Proceedings of the First Workshop on

Building Analysis Datasets and Gathering Experience Returns for Security

BADGERS 2011

April 10, 2011
Salzburg, Austria

Preface

The BADGERS workshop is intended to encourage the development of large scale security-related data collection and analysis initiatives. It provides an environment to describe already existing real-world, large-scale datasets, and to share with the systems community the return on experiences acquired by analyzing such collected data. Furthermore, novel approaches to collect and study such data sets are presented at the first edition of this workshop. By giving visibility to existing solutions, we expect that the workshop will promote and encourage the better sharing of data and knowledge.

We are happy to report that the first BADGERS workshop received many interesting submissions, spanning three continents, and many aspects of data collection and analysis initiatives. In the end, the program committee accepted 15 papers (including three short papers) out of 21 submissions (71%) for publication and all of the papers received at least three reviews from our program committee. This workshop would never have taken place without the truly excellent program committee and external reviewers and we are grateful for all the hard work they put in.

In our opinion the resulting program is quite interesting and promises to spark lively discussions. In summary, the accepted papers address topics that range from testbeds that can be used to study current attacks, to large scale data collection systems, to legal issues associated with data collection and sharing. All very different papers and presentations, but all focussing on the problem of data collection and analysis initiatives. They were selected for their novelty, and their potential for interesting debate. We sincerely hope you will enjoy the workshop.

April 2011

Engin Kirda and Thorsten Holz
Program Co-Chairs BADGERS 2011

Program Chairs

Engin Kirda (Northeastern University)

Thorsten Holz (Ruhr-University Bochum)

Program Committee

Ali Khayam, National University of Science & Technology (NUST)

Christian Kreibich, International Computer Science Institute

Daisuke Inoue, National Institute of Information and Communications Technology (NICT)

David Dagon, Independent

Davide Balzarotti, Institute Eurecom

Federico Maggi, Politecnico di Milano

Jose Nazario, Arbor Networks

Julio Canto, Hispasec

Marc Dacier, Symantec Research

Mihai Christodorescu, IBM Research

Paolo Milani Comparetti, Vienna University of Technology

Piotr Kijewski, CERT Polska/NASK

Sotiris Ioannidis, FORTH

William Robertson, UC Berkeley

Publication Chair

Federico Maggi, Politecnico di Milano

Table of Contents

Study on Information Security and e-Trust in Spanish households

Pablo Perez San-José
Information Security Observatory
INTECO (National Institute of Communication Technologies)
Av. José Aguado 41 - E-24001 León (Spain)
pablo.perez@inteco.es

Susana de la Fuente Rodriguez
Information Security Observatory
INTECO (National Institute of Communication Technologies)
Av. José Aguado 41 - E-24001 León (Spain)
susana.fuente@inteco.es

ABSTRACT

The study on Information Security and e-Trust in Spanish households has been conducted by INTECO – The National Institute of Communication Technologies (www.inteco.es) – through the Information Security Observatory (http://observatorio.inteco.es) It is a study on the incidence and trust of users in the Internet by means of measuring the frequency of the episodes of individual risk in a wide sample of users that are monitored online on a monthly basis, combining quantitative data of incidences (monthly scans of home computers) and qualitative perception data (quarterly surveys). The study is supplied with data from more than 3,000 households with Internet connection, spread across the whole country.

For the first time, it allows an evolutionary comparison of the situation regarding security, trust and the level of security incidents in the households of Spanish Internet users. In addition, it shows the habits that affect security on the Internet: security equipment at the households, the measures users take before and after incidents and the perception regarding security on the Internet in Spanish households. It also shows the increasing need of users to force Public Administrations "to make the Internet a safe place".

Categories and Subject Descriptors

A.1 [**General Literature**]: Introductory and survey

General Terms

Measurement, Documentation, Security, Human Factors.

Keywords

Study, e-Trust, Spain, Indicators,

1. OBJECTIVES

The general objective of this Study is to assess the security, trust and security level incidents in Spanish Internet user households. All this with the aim of promoting the knowledge and monitoring the main indicators and public policies related to Information Security and e-Trust.

This assessment is carried out with a temporal perspective, with the aim of supporting and generating proposals with the aim of the Government making decisions to reduce the possible limitations and obstacles related to the security and trust of the users of the Net that affect the development of the Information Society in Spain.

1.1 Security Habits

- Know the intentions for adopting the progresses regarding security in Internet in the near future.
- Study the general demands of Internet users, households and citizens, for the better development of a secure and trustworthy Information Society.
- To find out how habits of Internet use are developing and their possible influence on security risks.

1.2 Security and Vulnerability Incidents

- Determine the level of the general impact of the risks of malware: computer viruses, Trojan horses, worms, spyware, etc.
- Catalogue the most frequent types of malware, their capacity for spreading and their seriousness.
- Itemize the differential exposure to risks per age group, level of training, experience as an Internet user, income, and other sociologically relevant variables.

1.3 Perception of Security

- Obtain the general perception of the risk in view of computer viruses, threats to privacy and the security of payments, amongst others, as well as their evolution in time.
- Determine the level of electronic trust from the users' point of view.

1.4 System of Indicators

- To establish a complete and consistent system of indicators to enable monitoring of the evolution of security on the Internet in access from households.
- To understand the situation regarding Incidents and e-Trust of various groups and social strata, as well as the temporal trends of this situation.

2. METHODOLOGICAL DESIGN

With a view to reach the objectives explained, strict incidence measures have been combined with subjective ones of security perception and trust in the Net.

The aim is to establish a solid base on which information about the changes in the level of security and trust in Spanish homes can be gathered. This requires obtaining solid data about a sample that will provide longitudinal information, i.e., it is necessary to collect data about the homes and users at different moments of time. The methodology that best fulfils these criteria is the Dedicated online panel.

INTECO has developed an innovative methodology for its household Panel. It is made up of more than 3,000 households from around the country that have an Internet connection from which information is extracted from two sources:

- On the one hand, the real level of security is analyzed with software that analyses the security incidents in home computers. This computer program, iScan – developed by INTECO –, is given to the panelists for them to install it on their computers. This software monthly scans the panelists' computers, detecting all malware present in them and also gathers data on the operating system and the state of its up-to-dateness. The software sends this information to INTECO, who treats it completely anonymously and as a whole. The panelists are informed that they will not receive any information on their security incidents, even though the incidents may be dangerous for their computers, as the interest for knowing the general situation as reliably as possible prevails over warnings for solving individual problems. The panelists are duly informed of this situation and accept to participate under these conditions.
- On the other hand, the perception and level of trust of domestic users will be analyzed by means of personal surveys. The panelists will answer a quarterly survey on their perception of security and their practices and behaviour on the Net.

This allows analyzing and contrasting two parallel sources of information in the computer security area, which provides a great comparative advantage: it is possible to know the differences existing between the perception of security and the real situation of the panelists.

In addition, this methodology also allows monitoring the following aspects through time:

- The real security level.
- The changes in perspective, opinions and habits, regarding security, undergone by users.

In general, the gathering of information will be carried out according to the following plan:

- Recruitment of the dedicated panel, by means of e-mail invitations.
- Information on the type of collaboration required, incentive system and confidentiality conditions.
- Invitation to scan the panelist's computer that has access to the analysis program by a personalized identifier, in order to control participation and merge the data from the survey.
- Quota control according to the sample design indicated in the "Sample size and distribution".
- Each wave of the Study represents a complete quarterly scanning and survey cycle. It allows for an evolutive comparison of the situation of Security and e-Trust.

2.1 Data sheet

2.1.1 Scope

Spanish Internet users that have frequent access to the Internet from their homes and that are older than 15 years. To delimit the concept of user with more precision, we limit ourselves to users that connect to the Internet at least once a month from their homes.

2.1.2 Sample size and distribution

A representative sample of more than 3,000 Internet users (3,538 in the 14th wave of the Study, covering the 3rd quarter 2010), with a stable participation in the Panel has been extracted. This participation has been considered valid only in the cases in which the panelist had correctly completed the quarterly survey and correctly carried out a scan of her/his computer in, at least, two of the three months of each wave.

The sample has been fixed according to a multistage model:

- Stratification by regions, in order to guarantee a minimum set of subjects in each of these entities.
- Sampling by quotas of home size, age, gender, work activity and size of habitat.

Table 1. Sample distribution by sociodemographic categories (%)

Concept	Sample obtained (14th wave, jul-sep'2010)	Theoretical sample
Activity		
Employed	53.8	71.7
Unemployed	18.2	4.6
Studying	17.3	16.1
Retired	5.9	3.0
Other non-workers	4.7	4.6
Size of the household		
1	7.9	3.2
2	25.3	15.4
3	30.9	28.7
4 and more	35.9	52.7
Gender		
Male	52.5	53.7
Female	47.5	46.3
Age		
Up to 24	21.8	n.a.
25-34	28.2	n.a.
35-44	24.2	n.a.
45-54	15.9	n.a.
55 and more	9.8	n.a.

Although the differences between the obtained sample and the theoretical one have been small, the sample has been adjusted to the scope, based on the data of the population by region, for the previously described scope and the quota variables, in order to reach a more perfect adjustment. In Table 1, we can see the sample distribution, according to demographical variables used to establish the said quotas.

Given the fact that the final data of the survey has been adjusted to the same scope of the study, they are perfectly homogeneous when it comes to the geographical distribution, gender, size of the household and other relevant sociodemographic variables, that is to say, they do not show variations in those aspects for the purposes of the analysis.

2.1.3 Sampling error

In accordance with the criteria of simple random sampling for dichotomic variables in which p=q=0.5 and for a confidence level of 95.5%, the following calculations of the sampling error are established.

Sampling error ±1.68%.

2.2 Consistency and robustness of the sample

The consistency of the sample, in terms of a possible self-selection bias because of accepting panelists scanning their computer, has been analyzed in detail. It has been concluded that the sample does not show significant bias in this aspect.

In order to check the robustness of the analysis, the results of the scans and surveys are monitored throughout the life of the panel.

- The results regarding the habits, opinions and attitudes and the Security indicators panel show a considerable consistency, which corresponds to variables, which change rather slowly under stable conditions.
- The data of the scanning, expressed as a percentage of malware detections in the months of the life of the panel since January 2007, also show that the variations of the sample are included in the normal variation, established by the sampling error and by the logical and normal development of security habits of Spanish users.

The obtained results can be considered suitable and it is possible to establish them as a basis for a future analysis of temporal series, which will allow to measure the past development and predict possible future situations.

The sample is, therefore, exempt from bias and structural problems. The variations produced in the sample over time are the result of the panel’s dynamism, which reflects how the incidents detected in the users are evolving.

2.3 Technical design of the system of statistical indicators

Every analysis and all the information about security incidents and e-trust, shown in the final report can be simplified in the calculation of a series of indicators that systematically customize the information of the Study in a segmented way.

The system consists of six indicators and includes, for example, usage habits, such as the equipment in security or real malware incidents:

- Security indicator 1 (SI.1) Tools and security measures indicator
- Security indicator 2 (SI.2) Security behaviour and habits indicator
- Security indicator 3 (SI.3) e-trust indicator
- Security indicator 4 (SI.4) Malware incidents indicator
- Security indicator 5 (SI.5) Computers at high risk indicator
- Security indicator 6 (SI.6) Computers with high dissemination potential indicator

2.3.1 Objectives and Advantages

The system of indicators is expected to serve as a means to monitor the evolution and trends of security on the Internet, as well as the trust of households.

The system of indicators, designed by INTECO, has the following benefits:

- It is integral, as it encompasses both usage habits and equipment in security, or the real malware incidents.
- It is synthetic, as it summarizes all relevant aspects of security into a set of six indicators.
- It is sensitive, as it has detected small variations of security and has shown to be relevant to detect risk situations in specific segments of the population.
- It is stable, as it permits to have a general vision of the situation of security of any market, segment or sub-segment, related to the scores, whose reference is always 100 on the scale. Even if the number of questions that form the indicator varies, the system would maintain its stability and historical comparability.
- It is operative, as it permits to easily detect the system’s vulnerabilities and to instigate measures to reduce them.
- It is strategic, as it helps to understand the consequences of the individual situations regarding the lack of protection for the system and it permits to introduce the connection between the Administration’s security policy and users´ individual behaviour.

2.3.2 Status and values of the indicators

The value of the six indicators ranges from 0 to 100 points. It is given in points, except for the indicator SI.4: Malware incidents indicator, which shows the percentage of computers with at least one incident of malware, coming from the data of the scanning, and which is included within the system of Indicators, due to the relevance of the data. That is to say, even though the indicator SI.6, Computers with high dissemination potential indicator, has a value of, for example, 27.3, it does not mean that 27.3% of computers have a high dissemination risk, but that the result of the combined calculations used to obtain the result shows a value of 27.3 points.

They show a combined calculation of different items and parameters that form each indicator.

This system facilitates temporary analysis and comparisons between the different waves.

2.3.3 Structure of the System of Indicators

The six indicators are classified into two groups:

- Indicators related to protection:SI.1 and SI.2
- Indicators related to risk: SI.4, SI.5, and SI.6.

IS.3, which completes the list, presents users´ perception, i.e. the e-trust indicator represents the balance variable, which evaluates the protection against risks.

The first group includes the factors that increase protection, while the second group includes those factors that measure the risks. The system modifies the parameters of the indicators of both groups, in order to keep the perception of users high. In this way, changes in the habits and behavior of Spanish households may be analyzed. The system of the set of indicators is balanced: an increase in the incidents tends to be compensated by more security equipment and prudent habits, in order to restore the balance, marked by high e-Trust (Figure 2).

2.3.3.1 SI.1 Tools and security measures indicator

It measures the equipment and the adoption of security measures.

It is calculated according to certain measures of the available security equipment by comparing the data with an optimal security situation, which is reached with full equipment. The equipment for the calculation of the indicator includes the security measures that are most used: antivirus programs, firewalls, pop-up blockers, deletion of temporary files and cookies, antispam programs, antispyware, passwords (equipment and documents), security updates of the operating system, backups of important files and document encryption. The calculation of the indicator does not only focus on the security of the system, but also includes measures that favor the security of information.

2.3.3.2 SI.2 Security behaviour and habits indicator

Measures the type of behaviour and secure habits during Internet browsing and the use of specific online services, synthesising the points obtained on the following aspects:

- Behaviour when browsing.
- Behaviour in electronic mail.
- Behaviour in the use of social networks, chats and instant messaging.
- Behaviour in online banking and electronic commerce.
- Behaviour in the use of file exchange networks (P2P).

These sections in turn are subdivided into conceptual subgroups for each area.

- Internet browsing: includes behaviours such as clicking on interesting or attractive advertisements although the advertiser is not known; or not analysing, manually or automatically, with the antivirus program every file downloaded from the Internet before opening/executing it.
- Using electronic mail: whether users download and open files attached to electronic mails from strangers or open files they did not request if they seem interesting, or if they analyse all attachments with an antivirus program before opening them, etc.
- Using social networks/chats/instant messaging: whether users reject invitations/messages from users they do not know, if they avoid clicking on invitations from strangers to visit web sites, or if they add strangers contact details to their instant messaging program, etc.
- Use of online banking and electronic commerce: whether users perform online transactions (payments, purchases, transfers, etc.) checking that the connection is secure (https protocol, validity and currency of certificate), etc.
- Use of file exchange networks (P2P): whether, for example, users analyse with their antivirus program all files downloaded from P2P networks, etc.

All these features are recorded for the entire set of users depending on the use they make of the aspects and the importance assigned to each section.

2.3.3.3 SI.3 e-trust indicator

This indicator measures users' subjective perception of security when they use the Internet. It is made up with the scores obtained for the following criteria on perception of security (with respect to the maximum possible score): whether, in general, the Internet is more secure; if users think that security is or is not a limiting factor when taking on new Internet services; their perception on the change in security (measured in number of incidents and seriousness compared to 3 months ago), and degree of agreement with the statement "I consider my computer to be reasonably well protected".

2.3.3.4 SI.4 Malware incidents indicator

This indicator shows the percentage of computers with some malware detected during scanning of household computers.

2.3.3.5 SI.5 Computers at high risk indicator

This indicator reflects the percentage of domestic equipment in which the audit detected at least one high risk malware incident.

Detected malicious codes are catalogued into 4 risk groups (from higher to lower) according to the following distribution:

- High risk: trojans - backdoor programs, bankers, keyloggers, diallers - viruses, worms, exploits and rootkits.
- Medium risk: spyware programs, adware programs, scripts and files detected heuristically
- Low risk: joke programs and intrusion tools.
- Without risk: equipment where malware was not detected.

2.3.3.6 SI.6 Computers with high dissemination potential indicator

This synthetic indicator takes into consideration users' behaviour and habits that, to a greater or lesser extent, could result in a high level of dissemination of malware to other users and to their own system. It includes the following elements:

- Whether the equipment is up to date with respect to operating system updates (data obtained from the program for analysing actual incidents).
- Whether any malware in the worm or script categories was detected on the equipment.
- If users declared that they download any type of file from the Internet.
- If instant messaging services were used.
- If users declared that they share files/software without checking whether they are or are not infected.

- If users declared that they send/forward electronic mails without checking whether they contain any attached malicious file.

3. MAIN RESULTS

The main question mark which has motivated this Study is determining the security level in Spanish Internet households/users and the trust they generally have in the Information Society, and particularly regarding Internet. The way to reach this objective has been twofold: on the one hand assessing the trust and perception of security of the users, and on the other, carrying out an analysis on the real security level incidents in the computers in Spanish Internet user households. The result of contrasting the two variables is a series of user segments and profiles according to information security.

Seen as a whole (socially and technologically), the Internet security system is empirically defined by the following characteristics:

3.1 Security measures and habits

Automatable measures occupy the first places in the table of the declared use of security measures: antivirus programs (92.5%), firewalls (81.3%) and operating system updates (80.7%). These positions are the same as in previous quarters.

Users of social networking sites are becoming increasingly careful with their privacy: 66.2% declare that their profiles can only be seen by friends or contacts.

47.8% of parents state that they have created a limited user account for their children to go online. This information is very positive as it reduces the impact possible dangerous behaviour by the minor may have on the computer.

3.2 Security incidents

The most common security incident in the past three months as declared by Internet users is receiving unwanted e-mails or spam (66.9%). According to the INTECO network of sensors, the real figure rises to 77.4%.

53.6% of the computers analysed with the iScan programme are infected with malware. Trojans, at 38.7%, and adware, at 27.1%, are the types of malicious code most commonly present on users' computers, followed by tools (23.8%).

3.3 Users' reaction to security incidents and their consequences

60.6% of users have made no changes to their Internet browsing habits as a result of an incident they have experienced, compared to 39.4% that did adopt some precautionary measures.

Users state that they are acting to a greater degree on security programmes (55%), followed by changing passwords (45.9%).

Almost two out of every three users resolve security problems independently: 44.6% with no-one's help and 19% with the help of an expert.

3.4 e-Trust in Spanish households

The majority of Spanish Internet users trust the Internet (89.9%) during the third quarter of 2010 and believe that their computer is reasonably well protected (81.5%).

Users continue to show more trust in carrying out banking transactions in a branch (72.9%) than via the Internet (50.8%).

Lastly, 79.3% of panellists would like the Government to be more involved in guaranteeing Internet security.

The analysis carried out highlight that e-trust stands at an average of 76.6 points in a scale from 0 to 100 (Figure 2).

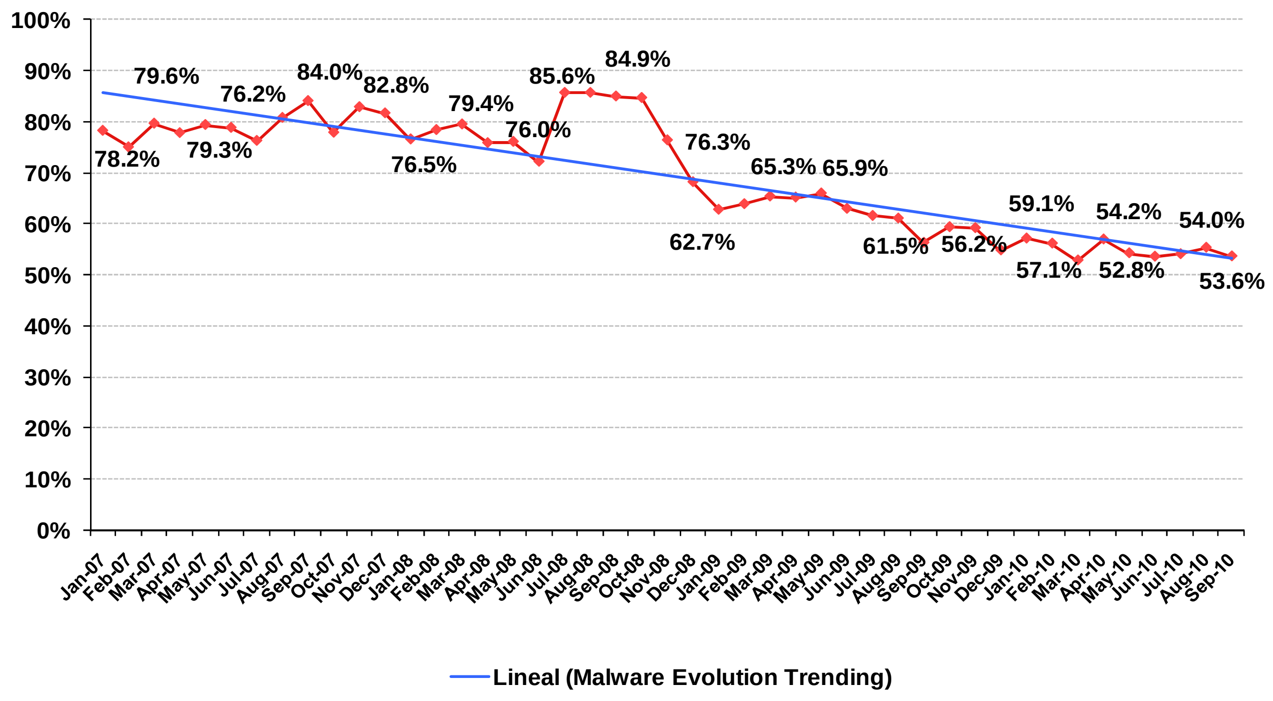


Figure 1. Malware incidence evolution 2007 – 2010 (%)

The conclusion that can be drawn is that high trust is a prerequisite for a rewarding Internet use. The users tend to keep this e-trust above 75 points.

When the sensation of security is broken by an unexpected incident, the user tries to repair the balance increasing their security equipment, increasing their caution or both at the same time.

Generally, these changes help to return to a comfortable level of e-trust. But if things are not as expected (repeated incidents), the need for support from a third party starts to become necessary. This third party for support is the Government.

Specifically, the role the users give the Government in security matters seems to consist of being a last resort. The result must be to guarantee security when the measures within the reach of the user and sensible navigation habits are revealed to be insufficient. In general, this intervention is accepted and demanded by 79.3% of the users.

The overall result of this re-stabilization process in time is that users think that both the amount and seriousness of the incidents suffered in their computers has been reduced in the last year. This reinforces their idea that re-stabilization the suitable strategy.

Given that the level of basic equipment is similar in the majority of users, caution in usage habits has been revealed to be an important additional protection factor. In fact, the results of the computer scanning show how the security habits generally mark the differences in incidents amongst the users with antivirus and updated operating systems.

The Government has a key role: information must be channeled both regarding protection systems and safe practices.

The data indicates that the actual security incidents detected in the scanning seem to have their solution in two relatively independent factors: the real presence of security devices and the preventative and considerate usage habits. Both factors constitute the pillars of the system security and its complementary nature must be strengthened as far as possible: there is no security without the simultaneous presence of both of them.

We must stress a culture of security. It is necessary for the users to be aware of the utility of the solutions such as the antivirus, firewalls, antispam, security updates, etc., but they must also know their limits, the real threats, and the additional recommendations, so that a false sense of security is not created. To increase security it is vital to provide users with greater information with a view to using the new technologies responsibly and safely, with usage habits based on caution and protection.

4. CONCLUSION

Even though both security measures and users' behaviour reduce the number of detected incidents, it is noteworthy the fact that computers' behaviour is the reason for a greater decrease in the amount of malware found on the machines.

Given that the basic level of equipment is similar for most users, prudence in usage habits has become an important additional factor for protection. In fact, computer scan results show how security habits differentiate incidents between users with antivirus software and up-to-date operating systems and those without.

It exists a false feeling of security, i.e. users have the perception that incidents do not reach the level that in fact exists and that they are less and less serious. Moreover, it has been detected that many users neglect their security habits after installing protection measures on their computers, which means that the risks to their systems will increase instead of decreasing, as it should be.

Thus, it is confirmed that the installation of security tools is necessary but not enough. It is also important to take other complementary actions, such as good practices and proper security habits. Security on the Internet is not a question of machines and technology, but of people!

Therefore, the use of security measures and devices - e.g. anti-virus software, firewall, anti-spam software or security updates - must be encouraged and users must be trained in security habits at a technical level, so that technical measures, whether active and passive, can be really efficient.

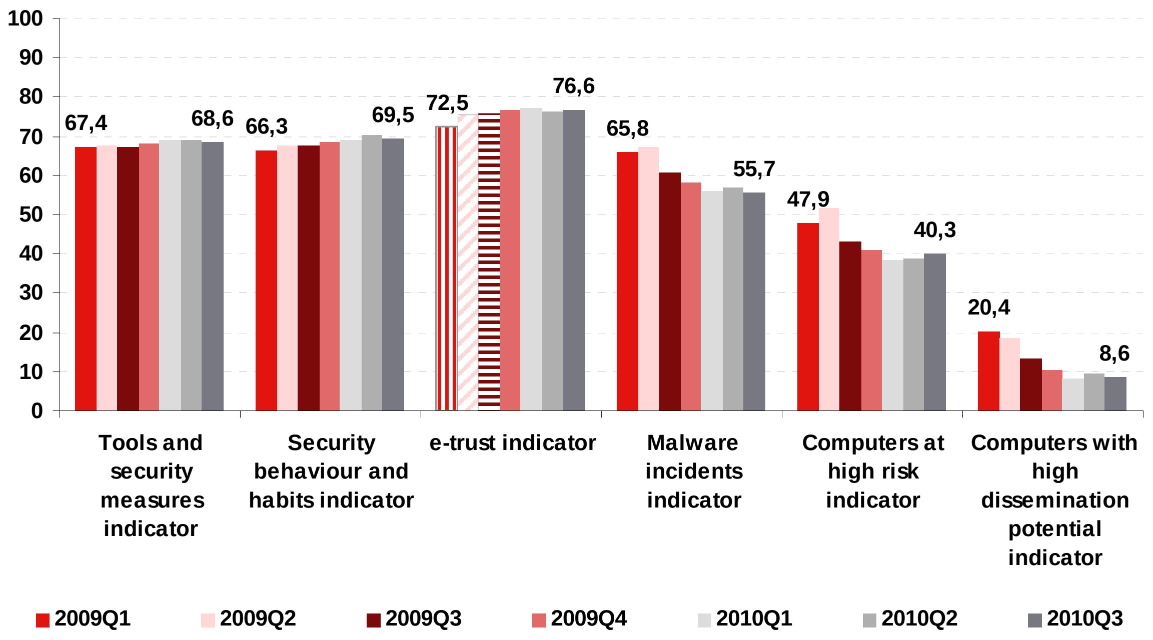


Figure 2. System of Statistical Indicators

Reflections on the Engineering and Operation of a Large-Scale Embedded Device Vulnerability Scanner

Ang Cui
Department of Computer Science
Columbia University
New York NY, 10027, USA
ang@columbia.edu

Salvatore J. Stofo
Department of Computer Science
Columbia University
New York NY, 10027, USA
sal@columbia.edu

ABSTRACT

We present important lessons learned from the engineering and operation of a large-scale embedded device vulnerability scanner infrastructure. Developed and refined over the period of one year, our vulnerability scanner monitored large portions of the Internet and was able to identify over 1.1 million publicly accessible trivially vulnerable embedded devices. The data collected has helped us move beyond vague, anecdotal suspicions of embedded insecurity towards a realistic quantitative understanding of the current threat. In this paper, we describe our experimental methodology and reflect on key technical, organizational and social challenges encountered during our research. We also discuss several key technical design missteps and operational failures and their solutions.

1. INTRODUCTION

Over the past year, the Columbia University Intrusion Detection Systems Lab conducted a quantitative study on the distribution of trivially vulnerable embedded network devices openly accessible over the Internet. Trivially vulnerable devices are those which have well-known default root-level credentials configured on publicly accessible administrative interfaces. At the time of writing, our vulnerability scanner has identified over 1.1 million such vulnerable embedded devices. Our latest data indicates that approximately **1 in 5** publicly accessible embedded device on the Internet is configured with well-known default administrative credentials. In order to accurately establish a quantitative lower bound on the number of vulnerable devices on the Internet, we engineered a parallelized scanning infrastructure capable of monitoring the entire IPv4 space with reasonable speed. Each automated scan takes approximately two weeks and produces a snapshot of all accessible HTTP and Telnet servers on the Internet, along with a list of embedded devices which are confirmed to be trivially vulnerable. Table 1 shows several key metrics of our latest scan results. Figure 1 graphically illustrates the distribution of trivially vulnerable embedded devices across IPv4 space.

Each iteration of the global scan sweeps across over 3.2 billion IP addresses, cataloging every publicly accessible HTTP and Telnet server on the Internet over a period of two weeks. The initial output of each responding server, various meta-data, as well as logs and results of vulnerability tests performed on every candidate embedded device is stored in a cluster of SQL databases. On average, each global scan records over 90 GB of data and involves over 10 billion database transactions. The remainder of this paper is a description and reflection on the successes and failures of our engineering process and will discuss:

Total IPs Scanned	Devices Targeted
3,223,358,720	5,652,358
Vulnerable Devices	Vulnerability Rate
1,134,535	20.07%

Table 1: Scale and Result of the Latest Global Default Credential Scan.

- Major design choices of our scanner infrastructure.
- Technical, organizational and social challenges encountered throughout the course of the study.
- Description and reflection on engineering missteps and failures of our system and protocol.
- Lessons learned from this large-scale study.

The remainder of this paper is organized as follows. Section 2 briefly describes the experimental methodology of our study. For a more detailed description of our study and findings, please see [1]. Section 3 discusses major technical bottlenecks and missteps, the iterative redesign of our scanner infrastructure, as well as organizational challenges and social implications of our study. Section 4 summarizes key lessons learned from our experiences engineering and operating our scanner infrastructure. Lastly, we present our conclusion and acknowledgements in Sections 5 and 6. Appendix C contains logical and physical diagrams of our project's IT infrastructure.

2. EXPERIMENTAL METHODOLOGY

The default credential scan process is straightforward and can be broken down into three sequential phases: **recognizance**, **identification**, and **verification**.

Recognizance: First, *nmap* is used to scan large portions of the Internet for open TCP ports 23 and 80. The results of the scan is stored in a SQL database.

Identification: Next, the device identification process connects to all listening Telnet and HTTP servers to retrieve the initial

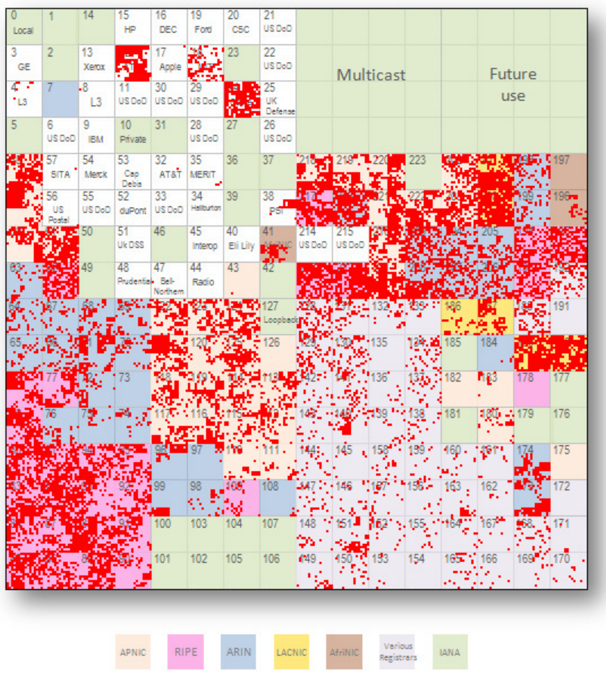


Figure 1: Distribution of Vulnerable Embedded Devices in IPv4 Space.

output of these servers[1]. The server output is stored in a SQL database then matched against a list of signatures to identify the manufacturer and model of the device in question.

Verification: The verification phase uses an automated script to determine whether it is possible to log into devices found in the identification phase. This script uses only well known default root credentials for the specific device model and does not engage in any form of brute force password guessing. We create a unique *device verification profile* for each type of embedded device we monitor. This profile contains all information necessary for the verification script to automatically negotiate the authentication process, using either the device's Telnet or HTTP administrative interface. Figure 2 shows two typical device verification profiles, one for the administrative Telnet interface for Cisco switches and routers, the other for the HTTP administrative interface for Linksys WRT routers using HTTP Basic Authentication. Each device verification profile contains information like the username and password prompt signatures, default credentials as well as authentication success and failure conditions for the particular embedded device type. Once the success or failure of the default credential is verified, the TCP session is terminated and the results are written to an encrypted flash drive for off-line analysis.

The device selection process is manual and iterative. We begin by analyzing data gathered by the recognizance phase of our scanner, which collects the initial output from active Telnet and HTTP servers found by NMAP. We maintain three sets of signatures: non-embedded devices; non-candidate embedded devices; and candidate embedded devices. Signatures of non-embedded devices include those of popular HTTP servers such as Apache and IIS as well as Telnet common authentication prompts of general purpose operating systems. Signatures of non-candidate embedded devices include those that do not ship with a well known default credential[2]. Signatures of candidate embedded devices include string patterns that positively identify the device as one that we are actively monitoring. After the recognizance data is tagged using these three signature sets, we manually inspect and tag the remaining records, creating new signatures and device verification profiles. The process of identifying and incorporating new embedded devices into the scanner is arguably the most labor intensive part of the project. Once the scanner infrastructure was sufficiently developed, its maintenance and operation required little manual intervention. However, as the popularity of different embedded devices rise and fall, constant effort was required to update what is essentially an rule-based expert system.

The technical methodology of our experiment is simple and straight forward. However, the scale of the operation and the sensitivity of the data collected by our scanner posed several major challenges. The next section will discuss the technical, organizational and social challenges we encountered throughout our initial study.

```
root:
    username_prompt: ['sername:']
    username: cisco
    askuser: true
    passstr: ['assword:']
    incorrect: [sername, assword]
    success: ['\#']
    passwords: ['cisco']
    deviceType: cisco
    isActive: 'true'
```

```
root:
    authType: basicAuth
    passwds: ['admin']
    authRealm: WRT54G
    username: ''
    deviceType: linksys-wrt
    loginURL: '/'
    isActive: 'true'
```

Figure 2: Two typical device profiles, stored as YAML files. Left: Cisco telnet. Right: Linksys WRT HTTP.

3. MAJOR CHALLENGES

Using *nmap* to scan the Internet and writing the initial script to verify trivial vulnerability of specific embedded devices is not technically difficult. However, ensuring that the entire scanner infrastructure is scalable, safe (for the scanned networks as well as our host network), secure, resilient and accurate gave rise to several interesting technical challenges. Section 3.1 describes the iterative improvements we made to the scanner infrastructure throughout its development.

Furthermore, conducting an *active* study involving the entire Internet and handling and safe guarding a database of over a million vulnerable embedded devices complicated our efforts beyond mere technical hurdles. Scalability and security concerns were compounded by social and organizational implications of operating within an university environment. Safeguards were instituted within our experimental protocol and internal IT environment to prevent intentional and accidental exfiltration of sensitive data. For example, an isolated and fortified network environment had to be created for this project to ensure that the rest of the university and even other members of our research group could not access our production scanners and databases (See Appendix C). Furthermore,

[1]In case of HTTP, we issue the 'get /' request

[2]For example, the Polycom VSX 3000 video conferencing unit uses the device's serial number as the default password.

an identical development environment was created to allow new project students to contribute to our code base without granting them access to the actual vulnerability database. Section 3.2 discusses the measures taken to ensure that our study is conducted ethically and that sensitive vulnerability data is safe-guarded at every phase of our study.

The public-facing nature of our global vulnerability assessment further complicates our efforts for at least three reasons. We intentionally carried out the scans publicly without employing any stealthy or anonymizing techniques. This is done primarily to ensure that no harm is done by our scanners, and that network operators can easily communicate with our research group. With the help of our university NOC and network security group, we formed an informal response team in order to handle the anticipated responses to our public scanning activities. In retrospect, the volume of responses received regarding our scans was far smaller than first anticipated. This may be an indication that most networks no longer bother with scan detection, or they may ignore scanning since it is so commonplace.

Lastly, the dissemination of our vulnerability data to the proper parties can help network operators secure large numbers of vulnerable embedded devices. However, vulnerability disclosure must be done carefully through the proper channels. While we recognized the need to alert operators of large populations of vulnerable devices, we lacked the experience and means to carry this out properly. Section 3.3 describes how we partnered with Team Cymru to disseminate our research findings in order to reduce the vulnerable device population.

3.1 Technical Challenges and Missteps

The scanner infrastructure underwent three major redesigns over a period of one year. Each redesign iteration involved a partial rewrite of our code base while the network topology of our infrastructure remained unchanged. After the technical goals of our quantitative study were first outlined, a working scanner was hastily written using a collection of bash and python scripts which invoked *nmap* and processed its XML output. This proof of concept scanner was able to quickly identify thousands of trivially vulnerable embedded device on our own University's network.

Encouraged by our initial success, the research group set out to make several obvious improvements to our infrastructure:

3.1.1 Automation:

The initial scanner implementation produced encouraging results. However, its operation required almost constant manual intervention. We implemented all three phases of the scanner workflow as individual command-line tools. Other scripts were written to map out sections of the Internet for the scanner to target. Yet other scripts were written to tabulate and analyze the raw output of the scanner to produce the final vulnerability report. Automation and coordination between each individual component of the scanner was clearly needed. To achieve this, we migrated all standalone tools into Apache/mod_py and created a master job scheduler. Using mod_py allowed us to expose the scanner's API over HTTP. This greatly reduced the complexity of the job scheduler process, which simply made periodic HTTP GET requests to Apache. Leveraging Apache's infrastructure also gave us improved stability, parallelism and logging at an acceptable overhead.

3.1.2 Data Wrangling:

The initial scanner stored all intermediate outputs as plain text files, which quickly ran into scalability issues when we scanned even modest sized networks. To solve this problem, the scanner was modified to store all intermediate outputs in a MySQL database. This improved the performance of the scanner and the analytics code which produced the final vulnerability report and greatly simplified our data generation code which delegated all caching, concurrency and transactional storage logic to the MySQL server.

Furthermore, the use of SQL standardized the schema of the various record types used by the scanner and allowed us to easily export intermediate data to other applications via ODBC. A dedicated MySQL server was created to service all scanner nodes to centralize data storage. A mirrored disk volume was used to store the MySQL database to prevent data loss due to disk failure. A set of *cron* scripts were also put in place to backup the database contents on an hourly basis to a separate disk volume.

3.1.3 Scalability:

The initial scanner design was limited to a single host. The migration of the scanner API over to Apache/mod_py and MySQL allowed us to easily parallelize scanning activity across an array of identical scanner nodes. During the first redesign of the scanner infrastructure, the single server implementation was scaled out to a scanner which utilized 8 scanner nodes and a single centralized database.

The second iteration of the scanner design automated the entire workflow, improved performance and reliability and simplified the codebase by leveraging exiting platforms like MySQL and Apache. However, we quickly noticed several major bottlenecks within the new design.

Most noticeably, the central database became overloaded during periods of high utilization. Since the job scheduler and all scanner nodes depended on the central MySQL server, the entire system frequently thrashed and halted. To solve this problem, the job scheduler was improved to frequently monitor the load of all scanner nodes to prevent over-allocation of scan jobs to nodes which are already heavily utilized. Node utilization polling was done using SNMP once per second. Scanner nodes were removed from the allocatable queue when their system load value exceeded a configurable threshold, or when the maximum allowable running *nmap* instances is exceeded.

Each scanner node is designed to execute up to 32 *nmap* scans and potentially thousands of vulnerability verifications simultaneously[3]. Our code did not restrict the total number of simultaneous MySQL connections created by each node. This quickly caused the MySQL server to reject incoming connections as multiple scanner nodes were brought online. To solve this problem, we incorporated a connection pool controller into our scanner node using the pysqlpool[4] package. Furthermore, we increased the default *max_connections* value in MySQL, which was set to a conservative 151 by default.

Inefficient SQL queries within the analytics and report generation code frequently overloaded the central database, causing the entire

[3]We arrived at this number through experimentation on our own hardware. Various kernel options can be tweaked to improve *nmap* performance. For example, see http://seclists.org/nmap-dev/2008/q1/354

[4]http://code.google.com/p/pysqlpool/

scanner system to halt. We reasoned that delegating read and write operations to separate physical servers would improve performance by isolating the resource drain of the report generation code from the scanner nodes. While this was certainly true, it also led us to our first major technical misstep.

3.1.4 Technical Misstep #1: Database Cluster Fiasco.

The central database became a prominent performance bottleneck after we implemented the first version of the parallelized scanner infrastructure. To improve performance, we experimented with using the Community Edition of MySQL Cluster. In theory, this would give us a single high performance logical MySQL instance which is scaled out to several physical machines. A three node MySQL cluster was deployed, along with MySQLProxy[5], which allowed us to control how read and write queries are dispatched within the database cluster.

This solution delivered the performance boost we needed, but came at the cost of unreliability and ultimately, total data loss. The physical cluster nodes were configured according to available best practice guides and connected to a single low-latency gigabit switch (Cisco 4948). However, we noticed that various database nodes sporadically dropped out of the cluster during periods of high utilization. This eventually led to database inconsistency, long periods of re-synchronization, and after a two week trial-run period, total data loss on several large tables.

We still plan to revisit the MySQL Cluster solution in the future, as a HA MySQL cluster is preferable to our current solution. In the interest of time, we abandoned the clustered approach in favor of replicating the central database sever. Instead of a single scanner infrastructure, we essentially created three identical deployments and divided the IP space to be scanned equally between each scanner cluster.

This deployment also provided an unexpected advantage. Since *nmap* and our vulnerability verification code are both sensitive to network latency and packet loss, we divided the target IP space geographically among the three scanner clusters. This allowed us to customize latency and packet loss related parameters to best suite the part of the World a particular scanner cluster was targeting.

3.1.5 Technical Misstep #2: Data At Rest, Forever.

The second technical fiasco struck shortly after the first. Recognizing the sensitivity of our vulnerability report, which contained the IP, device type, username and password of every vulnerable embedded device discovered by our scanners, we put forth a policy of encrypting any vulnerability-related data leaving our production environment. Communication with Team Cymru, who mediated data dissemination to vulnerable counter-parties was done over PGP. To further reduce the possibility of unauthorized exfiltration of sensitive data, once analysis is performed on a scan iteration, all sensitive data is purged from our production databases. One copy of the final report is stored on an IronKey Encrypted USB drive for posterity. We chose the IronKey for its onboard AES-CBC encryption engine, its self-destruct functionality, and the ability to configure it with no possibility of password recovery.

Suffice it to say, several months into the vulnerability study, the only member of the research team entrusted with the IronKey passphrase forgot the passphrase. Said passphrase was never written down or told to anyone else. Thus, several months of vulnerability data was lost as the IronKey quietly self-destructed in front of most of the research group. Despite this setback, we still currently employ the same data at rest protocol as before.

[5]http://forge.mysql.com/wiki/MySQL_Proxy

3.1.6 Performance Monitoring.

Detailed performance monitoring was the latest major addition of the scanner infrastructure. Despite our efforts to improve the job scheduler implementation, scanner nodes and database servers inevitably thrashed or became over or under utilized. At the time of implementation, the scanner infrastructure had grown from a single standalone host to a three-cluster deployment, each composed of one database server and six scanner nodes. A read-only analysis database server was also created. In total, 22 machines were operating 24x7. Monitoring the performance of these machines again required significant manual intervention. As a response, we augmented the standard linux SNMPD with performance metrics specific to our scanner infrastructure. Such metrics include the number of active verification jobs, the number of incomplete transactions from all phases of the workflow, the current number of running *nmap* instances, etc. We then created a simple performance monitor dashboard which polled SNMP values and graphed them using RRDTool. Time permitting, we would like to incorporate these performance metrics into our existing Nagios deployment.

The visibility that the performance dashboard gave us allowed us to quickly fix several previously unknown performance issues. The graphical dashboard also allowed us to easily verify that the scanner was operating with reasonable speed. This greatly reduced the time to resolution of most routine problems the scanner encountered.

3.2 Organizational Challenges

Conducting large-scale, active vulnerability assessments within an University environment gave rise to at least two organizational challenges.

First, our responsibility as a research University to involve undergraduate and masters students in our research can conflict with our ethical responsibility of prudently limiting access to sensitive information to a small group of core researchers. The bulk of the engineering was done by PhD students and long time members of the research group. Towards the end of the development process, outside students who showed interest in our work were invited to participate as single-semester research project students. This is perhaps an unique feature of working within an University environment. While we had no reason to distrust project students, we also did not feel it was appropriate to grant them access to sensitive vulnerability data without proper training in our experimental protocol. To address this, another scanner cluster was cloned, wiped of sensitive data, and moved onto an isolated development network. There, the project students acclimated themselves to the codebase and workflow. After a month of preparation, students were individually granted access to the production environment once they have demonstrated proficiency and a full understanding of our policies and experimental protocol.

Second, an *active* vulnerability assessment can only be responsibly conducted with proper support and mitigation staff. Should our scanner misbehave or cause any complications for the target networks we are scanning, such problems must be acknowledged and addressed in a timely manner. To keep the lines of communication open between our research group and counter-parties who are interested in our activities, we publicized a mailing list on our official project page which forwarded all incoming emails to the entire research group as well as our University's network security operations group. With the gracious help of the University security group, we were able to quickly respond to, and in almost all cases, address opt-out requests and requests for information within 24 hours. Section 3.3.2 discusses the surprisingly low volume of responses we received regarding our experiment.

3.3 Social Implications of Large-Scale Vulnerability Research

3.3.1 Proper Disclosure and Dissemination of Data:

Perhaps the biggest challenge of our research project is the responsible dissemination of vulnerability data to the appropriate organization. As we pointed out in our recent publication [1], several ISPs operating in Asia, particularly South Korea, operated the majority of vulnerable embedded devices discovered by our scanner. This was in fact welcomed news. Hundreds of thousands of vulnerable embedded devices are owned and centrally managed by three private businesses. Thus, a significant portion of all trivially vulnerable embedded devices can be fixed with the cooperation of possibly a handful of people. However, reaching and convincing the right group of people proved quite difficult. Our research group had neither the experience nor the social network to manage the dissemination of vulnerability data to counter-parties throughout the World. Instead, we partnered with Team Cymru[6] to accomplish this task. Each time a major nexus of vulnerable devices is discovered, a member of our research group reaches out to Team Cymru with the appropriate vulnerability report. We are currently attempting to reach several major organizations regarding our findings. We are hopeful that our longitudinal study will reveal measurable impacts on the global vulnerable device population as a result of our collaborative efforts.

3.3.2 Responses to our Scanning Activities:

As shown in our recent publication [1], we received far fewer responses from the Internet regarding our study than first anticipated. Over the last seven months, our public project page served 329 unique visitors, most of which were obvious automated scans for vulnerable web services. At the time of writing, we have handled approximately 70 conversations with various network operators regarding our activities, half of which were requests for vulnerability data on their own networks. When we first began our scanning activity, we anticipated a large initial volume of inquires and opt-out requests followed by an eventual decline. In reality, we still see approximately the same volume of inquires during each iteration of the scan one year after our initial scan. This may be an indication that most networks no longer bother with scan detection, or they may ignore scanning since it is so commonplace.

In order to handle the occasional opt-out requests from network operators, we created an automated console which allowed any member of the research group to remove arbitrary network ranges from all scanner nodes. We also implemented an emergency shutdown plan which the Columbia network operators can invoke as an option of last resort. This plan essentially involved powering down the scanner nodes, then disconnecting our production network gateway router from the campus network. We have not yet had to invoke this plan.

[6]http://www.team-cymru.org/

4. LESSONS LEARNED

4.1 Measure Three Times, Cut Once.

Proper sizing of the experimental endeavor, followed by **sufficient investment** in its infrastructure is a message which has been clearly highlighted by our various technical missteps and failures. In retrospect, the creation of a clean, scalable, efficient and highly available codebase could have saved large amounts of time reimplementing bottlenecks and hunting bugs. It is tempting to hastily produce a proof of concept system to show that an idea can stand on its own legs. However, when the commitment is made to scale an experiment up to a global level, reliability and scalability should be carefully factored into the design requirements of the project infrastructure. We found that aggressively attacking bottlenecks and fixing them the "right way" often involved bigger initial investments, but quickly paid off as the experiment went on.

4.2 Do Not Reinvent the Wheel.

Leveraging existing code such as Apache, MySQL and pysqlpool increased the performance and reliability of our scanner design. Building on top of standard software platforms and protocols also greatly simplified the complexity of our codebase. Inevitably, some software packages will be buggy, unstable and inappropriate for the production environment. However, we benefitted greatly from incorporating our code into existing software platforms.

4.3 Infrastructure Counts.

Ultimately, the capability, reliability and security of our technical infrastructure allowed us to engineer and operate our vulnerability scanner in a responsible fashion. It is trivial to use *nmap* to find potentially vulnerable routers on the Internet. The existence of a fortified production network which can only be accessed via IPSec VPN, the existence of a non-sensitive development environment and our ability to track and protect sensitive data as it moves through the experimental process has enabled us to carry out our research with reasonable assurance that our servers are locked down and monitored, and that sensitive data can not be exfiltrated due to a failure in our experimental protocol or negligence. A large amount of effort was invested in building a fortified, high performance research environment. This effort created the bedrock on which the vulnerability assessment and several other sensitive ongoing projects are built. Without this solid foundation, our vulnerability assessment would not have been possible. Appendix C illustrates our network and computing infrastructure.

4.4 (Human) Infrastructure Counts.

Without the cooperation and help of Columbia University IT, our Network Security group and the IT administrators of our CS department, the vulnerability assessment most likely would not have been conducted. Thus, relationships with related groups who serve to ensure proper management and proper adherence to policy is of paramount importance, especially when the project is regarded as sensitive.

5. CONCLUSION

In this paper, we described the engineering and operations of a large-scale embedded device vulnerability scanner infrastructure. Using this system, we monitored large portions of the Internet, and were able to identify over 1.1 million publicly accessible embedded devices as trivially vulnerable, or devices which have well-known root passwords configured in their administrative interface. The scanner infrastructure was developed over a period of one year and underwent several major design iterations. We highlighted key performance bottlenecks, design missteps, technical failures and lessons learned from our experiences. We also discussed important non-technical challenges of conducting sensitive security research within an open University environment as well as the social implications of large scale *active* vulnerability assessment efforts. We believe that large-scale quantitative analysis of real-world vulnerability data is crucial to understanding and mitigating serious emerging threats on the Internet. While such research activities can yield pivotal insights into the reality of Internet insecurity, they can also have serious real-world impact if not conducted properly. Therefore, it is crucial that researchers size the effort properly, build the proper infrastructure (both organizational and technical) and clearly define experimental protocols with security and safety in mind before engaging in research activities which may adversely affect external parties.

