogue security software programs, as determined by the number of consumer reports per program, this metric will discuss how attackers lure users into downloading the rogue security software. Information about each of the top 50 programs will be analyzed to determine which advertising methods each program uses. The resulting data will be combined with the number of consumer reports to determine the prevalence of each advertising method during the reporting period. Advertising methods will include dedicated websites and advertisements on websites (either legitimate or illegitimate) such as social networking sites or blogs.

Rogue security software servers

The data collection and analysis for this section occurred over a period of two months in July and August, 2009. For the servers, data was collected and analyzed on "network observables" including IP addresses, DNS domain names, other DNS entries pointing to the same IP, geolocation information on IP addresses, server identification string and version number, ISP identity, DNS Registrar, DNS registrant information, uptime, and DNS-to-IP resolution changes and the speed with which such changes occurred. In total, 6,500 DNS entries pointing to 4,305 distinct IP addresses hosting rogue security software servers were analyzed.

Using a novel attack attribution method based on a multi-criteria fusion algorithm developed by Symantec and six other academic and industrial external partners as part of a research project, known as the Worldwide Observatory of Malicious Behaviors and Attack Threats (WOMBAT),[150] rogue security software domains were automatically grouped together based upon common elements likely due to the same root cause.[151] This method was used to identify patterns of various types of relationships among rogue security domains and the manner in which they operate, resulting in the creation of domain clusters.

150-WOMBAT is a three-year European Commission-funded project, which aims at providing new means to understand the existing and emerging threats that are targeting the Internet economy and its users. See http://www.wombat-project.eu/

151-For further details on this attack attribution method, please see "Addressing the attack attribution problem using knowledge discovery and multi-criteria fuzzy decision-making", http://www.eurecom.fr/util/publidownload.en.htm?id=2806

Credits

Marc Fossi
Executive Editor
Manager, Development
Security Technology and Response

Dean Turner
Director,
Global Intelligence Network
Security Technology and Response

Eric Johnson
Editor
Security Technology and Response

Trevor Mack
Editor
Security Technology and Response

Téo Adams
Threat Analyst
Security Technology and Response

Joseph Blackbird
Threat Analyst
Security Technology and Response

Mo King Low
Threat Analyst
Security Technology and Response

David McKinney
Threat Analyst
Security Technology and Response

Marc Dacier
Senior Director
Symantec Research Labs Europe

Angelos D. Keromytis
Senior Principal Software Engineer
Symantec Research Labs Europe

Corrado Leita
Senior Research Engineer
Symantec Research Labs Europe

Marco Cova
Ph.D. candidate
University of California Santa Barbara

Jon Orbeton
Independent analyst

Olivier Thonnard
Royal Military Academy, Belgium

About Symantec

Symantec is a global leader in providing security, storage and systems management solutions to help consumers and organizations secure and manage their information-driven world. Our software and services protect against more risks at more points, more completely and efficiently, enabling confidence wherever information is used or stored.

For specific country offices and contact numbers, please visit our website.

Symantec World Headquarters
350 Ellis St.
Mountain View, CA 94043 USA
+1 (650) 527 8000
1 (800) 721 3934
www.symantec.com

10/2009 20100385

4 Conclusion

This deliverable aimed at giving an overview of existing techniques for root cause analysis, and provided some preliminary results with respect to the root cause analysis work performed in the project so far.

We listed 6 papers that have been published at well-known, peer-reviewed workshops and conferences. Furthermore, we included the Symantec threats intelligence report that was partially-based on the work performed in WP5.

Bibliography

[1] J. Armin, J. McQuaid, and M. Jonkman. Atrivo - Cyber Crime USA. `http://hostexploit.com/downloads/Atrivowhitepaper082808ac.pdf`, 2008.

[2] D. Barbará, J. Couto, S. Jajodia, L. Popyack, and N. Wu. Adam: A testbed for exploring the use of data mining in intrusion detection. In *ACM SIGMOD Record, 30(4)*, pages 15–24, 2001.

[3] D. Barbará and S. J. (Eds), editors. *Applications of Data Mining in Computer Security*, volume 6 of *Advances in Information Security*. Springer, 2002.

[4] D. Barbará and S. Jajodia, editors. *Applications of Data Mining in Computer Security*, volume 6 of *Advances in Information Security*, chapter Data Mining For Intrusion Detection - A Critical Review (K. Julisch). Springer, 2002.

[5] T. Bass. Intrusion detection systems and multisensor data fusion. *Communications of the ACM*, 43(4):99–105, 2000.