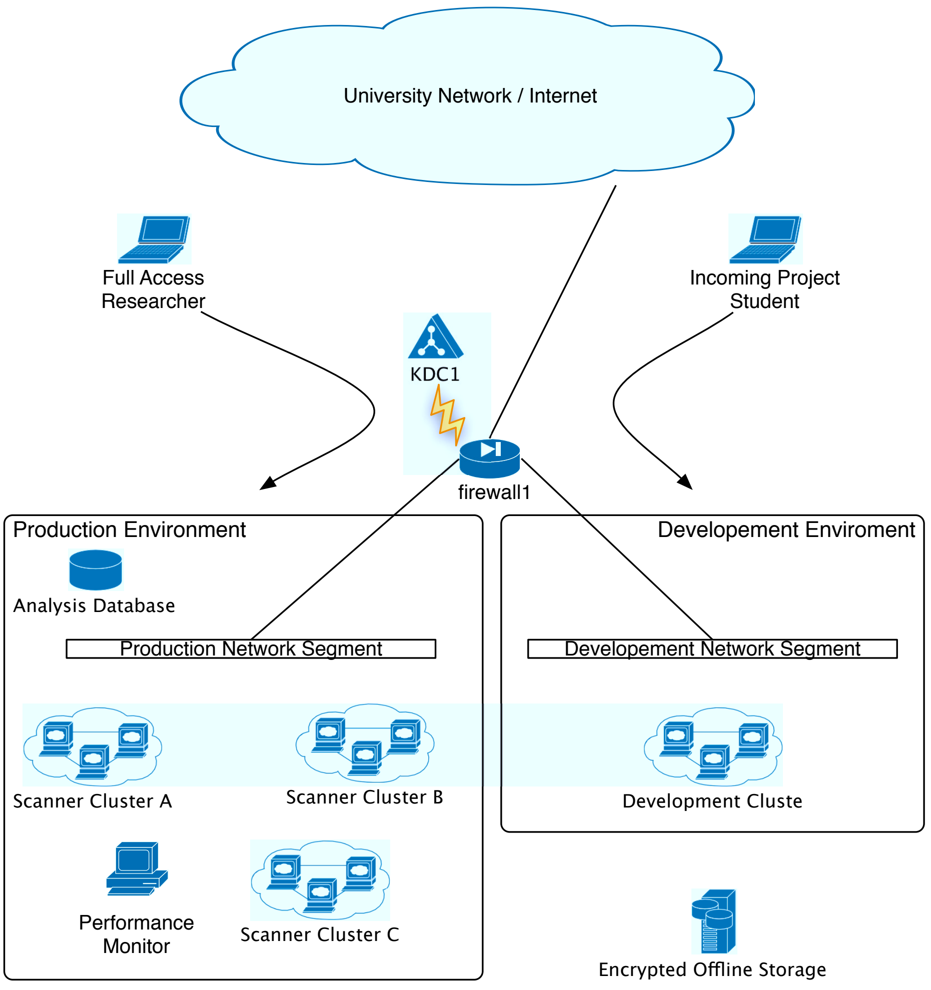
6. ACKNOWLEDGMENT

We would like to sincerely thank Joel Rosenblatt and the Columbia Network Security group for their help and support. We would also like to thank Team Cymru for making the dissemination of our vulnerability data to the proper parties possible. We would also like to thank all the past project students whose hard work made our study possible. They are: Fengwei Zhang, Keng He, Pratap Prahbu, Yuan Jochen Kang and Baibhav Pendse. This work is supported by The Office of Naval Research under grant N000140910757 and the Defense University Research Instrumentation Program (DURIP) equipment grant FA9550-09-1-0389.

7. REFERENCES

[1] Ang Cui and Salvatore J. Stolfo. A quantitative analysis of the insecurity of embedded network devices: results of a wide-area scan. In Carrie Gates, Michael Franz, and John P. McDermott, editors, *ACSAC*, pages 97–106. ACM, 2010.

.

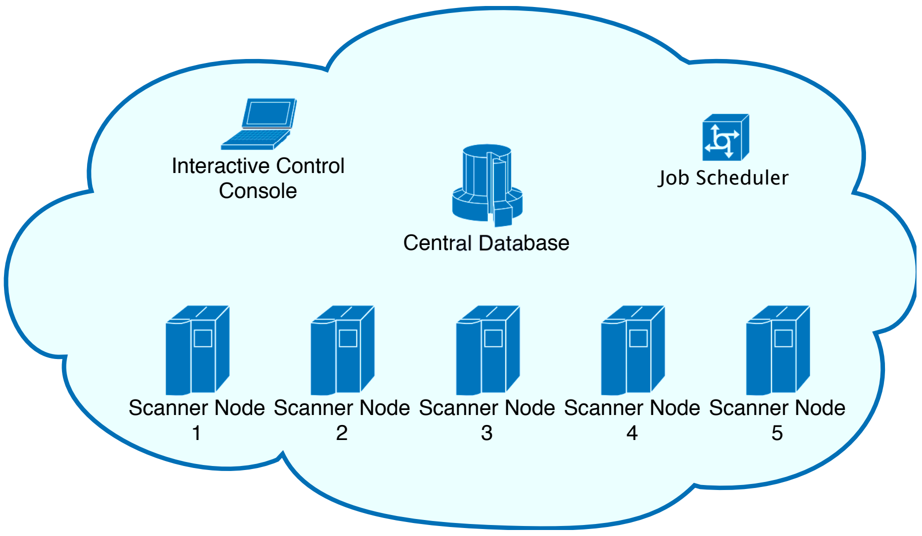


APPENDIX

APPENDIX

A. NETWORK TOPOLOGY OF SCANNER INFRASTRUCTURE

Hacktory.cs.columbia.edu, our research infrastructure is isolated from the University network and Internet by a Cisco ASA5550 firewall. Access to either the production or development environment is only possible through IPSec VPN. Users are authenticated through the VPN by the firewall via TACACS to our central kerberos server, kdc1. A *full access researcher* can access both the production and development networks while a *incoming project student* can only access the development environment. The two networks are isolated from each other. No traffic is permitted between the two segments. The production environment houses three active scanner clusters (see Appendix B, the monitoring station, and an analysis database where all sensitive data is stored while vulnerability reports are being generated. The development environment contains a single standalone host containing all components of the scanner cluster. No sensitive data is stored on this host. Lastly, vulnerability data is periodically purged from our database and stored on an IronKey.



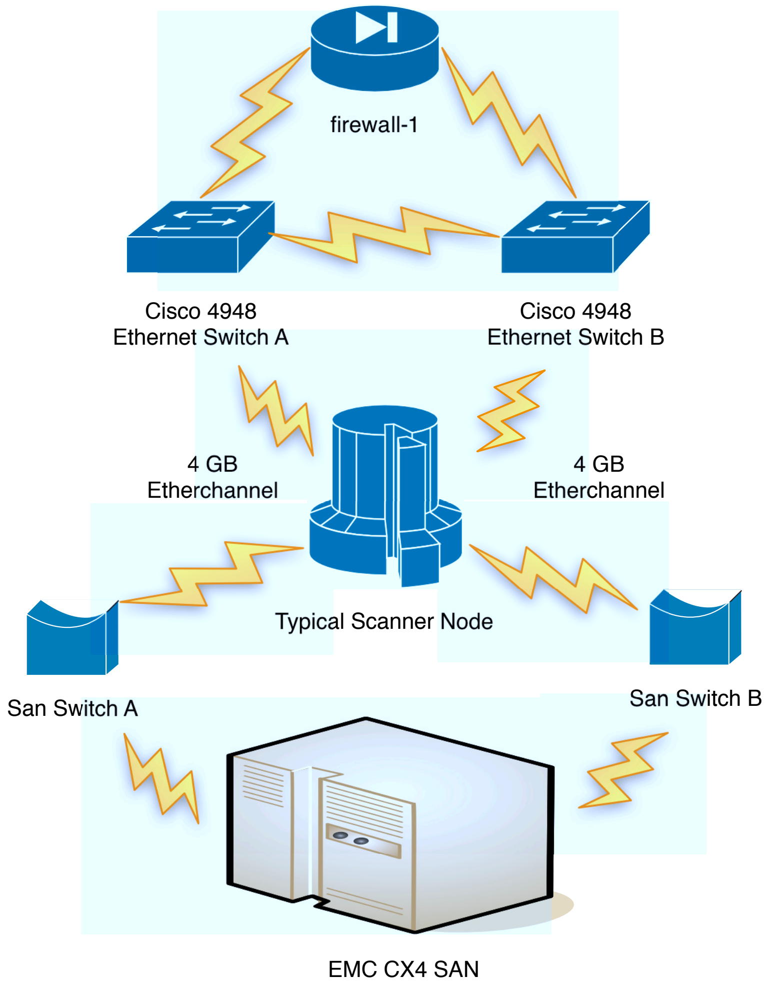
A Single Scanner Cluster

B. COMPONENTS OF A SINGLE SCANNER CLUSTER

A single scanner cluster consists of:

1. A central MySQL database.
2. 5 scanner nodes.
3. 1 job scheduler node.
4. 1 host running the interactive control console.

The job scheduler node periodically monitors the system utilization of each of the five scanner nodes, allocating scan jobs accordingly. Each scanner node is configured to run no more than 32 *nmap* instances concurrently. All scan and vulnerability information is stored within the central database. Once a scan iteration is complete, the central databases of all three scanner clusters are merged onto the main analysis database, where reports are generated.



C. PHYSICAL TOPOLOGY OF A SINGLE SCANNER NODE

The hacktory infrastructure was designed with performance and availability in mind. Scanner nodes are VM instances living on one of 10 servers running VMWare vSphere. Each vSphere node is dually connected to a pair of Cisco 4948 switches via 4GB ether-channel. Each node is also dually connected to a pair of 8GB SAN switches, which connects the server to an EMC CX4 SAN chassis containing approximately 15TB of redundant storage. For IP connectivity, the Cisco 4948 switches are then connected to our Cisco ASA5550 gateway firewall.

Blueprints of a Lightweight Automated Experimentation System: A Building Block Towards Experimental Cyber Security

Frédéric Massicotte and Mathieu Couture
Communications Research Centre Canada
Ottawa, Canada
{Frederic.Massicotte,Mathieu.Couture}@crc.gc.ca

ABSTRACT

Many research projects studying security threats require realistic network scenarios while dealing with millions of cyber threats (e.g., exploit programs and malware). For instance, studying the execution of malware may require to take into account different network configurations in which malware can propagate, as well as dealing with thousands (or millions) of different malware samples. The same challenge occurs if one wants to evaluate IDSs, study exploit programs or conduct vulnerability assessment using realistic network scenarios.

Moreover, cyber threats are highly dynamic. Every day, new vulnerabilities are identified and documented in software commonly used by computers connected to the Internet, and new malware instances and exploit programs are also identified. Consequently, it is not viable to develop (deploy) an environment every time cyber threats or security products (e.g., IDSs and anti-virus) have to be studied from a different perspective.

New research methodologies and tools are needed to systematically conduct cyber security research. Automation is required to deal with the increasingly large number of cyber threats. In this paper, we draw the foundations of an experimental approach to cyber security by providing the blueprints of a lightweight Automated Experimentation System (AES). The AES automatically executes experiments based on a specification file. We describe its usage in different network security research projects, from IDS evaluation to static and dynamic malware analysis. The results we derived from these different research projects show that our experimental approach to cyber security, enabled by the AES, enhances the scope (and scale) of research in this field. Consequently, the AES improves our understanding of cyber threats and our assessment of the current state of security products.

Categories and Subject Descriptors

C.2 [**Computer-Communication Networks**]: Network Architecture and Design; H.1 [**Models and Principles**]: Miscellaneous; I.6 [**Simulation and Modeling**]: Miscellaneous

1. INTRODUCTION

Many security products such as intrusion detection systems (IDSs), firewalls and anti-virus systems protect computer networks against attacks and intrusions. Several studies report problems with these systems and propose solutions to address those problems. In the problem investigation and solution assessment phases of cyber security research projects, researchers have to deal with thousands (or millions) of cyber threats (e.g., exploit programs and malware) that affect the thousands of different computer devices, operating systems and applications available today. Moreover, because cyber threats are highly dynamic in nature, the proposed solutions have to be constantly re-assessed.

For instance, several vulnerabilities are identified in software commonly used by computer systems connected on the Internet (e.g., NVD[1] SecurityFocus[2], CVE[3]) on a daily basis. Software programs that exploit these vulnerabilities are developed. Malware programs are among the most prevalent security threats on the Internet. For instance, McAfee has identified, over a period of 22 years (from 1986 to March 2008), 10 million unique malware samples.[4] From March 2008 to March 2009, the number of malware samples they identified doubled (i.e., it reached 20 millions). Only in the first half of this year, they have catalogued 10 million new pieces of malware for a total of around 43 million unique malware samples (including variants).[5]

Given what the Internet has become, with its diversity of computing devices, operating systems and software applications, and the dynamic nature of cyber threats, we argue that cyber security needs to be studied in a systematic manner. In particular, we need a new experimental science hav-

[1]nvd.nist.gov
[2]www.securityfocus.com
[3]cve.mitre.org
[4]www.avertlabs.com/research/blog/index.php/2009/03/10/avert-passes-milestone-20-million-malware-samples/
[5]www.mcafee.com/us/local_content/reports/q22010_threats_report_en.pdf

ing cyber security as its subject of interest. This new science would enrich the already established theoretical and practical approaches to cyber security. New research methodologies, metrics and tools are required to build (define) the foundations of experimental cyber security.

In this paper, to address this shortage of experimentation tools and to define the foundations of an experimental science approach to cyber security, we present the blueprint for an Automated Experimentation System (AES) that can be used to conduct cyber security experiments. The AES is the control component of our Cyber Observatory, in which cyber threats are the subjects of experiments to derive, among other things, cyber threat analysis and assessment of security products. The AES differentiates itself from other experimentation tools such as DETER [8] and vGround [12] on two aspects. First, it was designed to conduct multiple lightweight experiments (i.e., only using small networks composed of a few nodes) as quickly as possible, to be able to experiment with large numbers of cyber threats. DETER and vGround, on the other hand, are designed to conduct smaller numbers of larger experiments (i.e., using large networks with many nodes). Second, DETER and vGround only automate the construction of the environment where the experiment takes place. In addition to this, the AES automates the experimentation process, making the experimentation process fast, repeatable, and self-documenting.

The contributions are two-fold. (1) We propose a new approach, enabled by our Automatic Experimentation System, to conduct cyber security experimentation (e.g., generate IDS test cases, analyze malware and vulnerabilities). (2) We present different studies performed using the AES to illustrate how it can be used to assess proposed solutions to cyber security problems.

The remainder of this paper is structured as follows. Section 2 lays out the requirements for an experimentation tool in cyber security. Section 3 presents the AES and the Cyber Observatory. Section 4 describes research projects conducted using the Cyber Observatory. Limitations are presented in Section 5. Conclusions are drawn in Section 6.

2. EXPERIMENTAL CYBER SECURITY

Some authors have recognized the need for Experimental Computer Science (ECS) and proposed a definition ([2, 7, 10, 19]). However, what is ECS exactly has yet to be agreed [9]. Nonetheless, each proposed ECS definition is largely inspired from the scientific method used in the natural sciences (e.g., physics, chemistry, and biology). The ECS process entails (step 1) forming a hypothesis, (step 2) constructing a model and making a prediction, (step 3) designing and conducting experiments to collect data and (step 4) analyzing the results of the experiments to support or negate the hypothesis.[6]

There are also a few requirements that need to be fulfilled when conducting experimental research. An essential requirement is repeatability (req. 1). This requires proper documentation of the experiments, so that others can reproduce experiments and obtained results independently to confirm or refute the claims derived from these results. In the case of cyber security, the experimentation subjects are, among other things, malware samples, exploit programs, software having known vulnerabilities, etc. Thus, to conduct experiments in cyber security, a controlled environment is required: (req. 2) for recording the experimentation execution (e.g., network traffic and logs); (req. 3) for controlling the propagation of the experiment subject (e.g., exploit and malware). This controlled environment has (req. 4) to use real and heterogeneous system configurations (e.g., exploiting and attacking real vulnerabilities); (req. 5) to enable recovery to initial state (e.g., recover after a malware infection for executing every experiments in the same initial conditions); and (req. 6) to automate the experimentation process to account for the large number of experimentation scenarios due to the dynamic nature of cyber threats.

The AES was designed to automate step 3 of the ECS process and fulfill req. 1 to 6.

3. CYBER OBSERVATORY

In this section, we present the Cyber Observatory, which we have been using for five years to obtain meaningful results despite the dynamic nature of cyber threats. The Cyber Observatory is composed of four entities: a software library called the AES Engine (Section 3.1), a collection of more than 200 virtual machine images called the Virtual Laboratory (VLab) (Section 3.2), an array of networked computers located in our labs called the AES Farm (Section 3.3) and a monitoring system called the AES Dashboard (Section 3.4).

To create environments for study of computer programs, virtual machines are often used. They are an excellent automation enabler. For the VLab, we selected VMware,[7] among others (e.g. QEMU[8], VirtualBox[9] and Xen[10]), as we already had a positive experience with it [6].

The AES Engine was designed to automate step 3 of the ECS process and to fulfill req. 1 and 6. The VLab fulfills req. 2 to 5 (Section 2). The AES Engine uses as input an experiment description file (Section 3.1.1) that specifies (and documents) each experiment, step by step, as well as its results (Section 3.1.3), allowing the same experiment to be conducted many times and to be reproduced (req. 1). Virtualization provides an environment that allows the capture of information produced by the virtual machines during the experiment. Consequently, the captured information (traffic traces and various logs) can then be used to study cyber threats (req. 2). With virtualization[11] the propagation of the studied cyber threats is confined, thus preventing infection of the AES Farm (req. 3). Virtualization also facilitates the creation of template virtual machines having dif-

[6]http://www.jiahenglu.net/course/researchmethod/slides/lec9.pdf

[7]www.vmware.com

[8]wiki.qemu.org

[9]www.virtualbox.org

[10]www.xen.org

[11]We are aware that some malware samples are able to detect virtual machines. However, we believe that as virtualization is becoming more commonly used, malware samples would have to execute on those virtual systems to be efficient. If not, using virtualization for your systems could become a way to prevent malware infection.

ferent software configurations (operating systems, services, etc) that can be used to rapidly deploy customized network configurations within a single computer (req. 4). Also, snapshots allow restoration of the virtual machines used in the experiment to the state they were in before each experiment. All experiments can then be performed under the same initial conditions (req. 5). Finally, the AES Engine automatically conducts the experiments (req. 6).

The experimental results (e.g., traffic traces from malware and exploit program execution) are post-processed by BEAVER (BEhavioural Analysis using Virtualization and Experimental Research). BEAVER extracts behavioural information (e.g., HTTP requests made by a malware, IDS events generated) on the experiment subjects, and stores this information in a database that can be accessed through a web portal. Due to space restrictions, the BEAVER analysis system is not presented in this paper.

3.1 AES Engine

In this section, we present the AES Engine. Section 3.1.1 presents the language we developed to specify experiments. Section 3.1.2 describes how an experiment is conducted by the AES Engine. Section 3.1.3 describes the documentation process of the experiment results.

3.1.1 Experiment Language

The AES Engine module (programmed in Java) is used to automatically conduct the experiments provided by the user, and to gather the results. These experiments are either generated manually or generated by a script from a template based on the experimental hypothesis (ECS step 1). These experiments are specified in XML files using our own experiment language. The experiment language specifies (1) the network topology and (2) an Experiment Execution Graph (EEG).

The network topology specifies the virtual machines required by the experiment and how these virtual machines are connected together. Note that the network topology is not limited to a LAN. More complex topologies can be used as long as the network infrastructure (e.g., routers) can be virtualized. The virtual machines that are part of an experiment are called *actors*. Each of these *actors* has a set of network interfaces identified by a unique label. Each interface is connected to a specific LAN. Thus, the actors and the network interfaces specify the network topology in which the experiment will take place.

The EEG specifies the different steps of the experiment and the order in which they are to be executed. The EEG is composed of labeled steps, each separated by an operator. The EEG supports two operators: a sequencing operator (i.e., **.**) and a concurrency operator (i.e., |).

- **Sequencing (.):** The sequencing operator specifies an ordering of the steps. For example, the EEG $S_1.S_2$ specifies that S_1 has to be performed first and then S_2 can be performed. The sequencing operator also specifies that S_2 cannot be performed before S_1 is completed.
- **Concurrency (|):** The concurrency operator specifies that two steps can be performed in parallel. For example, the EEG $S_1.(S_2 \mid S_3).S_4$ specifies that S_1 has to be performed and completed first, then S_2 and S_3 can be performed concurrently and finally S_4 can be performed when both S_2 and S_3 are completed.[12]

Figure 1 shows a simplified example of an experiment used to study the exploit program `jill` against Windows 2000 Server. Lines 2 to 10 specify the network topology. There are three actors: `VMWin2000ServerTarget` (i.e., the target system), `VMLinuxRH80Attack` (i.e., the attacker system) and `VMWin2000ServerDNS` (i.e., the Domain Name System used to resolve computer names). Each actor is powered on on a specific snapshot and all network interfaces are connected to the virtual network (i.e., `vmnet`) 1. This network topology is shown in Figure 2. Line 1 specifies the EEG and lines 11 to 15 specify each step of the EEG. Each step is associated with an actor (e.g., `actor="VMLinuxRH80Attack"`) that has to execute a specific command (e.g., `cmd="jill 10.92.39.14 80 10.92.39.3 30"`). Step S_1 and S_2 (lines 11 and 12) are used to check whether the attack system (i.e., `VMLinuxRH80Attack`) is able to communicate with the target system and the Domain Name System (DNS). To accomplish this task, the attacker system uses the command `ping -c 4` with the target system (S_1) and DNS (S_2) IP addresses in parallel. Thus, the Step S_3 will only begin when both S_1 and S_2 are completed. Step S_3 (line 13) instructs the `VMLinuxRH80Attack` actor (i.e., the attacker system) to start capturing the network traffic that will be generated by the exploit program on the interface `eth0`. Step S_4 (line 14) specifies to execute `jill` against the target system and to keep the output of this command in a file called `jill.res.txt`. Step S_5 (line 15) instructs the actor to stop capturing the network traffic; and to keep the output of the sniffer in a file called `jill.res.pcap`

The step specification in our experiment language is more expressive than what is shown in the simplified example presented in Figure 1. For instance, the experiment steps have to be resilient to problems occuring during the experimentation (e.g., a blue screen or an actor reboot caused by a malware). Here is the description of other parameters that can be used in the step specification.

- `id`: The identifier of the step (e.g., S_0) which is used by the EEG.
- `description`: A string describing the step (e.g., "Capture Network Traffic") which provides a human readable description of the step.
- `actor`: The actor executing the step.
- `cmd`: The command to execute. This field can contain any command that can be executed in a command prompt or shell. Thus, this field can be used to execute batch files, shell scripts, and AutoIt[13] scripts to control the graphical interface of applications. Finally, it can contain AES predefined commands (Table 1).

[12]The concurrency operator will be implemented when it is needed to support some future experiments.
[13]`www.autoitscript.com`

```
<experiment executiongraph="(S1|S2).S3.S4.S5"/>
<actor id="VMWin2000ServerTarget" snapshot="Target">
      <nic interface="1" VMNet="1" />
</actor>
<actor id="VMLinuxRH80Attack" snapshot="Attack">
      <nic interface="1" VMNet="1" />
</actor>
<actor id="VMWin2000ServerDNS" snapshot="DNS">
      <nic interface="1" VMNet="1" />
</actor>
<step id="S1" actor="VMLinuxRH80Attack" cmd="ping -Űc 4 10.92.39.14"/>
<step id="S2" actor="VMLinuxRH80Attack" cmd="ping -Űc 4 10.92.36.1"/>
<step id="S3" actor="VMLinuxRH80Attack" cmd="START SNIFFER eth0"/>
<step id="S4" actor="VMLinuxRH80Attack" cmd="jill 10.92.39.14 80 10.92.39.3 30" output="jill.res.txt"/>
<step id="S5" actor="VMLinuxRH80Attack" cmd="STOP SNIFFER eth0" output="jill.res.pcap"/>
</experiment>
```

Figure 1: XML Experiment File Example.

- `output`: The name of the file where the output of the command `cmd` will be stored.

- `keepoutput`: A Boolean value that specifies whether or not to keep the file specified by the `output` field as one of the results of the experiment.

- `timeout`: A numerical value that specifies in milliseconds the maximum amount of time allocated to the command `cmd` to complete. For example, if the timeout is 2000, the command has a maximum of two seconds to complete. If not, the AES Engine will kill the process and continue to the next step. A value of 0 for the `timeout` field specifies that the command `cmd` has no restriction for its completion time. Note that the `timeout` can also be used on the experiment itself (not only on a step) to specify the maximum amount of time allocated for the entire experiment.

- `mandatory`: A Boolean value that specifies whether or not this step is essential (mandatory) to the success of the experiment. For instance, the ssh daemon used to communicate with a given actor of the experiment might become unresponsive (either because it has been shutdown by malware, a blue screen has occurred, or whatever other reason), thus causing the download of some log files to fail. If the step is mandatory, then the experiment will be aborted, all the files gathered up to this point will be deleted, and the experiment will eventually be re-attempted. If the step is not mandatory, then the step will simply be marked as failed in the log file.

The AES Engine offers a set of predefined commands, the list is shown in Table 1. Note that the AES predefined command `SCREENSHOT` is useful when the subject of the experiment (e.g., a malware or an exploit programs) causes damages (e.g., blue screen) or generates errors (e.g., missing library errors) that can be visually noticed by users. For example, we post-process these screenshot images after the experiment to automatically identify what went wrong by searching for sub-images (e.g., a missing library pop-up window) within the screenshot image. Also note that one could use standard commands to start and to stop a sniffer such as *Windump*[14], *Tcpdump*[15] or *WireShark*[16]. However, we noticed that using our AES predefined commands, `START_SNIFFER` and `STOP_SNIFFER`, provide in some situations more reliable experiment results since they execute a series of commands and checks, that are sometimes essential to the quality of the traffic traces.

AES Command	Description
`RECEIVE_FILE` *filename*	Send the file *filename* to the `actor`.
`GET_FILE` *filename*	Get the file *filename* from the `actor`.
`SCREENSHOT`	Take a screenshot (i.e., capture the screen) of the `actor`.
`SLEEP` *n*	Sleep for *n* millisecond(s).
`START` *p*	Start process *p*.
`STOP` *p*	Stop process *p*.
`START_SNIFFER` *interface*	Start capturing traffic on *interface.*
`STOP_SNIFFER` *interface*	Stop capturing traffic on *interface.*

Table 1: AES Predefined Command.

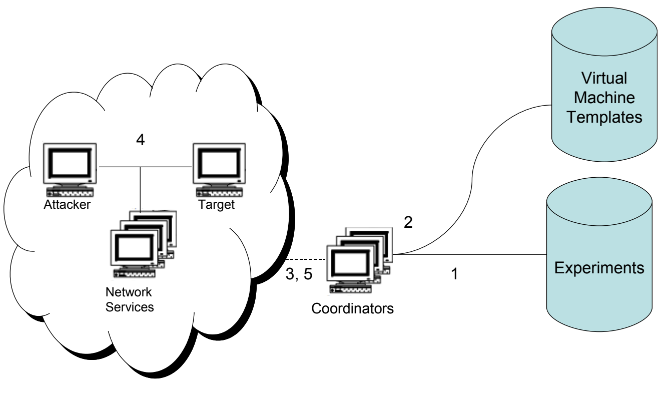


Figure 2: Automated Experimentation System

3.1.2 Experimentation

In this section, we present the sequence of actions that the AES Engine has to accomplish to fulfill a set of experiments.

[14]`www.winpcap.org/windump/`
[15]`www.tcpdump.org`
[16]`www.wireshark.org`

Figure 2 presents the AES experimentation process. The coordinators (part of the AES Farm) are computer systems running the AES Engine code to control the virtual machines of the VLab. This process is repeated for each experiment. Each coordinator of the AES Farm is able to execute one experiment at a time. However, many coordinators can be used in parallel on a physical machine (Section 3.3). Note that the network topology in Figure 2 is similar to the one of the experiment presented in Figure 1, where the experiment contains an attacker (Attacker), a target system (Target) and network infrastructure services (Network Services). Here are the phases of the AES experimentation process (the phase numbers refer to the numbers in Figure 2):

1. *Experiment Negotiation.* Each coordinator negotiates the experiment to conduct with the other coordinators to ensure that it does not conduct an experiment that was already conducted or that is currently conducted by another coordinator. When the coordinator has identified an experiment that is available, it *locks* the experiment so no other coordinator can conduct it.
2. *VLab Setup.* Based on the network topology of the experiment, the coordinator builds the virtual network where the experiment is to be conducted.
3. *Opening Communication Channel.* Once the virtual network is ready, the coordinator opens the communication channels with the actors of the experiments. These communication channels (e.g. a hard drive shared via VMware, a virtual parallel port, a virtual serial port, ssh, etc.) are the only way the coordinator can communicate the experiment steps to be executed to the actors. This enables the isolation of the virtual network (from a networking point of view) while keeping some communication capabilities.
4. *Experiment Execution.* The coordinator executes the EEG by providing commands to the virtual machine in the VLab.
5. *Tear Down.* The coordinator gathers the experiment results (e.g., the traffic traces) using the communication channel. Then, the coordinator documents and stores the experiment results and restores the virtual machines to their initial state.

3.1.3 Documentation Process

After an experiment is completed by the AES Engine, its results are documented. In the example of Figure 1, the AES Engine gathers the results of the execution of `jill` and the traffic trace that contains the execution of this attack.

Thus, the experiment file (`jill.exp.xml`) is stored with the results (`jill.res.txt` and `jill.res.pcap`) and a report describing the experiment execution (`jill.aes.xml`). This report file (Appendix 3.4.2) keeps among other things the success or not of the experiment, the time required to execute each step of the experiment, which coordinator conducted the experiment and which non-mandatory step were not successfully executed.

This documentation process, in combination with the experimentation specification, are essential for another computer system (e.g., BEAVER) to automatically use the experiment results as well as to make this experiment repeatable (the actors are also required), so the results can be analyzed independently to confirm or refute the claims derived from these results. It is also important to document the experiments to share results with other researchers (e.g., to make the traffic traces publicly available), so they can understand and reproduce the experiments.[17]

3.2 Virtual Laboratory

The VLab is the environment where the experiments are conducted. It is constructed from a collection of virtual machine images (VLab database). The AES Engine is the system that automatically conducts the experiments within the VLab environment. In its current implementation, the AES does not support on-the-fly configuration of the actors. The virtual machines have to be created and configured (e.g., the IP addresses have to be set) prior to conducting the experiments.

Our VLab database is a collection of more than 200 WMware Workstation and ESX virtual machine images. These virtual machines can be grouped into four categories. The *Attacker* group is composed of virtual machines loaded with vulnerability exploitation programs and frameworks (e.g., MetaSploit[18]) and vulnerability scanners (e.g., Nessus[19]). The *Network Infrastructure* group is composed of various networking components (e.g., DNS, router) to facilitate the construction of the network topology required for an experiment. The *Security Tools* group is composed of virtual machines loaded with security protection software such as IDSs and anti-virus. The *Target* group is composed of the installation of nearly all Windows, RedHat Linux, SUSE Linux, FreeBSD, NetBSD and OpenBSD operating systems available. Every operating system and network service is installed using its default settingd. This consists of more than 150 different operating system versions. These are used in our experiments as the cyber attack victims.

3.3 AES Farm

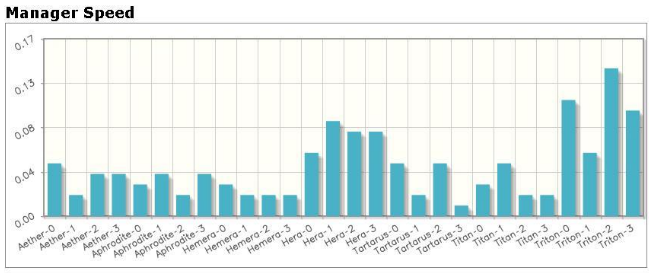
Our first AES Farm, that was used to generate our original operating system fingerprinting [6] and IDS evaluation [14] data sets, was composed of five Pentium 4 computers with 1 GB of RAM each, running Windows 2000.

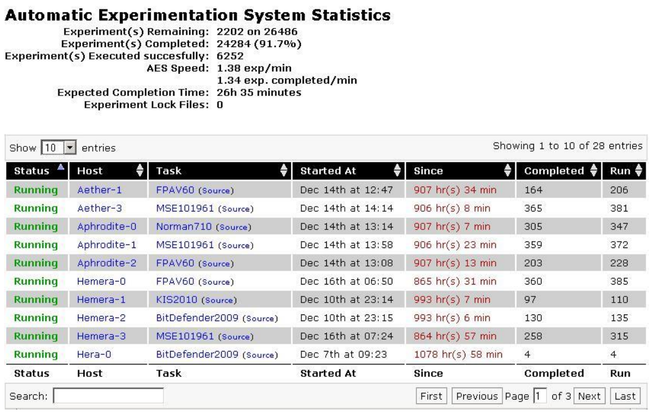
Over time, we have performed several upgrades to our farm, which is now composed of about 50 desktop computers with i7 processors and 12 GB of RAM. Each of these machines has five hard drives: one for the operating system, four for the virtual machine images. Since we discovered that hard drives are a bottleneck, we assigned one hard drive per experiment coordinator (each of these desktop computers runs four experiments simultaneously). Of these 50 machines, around 20 are used with VMware Workstation running on Windows XP, 7 or 2008 and 30 are used with VMware ESX.

[17]Note that most of the experiment outputs presented in Section 4 are available upon request by sending an email to `networksystem-security@crc.gc.ca` with the subject `Experimentation Data Request`, your affiliation, a brief description of your research project and an explanation of why this data could be useful to your research.

[18]`www.metasploit.com`

[19]`www.nessus.org`




Automatic Experimentation System Statistics

Experiment(s) Remaining:	2202 on 26486
Experiment(s) Completed:	24284 (91.7%)
Experiment(s) Executed succesfully:	6252
AES Speed:	1.38 exp/min
	1.34 exp. completed/min
Expected Completion Time:	26h 35 minutes
Experiment Lock Files:	0

Show 10 entries Showing 1 to 10 of 28 entries

Status	Host	Task	Started At	Since	Completed	Run
Running	Aether-1	FPAV60 (Source)	Dec 14th at 12:47	907 hr(s) 34 min	164	206
Running	Aether-3	MSE101961 (Source)	Dec 14th at 14:14	906 hr(s) 8 min	365	381
Running	Aphrodite-0	Norman710 (Source)	Dec 14th at 13:14	907 hr(s) 7 min	305	347
Running	Aphrodite-1	MSE101961 (Source)	Dec 14th at 13:58	906 hr(s) 23 min	359	372
Running	Aphrodite-2	FPAV60 (Source)	Dec 14th at 13:08	907 hr(s) 13 min	203	228
Running	Hemera-0	FPAV60 (Source)	Dec 16th at 06:50	865 hr(s) 31 min	360	385
Running	Hemera-1	KIS2010 (Source)	Dec 10th at 23:14	993 hr(s) 7 min	97	110
Running	Hemera-2	BitDefender2009 (Source)	Dec 10th at 23:15	993 hr(s) 6 min	130	135
Running	Hemera-3	MSE101961 (Source)	Dec 16th at 07:24	864 hr(s) 57 min	258	315
Running	Hera-0	BitDefender2009 (Source)	Dec 7th at 09:23	1078 hr(s) 58 min	4	4
Status	Host	Task	Started At	Since	Completed	Run

Search: First Previous Page 1 of 3 Next Last

Figure 3: AES Monitor

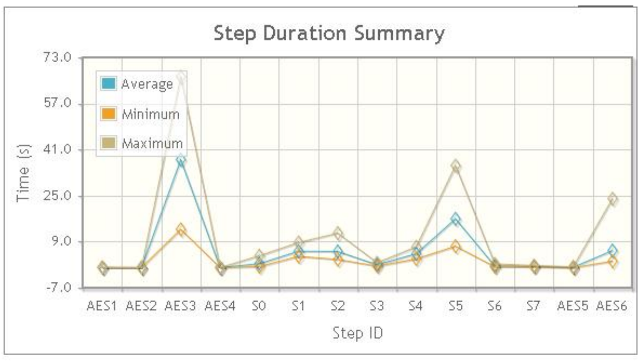
We also have five server-class computers running VMware ESX, two with 8 Intel Xeon dual core processors and 64 GB of RAM, three with 2 Intel Xeon quad core processors and 32 GB of RAM. Since the AES is designed to conduct large numbers of lightweight (small) experiments (i.e., using small networks) as quickly as possible, pulling the virtual machine images from the hard drive (at the beginning of every experiment) is a time-consuming process. Consequently, the desktop computers are more cost-effective for us in terms of experiment per minute per dollar.

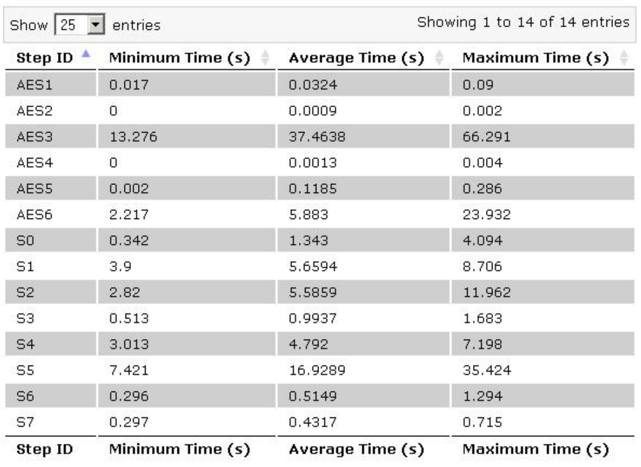
3.4 AES Dashboard

Simultaneously conducting dozens of experiments can easily become overwhelming in terms of monitoring the progression of the whole experiment set. A given experiment may stall, one of the virtual machines may freeze, steps may require more time than expected, etc. For this reason, we developed tools to monitor the system. We present them in this section, as well as some other existing tools that we found useful.

3.4.1 Progression

We developed a web-based application allowing to visualize the overall progress and speed of the AES. Figure 3 shows the main view of this application. On top, a graphical view of the respective speed (in terms of experiments per minute) of each experiment coordinator is provided. This allows us to compare the performance of the different software and hardware configurations of the various servers within our AES Farm. Detailed information in terms of the number of experiments performed, with a distinction between those that failed and those that succeeded, is also provided for each experiment coordinator. Finally, global performance of the AES is also provided.




Show 25 entries Showing 1 to 14 of 14 entries

Step ID	Minimum Time (s)	Average Time (s)	Maximum Time (s)
AES1	0.017	0.0324	0.09
AES2	0	0.0009	0.002
AES3	13.276	37.4638	66.291
AES4	0	0.0013	0.004
AES5	0.002	0.1185	0.286
AES6	2.217	5.883	23.932
S0	0.342	1.343	4.094
S1	3.9	5.6594	8.706
S2	2.82	5.5859	11.962
S3	0.513	0.9937	1.683
S4	3.013	4.792	7.198
S5	7.421	16.9289	35.424
S6	0.296	0.5149	1.294
S7	0.297	0.4317	0.715
Step ID	Minimum Time (s)	Average Time (s)	Maximum Time (s)

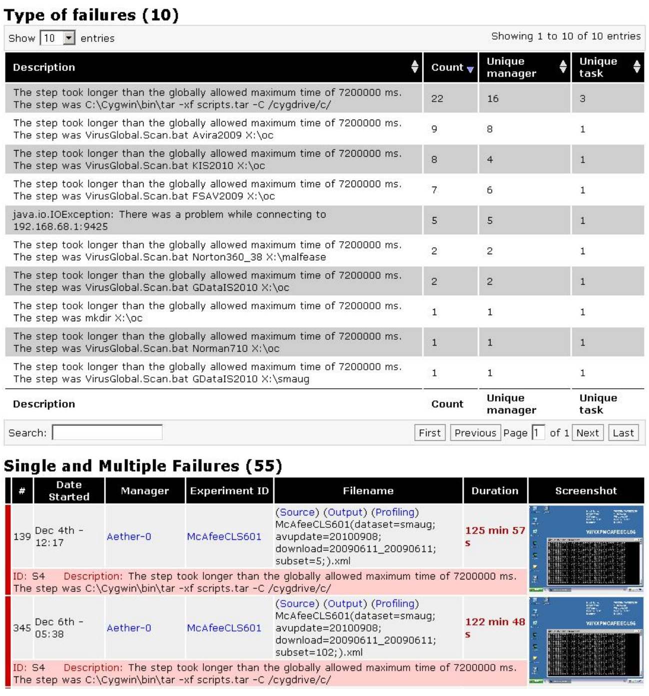
Figure 4: AES Profiler

3.4.2 Profiling

When designing an experiment, it is not always clear which steps will be time consuming, and optimizing an experiment is not a trivial process. To facilitate this process, the AES produces a report which is stored with each experiment results. This report contains the time required for each step. It also contains the time it is using for its *hidden* steps, such as powering on the virtual machines, setting up the topology, etc. Figure 4 shows a screenshot of a web-based application that we developed to visualize the average time of each step of an experiment set. In this case, each experiment is made of eight steps, and there are six *hidden* steps (four to prepare the experiment, and two to shut it down). The most time-consuming step, in this case, is the third hidden step of the experiment setup, which consists in powering on the virtual machines. This is due to the significant amount of time that is required to pull the virtual machine images from the hard drive. Once we know that, we know that the hard drive is a bottleneck in our system. Incidentally, we are currently investigating the possibility of using solid state hard drives to improve the overall performance of the AES.

3.4.3 Error Reporting

The code of the AES has its own error handling mechanism. When an error occurs, causing the experiment to fail, it is logged and the experiment remains in the queue of experiments to be performed. Sometimes, the error is only incidental and the experiment completes successfully on the next attempt. However, it could be due to a design



Type of failures (10)

Show 10 entries — Showing 1 to 10 of 10 entries

Description	Count	Unique manager	Unique task
The step took longer than the globally allowed maximum time of 7200000 ms. The step was C:\Cygwin\bin\tar -xf scripts.tar -C /cygdrive/c/	22	16	3
The step took longer than the globally allowed maximum time of 7200000 ms. The step was VirusGlobal.Scan.bat Avira2009 X:\oc	9	8	1
The step took longer than the globally allowed maximum time of 7200000 ms. The step was VirusGlobal.Scan.bat KIS2010 X:\oc	8	4	1
The step took longer than the globally allowed maximum time of 7200000 ms. The step was VirusGlobal.Scan.bat FSAV2009 X:\oc	7	6	1
java.io.IOException: There was a problem while connecting to 192.168.68.1:9425	5	5	1
The step took longer than the globally allowed maximum time of 7200000 ms. The step was VirusGlobal.Scan.bat Norton360_38 X:\malfease	2	2	1
The step took longer than the globally allowed maximum time of 7200000 ms. The step was VirusGlobal.Scan.bat GDataIS2010 X:\oc	2	2	1
The step took longer than the globally allowed maximum time of 7200000 ms. The step was mkdir X:\oc	1	1	1
The step took longer than the globally allowed maximum time of 7200000 ms. The step was VirusGlobal.Scan.bat Norman710 X:\oc	1	1	1
The step took longer than the globally allowed maximum time of 7200000 ms. The step was VirusGlobal.Scan.bat GDataIS2010 X:\smaug	1	1	1
Description	Count	Unique manager	Unique task

Search: — First Previous Page 1 of 1 Next Last

Single and Multiple Failures (55)

#	Date Started	Manager	Experiment ID	Filename	Duration	Screenshot
139	Dec 4th - 12:17	Aether-0	McAfeeCLS601	(Source) (Output) (Profiling) McAfeeCLS601(dataset=smaug; avupdate=20100908; download=20090611_20090611; subset=5;).xml	125 min 57 s	
ID: S4 Description: The step took longer than the globally allowed maximum time of 7200000 ms. The step was C:\Cygwin\bin\tar -xf scripts.tar -C /cygdrive/c/						
345	Dec 6th - 05:38	Aether-0	McAfeeCLS601	(Source) (Output) (Profiling) McAfeeCLS601(dataset=smaug; avupdate=20100908; download=20090611_20090611; subset=102;).xml	122 min 48 s	
ID: S4 Description: The step took longer than the globally allowed maximum time of 7200000 ms. The step was C:\Cygwin\bin\tar -xf scripts.tar -C /cygdrive/c/						

Figure 5: AES Error Reporting

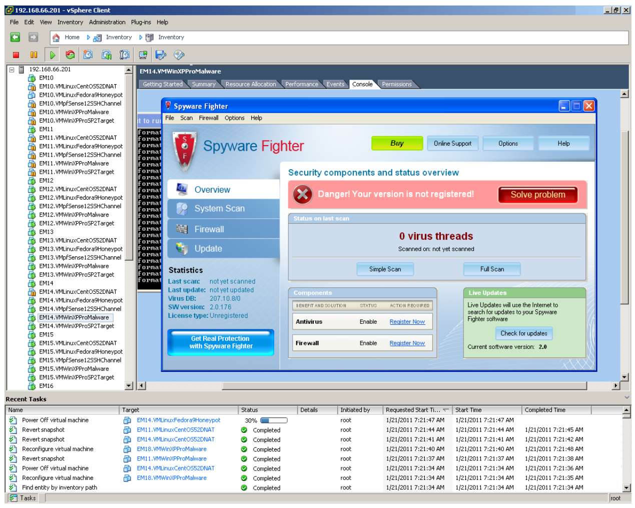
error in the experiment, a configuration error in the virtual machine, something specific about the object of the experiment (a particular malware sample or exploit), an error in the host (configuration error in VMware), etc. To help us discriminate between these various situations, and identify the source of the error more quickly, we developed another web-based application to visualize these various errors. Figure 5 shows error visualization for our static malware analysis experiment set. Most of them are due to scanning taking longer than expected. When an error is raised, a screenshot of the actor of the step causing the error is taken. For instance, Figure 5 shows a list of experiment errors that we obtained with several anti-virus. In these cases, our AutoIt scripts were responsible for the errors.

3.4.4 Desktop Viewers

To visualize the display of each experiment coordinator and virtual machines, we use VMware Infrastructure Client[20] and an open source VNC client called VNC Thumbnail Viewer[21], respectively shown in Figure 6 and Figure 7. The VMware client allows us to see the display of all virtual machines, regardless or whether they are in an isolated network or not. We use the VNC client for seeing and controlling the experiment coordinators. Also, with the latest version VMware Workstation, one can connect to any virtual machine through VNC. Since this feature does not rely on a VNC server being installed inside the virtual machine, but is rather managed directly by VMware Workstation, this feature enables the remote visualization of all virtual machines, even those located in an isolated network.

[20] `www.vmware.com/products/vsphere/`

[21] `thetechnologyteacher.wordpress.com/vncthumbnailviewer/`


192.168.66.201 - vSphere Client
File Edit View Inventory Administration Plug-ins Help
Home Inventory Inventory
EM14.VMWinXPProMalware
Getting Started Summary Resource Allocation Performance Events Console Permissions
Spyware Fighter
Buy Online Support Options Help
Overview
System Scan
Firewall
Update
Security components and status overview
Danger! Your version is not registered!
Solve problem
0 virus threads
Simple Scan
Full Scan
Statistics
Get Real Protection with Spyware Fighter
Components
Live Updates
Check for updates

Last update: not yet updated
Virus DB: 207.10.8/0
SW version: 2.0.176
License type: Unregistered

BENEFIT AND SOLUTION	STATUS	ACTION REQUIRED
Antivirus	Enable	Register Now
Firewall	Enable	Register Now

Live Updates will use the Internet to search for updates to your Spyware Fighter software

Current software version: 2.0

Recent Tasks

Name	Target	Status	Details	Initiated by	Requested Start Ti...	Start Time	Completed Time
Power Off virtual machine	EM14.VMLinuxFedora9Honeypot	30%		root	1/21/2011 7:21:47 AM	1/21/2011 7:21:47 AM	
Revert snapshot	EM11.VMLinuxCentOS52DNAT	Completed		root	1/21/2011 7:21:44 AM	1/21/2011 7:21:44 AM	1/21/2011 7:21:45 AM
Revert snapshot	EM14.VMLinuxCentOS52DNAT	Completed		root	1/21/2011 7:21:41 AM	1/21/2011 7:21:41 AM	1/21/2011 7:21:42 AM
Reconfigure virtual machine	EM18.VMWinXPProMalware	Completed		root	1/21/2011 7:21:40 AM	1/21/2011 7:21:40 AM	1/21/2011 7:21:48 AM
Revert snapshot	EM11.VMWinXPProMalware	Completed		root	1/21/2011 7:21:37 AM	1/21/2011 7:21:37 AM	1/21/2011 7:21:38 AM
Power Off virtual machine	EM14.VMLinuxCentOS52DNAT	Completed		root	1/21/2011 7:21:34 AM	1/21/2011 7:21:34 AM	1/21/2011 7:21:36 AM
Reconfigure virtual machine	EM18.VMWinXPProMalware	Completed		root	1/21/2011 7:21:34 AM	1/21/2011 7:21:34 AM	1/21/2011 7:21:35 AM
Find entity by inventory path		Completed		root	1/21/2011 7:21:34 AM	1/21/2011 7:21:34 AM	1/21/2011 7:21:34 AM

Figure 6: VMware Infrastructure Client Experiment Viewer

4. THE CYBER OBSERVATORY IN ACTION

Figure 8 presents the Cyber Observatory. First, we automatically gather new cyber threats on a daily basis using our Web Crawler, honeypot networks and commercial/academic feeds. The VLab images are manually updated/created based on the requirements of the experiments. Second, the information about cyber threats, VLab images and analysis results from previous experimentations are used to generate the experiments that will be conducted by the AES. Since 2005, we have used the AES within our Cyber Observatory to perform various types of experimentations. Some are continuously running everyday (e.g., experiments presented in Section 4.2, 4.6 and 4.7) to provide insightful information on old, current and new cyber threats and some others were punctual experimentations (e.g., experiments presented in Section 4.1, 4.3, 4.4 and 4.5). Third, the AES conducts these experiments and the results are analyzed by BEAVER analysis module. Finally, the analysis results are stored in a database that can be accessed through a web portal. In this section, we described seven of these experimentations. For scientific conclusions drawn from each experimentation, we refer the reader to the bibliographic references provided in each sub-section.

4.1 Operating System Fingerprinting

We initiated our work in experimental cyber security while doing research on Operating System Fingerprinting [6]. To pursue this project, we needed traffic traces recorded during the execution of several fingerprinting tools against many different operating systems. We created nearly 200 *Target* images with different operating system versions, which constituted the very first version of the VLab. However, as we had not yet foreseen the long-term need for automation, the execution of the various fingerprinting tools and recording of the traffic traces required manual intervention. It quickly became clear that automation was required for conducting such as large experiment.

4.2 Intrusion Detection System Evaluation

7

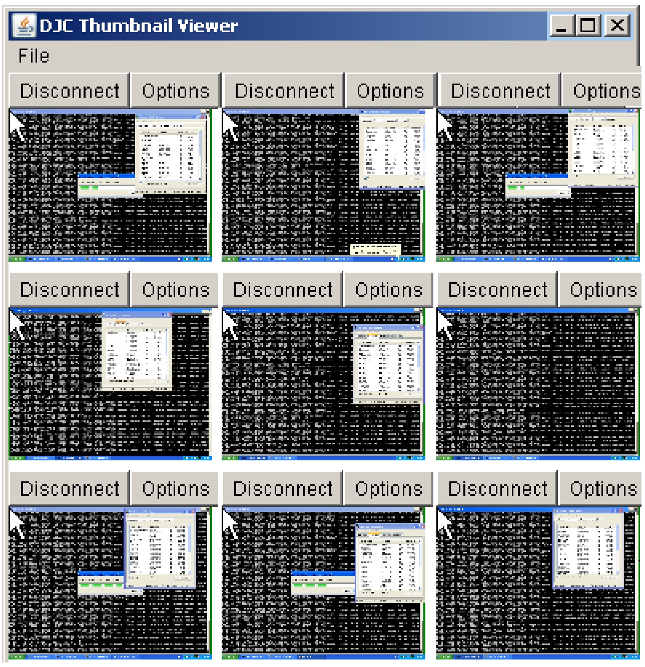


Figure 7: VNC Thumbnail Experiment Viewer

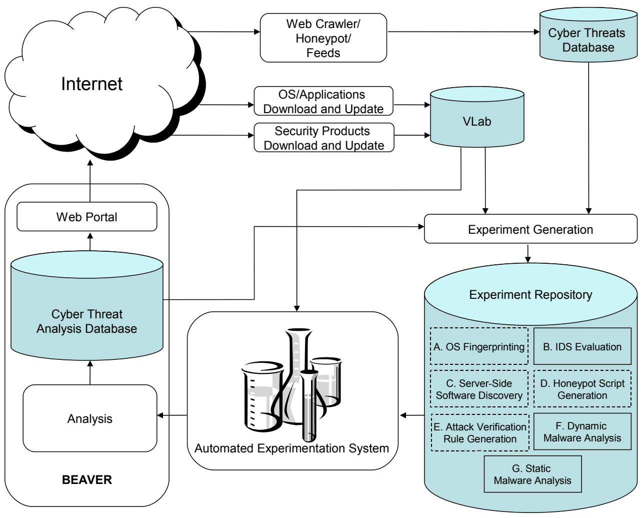

Internet
Web Crawler/ Honeypot/ Feeds
Cyber Threats Database
OS/Applications Download and Update
Security Products Download and Update
VLab
Web Portal
Cyber Threat Analysis Database
Analysis
BEAVER
Experiment Generation
Experiment Repository
A. OS Fingerprinting
B. IDS Evaluation
C. Server-Side Software Discovery
D. Honeypot Script Generation
E. Attack Verification Rule Generation
F. Dynamic Malware Analysis
G. Static Malware Analysis
Automated Experimentation System

Figure 8: Cyber Observatory

Our experience with Operating System Fingerprinting inspired us to develop the Automated Experimentation System (AES). The first time we used it was to produce an Intrusion Detection System (IDS) evaluation data set [14]. Our goal was to mitigate some known issues of other IDS evaluation data sets [16]. The generated data set is composed of traffic traces recorded during the execution of about 150 server-side exploit programs against more than 120 target systems.[22] The generation of this dataset demonstrated the scalability of the AES, as we used it to conduct more than 18000 experiments (one for each pair of target-exploit program, plus a few more for the various optional command-line arguments of each exploit). Although the generated data set was best suited for evaluating signature-based IDSs, it addresses some of the problems described in [16]. Later, we generated other IDS evaluation data sets that included IDS evasion techniques [14] and client-side exploit programs.

4.3 Server-Side Software Discovery

It has often been claimed that using network context information can improve IDS accuracy by discarding attacks that are unlikely to have succeeded. This approach relies on the knowledge of the target system configuration (e.g., name and version of the various software installed on it) and vulnerability databases (e.g., SecurityFocus) to determine whether or not the target system is indeed vulnerable to an attack. Although several authors proposed to use network context information in IDS signatures [5, 18], we observed that nobody had systematically assessed the effectiveness of this approach. We believe that the cumbersomeness of manually performing the numerous required test cases is the reason why no one had done it before. The AES, together with our work on operating system fingerprinting and IDS evaluation, placed us in an ideal position to perform this assessment [11].

First, we used the intrusion detection data set described in the previous section together with existing vulnerability databases to assess the effectiveness of the network context information approach in an ideal world, i.e., one where the target system configuration is already known. Secondly, we used the AES to generate a new network context information discovery data set in order to determine how effective the approach is when the target system configuration is not already known. Among other things, this new data set contained the updated operating system fingerprinting test cases of the data set presented in [6]. These new traffic traces were generated by the AES, using 15 coordinators, without human intervention within a matter of days.

4.4 Honeypot Script Generation

In a collaboration with Leita et al. at the Eurecom Institute [13], the AES was used to contribute to the improvement of a honeypot technology called ScriptGEN. What distinguishes ScriptGEN from other honeypot technologies is its ability to adapt to 0-day vulnerability exploitation techniques by automatically generating vulnerability emulation scripts from recorded attack attempts. The recorded attack attempts are replayed against a vulnerable system, and appropriate emulation scripts are generated based on the vulnerable system reaction. This allows the ScriptGEN platform to capture new self-propagating malware without human intervention.

In order to facilitate the development of the ScriptGEN algorithms, the AES was used to record about a hundred different execution instances for several exploit programs. Randomness was incorporated into each execution so that even if the target system was always at the same initial state, attack instances actually differed from each other. The traffic

[22]We did not program the AES to interact with all the virtual machine images (i.e., the different virtualization technologies) used in the operating system fingerprinting experiment.

traces generated using the AES allowed the bootstrapping of ScriptGEN, which now evolves on its own from what its various Internet sensors capture.

4.5 Attack Verification Rule Generation

Our experience with the ScriptGEN project taught us that the AES can be used to generate the large number of instances that are required by machine learning algorithms. On a subsequent project [15], we used the AES to generate machine learning instances for the attack verification problem. The attack verification problem consists in determining whether or not a detected attack attempt was indeed successful. For each type of attack (e.g., denial-of-service, buffer overflow, etc.), a list of indicators can often be established (connections refused, new ports being open on the target machine, reply packets matching a given pattern, etc.). Matching these indicators with each possible way of exploiting a vulnerability is a problem that requires more subtlety than one would imagine. For instance, a denial-of-service could affect the host (the host freezes), the service (the application does not respond, even if the TCP handshake successfully completes), the TCP/IP stack (connections are actively refused or are simply ignored) or specific users.

Building on our previous work on IDS evaluation, we used the same exploit programs and target systems (with a few new ones) we had used for [14], and generated a new data set also containing the execution of the stimuli that are required to reveal the indicators. We fed existing machine learning tools with this data set, and were able to generate attack verification rules, in a completely automated manner.

4.6 Dynamic Malware Analysis

Dynamic Malware Analysis means analyzing malware by executing it. Our Malware Analysis research program focuses on network-related aspects of dynamic malware analysis. A question we are asking is the following: what kind of network is required in order to obtain a maximum amount of information from executing malware? In [3, 4], we described network topologies and tools that we developed to extract information from malware samples by executing them in a network that is isolated from the Internet. Isolating malware from the Internet imposes limitations, however, our studies have demonstrated that very useful information can still be obtained in such a context.

The AES relies on the existence of a virtualization technology. Although it is known that most virtualization technologies can be detected by malware, our experience has shown that there seems to be enough malware authors that do not care about it for us to be able to analyze a large proportion of samples within the Cyber Observatory. It should be noted that the AES has been developed independently from any particular virtualization technology or operating system. Therefore, efforts that are being made by researchers who are focusing on internal host activity ([1, 17, 20]) could be integrated within the AES, and should be perceived as complements rather than alternatives.

One of the conclusions we drew from this project is that a network of four computers is sufficient to produce a significant amount of information [3, 4]. The solution was to use a *DNAT*[23] to ensure that the handshakes of most TCP connections are completed.

In [4], we introduced a tool that configures DNAT rules on-the-fly in order to appropriately handle things like backdoors being open on the infected system during the experiment. This tool allows for more accurate analysis of malware samples in isolated environments like the AES. The AES was also used to asses the usefulness of the tool through a case study performed using 25118 malware samples.

4.7 Static Malware Analysis

Static Malware Analysis includes any analysis that is performed on malware samples without executing them. This includes, but is not restricted to, anti-virus scanning, hard-coded strings extraction, hash computation, magic number analysis to determine the file type, etc. The VLab contains various virtual machine images that perform these tasks (i.e., about twenty anti-virus and various static malware analysis tools). In a sense, we have made our own home brew of Virus Total.[24] An important distinction is that our focus is on batch processing rather than on-demand analysis.

A static malware analysis experiment consists of scanning several samples by one anti-virus or other analysis tool. Grouping samples together in one experiment rather than performing one experiment per sample is mainly done for performance purposes. There is no technical obstacle to scanning only one sample per experiment.

One advantage of scanning malware samples in an experimental framework like the AES is that since virtual machines are reverted to an original state at the end of each experiment, the whole process becomes resilient to malware causing the anti-virus to freeze or to crash. The only samples for which the scanning results will be affected are the ones that are within the same group as the one causing the problem. Careful design of the experiment and of the results analysis scripts may even allow for flagging the sample as being a potential anti-virus crasher (an idea that we are currently investigating).

5. LIMITATIONS

There are two main limitations in the AES. First, although the AES framework was designed to support different virtualization technologies, the AES only interfaces with VMware Workstation or VMware ESX. However, we do not foresee any reason why plugins for other virtualization technologies such as QEMU, VirtualBox or Xen could not be developed. Also, we believe that it might be possible to develop an extension of the AES that would be virtualization-free. The main reason why we need virtualization is to be able to quickly revert to an initial state. We believe that using tools such as Faronics Deep Freeze,[25] which restores a fresh image upon reboot, may provide an alterative to virtualization.

Second, the size of the virtual environment (i.e., number of

[23]DNAT stands for Destination Network Address Translation. It is the opposite of NAT: it changes the destination IP address of packets.

[24]`www.virustotal.com`

[25]`www.faronics.com/html/deepfreeze.asp`

virtual machines) is also a current limitation. For all the experiments presented in Section 4, we were able to derive insightful information by only using lightweight experiments. However, for some experiments, such as simulating peer-to-peer botnets, larger experiments may be needed. Hardware resources are not quite the issue, as the AES can control experiments that run across several physical hosts. The problem mostly arises from the fact that each virtual machine in the experiment has to be manually created and configured prior to the experiment (once this is done, it can be used for an unlimited number of experiments). It would be useful if each virtual machine could be used as a template that is automatically cloneable and configurable (IP configuration, firewall rules, etc.) during the experiment setup phase.

6. CONCLUSION

In this paper, we have presented the AES, which implements an approach to automatically and systematically conduct cyber security experiments and helps to build a foundation for experimental cyber security. We presented the AES within the context of our Cyber Observatory and provided evidence that the AES has a positive impact on the scope and the scale of research projects that can be undertaken in cyber security.

The AES leverages our capabilities to conduct cyber security experiments by changing our focus from *how* to conduct cyber security experiments to *what* experiments should and can be conducted. It helps to reduce the effort required to study cyber threats. However, with the large amount of information (e.g., experimentation results) that can be generated with an experimentation system such as the AES, researchers will be facing a new problem. The next challenge will be to analyze, process and render this information so that it can be useful.

7. REFERENCES

[1] U. Bayer, A. Moser, C. Krügel, and E. Kirda. Dynamic analysis of malicious code. *Journal in Computer Virology*, 2(1):67–77, 2006.

[2] D. E. Comer, D. Gries, M. C. Mulder, A. Tucker, A. J. Turner, and P. R. Young. Computing as a discipline. *Commun. ACM*, 32:9–23, January 1989.

[3] M. Couture and F. Massicotte. Studying Malware in an Isolated Network. CRC Technical Note CRC-TN-2009-02, Communications Research Center Canada, September 2009.

[4] M. Couture and F. Massicotte. Last Minute Traffic Forwarding for Malware Analysis in a Honeynet. CRC Technical Note CRC-TN-2010-01, Communications Research Center Canada, June 2010.

[5] B. Dayioglu and A. Ozgit. Use of Passive Network Mapping To Enhance Signature Quality of Misuse Network Intrusion Detection Systems. In *Proceedings of the International Symposium on Computer and Information Sciences*, 2001.

[6] A. DeMontigny-Leboeuf. A Multi-Packet Signature Approach to Passive Operating System Detection. CRC Technical Note CRC-TN-2005-001 / DRDC-Ottawa-TM-2005-018, Communications Research Center Canada, December 2004.

[7] P. J. Denning. Acm president's letter: performance analysis: experimental computer science as its best. *Commun. ACM*, 24:725–727, November 1981.

[8] DETER. A laboratory for security research. `http://www.isi.edu/deter/` (accessed March 22, 2011).

[9] D. G. Feitelson. Experimental computer science: The need for a cultural change. `http://www.cs.huji.ac.il/~feit/papers/exp05.pdf` (accessed March 22, 2011).

[10] N. Fenton, S. L. Pfleeger, and R. L. Glass. Science and substance: A challenge to software engineers. *IEEE Softw.*, 11:86–95, July 1994.

[11] F. Gagnon, F. Massicotte, and B. Esfandiari. Using Contextual Information for IDS Alarm Classification. In *Proceedings of the Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*, 2009.

[12] X. Jiang, D. Xu, H. J. Wang, and E. H. Spafford. Virtual playgrounds for worm behavior investigation. In *Proceedings of the Recent Advances in Intrusion Detection (RAID)*, 2005.

[13] C. Leita, M. Dacier, and F. Massicotte. Automatic Handling of Protocol Dependencies and Reaction to 0-Day Attacks with ScriptGen Based Honeypots. In *Proceedings of the Recent Advances in Intrusion Detection (RAID)*, pages 185–205, 2006.

[14] F. Massicotte, F. Gagnon, Y. Labiche, M. Couture, and L. Briand. Automatic Evaluation of Intrusion Detection Systems. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, pages 361–370, 2006.

[15] F. Massicotte, Y. Labiche, and L. Briand. Toward Automatic Generation of Intrusion Detection System Verification Rules. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, pages 279–288, 2008.

[16] J. McHugh. Testing Intrusion Detection Systems: A Critique of the 1998 and 1999 DARPA Intrusion Detection System Evaluations as Performed by Lincoln Laboratory. *ACM Transactions on Information and System Security*, 3(4), November 2000.

[17] Norman Solutions. Norman sandbox whitepaper. http://download.norman.no/whitepapers/whitepaper_Norman_SandBox.pdf, 2003.

[18] R. Sommer and V. Paxson. Enhancing Byte-Level Network intrusion Detection Signatures with Context. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 262–271, 2003.

[19] W. F. Tichy. Should computer scientists experiment more? *Computer*, 31:32–40, May 1998.

[20] C. Willems, T. Holz, and F. Freiling. Toward automated dynamic malware analysis using cwsandbox. *IEEE Security and Privacy*, 5(2):32–39, 2007.

Statistical Analysis of Honeypot Data and Building of Kyoto 2006+ Dataset for NIDS Evaluation

Jungsuk Song
National Institute of Information and Communications Technology (NICT)
song@nict.go.jp

Hiroki Takakura
Information Technology Center, Naogya University
takakura@itc.nagoya-u.ac.jp

Yasuo Okabe
Academic Center for Computing and Media Studies, Kyoto University
okabe@i.kyoto-u.ac.jp

Masashi Eto
National Institute of Information and Communications Technology (NICT)
eto@nict.go.jp

Daisuke Inoue
National Institute of Information and Communications Technology (NICT)
dai@nict.go.jp

Koji Nakao
National Institute of Information and Communications Technology (NICT)
ko-nakao@nict.go.jp

Abstract

With the rapid evolution and proliferation of botnets, large-scale cyber attacks such as DDoS, spam emails are also becoming more and more dangerous and serious cyber threats. Because of this, network based security technologies such as Network based Intrusion Detection Systems (NIDSs), Intrusion Prevention Systems (IPSs), firewalls have received remarkable attention to defend our crucial computer systems, networks and sensitive information from attackers on the Internet. In particular, there has been much effort towards high-performance NIDSs based on data mining and machine learning techniques. However, there is a fatal problem in that the existing evaluation dataset, called KDD Cup 99' dataset, cannot reflect current network situations and the latest attack trends. This is because it was generated by simulation over a virtual network more than 10 years ago. To the best of our knowledge, there is no alternative evaluation dataset. In this paper, we present a new evaluation dataset, called Kyoto 2006+, built on the 3 years of real traffic data (Nov. 2006 ∼ Aug. 2009) which are obtained from diverse types of honeypots. Kyoto 2006+ dataset will greatly contribute to IDS researchers in obtaining more practical, useful and accurate evaluation results. Furthermore, we provide detailed analysis results of honeypot data and share our experiences so that security researchers are able to get insights into the trends of latest cyber attacks and the Internet situations.

Keywords NIDS, Honeypot Data, Kyoto 2006+ Dataset

1. Introduction

In general, a botnet is referred as a collection of infected hosts, *i.e.*, zombie PCs or bots, and the botnet herders use their botnets for launching large-scale cyber attacks such as Distributed Denial of Service (DDoS), spam emails, network scanning and so on. Also, in many cases, they rent the services of the botnets out to third parties who want to advertise their products or to attack a certain victim host. Due to the rapid evolution and proliferation of the botnets, network based security technologies such as Network based Intrusion Detection Systems (NIDSs), Intrusion Prevention Systems (IPSs), firewalls have received remarkable attention to defend our crucial computer systems, networks and sensitive information from attackers on the Internet. In particular, there has been much effort towards high-performance NIDSs based on data mining and machine learning techniques[1, 2].

In intrusion detection field, KDD Cup 99' dataset[3] has been used for a long time as evaluation data of NIDSs. However, there is a fatal problem in that the KDD Cup 99' dataset cannot reflect current network situations and the latest attack trends. This is because it was generated by simulation over the virtual network more than 10 years ago. To the best of our knowledge, there is no alternative evaluation dataset for NIDSs. This is because it is quite difficult to get high-quality real traffic data which contain both normal and attack data for long time constantly. In addition, it is extremely time-consuming to label traffic data as either normal or intrusion, because security experts have to inspect every traffic data and classify them accurately. To make matters worse, due to the privacy and competitive issues, many organizations and researcher do not share their data with other institutions and researchers, even if they have real traffic data.

In this paper, we present a new evaluation dataset, called Kyoto 2006+, built on the 3 years of real traffic data (Nov. 2006 $\sim$ Aug. 2009). It consists of 14 statistical features derived from KDD Cup 99' dataset as well as 10 additional features which can be used for further analysis and evaluation of NIDSs. By using Kyoto 2006+ dataset, IDS researchers and operators are able to obtain more practical, useful and accurate evaluation results. Furthermore, we provide very detailed analysis results of honeypot data using five criteria (*i.e.*, SNS7160 IDS[6], ClamAV software[7], Ashula[5], source IP addresses and destination ports), and share our experiences so that security researchers are able to get insights into the trends of latest cyber attacks and the Internet situations.

Our key findings are:

- about 50% of cyber attacks were launched from China, United States and South Korea;
- the total number of unique IDS alerts, AV alerts, shellcodes, source IP addresses and destination ports was 290, 832, 231, 4,420,971 and 61,942, respectively;
- the average number of unique IDS alerts, AV alerts, shellcodes, source IP addresses and destination ports in each day was 41, 5.5, 9, 5,851 and 557, respectively;
- *MSSQL StackOverflow* (29%), *SMB Large Return Field* (17%) and *Too Many SYNs for a TCP Connection* (12%) occupied about 60% of the all IDS alerts;
- most of AV alerts were related with Trojan, Worm, Phishing and Email;
- only a single shellcode (ID 58) occupied about 88% of the all shellcodes, which is used for exploiting the vulnerability of *MS02-039*[13] or *CAN-2002-0649*[9] and its malware name is *MS-SQL Slammer*[14];
- top 6 destination ports (*i.e.*, 445, 22, 0, 80, 139, 1434) occupied about 70% of the all destination ports;
- 27 new shellcodes related with *Win32/Conficker.A* worm were detected during its development period (from Oct. 29th 2008 to Nov. 21st 2008).

The rest of the paper is organized as follows. In Section 2, we describe overview of honeypots used for obtaining real traffic data briefly. In Section 3, we show our honeypot data and present their analysis results in detail. In Section 4, we introduce Kyoto 2006+ dataset built by honeypot data. Finally, Section 5 gives some conclusion and future work.

2. Overview of honeypots

Table 1 shows the types of honeypots used for collecting real traffic data. In fact, we used many different types of real and virtual machines as our honeypots such as Windows machines (*e.g.*, Windows XP SP2, full patched Windows XP, Windows XP with no patch), Linux/Unix machines (*e.g.*, Solaris 8, MacOS X), dedicated honeypots introduced in [4], network printer, home appliance (*e.g.*, TV set, HDD Recorder) and so on. Also, we have deployed SGNET honeypots[15]. We have deployed these various types of honeypots on the 5 different networks which are inside and outside of Kyoto University: 1 class A and 4 class B networks. The total number of our honeypots are 348 including two black hole sensors with 318 unused IP addresses. Most of our honeypots are rebooted immediately after a malicious outgoing packet is observed. At the reboot, an image of HDD is overwritten by the original one, so that the honeypots return to the original condition. Some Windows based honeypots are, however, allowed to run for several weeks. Because we deploy in-line IPS between these honeypots and the Internet, all of detectable attacks to outside are blocked and we write custom signatures to detect exploit codes by using Ashula[5].

Table 1. Overview of honeypots

Type	Number of machines
Solaris 8 (Symantec based)	4
Windows XP (full patch)	1
Windows XP (no patch)	5
Windows XP SP2	2
Windows Vista	1
Windows 2000 Server	1
MacOS X	2 (one is mail server)
Printer	2
TV set	1
HDD recorder	1
dedicated honeypots[4]	5
SGNET honeypots[15]	4
Web Crawler	1
Balck hole sensor /24	1
Balck hole sensor /26	1

We have collected all traffic data to/from our honeypots, and thoroughly inspected them using three security softwares, SNS7160 IDS system[6], Clam AntiVirus software[7] and Ashula[5] which is a dedicated shellcode detection software, so that we can identify what happened on the networks. Currently we use only ClamAV and its detection patterns are being updated every hour. Also, since Apr. 1st 2010, we have deployed another IDS provided by Soucrefire, *i.e.*, snort. Before that, we only used SNS7160. The detailed analysis results of honeypot data using those security softwares are described in subsections 3.2, 3.3 and 3.4. On the other hand, since most of the honeypot traffic data are composed of attack data, we need to prepare normal traffic data in order to build evaluation dataset for IDS. For generating normal traffic data, we have deployed a server on the same network with the above honeypots, and the server has two main functions of mailing service and DNS server which services only one domain. The server was also operated with several communication protocols, *e.g.*, ssh, http and https, for its management. We regard all traffic data related to the server as normal data, because we observed that there is few attack data even if the server has received cyber attacks.

Table 2. Overall property of honeypot data

	Number of sessions	Average number of sessions per day
Total	93,076,270	93,638
Normal	50,033,015	50,335
Known attack	42,617,536	42,874
Unknown attack	425,719	428

3. Honeypot Data

3.1 Overall Property

In this section, we present the overall property of our honeypot data which were captured from Nov. 2006 to Aug. 2009, even if we could not capture honeypot data for 36 days due to the maintenance of our honeypot systems and sometimes the unmanageable events such as power failure. During the observation period, there were 50,033,015 normal sessions and 43,043,255 attack sessions as shown in Table 2. Note that we regarded all traffic data captured from our honeypots as attack data and regarded all traffic data of the mail and DNS server as normal data. Also, among the attack sessions, we observed that 425,719 sessions were related to unknown attacks, because they did not trigger any IDS alerts, but they contained shellcodes detected by Ashula. In other words, in case of known attacks, they have to be recorded by our IDS, and if a session contains shellcode(s), there is a high possibility that it is related to a real attack. From our investigation, we found that IDS sometimes failed to detect known attacks, *i.e.*, false negative, but ratio of the failure is negligibly small. Figure 1 shows the distributions of normal (blue bars), known attack (red bars) and unknown attack (deep red points) sessions in our honeypot data. In our investigation, we observed that the average number of normal sessions, known attack sessions and unknown attack sessions are 50,335, 42,874 and 428 per day, respectively. In particular, we observed that unknown attack were prevalent in two green areas in Figure 1.

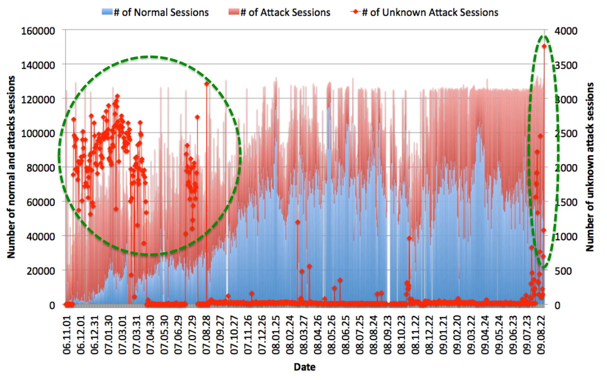


Figure 1. Distributions of normal, known attack and unknown attack sessions in honeypot data.

In order to investigate the geographical locations of source IP addresses, we extracted their country information with respect to attack sessions including unknown attack sessions. Figure 2 shows the national distributions of attack source IP addresses observed in our honeypots. From Figure 2, it is easily seen that the total number of attack countries is 228 and about 60% of the all attack sessions was caused by only four countries, *i.e.*, Japan, China, United States and South Korea. However, in our further investigation, we observed that most source IP addresses in Japan are located in black hole sensors and honeypots, and some of them were caused by Unicast Flooding of L2 Switches[8] which generates incomplete data of sessions and makes IDS raises false alarms. In campus network, several servers, routers and firewalls had been improperly configured. They generated so many amount of incorrect packets that arp table and forwarding table of L2/L3 switches are exhausted. During the situation, most of the switches work as repeaters. Even Cisco's L2 switches ignore VLAN and broadcast all packets to all ports. Therefore, our honeypots were forced to receive to such packets. Table 3 shows the locations of top 10 source IP addresses (we sanitized IP addresses due to the secrecy of communication) which are responsible for about 42% of all source IP addresses in Japan. Among the top 10 source IP addresses, only x.x.x.9 IP address was a real attack host. If we exclude no-attack source IP addresses in Japan, about 50% of cyber attacks were launched from China, United States and South Korea.

3.2 Statistical Analysis of IDS Detected Sessions

In this section, we describe the analysis results of honeypot data according to IDS alerts. Figure 3 shows statistical information of IDS detected sessions and the distribution of

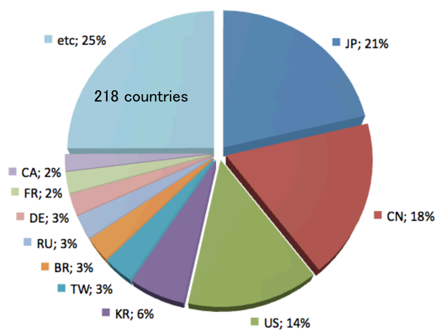
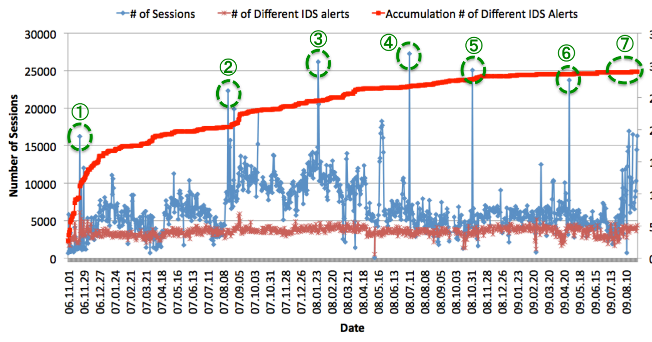
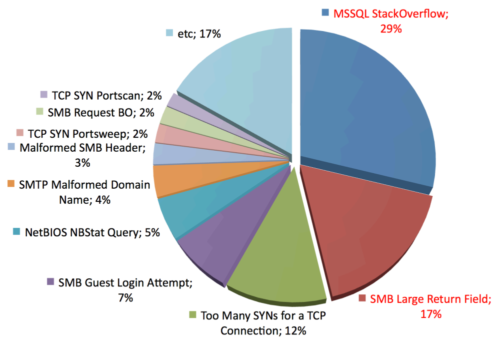


Figure 2. National distribution of attack source IP addresses.



(a) Statistics of IDS detected sessions



(b) Distribution of IDS alerts

Figure 3. Statistics of IDS detected sessions and distribution of IDS alerts.

Table 3. Locations of top 10 source IP addresses in Japan

IP address	Count	Location
x.x.x.1	1,932,303	Darknet
x.x.x.2	530,039	L2 Switch (Unicast Flooding)
x.x.x.3	377,599	Darknet
x.x.x.4	355,607	L2 Switch (Unicast Flooding)
x.x.x.5	170,182	Honeypot (Windows 2k)
x.x.x.6	131,115	Darknet
x.x.x.7	118,006	No honeypot (MacOS X)
x.x.x.8	105,832	Honeypot (Fedora Core)
x.x.x.9	100,824	No honeypot
x.x.x.10	92,509	Honeypot (Original WinXP)

IDS alerts. During the analysis, we first counted the number of sessions (*i.e.*, blue lines in Figure 3(a)) detected by SNS7176 IDS system, and we observed that among the all of 43,043,255 attack sessions, 6,650,335 sessions triggered IDS alerts. Also, the average number of IDS detected sessions in each day was 6,690. In particular, from Figure 3(a), we can see that the number of IDS detected sessions of only 6 days (*i.e.*, ① ∼ ⑥) is extremely larger than the other days. In our further investigation, as shown in Table 4, we recognized that there were two many P2P connection requests (①), SYN scanning activities for IPv4 IP addresses by a single host (②), SYN flooding attacks to a single spam mail server (③ ∼ ⑤) and backscatters from a single host (⑥).

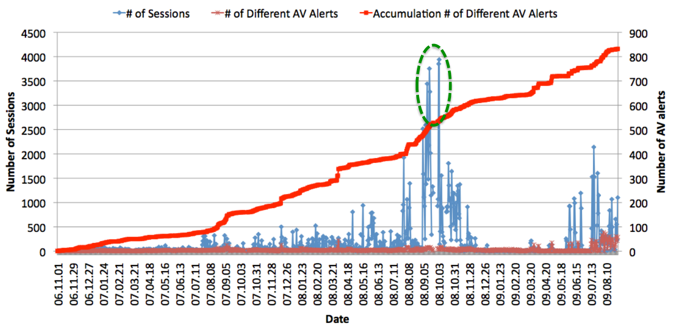
Secondly, we counted how many different types of IDS alerts were recorded in each day (*i.e.*, brown lines in Figure 3(a)), and the accumulation number of different IDS alerts during the observation period (*i.e.*, red lines in Figure 3(a)). From our analysis, we observed that there are 41 unique IDS alerts in each day on average and the total number of unique IDS alerts is 290. From Figure 3(a), we can see that the total number of unique IDS alerts converges to 300 (⑦). Since we enabled all IDS signatures basically and updated them periodically, it is natural to be increasing gradually. However, The reason why the total number of unique IDS alerts converges to 300 is that the updating support of IDS signatures was suspended in Dec. 2009. In fact, the number of updated IDS signatures was rapidly decreased from several months ago of Dec. 2009.

Table 4. IDS alerts observed during 6 days

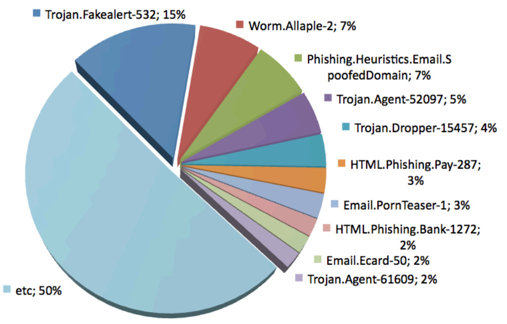
Date	Signature name	Count
①	P2P BitTorrent Activity	1,802
	P2P Edonkey Start Upload Request	2,867
	Too Many SYNs for a TCP Connection	3,011
	Emule File Traffic Detected	5,586
	P2P eMule Hello	5,369
	P2P Emule Kademlia Request	8,100
②	Too Many SYNs for a TCP Connection	1,341
	Out-of-Sequence TCP RST Packet	4,779
	Out-of-Sequence TCP SYN Packet	13,859
③	Too Many SYNs for a TCP Connection	13,364
	MS SQL Stack BO	4,685
④	Too Many SYNs for a TCP Connection	22,223
	MS SQL Stack BO	2,508
⑤	Too Many SYNs for a TCP Connection	21,285
	Unauthenticated OSPF	5,893
⑥	Repeated TCP SYN with Diff ISN and TTL	6,820
	MS SQL Stack BO	10,264

Table 5. Number of five malwares.

	Count	Count
Malware Name	(One month)	(Total)
Trojan.Fakealert-532	17,118	22,802
Trojan.Agent-52097	6,803	10,586
HTML.Phishing.Bank-1272	4,205	5,411
Trojan.Goldun-278	2,237	2,237
Trojan.Goldun-280	1,156	1,156

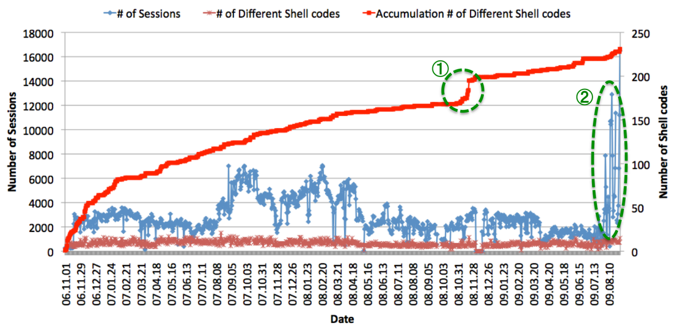


(a) Statistics of AV detected sessions

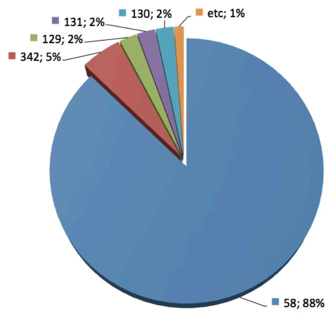


(b) Distribution of AV alerts

Figure 4. Statistics of AV detected sessions and distribution of AV alerts.



(a) Statistics of shellcode detected sessions



(b) Distribution of Shellcodes

Figure 5. Statistics of shellcode detected sessions and distribution of shellcodes.

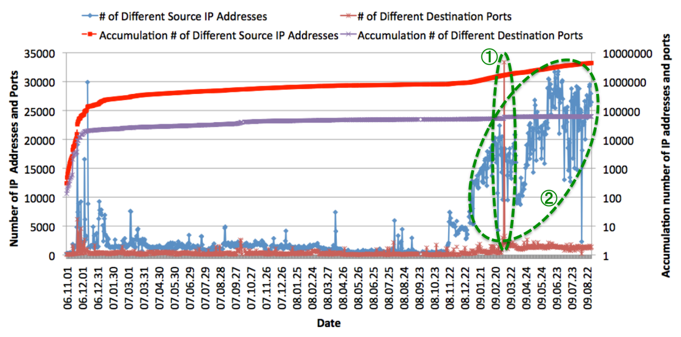
Finally, we examined the total number of each IDS alert and Figure 3(b) shows the distribution of IDS alerts. From Figure 3(b), we can see that top 3 IDS alerts (*i.e.*, *MSSQL StackOverflow*, *SMB Large Return Field* and *Too Many SYNs for a TCP Connection*) occupy about 60% of the all IDS alerts. In our further investigation, we observed that the first two IDS alerts are aiming to exploit the very old vulnerabilities (*i.e.*, *CAN-2002-0649*[9] and *CAN-2005-1206*[10]) of MSSQL and Windows SMB, respectively. Furthermore, they are still popular in 2011. However, it is unnatural to consider that attackers really tried to exploit these vulnerabilities, because in most cases, they will fail due to the oldness of them. Therefore, this situation could be said that attackers intentionally triggered these old IDS alerts before they try to attack their real targets, so that they can trick IDS operators. Because, if IDS operators observe a large number of these old IDS alerts caused by a certain host, they will regard its all IDS alerts as usual false positives, and consequently they will fail to recognize a real attack which was hidden in the stack of usual false positives. This attack scenario was also introduced in [11].

3.3 Statistical Analysis of AV Detected Sessions

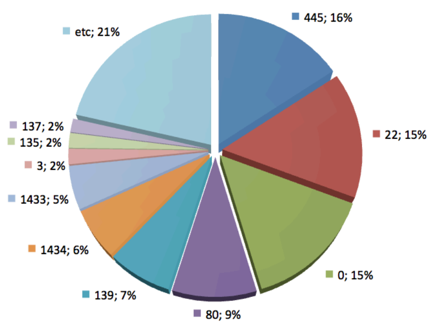
In this section, we describe the analysis results of honeypot data according to AV alerts. Figure 4 shows statistical information of AV detected sessions and the distribution of AV alerts. Figure 4(a) shows the number of sessions (*i.e.*, blue lines) detected by Clam AntiVirus software, the number of unique AV alerts in each day (*i.e.*, brown lines) and the accumulation number of unique AV alerts (*i.e.*, red lines). In our investigation, we observed that among the all of 43,043,255 attack sessions, 165,717 sessions triggered AV alerts and the average number of AV detected sessions in each day was 166. Also, there were 5.5 unique AV alerts in each day on average and the total number of unique AV alerts was 832.

From Figure 4(a), we can see that the number of AV detected sessions in the green area (*i.e.*, from Sep. 4th 2008 to Oct. 4th 2008) is extremely larger than the other days. Through our examination, we discovered that a large number of Trojan and Phishing attacks happened during this period. Specifically, 5 different types of malwares shown in Table 5 were detected by Clam AV software and most AV alerts related with them were concentrated on only this period.

We also counted the total number of each AV alert and Figure 4(b) shows the distribution of AV alerts. From Figure 4(b), we can see that top 10 AV alerts occupy about 50% of the all AV alerts. In addition, it is easily seen that most AV alerts are related to Trojan, Worm, Phishing and Email. The reason why there are many email related AV alerts is that we



(a) Statistics of Source IP addresses and Destination Ports



(b) Distribution of Destination Ports

Figure 6. Statistics of source IP addresses and destination ports, and distribution of destination ports.

deployed a mail server for generating normal traffic data as well as several honeypots for collecting spam emails.

3.4 Statistical Analysis of Shellcode Detected Sessions

Figure 5 shows statistical information of shellcode detected sessions and the distribution of shellcodes. Figure 5(a) shows the number of sessions (*i.e.*, blue lines) detected by Ashula, the number of unique shellcodes in each day (*i.e.*, brown lines) and the accumulation number of unique shellcodes (*i.e.*, red lines). In our investigation, we observed that among the all of 43,043,255 attack sessions, 2,818,133 sessions contained shellcodes and the average number of shellcode detected sessions in each day was 2,835. Also, there were 9 unique shellcodes in each day on average and the total number of unique shellcodes was 231.

From Figure 5(a), we can see that the accumulation number of unique shellcodes are rapidly increasing from Oct. 29th 2008 to Nov. 21st 2008 (①). This means that lots of new shellcodes were suddenly emerged during this period. As a result of our investigation, we discovered that it was caused by a famous malware, *Win32/Conficker* worm (also known as *Kido* and *Downadup*) which was aiming to exploit a new vulnerability of Windows OSes, *i.e.*, *MS08-067*[12]. In fact, the new vulnerability was published in Oct. 23rd 2008 for the first time and in our honeypots, we observed the first attack which contains a shellcode for exploiting the vulnerability in Oct. 29th 2008. Since the first observation, we observed 27 new types of shellcodes associated with *Win32/Conficker* worm until Nov. 21st 2008. On Nov. 21st, the first version of the worm got in the wild. In addition, it is easily seen that the number of shellcode detected sessions is dramatically increasing from Aug. 4th 2009 ((②). This is because we have deployed a part of our honeypots in that they can make a communication with *Win32/Conficker* worm. In other words, we could get high quality shellcodes which are sent from attackers only after session establishment.

Finally, we counted the total number of each shellcode and Figure 5(b) shows the distribution of shellcodes. From Figure 5(b), we can see that shellcode ID 58 occupies about 88% of the all shellcodes. Actually, this shellcode is used for exploiting the vulnerability of *MS02-039*[13] or *CAN-2002-0649*[9] and its malware name is *MS-SQL Slammer*[14].

3.5 Statistical Analysis of Source IP Addresses and Destination Ports

In this section, we present the analysis results of attack data according to source IP addresses and destination ports. Figure 6 shows statistical information of source IP addresses and destinations ports, and the distribution of destination ports. Figure 6(a) shows the number of unique source IP addresses (*i.e.*, blue lines) and unique destination ports (*i.e.*, brown lines) in each day, and the accumulation number of unique source IP addresses (*i.e.*, red lines) and unique destination ports (*i.e.*, purple lines). In our investigation, we observed that the total number of unique source IP addresses and unique destination ports is 4,420,971 and 61,942, respectively. Also, the average number of unique source IP addresses and unique destination ports in each day was 5,851 and 557 on average, respectively.

From Figure 6(a), we can see that the number of unique destination ports are tremendously large in Mar. 10th 2009 (①). In our investigation, we identified that an attacker compromised a single honeypot (Solaris) through ssh and he/she carried out UDP flooding attacks[1]. In fact, the number of

[1] All the attacks were blocked by our IDS and L3 filtering.

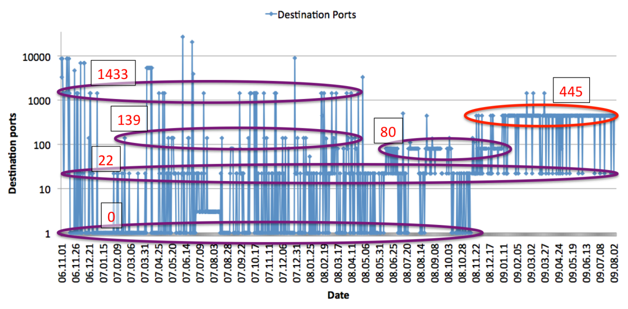


Figure 7. Trend of the most popular destination ports.

unique destination ports that this honeypot accessed in this day was 32,699 which is responsible for about 98% of the total unique destination ports, *i.e.*, 33,359. In addition, we have to give an attention to the other green area (②), because the number of unique source IP addresses has been growing constantly since Nov. 19th 2008. As mentioned in Section 3.4, *Win32/Conficker* worm has started its activity since Nov. 21 2008, and its subspecies such as *Conficker.B*, *Coficker.C* and *Conficker.D* also have been emerged and continued their activities. Based on this fact, it could be said that *Win32/Conficker* worms influenced the rapid increasing of unique source IP addresses since Nov. 19th 2008.

Finally, we counted the total number of each destination port and Figure 6(b) shows the distribution of destination ports. From Figure 6(b), we can see that top 6 destination ports (*i.e.*, 445, 22, 0, 80, 139 and 1434) occupy about 70% of the all destination ports. Among them, we could not identify the target of port 0. Furthermore, in order to understand the trends of these destination ports, we depicted the most popular destination port in each day as shown in Figure 7. From Figure 7, we can see that the popular destination ports are changing with time except 22 port which is abused for SSH dictionary attack.

4. Kyoto 2006+ Dataset

In this section, we describe Kyoto 2006+ dataset for evaluating performance of NIDSs. With respect to 93,076,270 sessions (50,033,015 normal sessions, 42,617,536 known attack sessions and 425,719 unknown attack sessions) of honeypot data introduced in Section 3, we extracted 24 features as described in Sections 4.1 and 4.2. Note that we used Bro[19] to convert raw traffic data into session data. Kyoto 2006+ dataset is open to the public at [17].

4.1 Extracting 14 Statistical Features

Based on the 41 original features of KDD Cup 99 data set, we extracted the following 14 significant and essential features from our honeypot data. The reason why we extracted the 14 statistical features is that among the original 41 features of the KDD Cup 99 dataset[3] there exist substantially redundant and insignificant features. In addition, we excluded contents features such as 'num_file_creations' (number of file creation operations) and 'num_access_files' (number of operations on access control files), because they are not suitable for network based intrusion detection systems and it is time-consuming or impossible to extract them without domain knowledge. The fourteen features consist of the following 13 continuous features and one categorical feature (*i.e.*, "flag").

1. **Duration**: the length (seconds) of the connection
2. **Service**: the connection's service type, e.g., http, telnet.
3. **Source bytes**: the number of data bytes sent by the source IP address
4. **Destination bytes**: the number of data bytes sent by the destination IP address
5. **Count**: the number of connections whose source IP address and destination IP address are the same to those of the current connection in the past two seconds.
6. **Same_srv_rate**: % of connections to the same service in Count feature
7. **Serror_rate**: % of connections that have "SYN" errors in Count feature
8. **Srv_serror_rate**: % of connections that have "SYN" errors in Srv_count(the number of connections whose service type is the same to that of the current connection in the past two seconds) feature
9. **Dst_host_count**: among the past 100 connections whose destination IP address is the same to that of the current connection, the number of connections whose source IP address is also the same to that of the current connection.
10. **Dst_host_srv_count**: among the past 100 connections whose destination IP address is the same to that of the current connection, the number of connections whose service type is also the same to that of the current connection
11. **Dst_host_same_src_port_rate**: % of connections whose source port is the same to that of the current connection in Dst_host_count feature
12. **Dst_host_serror_rate**: % of connections that have "SYN" errors in Dst_host _count feature
13. **Dst_host_srv_serror_rate**: % of connections that "SYN" errors in Dst_host_ srv_count feature
14. **Flag**: the state of the connection at the time the connection was written.

4.2 Extracting 10 Additional Features

Addition to the above 14 statistical features, we have also extracted additional 10 features which may enable us to investigate more effectively what kinds of attacks happened on our networks. They also can be utilized for IDS evaluation with the14 conventional features, and users are able to extract more features using the additional 10 features.

1. **IDS_detection**: reflects if IDS triggered an alert for the connection; '0' means any alerts were not triggered, and an arabic numeral means the different kinds of the alerts. Parenthesis indicates the number of the same alert.
2. **Malware_detection**: indicates if malware, also known as malicious software, was observed in the connection; '0' means no malware was observed, and a string indicates the corresponding malware observed at the connection. Parenthesis indicates the number of the same malware.
3. **Ashula_detection**: means if shellcodes and exploit codes were used in the connection; '0' means no shellcode nor

exploit code was observed, and an arabic numeral means the different kinds of the shellcodes or exploit codes. Parenthesis indicates the number of the same shellcode or exploit code.

4. **Label**: indicates whether the session was attack or not; '1' means the session was normal, '-1' means known attack was observed in the session, and '-2' means unknown attack was observed in the session.
5. **Source_IP_Address**: means the source IP address used in the session. The original IP address on IPv4 was sanitized to one of the Unique Local IPv6 Unicast Addresses[16]. Also, the same private IP addresses are only valid in the same month: if two private IP addresses are the same within the same month, it means their IP addresses on IPv4 were also the same, otherwise they are different.
6. **Source_Port_Number**: indicates the source port number used in the session.
7. **Destination_IP_Address**: it was also sanitized.
8. **Destination_Port_Number**: indicates the destination port number used in the session.
9. **Start_Time**: indicates when the session was started.
10. **Duration**: indicates how long the session was being established.

5. Conclusion and Future Work

In this paper, we have proposed Kyoto 2006+ dataset built on 3 years of honeypot data. It consists of 14 statistical features derived from KDD Cup 99' dataset as well as 10 additional features which can be used for further analysis and evaluation of NIDSs. By using Kyoto 2006+ dataset, IDS researchers are able to obtain more practical, useful and accurate evaluation results.

Furthermore, in order to give the recent trends of the cyber attacks to security researchers and share our useful experiences, we first have provided a lot of statistical information regarding the latest cyber attacks observed in our honeypots: national distributions of attack hosts, the number and distributions of unique IDS alerts, AV alerts, shellcodes, source IP addresses and destination ports. Secondly, from the distributions of IDS alerts, we can learn that it is very important to constantly trace usual and old false positives in that dangerous and real attacks are hidden. Thirdly, the result that 27 new shellcodes related with *Win32/Conficker.A* worm were detected during its development period shows that we need to chase activities of a new malware as soon as possible when it is emerged. Fourthly, our honeypots have succeeded in observing the new types of IDS alerts, AV alerts and shellcodes incessantly. As a result, these anslysis results demonstrate that our long-term (*i.e.*, 3 years) observations based on various types of honeypots are extremely effective to better understand the major trends and characteristics of recent cyber threats and to devise countermeasures against them.

In our future work, we will collect more benign traffic from different servers and network environments, extract files (*e.g.*, exe, dll, jpg) from each session of honeypot data and inspect them using various security softwares. Currently we continue to collect honeypot data. Because of discontinue of Symantec IDS, we added new IDS from Sourcefire[18]. Finally, we will consider to add sanitized packet headers of honeypot data to Kyoto 2006+ dataset.

This work was partially supported by Strategic Information and Communications R&D Promotion Programme (091603006).

References

[1] Reza Sadoddin and Ali A. Ghorbani, *"A Comparative Study of Unsupervised Machine Learning and Data Mining techniques for Intrusion Detection"*, *MLDM 2007*, pp. 404–418, 2007.

[2] Jungsuk Song, Hiroki Takakura, Yasuo Okabe, Daisuke Inoue, Masashi Eto, Koji Nakao, *"A Comparative Study of Unsupervised Anomaly Detection Techniques Using Honeypot Data"*, *IEICE Transactions on Information and Systems*, Vol.E93-D, No.9, pp.2544-2554, Sep. 2010.

[3] The third international knowledge discovery and data mining tools competition dataset KDD99-Cup http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 1999.

[4] Jungsuk Song, Hiroki Takakura and Yasuo Okabe, *"Cooperation of intelligent honeypots to detect unknown malicious codes"*, *WISTDCS 2008*, IEEE CS Press, pp. 31-39, 2008.

[5] http://www.secure-ware.com/contents/product/ashula. html

[6] Symantec Network Security 7100 Series.

[7] http://www.clamav.net/

[8] http://www.cisco.com/en/US/products/hw/switches/ps700/products_tech_note09186a00801d0808.shtml

[9] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2002-0649

[10] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2005-1206

[11] Jungsuk Song, Hayato Ohba, Hiroki Takakura, Yasuo Okabe, Kenji Ohira and Yongjin Kwon, *"A Comprehensive Approach to Detect Unknown Attacks via Intrusion Detection Alerts"*, *ASIAN2007*, LNCS 4846, pp. 247-253, Dec. 2007.

[12] http://www.microsoft.com/technet/security/bulletin/ms08-067.mspx

[13] http://www.microsoft.com/technet/security/bulletin/ms02-039.mspx

[14] http://www.sans.org/security-resources/malwarefaq/ms-sql-exploit.php

[15] http://wombat-project.eu/WP3/FP7-ICT-216026-Wombat_WP3_D13_V01-Sensor-deployment.pdf

[16] RFC4193:http://www.ietf.org/rfc/rfc4193.txt

[17] http://www.takakura.com/Kyoto_data/

[18] http://www.sourcefire.com/

[19] http://www.bro-ids.org/

nicter : A Large-Scale Network Incident Analysis System

Case Studies for Understanding Threat Landscape

Masashi ETO
National Institute of Information and Communications Technology
eto@nict.go.jp

Daisuke INOUE
National Institute of Information and Communications Technology
dai@nict.go.jp

Jungsuk SONG
National Institute of Information and Communications Technology
song@nict.go.jp

Junji NAKAZATO
National Institute of Information and Communications Technology
nakazato@nict.go.jp

Kazuhiro OHTAKA
National Institute of Information and Communications Technology
ohtaka@nict.go.jp

Koji NAKAO
National Institute of Information and Communications Technology
ko-nakao@nict.go.jp

Abstract

We have been developing the Network Incident analysis Center for Tactical Emergency Response (nicter), whose objective is to detect and identify propagating malwares. The nicter mainly monitors darknet, a set of unused IP addresses, to observe global trends of network threats, while it captures and analyzes malware executables. By correlating the network threats with analysis results of malware, the nicter identifies the root causes (malwares) of the detected network threats. Through a long-term operation of the nicter for more than five years, we have achieved some key findings that would help us to understand the intentions of attackers and the comprehensive threat landscape of the Internet. With a focus on a well-knwon malware, i.e., W32.Downadup, this paper provides some practical case studies with considerations and consequently we could obtain a threat landscape that more than 60% of attacking hosts observed in our darknet could be infected by W32.Downadup. As an evaluation, we confirmed that the result of the correlation analysis was correct in a rate of 86.18%.

Keywords network monitoring, malware analysis, correlation analysis

1. Introduction

The recent outbreak of the W32.Downadup worm shows that the worm problem remains relevant and requires further analysis. As countermeasures against malwares especially related to zero-day attacks, practical solutions should be effectively developed in an urgent manner.