[6] A. Belenky and N. Ansari. On deterministic packet marking. *Comput. Netw.*, 51(10):2677–2700, 2007.

[7] G. Beliakov, A. Pradera, and T. Calvo. *Aggregation Functions: A Guide for Practitioners*. Springer, Berlin, New York, 2007.

[8] D. Bizeul. Russian Business Network Study. `http://www.bizeul.org/files/RBN_study.pdf`, 2007.

[9] S. T. Brugger. Data Mining Methods for Network Intrusion Detection. In *dissertation proposal, submitted to ACM Computer Surveys (under revision), 2009*, 2009.

[10] H. Burch and B. Cheswick. Tracing anonymous packets to their approximate source. In *LISA'00: Proceedings of the 14th USENIX conference on System administration*, pages 319–328, Berkeley, CA, USA, 2000. USENIX Association.

[11] V. Chatzigiannakis, G. Androulidakis, K. Pelechrinis, S. Papavassiliou, and V. Maglaris. Data fusion algorithms for network anomaly detection: classification and evaluation. In *IEEE International Conference on Networking and Services, ICNS'07, Athens, Greece, June 2007*, June 2007.

[12] H. Chen, W. Chung, Y. Qin, M. Chau, J. J. Xu, G. Wang, R. Zheng, and H. Atabakhsh. Crime data mining: an overview and case studies. In *Proceedings of the 2003 annual national conference on Digital government research*, pages 1–5. Digital Government Society of North America, 2003.

[13] Z. Chen, C. Ji, and P. Barford. Spatial-temporal characteristics of internet malicious sources. In *Proceedings of INFOCOM*, 2008.

[14] S. I. Cyber-Threat Analytics (Cyber-TA). www.cyber-ta.org, [sep 2009].

[15] D. Dagon, G. Gu, C. Lee, and W. Lee. A Taxonomy of Botnet Structures. In *Annual Computer Security Applications Conference (ACSAC)*, 2007.

[16] H. Debar and A. Wespi. Aggregation and correlation of intrusion-detection alerts. In *RAID '00: Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection*, pages 85–103, London, UK, 2001. Springer-Verlag.

[17] dn1nj4. The Shadowserver Foundation: RBN "Rizing". `http://www.shadowserver.org/wiki/uploads/Information/RBN_Rizing.pdf`, 2008.

[18] Ertoz, Eilertson, Lazarevic, Tan, Kumar, Srivastava, and Dokas. MINDS - Minnesota Intrusion Detection System. In *Next Generation Data Mining, MIT Press, 2004*, 2004.

[19] F-Secure Blog. Mass SQL Injcetion. `http://www.f-secure.com/weblog/archives/00001427.html`, 2008.

[20] J. Figueira, S. Greco, and M. E. Ehrgott. *Multiple Criteria Decision Analysis:State of the Art Surveys*. Springer, International Series in Operations Research & Management Science , Vol. 78, 2005.

[21] V. Hanna. Spamhaus: Cybercrime's U.S. Hosts. `http://www.spamhaus.org/news.lasso?article=636`, 2008.

[22] T. Holz, C. Gorecki, F. Freiling, and K. Rieck. Detection and Mitigation of Fast-Flux Service Networks. In *Network and Distributed System Security Symposium (NDSS)*, 2008.

[23] Honeynet Project. Know Your Enemy: Fast-Flux Service Networks. http://www.honeynet.org/papers/ff/fast-flux.html, 2007.

[24] K. Julisch and M. Dacier. Mining intrusion detection alarms for actionable knowledge. In *Proceedings of the 8th ACM International Conference on Knowledge Discovery and Data Mining*, 2002.

[25] B. Krebs. Taking on the Russian Business Network. http://voices.washingtonpost.com/securityfix/2007/10/taking_on_the_russian_business.html, 2007.

[26] B. Krebs. Report Slams U.S. Host as Major Source of Badware. http://voices.washingtonpost.com/securityfix/2008/08/report_slams_us_host_as_major.html, 2008.

[27] W. Lee, S. Stolfo, and K. Mok. A data mining framework for building intrusion detection models. In *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, pages 120–132, 1999.

[28] W. Lee and S. J. Stolfo. Combining knowledge discovery and knowledge engineering to build IDSs. In *RAID '99: Proceedings of the 3th International Symposium on Recent Advances in Intrusion Detection*, 1999.

[29] McCue. *Data Mining and Predictive Analysis: Intelligence Gathering and Crime Analysis*. Butterworth-Heinemann (Elsevier), May 2007, 2007.