In order to fight against threats especially induced by malwares, we have been developing and researching Network Incident Analysis Center for Tactical Emergency Response (nicter) [1–3]. The nicter mainly monitors darknet, a set of unused IP addresses, to observe global trends of network threats, while it captures and analyzes malware executables. The nicter realizes a practical implementation of Macro-Micro Correlation Analysis, in which the global observations in a macroscopic view and malware analysis in a microscopic view are correlated to bind the observed attacks (mainly scans) with their possible root causes, namely malwares based on the fundamental propagation steps of malwares such as scan $\rightarrow$ exploit code $\rightarrow$ malware download.

This paper presents how the nicter collects and stores numerous amount of data such as network traffic, malware samples and even analysis results, in order to provide them to the various analysis engines. With some practical case studies on practical data, the experimental results are presented, that indicate that there are still many remaining hosts infected by W32.Downadup.

2. Related Work

Various commercial, academic, or government-backed projects are ongoing to research and develop the countermeasure technologies [4–6] against malicious activities observed in the global Internet.

Many of these projects are concentrating on events analysis providing statistical data, such as rapid increase of accesses on certain port numbers, by using network events monitoring. Particularly, it is getting popular and easier to monitor a dark address space, which is a set of globally an-

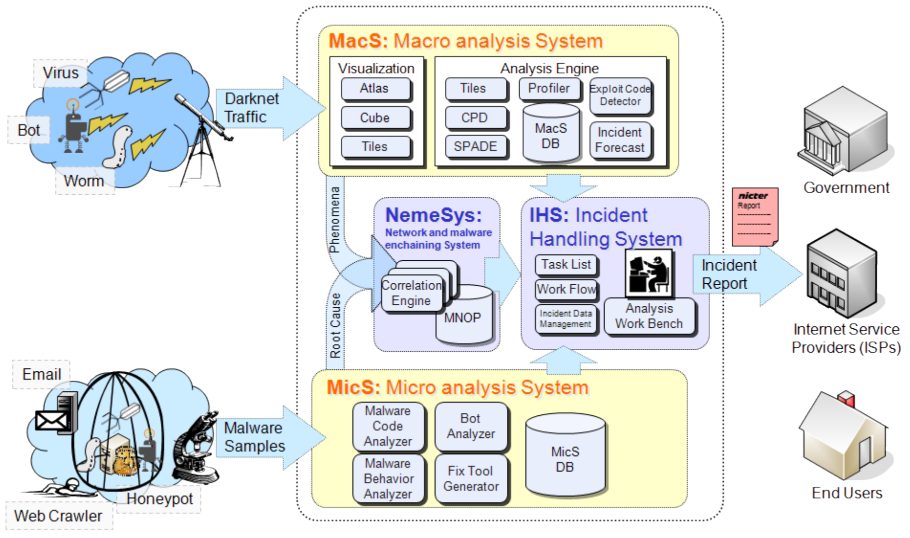


Figure 1. Overview of nicter

nounced unused IP addresses [4, 7, 8]. One can set up honeypots [9–12] on these addresses to masquerade as vulnerable hosts in order to monitor and record the malicious activities or listen quietly (black hole monitoring) to the incoming packets, which often contain great amount of malware scans, DDoS backscatter, etc. This paper calls these global observations over the Internet in a macroscopic view 'Macro Analysis'. That is, Macro Analysis can be applied to efficiently grasp the macroscopic behaviors (such as global scans) which are the first stage of malware activities over the Internet. However, since it is based on 'events (scans) observations' in the macroscopic level and is performed without any explicit information regarding the attacker's behavior, its results often leave certain level of uncertainty on the attack caused by the malware.

On the other hand, apart from the macroscopic view, analyzing an actual malware executable has been another challenge. Reverse engineering techniques are applied to disassemble a malware executable in order for the analyst to understand its structure [13, 14]. Also, sandbox analysis, in which a malware code is actually executed in closed (or access-controlled) experimental environment, is capable to observe its behavior [14–16]. We call these direct malware analyses in a microscopic view 'Micro Analysis'. Micro Analysis reveals detailed structures and behaviors of malwares although it does not provide any information on their activities in real networks simply because it is performed in the closed experimental environment.

Even though the above Macro Analysis and Micro Analysis have been studied and deployed in various analysis systems, the knowledge obtained from these activities has not been effectively and efficiently *linked*, which is making the identification of the root causes of security incidents more difficult. Therefore, it is important to achieve the link between Macro and Micro Analysis in real time, that will provide a strong countermeasure against threats such as an outbreak of new malware, a stealthy activity of botnet and a new type of attack on unknown vulnerability, etc.

3. Overview of nicter

The nicter is composed of four main systems as depicted in Fig. 1 namely; the Macro analysis System (MacS), the Micro analysis System (MicS), the Network and malware enchaining System (NemeSys), and the Incident Handling System (IHS).

The MacS uses distributed sensors to monitor darknets deployed in several universities and corporations. A darknet is a set of globally announced unused IP addresses and using it is a good way to monitor network attacks such as malware's scans. Since there is no legitimate host using these addresses, and we can consider all incoming traffic as a consequence of some kind of malicious activities (or that of a misconfiguration.) All incoming traffic is input to analysis engines to detect incident candidates such as detection of new scan patterns or sudden increase of scans. We call the monitoring method that quietly monitors incoming packets of a darknet black hole monitoring. Meanwhile, the MicS captures malwares in the wild by utilizing various types of captures such as honeypots, dummy email accounts, and a web crawler. Captured malware executables are fed into a malware behavior analyzer and a malware code analyzer to extract their characteristics and behaviors. Analysis results are stored in a database called Malware kNOwledge Pool (MNOP). The NemeSys enchains the phenomena, i.e., incident candidates, and their root causes, i.e., malwares. Once it has been given an attacking host observed in the MacS, the correlation analyzer in the NemeSys outputs a list of malwares that have similar network behavior (i.e., scans) with the host. By finding the root causes of the observed network attacks, we can grasp a much clearer view of what is happening in the Internet. Finally, the IHS helps the operator to diagnose the results from the above analyses and make an incident report.

3.1 Macro Analysis System (MacS)

The MacS consists of widely distributed sensors and various visualization and analysis engines. The sensors monitor the network traffic and detect security events to be sent to the analysis engines. The analysis engines receive and analyze the security events from the sensors. The analysis results are a collective set of attributes such as sensor ID, analyzer ID, timestamp, and other analyzer-specific attributes for correlation of the analysis results.

We presently have several /16 and /24 darknets for observations, in which we are deploying wide range of black hole sensors that only listen to the incoming packets, a number of sensors that respond to certain incoming packets such as TCP SYN packet and ICMP echo request as low interaction sensors. The latter sensors are often configured to disguise themselves as systems with unfixed vulnerabilities to attract attacks, namely they are deployed as the honeypots.

3.1.1 Data Collection and Storage System

Databus Architecture For effective incident response, it is important to detect suspicious events and derive adequate countermeasures as soon as possible. In order to perform a such realtime analysis, the nicter has a unique data delivery system (*the Databus*) that employs the IP multicast protocol to feed traffic data into various analysis engines at one time. Fig. 2 depicts the overview of the databus architecture.

At the sensor module, captured packets are summarized one by one, in a specific format that consists of TCP/IP headers and meta data (time stamp, packet length, etc.,). The summarized packets are sent to a gate module deployed at the entry of the analysis center via an VPN session.

Finally, the collected packets are encapsulated into an UDP packet and sent to an isolated multicast segment, in which various analysis engines and databases are deployed. This architecture makes the analysis engines and databases scalable so that we can easily add new analysis engines and databases when their computational resources are not enough.

MacS DB In contrast to the realtime analysis engines, some engines conduct events analysis based on long-term accumulated traffic data. In order to store all of incoming packets and provide the data to those analysis engines, we developed a fully customized MySQL based packet database (MacS DB) that stores all of packets individually. We depict the architecture of the MacS DB in Fig. 3.

The MacS DB employs the master/slave architecture that provides a functionality of data replications from a master to slaves so that tasks of data accumulation and handling SQL queries from analysis engines can be divided between the master and slaves. Namely, a master can dedicate itself on capturing packets and replicating them to slaves, while slaves concentrate on providing required data to analysis engines. Additionally, on slaves, we deployed index fields of databases on RAM disks in order to improve their performance. Moreover, all queries from analysis engines are firstly processed by load balancer in order to distribute them to slaves whose computational loads are lower than the others. Consequently, the MacS DB shows a performance of data insertion of more than 65,000 packets per second (pps). Since the number of packets collected in the nicter is approximately 32,000 pps at a maximum, the performance of the MacS is efficient enough. Note that even if the number of incoming packets become larger than the capacity of the MacS DB system, we can easily scale up the performance of the MacS DB by adding a master or a slave in this architecture.

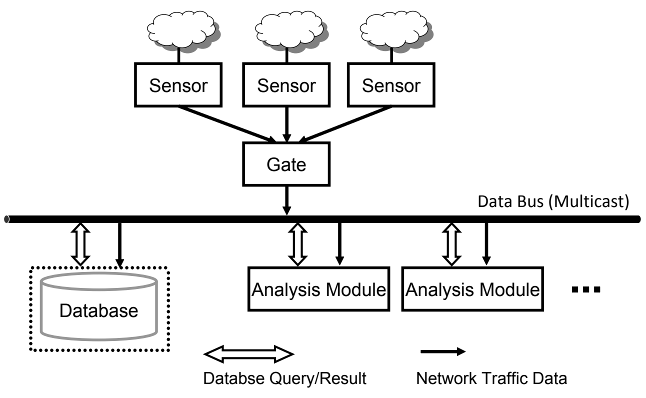


Figure 2. Databus Architecture

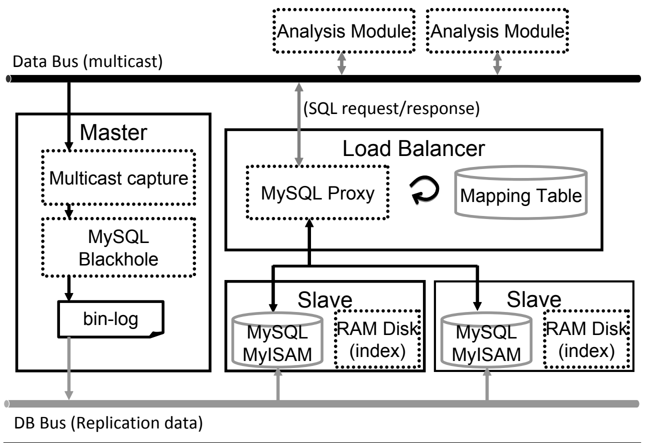


Figure 3. Architecture of MacS DB

3.2 Micro Analysis System (MicS)

The purpose of the Micro Analysis is to conduct automated in-depth examinations of malwares in order to grasp their characteristics and activities. As mentioned in 3.1, the nicter has several honeypots to collect malware executables in the wild. These collected executables are input into the MicS. Consequently, analysis results are stored in the MNOP.

A single set of analysis engine can handle 150 to 250 malware samples per day: a rate of approximately six to nine minutes to analyze one malware sample. Presently, the MicS has several sets of analysis engines in parallel; consequently it can analyze more than a thousand malware samples per day.

There mainly exist two approaches in malware analysis: static analysis and dynamic analysis. The MicS deploys two analysis engines: the code analyzer, which is based on the static analysis, and behavior analyzer, which is based on the dynamic analysis.

3.2.1 Collection of Malware Samples

In order to collect malware samples, we firstly deployed several different types of honeypots, namely, high and low interaction honeypots.

A type of the high interaction honeypots we deployed is developed on a real physical computer as a normal client computer. One of the purpose of this honeypot is to observe interactions between bots on the honeypot and command and control (C&C) servers. Therefore, activities of the honeypot are observed by a human operator and all incoming/outgoing traffic is captured by an intermediate node. In front of the high-interaction honeypot, we also deployed an IDS that monitors outgoing traffic from the victim machine to trap malicious activities other than C&C messages so that we can prevent secondary infections of any legitimate hosts in the Internet. Once a malicious activity is trapped by the IDS, the high interaction honeypot is rebooted immediately for the recovery of the disk image. During a rebooting process, executable files assumed as malware are extracted by comparing the infected hard disk and an original image (i.e., an unpolluted snapshot).

3.3 Macro-Micro Correlation Analysis

NemeSys correlates the results from the MacS and MicS to identify the observed attacks in more accurate level. The NemeSys is based on an approach called network behavior profiling, in which network behaviors of captured malwares in the MicS and attacking hosts observed by the darknet in the MacS are summarized into profiles for fast and diverse correlation.

Malware kNOwledge Pool (MNOP) The purpose of the MNOP is to conduct correlation analysis efficiently by storing all of summary data observed by sensors and honeypots, and analysis results of the MacS and MicS. Before the MNOP was deployed, we had to manually explore these data that were distributed in many databases and log files. The architecture of the MNOP is illustrated in Fig. 4. Specifically, in the MacS, traffic data collected by black hole sensors and honeypots are fed to various analysis engines such as the Basic Profiler, the Tiles, the exploit code (shellcode) detector [17], etc., and the analysis results are stored in the MNOP with the attacker's IP address. On the other hand, a malware sample collected by a honeypot is firstly analyzed by the MicS, then its network activity data (i.e., packets collected in the behavior analyzer) is stored into the MNOP through the MacS engines in the same manner with the MacS. We conduct the MacS analysis for the data of the MicS so that we can link both analysis results that are summarized in the same format. We note that even if we could not derive network activity data of a malware sample in the MicS, we have another way to supplement it by extracting from the log of honeypot when the sample was captured.

Finally, in order to link the analysis results in the MNOP, the NemeSys enchains the result of each analysis engines, that is realized by issuing database queries to the MNOP. Basically, the correlation analysis is performed based on the similarity of results of each analysis engine between the MacS and MicS. As a result, candidates of malwares infecting the attacking host are reported by the NemeSys. The method of the correlation analysis is detailed in [2] and [17].

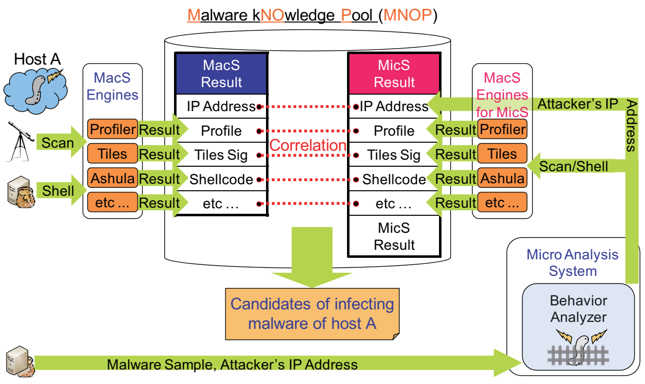


Figure 4. Architecture of MNOP

4. Landscape of Network Attacks

Through a long-term operation of the nicter for more than five years, we have found the comprehensive attack trends of the Internet that give us some key findings that indicate the importance of the large-scale network monitoring. The advantage of the nicter is that we have a large number of darknet IP addresses that are distributed in wide area in terms of position of network address and geographical location. Consequently, by comparing attack trends among sensors, we can promptly grasp many types of propagation strategies that reflect characteristics of each malware, and the global trends such as outbreaks of emerging malwares. In this section, we introduce some practical case studies of the observation that are derived by macro, micro and correlation analysis. Especially, we focus on W32.Downadup that has strong infectability and induced a serious symptom of a pandemic in Oct, 2008.

4.1 Study of W32.Downadup

W32.Downadup is equipped with multiple routes of infection such as a global network, a local network and removable media. A computer infected by W32.Downadup scans on 445/TCP of multiple global IP addresses in the Internet and then tries to exploit the vulnerability of Windows Server Service (MS08-067 [18]). In order to avoid being detected, the number of scan packets per unit time is automatically limited according to the condition of the computer. It also has capabilities to infect via the Windows Network and removable media such as USB memory. W32.Downadup computes domain names using a time-seeded random domain name generator and attempts to resolve these addresses, then downloads an update file of itself, so that it can autonomously update itself without rendezvous point. Additionally, W32.Downadup.C and latter variants construct a P2P network for updating itself. The bootstrap IP address and port number are also generated using a time-seeded information, therefore it does not require any static rendezvous point. Thus, W32.Downadup induced a pandemic because of its strong infectability described above as a result.

4.2 Detail of Darknet Sensors

As we mentioned in Sect 3.1, we have several sensors that monitor various types of darknet such as /24, /16 of network address, and totally we are monitoring more than 140,000 darknet IP addresses. In the following sections, we introduce several case studies observed in four main sensors that are deployed in geographically distributed areas in Japan (Fig. 5).

The sensor I is monitoring /24 unused IP addresses (i.e., darknet) allocated from a bunch of /16 used IP address range in which client and server computers are deployed (i.e., livenet). The two /16 networks monitored by sensor III and IV, consist of some darknet areas and used areas in a same fashion with the sensor I. In contrast, the sensor II is monitoring /16 darknet IP addresses that are fully unused. In terms of the location of network address, sensor II and IV belong to same /8 network while networks of sensor I and III belong to different /8 networks each other.

4.3 Case Studies

Outbreak of W32.Downadup Fig. 6 shows the moving average (duration : 10 days) of the number of hosts observed on each sensor from 2005 to 2011. In Sep, 2008, we observed a rapid increase of the number of hosts that send at least one TCP packet to our darknet sensors. At that time, the number of hosts increased approximately eleven times of the previous period, although the number was decreasing until that time. As a further analysis, we found that hosts

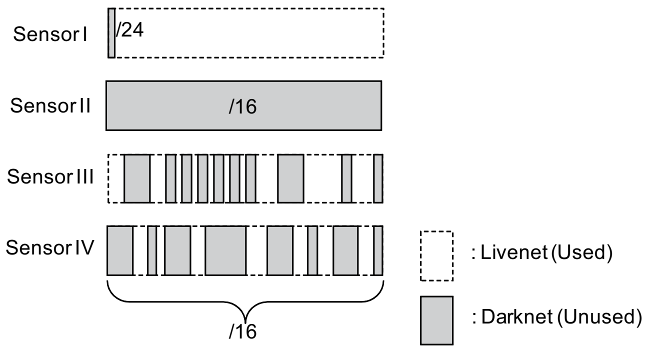


Figure 5. Detail of Darknet Sensors

that access to port 445 of TCP (Fig. 7) are dominant in the hosts in Fig. 6. This port is widely used for the server service of Windows OS family, while critical vulnerability has been discovered frequently.

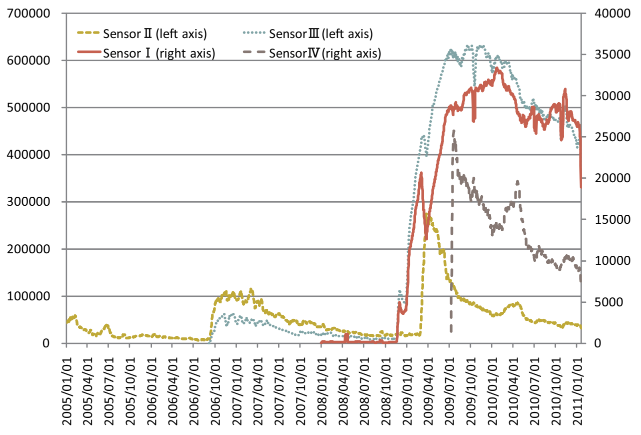


Figure 6. Number of Hosts on TCP

Triggered by this report, we conducted a further analysis in order to reveal the malware that induced this phenomenon. At first, as shown in Fig. 8, we profiled the scan behavior of one of attacking hosts which is illustrated by the Tiles [17]. Its scan behavior is translated into a macro profile as follows.

```
Protocol: TCP
TCP flag: SYN
Destination port: Single (445)
Source port: Multiple (2)
Destination IP Address: Multiple (4 addresses)
Scan type: Network scan
Number of packets: 4 packets (4 packets/30 seconds)
```

This host sent two scan packets to an IP address and scanned 445/TCP of two IP addresses during 30 seconds. In our further investigation, we discovered that there exist a tremendous number of attack hosts whose scan behavior is the similar to that of this host. With this profile, the Nemesys, the correlation analysis system explored the mal-

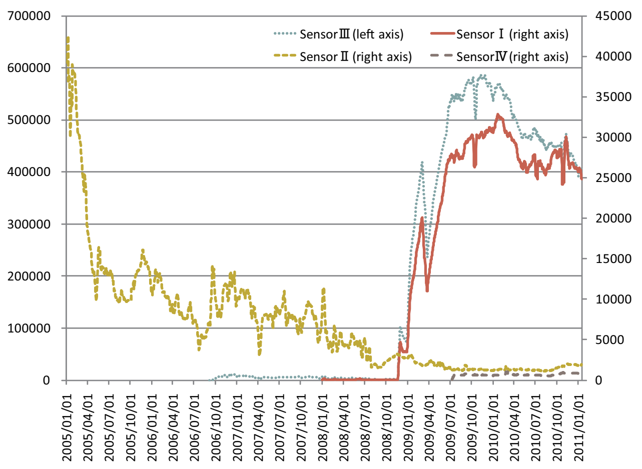


Figure 7. Number of Hosts on 445/TCP

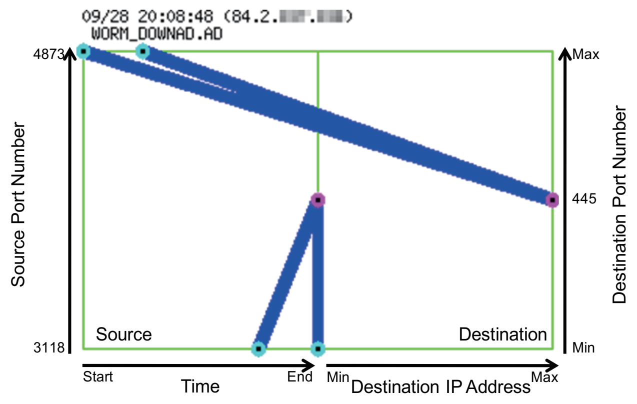


Figure 8. Scan Behavior Illustrated by Tiles

ware samples, which were captured by the honeypots around the same time of the attacks and have similar scan behavior with the attackers. As a result, the Nemesys could identified the emerging malware (i.e., W32.Downadup) that induced the phenomenon.

As an interesting fact, the number of hosts in the sensor II did not increase at the period of Sep, 2008. We assume that this is because the original specimen of W32.Downadup scans only neighboring IP addresses, namely, it scans only a class C (/24) block where the infected computer belongs and the previous ten /24 blocks [19]. As we mentioned before, there were no any computers in the /16 network observed by sensor II, therefore this darknet was not targeted by W32.Downadup. Indeed, we can confirm that the sensor II in Fig. 7 does not indicate any increase of hosts to 445/TCP at all.

Emergence of Variant of W32.Downadup On March 2009, four months after the emergence of the original W32.Downadup, the sensor II observed increase of the number of hosts (in Fig. 6) that scan random port numbers of TCP. Most of these hosts also sent multiple UDP packets although that is not illustrated in Fig. 6. According to the further analysis result of the Nemesys, this event was assumed to be caused by an emergence of a variant of W32.Downadup (i.e., W32.Downadup.C) which uses various port numbers for rendezvous of P2P connections with other infected hosts.

These two cases indicate that we can observe various types of events that depend on characteristics of darknet, namely a pure darknet such as the sensor II or a darknet neighboring livenet such as the other sensors.

Attacks from Botnets The diversity of IP address range of observed darknet is an important factor of global trend analysis. As we mentioned in Sect. 4.2, the nicter is monitoring several darknet segments that belong to different /8 networks.

Fig. 9 depicts a typical event in that each sensor shows different characteristics according to the differences of network address. In February, 2010, we observed a short-term spike of the number of hosts that scan 139/TCP. Since their scanning behavior is identical and their activities were observed almost simultaneously, we determined this event was induced by a botnet. As an interesting fact, this event was observed by only the sensor II and IV that belong to the same /8 network. From this fact, we can assume they were controlled to scan a specific /8 network address where sensor II and IV were monitoring. Indeed, most of the source IP addresses of attacking hosts in sensor II were same as the one in sensor IV.

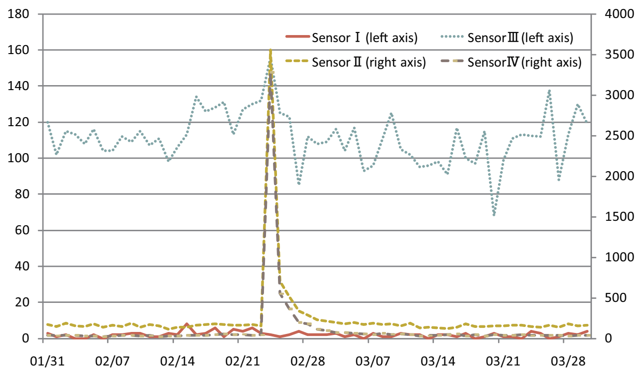


Figure 9. Attacks from Botnets to 139/TCP observed in 2010

A Global Trend If the all sensors observe the same characteristics of scans, we can assume that the event is a global trend as mentioned in [20]. In Sep, 2010, we found a rapid increase of scan packets to 5060/UDP observed by all the sensors as shown in Fig. 10. We could assume that this attack has arisen all over the world, indeed, from this time, the attack to SIP server increased in the world and emerged as one of the social issues. These scans tried to find SIP servers and intrude into them by the brute-force attack of accounts of the servers. Although the attacks are decreasing from the first spike, they are still continuing until today.

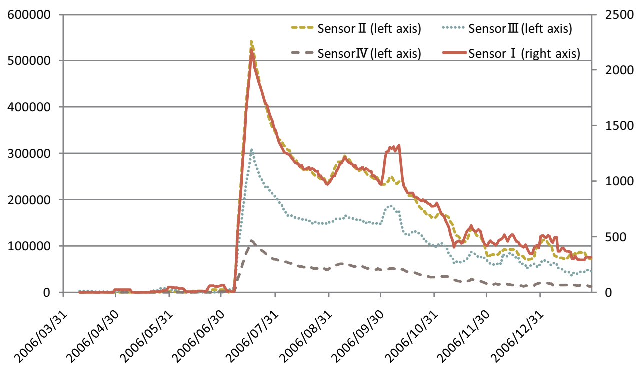


Figure 10. Number of Packets on 5060/UDP

Micro Analysis We have collected more than 1,600,000 malware samples that have unique hash values of the file of themselves by several honeypots, the web crawler and spams as mentioned in Sect. 3.2.1. Fig.11 shows the distribution of the top 10 malware samples captured by one of our high interaction honeypot and the other one of low interaction honeypot during one month of Oct, 2010. Note that we show the samples that could be named by Symantec's antivirus software. In the month, the honeypots captured 253 samples in total, in which the most common malware was W32.Downadup.B (104 samples) and the second was W32.Virut.W (46 samples). Although these malwares are old one found in 2008 and 2007 respectively, this result shows us that there are many old malwares that are still active as background noises. Therefore, it is important for us to remove influence of these old malwares so that we can detect emerging malwares efficiently.

In order to realize the demand, we should grasp the characteristics and activities of malwares. In the nicter's micro analysis system, we can automatically derive behavior and characteristics of malware samples. As an example, we analyzed W32.Downadup.B in the micro analysis system. As a result, we could reveal detail activities of W32.Downadup.B such as creation, modification, falsification and deletion of mutex, files, registries and so on. We could also recognize that the malware tries to propagate via USB memories from the analysis result. As the network activities, the sample tried to access various web servers such as Google for the confirmation of network environment, and the other that are assumed as C&C servers. The detail of analysis result of W32.Downadup.B is shown in the appendix.

Correlation Analysis Finally, we conducted an inspection of the global trend in 445/TCP again by use of the pro-

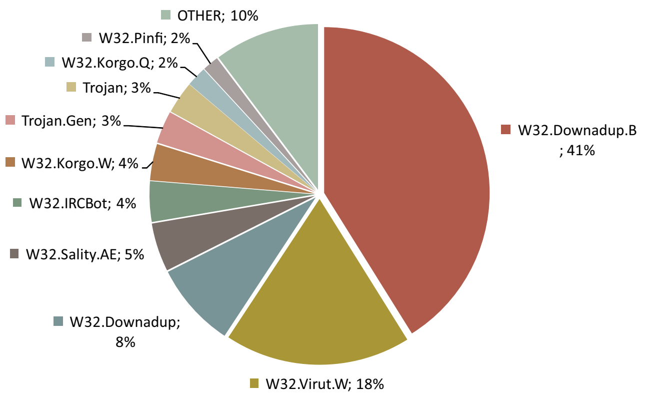


Figure 11. Distribution of Malware Samples in Oct, 2010

file of W32.Downadup derived in the previous case in order to reveal the distribution of attacking hosts that have similar scan behavior to W32.Downadup. In Nov 28th, 2010, the two /16 black hole sensors observed 662,902 attacking hosts in total. The Nemesys made macro profiles of the 662,902 hosts and automatically conducted the correlation analysis between the macro profiles and the micro profiles of samples of W32.Downadup. As a result, 398,780 out of 662,902 hosts (60.15%) had the similar scan behavior with the W32.Downadup. As a evaluation of this result, we confirmed the precision and recall of the cluster of 398,780 hosts determined as W32.Downadup.

As premises for the confirmation, note that when a malware sample is captured by a honeypot, the attacker is assumed to be also infected by the same malware with a high probability. Therefore, we extracted hosts who injected any malwares into our honeypot (i.e., the hosts were surely infected by the injected malwares), from the 398,780 hosts. Where the number of hosts that injected any malwares to our honeypots was 427, we confirmed the number of hosts infected by W32.Downadup in the cluster was 368. This means the precision of the correlation analysis was 86.18% ($368/427 * 100$). Furthermore, we confirmed that most of hosts randomly sampled from the 398,780 hosts had the same scan pattern with W32.Downadup as illustrated in Fig. 8. Thus, we can conclude that the nicter can recognize malwares that are probably infecting attacking hosts by the correlation analysis. By expanding this methodology through many types of scan behavior observed in the Internet, we will be able to effectively clarify the distribution of the malicious activities.

5. Data Sharing with Researchers

Through a long-term operation, we are holding huge amount of traffic data, malware samples and analysis results that are useful for students, researchers and security operators in other organizations. In order to provide those data for them, we have been developing the Nicter Open Network

Security Test-Out Platform (NONSTOP) that is a kind of a research environment and provides the various data in the nicter safely and conveniently. Since the data has risks of baneful effects for the public community when the data outflows despite our intentions, we applied some measures for the data leakage in the NONSTOP. After an experimental operation from Feb, 2011, we will provide the NONSTOP to other organizations including research institutes, CERTs, etc., within 2011.

6. Conclusion

As a countermeasure against malwares that often induce serious threats in the world, we have been developing the nicter that conducts the large-scale network incident analysis (MacS), collects and further analyzes malware samples (MicS) and figures out the root causes of the incidents (NemeSys). This paper introduced how the nicter collects and stores numerous amount of data such as network traffic, malware samples and even analysis results, in order to efficiently provide them to the various analysis engines. Furthermore, through the operation of the nicter for more than five years, our experimental results indicated a threat landscape that there are still many remaining hosts infected by W32.Downadup although it was old specimen found in 2008. Specifically, more than 60% of attacking hosts observed in our darknet could be infected by W32.Downadup. Furthermore, we confirmed that the result of the correlation analysis was correct in a rate of 86.18%. We conclude that the nicter is effective to grasp the landscape of threats on the Internet and we have to continue further development and operation of it for days to come.

References

[1] K. Nakao, K. Yoshioka, D. Inoue, M. Eto, and K. Rikitake. nicter: An Incident Analysis System using Correlation between Network Monitoring and Malware Analysis. In *The 1st Joint Workshop on Information Security (JWIS06)*, pages 363–377, 2006.

[2] K. Nakao, K. Yoshioka, D. Inoue, and M. Eto. A Novel Concept of Network Incident Analysis based on Multi-layer Observations of Malware Activities. In *The 2nd Joint Workshop on Information Security (JWIS07)*, pages 267–279, 2007.

[3] D. Inoue, M. Eto, K. Yoshioka, S. Baba, K. Suzuki, J. Nakazato, K. Ohtaka, and K. Nakao. nicter: An Incident Analysis System Toward Binding Network Monitoring with Malware Analysis. In *WOMBAT Workshop on Information Security Threats Data Collection and Sharing*, pages 58–66, 2008.

[4] M. Bailey, E. Cooke, F. Jahanian, J. Nazario, and D. Watson. The Internet Motion Sensor: A distributed blackhole monitoring system. In *Proceedings of the 12th ISOC Symposium on Network and Distributed Systems Security (NDSS)*, pages 167–179. Citeseer, 2005.

[5] SANS Internet Storm Center. `http://isc.sans.org/`.

[6] F. Pouget, M. Dacier, and V.H. Pham. Leurre.com: On the Advantages of Deploying a Large Scale Distributed Honeypot Platform. In *E-Crime and Computer Conference (ECCE' 05)*, 2005.

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

[8] M. Bailey, E. Cooke, F. Jahanian, A. Myrick, and S. Sinha. Practical darknet measurement. In *Information Sciences and Systems, 2006 40th Annual Conference on*, pages 1496–1501. IEEE, 2007.

[9] N. Provos. Honeyd-a virtual honeypot daemon. In *10th DFN-CERT Workshop, Hamburg, Germany*, 2003.

[10] C. Leita, M. Dacier, and F. Massicotte. Automatic handling of protocol dependencies and reaction to 0-day attacks with ScriptGen based honeypots. In *Recent Advances in Intrusion Detection*, pages 185–205. Springer, 2006.

[11] N. Provos. A virtual honeypot framework. In *Proceedings of the 13th conference on USENIX Security Symposium-Volume 13*, page 1. USENIX Association, 2004.

[12] E. Alata, V. Nicomette, M. Kaâniche, M. Dacier, and M. Herrb. Lessons learned from the deployment of a high-interaction honeypot. In *Dependable Computing Conference, 2006. EDCC'06. Sixth European*, pages 39–46. IEEE, 2006.

[13] R. Isawa, S. Ichikawa, Y. Shiraishi, M. Mori, and M. Morii. A Virus Analysis Supporting System-For automatic grasping virus behavior by code-analysis result. *Joho Shori Gakkai Shinpojiumu Ronbunshu*, 1(13):169–174, 2005.

[14] D. Inoue, M. Eto, K. Yoshioka, Y. Hoshizawa, Isawa R., M. Morii, and K. Nakao. Micro analysis system for analyzing malware code and its behavior on nicter. In *Symposium on Cryptography and Information Security (SCIS) 2007*. IEICE, Jan 2007.

[15] C. Willems, T. Holz, and F. Freiling. Toward automated dynamic malware analysis using cwsandbox. *IEEE Security & Privacy*, pages 32–39, 2007.

[16] N. Solutions. Norman sandbox whitepaper. `http://download.norman.no/whitepapers/whitepaper_Norman_SandBox.pdf`, 2003.

[17] K. Nakao, D. Inoue, M. Eto, and K. Yoshioka. Practical Correlation Analysis between Scan and Malware Profiles against Zero-Day Attacks Based on Darknet Monitoring. *IEICE TRANSACTIONS on Information and Systems*, 92(5):787–798, 2009.

[18] Microsoft Corporation. `http://www.microsoft.com/technet/security/Bulletin/MS08-067.mspx`.

[19] E Chien. Downadup: Attempts at Smart Network Scanning. http://www.symantec.com/connect/blogs/downadup-attempts-smart-network-scanning, 2009.

[20] E. Cooke, Z.M. Mao, and F. Jahanian. Hotspots: The root causes of non-uniformity in self-propagating malware. In *Dependable Systems and Networks, 2006. DSN 2006. International Conference on*, pages 179–188. IEEE, 2006.

Appendix. Analysis Result of W32.Downadup.B

```
Summary of the file:
    Name of file        : a5dc171f07ab85ec8526144ce56fbde1********_dll.exe
    Name by antivirus   : [S:W32.Downadup.B | T:WORM_DOWNAD.AD | M:w32/conficker.worm.gen.b]
    Category of file    : WORM
    Size of file        : 98980 bytes
    Create date         : 2004-09-07 03:23:06
    Hash Value (MD5)    : 0x5ecb255ffd18c11c17855e501*******
    Hash Value (SHA1)   : 0xa5dc171f07ab85ec8526144ce56fbde1f*******

This program creates mutex named as follows.
    Mutex 名: ZonesCounterMutex
    Mutex 名: ZonesCacheCounterMutex
    Mutex 名: ZonesLockedCacheCounterMutex
    Mutex 名: buxicddmksjytn
    Mutex 名: Global\1692050475-7

........ snip ........

This program creates the following files.
    Connects to Microsoft RCP service via named pipe:
        File: \\.\PIPE\lsarpc

    Connects to Workstation service via named pipe:
        File: \\.\PIPE\wkssvc
        File: { USB memory }\RECYCLER\S-5-3-42-2819952290-8240758988-87931****-****\jwgkvsq.vmx
        File: { USB memory }\autorun.inf

    Connects following file via named pipe:
        File: \\.\PIPE\browser

This program modifies the following files.
    { $SYSTEM$ }\01.tmp
    { USB memory }\RECYCLER\S-5-3-42-2819952290-8240758988-87931****-****\jwgkvsq.vmx
    { USB memory }\autorun.inf

This program installs the following services.
    Name of service: hyhwjah
        Explanation: hyhwjah
    The path of executed file: C:\WINDOWS\system32\01.tmp

This program deletes the following files.
    { $SYSTEM$ }\01.tmp
    HKEY_USERS\S-1-5-21-1085031214-1364589140-72534****-****\Software\Microsoft\windows\CurrentVersion\Internet Settings

........ snip ........

This program accesses to the following Web servers.
    http://www.google.com/
    http://jzuge.cn/search\q=0
    http://pioio.net/search\q=0
    http://kieurok.com/search\q=0
    http://zvlfx.net/search\q=0

........ snip ........

This program creates the following registry keys.
    HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile\GloballyOpenPorts\List
    HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders
    HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
    HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders
    HKEY_USERS\S-1-5-21-1085031214-1364589140-72534****-****\Software\Microsoft\Windows NT\CurrentVersion\Winlogon

........ snip ........

This program adds the following registry keys.
    HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\SharedAccess\Parameters\FirewallPolicy\StandardProfile\GloballyOpenPorts\List
    5310:TCP=5310:TCP:*:Enabled:jjjoeii
    HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
    Cache=C:\Documents and Settings\{ $USERNAME$ }\Local Settings\Temporary Internet Files
    HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Cache\Paths
    Directory=C:\Documents and Settings\{ $USERNAME$ }\Local Settings\Temporary Internet Files\Content.IE5

........ snip ........

This program disables proxy servers.
    HKEY_CURRENT_CONFIG\Software\Microsoft\windows\CurrentVersion\Internet Settings
        ProxyEnable=0
    HKEY_USERS\S-1-5-21-1085031214-1364589140-72534****-****\Software\Microsoft\windows\CurrentVersion\Internet Settings\Connections
        SavedLegacySettings=

This program creates the following backdoors.
    Accepts connections on 48660/TCP.

This program searches the following files/folders.
    Searches Microsoft Phonebook files.
    C:\Documents and Settings\All Users\Application Data\Microsoft\Network\Connections\Pbk\*.pbk
    C:\Documents and Settings\{ $USERNAME$ }\Application Data\Microsoft\Network\Connections\Pbk\*.pbk
    { USB memory }\RECYCLER\S-5-3-42-2819952290-8240758988-87931****-****\jwgkvsq.vmx
    { USB memory }\autorun.inf

This program deletes the following registry keys.
    HKEY_USERS\S-1-5-21-1085031214-1364589140-72534****-****\Software\Microsoft\windows\CurrentVersion\Internet Settings
        ProxyServer
    HKEY_USERS\S-1-5-21-1085031214-1364589140-72534****-****\Software\Microsoft\windows\CurrentVersion\Internet Settings
        ProxyOverride

........ snip ........

The hash values of files created by this program is as follows.
    0x5ecb255ffd18c11c17855e501*******
    0x67d6145f423ad7f02ec127d01*******
```

HARMUR: Storing and Analyzing Historic Data on Malicious Domains

Corrado Leita
Symantec Research Labs
Sophia Antipolis, France
corrado_leita@symantec.com

Marco Cova*
School of Computer Science
University of Birmingham
Birmingham, United Kingdom
m.cova@cs.bham.ac.uk

ABSTRACT
A large amount of work has been done to develop tools and techniques to detect and study the presence of threats on the web. This includes, for instance, the development of a variety of different client honeypot techniques for the detection and study of drive-by downloads, as well as the creation of blacklists to prevent users from visiting malicious web pages. Due to the extent of the web and the scale of the problem, existing work typically focuses on the collection of information on the current state of web pages and does not take into account the temporal dimension of the problem.

In this paper we describe HARMUR, a security dataset developed in the context of the WOMBAT project that aims at exploring the dynamics of the security and contextual information associated to malicious domains. We detail the design decisions that have led to the creation of an easily extensible architecture, and describe the characteristics of the underlying dataset. Finally, we demonstrate through examples the value of the collected information, and the importance of tracking the evolution of the state of malicious domains to gather a more complete picture on the threat landscape.

1. INTRODUCTION
The Internet threat scenario is extremely diverse and is in continuous evolution. In the last years, we have witnessed a partial shift of attention from server-side attacks to client-side ones. An increasingly popular vector for malware propagation leverages the web to propagate to victim hosts through their interaction with client software (i.e. web browsers). For instance, in the so-called *drive-by downloads*, the user is infected by simply visiting a malicious web-page, or a benign web-page modified by malicious actors to redirect client traffic towards exploit-distribution servers [16]. Drive-by downloads are responsible for the spread of most of the recent malware infections, such as the Torpig botnet [18] or the Hydraq trojan [20].

Similarly to what has been done in the past for server-site attacks, researchers have studied solutions to identify these threats, both to protect users and to quantify the extension of the phenomenon. This has been mainly achieved by crawling the web or by visiting suspicious URLs and then analyzing the discovered web content to detect exploits. A variety of client honeypots with different characteristics has been proposed in the literature [9, 14, 17, 21, 23, 26].

However, the problem to be tackled by these techniques profoundly differs from that previously addressed for the analysis of server-side threats. A tool that has been widely used for the collection of data on server-side threats is that of the honeypots, network hosts with no specific function other than interacting with malicious actors scanning their network. While server-side honeypots are by definition passive systems, that react to traffic initiated by malicious actors or infected hosts, client-side honeypots are active components that need to be driven towards a URL in order to assess its maliciousness. This difference in operational pattern has important impacts on the collection of data on client side threats:

1. By simply waiting for incoming activities, a server-side honeypot has immediate visibility on the temporal evolution of an infection. Previous work has underlined the importance of looking at the threat dynamics to correlate apparently dissimilar activities [12] and to understand the propagation of a malware infection [2]. Information on the *temporal evolution of client-side threats* is much more challenging to obtain, since it requires to actively reconsider previous analyses on a regular basis.

2. No legitimate activity should ever be generated towards a server-side honeypot. Aside from traffic generated as a consequence to misconfigurations, most of the traffic targeting a honeypot is likely to have a malicious nature. The challenge in the analysis of such traffic consists of characterizing the type of activity, for instance by employing techniques able to discern code injections from lower impact activities [1, 7, 13]. Conversely, the security state of a URL analyzed by different client honeypots is much more difficult to assess. The *URL may exhibit different characteristics*

*The work of Marco Cova was supported by the Symantec Research Labs Graduate Fellowship Program.

if analyzed at different times (e.g., the exploit server may have been shut down, or the infected websites may have been cleaned) or if analyzed from different network locations (e.g. malicious ads may target users belonging to a specific geographical location [6]).

HARMUR, the **H**istorical **AR**chive of **M**alicious **URL**s, is a security dataset developed in the context of the WOMBAT project[1] that tries to address these two challenges. HARMUR leverages publicly available information on the security and network state of suspicious domains to build a "big picture" instrumental to a better understanding of web-borne threats and their evolution. HARMUR specifically addresses the two previously introduced challenges by focusing on the threat dynamics and on the threat context.

Threat dynamics. HARMUR is designed to perform a set of analysis tasks for each of a set of tracked domains. Each analysis task aims at collecting information about the state of a domain by querying different information sources. Since the moment in which a specific domain is first introduced in the dataset, HARMUR implements a scheduling policy aiming at repeating the analysis on a regular basis, giving priority to domains that are believed to be "most interesting" and trying to allocate its resources on a "best-effort" fashion. In the long term, this allows the reconstruction of an approximate timeline of the evolution of the domain state that can be instrumental to a better understanding of the threat dynamics.

Threat context. HARMUR aggregates information from a variety of sources to gain a more complete understanding of the state of a monitored resource. By collecting information from different security and networking feeds on a regular basis, it is possible to rebuild a partial "ground truth" on the state of a website at a given point in time. For instance, it is possible to correlate a change in the security state (e.g., from malicious to benign) with a change in the DNS records, or with the fact that the server has stopped responding to HTTP requests.

2. RELATED WORK

When analyzing web threats, previous work has often proposed ad-hoc analysis solutions that focus the attention on the mechanics and dynamics of specific threat instances [8, 11, 18]. Various detection techniques have been proposed for the detection of web-borne malware propagations. Most of the work has focused on the analysis and detection of drive-by downloads: similarly to what previously happened for server-side honeypots, researchers have proposed techniques with varying resource costs and levels of interaction.

High interaction client honeypots leverage a full-fledged and vulnerable browser running in a contained environment. High interaction client honeypots include Capture-HPC [21], HoneyClient [23], HoneyMonkey [26] and Shelia [25]. In most cases, a website is considered as malicious if visiting it with the browser results in an unexpected system modification, such as a new running process or a new file (the exception is Shelia, that leverages memory tainting to detect an exploit condition). In all cases, a website can be flagged as malicious if and only if the threat is targeting the specific setup found on the honeypot. For instance, a website exploiting a vulnerability in the Adobe Flash plugin will be flagged as benign by a honeypot on which the plugin is not installed.

Low interaction client honeypots leverage a set of heuristics for the detection of vulnerabilities within a specific web page. For instance, HoneyC [17] leverages Snort signatures for the detection of malicious scripts, while SpyBye scans the web content using the ClamAV open-source antivirus [14]. In both these cases, the detection of a threat depends on the generation of some sort of signature, and therefore requires a relatively detailed knowledge of the threat vector. More sophisticated approaches have been proposed, such as PhoneyC [10], that implements vulnerability plugins to be offered to the malicious website (similarly to what Nepenthes [1] does for server-side honeypots) and Wepawet [3, 5] that employs sophisticated machine learning techniques to analyze Javascript and Flash scripts.

Despite the variety of the proposed approaches, it is worth noticing that none of the proposed solutions is likely to achieve 100% detection rate. Either because of the impossibility of detecting exploits targeting a different configuration, or because of the limitations implicit to the usage of heuristics, both high- and low-interaction client honeypots have a non-null failure probability that may lead them to mark a malicious website as benign. Previous work has proposed to deal with these limitations by combining low- and high-interaction techniques [6, 15]. With HARMUR, we push this attempt further by proposing a generic aggregation framework able to collect and correlate information generated by different security feeds (e.g. different client honeypots with different characteristics) with generic contextual information on the infrastructure underlying a specific domain. By correlating such information, we aim at learning more on the structure the characteristics of the web threats, as well as on the characteristics and limitations of modern web threats analysis techniques. For instance, HARMUR has been used in the past as information source for the analysis of a specific threat type, that of Rogue AV domains [4], although it collects information on a variety of other different web-borne threats.

3. HARMUR ARCHITECTURE

As previously explained, HARMUR is not, per se, a client honeypot. HARMUR is an aggregator of information generated by third parties, and uses this information to generate a historical view for a set of domains that are believed to be malicious. In order to allow HARMUR to scale to a significant number of domains while building this historical view, all the analysis operations must have low cost: we have decided to avoid by design any expensive operation such as crawling the domains' content, and to rely on existing crawling infrastructures for the analysis of the domain security.

The HARMUR dataset and associated framework is built around the concept of *domain*, more exactly the Fully Qualified Domain Name (FQDN) that is normally associated with one or more URLs. Each domain is associated to a color,

[1]`http://www.wombat-project.eu`

that identifies its *current* status following the following coding:

Red. At least one threat is currently known to be served by one of the hostnames belonging to the domain.

Green. No threat has ever been known to be associated to any of the hostnames belonging to the domain (the domain has never been red).

Orange. No threat is currently known to be hosted within the domain, but the domain has been red in the past.

Gray. None of the currently available security feeds is able to provide any information on the domain. This is likely due to the fact that the domain has never been analyzed.

Black. None of the hostnames associated to the domain is currently reachable. This can be due to removal of the associated DNS records or to a failure in responding to HTTP requests.

The HARMUR framework is made of two main building block types that are in charge of populating the underlying dataset with various types of metadata.

URL feeds. URL feeds are in charge of populating the HARMUR dataset with lists of fresh URLs (and associated FQDNs) that are likely to be of interest. This can include lists of URLs detected as malicious by crawlers, but also URLs available in public lists of malicious domains or phishing sites.

Analysis modules. An analysis module wraps an action to be periodically performed on a specific domain. An analysis module is defined by an action, timing information for its repetition, a set of dependencies with respect to other analysis modules, and a list of color priorities.

Action. The specific analysis to be executed on the domain. An analysis module has full visibility over the information currently available for the assigned domain, and is in charge of updating the domain status. It should be noted that an analysis module never deletes any information from a domain state: any information in HARMUR is associated to a set of timestamps that define the time periods in which it has been seen holding true.

Timing settings. Each analysis module defines a function $T(color)$ that specifies the frequency with which a given domain should be analyzed (given unliminted resources) as a function of its current color. For instance, the security state of a red domain is likely to change more quickly than that of a green domain, and should therefore be checked more frequently (e.g. on a daily basis) than that of a green one (that can be checked on a weekly or even monthly basis).

Module dependencies. Each analysis module is likely to depend on the output generated by other analysis modules: for instance, a module in charge of analyzing the geographical location of the web servers associated to a specific domain requires to have access to the DNS associations, generated by another module. This information is used by HARMUR in the scheduling process to decide which domains qualify for the analysis from a specific module.

Color priorities. Each module defines the priority rules for the choice of the batch of k domains to be processed at a given analysis round. For instance, checking the availability of a red domain should have priority over the execution of the same action for a domain that has been green for a long time. Still, a green domain should still be checked whenever resources are available.

HARMUR's core consists of a simple scheduler in charge of assigning tasks to a pool of threads. Each analysis task is composed of an analysis module (defining a specific analysis action) and a batch of k domains to be processed by the analysis module (where k is a configuration parameter). The k domains are picked among those that have not been analyzed by the specific analysis module in the last $T(color)$ minutes. Among all the domains requiring analysis, the choice of the k domains is based on their current color and the color priorities specified by the analysis module using a simple assignment algorithm. The scheduler initially assigns a total of n (with $n \ll k$) domains to each priority class, and then proceeds to fill the remaining positions starting from the highest priority. For instance, if a module specifies the following color priority order: $[red, green, yellow]$, with $k = 100$ and $n = 10$ and if a total of 50 red domains, 60 green domains, and 100 yellow domains need to be analyzed, the batch will be composed of 50 red domains, 40 green domains and 10 yellow domains.