[30] J. Mena. *Investigative Data Mining for Security and Criminal Detection*. Butterworth-Heinemann (Elsevier,) Avril 2003, 2003.

[31] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson. Characteristics of Internet Background Radiation. In *Proceedings of the 4th ACM SIGCOMM conference on the Internet Measurement*, 2004.

[32] V.-H. Pham and M. Dacier. Honeypot traces forensics : the observation view point matters. In *NSS 2009, 3rd International Conference on Network and System Security, October 19-21, 2009, Gold Coast, Australia*, Dec 2009.

[33] V.-H. Pham, M. Dacier, G. Urvoy Keller, and T. En Najjary. The quest for multi-headed worms. In *DIMVA 2008, 5th Conference on Detection of Intrusions and Malware & Vulnerability Assessment, July 10-11th, 2008, Paris, France*, Jul 2008.

[34] J. P.Scott. *Social Network Analysis: A Handbook*. Sage Publications Ltd; 2nd edition (March 25, 2000), 2000.

[35] S. Savage, S. Savage, D. Wetherall, D. Wetherall, A. Karlin, A. Karlin, T. Anderson, and T. Anderson. Practical network support for ip traceback. In *In Proceedings of the 2000 ACM SIGCOMM Conference*, pages 295–306, 2000.

[36] C. Seifert. Capture-HPC - Honeypot Client. `https://projects.honeynet.org/capture-hpc`, 2008.

[37] G. Shafer. *A mathematical theory of evidence*. Princeton university press, 1976.

[38] S.Jajodia, P.Liu, V.Swarup, and C.Wang, editors. *Cyber Situational Awareness: Issues and Research*, volume 46 of *Advances in Information Security*. Springer, Nov 2009.

[39] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, B. Schwartz, S. T. Kent, and W. T. Strayer. Single-packet ip traceback. *IEEE/ACM Trans. Netw.*, 10(6):721–734, 2002.

[40] D. X. Song and A. Perrig. Advanced and authenticated marking schemes for IP traceback. In *Proceedings IEEE Infocomm 2001*, 2001.

[41] S.Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Cambridge University Press; 1 edition (November 25, 1994), 1994.

[42] O. Thonnard and M. Dacier. A framework for attack patterns' discovery in honeynet data. *DFRWS 2008, 8th Digital Forensics Research Conference, August 11- 13, 2008, Baltimore, USA*, 2008.

[43] O. Thonnard and M. Dacier. Actionable knowledge discovery for threats intelligence support using a multi-dimensional data mining methodology. In *ICDM'08, 8th IEEE International Conference on Data Mining series, December 15-19, 2008, Pisa, Italy*, Dec 2008.

[44] O. Thonnard, W. Mees, and M. Dacier. Addressing the attack attribution problem using knowledge discovery and multi-criteria fuzzy decision-making. In *KDD'09, 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Workshop on CyberSecurity and Intelligence Informatics, June 28th - July 1st, 2009, Paris, France*, Dec 2009.

[45] X. Wang, X. Wang, D. S. Reeves, D. S. Reeves, S. F. Wu, S. F. Wu, J. Yuill, and J. Yuill. Sleepy watermark tracing: An active network-based intrusion response framework. In *in Proc. of the 16th International Information Security Conference*, pages 369–384, 2001.

[46] C. Westphal. *Data Mining for Intelligence, Fraud & Criminal Detection: Advanced Analytics & Information Sharing Technologies*. CRC Press, 1st edition (December 22, 2008), 2008.

[47] K. J. Wheaton. Top 5 intelligence analysis methods, http://sourcesandmethods.blogspot.com, [sep 2009].

[48] D. Wheeler and G. Larsen. Techniques for Cyber Attack Attribution. *Institute for Defense Analyses, Oct 2003*, 2008.

[49] R. R. Yager. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Trans. Syst. Man Cybern.*, 18(1):183–190, 1988.

[50] Yegneswaran, Barford, and Johannes. Internet intrusions: global characteristics and prevalence. In *SIGMETRICS*, pages 138–147, 2003.

[51] V. Yegneswaran, P. Barford, and V. Paxson. Using honeynets for internet situational awareness. In *Fourth ACM Sigcomm Workshop on Hot Topics in Networking (Hotnets IV)*, 2005.

[52] K. P. Yoon and C.-L. Hwang. *Multiple Attribute Decision Making: An Introduction*. SAGE Publications, Quantitative Applications in the Social Sciences, 1995.

[53] Y. C. Z. Li, A. Goyal and V. Paxson. Automating analysis of large-scale botnet probing events. In *Proc. of ASIACCS*, March 2009.