3.1 Input feeds

Thanks to the definition of these basic building blocks, the HARMUR framework is easily expandable with new URL feeds or analysis modules. The currently implemented components are represented in Figure 1. HARMUR receives URL feeds from a variety of sources:

- Norton Safeweb (`http://safeweb.norton.com`)
- Malware Domain List (`http://malwaredomainlist.com`)
- Malware URL (`http://www.malwareurl.com`)
- Hosts File (`http://www.hosts-file.net`)
- Phishtank (`http://www.phishtank.com/`)

On top of these basic URL sources, HARMUR has the possibility to enrich the initial URL feed for URLs having specific characteristics. This is achieved by leveraging passive DNS information (obtained from `http://www.robtex.com/`)

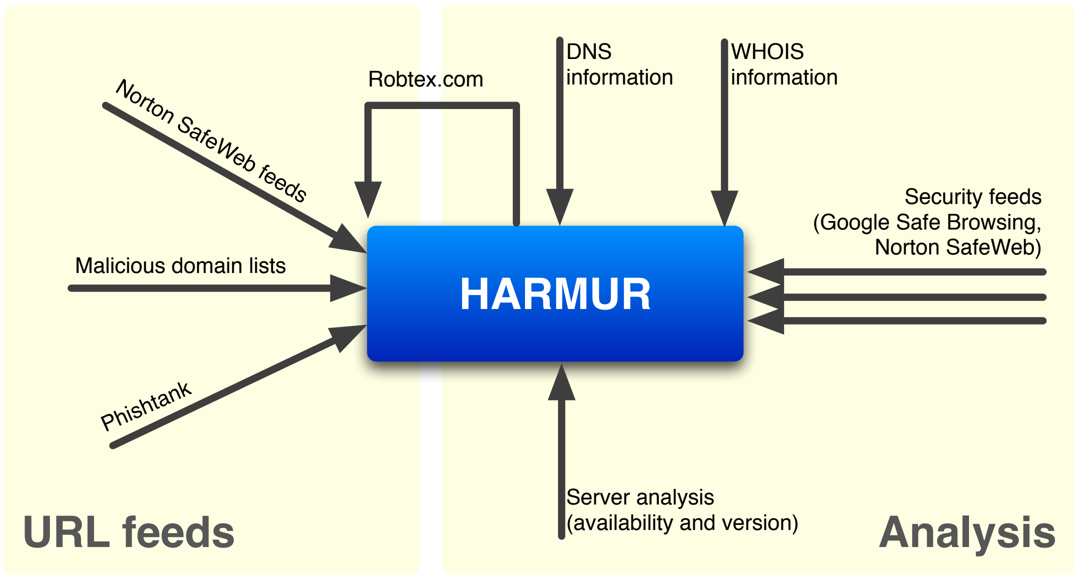


Figure 1: HARMUR architecture

to discover all the hostnames resolving to the same IP address as that of the tracked URL, therefore extracting a rather complete list of all the domains hosted on the same web server.

Each of the domains discovered by the URL feeds is analyzed periodically by a set of analysis modules, aiming at retrieving information on 1) DNS associations; 2) domain registration (WHOIS) records; 3) web server status; 4) security status for the domain. In the following Sections we briefly describe the information collected by each module; the interested reader can find in Figure 4 (in Appendix) a more detailed view of the relationships among the different information entities composing the HARMUR dataset.

3.2 DNS associations

For each domain name, HARMUR collects as much information as possible on its association to physical resources such as its authoritative nameservers, as well as the address of the servers associated to the hostnames known to belong to the domain. DNS information is retrieved by HARMUR by directly contacting the authoritative nameserver for each domain, avoiding the artifacts normally generated by DNS caching. When looking at DNS records for each tracked domains, HARMUR stores historical information on the association of a domain to its authoritative nameserver(s) (NS records) as well as the association of each hostname to host IP addresses (A records).

3.3 Domain registration

On top of the basic DNS information, HARMUR leverages the WHOIS protocol to retrieve information on the registration information for each domain name. This includes the date in which a given domain was registered, as well as the name of the registrar and registrant.

It should be noted that WHOIS information is very difficult to extract in practice. RFC 3912, describing the WHOIS protocol specification, clearly states that *the protocol delivers its content in a human-readable format.* In the context of an automated retrieval and dissection of the whois information, this is a non-negligible problem: every registrar uses different notation and syntax to represent the managed registration records. HARMUR implements parsers for the most common registrars, but WHOIS information is incomplete for those domains that belong to unsupported ones.

3.4 Web server status

HARMUR extracts a variety of different information on each web server known to be hosting content associated to the analyzed domain.

Location. HARMUR takes advantage of the Maxmind geolocation database to collect information on the geographical location of the servers hosting the tracked content.

Autonomous system. In parallel to the physical location of the servers, HARMUR maps each server to the Autonomous System (AS) it is in. As shown by FIRE [19], online criminals have been often masquerading behind disreputable Internet Service Providers known to give little or no attention to the type of activity carried out by their customers. Collection of AS information in HARMUR is possible thanks to Team Cymru's IP to ASN mapping project[2]. To disambiguate cases in which the service provides an ambiguous answer (sometimes multiple AS numbers are returned), HARMUR cross-checks the Team Cymru's information with that provided by the Route Views project (`routeviews.org`).

Server availability and version. HARMUR was designed to be able to scale to an extremely large number of

[2]`http://www.team-cymru.org/Services/ip-to-asn.html`

domains, and is therefore, by design, incapable of performing expensive analyses on the tracked domains. Still, it is possible to retrieve from web servers very valuable information through simple HTTP interactions. HARMUR probes each server with an HTTP HEAD request for the root page of the hosted site. By parsing the associated answer, HARMUR collects information on the availability of the server, but also on the server configuration as advertised in the HTTP headers. This information can be valuable for the discovery of popular configurations that may be indicative of "deployment campaigns", in which the very same setup is replicated on a large number of servers.

3.5 Security state

Currently HARMUR offers analysis modules for the collection of security data from two information sources commonly used in web browsers for blocking users from visiting malicious domains: Google Safe Browsing (integrated in popular browsers such as Chrome, Safari and Firefox) and Norton Safeweb[3], part of the Norton Antivirus security suite. The information generated by these two analysis modules determines the current color of a domain.

The two feeds have very different characteristics: Google Safe Browsing is designed as a simple blacklist, and provides no distinction between unknown and safe domains. This peculiarity is handled within HARMUR through the assignment of only two colors: red for blacklisted websites, and gray for all the others. Norton Safeweb provides instead a more detailed report, that associates a specific domain to a specific color, as well as to a list of URLs and associated threat categories.

4. USE CASES

This paper has described at length the characteristics of the HARMUR framework and of the associated dataset. The current HARMUR prototype has been written entirely in python and has been running for more than two years on a quad-core Xeon 2.66 GHz CPU with 8GB of RAM. HARMUR is currently tracking 91,561 red domains, 106,303 orange domains, 643,308 black domains, and more than 2 million domains whose security state is either benign or unknown (gray). The analysis of these domains over a period of two years has led to the identification of 1.2M web servers that have been witnessed as active by HARMUR at least once throughout its activity, and has led to associate 5M distinct URLs to at least one threat. This considerable amount of data has been continuously tracked by the analysis modules, that were configured to reconsider the state of domains associated to "interesting" colors on a weekly basis. Looking at the last year of operation, each analysis module has been able to process an average of 16k domains per day, although this number may be easily increased through implementation refinements and better parallelization of the code.

While a large scale analysis of the dataset content is left for future work, important lessons have been learned through the analysis of specific cases that underline the importance and the potential of the collected data.

[3]`http://safeweb.norton.com`

2009-08-07 12:28:14	red
...	
2009-09-04 23:50:33	red
2009-09-21 10:03:02	gray
...	
2009-12-05 03:53:31	gray
2009-12-13 09:07:03	red
2009-12-24 20:49:19	gray
...	
2010-03-22 18:24:46	gray
2010-04-07 03:04:39	red
...	
2010-04-14 19:18:19	red
2010-04-18 06:11:49	gray
...	

Table 1: Security evolution for a specific domain.

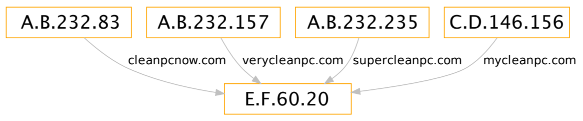


Figure 2: Rogue site dynamics (all IP addresses have been anonymized)

Infection history. Table 1 shows the evolution over time of a specific domain. This domain, hosted in United Kingdom, is currently ranked as benign in all security information feeds. However, its past shows us a different scenario. Looking at Google Safe Browsing information, we can see that the domain has been flagged as malicious three times since August 2009. Looking more in-depth at the information provided by Norton Safeweb, we discover that this domain had been infected multiple times with an exploit toolkit widely used as part of the Mebroot/Torpig campaign [18], in which websites having poor administrator passwords were being modified by attackers to serve as landing pages for the malware infection. While currently benign, we can infer from the domain history that the likelihood for this domain to be malicious in the future is higher than that of a domain having a "clean past".

The dynamics associated to the hosting infrastructure can also give us important hints on the structure and the modus operandi of a specific malicious campaign.

Malware campaigns. The HARMUR data collection mechanism aims at generating information on the evolution of the tracked domains. All the features previously described are therefore repeatedly extracted throughout the lifetime of a malicious domain. For instance, Figure 2 shows an example of "domain movement". The boxes in the graph represent sets of IP addresses associated to a domain at a given point in time, and the edges connecting the boxes graphically represent the transition of a domain to a new set of addresses. In Figure 2 we can see how four distinct domains, all associated to the distribution of rogue security software and

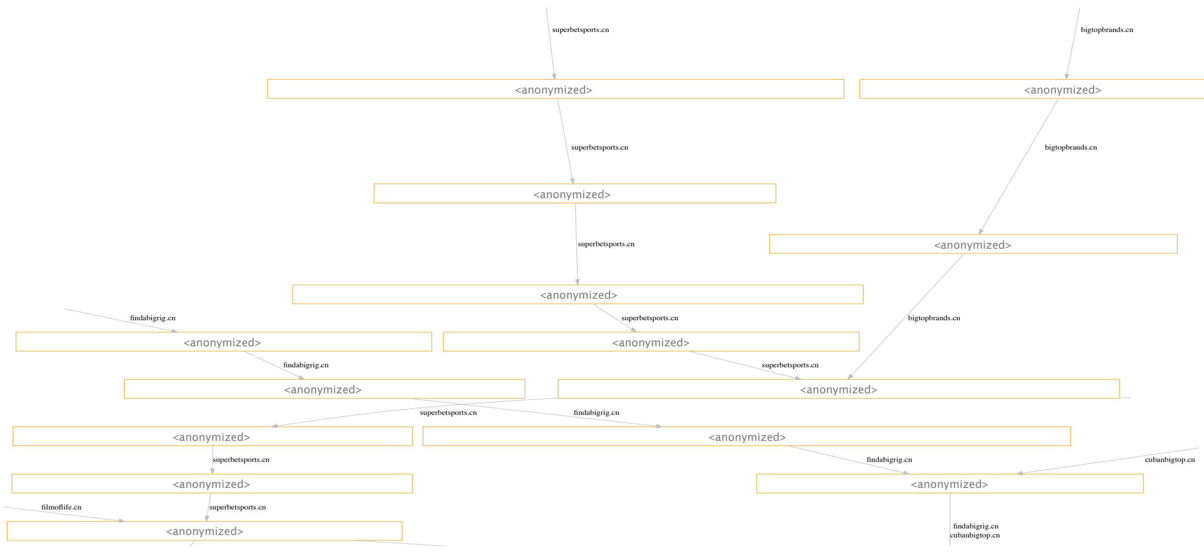


Figure 3: Evolution of DNS information for fast-flux networks (all IP addresses have been anonymized)

initially hosted on different servers, have migrated approximately at the same time to a single server, which was also found to be malicious (time information is not represented for the sake of simplicity). It is important to note that these four domains were initially thought of being completely unrelated. It is only through their subsequent movement that it has been possible to link them all to a probable common root cause.

An even more apparent example of the value of the dynamic correlation possible through the HARMUR dataset is represented in Figure 3. The domains taken into consideration here are hidden behind Fast Flux networks [22] to protect the identity and availability of the associated server. The periodic movement of the DNS association among the pool of available addresses leads to long association chains, but also to intersections among domains that are likely due to the leveraging of the same pool of infected machines. Once again, apparently unrelated domains are correlated thanks to the periodic analysis of their state.

Attackers modus operandi The data collected by HARMUR proved to be extremely valuable in understanding the modus operandi of the attackers. For instance, by leveraging the information retrieved by the WHOIS component, we noticed that a single registrant registered 71 distinct domains exactly on the same day on ONLINENIC. The domain names were the result of the permutation of a few dictionary words associated to the name of an antivirus software, and the all the hostnames known to HARMUR as belonging to these domains resolved to a single physical web server. A more in depth analysis revealed that all these domains were ultimately used for the distribution of rogue security software [4].

5. CONCLUSION

This paper has presented HARMUR, a tool to take into consideration the evolution of the state of malicious domains to gather insights on the threat landscape dynamics. Differently from traditional datasets monitoring the web for client-side threats, with HARMUR we have tried to go beyond the collection of information on the current state of a domain, and rebuild an approximate timeline of its history and its evolution over time. While a large-scale analysis of the information contained in the dataset is left for future work, we demonstrate through examples the value of looking at the threat dynamics to gather more in-depth insights on the modus operandi of attackers, and on the identification of groups of domains likely to be associated to the same root cause or campaign.

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.

6. REFERENCES

[1] P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. Freiling. The Nepenthes Platform: An Efficient Approach to Collect Malware. In *9th International Symposium on Recent Advances in Intrusion Detection (RAID)*, September 2006.
[2] Caida Project. Conficker/Conflicker/Downadup as seen from the UCSD Network Telescope.
[3] M. Cova, C. Kruegel, and G. Vigna. Detection and analysis of drive-by-download attacks and malicious JavaScript code. In *Proceedings of the 19th international conference on World wide web*, pages 281–290. ACM, 2010.
[4] M. Cova, C. Leita, O. Thonnard, A. Keromytis, and M. Dacier. An analysis of rogue av campaigns. In *Proc. of the 13th International Symposium on Intrusion Detection (RAID 2010)*, September 2010.
[5] S. Ford, M. Cova, C. Kruegel, and G. Vigna. Analyzing and Detecting Malicious Flash

Advertisements. In *Proceedings of the 25th Annual Computer Security Applications Conference*, 2009.

[6] P. Kijewski, C. Overes, and R. Spoor. The HoneySpider Network – fighting client-side threats. In *FIRST Annual Conference*, 2008.

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

[8] K. McGrath and M. Gupta. Behind Phishing: An Examination of Phisher Modi Operandi. In *Proc. of the USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2008.

[9] A. Moshchuk, T. Bragin, D. Deville, S. D. Gribble, and H. M. Levy. Spyproxy: Execution-based detection of malicious web content. In *Proceeding of the 16th USENIX Security Symposium*, 2007.

[10] J. Nazario. Phoneyc: A virtual client honeypot. In *Proceedings of the 2nd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more*, page 6. USENIX Association, 2009.

[11] H. O'Dea. The Modern Rogue — Malware With a Face. In *Proc. of the Virus Bulletin Conference*, 2009.

[12] 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.

[13] G. Portokalidis, A. Slowinska, and H. Bos. Argos: an emulator for fingerprinting zero-day attacks. In *ACM Sigops EuroSys*, 2006.

[14] N. Provos. Spybye, `http://www.monkey.org/~provos/spybye`.

[15] N. Provos, P. Mavrommatis, M. Rajab, and F. Monrose. All Your iFRAMEs Point to Us. In *Proc. of the USENIX Security Symposium*, 2008.

[16] N. Provos, D. McNamee, P. Mavrommatis, K. Wang, and N. Modadugu. The ghost in the browser. analysis of web-based malware. In *First Workshop on Hot Topics in Understanding Botnets (HotBots 07)*. Google, Inc, 2007.

[17] C. Seifert, I. Welch, and P. Komisarczuk. HoneyC-The Low-Interaction Client Honeypot. *NZCSRCS, Hamilton, 2007, `http://www.mcs.vuw.ac.nz/cseifert/blog/images/seiferrt-honeyc.pdf`*, 2006.

[18] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna. Your Botnet is My Botnet: Analysis of a Botnet Takeover. In *Proc. of the ACM Conference on Computer and Communications Security*, 2009.

[19] B. Stone-Gross, A. Moser, C. Kruegel, K. Almaroth, and E. Kirda. FIRE: FInding Rogue nEtworks. In *25th Annual Computer Security Applications Conference (ACSAC)*, December 2009.

[20] Symantec. The trojan.hydraq incident, `http://www.symantec.com/connect/blogs/trojanhydraq-incident`.

[21] The Honeynet Project. Home page of Capture-HPC, `https://projects.honeynet.org/capture-hpc`.

[22] The Honeynet Project. Know your enemy: Fast-flux service networks, `http://www.honeynet.org/book/export/html/130`.

[23] The MITRE Honeyclient Project Team. HoneyClient, `http://www.honeyclient.org`.

[24] The WOMBAT FP7 project. Second WOMBAT workshop proceedings, `http://wombat-project.eu/2010/02/wombat-deliverable-d10d63-seco.html`.

[25] VU Amsterdam. Shelia, `http://www.cs.vu.nl/~herbertb/misc/shelia/`.

[26] Y. Wang, D. Beck, X. Jiang, R. Roussev, C. Verbowski, S. Chen, and S. King. Automated web patrol with strider honeymonkeys. In *Proceedings of the 2006 Network and Distributed System Security Symposium*, pages 35–49, 2006.

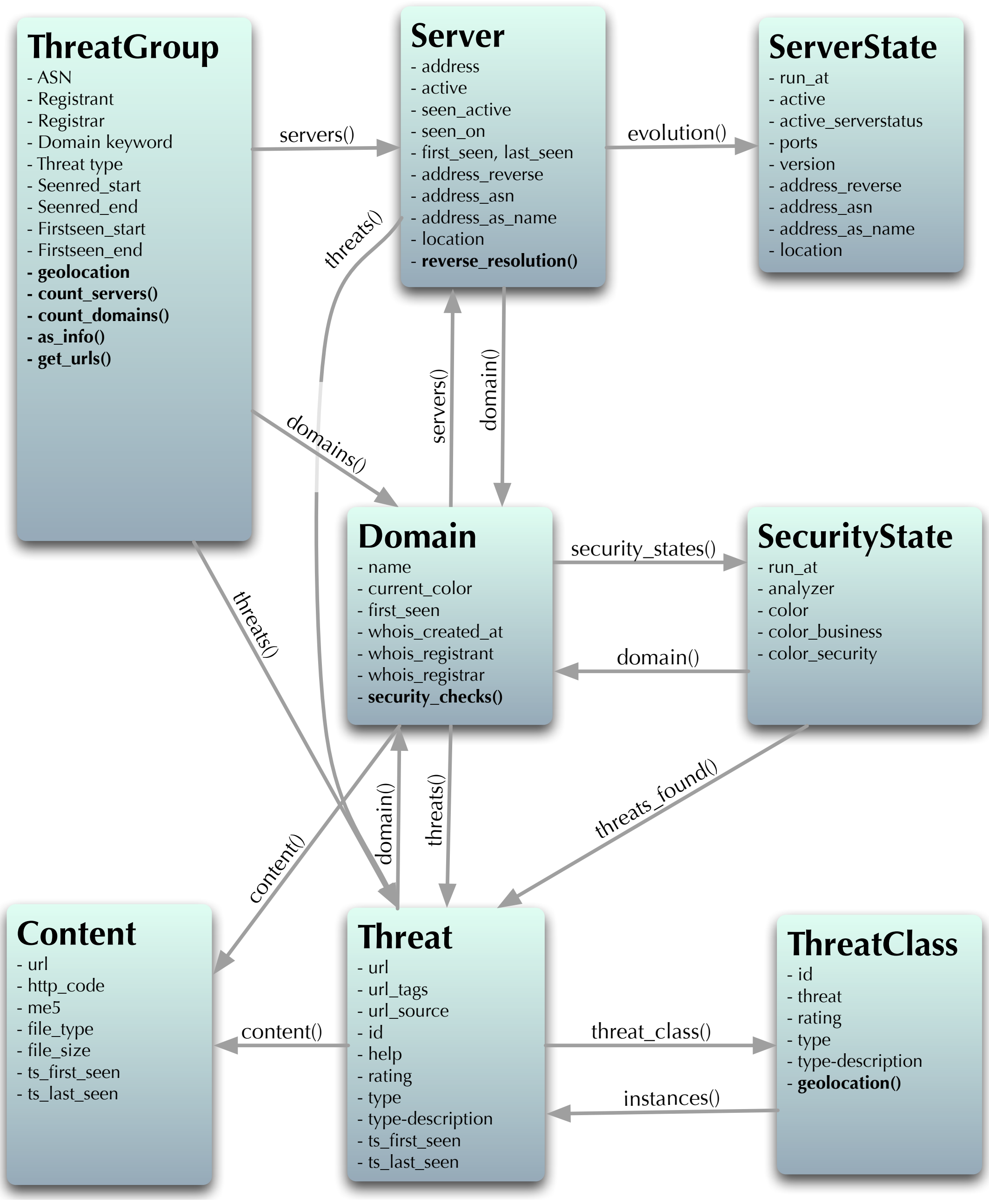


Figure 4: Relationships between the main HARMUR dataset objects, as they are implemented in the current version of the WOMBAT API [24].

Adversaries' Holy Grail: Access Control Analytics*

Ian Molloy, Jorge Lobo, Suresh Chari
IBM T.J. Watson Research Center
{molloyim, jlobo, schari}@us.ibm.com

ABSTRACT

The analysis of access control data has many applications in information security, including: role mining and policy learning; discovering errors in deployed policies; regulatory compliance; intrusion detection; and risk mitigation. The success of research in these areas hinges on the availability of high quality real-world data. Thus far, little access control data has been released to the public. We analyze eight publicly released access control datasets and contrast them with three client policies in our possession. Our analysis indicates there are many differences in the structure and distribution of permissions between the public and client datasets, including sparseness, permission distributions, and cohesion. The client datasets also revealed a wide range of semantics and granularities of permissions, ranging from application-specific rights to general accounts on systems we could not observe on the public data due to anonymization. Finally, we analyze the distribution of user-attributes, which the public datasets lack. We find techniques that work well on some datasets do not work equally well on others and discuss possible future research and directions based on our experience with real-world data.

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*Access Controls*

*This Research is conducted through participation in the International Technology Alliance sponsored by the U.S. Army Research Laboratory and U.K. Ministry of Defense and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the US Army Research Laboratory, the U.S. Government, the UK Ministry of Defense, or the UK Government. The US and UK Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

General Terms

Experimentation, Security

Keywords

Access Control, Real-World Data, RBAC, Analysis, Metrics

1. INTRODUCTION

Provisioning entitlements in an organization is a challenging task, especially in organizations with thousands of users and tens of thousands of resources. A common solution is to use role-based access control: roles are assigned permissions and users are authorized to roles. However, before roles can be used, they must first be defined by administrators, a challenging and time consuming task known as role engineering.

Simplifying the role engineering task has become an active area of research. Many, including ourselves, have investigated how to apply data mining and analytics techniques to existing data [3, 4, 6, 8–11, 13, 16, 17]. We have spent the past four years building and validating our own techniques and other's solutions on real customer data. The best way to validate academic work is with real data, using appropriate metrics that access the quality and fitness of solutions to real-world problems. In this paper we will discuss our experience with customer access control data, and some of the differences we have observed between real-world data and assumptions made in theoretical work.

We analyze and compare eleven access control datasets: eight have been publicly released, and three are confidential policies from clients. We found the public and private data differs in several key aspects that critically impacted the utility of well-studied solutions on private data. Key differences include:

- Customer data is more sparse, assigning users a small fraction of the entitlements.
- Public data is more compressible, allowing them to be expressed with a smaller relative number of roles.
- Customer data has higher entropy; clusters of users and permissions are less well defined, smaller, and lack cohesion.
- Customer data has a long tail distribution, while groups of permissions in public data are assigned to a similar number of users.
- The granularity of permissions varies, e.g., accounts on systems to columns on tables in databases.
- There is known noise in the private data, while the anonymization of public data makes the identification of noise difficult.

- The utility of attributes in determining levels of access is inconsistent across datasets.

While we cannot disclose the client data, we hope to point out how it is different from many of the theoretical assumptions made in academic works, aiding future endeavors. We tried to give as much detail and information as possible while not compromising confidentiality. It is our aim that this analysis provides hints and helps guide future research into models of access control in real-world organizations. Several assumptions made in many academic works, including our own such as the distribution of permissions, did not hold true in the client data. Such analysis may lead to more realistic data generators than those used previously [12, 17]. To this end, the remainder of this paper will focus on the analysis of the data, and not the underlying theory of role mining or access control analytics.

The presentation is roughly divided in three parts. First, we discuss the pre-processing of the data. Access control information comes in many forms and at are many levels of granularity. What is considered a resource or permission can be very different from case to case. Even in the same data set, the granularity of the information can vary given that information is collected from different types of systems. During pre-processing, there is a considerable amount of manual labor and simplification. In the second part we analyzed the structure of the data after distillation into a binary access control matrix. We will discuss difference and similarities of the different data sets. The third part is about the use of semantic information during analysis. In particular we will discuss the inclusion of attribute information for data analysis. Surprisingly, having attributes does not necessary help with the analysis. We conclude with our thoughts on the current state of the data and the types of data that we are seeking to continue our research.

2. DESCRIPTION OF THE DATA

We now describe the real-world access control datasets in our possession. We will divide these into two categories: datasets from customers that contain confidential and sensitive information and cannot be released; and those publicly available and used in the access control analytics literature.

2.1 Private Data

Access control data is particularly difficult to obtain for researchers, and only a small handful have been released to the public. Datasets have primarily been used for research on *role mining*, automatically converting a non-role-based access control system to an role-based system. We have obtained several datasets from clients for use in research and evaluation of role mining and access control analytics. To protect the confidentiality of the data owners we simply refer to these as **customer 1**, **customer 2**, and **customer 3**. When necessary, we may obfuscate attribute names and other key values, though they are known to us.

Customer 1.

The first dataset is from a medium size organization used to provision IT system administrators with entitlements. The dataset contains 311 users and 1105 permissions with 7868 user-permission assignments (a density of 2.15%). A permission is an account on a system, such as a file server, and may indicate administrative (e.g., `sudo`) access. There are six attributes describing each user that provide insights into their duties for the organization.

Customer 2.

The second dataset is an access control policy from a system that provisions administrative access to an outsourced IT system. The policy consists of 881 users and 852 permissions with 6691 authorizations (a density of 0.89%). There are 25 attributes describing each user, however many contain attributes like telephone number, which we trim to only eight that are applicable for determining user-entitlements based on known semantics.

Customer 3.

The third dataset contains 3068 users, 3133 permissions, and 71596 user-permission assignments, giving it a density of 0.74%. Permissions may be broadly categorized as Active Directory groups or Applications technical roles[1], and are not permissions in the traditional sense. However, each group or technical role constitutes one or more levels of access, and will be treated as semantically identical to permissions. Application permissions are given as an application-account pair. Users are assigned a total of eight attributes.

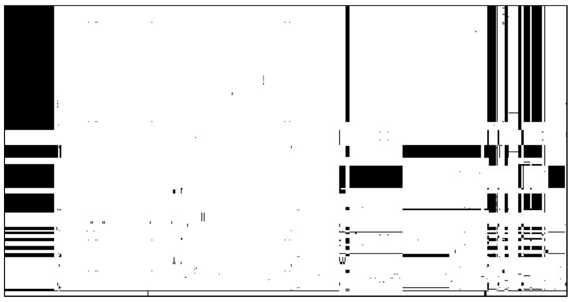
2.2 Publicly Available Data

To contrast the confidential client datasets, we analyze eight real-world datasets that have been highly anonymized and released to the public. The bulk of these datasets were released by researchers at HP Labs for use in evaluation in [4]. They have since been used extensively for evaluating role mining algorithms [10, 12, 13]. There are eight datasets (their **customer** dataset has not been released) in total: the **healthcare** data was obtained from the US Veteran's Administration; the **domino** data was from a Lotus Domino server; **americas** (both **large** and **small**), **emea**, and **apj** data were from Cisco firewalls used to provide external users access to HP resources. There are also two firewall policies, **firewall1** and **firewall2**. These datasets are all provided as binary relations, e.g., *user i has permission j*, and all semantics of users and permissions are speculative.

The **university** dataset, while not from a real-world access control policy, has been used in the literature and is believed to be representative of an access control policy in a university setting. The data was generated from a template[2] use in [15]. We present it here for comparison only, and will focus our analysis on the real datasets.

The size of each dataset is most succinctly represented by the number of users, permission, permissions assigned to users (UP), and the density of the data, the number of granted assignments per possible request ($|\,UP\,|\,/\,|\,U \times P\,|$). A summary of the datasets is given in Table 1. Indicated are the number of users (U), permissions (P), attribute types (A), granted assignments (UP), and the density. For the attributes[3], we indicate those applicable for access control (by semantics of the attributes), and the total number of attribute in parenthesis. The user-permission data can be visualized as a black-and-white image where the Y-axis, or height, is a user, the X-axis are permissions, and a black

[1]A technical role is a group in an application or system. A technical role can be characterized as a group of permissions.
[2]`http://www.cs.sunysb.edu/~stoller/ccs2007/university-policy.txt`
[3]Attributes in **university** are tags, and not unique keys.


(a) Firewall 1

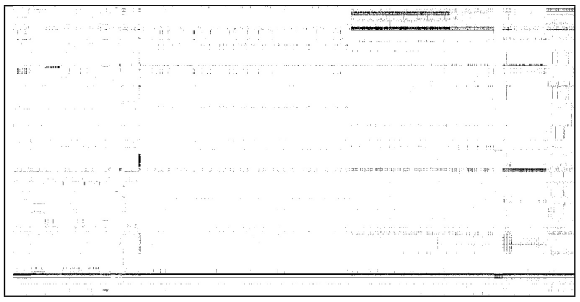

(b) Customer 3 (Cropped)

Figure 1: A black dot indicates a granted permission. Note the well defined clusters of users and permissions in firewall1, and the difference in cluster shape, size, and completeness in customer 3.

Dataset	$\mid U \mid$	$\mid A \mid$	$\mid P \mid$	$\mid UP \mid$	Density
University	493	5	56	3955	0.143
Americas L	3485		10127	185294	0.005
Americas S	3477		1587	105205	0.019
APJ	2044		1146	6841	0.003
Domino	79		231	730	0.040
EMEA	35		3046	7220	0.068
Firewall 1	365		709	31951	0.123
Firewall 2	325		590	36428	0.190
Healthcare	46		46	1486	0.702
Customer 1	311	6 (6)	1105	7868	0.022
Customer 2	881	8 (25)	852	6691	0.009
Customer 3	3068	6 (14)	3133	71596	0.007

Table 1: Sizes of the real-world datasets presented

dot indicates the permission is assigned to the user. The **firewall1** policy is shown in Figure 1(a), and (a crop of) the **customer 3** policy is shown in Figure 1(b). The users and permissions have been clustered using the k-means algorithm with the hamming distance to aid in visualization only.

2.3 Comments and Observations

Before we provide a detailed analysis of what we have learned regarding how access control policies are managed in medium sized organizations, we provide some cursory analysis of the structure, format, and semantics of enforcement.

Parsing and Format.

First, each customer dataset has a unique format, requiring custom parsing and post processing. Most datasets were provided as a set (two or more) comma delimited files. These were typically user-attributes and user-permission relations. In most, each column is a standard key-value pair, e.g., the user's department, title, or job location. In other instances, values were structured data themselves, such as an LDAP distinguished name, and required special parsing.

Noise.

We also encountered several problems identifying a correct set of users, permissions, and attributes for a given set of input files. For example, **customer 3** contains 71943 user-permission assignments, however only 71596 of these are unique. If we further define a permission to be case insensitive, e.g., "SYSTEM," "system," and "SySTeM" are all identical, then the number of unique user-permission assignments is further decreased to 63582. We encountered another problem with the **customer 2** dataset; the user-attribute file contained 1400 users, but only 881 of these are assigned permissions in the user-permission relation.

Joining Accounts.

The **customer 2** dataset was provided as several separate relations between users and accounts, groups, roles, or the corporate hierarchy. This required a small amount of additional effort to join tables and calculate closures to obtain the final set of relations. For example, the company developed a concept of "virtual users" to simplify administration. Virtual users are assigned accounts and permissions on systems, but aren't real employees. Instead, the real user that is the manager of a virtual user is entitled the virtual user's accounts. In this setting, virtual users are similar to roles[4], except each virtual user is assigned to a single real employee. In these instances the transitive closure of permissions assigned to virtual users—and not real users—had to be calculated.

Granularity and Semantics.

Finally, we observed differences in the structure and semantics of permissions. In an academic setting, permissions in access control systems are treated as abstract rights to objects, and this is a definition used in many standards [1]. This is consistent with the public datasets where the data is published as binary relations between users and permissions. The semantics of these permissions are unknown, and the data has been heavily anonymized such that all are abstract statements such as *user i is granted permission j*. From the descriptions of the data we can infer types of access, for example, the firewall policies are likely port-ip pairs, however, this is purely speculative.

The three customer datasets have richer permissions, and raise concerns when applying analysis based on a less-rich model to many real-world datasets and problems. The **customer 1** and **customer 2** datasets manage entitlements at the level of accounts on systems. Because these access control policies provision administrative access to systems, two different administrative accounts on a system may provision the same set of users. For example, if both accounts have `su` access, they are largely equivalent. In many other

[4]Their systems did not natively support RBAC.

settings, a permission at the account level is too coarse grained to be able to make meaningful inferences because it is unknown what permissions two accounts share. Even in the case above, simple questions, such as *are both accounts granted administrative permissions?* become difficult to answer. The **customer 3** dataset differentiates between regular and administrative accounts.

Finer-grained permissions, such as those found in **customer 3**, can add additional complications and ambiguity. This dataset provides the finest-grained and most detailed permissions of our datasets. Recall that permissions are divided into Active Directory and Application permissions, and consider a class of applications, a database management system such as DB2. At the account granularity one can only determine a user has an account on a system. At the application level we may know the system is running an instance of DB2, and the user is granted *some* rights to the database. However, DB2 has its own rich model for access control, and researchers must be able to model such application-level models and their interactions. In a database, objects can be referenced in whole, or in part. For example, a user may be granted SELECT on the database, a particular table in the database, or only a select number of columns on a table. If the user is granted SELECT on each individual column of a table, without the database schema, researchers have no means to infer this level of access is identical to granting SELECT over the table. Further, access rights may be delegated in many applications, or commands executed using the rights of another user using *setuid* on Unix systems or stored procedures in databases.

Looking through the types and granularities of permissions provided in the customer datasets paints a different, and vastly more complicated and intricate, picture of access control than what has been released publicly. This leaves us to conclude that the task of access control analytics is substantially more complicated than the binary user-permission relation model frequently posited in the literature.

3. EXPERIMENTS AND EVALUATION

In this section we will present some of our findings and experience when analyzing publicly available and confidential customer access control datasets. The number of datasets we have analyzed is still small, and we have thus far not been able to draw and broad generalizations regarding how access is provisioned in a small to medium organization. Each dataset has unique characteristics, varying levels of sparseness, and skewed distributions of permissions and attributes to users.

3.1 Distributions of Permissions

In this section we analyze the distribution of permissions and find the data has a long tail distribution, and our customer datasets more closely resemble a power law distribution.

For the distribution of permissions to follow a power law, the number of users assigned a permission must vary as a power of its rank. We plot the probability mass and cumulative distribution of several datasets in Figure 2; if the data is distributed as a power law, it should appear as a linear line when plotted on a log-log scale. Many public datasets are not distributed as a power law, and have a more *stair step* appearance. Permissions are often assigned to users in groups—a motivation behind RBAC. Each permission in a role is assigned a similar number of times (to the users assigned the role), causing a horizontal line in the probability mass function. After the last permission in the role is considered, there is a sudden drop to the next most frequent permission, or the next role. The **customer 3** dataset, on the other hand, more strictly resembles a power law, a downward slope of -1.28 with little variation. Even the **americas** datasets, those most closely resembling **customer3**, has an initial step for the most frequent permissions.

The Cumulative distribution function illustrates the common *80–20 Rule* that accompanies many power law distributions. For an access control system, this implies around 80% of the user-permission assignments come from 20% of the permissions, but actually refers to any $k\%$–$(100-k)\%$ division where $k > 50$. Larger values of k are an indication most of the policy corresponds to a small number of permissions, and the remaining permission relate to a small number of users. From a security standpoint, the most frequent permissions may correspond to less sensitive resources, and over assigning them to users may have a less significant negative impact. However, the remaining permissions are rare, potentially pertaining to the most sensitive resources, and demanding the most attention from administrators to ensure correct assignments and proper auditing. We plot the **customer** datasets against the **americas** datasets, as those are the public data that most closely resemble a power law distribution.

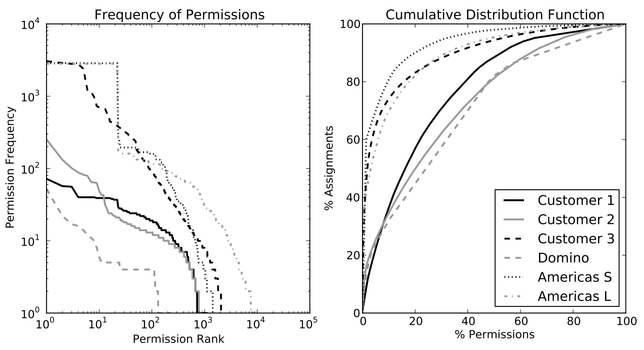


Figure 2: Probability mass function and cumulative distribution of size datasets illustrating the long tail and power law distribution.

3.2 Rank and Clusters

Because several of these datasets have been released to facilitate research on role mining, we perform some role-mining based analysis. Administration of large access control policies can be a challenging task, especially as the number of users and permission in an organization begins to grow. Role-based access control (RBAC) is a popular model that reduces administration costs by placing an intermediate entity, a role, between users and permissions. Instead of assigning permissions directly to users, permissions are first assigned to roles, and users are authorized roles. Users are thus provisioned any permissions assigned to roles they are authorized for. If there are n users and m permissions, RBAC can reduce the size and complexity of the policy from $O(nm)$ to $O(n+m)$, the size of the user-role and role-permission relations.

There has been extensive research in applying role mining techniques to existing access control policies [4,9,11,17]. The goal of this section is not to provide extensive background on role mining, but to use common role mining metrics, such as role minimization, as an analysis to contrast the customer and public datasets. We use two measures, an approximation of the minimum number of roles from [4] and the number of formal concepts [7, 11], as indications of the complexity and structure of the data. Role minimization indicates the minimum number of roles required to exactly represent a given policy; the fewer the number of roles, the more easily the data can be expressed. Concepts are a popular construction in role mining, and can be described as a role such that no users or permissions may be added without violating the input user-permission relation, i.e., adding more users or permissions to the role would grant rights not previously held. It can easily be shown that the minimal set of roles is a subset of the formal concepts. The number of formal concepts also provides a good indication for how much structure there is in the data. Few concepts imply the data is simple and well structured with a small number of factors used when making provisioning decisions. Data that contains many concepts may be especially noisy, or contain ad personam permissions, and there may exist many clusters of users or factors influencing how users have been provisioned. The total number of concepts is bounded by $2^{\min(|U|,|P|)}$.

The results from our analysis are given in Table 2. We can see that, with the exception of the **emea** dataset which contains many ad personam permissions, the minimum number of roles required to fully represent the user-permission relation is around two times greater for the customer datasets than the publicly available dataset, and as high as 18.6 times greater between the **firewall2** and **customer 3** datasets. A more significant and revealing difference is the number of concepts, which is representative of the complexity of the data. Again, the **emea** dataset stands out as an exception in the public data and the **customer 3** dataset produces substantially more concepts than the other datasets. This analysis quantifies the difference that is easily observed when comparing between Figure 1(a) and Figure 1(b).

This final analysis implies real datasets we have observed are more complex, with higher amounts of entropy, and potentially more complex and fractured roles. This makes the datasets more difficult to describe and summarize mathematically. It is also an indication the real-world data contains more noise, such as over- and under-assigned users or a more precise access control policy. For example, it is possible the public datasets are either easier to express, or are more resilient to over-assigning permissions to users to keep the policies simple, compact, and easy to express and analyze for the administrators.

4. ANALYSIS OF ATTRIBUTES

There has been a recent trend towards attribute-based access control systems, such as the XACML standard. In these systems attributes describing users, such as job title or generic tags, uniquely determine if a request is valid. This is in contrast to identity based access control where entitlements are assigned directly to users. Even in identity based systems, a desirable property for many medium to large organizations is the ability to attribute authorizations, such as roles, to attributes of users [8,13], which allows the organization to more efficiently audit policies and maintain regulatory compliance. The customer datasets have produced mixed results when using existing techniques to learn a joint model of the user-attribute and user-permission relations. In this section we analyze the distribution of attributes, and how well these aid in analyzing permission assignments.

Dataset	Users	Roles	R/U	Concepts	C/(U,P)
University	493	17	0.034	33	0.589
Americas L	3485	421	0.121	36991	10.614
Americas S	3477	317	0.091	2764	1.742
APJ	2044	477	0.233	798	0.696
Domino	79	27	0.342	73	0.924
EMEA	35	34	0.971	780	22.286
Firewall 1	365	78	0.214	317	0.868
Firewall 2	325	12	0.037	22	0.068
Healthcare	46	14	0.304	31	0.674
Customer 1	311	127	0.408	1008	3.241
Customer 2	881	316	0.359	3042	3.570
Customer 3	3068	2115	0.689	400235	130.455

Table 2: Approximate minimum number of roles and structure of the data.

4.1 Entropy of Attributes and Permissions

When analyzing attributes, it was discovered many may be noisy, and obtaining non-anonymized data is important for analysis. For example, in one dataset there was an attribute for the country of the user. Here, values included "United States," "USA," "US," and "United States of America." If user-attributes had been anonymized in a similar manner to the public user-permission data, such errors could not be easily detected. In the remainder of this section we perform a minimum of cleaning the attributes by converting all attribute values to lower case.

The analysis of attributes have previously been used to increase the semantic meaning of roles when role mining [3, 6, 11] and noise detection [13]. To measure the significance of attributes we use the entropy reduction [6,13] $I(p_j, A)$ of a permission p_j's assignments given knowledge of a user's attribute A:

$$I(p_j, A) = h(p_j) - h(p_j \mid A)$$

where $h(p_j)$ is the entropy of the permission p_j,

$$h(p_j) = -\sum_{s\in\{0,1\}} \Pr[p_j = s] \log_2 \Pr[p_j = s]$$

and $h(p_j \mid A)$ is the entropy of p_j conditional on the value of the attribute A,

$$h(p_j \mid A) = -\sum_{a\in A} \Pr[A = a] \sum_{s\in\{0,1\}} \Pr[p_j = s \mid A = a] * \log_2 \Pr[p_j = s \mid A = s].$$

We analyze how well each attribute reduces the uncertainty of permission assignments on the three customer datasets, and discard any attribute value that is assigned to fewer than ten users. To ease comparison, we normalize the entropy reduction by the number of possible assignments, $|U| * |P|$. For each attribute, the cardinality is the number of unique values, and $H(A)$ is the total entropy in the

Attribute	Cardinality	$H(A)$	$\parallel \sum I(\cdot, A) \parallel$
—	—	—	$2.436 * 10^{-2}$
Type	4	170.8	$7.705 * 10^{-3}$
Country	12	250.9	$1.496 * 10^{-2}$
Category	15	258.5	$1.401 * 10^{-2}$
l	48	297.9	$1.288 * 10^{-2}$
Dept	79	328.8	$1.345 * 10^{-2}$
Title	249	319.0	$3.387 * 10^{-3}$

Table 3: Entropy of Customer 1

Attribute	Cardinality	$H(A)$	$\parallel \sum I(\cdot, A) \parallel$
—	—	—	$9.226 * 10^{-3}$
C	2	8.14	$6.454 * 10^{-6}$
G	2	72.70	$4.263 * 10^{-4}$
B	6	97.96	$7.820 * 10^{-4}$
A	48	845.86	$7.370 * 10^{-3}$
M	53	816.69	$5.131 * 10^{-3}$
H	254	878.00	$3.663 * 10^{-3}$
X	756	858.61	$4.033 * 10^{-17}$
L	849	857.27	$4.661 * 10^{-17}$

Table 4: Entropy of Customer 2

attribute assignment in bits. The results are shown in Tables 3–5 for **customer 1**, **customer 2**, and **customer 3**, respectively. To help further analyze the results of our entropy calculations, we also plot the distribution of the twenty most frequent values for each attribute in Figures 3–5. If there are more than twenty values for each attribute, the tail is aggregated into a final black bar.

We expect attributes that cause significant decreases in the entropy of permission assignments, and have a low entropy themselves, are likely useful for making access control predictions and attributing authorizations. Attributes with high entropy, or a high cardinality, are unlikely to be useful. In [13] it was illustrated how an attribute that significantly reduces the entropy of permission assignments, such as the user's last name, may be semantically meaningless from a security context. If the data were too heavily anonymized, including the attribute name, it may be difficult to make such a distinction.

There are several interesting observations we can draw from the entropy of user-attributes and the entropy of permission assignments conditioned on attributes. For example, consider the C and G attributes from the **customer 2** dataset. Both are binary variables, yet the G attribute has almost 70 times more entropy for the dataset, and decreases permission uncertainty by two orders of magnitude more. By looking at the distribution of these attributes in Figure 4, we can see there are very few users with the second attribute type. Other interesting properties are found in **customer 2**, such as the distribution of attributes, X, H, and L, which are almost completely contained within the long tail.

4.2 Prediction Using Attributes

In the previous section we use the measure of the entropy reduction from Frank et al. [6] to predict which attributes will provide the best predictive performance. Previously,

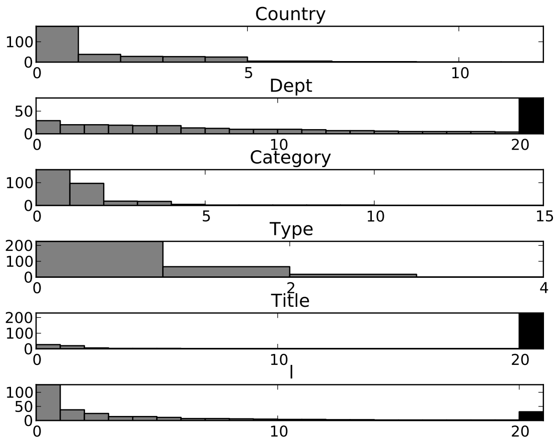


Figure 3: Customer 1 User Attribute Distributions

Attribute	Cardinality	$H(A)$	$\parallel \sum I(\cdot, A) \parallel$
—	—	—	$7.181 * 10^{-3}$
Contractor	2	592.1	$1.02 * 10^{-3}$
Organization	12	3132.6	$5.87 * 10^{-3}$
Level	17	618.2	$1.50 * 10^{-3}$
Location	53	3278.4	$6.85 * 10^{-3}$
Dept	192	3195.8	$6.01 * 10^{-3}$
Manager	298	3152.2	$4.87 * 10^{-3}$
Title	525	3184.1	$4.53 * 10^{-3}$

Table 5: Entropy of Customer 3

Molloy et al. [13] illustrated how to use user-attributes to detect errors in access control policies using collective matrix factorization [14], a process that involves learning both a factorization of the user-permission and the user-attribute relations such that they share a common factor. In matrix decomposition, one decomposes a matrix Y, such as the user-permission relation, into two matrices such that $Y \approx AB^T$. If A and B are binary, these are the user-assignment and permission-assignment relations in RBAC, otherwise they have limited semantic utility. In collective matrix factorization we have a second (or more) matrix X, such that X and Y share a dimension (in this instance, the users). We wish to decompose both matrices such that they share a common factor, $X \approx CA^T$ and $Y \approx AB^T$, where the factor A is shared. When decomposing a single matrix, one is interested in minimizing a distance function $D(Y \parallel AB^T)$, such as the Frobenius norm in singular value decomposition. When we have two (or more) relations, we minimize a linear combination of their losses, $\alpha D(X \parallel CA^T) + (1 - \alpha)D(Y \parallel AB^T)$. Here α is a mixing parameter of the importance of reconstructing X versus Y.

Using collective matrix factorization, we can find a good value of α and balance the utility of reconstructing the user-permission relation; from a security standpoint, this is the most important relation. We use logistic PCA to reconstruct both the user-permission and user-attribute relations, which have been converted to binary relations. We hold out 20% of the data and measure the mean absolute error (MAE) of the zero-one loss for reconstructing the binary relation. This

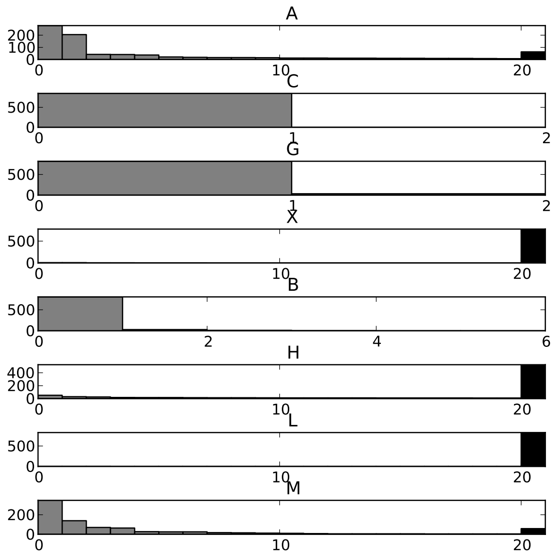


Figure 4: Customer 2 User Attribute Distributions

measures how well we can predict missing values, and how well we can model the access control policies. The results, varying the mixing parameter α, are shown in Figure 6, and the improvement in predictive performance is given in Table 6. There is a correlation of 0.68 between $I(\cdot, A)$, and the predictive improvement.

Attribute	Predictive Improvement	Std. Dev.
Manager	20.34%	0.14
Department	25.41%	0.24
Title	15.03%	0.02
Location	21.44%	0.14
Organization	18.53%	0.51
Level	18.59%	0.1
Contractor	12.25%	0.24

Table 6: The total entropy of the user-permission relation given knowledge of a user's attribute.

While collective matrix factorization worked well for the **customer 3** dataset, and key attributes, such as *manager*, *department*, and *location* were able to decrease the reconstructive error by over 20% each, we did not observe equally impressive results for the other datasets. On the **customer 1** dataset, if fact, we did not observe a decrease in reconstruction error by adding user-attributes. The MAE loss for both the department user-attribute and user-permission relations using 15 latent variables are shown in Figure 7. We observed similar behavior for the other attributes and saw no remarkable improvement by either increasing or decreasing the number of latent variables.

Similar behavior has been observed in the context of role mining when trying to find semantically meaningful roles. For example, Frank et al. [6] select roles with high attribute compliance from roles generated with multi-assignment clustering [16]. Their experiments on a real-world dataset from

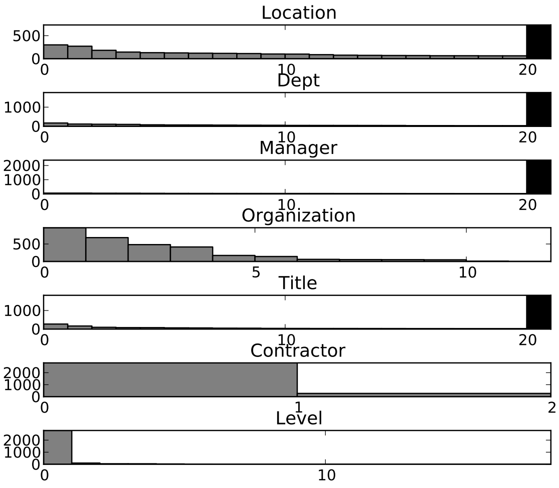


Figure 5: Customer 3 User Attribute Distributions

a bank found that by increasing the attribute compliance of mined roles increased the number of errors, over- and under-assigning permissions to users.

One possible explanation is that collective matrix factorization (and the method in [6]), as applied above, uses a linear model of the user attributes. For example, a user's department or title may be used to determine factors influencing assignments, but cannot express preconditions such as "Department = 'Research' AND Title = 'Manager'." It may be possible to observe better predictive performance by applying a kernel trick at the expense of increasing the effective number of attributes. How best to model access control policies leveraging user-attributes remains an open problem and requires additional real-world datasets to validate new models.

Another hypothesis is that **customer 3** is more strict with how it manages its access control policies and ensures all data is accurate and up-to-date. The industry in which **customer 3** operates imposes more laws and regulations governing its IT infrastructure than either **customer 1** and **2**. Further, some of **customer 3**'s operations have greater security needs and requirements, and this in turn may result in user entitlements being traced back to user attributes more effectively for auditing purposes.

Finally, we should note that just because user attributes result in a decrease in entropy, does not imply the permis-

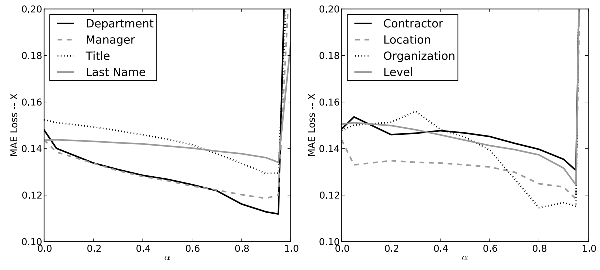
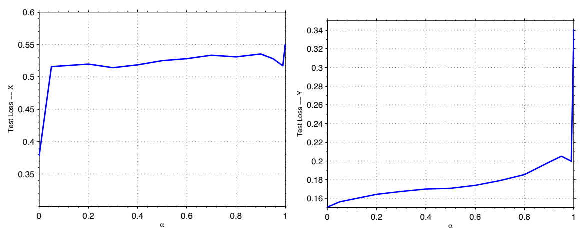


Figure 6: The MAE for the user-permission relation X given several attributes.



(a) User-Attribute (b) User-Permissions

Figure 7: Collective Matrix Factorization of customer 1 with 15 latent variables.

sions should result in an measurable decrease in reconstructive error. Matrix factorization, unlike the entropy calculations, is impacted by the assignments of *other* permissions. For example, user i with permission j may be granted a right k because many users with j also have k. Thus, the conditional entropy of an attribute A on a permission j should really be measured $\sim H(p_j \mid P_{\setminus\{p_j\}}) - H(p_j \mid P_{\setminus\{p_j\}}, A)$.

5. CONCLUSIONS

This paper examined eleven real-world datasets of access control policies typically used for research in role mining, access control analytics, and policy learning. Of these, eight datasets a public and have been used in the academic literature for research and evaluation, and three are confidential datasets from clients. Analysis indicated the public data contains more distinct clusters of users and permissions, making it well suited to role-based access control, and easier to learn models in general. To contrast, the client datasets were sparser on average, were more fragmented (as calculated by the number and ratio of formal concepts), and less compressible using roles. Because the public data lacks semantic meaning, it limits the types of analysis we can perform on them.

Next, we more closely analyzed the client data which contains semantic information about both the users and permissions. We found the permissions to have a longer tail, and the **customer 3** dataset follows a power law distribution. The sparseness of these datasets leads us to speculate there are ad personam permissions, e.g., rights to a user's home directory, that should be parameterized. This would simplify the complexity and analysis of such datasets and cut the long tail.

We have also discovered the need to develop a normalized model of an entitlement in access control policies. As illustrated in Section 2.3, entitlements may appear at different levels of granularity, such as rights to a column, table, or database, and entitlements may subsume others, e.g., `sudo`. To perform powerful analytics on the rights assigned to users, it is necessary to have an accurate and uniform model of such entitlements. In the database example, this requires knowledge of the objects themselves, while administrative access requires additional information about the relationship between rights.

While performing our research, a missing element to access control data is feedback from administrators who understand the policies and the constraints they were authored under. This includes quality measure for mined roles, validation of discovered policy errors, and comments on automatically suggested role names. During our engagements we received feedback such as "This seems to look good," without further input.

We hope this work helps open up new data sources that can drive future research. For example, future work may model how roles evolve over time, how users use permissions, or separation of duty policies. In general this line of research is attempting to discover a meta-model of access control, a topic which has garnered discussion recently [2, 5]. Further, obtaining multiple datasources than can be correlated, such as network traces, email, and permission usage patterns, can allow researchers to model the interaction of systems and its users, possibly leading to solutions to insider threats.

6. REFERENCES

[1] ANSI. Role-based access control. Technical Report ANSI INCITS 359-2004, 2004.
[2] S. Barker. The next 700 access control models or a unifying meta-model? *SACMAT*, 2009.
[3] A. Colantonio, R. Pietro, A. Ocello, and N. Verde. A formal framework to elicit roles with business meaning in rbac systems. *SACMAT*, 2009.
[4] A. Ene, W. Horne, N. Milosavljevic, P. Rao, R. Schreiber, and R. E. Tarjan. Fast exact and heuristic methods for role minimization problems. *SACMAT*, 2008.
[5] D. Ferraiolo and V. Atluri. A meta model for access control: why is it needed and is it even possible to achieve? *SACMAT*, 2008.
[6] M. Frank, A. Streich, D. Basin, and J. Buhmann. A probabilistic approach to hybrid role mining. *CCS*, 2009.
[7] B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. 1998.
[8] C. Giblin, M. Graf, G. Karjoth, A. Wespi, I. Molloy, J. Lobo, and S. Calo. Towards an integrated approach to role engineering. *SafeConfig*, 2010.
[9] M. Kuhlmann, D. Shohat, and G. Schimpf. Role mining - revealing business roles for security administration using data mining technology. *SACMAT*, 2003.
[10] R. Kumar, S. Sural, and A. Gupta. Mining RBAC roles under cardinality constraint. *Information Systems Security*, 2010.
[11] I. Molloy, H. Chen, T. Li, Q. Wang, N. Li, E. Bertino, S. B. Calo, and J. Lobo. Mining roles with semantic meanings. *SACMAT*, 2008.
[12] I. Molloy, N. Li, T. Li, Z. Mao, Q. Wang, and J. Lobo. Evaluating role mining algorithms. *SACMAT*, 2009.
[13] I. Molloy, N. Li, J. Lobo, Y. A. Qi, and L. Dickens. Mining roles with noisy data. *SACMAT*, 2010.
[14] A. P. Singh and G. J. Gordon. Relational learning via collective matrix factorization. *KDD*, 2008.
[15] S. D. Stoller, P. Yang, C. R. Ramakrishnan, and M. I. Gofman. Efficient policy analysis for administrative role based access control. *CCS*, 2007.
[16] A. P. Streich, M. Frank, D. Basin, and J. M. Buhmann. Multi-assignment clustering for boolean data. *ICML*, 2009.
[17] J. Vaidya, V. Atluri, and J. Warner. Roleminer: mining roles using subset enumeration. *CCS*, 2006.

On Collection of Large-Scale Multi-Purpose Datasets on Internet Backbone Links

Farnaz Moradi, Magnus Almgren, Wolfgang John, Tomas Olovsson, Philippas Tsigas
Computer Science and Engineering
Chalmers University of Technology
Göteborg, Sweden
{moradi,almgren,johnwolf,tomasol,tsigas}@chalmers.se

ABSTRACT

We have collected several large-scale datasets in a number of passive measurement projects on an Internet backbone link belonging to a national university network. The datasets have been used in different studies such as in general classification and characterization of properties of Internet traffic, in network security projects detecting and classifying malicious traffic and hosts, and in studies of network-level properties of unsolicited e-mail (spam) traffic. The *Antispam* dataset alone contains traffic between more than 10 million e-mail addresses.

In this paper we describe our datasets, the data collection methodology including experiences in collecting and processing data on a large scale. We have in particular selected a dataset belonging to an anti-spam project to show how a practical analysis of highly privacy-sensitive data can be done, in this case containing complete e-mail traffic. Not only do we show that it is possible to collect large datasets, we also show how to solve different issues regarding user privacy and give experiences from how to work with large datasets.

Categories and Subject Descriptors

C.2.3 [**Network Operations**]: Network Monitoring; C.2.2 [**Network Protocols**]: Applications (SMTP, FTP, etc.)

General Terms

Measurement

Keywords

Internet Measurement, Large-Scale Datasets, E-mail traffic, Spam

1. INTRODUCTION

Access to real-life large-scale datasets is in many cases crucial for understanding the true characteristics of network traffic and application behavior. The collection of large datasets from backbone Internet traffic is therefore very important for such analysis although the data collection projects in themselves face several challenges [13]. Not only is mere physical access to optical Internet backbone links needed, but also rather expensive equipment in order to deal with the large data volumes arriving at high speeds. Adding to the complexity, the collected data traces must be desensitized due to privacy issues because they may contain privacy-sensitive data. This anonymization process must be done in such a way so that a satisfactory analysis to answer the research question still can be performed, without leaking any sensitive user data. Packets also need to be reassembled into application level "conversations" so that, finally and maybe the most challenging part, methods and algorithms suitable for analysis of massive data volumes can be run. Finding these scalable methods is difficult.

We have over the years performed several data collection projects where large datasets have been gathered and analyzed. Different projects have had different goals with the data collection and for each project, unique tools have been developed and used. In this paper we describe the data collection procedure and the challenges we have faced with dealing with high-speed data collection and give examples of how data have been used in different projects. In particular, we describe a current project, the Antispam project, aiming for spam detection mechanisms on the network level where characteristics of SMTP traffic are collected and analyzed. Not only does this involve collection of vast amounts of e-mail traffic but the data collected is also highly sensitive so that automated ways to handle message privacy are essential. We also describe a method that could be deployed for analyzing the large-scale *Antispam* dataset. This method allows us to find distinguishing characteristics of legitimate and unsolicited e-mails which could be used for complementing current anti-spam tools.

The rest of this paper is organized as follows. Section 2 presents our methodology for data collection, including challenges we encountered and the solutions we deployed. Section 3 introduces the large-scale datasets collected during different years for different projects. Section 4 describes the collection of a particular dataset, the Antispam dataset, and in Section 5 we shift focus to describe the analysis of this Antispam dataset and how we compare unsolicited with legitimate e-mails. In Section 6 we present related work by comparing other sources of data collection with our collection method and resulting datasets. Finally, Section 7 con-

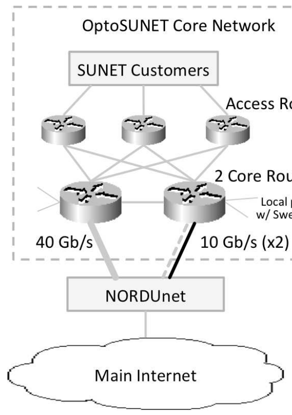


Figure 1: OptoSUNET core topology. All SUNET customers are via access routers connected to two core routers. The SUNET core routers have local peering with Swedish ISPs, and are connected to the international commodity Internet via NORDUnet. SUNET is connected to NORDUnet via three links: a 40 Gbps link and two 10 Gbps links. Our measurement equipment collects data on the first of the two 10 Gbps links (black) between SUNET and NORDUnet.

cludes the paper.

2. DATA COLLECTION METHODOLOGY

In this section, we describe the current measurement setup used to collect data, as well as some of the challenges encountered and our solutions to these. It is vital to understand the underlying data collection platform to correctly be able to use the resulting datasets. As with any experimental platform, problems do occur, but as we show below, these can sometimes be compensated for either in the collection phase or in the analysis stage.

2.1 Current Measurement Setup

We collect backbone traffic on an OC-192 (10 Gbps in each direction) link in the core-backbone of SUNET, the Swedish University Network (SUNET),[1] which not only serves as a backbone for university traffic but also for a substantial number of student dormitories, research institutes, as well as some museums and government agencies. It contains a large amount of exchange traffic with commercial companies.

Its current version, *OptoSUNET*, is a star structure over leased fiber, with a central exchange point in Stockholm. OptoSUNET connects all SUNET customers redundantly to a core network in Stockholm, as depicted in Figure 1. Traffic routed to the international commodity Internet is carried on three links between SUNET and NORDUnet, where NORDUnet peers with Tier-1 backbone providers, large CDNs (Content Distribution Networks) and other academic networks.

We use two high-end rack mounted systems (Linux) as measurement systems, one for outgoing and one for incoming traffic. At the core network in Stockholm, we apply optical splitters to tap the two OC-192 links. Each optical splitter, tapping either the inbound or outbound OC-192 link, is attached to an Endace DAG6.2SE card in one of the measurement nodes. The cards are capable of collecting data on PoS and 10 Gbit-Ethernet links with bandwidths of up to 10 Gbps. We usually collect network data simultaneously for both directions.

Depending on the project (see Table 1) and its goal with the data collection, we then perform some pre-processing of the raw data before transferring them for further analysis and storage at the processing platform at Chalmers University. This pre-processing ranges from *anonymization* (see Section 2.2.3) of the data (all projects), to spam categorization (the *Antispam* project). The experimental infrastructure is further described in [7].

2.2 Challenges and Solutions

We categorized the problems we encountered and our solutions in three clusters, *general problems* relating to the setup of the system, problems related to the *collection process*, and finally problems related to the *pre-processing of the dataset* before the final analysis and storage.

2.2.1 General Challenges

One of the most difficult problems we faced was actually not of a technical nature, but involved gaining access to the network infrastructure in the first place. Fortunately, there is a long tradition of work between our department and SUNET, so there was already a basis for trust. Furthermore, we also consulted an ethical vetting board, and based on their feedback we could proceed with the measurements. However, as will be described below, the required *anonymization of user data* is very important and permeates many of our decisions on what kind of data we can collect and how it can be analyzed.

A more technical problem involved the equipment. At the time of purchase (2004), we faced problems with finding systems with enough internal bus bandwidth to cope with full link-speed data collection. Captured data should be received by the network card, be moved to main memory, and then be written to disk in speeds up to about 1 GB/s. The used high-end RAID system with six striped disks offered around 0.4 GB/s disk throughput, which turned out to suffice due to the large over-provisioning of the 10 Gbps link by SUNET. The network architecture changed somewhat during 2007 when one (parallel) 40 Gbps and one 10 Gbps link were added. Unfortunately, equipment for collecting data from all links was too expensive to acquire for our projects (each direction would require 5 times as much hardware).

Finally, there are limitations in using real-life datasets, that are not specific to our datasets. To mention one is that the measurements give us snapshots of traffic from a single vantage point during a limited time period. The results should thus be extended with similar data from other times and locations.

2.2.2 Collection Challenges

Regarding the collection phase of large datasets, the first problem we must cope with is the sheer volume of traffic.

[1]`http://www.sunet.se/`

At heavily loaded links, data may arrive with up to 1 GB/s. Even if this theoretical maximum is rarely reached due to over-provisioning of the links, different data reduction methods must be applied to further decrease the load on the measurement nodes. However, these methods must not influence the real-time nature of the traffic capture, i.e. we must be able to keep up with the traffic. We partially solved this by having very well-defined experimental plans with clear goals of exactly what we should capture to do our analysis.

For example, for datasets spanning long time intervals, we only capture flow summaries instead of individual packets, while we collect short snapshots (10 or 20 minutes) of packet traces to investigate protocol properties. To allow for a more dynamic traffic capture, we currently investigate real-time computations on the DAG cards so that decisions can be taken in real time based on more complex traffic properties.

Moreover, any dataset we collect and analyze should be safely stored so that others can repeat the analysis or compare results. This poses additional archiving requirements on the measurement system.

To ensure sanity of the resulting data, we apply consistency and sanity checks immediately after collection, allowing us to document both measurement related problems (e.g. measurement card failures) and network-related anomalies (e.g. network maintenance tasks by SUNET). These include inspection of time stamps, frame types, packet header information, etc. With these sanity checks we can improve the system in the longer term, but they can also be applied during the analysis phase to explain certain traffic behavior. For sanity checking, we use existing tools such as CAIDA's CoralReef [15] and Endace's dagtools. Additionally, we developed our own software and modified publicly available software to suit our needs. The use of our own methods and programs requires substantial effort, but gives us complete control of the quality of the data.

As an example of trace insanity, we have experienced some cases of completely garbled data, most likely occurring due to hardware problems in the DAG cards loosing framing synchronization. These traces have been discarded immediately. To reduce this problem we now restart the cards in regular intervals, which in turn may lead to some missing packets in the second between such data collection periods. We are currently installing new equipment, including new DAG cards, which should eventually solve this issue.

During normal operation, we have not detected any packet loss. However, during the collection of one of the datasets, the Malbone dataset, there have been a few short, but immense traffic surges, where traffic was increasing from the normal rate of $<200k$ to $>400k$ packets per second. During these surges, our nodes could not keep up with the speed and dropped packets, which was logged by the measurement cards. As the information was logged, it was easy to accommodate in the analysis stage. We have also detected some minor errors with the IP header checksums (1 out of 300 million frames) and 1 out of 191 million frames were discarded due to receiver errors.

Finally, the measurements are done over an operational large network, meaning that parameters change over the course of the data collection, both in a longer perspective with planned upgrades as well as with transient failures of certain equipment.

2.2.3 *Pre-processing Challenges*

As can be seen from Figure 1, we collect data from one link (out of three) with two separate systems to collect traffic in two directions, meaning that we have two datasets with unidirectional traffic traces. The traffic is load balanced between the links and, according to SNMP statistics collected during the last measurement campaign, we see around one third of all incoming traffic and 15% of all outbound traffic. This effect introduces an observed routing asymmetry, as we can sometimes only see the traffic going in one direction of a TCP connection [8].

Assembling bidirectional TCP flows requires very good time synchronization between the two systems. During measurements, the DAG cards are thus synchronized with each other using DUCK Time Synchronization [4], allowing synchronization within $\pm 30ns$, which suffices for trace assembly.

A key processing step is also desensitization of the data, i.e. removing any privacy-sensitive information. Besides our responsibilities as ethical researchers, this is also one of the requirements of the ethical vetting board (see Section 2.2.1). As a basic step, we discard sensitive payload and anonymize IP addresses in the resulting trace based on the prefix-preserving CryptoPAN [21]. Throughout all measurements campaigns we use a single, unique encryption-key to allow us to track specific hosts and IP ranges between all measurements. Unfortunately, this anonymization step is not without penalty but may influence the type of analysis method we can use as well as restricting the refinement of the result. As an example, even if we find a very aggressive host spreading malware within SUNET, we cannot inform the owner due to the anonymization of the data. Furthermore, there still exists challenges to improve the anonymization before datasets potentially could be shared to others on a large scale. Other desensitization steps performed in projects include payload removal (MonNet), and e-mail anonymization (Antispam), which is detailed further in Section 4.3.

2.2.4 *Summary*

To summarize, our analysis is network centric and does not consider the end hosts in detail. Overall, it was not a trivial effort to initially setup the data collection platform but it took both time (years) and effort. As with any experimentally collected data, it is important to understand the limitations of the experiment setup for correct analysis of the data. Some problems, given their careful documentation in the collection phase, can be accommodated for in the analysis stage.

3. DATASETS

We have collected several large-scale datasets on the Internet backbone links. The first traces were collected in 2005 and we still have active projects collecting data from the links. The datasets differ in the information they contain, reflecting the type of research question we want to investigate. As we detailed in Section 2, we simply cannot collect

Table 1: Datasets Overview

Dataset	Location	Collection Period	Collection Duration	Number of traces	Number of Packets $/10^9$
MonNet I	GigaSUNET	2006-04	20 minutes	74	10.8
MonNet II	GigaSUNET	2006-09 to 2006-11	10 minutes	277	27.9
MonNet III	OptoSUNET	2008-12 to 2009-11	10 minutes	151	33.0
Malbone	OptoSUNET	2010-03 to 2010-10	24 hours	34	12 (flows)
Antispam	OptoSUNET	2010-03	24 hours	14	0.8 (SMTP only)

"all" information but need to reduce it both in regard to storage as well as for anonymization purposes.

The datasets are summarized in Table 1. Below we briefly describe the characteristics of each dataset. We then describe the collection and use of the *Antispam* dataset in detail as a case study.

3.1 The MonNet Datasets

The largest datasets until today were collected within the MonNet project to classify and understand the characteristics of Internet traffic and to see how it changes over time. Later analysis of this dataset also included finding malicious traffic in order to see how and to what extent protocols are abused.

The MonNet datasets represent 95 hours of backbone traffic, collected on 156 different days mainly during 2006 and 2009. The first set of data was collected during 80 days from September to November 2006 as 277 randomly selected 10-minute snapshots. When recording these traces, payload beyond transport layer was removed. About 27.9 billion IPv4 frames containing around 480 million flows were collected and analyzed. The size of the dataset was almost 20 TB in size (headers only). A second (slightly larger) dataset was collected during 2009 where 33 billion packets were collected. Traffic analysis from this data set reveals inbound traffic from $2,270,000$ distinct IP addresses to $360,000$ unique internal addresses.

The datasets collected in the MonNet project have been studied in detail. Initial studies investigated protocol features of packet headers [10] and packet header anomalies in order to discuss potential security problems, such as incorrect use of IP fragmentation [9]. Additional flow-level analysis of the MonNet data allowed investigation of trends and changes in connection behavior of Internet traffic over time, e.g. how the popularity of p2p traffic has caused a change in Internet traffic patterns in the last few years [11, 12, 22].

3.2 The Malbone Dataset

The objective of the Malbone project is to measure and understand larger communication patterns among hosts over a longer time period. This may include normal as well as more malicious behavior.

For more than six months, a 24h snapshot of all flows was regularly collected once a week. The dataset contains a total of 12 billion flows for both directions. In Table 2, we have summarized all unique IPs we found during a single collection day to give an idea of the scale of the traffic passing by the measuring point.

This dataset also contains metadata, including, for example, hosts known to aggressively spread malware at the time of the collected snapshots. By using the flow data together with this information, we can then make more targeted types of analysis of hosts, despite their addresses being anonymized.

The analysis of this dataset is still in its infancy, but some results documenting malicious behavior of scanning hosts has been published as well as particulars of the timing behavior of hosts. [1].

3.3 The Antispam Dataset

In the Antispam project SMTP traffic was collected to permit the study of the differences in traffic characteristics between spam and legitimate traffic. The goal is to find methods for early detection of spamming nodes on the Internet as close to the source as possible. This method should be an alternative to spam removal in the receiving hosts. There is a clear need for moving the defense against spam as close to the spammers as possible, in order to reduce problems such as the amount of unwanted traffic and waste of mail server resources.

Within this project, during 14 days in March 2010, more than 797 million SMTP packets (filtered on TCP port 25 in the hardware) were passively captured. More than 627 million packets were incoming packets to SUNET and the rest were outgoing. We aggregated these packets into 34.9 million incoming and 11.9 outgoing SMTP flows. The captured flows on the incoming direction were originating from $2,300,660$ distinct IP addresses and were destined to $569,591$ internal distinct IP addresses. The outgoing flows were sent from $10,795$ to $1,943,919$ distinct IP addresses.

The main challenges in this project relate to the highly privacy-sensitive data as well as how to analyze the characteristics of this type of traffic on a large scale. This project is described more in detail in Section 4.

4. ANTISPAM DATASET COLLECTION

In this section we use the dataset *Antispam* as a case study to concretely illustrate the collection and analysis of a large-scale dataset. Such an e-mail dataset can be studied for better understanding of the behavior of spam traffic, often a means to propagate malicious content. Research such as [19] has suggested that spam mainly originates from botnets. These botnets are also most likely active in other malicious activities on the Internet. Therefore, detecting spam close to its source instead of just discarding it by the receivers can also lead to detection of other malicious traffic from the same origin.

4.1 SMTP Data Collection

In order to analyze characteristics of e-mail traffic, SMTP packets were passively collected during two consecutive weeks of measurements in March 2010.

Table 2: Unique hosts during the data collection 2010-04-01

	Inside SUNET		Outside SUNET	
Incoming Link	Destination IPs	970,149	Source IPs	24,587,096
Outgoing Link	Source IPs	23,600	Destination IPs	18,780,894

To overcome the storage problem described in Section 2.2.2, we used a hardware filter to only capture traffic to and from *port 25* using the *crl_to_dag* utility of CoralReef [15]. This still resulted in more than 183 GB of SMTP data, divided into two unidirectional datasets (see Section 2.2.3).

The captured packets belonging to a single flow were then aggregated to allow the analysis of complete SMTP sessions. To reconstruct the sequence of packets into flows, we used the tcpflow program,[2] which understands sequence numbers and correctly compensates for problems such as out-of-order packets and retransmitted packets.

The collected data contains both TCP flows with destination port 25 (*SMTP request*) and TCP flows with source port 25 (*SMTP reply*). As each *SMTP request* flow corresponds to an SMTP session, it can carry one or more e-mails; thus we had to extract each e-mail from the flows by examining the SMTP commands. The resulting extracted e-mail transaction contains the (1) SMTP commands containing the e-mail addresses of the sender and the receiver(s), (2) e-mail headers, and (3) the e-mail content. Each *SMTP reply* contains the corresponding response code to an SMTP request command, and by also including these in the analysis one can gain a better insight into the behavior of the receiving mail servers.

4.2 E-mail Classification

After the collection phase, (1) the dataset is pruned of all unusable e-mail traces, (2) the remaining e-mail transactions are classified into either being *accepted* or *rejected*, and finally (3) the e-mails in the *accepted* category are refined into either being *spam* or *ham*. These three steps are described in detail below.

Before any classification, we begin by discarding all unusable traces. For example, flows with no payload are mainly scanning attempts and should not be considered in the classification. Also, SMTP flows missing the proper commands are excluded from the dataset as they most likely belong to other applications using port 25. Encrypted e-mail communications cannot be analyzed, and were also eliminated.[3] Any e-mail with an empty sender address is a notification message, such as a non-delivery message [14]; it does not represent a real e-mail transmission and is also excluded. Finally, any e-mail transaction that is missing either the proper starting/ending or any intermediate packet is considered as incomplete and one might decide to leave out these e-mails when analyzing the dataset. Possible reasons for having incomplete flows include transmission errors and measurement hardware limitations caused by the framing synchronization problem (Section 2.2.3).

The remaining e-mail transactions are then classified as *accepted*, i.e. those e-mails that are delivered by the mail servers, or *rejected*. An e-mail transaction can fail at any time before the transmission of the e-mail data (header and content) due to rejection by the receiving mail server. Therefore, *rejected* e-mails are those that do not finish the SMTP command exchange phase and consequently do not send any e-mail data. The rejections are mostly because of spam prefiltering strategies deployed by mail servers including blacklisting, greylisting, DNS lookups, and user database checks.

Examining SMTP replies sent by the receiving mail servers has no effect on the classification of accepted e-mails. However, they could have been consulted for finding the reasons of e-mail rejections. In our dataset, due to asymmetric routing (see Section 2.2.3) only approximately 10% of the flows are symmetric, where both the e-mail and the corresponding mail server reply are available in the collected traces. Therefore, we have decided to not further classify the rejected e-mail transactions. However, existing responses can always be queried if required in the analysis.

Finally, we discriminate between *spam* and *ham* in our dataset. As we have captured the complete SMTP flows, including IP addresses, SMTP commands, and e-mail contents, we can establish a ground truth for further analysis of *only* the spam traffic properties and a comparison with the corresponding legitimate e-mail traffic. We deploy the widely-used spam detection tool called SpamAssassin[4] to mark e-mails as spam and ham. SpamAssassin uses a variety of techniques for its classification, such as header and content analysis, Bayesian filtering, DNS blocklists, and collaborative filtering databases.[5]

We would like to stress that these classification steps are carried out automatically after the data collection. As we describe in the next section, the contents of the e-mails are then discarded and all other user data desensitized before we can manually analyze the dataset.

4.3 Anonymization

The final pre-processing step of the Antispam dataset is to desensitize any user data. As mentioned in Section 2.2.3, any real data collection is in general privacy sensitive and large scale e-mail collection even more so. For that reason, we complete the pre-processing with a complete anonymization step. For the anti-spam project where we study traffic characteristics of ham versus spam traffic, we actually have little use of the full contents of the e-mails after they have been properly labeled. On the contrary, given that we reduce the size of the dataset significantly by throwing away user data, it actually gets easier for us to process and store.

[2] `http://www.circlemud.org/~jelson/software/tcpflow/`

[3] Around 3.8% of the flows carried encrypted SMTP sessions.

[4] `http://spamassassin.apache.org`

[5] The well-trained SpamAssassin applied to our dataset was in use for a long time at our university, incurring an approximate false positive rate of less than 0.1%, and an detection rate of 91.3% after around 94% of the spam being rejected by blacklists.

Immediately after the classification, we started by discarding the body of the e-mails as well as the subject of the e-mail and the names of the sender and receiver(s). The IP addresses in the packet headers and payload are anonymized in a prefix-preserving fashion using CryptoPAN [21], similarly to all of our other projects.

Finally, we are left with the sensitive data carried in the SMTP requests and replies, namely e-mail addresses and host/domain names. These form a structure of the underlying communication pattern and cannot simply be discarded but should instead be anonymized. We have introduced the following approach for performing domain-preserving anonymization:

- First, each e-mail address is divided into the user name and the domain name (i.e. user@domain).
- The user name is local to each domain and is simply hashed using a secure hash function.[6]
- The domain name, consisting of one or more dot-separated components, is split into its parts, and a secure hash function is applied separately to each component.
- The outputs of the hash function is then re-encoded into printable ASCII characters.
- Finally, the hashed items are appended to each other to form an *anonymized* e-mail address or domain name. This anonymized name then replaces the original one in the dataset.

Hashing each domain name component individually allows us to generate domain preserving anonymized addresses and names. This gives us the possibility to study the behavior of e-mail traffic originating from the same domain and to compare them with traffic from other domains.

Once the sensitive data was discarded, the resulting anonymized dataset had a size of 37 GB.

4.4 Summary

The anti-spam dataset was collected in a similar fashion to the other datasets (Section 2.1). However, as the collection also included packet payloads, this dataset required a more complete pre-processing step before any manual analysis could be performed. Automatic extraction of e-mail transactions from SMTP sessions, classification of the e-mails, extracting followed by discarding the e-mail bodies, finding and replacing all IP, e-mail addresses, and host/domain names inside the headers with a corresponding anonymized version, etc. are just a number of challenges associated with the collection of this type of traffic that we had to overcome.

5. ANTISPAM DATASET ANALYSIS

In the previous sections we described the necessary automatic pre-processing of the Antispam dataset before the analysis could start. In this section we change focus and present our analysis methodology of the dataset.

As we stated before, the goals of the Antispam project is to study the statistical characteristics of e-mail traffic and finding the distinguishing properties of spam and legitimate e-mails. Understanding these properties is necessary for the development of new spam detection mechanisms to detect spam already on the network level as close to its source as possible. In this section, we present some overall statistical properties of the collected e-mail traffic, and briefly describe an approach to spam mitigation we have developed.

[6]The secure hash function is a one-way function, which takes a secret cryptographic key as input.

Table 3: Antispam dataset statistics

	Incoming ($/10^6$)	Outgoing ($/10^6$)
Packets	626.9	170.1
Flows	34.9	11.9
Distinct srcIPs	2.30	0.01
Distinct dstIPs	0.57	1.94
SMTP Replies	2.84	9.14
E-mails	23.5	0.90
Ham	1.15	0.19
Spam	1.43	0.16
Rejected	17.3	0.35
Unusable	3.64	0.20

5.1 Overall E-mail Traffic Characteristics

After the exclusion of unusable flows described in Section 4.2, we ended up with 24.4 million e-mails and approximately 12 million SMTP replies. The e-mails contained $10,544,647$ distinct e-mail addresses in the SMTP headers from $532,825$ distinct domains. The unusable e-mails were then discarded.

After e-mail classification, more than 17.6 million e-mails in our dataset were classified as *rejected* and only around 2.6 million incoming and 350 thousand outgoing e-mails were classified as *accepted*. This observation is similar to what was observed in [20] where the logs of a university mail server was analyzed. In this study more than 78% of the SMTP sessions were rejected by pre-acceptance strategies deployed by the mail server to filter out spamming attempts. Table 3 shows the dataset statistics for our e-mail data captured in each direction.

5.2 E-mail Analysis for Spam Mitigation

One approach to spam detection is to conduct a social network based analysis of e-mail communication. This approach was first proposed in [3] and has since then gained a large interest. In such analysis, an e-mail network based on e-mail communication is generated and then graph-theoretical analysis is applied. By using e-mail addresses as nodes and letting edges symbolize any e-mail exchange, an e-mail network captures the social interactions between e-mail senders and receivers. Even though our dataset has been anonymized, we can still generate an equivalent e-mail network to the originally collected traffic due to the properties of the anonymization process. In [17] we study the structural properties of such a network generated from one week of traffic.

Any type of analysis of large datasets is challenging from both a memory and computational time requirement perspective, but we also faced some additional challenges in our graph-theoretical analysis. Many of the standard graph-theoretical functions used for analysis of graph structures are very computationally expensive. For instance, the calculation of the average shortest path length between all the nodes in the network (a measure of the graph connectivity) is computationally prohibitive for larger graphs. One method to reduce the complexity is to use *sampling*, but the interpretation of the results must then be done with caution [2].

The generated e-mail network from two weeks contains 10, 544, 647 nodes and 21, 537, 314 edges. To the best of our knowledge this is the largest e-mail dataset that has been used for studying the characteristics of e-mail networks. We used the `networkx`[7] package in python to create and analyze the structure of the constructed e-mail network. This package tries to load the whole graph into main memory to increase the performance. However, loading the complete graph based on two weeks of e-mail traffic was not possible, despite the fact that our processing machine has 16 GB of memory. In order to reduce the required memory we used methods such as mapping e-mail addresses to integer labels.

We also built more specific e-mail networks based on a subset of the data according to the e-mail classification into the described categories of *rejected*, *accepted/ham*, and *accepted/spam*. For example, a *spam e-mail network* is an e-mail network containing only the e-mail addresses sending and receiving spam as nodes with the edges representing any spam communication. By comparing the generated e-mail networks, many structural differences are revealed between networks built from legitimate e-mails and unsolicited traffic. A remarkable observation from our study [17] is that the structure of a ham network exhibits similar properties to that of online social networks, Internet topology, or the World Wide Web. A spam network, on the contrary, has a different structure as well as a rejected traffic network. This does in turn, given the large number of spam e-mails, affect the structural properties of the *complete* e-mail network. Our observations suggest that these distinguishing properties could potentially be exploited for detection of spamming nodes on the network level.

Our research so far has thus led to two important findings. First, we have observed differences in the characteristics of spam and ham traffic, which could lead to spam detection methods complementing current antispam tools. The acquired knowledge from our analysis of the data also provides us with the means to produce realistic models of e-mail traffic. These models could in turn be used to generate synthetic datasets as an alternative to the costly collection and challenging distribution of the large-scale original data.

6. RELATED WORK

In this section existing sources of data collection that can be deployed for performing security-related research are introduced and compared with our collection methodology.

To study malicious traffic, methods such as *distributed sensors*, *honeypot networks*, *network telescopes/darknets*, as well as *passive measurements* can be deployed for data collection. Network telescopes monitor large, unused IP address spaces (darkspaces) on the Internet[16], and are typically only traffic sinks which attract unsolicited traffic without responding to them. Distributed sensors are usually placed at diverse geographical and logical network locations by some companies including antivirus companies, allowing them to summarize wide-area trends by correlating sensor data. However, they introduce a serious bias, as the users obviously care about security. Networks of honeypots collect a large aggregation of traffic behavior from dedicated, unprotected but well monitored hosts, but passive honeypots are not very suitable for analysis of normal user responses.

Our approach, passive measurements on large-scale links, is generally viewed as the best way to study Internet traffic, as it includes real behavioral responses from a diverse user population.

Research attempts to characterize and analyze spam have used a wide range of different datasets, such as data extracted from *users' mailboxes*, *mail server log files*, *sinkholes*, and *network flows*.

Collecting sent and received e-mail headers in one user's mailbox is used in [3], but this collection methodology does not scale and any such dataset is limited to an individual user. Mail server SMTP log files, on the other hand, contain information about more users but are usually limited to incoming e-mails to a single domain. Such datasets have been used, for example by Gomes et al. [6] where eight days of SMTP log files of incoming e-mails to a university mail server was used after a pre-filtering phase and categorization by SpamAssassin.

Spam collected at sinkholes (honeypots) are usually not restricted to a single domain, as these can either just receive spam passively, or imitate an open relay which spammers can exploit to relay spam. However, as described above, sinkholes, does not include the normal user's behavior and do not provide the possibility of comparing characteristics of spam and ham. Ramachandran and Feamster [19] collected spam e-mails from two sinkholes and complemented their traces with other sources of data such as external log files of legitimate e-mails, BGP routing information, IP blacklists, etc. Pathak et al. [18] collected spam during three months from an open relay sinkhole together with information about the sending host such as TCP fingerprints, IP blacklists, etc.

Collection of flow-level data at gateway routers can lead to very large datasets; however, no ground truth and limited possibility of validating the findings are its main shortcomings. Schatzmann et al. [20] have studied NetFlow data captured during 3 months at the border router of a national ISP, and complemented their dataset with the log of a university mail server to discriminate between rejected spam and ham flows. Ehrlich et al. [5] have collected large network flow datasets from a router connecting their network to other ISPs and used local IP blacklists and whitelists to distinguish spam from ham.

Our Antispam dataset, which was passively collected on an Internet backbone link, is not limited to a single user or domain. Not only does it give us the possibility of studying the flow-level characteristics of e-mail traffic, but it also shows which flows carry spam or ham traffic, a property which is difficult to accurately determine without consulting the e-mail content.

[7]`http://networkx.lanl.gov/`

7. CONCLUSIONS

We have described a number of large-scale datasets collected on a high-speed backbone link. The datasets have been far from trivial to collect, and for that reason we shared the challenges we faced as well as our solutions for processing the large-scale data.

To exemplify the analysis process, we used the *Antispam* dataset to concretely discuss the collection and analysis of a large-scale dataset. This included our methodology for anonymization, i.e. the removal of any user-sensitive data in such a way that also allowed accurate traffic analysis, as well as a discussion of applying graph-theoretical thechniques to the generated e-mail network. To the best of our knowledge, this e-mail network is the largest that has been used to study the characteristics of such networks. We could find clear differences in the communication patterns of spam and ham traffic, something that we suggest can be used to both discriminate between them on the network level and to create more complete simulation models.

The described type of data collection is necessary for such analysis since most other contemporary data collection approaches either lack participants' e-mail addresses or do not have any legitimate traffic.

We believe that the collection of large-scale datasets such as the datasets presented in this paper is crucial for understanding the behavior of the Internet and its applications. Security research in particular needs contemporary Internet traffic in order to show the usefulness and correctness of security mechanisms and algorithms.

8. ACKNOWLEDGMENTS

This work was supported by .SE – The Internet Infrastructure Foundation, SUNET, and the Swedish Civil Contingencies Agency (MSB). The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 257007.

9. REFERENCES

[1] M. Almgren and W. John. Tracking malicious hosts on a 10gbps backbone link. In *15th Nordic Conference in Secure IT Systems (NordSec 2010)*, 2010.

[2] P. R. V. Boas, F. A. Rodrigues, G. Travieso, and L. da F Costa. Sensitivity of complex networks measurements. *Statistical Mechanics*, 2010.

[3] P. O. Boykin and V. P. Roychowdhury. Leveraging social networks to fight spam. *Computer*, 38(4), 2005.

[4] S. Donnelly. Endace dag timestamping whitepaper, `endace,http://www.endace.com/`, 2007.

[5] W. K. Ehrlich, A. Karasaridis, D. Liu, and D. Hoeflin. Detection of spam hosts and spam bots using network flow traffic modeling. In *Proceedings of the 3rd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more*, LEET'10, pages 7–7, 2010.

[6] L. H. Gomes, C. Cazita, J. M. Almeida, V. Almeida, and W. Meira, Jr. Workload models of spam and legitimate e-mails. *Perform. Eval.*, 64(7-8), 2007.

[7] W. John. *Characterization and Classification of Internet Backbone Traffic*. PhD dissertation, Chalmers University of Technology, 2010.

[8] W. John, M. Dusi, and k. claffy. Estimating routing symmetry on single links by passive flow measurements. In *Proc. of the Wireless Communications and Mobile Computing Conference*, 2010.

[9] W. John and T. Olovsson. Detection of malicious traffic on backbone links via packet header analysis. *Campus-Wide Information Systems*, 25(5), 2008.

[10] W. John and S. Tafvelin. Analysis of internet backbone traffic and header anomalies observed. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, IMC '07, pages 111–116, 2007.

[11] W. John and S. Tafvelin. Differences between in- and outbound Internet Backbone Traffic. In *TERENA Networking Conference (TNC)*, 2007.

[12] W. John, S. Tafvelin, and T. Olovsson. Trends and Differences in Connection-behavior within Classes of Internet Backbone Traffic. In *Passive/Active Measurement (PAM)*, 2008.

[13] W. John, S. Tafvelin, and T. Olovsson. Passive Internet measurement: Overview and guidelines based on experiences. *Computer Communications*, 33(5), 2010.

[14] J. Klensin. Simple Mail Transfer Protocol. RFC 5321 `http://www.ietf.org/rfc/rfc5321.txt`, 2008.

[15] D. Moore, K. Keys, R. Koga, E. Lagache, and k. claffy. The CoralReef Software Suite as a Tool for System and Network Administrators. In *USENIX LISA*, 2001.

[16] D. Moore, C. Shannon, G. Voelker, and S. Savage. Network Telescopes. Tech.rep., CAIDA, 2004.

[17] F. Moradi, T. Olovsson, and P. Tsigas. Analyzing the social structure and dynamics of e-mail and spam in massive backbone internet traffic. Technical report, Chalmers University of Technology, no. 2010-03, 2010.

[18] A. Pathak, Y. C. Hu, and Z. M. Mao. Peeking into spammer behavior from a unique vantage point. In *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, pages 3:1–3:9, 2008.

[19] A. Ramachandran and N. Feamster. Understanding the network-level behavior of spammers. *SIGCOMM*, 36(4), 2006.

[20] D. Schatzmann, M. Burkhart, and T. Spyropoulos. Inferring spammers in the network core. In *Passive and Active Measurement Conference*, 2009.

[21] J. Xu, J. Fan, M. Ammar, and S. B. Moon. On the design and performance of prefix-preserving ip traffic trace anonymization. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, IMW '01, pages 263–266, New York, NY, USA, 2001. ACM.

[22] M. Zhang, M. Dusi, W. John, and C. Chen. Analysis of udp traffic usage on internet backbone links. In *Proceedings of the 2009 Ninth Annual International Symposium on Applications and the Internet*, pages 280–281, 2009.

An Experimental Study on the Measurement of Data Sensitivity

Youngja Park, Stephen C. Gates, Wilfried Teiken, Pau-Chen Cheng
IBM T. J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598, USA
{young_park, scgates, wteiken, pau}@us.ibm.com

ABSTRACT

Data-centric security proposes to leverage the business value of data to determine the level of overall IT security. It has gained much enthusiasm from the security community, but has not materialized into a practical security system. In this paper, we introduce our recent work towards fine-grained data centric security, which estimates the sensitivity of enterprise data semi-automatically. Specifically, the categories of sensitive data and their relative sensitivities are initially determined by subject matter experts (SMEs). We then apply a suite of text analytics and classification tools to automatically discover sensitive information in enterprise data, such as personally identifiable information (PII) and confidential documents, and estimates the sensitivity of individual data.

To validate the idea, we developed a proof-of-concept system that crawls all the files in a personal computer and estimates the sensitivity of individual files and the overall sensitivity level of the computer. We conducted a pilot test at a large IT company with its employees' laptops. The pilot scanned 28 different laptops, in which 2.2 million files stored in various file formats were analyzed. Specifically, the files were analyzed to determine if they contain any of the pre-defined sensitive information, comprising 11 different PII types and 11 sensitive topics. In addition to the sensitivity estimation, we also conducted a risk survey to estimate the risk level of the laptops.

We found that, surprisingly, 7% of the analyzed files belong to one of the eleven sensitive data categories defined by the SMEs of the company, and 37% of the files contain at least one piece of sensitive information such as address or person name. The analysis also discovered that the laptops have similar overall sensitivity levels, but a few machines have exceptionally high sensitivity. Interestingly, those few highly sensitive laptops were also most at risk of data loss and of malware infection, according to user survey responses. Furthermore, the tool produces the evidence of the discovered sensitive information including the surrounding context in the document, and thus users can easily redact the sensitive information or move it to a more secure location. Thus, this system can be used as a privacy enhancing tool as well as a security tool.

1. INTRODUCTION

Identity theft and large-scale data leakage incidents are rapidly growing in recent years [1]. Loss or exposure of data, especially highly sensitive data, can cause a great deal of damage, both tangible and intangible, to the enterprise. Data-centric security has been proposed to provide fine-grained security to important and sensitive data [2]. The concept has gained much enthusiasm among security researchers and practitioners alike, but it has not fully materialized into a practical security system, mostly because enterprises do not have a clear idea of where their sensitive data resides [3]. Recently, many data loss protection (DLP) solutions have been proposed to prevent sensitive information from being leaked externally [4, 5, 6]. The state-of-the-art technologies for DLP aim to discover sensitive information in data (e.g., for regulatory compliances such as HIPAA [1] and PCI-DSS [2]), but do not have any automated mechanisms to measure the value or sensitivity of individual data items. For instance, these systems treat a file with a credit card number and a file with 100 credit card numbers equally.

We argue that security protection should be commensurate with the value or the sensitivity level of the data, and propose a new systematic, end-to-end approach for helping large enterprises manage the security risks associated with their data. Specifically, our system aims to automatically estimate the sensitivity of enterprise data based on their content and the risk level of the data based on the usage patterns.

To validate the idea, we developed a proof-of-concept (PoC) of the system, which primarily focuses on the sensitivity estimation of unstructured text, and conducted a pilot test at a large IT company with its employees' laptops. The system scans the files in a personal computer and estimates the sensitivity of individual files and the overall sensitivity level of the computer. For risk assessment, the system estimates the risk level of laptop theft and infection by malware through a user survey. The goal is to find individual laptops which have high levels of sensitive data on them, and which are at

[1]Health Insurance Portability and Accountability Act
[2]Payment Card Industry Data Security Standard

higher-than-usual risk of both theft and malware infection.

Specifically, we try to answer the following questions:

- Are the laptops of an organization similarly sensitive or are some laptops more sensitive than other laptops?
- Are the owners of more sensitive laptops aware of the sensitive information in their machines?
- Is there any correlation between the sensitivity level and the risk level of a laptop?

39 employees volunteered for the pilot test, and, in total, about 2.3 million files in various file formats were analyzed. All analyses were done locally inside a machine, and the pilot collected only statistical information from the individual machines such as the number of confidential documents or number of files containing credit card numbers. Thus, no file was needed to be transferred or stored in a central storage preserving the privacy. In this pilot, the system estimated data sensitivity based on the occurrences of 11 different PII types and 11 sensitive topics in the data, which were identified as sensitive to the company by the SMEs. Our analysis results show that about 7% of the 2.3 million files belong to one of the 11 sensitive data categories, and 37% of the scanned files contain at least one piece of sensitive information. Among sensitive data categories, company confidential documents and PII documents are most common categories. The results also reveal that most laptops have a similar overall sensitivity level, but a few machines have exceptionally high sensitivity. By associating the sensitivity and risk levels of the laptops, we discovered that those few highly sensitive laptops were also most at risk of data loss and of malware infection.

In the remainder of this paper, we present more detailed characteristics of the pilot data and interesting findings obtained by analyzing a large number of documents in business laptops. We also discuss several pragmatic issues to deal with in conducting a deep content inspection of primary workstations, including how to scan and analyze files efficiently while the users are still using the laptops and how their privacy is protected.

2. SEMI-AUTOMATED SENSITIVITY ANALYSIS

In this section, we describe a high-level overview of the sensitivity estimation method. The system is designed to scan files in a computer and to estimate the sensitivity of the computer based on the contents of the files. In addition, we try to estimate the risk level of the machine using various factors including the owner's organizational status and the the usage patterns. The high-level process for estimating both the laptop sensitivity and risk levels are depicted in Figure 1.

In this section, we describe more details on how to define sensitive data categories and their sensitivity scores, and the automated process for estimating the sensitivity of a computer. The risk assessment step is discussed in Section 4.

2.1 Defining Sensitive Data Categories and The Sensitivities

The first step towards measuring data sensitivity is to define what kinds of information are important or sensitive

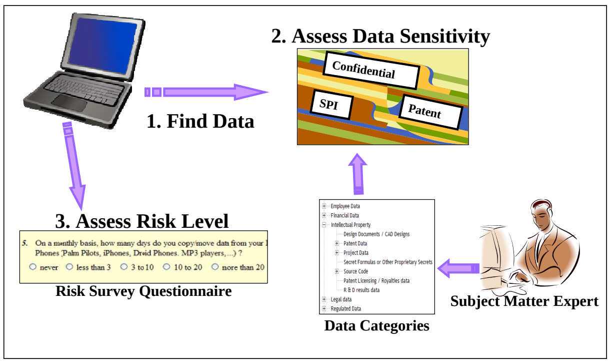
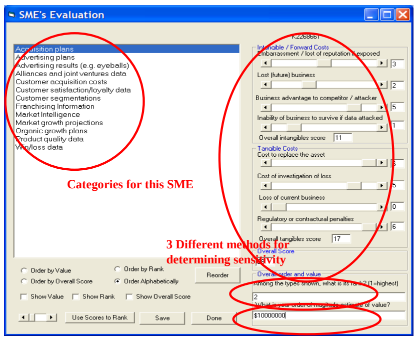

2. Assess Data Sensitivity
Confidential
SPI
Patent
1. Find Data
3. Assess Risk Level
5. On a monthly basis, how many days do you copy/move data from your Phones (Palm Pilots, iPhones, Droid Phones. MP3 players,...) ?
never
less than 3
3 to 10
10 to 20
more than 20
Risk Survey Questionnaire
Employee Data
Financial Data
Intellectual Property
Design Documents / CAD Designs
Patent Data
Project Data
Secret Formulas or Other Proprietary Secrets
Source Code
Patent Licensing / Royalties data
R & D results data
Legal data
Regulated Data
Data Categories
Subject Matter Expert

Figure 1: Process for Estimating the Sensitivity and Risk Levels

and what their respective values are to the company. Categories of sensitive data can vary across different countries and even across organizations within a country [7]. Therefore, the categories can best be defined by the people inside the organization.

We gathered 104 sensitive data types for the IT company by asking 30 subject matter experts (SMEs), most of whom are executive level employees from various parts of the company. The data types range from employee data and financial data to future business plans and intellectual property data. Next, we asked the SMEs to rank-order the data categories by their importance to the company considering both tangible and intangible factors. We showed each of the SMEs the full list of the data categories and asked to rank all or a subset of the categories they are knowledgable about. Ties were allowed in ranking. We developed a GUI tool to help the SMEs with the ranking process. Figure 2 shows a snapshot of the tool. The tool allows users to select a subset of data categories they want to rank, and provides multiple ways to enter sensitivity scores.


SME's Evaluation

K2268661

Acquisition plans
Advertising plans
Advertising results (e.g. eyeballs)
Alliances and joint ventures data
Customer acquisition costs
Customer satisfaction/loyalty data
Customer segmentations
Franchising Information
Market Intelligence
Market growth projections
Organic growth plans
Product quality data
Win/loss data

Categories for this SME

Intangible / Forward Costs
Embarrassment / lost of reputation if exposed 3
Lost (future) business 2
Business advantage to competitor / attacker 5
Inability of business to survive if data attacked 1
Overall intangibles score 11

Tangible Costs
Cost to replace the asset 6
Cost of investigation of loss 5
Loss of current business 0
Regulatory or contractual penalties 6
Overall tangibles score 17

3 Different methods for determining sensitivity

Overall Score

Overall order and value
Among the types shown, what is its rank? (1=highest) 2
What is your order of magnitude estimate of value? $10000000

Order by Value ○ Order by Rank ○ Reorder
Order by Overall Score ○ Order Alphabetically ●
Show Value ☐ Show Rank ☐ Show Overall Score ☐
Use Scores to Rank | Save | Done

Figure 2: SME Interview Tool

We, then, combined the partial rankings from all of the SMEs into a single ranked list, and data categories with very similar rankings were grouped into subgroups resulting in 8 distinct bands, from Band A to Band H. Finally, we asked the SMEs to provide relative sensitivity for each of the bands, where the sensitivity of the least sensitive band is 1. In this round, 6 SMEs provided relative sensitivity scores for the bands, and we use the average score for each band as its sensitivity score. The final sensitivity scores of

Band	Sample Data Categories	Sensitivity
Band A	Jewel Code	52
Band B	Acquisition Plans	45
Band C	Trade secrets	37
Band D	Sensitive Personal Information, Personal Health Data	29
Band E	Design Documents, Employee Personal Data	19
Band F	Employee Incentives Data, Project Plans	11
Band G	Non-Jewel Code, Network Log Data	5
Band H	Market Growth Data, Delivery Plans	1

Table 1: Data categories and their relative sensitivities

the 8 bands and some sample data categories in each band are shown in Table 1.

2.2 Sensitivity Estimation Process

The sensitivity estimation process consists of two components: file scanning and data classification.

Find Data: The component scans the hard disk of a computer and selects files for analysis. The system currently supports Windows, Linux and Mac OS X platforms. We allow the users easily customize the file retrieval process. Users can limit the scanning to a certain directories or certain type of files (e.g., analyze only "/projects" directory or "only PPT files"), or can exclude certain directories or files from the scanning (e.g., exclude "/MyPrivateDirectory"). Upon retrieval of a file, it converts a non-textual file (e.g., MS PowerPoint files and Adobe PDF files) into plain text for content inspection.

Assess Data Sensitivity: This component comprises a suite of text analytics and classification engines that categorize unstructured text into a set of sensitive data types. Currently, the text analytics engine supports the 11 sensitive data categories listed in Table 2. The data categories were chosen by the CIO's office at the company as the first target set for the study. As we can note from the list, the company is most concerned about satisfying regulatory compliance and protecting intellectual property.

The SPI category is for the company's HR (human resource) documents with personal information. The PHI category includes documents containing medical information (i.e., disease name or medical treatment name) together with personally identifiable information. The PCI category is assigned to documents containing a credit card number, an expiration date and a person name together. It is worth noting that the data categories are not independent each other. For instance, the SPI and PHI categories supersede the PII category. If a PII document contains a medical information or is a HR document, its category is escalated to PHI and SPI respectively. Similarly, most design documents and proprietary source code files are also company confidential documents.

Note that we need to recognize personal information to identify the SPI, PHI, PCI and PII categories. Currently, the system can identify the following sensitive data types.

- Social Security Number
- Passport Number
- Lotus Notes Email Address
- Internet Email Address
- Employee Name
- Non-employee Person Name
- Address
- Phone Number
- Date of Birth
- Credit Card Number
- Medical Information including disease names, treatment names and drug names.

Data Categories	Bands
Sensitive Personal Information (SPI)	Band D
Personal Health Information (PHI)	Band D
Payment Card Industry Information (PCI)	Band E
Personally Identifiable Information (PII)	Band E
Design Document	Band E
Patent Disclosure	Band E
Patent Application	Band E
Employee Record	Band E
Salary Information	Band F
Proprietary Source Code	Band G
Other Confidential Documents	Band G

Table 2: Sensitive data categories the pilot system discovers.

The system applies both rule-based approaches including signatures, regular expressions and linguistic patterns, and machine learning-based classification methods. Often, different approaches are applied together to enhance the efficiency and accuracy[8, 9]. For instance, a signature-based method is used to identify candidate medical information (i.e., known disease names), and a statistical classification method is applied to determine if the term really has a medical sense in the given context. Similarly, a set of linguistic patterns are used to identify candidate confidential documents, e.g., "do not (disclose|distribute|forward|share)", then a supervised machine learning method is used to decide if the identified confidential label indeed denotes the document is confidential or is used in a different context. The accuracy levels of discovering PII data types, confidential documents and proprietary source code files reach to 97–98%. Due to lack of ground truth data, we were not able to objectively measure the accuracy levels of other classifiers.

When the classification process is completed, the component then maps the classification results into their sensitivity scores as defined in the data taxonomy (Table 1). Note that the sensitivity scores are based on the value of a single data (e.g., one credit card number) or a single document (e.g., one source code file), so we count the number of occurrences of each type of sensitive information to estimate the overall sensitivity of a document and a laptop. For instance, if a laptop contains 10 Band A files, then the overall sensitivity of the laptop is 520.

3. PILOT STUDY ON DATA SENSITIVITY ESTIMATION

During the study, 39 employees volunteered for the pilot test, but 29 of the participants completed the laptop scanning and sensitivity assessment of the files in their laptops. In this section, we report the data used in the pilot and discuss the analysis results obtained from the 28 laptops.

3.1 Pilot Data Collection

The volunteers downloaded and ran the pilot system in their laptops. Both the file scanning and content inspection were done entirely locally in the participants' machines, and no file was transferred to a central storage to preserve the privacy. The pilot system encountered, in total, 2,577,066 files in the 28 laptops, but, 2,260,418 files (87.7% of the retrieved files) were successfully analyzed by the system. The main reason of the analysis failure for some files is that the pilot system did not support some file formats (e.g., tfm and jpeg) and thus failed to extract the content. Nonetheless, the analysis of over 2 million unstructured text can provide meaningful insights on the types of information in the files and their sensitivity levels.

To minimize the interruption to the volunteers, we limited the analysis time up to 8 hours and the total number of files to 200,000 per laptop. Modern laptops can easily store more than 200,000 files. Therefore, the system did not scan all the files in the laptops. However, the sensitive data categories in Table 2 appear mostly in certain file formats such as PDF, PPT, DOC, XLS, source code (e.g., JAVA and C++) and HTML. Thus, to maximize the probability of finding all relevant sensitive information on the laptop, the system first scanned these file types. If the number of these files in a laptop was smaller than 200,000, then the system analyzed other files. In this way, we could find most sensitive files within a reasonable time frame.

The pilot system can process various commonly used file formats including *Microsoft Office* document files, html/xml files and PDF files. Table 3 lists the 10 most common file formats and the distribution of files across the file formats in the file collection. As we can see from the table, plain text and html constitute over 50% of all the scanned files in business-use laptops. Plain text format contains many different types of files including log files and data files as well as text documents.

FileFormat	File Count	Ratio
txt	756,614	33.6%
htm	639,275	28.4%
code	545,543	24.2%
xml	250,862	11.1%
pdf	36,480	1.6%
doc	13,636	0.6%
ppt	10,163	0.5%
xls	4,325	0.2%
zip	3,501	0.2%

Table 3: 10 most common file formats and the number of files by the formats

The first column in Table 4 lists the number of analyzed files from each laptop. We can see that almost 200,000 files (i.e., the maximum number of files per machine for this pilot) were analyzed for many of the laptops, but several laptops contributed a small number of files because the volunteers restricted the system scanning to a small part of the laptop.

Machine ID	Number of Files	Files of sensitive topics	Files of sensitive topic or PII types
M1	162,256	12,966 (8.0%)	74,827 (46.1%)
M2	12,079	562 (4.7%)	4,772 (39.5%)
M3	10,060	241 (2.4%)	2,370 (23.6%)
M4	105,654	3,834 (3.6%)	29,758 (28.2%)
M5	8,925	84 (0.9%)	4,807 (53.9%)
M6	17,609	1,443 (8.2%)	5,520 (31.3%)
M7	50,736	4,887 (9.6%)	20,272 (40.0%)
M8	109,776	1,090 (1.0%)	32,592 (29.7%)
M9	7,980	1,160 (14.5%)	2,988 (37.4%)
M10	144,451	7,442 (5.2%)	59,814 (41.4%)
M11	59,919	3,862 (6.4%)	16,572 (27.7%)
M12	150,879	86,198 (57.1%)	98,172 (65.1%)
M13	132,790	0 (0.0%)	46,594 (35.1%)
M14	152,137	23,510 (15.5%)	43,411 (28.5%)
M15	140,862	6,141 (4.4%)	43,251 (30.7%)
M16	155,256	0 (0.0%)	128,838 (83.0%)
M17	177,495	2,030 (1.1%)	45,172 (25.4%)
M18	630	78 (12.4%)	382 (60.6%)
M19	4,270	31 (0.7%)	1,515 (35.5%)
M20	16,448	1,367 (8.3%)	6,663 (40.5%)
M21	39,047	958 (2.5%)	3,844 (9.8%)
M22	43,950	1,460 (3.3%)	7,370 16.8%
M23	5,114	69 (1.3%)	1,028 (20.1%)
M24	148,368	0 (0.0%)	81,896 (55.2%)
M25	133,870	16,698 (12.5%)	36,516 (27.3%)
M26	103,139	1,115 (1.1%)	34,528 (33.5%)
M27	162,496	1,823 (1.1%)	80,319 (49.4%)
M28	4,222	23 (0.5%)	929 (22.0%)
Total	2,260,418	179,072 (6.7%)	914,720 (37.0%)

Table 4: Number of analyzed files from each laptop. The second column denotes the number of files which are classified as one of the 11 sensitive data categories. The third column shows the number of files in each laptop that are a sensitive topic or contains any of the PII data types.

3.2 Sensitivity Assessment

For data sensitivity assessment, the pilot system first converted the files into plain text formats, and then applied text analytics and classification tools to identify what kinds of sensitive information the files contain. Then, the sensitivity scores of individual files were computed according to the SMEs' inputs (see, Table 1 and Table 2). The sensitivity of a document is measured based on the topic of the document; if a document is classified into one of the 11 sensitive data categories defined in Table 2, then the sensitivity score of the band the category belongs to is assigned to the document. For instance, the sensitivity of a document with PHI information is 29. If a document is classified into more than one categories, the sensitivity of the document is the cumulative score of all sensitivity scores. The sensitivity score of a laptop is the cumulative score of the sensitivity scores of all documents in the laptop.

Table 5 and Table 6 show the number of laptops where each sensitive category and each PII type appear respectively. Not surprisingly, the proprietary source code files and general confidential documents appear in most of the laptops given that the laptops are owned by an IT company. PII and PHI categories are also present in many laptops. Most PII types are present in all laptops, but, as expected, passport numbers and social security numbers, which are more sensitive than other PII types, appear in a small number of laptops.

Data Categories	Num.Machines
SPI	3
PII	25
PCI	0
PHI	24
Proprietary Code	24
Patent Disclosure	3
Patent Application	6
Salary Information	1
Employee Record	2
Design Document	5
Confidential Document	25

Table 5: Sensitive data categories and the number of laptops each category appears in out of 28 machines

PII Types	Num.Machines
Phone Number	28
SSN	17
Passport Number	5
Credit Card Number	20
Internet Email Address	27
LotusNotes Address	27
Employee Name	28
Person Name	27
Address	28
Date of Birth	25
Medical Information	28

Table 6: PII entities and the number of laptops each PII type appears in out of 28 machines

Let's look at how many documents of sensitive topics each laptop carries. The second column in Table 4 shows the number of files that were classified as one of the 11 sensitive categories in each laptop. The result shows that, on average, 6.7% of the files in the laptops belongs to the sensitive data categories. However, a few laptops, especially *M12*, contains very high rate of sensitive information which suggests that the laptop may need more security measures. The third column in Table 4 shows the number of files that are of sensitive categories or containing PII data types. The result also shows that 37% of all the files contain at least one piece of PII information.

Figure 3 depicts the ratio of each sensitive category in each machine. As we can see, "Proprietary Code", "Confidential Documents" and "PII" are most common data categories.

Once we identify the topics of all the files in a machine, we can compute the sensitivity score of the files and the overall sensitivity level of the laptop as described earlier. Figure 6 depicts the overall sensitivity of individual laptops. As we can see, there is a noticeable variation in the laptops' sensitivity. A small number of laptops have much higher sensitivity scores than other laptops. Specifically, $M12$ has an exceptionally high sensitivity than others, which may need additional security protection.

4. PRELIMINARY RISK ANALYSIS

To understand if there is any correlation between the sensitivity and the risk level of a machine, we conducted a small-scale risk assessment study using a user survey.

The survey contains 10 questions which are designed to find out the participants' laptop usage habits and their experience of laptop theft and compromise. The questionnaire is composed of 3 parts. The 1st part aims to find out the location and frequency of laptop usages: how often their laptops are used in public places such like restaurants, public transportation, airports, libraries, etc. The 2nd part asks how they use their laptops: how often they browse the web with their laptops, what kind of web sites they browse, how often removable media are used to transfer data to/from laptops. The 3rd part asks if they have experienced laptop theft, compromise or identity theft in the past year. We then compute the risk level of laptop theft or comprise as the conditional probabilities of the laptop theft or compromise given the answers to the first two parts.

All 39 volunteers participated in the risk survey. Even though the sample size is very small for any statistically significant results, the survey results provide some interesting insights. As Figure 5 shows, 67% and 56% of the laptops were measure to have 11-20% probability of laptop loss and malware infection respectively, which seem very high for business machines.

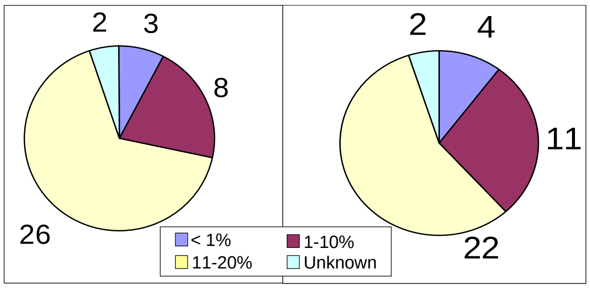


Figure 5: Probability distribution of Laptop Loss and Malware Infection

Finally, we put together the sensitivity scores and the risk scores to find out if there is any relationship between the laptop sensitivity and risk level. The charts in Figure 6 show the relationships. Interestingly, the most sensitive machines (M1, M12 and M14) also have the highest risk levels, making them good targets for additional security protection.

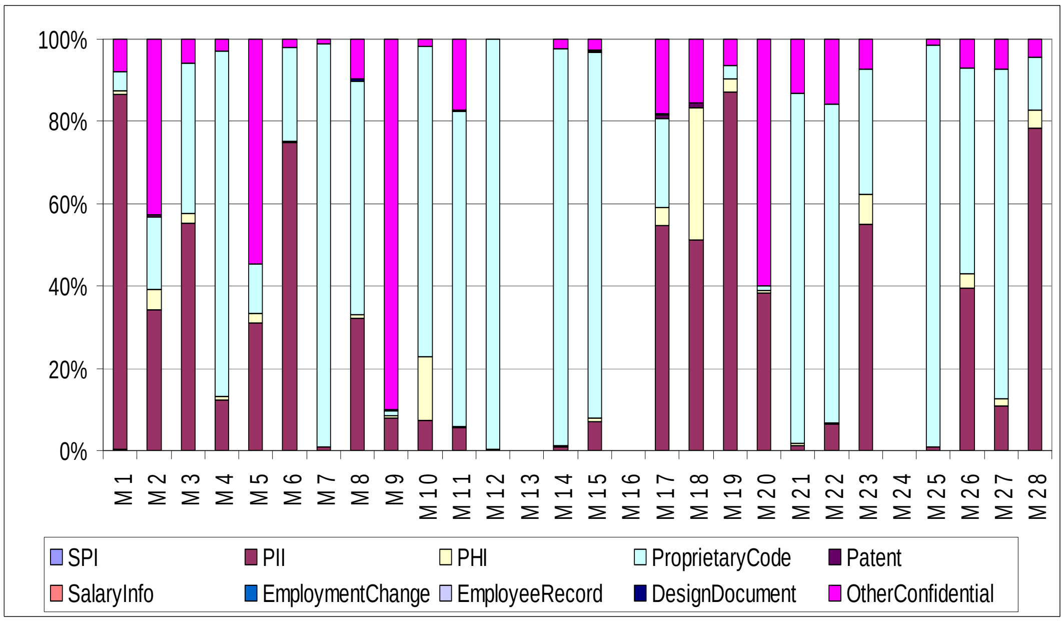


Figure 3: Ratio of each sensitive topics in individual laptops

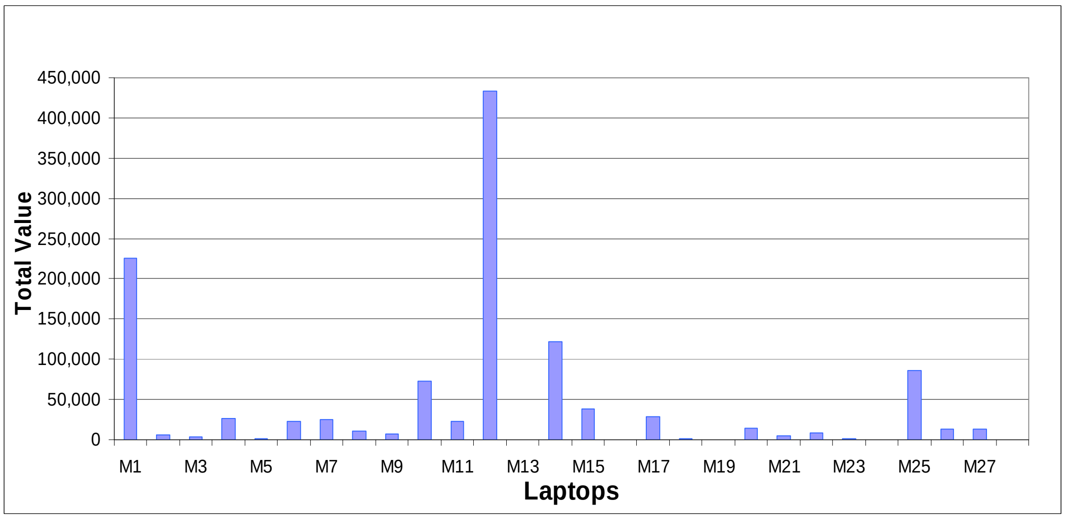
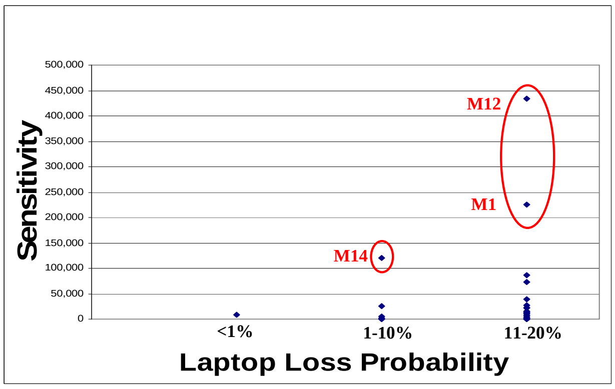
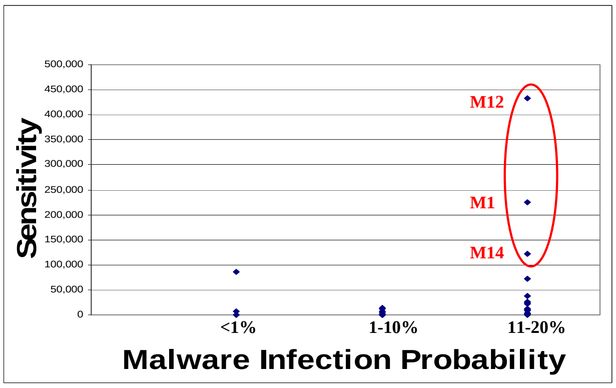


Figure 4: Overall sensitivity scores of the laptops



(a) Data Sensitivity vs. Risk of Laptop Loss



(b) Data Sensitivity vs. Risk of Malware Infection

Figure 6: Correlation between the sensitivity and the risk level of laptops

5. LESSONS LEARNED

There were several lessons learned from the operational aspect of the pilot. First and most importantly, the mistrust on software scanning the filesystem was prevalent, even though the participants had the options of limiting the scan and of reviewing the results. In this study, the participation was entirely voluntarily, but it raises the question of responses and behaviors to expect in a mandatory environment. This will be a topic of further study during the continued system development.

Another lesson learned during the pilot was that the diversity of platforms was larger than expected. We observed some pilot system failures on laptops with unexpected configurations. The pilot system failed to scan machines running Windows 98, or other machines with outdated java virtual machines. Some employees could not participate in the pilot due to these system issues. This created a conflict between the goal to accommodate every participant willing to run the pilot system and the limited resources to support the pilot.

The last and very positive lesson learned was that a lot of the participants were amazed by the results of the data analysis. Although the evaluation was not formally part of the pilot, we got feedback from participants that the pilot system discovered documents containing their personal information that they forgot about or temporary documents that were created unbeknown to them. This implies that many employees are not aware of the extent their private information is revealed which can lead to identity theft. We believe, from this finding and the finding on people's resistance on this tool, that this tool can be widely deployed if the company communicate with the employees on the benefits of the tool to the employees as well as the company.

6. CONCLUSIONS

In this paper, we described a semi-automated method for estimating sensitivities of enterprise data, and demonstrate the effectiveness of the system through a pilot test at a large company. The pilot results produced some interesting findings:

- Most laptops have a similar level of sensitivity with a few outliers.
- The few outliers have much higher sensitivity than others.
- Proprietary source code, confidential documents and PII are the most frequent sensitive data categories in the laptops.
- Though preliminary, the risk assessment results indicate that highly sensitive laptops are also at higher risk.

To the best of our knowledge, this is the first systematic attempt for estimating data sensitivity quantitatively. Even though the current system supports classification of a limited number of sensitive data types, the sensitive data category and PII data sets include very representative data types to many companies. A system that can automatically discover sensitive data types and estimate the sensitivities of employees' laptops is very useful and can bring a significant value to both companies and end users. In summary, the system can be used to

- Enable individual users to understand both security and privacy risks on their own machines and to act accordingly to minimize these risks
- Enable different levels of monitoring of various data types, depending upon their sensitivity
- Enable understanding of the current protection level for entire enterprise by data sensitivity, data type, and machine.
- Enable rational spending decisions on security by modifying data protection guidelines according to their sensitivity

We continue to build text anlaytics and classifiers to extend the supported data categories for other industry and companies. Note that, however, many data categories are industry-independent (e.g., HR documents, PHI, and PII), thus, the system can be applied to different companies and industries without much customization efforts. Furthermore we plan to build a risk model using company-wide data on laptop loss and malware infection, and to use the model to better estimate the risk level of individual laptops. Another research direction is to study the usability of the tool and to make it more acceptable by end users.

7. REFERENCES

[1] Open Security Foundation: OSF datalloss db. (*http://datalossdb.org/*)

[2] Grandison, T., Bilger, M., O'Connor, L., Graf, M., Swimmer, M., Schunter, M., Wespi, A., Zunic, N.: Elevating the discussion on security management: The data centric paradigm. In: Proceedings of the 2nd IEEE/IFIP International Workshop on Business-driven IT Management (BDIM). (2007)

[3] van Cleeff, A., Wieringa, R. In: Proceedings of the IADIS International Conference Information Systems. (2009) 105–112

[4] Mogull, R.: Dlp content discovery: Best practices for stored data discovery and protection. *http://www.emea.symantec.com/discover/downloads/DLP-Content-Discovery-Best-Practices.pdf* (2008)

[5] Liu, S., Kuhn, R.: Data loss prevention. In: IT Professional. Number 2 (2010) 10–13

[6] Parno, B., McCune, J.M., Wendlandt, D., Andersen, D.G., Perrig, A.: Clamp: Practical prevention of large-scale data leaks. IEEE Symposium on Security and Privacy (2009) 154–169

[7] McCullagh, K.: Data sensitivity: Proposals for resolving the conundrum. Journal of International Commercial Law and Technology **2** (2007)

[8] Sokolova, M., El Emam, K., Rose, S., Chowdhury, S., Neri, E., Jonker, E., Peyton, L.: Personal health information leak prevention in heterogeneous texts. In: Proceedings of the Workshop on Adaptation of Language Resources and Technology to New Domains. AdaptLRTtoND '09, Association for Computational Linguistics (2009) 58–69

[9] Park, Y.: A text mining approach to confidential document detection for data loss prevention. In: IBM Research Technical Report RC25055. (2010)

Sandnet: Network Traffic Analysis of Malicious Software

Christian Rossow[1,2], Christian J. Dietrich[1,3], Herbert Bos[2], Lorenzo Cavallaro[2], Maarten van Steen[2], Felix C. Freiling[3], Norbert Pohlmann[1]

[1] Institute for Internet Security, University of Applied Sciences Gelsenkirchen, Germany
[2] Department of Computer Science, VU University Amsterdam, The Netherlands
[3] Department of Computer Science, University of Erlangen, Germany

ABSTRACT

Dynamic analysis of malware is widely used to obtain a better understanding of unknown software. While existing systems mainly focus on host-level activities of malware and limit the analysis period to a few minutes, we concentrate on the network behavior of malware over longer periods. We provide a comprehensive overview of typical malware network behavior by discussing the results that we obtained during the analysis of more than 100,000 malware samples. The resulting network behavior was dissected in our new analysis environment called *Sandnet* that complements existing systems by focusing on network traffic analysis. Our in-depth analysis of the two protocols that are most popular among malware authors, DNS and HTTP, helps to understand and characterize the usage of these prevalent protocols.

1. INTRODUCTION

Dynamic analysis, i.e. runtime monitoring, has proven to be a well-established and effective tool to understand the workings of yet unknown software [8, 14, 17]. Understanding the behavior of malicious software may not only provide insights about actions of malicious intents, upcoming techniques, and underground economy trends, but it also gives the opportunity to develop novel countermeasures specifically built on top of that understanding. Current analysis systems have specialized in monitoring system-level activities, e.g., manipulation of Windows registry keys and accesses to the file system, but little effort has generally been devoted to understanding the network behavior exposed by malware. In fact, similarly to system-level activities, network-level activities also show very distinct behaviors that can back up the insights provided by system-level analyses. Second, the very same network behaviors can uniquely provide further specific understanding necessary to develop novel approaches to collect, classify and eventually mitigate malicious software. Driven by this observation we focus our research on dissecting, analyzing, and understanding the behavior of malicious software as observed at the network level.

BADGERS 2011 Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, Salzburg, 2011
Copyright 20XX ACM X-XXXXX-XX-X/XX/XX ...$10.00.

As we will show later, the observed malware behavior highly depends on the duration of the dynamic analysis. Current systems try to analyze as many malware samples as possible in a given period of time. This results in very short analysis periods, usually lasting only a few minutes, which makes it difficult to observe malicious network behavior that goes beyond the bootstrapping process. From a network behavior point of view, however, the post-bootstrap behavior is often more interesting than what happens in the first few minutes. A thorough analysis is key to understanding the highly dynamic workings of malware, which is frequently observed to be modular and often undergoes behavior updates in a pay-for-service model.

In this paper we present an in-depth analysis of malware network behavior that we gathered with a new system called *Sandnet* during the last 12 months. Sandnet [3] is an analysis environment for malware that complements existing systems by a highly detailed analysis of malicious network traffic. With Sandnet, we try to address two major limitations we see in publicly available dynamic analysis systems: a short analysis period and the lack of detailed network-behavior analysis. While existing systems have usually spent only a couple of minutes to run a malware sample, we ran each sample for at least one hour. In addition, using the data collected through Sandnet, we provide a comprehensive overview of network activities of current malware. We first present a general overview of network protocols used by malware, showing that DNS and notably HTTP are prevalent protocols used by the majority of malware samples. We will then provide an in-depth analysis of DNS and HTTP usage in malware. The results of our analysis [3] can be used to spawn new research such as clustering malware based on network-level features or network-level malware detection.

The main contributions of this work are:

- We have in operation a new data collection and analysis environment called Sandnet that will be up for the long run and that we will continuously use to gather information on malware network behavior.

- We give an overview of the network activities of more than 100,000 malware samples and compare the results with data from previous efforts.

- An in-depth analysis of DNS and HTTP traffic provides details on typical protocol-specific usage behaviors of malware, e.g. DNS fast-flux or click fraud.

This paper is structured as follows. In Section 2, we will give an overview of Sandnet. Section 3 describes the dataset our analysis is based on. We will then provide a general malware network traffic overview in Section 4. In Section 5, we will provide a deep analysis on the usage of the DNS protocol by malware. Section 6 describes the usage of the HTTP protocol by malware. We will discuss related work in Section 7 and show future work in Section 8.

2. SYSTEM OVERVIEW

In Sandnet, malware is analyzed in execution environments known as *sandpuppets* consisting of (virtualized) hardware and a software stack. Currently, we use VMs with Windows XP SP3 based on VirtualBox as sandpuppets. The machines are infected immediately after booting and gracefully shut down after a configurable time interval, which is typically one hour. Each sandpuppet is configured to have a local IPv4 address and a NATed Internet connection. A local DNS resolver is preconfigured.

The *sandherder* is a Linux system hosting the sandpuppet virtual machines. Besides virtualization, the sandherder also records, controls and transparently proxies network traffic to the Internet. We limit the potential damage of running malware samples by transparently redirecting certain traffic (e.g. spam, infections) to local sinkholes or honeypots. In addition, we limit the number of concurrent connections as well as the network bandwidth and packet rate per sandpuppet to mitigate DoS activities. Internet connectivity parameters such as bandwidth and packet rate must be shared fairly among all sandpuppets in order to avoid inter-execution artifacts. The current Sandnet setup comprises five bot sandherders with four sandpuppets each, resulting in twenty sandpuppets dedicated to malware analysis. Herders and sandpuppets can easily be added due to a flexible and distributed design.

After executing a malware binary, we dissect the recorded network traffic for further analysis. A *flow-extractor* converts raw .pcap-files into UDP/TCP *flows*. A flow is a network stream identified by the usual 5-tuple (layer 4 protocol, source IP addr., destination IP addr., source port, destination port). For TCP, a flow corresponds to a reassembled TCP connection. For UDP, a flow is considered to be a stream of packets terminated by an inactivity period of 5 minutes. Our experience shows that this timeout length is a reasonable mechanism to compensate the lack of UDP flow termination frames. Additionally, we use payload-based protocol detection in order to determine the application-level protocol of a flow. We define a flow to be *empty*, if no UDP/TCP payload is transmitted in this flow.

Automated execution of malicious software raises some ethical concerns. Given unrestricted network connectivity, malware could potentially harm others on the Internet. Possible attack scenarios are, but not limited to, Denial-of-Service attacks, spam or infection of other hosts. We tried to find the right balance between ethical concerns when designing Sandnet and restrict the Internet connectivity. Technically, we integrated certain honeywall techniques. The harm of DoS attacks is limited by network level rate-limiting, spam is transparently redirected to local mail servers and protocols known to be used for infection are redirected to local honeypots. Sandnet is closely monitored during execution. Admittedly, it is technically impossible to completely prevent all possible attacks. However, we are convinced that within the bounds of possibility we implemented a huge part of mitigation techniques and that the value of Sandnet strongly outweighs the reasonably limited attack potential.

3. DATASET

In order to study malicious network traffic, we analyzed malware samples that were provided to a great degree by partner research institutions. For each sample we acquire A/V scan results from VirusTotal [1]. 85% of the samples that we executed had at least one scan result indicating malware (see Figure 1). In order to avoid accidental benign samples we collated our set of samples with a list of known software applications using Shadowserver's bintest [4]. We randomly chose the samples from a broad distribution of all malware families. We tried to mitigate side-effects of polymorphism by extracting the family name of a given malware sample's A/V labels and limit the number of analyses per malware family.

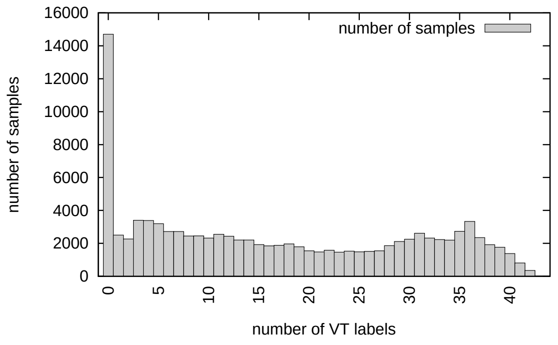


Figure 1: Histogram of VirusTotal Labels per Sample

For our analysis we defined the following set of samples. We analyzed a total of 104,345 distinct samples (in terms of MD5 hashes) over a timespan of one year. Samples were executed with regard to their age. On average, the samples were executed about 7.8 days after submission. We gradually increased the time between sample acquisition and execution from 6 hours to 150 days in order to evaluate whether the execution age significantly influences malware activity. Some statistics on the database of our data set is provided in annex F. The total analysis time of all samples in this data set sums up to an analysis period of 12 years.

4. NETWORK STATISTICS OVERVIEW

Of the 104,345 samples, the subset S_{Net} of 45,651 (43.8%) samples exhibited some kind of network activity. The network traffic caused by these samples sums up to more than 70 million flows and a volume of 207 GB. It remains an open issue to understand why a majority of the samples did not show any network activity. We suspect that most of the inactive samples a) are invalid PE files, b) operate on a local system only (e.g. disk encryption), c) are active only if there is user-activity or d) detected that they are being analyzed and stopped working.

Protocol inspection reveals that a typical sample in S_{Net} uses DNS (92.3%) and HTTP (58.6%). IRC is still quite popular: 8% of the samples exposed IRC. Interestingly, SMTP only occurred in 3.8% of the samples in S_{Net}. A complete list of the ISO/OSI layer-7 protocol distribution can be found in annex A. As DNS and HTTP are by far the most widely used protocols in Sandnet traffic, we will inspect these in more detail in Table 1. Table 1 also compares our protocol statistics with data based on Anubis provided by Bayer et al. [7] in 2009. Interestingly, when comparing the results, the samples we analyzed showed increased usage of all protocols. However, the ranking and the proportion of the protocols remain similar. We suspect this increase is a) due to a relatively long execution of malware samples and b) caused by a growing usage of different application-level protocols by malware.

Protocol	Reference	Sandnet
DNS	44.5	92.3
HTTP	37.6	58.6
IRC	2.3	8.0
SMTP	1.6	3.8

Table 1: Sandnet: Layer-7 protocol distribution compared with [7] (% of S_{Net})

30.1% of the flows were empty (no payload was transmitted). All these flows are presumable scanning attempts. Already 90% of the empty flows targeted NetBIOS/SMB services. The remaining empty flows are normally distributed over lots of different ports.

Of the remaining flows with payload (69.9%), for 22.8% no well-known protocol could be determined. Over 60% of these flows account for NetBIOS or SMB-related communication (mostly scanning) according to the destination port. Again, the remaining flows with failed protocol detection are normally distributed across many destination ports.

Payload-based protocol detection is a big advantage if protocols are used over other than their well-known ports. We found that 12.8% of S_{Net} use protocols over other than the well-known ports. We speculate that in these cases malware tries to communicate via ports opened in the firewall, independent from the actual communication protocol. For instance, we regularly found IRC bots connecting to IRC servers listening on TCP port 80. Thus, non-standard port usage might serve as a malware classification or detection feature. The top 3 affected protocols are listed in Table 2.

Protocol	S_{Net} Samples (%)	Distinct Ports
HTTP	8.17	303
IRC	7.13	174
Flash	0.91	9

Table 2: Top 3 protocols over non-standard ports

As additional analysis, we found out that a longer analysis period is indeed helpful for a better understanding of malware behavior. To judge on this, we performed three measurements each after an analysis period of 5 minutes and after 1 hour. First, we found out that only 23.6% of the communication endpoints that we have seen samples connecting to were contacted in the first 5 minutes of analysis. We then calculated that only a minor fraction (6.1%) of all flows started within the first 5 minutes. Lastly, we found that 4.8% of the samples started using a new protocol after 5 minutes that they have not used in the first minutes.

5. DNS

DNS is by far the most prevalent layer-7 protocol in Sandnet network traffic and gives an interesting insight into malware activity. The subset of samples using DNS is denoted by S_{DNS}.

5.1 DNS Resolution

Although all sandpuppets have their Windows stub resolver point to a working DNS resolver, we observed malware that used a different resolver or even carried its own iterative resolver. We developed the following heuristic in order to detect executions that carry an iterative resolver. An execution is considered as carrying an iterative resolver if there is an incoming DNS response from a server other than the preconfigured DNS resolver with a referral concerning a TLD (a resource record of type NS in the authority section) and the Recursion Available flag set to 0. We cross checked the resulting executions whether at least one of the DNS root servers had been contacted via DNS.

We can only speculate on the reasons why the preconfigured local DNS resolver is avoided. Using one's own resolver clearly has advantages. Resolution of certain domains might be blocked at the preconfigured resolvers in some environments (e.g. corporate ones). Additionally, using one's own resolver avoids leaving traces in logs or caches of the preconfigured resolver. If the Windows stub resolver is configured to use one's own resolver, local queries can be modified at will. This could be used for phishing attacks (redirect to a proxy) or to prevent A/V software from updating. Furthermore, preconfigured resolvers might be rate-limited.

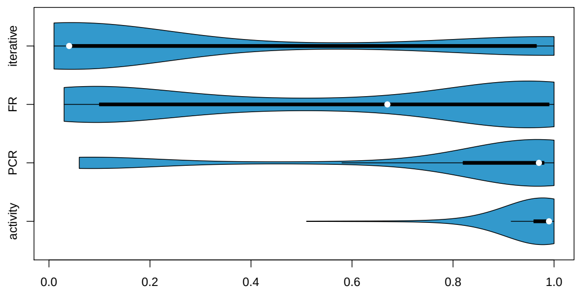


Figure 2: Violin plot of DNS activity end distribution

We found that 99% of the samples in S_{DNS} use the preconfigured resolver. Given this high ratio, a DNS resolver indeed turns out to be an interesting source for network-based malware detection - much more suitable than we had expected beforehand. We leave it up to future work to look into malware detection methods based on DNS resolver logs. 3% of S_{DNS} perform recursive DNS resolution with other resolvers than the preconfigured one (termed foreign resolvers in the following). Only 2% of S_{DNS} expose iterative DNS resolution. Note that the sets are not disjunct, as an execution may exhibit multiple resolution methods or resolvers. We speculate that this is due to the fact that malware occasionally downloads and executes multiple binaries, each of which might have different resolution methods. The foreign resolvers used include Google's Public DNS (used by 0.38%) as well as OpenDNS (0.25%). However, there is a large number of foreign resolvers that are used less frequently. One resolver that was located in China got our attention because queries for well-known popular domains such as facebook.com and youtube.com resolved into arbitrary IP addresses with no recognizable relation to the domain. We consider this to be an artifact of the so-called Great Firewall of China [10]. In total 932 of 5092 (18.3%) distinct DNS servers were used recursively at least once and thus can be regarded as publicly available recursive DNS resolvers.

Furthermore, we looked into the activity distribution of the different resolution methods (see Figure 2). The preconfigured resolver (PCR) was typically used throughout the whole analysis period. The end of the usage of foreign resolvers (FR) is wide-spread over time, leaning toward the end of the analysis. Interestingly, iterative resolution appears to end much sooner compared to the other resolution methods.

5.2 DNS TTL Analysis

The Time To Live parameter was of special interest to us, as it could be an indicator of fast flux usage. Fast flux is used as a means to provide flexibility among the C&C infrastructure of bots [12].

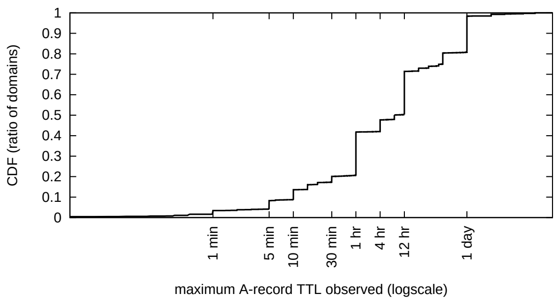


Figure 3: CDF of DNS TTL per domains

Figure 3 shows that 10% of all domains have a maximum TTL of 5 minutes or below. As spotted elsewhere [12], we expected domains with a small TTL and a large set of distinct answer records to be fast-flux candidates. However, when inspected manually, we found many domains of content distribution networks and large web sites. Using small TTLs seems to have become common among web hosters. As a result, the distinction between malicious fast-flux networks and legitimate hosting services becomes much more difficult. Interestingly, we also found a couple of responses with a TTL of zero that looked themselves like C&C communication. These responses were characterized by very long domain names as hex-strings. The TTL of zero prevents caching of these responses, effectively causing the resolver to always fetch the newest response from the authoritative DNS server. All in all, DNS suits well as a low-profile, low-bandwidth C&C channel in heavily firewalled environments, e.g. for targeted attacks.

5.3 DNS Message Error Rate

In order to measure DNS transaction failure, we defined the *DNS request error rate* as the number of DNS requests that were not successfully resolved over the total number of DNS requests. When aggregating the DNS message error rate per sample, we realized that for 10.1% of the samples in S_{Net} all of their DNS resolution attempts fail. However, the majority of the samples in S_{Net} (60.3%) have all DNS queries successfully resolved. The complete CDF is provided in Figure 4.

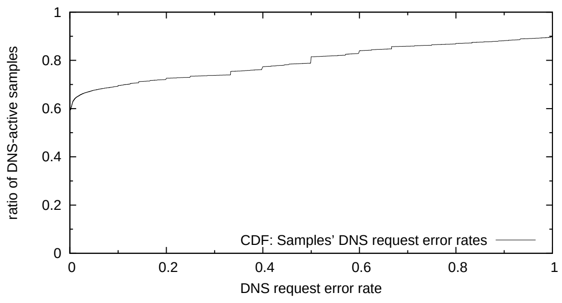


Figure 4: CDF of DNS message error rate

5.4 Resource Record Type Distribution

Figure 5 shows the distribution of the Resource Record types of the query section. Obviously, A records dominate DNS queries in Sandnet traffic, followed by queries for MX records. All samples in S_{DNS} have queried for an A record at least once. The high prevalence of A records is expected as A records are used to translate domain names into IP addresses. Furthermore, 2.3% of the samples in S_{DNS} queried blacklists. MX records have been queried by far less samples (8%). Interestingly, when comparing the MX query rate with SMTP activity, we have seen both: samples that performed MX lookups but had no SMTP activity and samples that successfully used SMTP but showed no MX queries at all. We assume that in the latter case, the required information on the MX destinations is provided via other means, e.g. C&C.

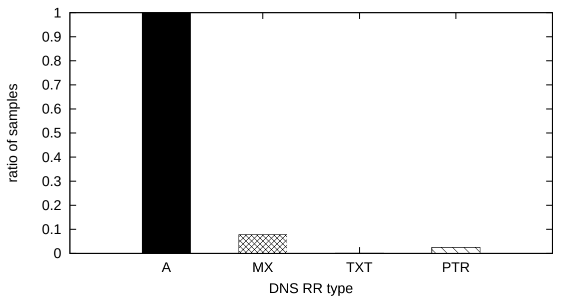


Figure 5: Resource Record distribution among samples

5.5 Resolution for Other Protocols

DNS, though itself a layer-7 protocol, plays a special role as it provides resolution service to all other layer-7 protocols. We analyzed how malware uses DNS before connecting to certain destinations. 23% of the samples in S_{DNS} show at least one flow without prior DNS resolution of the destination (DNS flows and scans excluded). In such a case either the destination's IP address is known (e.g. hard-coded in the binary) or resolution takes place via some other mechanism than DNS. A table providing flow destination DNS resolution by protocol can be found in annex I. Furthermore, 2.3% of the samples in S_{DNS} queried blacklists (26% of these also sent spam).

6. HTTP

HTTP traffic sums up to 88 GB inbound and 21 GB outbound, which makes HTTP by far the most prevalent protocol in Sandnet measured by traffic. The subset of samples using HTTP is denoted by S_{HTTP}. Given the high detail of the OpenDPI protocol classification, additional protocols that are carried in HTTP traffic are treated separately and thus contribute additional traffic: The Macromedia Flash protocol sums up to an additional 32 GB, video streams like MPEG and Apple Quicktime sum up to an additional 9 GB. We observed that the protocols carried in HTTP are usually caused by embedded objects included in websites that are visited by samples.

The immense potential abuse of HTTP-driven services motivated us to perform an in-depth analysis of typical malware HTTP traffic. Not only botnets started using HTTP as C&C structures. To name but a few, click fraud (i.e. the abuse of advertising services), mail address harvesting, drive-by downloads and DoS attacks on web servers are malicious activities of a wide range of malware authors. Of all samples with network activity (S_{Net}), the majority of 58.6% exposed HTTP activity. This section provides details to which extent, how, and why malware typically utilizes the HTTP protocol.

6.1 HTTP Requests

The analyzed samples typically act as HTTP clients and contact HTTP servers, mainly because the Sandnet communication is behind a NAT firewall. As a consequence, we can assume that virtually all recorded HTTP requests were made by malware. Figure 6 gives a general overview of how many HTTP requests malware typically made during the analysis period. The number of requests gives us a lead for which role malware has. Whereas one would expect a tremendous amount of requests during click fraud campaigns or DoS activities, malware update functionality and C&C channels potentially need little HTTP activity only. Interestingly, only 65% of the samples in S_{HTTP} made more than 5 HTTP requests. 16.3% of the samples in S_{HTTP} made only one HTTP request and then stopped their HTTP activity, although 70% of these samples continued with other network activity. We manually checked a fraction of these cases and found that many samples use HTTP to load second-stage binaries and continue with non-HTTP based damage functionality. The samples that completely ceased communicating after their initial HTTP flow presumably either failed to update themselves or waited for user-input triggers.

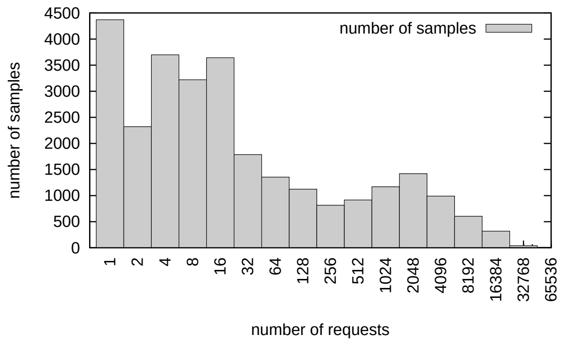


Figure 6: Histogram of HTTP Request Distribution

The GET request method was used by 89.5% of the samples in S_{HTTP}. We observed that 72% of the samples in S_{HTTP} additionally included GET parameters. Analysing just the fraction of GET requests with parameters, GET requests have on average 4.3 GET parameters. The average size of GET parameter were 12 characters for the key and 33.3 characters for the value. Although other means (such as steganography) allow to pass data to the sever, GET parameters seem to remain a popular method. On average, we have observed 1966 GET requests per sample with at least one request parameter. Interestingly, the number of unique GET parameter keys used by a sample is significantly lower than the total number of GET parameters per sample. This trend is particularly strong for samples with many parametrized GET requests and indicates that parameter keys are reused for follow-up requests. On average, the ratio between the number of distinct GET parameter keys and the total number of GET parameters is merely 1:16. We plan to further analyze the vast use of GET parameters, as started in [13], in the future.

The POST request method was used by 56.3% of the samples in S_{HTTP}. The average body size of POST requests is 739 bytes. We manually inspected a randomly chosen fraction of POST bodies to find out for what purpose malware uses POST requests. A large fraction of the inspected POST requests was used within C&C communication with a botnet server. We commonly observed that data passed to the server was base64-encoded and usually additionally obfuscated/encrypted. In addition, we frequently saw POST requests directed to search engines.

42% of the samples in S_{HTTP} used both POST and GET requests. Only 0.9% of the samples in S_{HTTP} showed HEAD requests at all. All other HTTP methods were used by less than 0.1% of the samples in S_{HTTP} and seem insignificant.

6.2 HTTP Request Headers

Annex C gives a comprehensive list of the 30 most popular HTTP request headers as observed in Sandnet. These HTTP headers include common headers usually used by benign web browsers. In total, we have observed 144 unique HTTP request headers. At a closer look at these, we identified a significant amount of misspelled or non-standard headers (excluding all extension headers, i.e. those starting with *'X-'*). Manual inspection shows that the fewer a specific header is used (in terms of samples), the more suspicious it is. Merely 5.7% of all samples in S_{HTTP} sent an HTTP request without any header at all. As a consequence, we see a need to further analyze specific request headers that we consider interesting.

6.2.1 User-Agent

In an ideal world, the HTTP *User-Agent* header specifies which exact web browser (including its version number) is requesting web content. However, the client and thus also malware samples can potentially forge the User-Agent header to be less suspicious. Annex B gives a detailed list of the 30 most popular raw User-Agent strings observed in Sandnet. Most samples (98.6% of S_{HTTP}) specified a User-Agent header at least once.

In an approach to get an overview of *actual* user agents we developed heuristics to filter the User-Agent list. First, we observed that 29.9% of the samples in S_{HTTP} specified wrong operating systems or Windows versions in their forged

HTTP User-Agent headers. Next, we identified that at least 13.4% of the samples in S_{HTTP} claim to use non-existing browser versions (e.g. *wget 3.0*, *Mozilla 6.0*). In addition, we saw that 37.8% of the samples in S_{HTTP} specified malformed or very short and to us unknown User-Agent values. In total, 67.5% of the samples in S_{HTTP} transmitted at least once a suspicious User-Agent string. Over the whole analysis period, only 31% of the samples in S_{HTTP} specified apparently correct User-Agent strings.

This result suggests that most samples have their own HTTP components that are bad in forging real web browsers. Interestingly, about half (50.6%) of the samples in S_{HTTP} change or alternate the User-Agent header during their analysis period. We hypothesize that this is due to the modular architecture of malware, where the modules have inconsistent User-Agent strings. Furthermore, based on this observation, we suspect that malware adapts the User-Agent header (and possibly other headers) depending on the target website.

6.2.2 Localization Headers

HTTP requests typically include headers that tell the server which languages and character sets the client understands (*Accept-Language* and *Accept-Charset*). We inspected these two localization headers and compared it with the locale setting (German) of the sandpuppets. While the Accept-Charset header was used by 0.35% of the samples in S_{HTTP}, the Accept-Language values are more interesting to analyze: In total, 44.3% of the samples in S_{HTTP} included Accept-Language as an HTTP request header. Of these samples, 24.1% did not respect the locale setting and specified a non-German language. Chinese (zh) and English (en) are the foreign languages specified most frequently, followed by Russian (ru). We speculate that in these cases malware authors forge HTTP headers either as observed at their own local systems or with respect to the target website. This would depict yet another indicator that malware carries its own (possibly self-made) HTTP implementation. Another reason could be that malware authors explicitly specify foreign languages to hoax web servers.

6.3 HTTP Responses

In Sandnet, all HTTP responses observed originated from HTTP servers on the Internet that were contacted by a sample. Therefore, the following analysis is not an analysis of the samples themselves, but may give indications to which type of servers malware communicates.

We observed that 97.8% of the HTTP requests were answered with an HTTP response. We define the *HTTP error rate* as the ratio between failed responses (HTTP status codes 4XX and 5XX) and all responses. Figure 7 shows a distribution of the sample-wise HTTP error rate. Only a small fraction (less than 10%) of samples virtually always get non-successful status-codes and apparently completely fail to retrieve the requested web content. Most samples have a relatively small error-ratio, indicating the web sites requested by the samples are still in place. We will give an overview of the requested servers in Section 6.6.

6.4 HTTP Response Headers

As opposed to HTTP request headers, response headers are set by servers and are not chosen by the malware samples. Analyzing the headers helps us to understand which servers are contacted by malware samples and gives information about the type of the retrieved content.

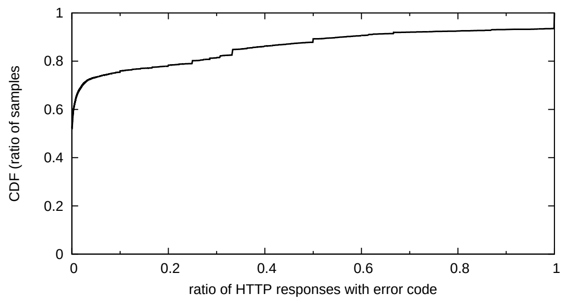


Figure 7: Distribution of HTTP error rates among samples

6.4.1 Content-Type

The Content-Type header shows which type of web content was retrieved by the samples. Figure 8 shows that most samples at least retrieve web sites with Content-Type *text/**. By far the most popular content-type of textual responses is *text/html.* However, only about half of all samples retrieved rich documents with Content-Type set to *images/** (48%) or *application/** (59.4%). 23.9% of the HTTP active samples with more than a single request got textual responses only. We see two reasons for such presumably light HTTP clients: First, spidering web sites without loading images is much more efficient. Second, we hypothesize that a considerable number of samples lacks a full-blown HTTP implementation that can recursively fetch objects embedded in web sites.

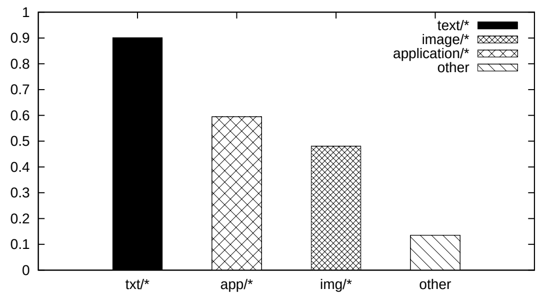


Figure 8: Ratio of samples using given Content-Type

6.4.2 Server

The *Server* HTTP response header indicates which type of web server is responding to the malware's HTTP request. Note that the content of this header can again be forged. Moreover, the majority of contacted web servers is presumably benign. However, when manually inspecting the HTTP Server response header, we spotted servers that presented suspicious banner strings. Annex E summarizes the list of the 30 most popular server types observed in Sandnet.

6.5 HTTP Responses with PE Binaries

After compromising a system with minimized exploits, attackers usually load so-called second-stage binaries. These binaries carry the actual malware functionality rather than just the exploit with minimized shell-code. In Sandnet, we usually analyze second-stage binaries instead of shell-code binaries. Yet, malware authors - as we will show - frequently load new portable executable (PE) binaries that expand or update the functionality of a malware sample. We assume this is due to a modular structure of the typical malware.

We extracted all binaries downloaded via HTTP by searching for the typical PE bytes in the body of HTTP responses. This straight-forward extraction of PE binaries already discovered that 16.7% of the samples in S_{Net} loaded additional PE files. To our surprise, we observed that 19% of these samples load binaries for multiple times - occasionally even more than 100 times. We verified that the five binaries downloaded most often were not corrupt and lack reasonable explanations why the binaries were downloaded that often. In total, we detected 42,295 PE headers, resulting in 17,676 unique PE files. The maximum size of a downloaded binary was 978 kB, the average size is 144 kB.

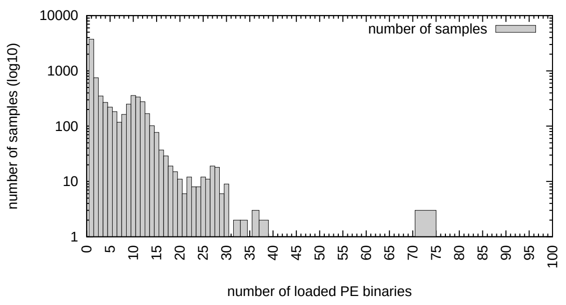


Figure 9: Distribution of # of PE binaries loaded

Figure 9 shows that most of the samples load more than a single PE binary. For readability of the graph we precluded 21 samples that loaded more than 100 and up to 1080 unique PE binaries. Annex D summarizes the Content-Type values of all HTTP responses that contain PE binaries. Most samples retrieve reasonable Content-Type values from the server. However, a significant number of servers tries to camouflage PE binary downloads as text, HTML, JavaScript or image files.

6.6 HTTP Servers

When recalling that HTTP is a protocol used by malware authors excessively, we see a need in analyzing which particular HTTP servers are visited by malware. We created a list of the 50 most popular domains ordered by the number of different samples visiting it in annex G. Obviously, many HTTP requests were put to presumably benign web sites. The next sections should briefly discuss why malware contacts these services.

6.6.1 Ad Services

We identified a significant number of ad service networks in the list of popular domains. Of the Top 50 domains in annex G, we manually identified 40 domains that are related to ads. Thousands of different malware samples use these services. A possible reason for this is that ads are included in virtually every web site and crawlers also follow the ads. However, after manually exploring the HTTP traffic of particular samples we assume that the reason for the popularity of ad services is vicious: click fraud. We leave it up to future work to analyze and mitigate the abuse of ad services by malware samples in greater detail.

6.6.2 Public Web APIs

Similarly to its popularity among benign users, Yahoo's and particularly Google's public Web APIs are present in Sandnet traffic, too. We suspect there are two reasons behind the popularity of these or similar services. First, some of these services are ubiquitous on the Internet. For example, a wide variety of web sites for example include Google Analytics to record statistics on the visitor behavior. Each time a sample visits such a web site and follows the embedded links, it will contact Google. As most of such services are open to anyone, we also suspect malicious usage of Google's and Yahoo's services by malware samples to be a reason for their popularity. A typical scenario that we observed was the abuse of search engines as a kind of C&C engine. In this case the malware searched for specific keywords and fetched the web sites suggested from the search results. Moreover, we have observed malware using the search engines to harvest new e-mail addresses for spamming campaigns. In general, benign human interaction with these services is particularly hard to be distinguished from abuse, especially from the client-perspective. We assume this is one of the main reasons malware authors use these HTTP-based services.

6.6.3 PE File Hosters

Based on the set of PE files that malware samples downloaded, we analyzed the file hosting servers. Annex H lists the most popular of all 1823 PE file hosters that we identified. 42.3% of the samples that downloaded PE files contacted the PE host directly without prior DNS resolution. This proves that still a significant number of malware samples include hard-coded IP addresses to download binaries. We further observed that a significant fraction of the URIs requested from file servers are non-static, although frequently only the parameters change. This observation may be important for blacklists trying to block entire URIs instead of IP addresses or domains.

6.6.4 HTTP C&C Servers

HTTP based botnets such as e.g. Torpig [15] switched from the classical IRC protocol to using HTTP. While manually inspecting Sandnet HTTP traffic, we occasionally encounter C&C traffic. What we see most is that samples include infection status information in their GET request parameters. Whereas some samples include clear-text status information, we and others [16] have observed many samples started encoding and encrypting the data exchanged with the server. However, we found it difficult to automatically spot C&C servers without knowing the command syntax of specific botnets. The big difference to IRC is that HTTP is a prevalent protocol on clean, non-infected systems and is thus harder to spot in the volume of HTTP data. Encouraged by the results reported in [9, 13], we believe that clustering the network behaviors of malware may help us in spotting generic C&C communication channels.

7. RELATED WORK

The malware phenomenon has been considerably studied over the last years by researchers and security practitioners. The community has proposed numerous techniques to collect [5], analyze [2, 8, 9, 11, 13, 17], or detect malware [9, 13].

For instance, Perdisci et al. [13] present an interesting system to cluster network-level behavior of malware by focusing on similarities among malicious HTTP traffic traces.

Similarly, Cavallaro et al. [9] present cluster-based analyses aimed at inferring interesting payload-agnostic network behaviors of malicious software. While Sandnet is currently limited to analyzing a large corpus of network protocols, it is clear how the adoption of similar cluster-level analyses can provide better understandings of the network behaviors of unknown software.

Anubis [7, 8] and CWSandbox [17] are probably the closest work related to our research. Although they both provide interesting—but basic—network statistics, their main goal is to provide insights about the host behaviors of unknown—potentially malicious—software. In this context, Sandnet complements Anubis and CWSandbox important results by providing an in-depth analysis of the network behaviors of the analyzed samples. As described elsewhere, this is only the first step toward a comprehensive understanding of the network activities perpetrated by such software. More analyses are currently being examined (e.g., [9, 13]) and are planned to extend Sandnet as part of our future research.

8. CONCLUSION AND FUTURE WORK

In this work, we presented a comprehensive overview of network traffic as observed by typical malware samples. The data was derived by analyzing more than 100k malware samples in Sandnet. Our in-depth analysis of DNS and HTTP traffic has shown novel malware trends and led to numerous inspirations to combat malware. The provided data is not only of great value because it is that detailed. It also perfectly complements related work that is either outdated, analyzes particular malware families only, or focuses mainly on the host behavior of malware. To share these insights with the research community, Sandnet is accessible via `http://www.if-is.net/sandnet/`.

We are currently expanding Sandnet to mitigate some of its current limitations and to perform a number of more detailed and sophisticated analyses, which will provide more insights into the behaviors of the network activities perpetrated by unknown, potentially malicious, software. For instance, clustering the network behavior of malware may automatically filter out uninteresting actions while unveiling the core patterns that represent the most interesting behavior of the malware [9, 13]. Furthermore, cross-correlation of network- and host-level clusters [6] may disclose interesting relationships among malware families. We also plan to assign public IP addresses to sandpuppets and to compare the malware behavior with a restricted, NATed network breakout. Similarly, we plan to integrate the analysis of system-level activities to Sandnet, such as linking process information to network activity. Including observations on how processes react to varying network input could further help to identify C&C channels. In addition, we strive to a more accurate view on the analysis data, particularly to distinguish benign from malicious communication endpoints.

Another direction our research may suggest is tailored toward performing a more detailed analysis of ad service abuse, especially click fraud. We plan on exploring click fraud detection mechanisms derived from the web site request behavior of malware observed in Sandnet. Possibly, we will expand this idea by also inspecting the abuse of public web services (e.g. the Google API).

We believe the data collected and analyzed by Sandnet represents a first step toward a comprehensive characterization of the network behaviors of malware. Driven by recent results, we thus hope our ongoing research to be of a great value to researchers and practitioners to help them acquiring a more detailed understanding of such behaviors. Not only this enables the development of more effective countermeasures and mitigation techniques, but it may also help to understand the social trends and facts of the underground malware economy.

9. ACKNOWLEDGEMENTS

Malware samples are to a great degree provided by partner research institutions. We thankfully acknowledge their generosity to let us use the samples.

10. REFERENCES

[1] Hispasec Sistemas - VirusTotal. `http://www.virustotal.com/`.

[2] Norman Sandbox. `http://www.norman.com/security_center/security_tools/`.

[3] Sandnet. `http://www.if-is.net/sandnet/`.

[4] The Shadowserver Foundation - bin-test. `http://bin-test.shadowserver.org/`.

[5] P. Baecher, M. Koetter, M. Dornseif, and F. Freiling. The Nepenthes Platform: An Efficient Approach to Collect Malware. In *RAID 2006*.

[6] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Kruegel, E. Kirda, and S. Barbara. Scalable, Behavior-Based Malware Clustering. In *NDSS 2009*.

[7] U. Bayer, I. Habibi, D. Balzarotti, E. Kirda, and C. Kruegel. A View on Current Malware Behaviors. In *USENIX LEET 2009*.

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

[9] L. Cavallaro, C. Kruegel, and G. Vigna. Mining the Network Behavior of Bots, Technical Report, 2009.

[10] R. Clayton, S. J. Murdoch, and R. N. M. Watson. Ignoring the Great Firewall of China. In *Privacy Enhancing Technologies*, pages 20–35, 2006.

[11] J. Goebel and T. Holz. Rishi: Identify Bot Contaminated Hosts by IRC Nickname Evaluation. In *USENIX HotBots '07*.

[12] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling. Measuring and detecting fast-flux service networks. In *NDSS*, 2008.

[13] R. Perdisci, W. Lee, and N. Feamster. Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces. In *NSDI 2010*.

[14] K. Rieck, P. Trinius, C. Willems, and T. Holz. Automatic Analysis of Malware Behavior using Machine Learning. In *Journal of Computer Security 2011*.

[15] B. Stone-Gross, M. Cova, B. Gilbert, L. Cavallaro, M. Szydlowski, C. Kruegel, G. Vigna, and R. Kemmerer. Your Botnet is My Botnet: Analysis of a Botnet Takeover. In *CCS 2009*, Chicago, IL, November 2009.

[16] M. van den Berg. A Taste of HTTP Botnets, 2008.

[17] C. Willems, T. Holz, and F. Freiling. Toward Automated Dynamic Malware Analysis Using CWSandbox. In *IEEE Security and Privacy Magazine*, volume 5, pages 32–39, March 2007.

APPENDIX

A. GENERAL TRAFFIC OVERVIEW

L7 Protocol	Samples	Flows	Bytes	Out Bytes	In Bytes	Destinations	Dst Domains
DNS	42143	11845193	3730 MB	1355 MB	2375 MB	241126	14732
HTTP	26738	13492189	110 GB	21 GB	88 GB	36921	55032
Unknown	18349	32265514	24 GB	14 GB	10 GB	9145625	86523
Flash	5881	299986	32 GB	692 MB	31 GB	2955	2205
SSL	5104	79344	1884 MB	139 MB	1745 MB	2278	1622
SMB	4275	8602414	6116 MB	4210 MB	1906 MB	7253975	10
IRC	3657	169833	70 MB	15 MB	55 MB	564	554
SMTP	1715	3155014	20 GB	19 GB	1124 MB	282401	118959
MPEG	1162	2200	220 MB	1050 kB	219 MB	58	44
SSDP	885	1861	3651 kB	3651 kB	0 bytes	2	0
Quicktime	389	1222	8315 MB	1518 kB	8313 MB	62	41
FTP	243	7523	3144 kB	860 kB	2285 kB	159	121
NetBIOS	184	134600	54 MB	36 MB	18 MB	108909	0
TDS	163	1086	31 MB	1044 kB	30 MB	44	36
NTP	102	2950	266 kB	156 kB	109 kB	13	5
STUN	68	276	71 kB	54 kB	18 kB	19	8
TFTP	48	12492	626 MB	5165 kB	621 MB	19	0
PPLIVE	37	1481	85 MB	9042 kB	76 MB	1321	0
Gnutella	32	20545	181 MB	102 MB	79 MB	15640	0
DDL	28	277	29 MB	140 kB	29 MB	52	35
Bittorrent	26	1180	147 MB	5090 kB	142 MB	588	32
Mysql	21	33	38 kB	4288 bytes	34 kB	12	7

B. HTTP USER AGENTS

User Agent	Requests	Samples
Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0.507	17193201	11168
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)	861353	5628
Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0.507	1937020	5376
Microsoft-CryptoAPI/5.131.2600.5512	17581	3485
Mozilla/6.0 (Windows; wget 3.0)	12851	3242
Download	5022	2042
Mozilla/4.0 (compatible; MSIE 8.0.6001.18702; Windows NT 5.1.2600)	23022	1802
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)	12569	1546
ClickAdsByIE 0.7.3	34615	1208
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1)	69078	992
XML	3403	891
Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; SV1)	1714	849
PinballCorp-BSAI/VER_STR_COMMA	3454	771
Mozilla/3.0 (compatible; Indy Library)	71971	761
Microsoft Internet Explorer	8652	750
gbot/2.3	22791	694
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; .NET CLR 1.1.4322)	23772	608
_	5327	589
NSISDL/1.2 (Mozilla)	692	535
Microsoft-ATL-Native/9.00	3827	524
Mozilla/5.0 (Windows; U; Windows NT 5.1; de; rv:1.9.2.6) Gecko/20100625 Firefox/	31078	514
Mozilla/4.0 (compatible)	6004	487
Mozilla/4.0 (compatible; MSIE 8.0; 10.1.53.64; Windows NT 5.1)	884	426
NSIS_Inetc (Mozilla)	515	403
wget 3.0	3917	339
Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; .NET CLR 1.1.4322)	946	311
opera	946	300
Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.3) Gecko/20100401 Firef	6764	300

C. HTTP REQUEST HEADERS

HTTP Header	Samples	HTTP Requests
Host	27771	21054208
User-Agent	26359	20923840
Connection	21205	20570434
Cache-Control	18529	1346260
Accept	18483	20554040
Content-Length	14811	977547
Accept-Encoding	14406	19424065
Content-Type	14135	1033111
Accept-Language	11382	18319897
Referer	10079	18311670
Cookie	10075	10939127
If-Modified-Since	5462	3044837
If-None-Match	4696	1005364
x-flash-version	4386	464334
Pragma	4290	73427
x-requested-with	2079	14329
Range	1597	15451
If-Range	1006	3882
Unless-Modified-Since	962	3868
Accept-Charset	922	69908
X-Agent	658	36302
Keep-Alive	642	87149
X-Moz	511	517
Content-length	438	1494
x-prototype-version	408	1831
http	287	13984
UA-CPU	208	12899
x-svn-rev	111	351
x-type	84	167
Content-type	81	4876

E. HTTP SERVER TYPES

Server	Ratio (%)	Servers
Apache	68.4	326237
Microsoft-IIS	49.4	102652
nginx	40.9	108104
Golfe	21.4	20534
lighttpd	21.4	32934
YTS	20.0	28320
sffe	19.4	15128
GFE	18.3	21089
Apache-Coyote	17.6	41875
QS	15.4	6906
PWS	14.7	16297
DCLK-AdSvr	13.9	6782
cafe	13.7	11399
AmazonS3	13.7	17203
ADITIONSERVER 1.0	10.9	6092
AkamaiGHost	10.3	3520
Cookie Matcher	10.1	4011
gws	9.5	6075
VM_BANNERSERVER 1.0	9.4	2620
CS	9.1	3987
Adtech Adserver	8.9	4842
CacheFlyServe v26b	8.0	2242
RSI	7.8	2196
yesup httpd 89	7.8	2160
yesup httpd 103	7.7	2151
Resin	7.7	4375
ECS (fra	6.7	7615
Oversee Turing v1.0.0	6.1	2579
JBird	6.0	1987
TRP Apache-Coyote	5.7	1966

D. CONTENT-TYPE OF PE DOWNLOADS

Content-Type	# Binaries	# Samples
application/octet-stream	6468	5908
text/plain	356	1716
application/x-msdownload	732	1082
application/x-msdos-program	550	786
image/gif	177	402
image/jpeg	390	365
text/plain; charset=UTF-8	166	344
text/html	776	326
application/x-javascript	190	78
image/png	68	55

F. DATABASE STATISTICS

Attribute	Value
Distinct samples	104,345
Total traffic	207 GB
Outbound traffic	61 GB
Inbound traffic	146 GB
Number of Flows	70,106,728

G. HTTP SERVERS

HTTP domain	# Samples
www.google-analytics.com	5286
ad.yieldmanager.com	5046
cookex.amp.yahoo.com	4716
content.yieldmanager.com	4655
ak1.abmr.net	4288
pixel.quantserve.com	4050
content.yieldmanager.edgesuite.net	4009
edge.quantserve.com	3957
ad.doubleclick.net	3677
ad.harrenmedianetwork.com	3470
ad.103092804.com	3458
s0.2mdn.net	3370
ib.adnxs.com	3280
pixer.meaningtool.com	3219
ad-emea.doubleclick.net	2972
www.google.com	2940
ad.harrenmedia.com	2920
www.mupimg.de	2823
imagesrv.adition.com	2770
www.mupads.de	2759
view.atdmt.com	2754
ad.xtendmedia.com	2726
cm.g.doubleclick.net	2669
googleads.g.doubleclick.net	2657
fpdownload2.macromedia.com	2619
www.myroitracking.com	2573
serw.clicksor.com	2489
ad.adition.net	2468
ads.clicksor.com	2466
ad.tlvmedia.com	2449
ad.adserverplus.com	2414
b.scorecardresearch.com	2376
pub.clicksor.net	2375
ajax.googleapis.com	2335
img.billiger.de	2308
tags.bluekai.com	2308
adx.adnxs.com	2287
adfarm1.adition.com	2286
admax.quisma.com	2201
pagead2.googlesyndication.com	2199
a.collective-media.net	2115
ads.revsci.net	2099
pix04.revsci.net	2065
js.revsci.net	2050
ad.globe7.com	2040
staging.pixer.meaningtool.com	2030
ad.reduxmedia.com	2028
suresafe1.adsovo.com	2012
adserver.adtech.de	2010
crl.verisign.com	1999

H. PE FILE HOSTERS

PE File Server	#S	#B
64.79.86.26	775	1340
66.96.221.102	681	1063
ku1.installstorm.com	487	944
origin-ics.hotbar.com	483	483
64.191.44.9	480	727
img.ub8.net	458	460
pic.iwillhavesexygirls.com	437	478
64.120.232.147	431	747
origin-ics.clickpotato.tv	390	390
p2pshares.org	389	391
sky.installstorm.com	363	363
208.43.146.98	331	531
file0129.iwillhavesexygirls.com	323	853
173.45.70.226	315	437
173.45.70.227	313	444
dl.ghura.pl	302	302
122.224.6.48	300	553

#S = number of samples contacting file hoster
#B = number of binaries downloaded

I. DNS RESOLUTION BY PROTOCOL

Protocol	Samples (%)
NetBIOS	0.00
MSN	0.00
SMB	0.00
SIP	0.00
DHCP	0.00
TFTP	0.00
Gnutella	0.00
STUN	0.00
SSDP	0.00
mDNS	0.00
Bittorrent	19.23
TDS	39.88
SMTP	41.05
Unknown	46.84
FTP	47.74
DDL	53.57
Oscar	57.89
Mysql	66.67
HTTP	67.72
IRC	80.45
NTP	82.35
POP	83.33
Flash	83.63
SSL	87.93
Quicktime	93.57
MPEG	96.04

Toward a Standard Benchmark for Computer Security Research

The Worldwide Intelligence Network Environment (WINE)

Tudor Dumitraș
Symantec Research Labs
tudor_dumitras@symantec.com

Darren Shou
Symantec Research Labs
darren_shou@symantec.com

Abstract

Unlike benchmarks that focus on performance or reliability evaluations, a benchmark for computer security must necessarily include sensitive code and data. Because these artifacts could damage systems or reveal personally identifiable information about the users affected by cyber attacks, publicly disseminating such a benchmark raises several scientific, ethical and legal challenges. We propose the Worldwide Intelligence Network Environment (WINE), a security-benchmarking approach based on rigorous experimental methods. WINE includes representative field data, collected worldwide from 240,000 sensors, for new empirical studies, and it will enable the validation of research on all the phases in the lifecycle of security threats. We tackle the key challenges for security benchmarking by designing a platform for repeatable experimentation on the WINE data sets and by collecting the metadata required for understanding the results. In this paper, we review the unique characteristics of the WINE data, we discuss why rigorous benchmarking will provide fresh insights on the security arms race and we propose a research agenda for this area.

1. Introduction

The security-related data sets that are available today are insufficient for answering many challenging questions or for rigorous experimental research. For example, little is known about the origins and prevalence of zero-day attacks, because the existing data on malware dissemination does not reach back in time *before* the discovery of the malware. We currently do not understand how scam sites conceal their presence and move to avoid detection, for lack of historical information on malicious URLs. So far, we have not been able to follow a security vulnerability over the course of its entire life—from a programming bug that evades testing, through its stealth exploitation in zero-day attacks, its discovery and description in a public advisory, the release of a patch for the vulnerability and of anti-virus signatures, the automatic generation of exploits based on the patch, and to the final race between these attacks and the remediation measures introduced by the security community. Answering such questions requires the analysis and the correlation of multiple data sets, collected independently from diversified sensors. The lack of such data sets prevents us from gaining the deep insights needed for tipping the balance of the security arms race from the attackers to the defenders.

Moreover, data sets used for validating computer security research are often mentioned in a single publication and then forgotten. For example, real malware samples are readily available on the Internet, and they are often used for validating research results. However, this experimental method does not accommodate a *sound validation* of the research, because other investigators do not have access to the same collection of samples and cannot reproduce the results. This prevents rigorous comparisons between alternative approaches proposed in the scientific literature. Additionally, the malware samples alone do not tell the whole story. Ancillary *field data* is needed to understand the malware lifecycle and the economic incentives of cybercrime.

We aim to fill these gaps by (i) making representative field data, which covers the entire lifecycle of malware, available to the research community, and (ii) developing a platform for repeatable experimentation around these data sets. We build on the lessons learned in other research fields where benchmarking is well established (*e.g.* networking and databases), while identifying some of the key differences for security benchmarking.

We center our benchmarking approach around the data sets available in WINE[1], Symantec's program for sharing data with the research community. For example, WINE includes information on *unknown binaries* found on the Internet. The users who opt in for the reputation-based security features of Symantec products accept to share the list of binary files downloaded on their machines in exchange for a whitelist of binaries with good reputation. The data includes historical in-

[1] More information on accessing the WINE data is available at `http://www.symantec.com/WINE`.

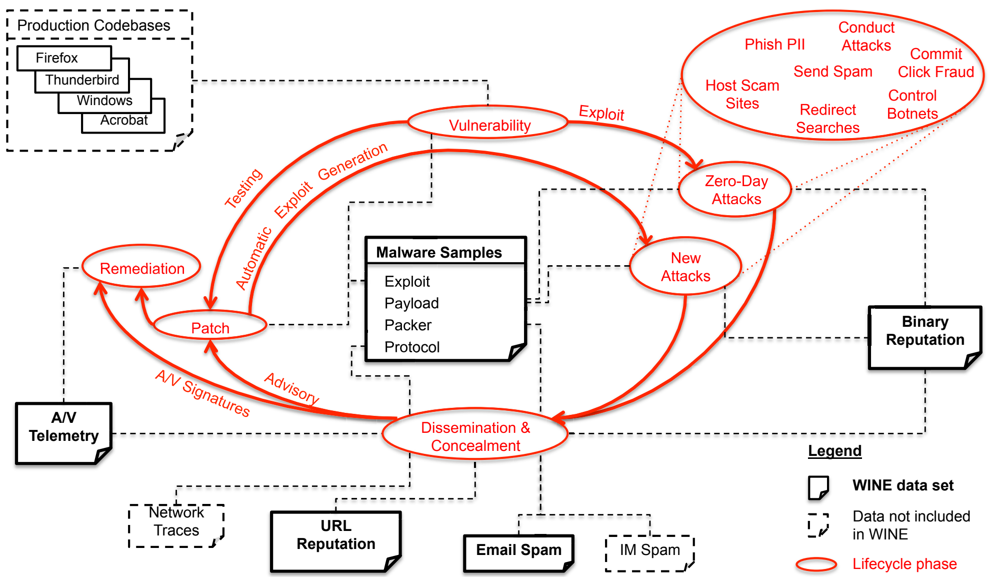


Figure 1. The WINE data sets enable the study of the entire lifecycle of security threats. By correlating research findings with additional data sets, available from other sources, experimenters can assemble an end-to-end image of the security arms race.

formation on 1 billion files that the security community has not yet classified as either benign or malware. The historical records start when each file appeared on the Internet (estimated through the discovery timestamps assigned by the 50 million active instances of the reputation feature) and can provide unique insights on the mechanisms of zero-day attacks. Similarly, Symantec tracks the spread of known host-based and network-based cyber threats, filters spam out of approximately one third of the world's email and has assembled perhaps the largest collection of malware samples. By combining five distinct data sets, sampled from this collection of field data, WINE provides an overview of the security-threat landscape (see Figure 1).

We are currently developing a data storage and analysis platform, which aims to ensure *experimental repeatability* by archiving snapshots of the data used in each experiment and by providing researchers with tools for recording all the information required for reproducing the results. This will enable comparisons of the effectiveness, performance and scalability of published techniques. Moreover, WINE will include *metadata* allowing researchers to establish whether a data set is representative for the real-world cyber threats. To protect the sensitive information included in the data and to ensure the reproducibility of experimental results, all the experiments and empirical studies will be conducted on the WINE platform hosted by Symantec Research Labs.

Our ultimate goal is to develop a rigorous benchmark for computer security research. Because defensive mechanisms can make different trade-offs, which might be appropriate for different systems and settings, we will avoid reporting a single number indicating which mechanism is the best. Like the TPC and SPEC benchmarks, which focus on performance evaluation, our security benchmark will not be definitive. The WINE data sets must be updated periodically in order to reflect the frequent changes in the security threat landscape.

While the WINE data sets are currently available to the research community, the data *per se* is not sufficient for defining a rigorous benchmark. In this position paper, our goal is not to present benchmark results or to discuss the lessons learned from this effort. Instead, we make two contributions:

- We propose a research agenda for security benchmarking, by identifying the main challenges (Section 2) and several open questions that could be answered once these challenges are overcome (Section 4);
- We propose an approach for benchmarking computer security (Section 3), which combines the WINE data sets with a platform for rigorous experimentation. We explain WINE's data sharing model, and we outline solutions to some of the key challenges for security benchmarking.

Our data sharing program does not focus exclusively on computer security—enabling, for example, research on software reliability or on machine learning techniques for billion-node graphs. Moreover, the results of experimental research will guide the inclusion of additional data sets in WINE. We believe that, in the future, the WINE data will provide key insights for the fields of security, dependability, machine learning and software engineering.

2. Challenges for benchmarking security

Unlike in the systems community, where data sets have sometimes outlived the system for which they were collected,[2] the

[2] For example, in case of the Sprite filesystem trace [Baker et al. 1991].

data sets used for validating computer-security research are often forgotten after the initial publication referencing them. This experimental method does not accommodate an independent verification of results and meaningful comparisons against the prior art. The lack of standard benchmarks for computer security is the result of scientific, ethical, and legal challenges for publicly disseminating security-related data sets. In this paper we focus on the scientific challenges, but we also review other challenges that are likely to have an impact on the benchmarking techniques.

2.1 Scientific challenges

ℂ1 *A benchmark for computer security must be based on field data.* Some benchmarking efforts in the past have addressed privacy concerns by generating synthetic data, based on the observed statistical distributions of the raw data samples collected [Lippmann et al. 2000]. Moreover, synthetically generated data provides considerable flexibility, allowing an experimenter to explore all the behavioral corner cases of the system-under-test [DeWitt 1993]. For security-oriented benchmarks, however, it is difficult to relate the benchmarking results to the real-world performance of the system-under-test. For example, the false positive rate of intrusion detection systems is influenced by the background noise, which should be consistent with the background data that the system is likely to encounter in a real deployment [McHugh 2000].

ℂ2 *The benchmarking approach must ensure experimental repeatability.* The data sets used in the experiments must be archived for future reference, and they must be considered again in research projects attempting quantitative comparisons against the prior results. Moreover, in order to make it possible for future projects to reproduce the experimental results, the benchmark must provide tools for recording the *experiment metadata*—*e.g.*, the hypotheses tested, the experimental design, the scripts and procedures used for data analysis, the statistical apparatus employed.

ℂ3 *The benchmark must be representative of the real-world threat landscape.* Any large data collection can ensure the statistical significance of the experimental results. However, the validity of these results can still be questioned in cases where small mutations of the test data can drastically change the outcome of the experiment. The benchmark should provide the *collection metadata* needed for establishing the real-world situations that each data set is representative of. Moreover, the benchmark must remain relevant, in spite of the frequent changes in the cyber threat landscape and of data filtering at multiple levels (see also Challenges ℂ5 and ℂ6). We point out that updating the benchmark regularly does not conflict with ℂ2. The benchmark must specify a predictable process for data collection [Camp et al. 2009], while preserving the reference data sets employed in prior experiments. Similarly, as security metrics are not well understood, the benchmark must suggest metrics in order to enable direct comparisons among similar techniques, but must allow researchers to define improved metrics that are more relevant for the hypotheses tested.

ℂ4 *Experiments must be conducted at a realistic scale.* Security is difficult to measure and assess objectively because it represents an end-to-end property of the system. Some metrics (*e.g.* resistance to intrusions) can not be measured directly and must be approximated through large-scale observations of the whole system, in order to achieve precise estimations.

ℂ5 *Benchmarking must take the information quality into account.* In many large scale collections, uncertainty about the data is explicit. For example, as heuristics and machine-learning techniques are used, increasingly, for detecting polymorphic malware, the labels applied to the binaries analyzed are no longer a black-and-white determination, but, rather, they express a certain level of confidence that the binary is malicious. In a commercial product, where monitoring and logging represent secondary concerns, the submissions are throttled back, and sometimes truncated, in over to avoid overloading the users' machines and to reduce the bandwidth costs incurred. Moreover, the hash functions used for identifying binaries may change, as the products evolve, and the techniques used for identifying user machines are not always reliable. We must develop new query languages and analysis tools that treat such information-quality metrics as first-class entities.

2.2 Ethical challenges

ℂ6 *Do no harm.* A benchmark for computer security must include sensitive code and data, which could damage computer systems or could reveal personally identifiable information about the users affected by the cyber attacks recorded. For example, the IP addresses of hosts initiating network-based attacks could point to personal computers that have been infected with malware, while the country codes of the attack destinations reveal further sensitive information. Binary samples of malware must not be made freely available on the Internet. It is challenging to determine, *a priori*, how to sample or filter the raw data collected in order to meet these challenges.

2.3 Legal challenges

ℂ7 *Compliance with privacy laws often restricts the data collection, storage and exchange.* The field data needed for security benchmarking (see Challenge ℂ1) is collected from real networks and users. There are several laws that limit access to network traffic or that regulate the storage of this information. In the United States, for example, the Wiretap Act prohibits the interception of content of electronic communications, the Pen/Trap statute prohibits the real-time interception of non-content, and the Stored Communications Act prohibits providers from knowingly disclosing their customer's communications. In contrast to HIPAA, which restricts disclosures of health information but provides means for researchers to obtain relevant information, the privacy laws contain no exceptions for research. The PREDICT project [DHS 2011b], sponsored by the Department of Homeland Security, could provide a framework for addressing this challenge.

3. A benchmark for computer security

We build upon the lessons learned from the failures and successes of the previous efforts for benchmarking computer security [for example: Camp et al. 2009, Leita et al. 2010, Lippmann et al. 2000, Maxion and Townsend 2004, McHugh 2000] and for building platforms allowing rigorous measurements and experimentation [for example: DeWitt 1993, Eide et al. 2007, Paxson 2004]. In addition to archiving snapshots of the data sets used in each experiment, we will store the scripts used for aggregating and analyzing the data, and we will maintain a *lab book* that records all the steps taken by the experimenter. This experimental metadata is essential for ensuring the reproducibility of the results (challenge $\mathbb{C}2$). Keeping a lab book is a common practice in other experimental fields, such as applied physics or cell biology.

The selection of the initial data sets for WINE was guided by our goal to establish a benchmark for computer security and by the needs expressed in the security community [Camp et al. 2009]. However, the access to the WINE data is not restricted to security researchers. WINE aims to aggregate the data feeds collected by Symantec in order to enable experimental research across a broad spectrum of disciplines, *e.g.*, dependability, machine learning, software engineering, networking, economics, visual analytics.

3.1 Operational model

To protect the sensitive information included in the data sets, WINE will only be accessed on-site at Symantec Research Labs. While researchers will have access to the raw data collected, *we will not create a malware library for anyone to download at will, and we will ensure that private information is not disseminated in public* (challenge $\mathbb{C}6$). Moreover, some aspects of the data collection process, such as the internal operation of the various Symantec sensors, will not be disclosed in detail. A snapshot of the data used in each experiment will be archived, for future reference, and all the analysis and experimentation will be conducted on the WINE infrastructure (described in Section 3.3). The researchers will retain all right, title and interest to the research results.

More information on accessing WINE is available at `http://www.symantec.com/WINE`.

3.2 The WINE data sets

WINE will provide access to a large collection of malware samples, and to the contextual information needed to understand how malware spreads and conceals its presence, how it gains access to different systems, what actions it performs once it is in control and how it is ultimately defeated. WINE includes representative field data, collected at Symantec (challenge $\mathbb{C}1$). WINE will include five data sets, summarized in Table 1: binary-reputation data, email-spam data, URL-reputation data, A/V telemetry and malware samples. These data sets enable two research directions: (i) empirical studies for *understanding each phase in the lifecycle of cyberattacks*, and (ii) quantitative evaluations and comparisons of attack prevention or detection techniques, for *benchmarking security systems*.

Understanding the lifecycle of cyberattacks. WINE aims to cover the entire lifecycle of malware attacks (see Figure 1). For example, the binary-reputation data set enables—for the first time, to the best of our knowledge—a study of the origins and prevalence of *zero-day attacks*, which exploit vulnerabilities that are unknown or unacknowledged publicly. Searching the history of binary-reputation submissions for files that are known to be malicious indicates for how long the file has existed in the wild before it was first detected (*i.e.*, before the security community created the corresponding anti-virus signatures). The subsequent proliferation of the attack and the *effectiveness of the remediation mechanisms* introduced (*e.g.*, patches for the vulnerability exploited, A/V signatures for detecting and blocking the attack) can be further traced in the A/V telemetry data set.

Similarly, by correlating the URLs recorded in the email spam samples, in the binary reputation and in the URL reputation data sets, we can begin to understand *how scam sites conceal themselves* to avoid detection (*e.g.*, by moving to a different IP address) and the effectiveness of the various *mechanisms for disseminating malware* (*e.g.*, spam, intrusions, drive-by downloads). The malware samples in WINE illustrate the attackers' aims—the actions that malware tries to perform once it takes control of a host—, and by corroborating these observations with data from the real-world victims of these attacks we can gain insight into the economic incentives of cybercrime. The data sets included in WINE are collected independently, from diversified sensors, allowing researchers to examine a phenomenon from multiple perspectives and to improve the confidence in the conclusions we draw from these investigations (challenge $\mathbb{C}3$).

Moreover, by combining WINE with data from additional sources, such as code repositories for open source software that have known vulnerabilities, we can study a security threat from the time when a programming bug introduces a vulnerability until the time when the last exploit of that vulnerability disappears from the A/V telemetry.

Benchmarking computer security. Because most of the techniques developed in the security community can serve both sides of the arms race, defensive mechanisms usually aim to force attackers to do more work than defenders have to do. WINE allows testing this, or similar, hypotheses for existing security systems, by defining *macro-benchmarks* that are representative for real-world workloads of systems aiming to fight viruses, worms or botnets. For example, the telemetry data can serve as the ground truth for heuristic threat-detection

Data set	Sources	Description
Binary reputation	50 million machines	Information on unknown binaries—*i.e.*, files for which an A/V signature has not yet been created—that are downloaded by users who opt in for Symantec's reputation-based security program. This data can indicate for how long a particular threat has existed in the wild before it was first detected. Each record includes the submission timestamp, as well as the cryptographic hash and the download URL of the binary.
A/V telemetry	130 million machines	Records occurrences of known threats, for which Symantec has created signatures and which can be detected by anti-virus products. This data set includes intrusion-detection telemetry. Each record includes the detection timestamp, the signature of the attack, the OS version of the attack's target, the name of the compromised process and the file or URL which originated the attack.
Email spam	2.5 million decoy accounts	Samples of phishing and spam emails, collected by Symantec's enterprise-grade systems for spam filtering. This data set includes samples of email spam and statistics on the messages blocked by the spam filters.
URL reputation	10 million domains	Website-reputation data, collected by crawling the web and by analyzing malicious URLs (a simplified interface for querying this data is available at `http://safeweb.norton.com/`). Each record includes the crawl timestamp, the URL, as well as the name and the type of threat found at that URL. A subset of this data was used to analyze the rogue A/V campaigns [Cova et al. 2010].
Malware samples	200 countries	A collection of both packed and unpacked malware samples (viruses, worms, bots, etc.), used for creating Symantec's A/V signatures. A subset of these samples was used for validating research on automatic malware detection [Griffin et al. 2009].

Table 1. The WINE data sets.

algorithms that operate on the binary-reputation data set. The data is also amenable to the statistical techniques that have been proposed in the past for insider attack attribution, such as naïve Bayes classification, Markov modeling or temporal sequence matching [Maxion and Townsend 2004].

These macro-benchmarks provide a corpus of field data for present and future experimenters, allowing them to measure multiple characteristics of a security tool, such as its latency, its scalability, and its threat detection accuracy. Because popular benchmarks can have a lasting impact on the design of security systems, we will regularly update the WINE data to ensure that the benchmarks are representative of the threat landscape in the real world (challenge $\mathbb{C}3$).

3.3 Experimental approach

The WINE data sets described above represent only half of the security benchmark. To achieve experimental reproducibility, we are currently building a platform for storing and analyzing the data. This platform enables data-intensive applications by adopting a *shared-nothing* architecture, illustrated in Figure 2. The data is partitioned across multiple storage nodes, attached directly to the hosts that execute data analysis tasks. The management infrastructure of the cluster minimizes the amount of data that must be transferred through the local area network by placing, whenever possible, the analysis tasks directly on the nodes that already store the data required. This is achieved by maintaining multiple indexes for each data set and by making these indexes available on all the nodes of the system. For example, the binary-reputation data set is indexed on both the hash of the binary and the download URL, to facilitate the correlation of data with the A/V telemetry, as well as with the email spam and URL-reputation data. This design will allow researchers to run experiments at scale (challenge $\mathbb{C}4$).

The experimental platform allows querying the data sets using either ANSI SQL or MapReduce tasks [Dean and Ghemawat 2004], for greater flexibility. WINE receives updates regularly from Symantec's collection of 240,000 sensors, which are distributed worldwide. Based on the raw data available in WINE, researchers define *reference data sets* that are relevant for their experiments. After the experiments are completed, the reference data sets are archived in network-attached storage, for future comparisons against the results obtained (challenge $\mathbb{C}2$).[3]

This design is similar to other architectures for data-intensive computing, such as MapReduce or parallel databases [Pavlo et al. 2009]. Unlike the prior work, we aim to ensure the *experimental reproducibility*, within the context of Symantec's data collection process. This goal will be achieved, in part, by providing integrated tools to help researchers manage and record their activities, either planned or unplanned [Eide et al. 2007]. These tools will facilitate the development of scripts that repeat the experimental procedure, *e.g.* by recording the interactive terminal sessions, and they will provide a

[3] The malware data set is stored and analyzed in a *red lab*, which does not have inbound/outbound network connectivity in order to prevent viruses and worms from escaping this isolated environment (challenge $\mathbb{C}6$).

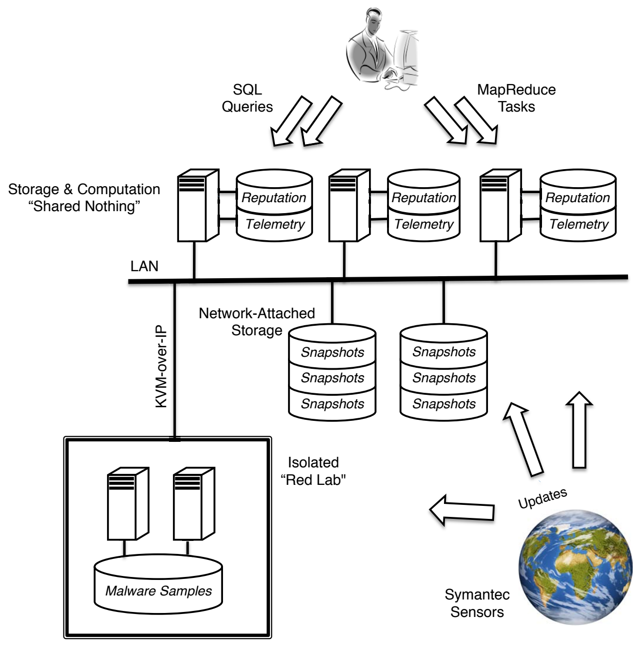


Figure 2. Architecture of the WINE platform. WINE is a data-intensive system, which focuses on ensuring the reproducibility and comparability of experimental results.

detailed record of the experiment. However, the lab book will also require a conscious effort from the researcher for documenting the experimental hypothesis and the purpose of each procedural step (challenge ℂ2). For example, when creating a taxonomy of the malware samples included in WINE, the lab book should detail the rationale for the selection of each classification feature.

Moreover, we will implement mechanisms for assessing the *information quality*, which is a measure of how fit the information is for benchmarking purposes [Keeton et al. 2009]. For example, as MapReduce is known to exhibit a significant response-time variability [Zaharia et al. 2008], we will estimate the measurement *precision* by repeating an experiment multiple times and recording the standard deviation of the results [Chatfield 1983]. When supplementing the data sets with information collected on the server-side—*e.g.*, by performing a reverse DNS query on a IP address that is observed to be the source of an attack, in order to determine its network location before the DNS record is deregistered—we will assess the data *staleness* by comparing the collection timestamps. Whenever possible, we will record the throttling rates of the submissions, and we will also maintain updated *aggregate statistics* on all the data sets. Such measures of information quality will allow us to incorporate statistical techniques for handling the measurement errors[4] into our automated tools for classifying, filtering and mining the data and will enable researchers to draw meaningful conclusions from the experiments (challenge ℂ5).

[4] For example, the precision of estimation can be improved by combining results from multiple instruments, which are characterized by different measurement errors, and results that are likely to be imprecise can be discarded after performing a 3σ test. Such techniques are widely used in engineering disciplines [Chatfield 1983].

Proposed metrics. Several metrics are needed for evaluating the detection accuracy, scalability and responsiveness of systems benchmarked. The *receiver operating curve (ROC)* plots the true-positive detection rate of an algorithm against the rate of false-positive warnings. For data sets where the ground truth is available, a *confusion matrix* tabulates the attack instances, as classified by the algorithm under evaluation, against the true classes of those attacks, and it can provide deeper insights about the strengths and weaknesses of the algorithm. These metrics have been used in the past for comparing the performance of techniques for detecting masqueraders [Maxion and Townsend 2004].

The ability to create reference data sets of different sizes and to provision resources in the experimental platform enables a further investigation of the system scalability. The *scaleup* measures the system's ability to maintain a constant response time when solving increasingly larger problems only by adding a proportional amount of storage and computational resources—*i.e.*, if we double the resources, can we solve a problem twice as large? In contrast, the *speedup* indicates whether adding resources results in a corresponding decrease in the response time—*i.e.*, if we double the resources, can we solve the same problem twice as fast? Both these metrics were introduced for evaluating the scalability of parallel database systems [DeWitt 1993].

Finally, the characteristics of the *response-time distributions* are important for systems where the detection of threats is time sensitive. In these situations, reporting the mean response time is not sufficient, as many data-intensive systems are known to be scalable, but to exhibit heavy-tailed latency distributions [Zaharia et al. 2008]. The *high percentiles* of the latency distributions should also be reported and compared, such as the 95th and 99th percentiles that are commonly used in the industry to specify the guarantees provided in service-level agreements [Google Inc. 2011].

4. Discussion

The WINE data sets and the platform for repeatable experimentation provide the opportunity to ask a number of research questions. While a complete list of such questions is beyond the scope of this paper, we provide a few examples to guide the research agenda for exploring this space.

How to avoid vulnerabilities in computer programs? The introduction of security vulnerabilities during software evolution was studied by analyzing the revision logs and bug databases of large, production-quality codebases. For example, this approach pointed out how effective the software vendors are in dealing with zero-day attacks [Frei 2009], which vulnerabilities occur repeatedly as a result of software reuse [Pham et al. 2010] and the most common programming errors that lead to vulnerabilities [CWE/SANS 2010]. However, these findings do not discern the security vulnerabilities that are ultimately exploited and that help malware propagate in the wild, which emphasizes a fundamental shortcoming in our assessment of software quality. By correlating data from open-source software repositories with the information provided by WINE, we have the opportunity to gain a deeper understanding of security vulnerabilities. This will allow us to minimize the impact of vulnerabilities by focusing on the programming bugs that matter.

What are the sources of zero-day attacks? These attacks exploit vulnerabilities that are not acknowledged publicly, *e.g.*, while the software vendor is working on patching the vulnerability. We currently do not know if malware creators identify vulnerabilities predominantly through a form of fuzz testing [Miller et al. 1990] or from insider information. We could gain insight into the sources and prevalence of zero-day attacks by analyzing the binary-reputation data set and by correlating this information with events recorded in other system logs.

Is malware installed predominantly through exploits or through voluntary downloads? This question could be answered by analyzing the telemetry and the binary-reputation data sets and has important implications for understanding the dissemination mechanisms of malware and for validating the working assumptions of current intrusion-detection systems.

Does the large-scale dissemination of security patches make the world a safer place? Techniques for exploiting vulnerabilities automatically—by reverse engineering security patches—have been introduced recently [Brumley et al. 2008], but we lack empirical data about their impact in the real world. The telemetry data set can highlight, for instance, if fewer attacks are recorded immediately after the release of updates and, in general, can shed additional light on this aspect of the security arms race.

While these questions originate from a domain that we are familiar with, we believe that the WINE data is interesting from other perspectives as well (*e.g.*, for the economical sciences, storage systems, network performance analysis). By lowering the bar for validating advances in these fields, WINE will promote controversially innovative research, which introduces new ideas with the potential to change the community's perspective. For example, investigating the feasibility of patching unknown software vulnerabilities automatically, at run-time, currently requires laborious and expensive red-teaming experiments [Perkins et al. 2009]. However, these controversial questions are the ones most likely to lead to disruptive innovations in our field. WINE will allow such research projects to establish credibility through rigorous, quantitative validations using representative field data.

5. Related work

Camp et al. [2009] compile a "data wish list" for cybersecurity research and emphasize the need for representative field data in the research community. In addition to specific data that is currently unavailable—such as annotated network traces, URLs received in spam emails, representative malware samples—the authors identify the need for a data-sharing process that facilitates the collection of metadata and that addresses the privacy and legal concerns. In this paper, we propose such a process for the WINE data sets. WINE provides many of the items on the wish list, and it also includes unique data sets that were not foreseen by Camp et al. (*e.g.*, historical information on malicious executables extending before the threat identification).

Lippmann et al. [2000] describe the Lincoln Labs data set for benchmarking intrusion detection systems. The data set is synthesized from the statistical distributions observed in the network traffic from several Air Force bases. McHugh [2000] criticizes this work for the lack of information on the validation of test data—such as measures of similarity with the traffic traces or a rationale for concluding that similar behaviors should be expected when exposing the systems-under-test to real world data. McHugh observes that the experimenter has the burden of proof for showing that the artificial environment does not affect the outcome of the experiment. Maxion and Townsend [2004] emphasize the importance of careful experimental design for the ability to identify subtle flaws in the data. These lessons learned endure in the community: the PREDICT data repository [DHS 2011b] was also criticized for the lack of adequate metadata, and Camp et al. [2009] emphasize the need for metadata that allows experimenters to distinguish meaningful conclusions from artifacts. One of the major thrusts in our benchmarking effort is to ensure that all the metadata on experiments and on the data-collection process is included in WINE.

We draw inspiration from other research fields, where benchmarking is well established. For example, Paxson [2004] catalogs the metadata that must be recorded when measuring the performance of network protocols. Eide et al. [2007] report their observations from running Emulab. which underlies the DETER testbed for experimental cybersecurity [DHS 2011a], and emphasize the importance of automatically recording experimental processes for the ability to reproduce the results later. DeWitt [1993] presents the design of the Wisconsin Benchmark, which produced the seminal ideas in database benchmarking. In this paper, we identify the key differences between these approaches and security benchmarking, such as the need for representative field data and for frequently updating the reference data sets, and we propose mechanisms for addressing these challenges.

6. Summary

Through WINE, we aim to develop a benchmark that covers the entire lifecycle of security threats. WINE includes five data sets, providing access not only to malware samples, but also to the contextual information needed to understand how malware spreads and conceals its presence, how it gains access to different systems, what actions it performs once it is in control and how it is ultimately defeated. The unique features of these data sets allow us to address several research questions that are still outstanding, such as the prevalence and origins of zero-day attacks. Moreover, by correlating these data sets with information from additional sources, *e.g.* the revision logs and bug databases of open source software, we can follow the entire lifecycle of a security threat from the introduction of a vulnerability in a software component to the disappearance of the last exploit of that vulnerability. We will enable the reproducibility of results by archiving the reference data sets used in experiments, by including the metadata required for determining what each data set is representative of and by providing integrated tools for recording the hypotheses tested and the procedures employed in order to draw meaningful conclusions from experimental results. We believe that this new benchmarking approach will provide key insights for the fields of security, machine learning and software engineering.

Acknowledgments

We thank Roy Maxion, Dave Brumley, Jon McCune, Ahren Studer, Jim Newsome and Marc Dacier for their early feedback on the ideas presented in this paper.

References

BAKER, M. G., HARTMAN, J. H., KUPFER, M. D., SHIRRIFF, K. W., AND OUSTERHOUT, J. K. 1991. Measurements of a distributed file system. In *ACM Symposium on Operating Systems Principles*. Pacific Grove, CA, 198–212.

BRUMLEY, D., POOSANKAM, P., SONG, D. X., AND ZHENG, J. 2008. Automatic patch-based exploit generation is possible: Techniques and implications. In *IEEE Symposium on Security and Privacy*. Oakland, CA, 143–157.

CAMP, J., CRANOR, L., FEAMSTER, N., FEIGENBAUM, J., FORREST, S., KOTZ, D., LEE, W., LINCOLN, P., PAXSON, V., REITER, M., RIVEST, R., SANDERS, W., SAVAGE, S., SMITH, S., SPAFFORD, E., AND STOLFO, S. 2009. Data for cybersecurity research: Process and "wish list". `http://www.gtisc.gatech.edu/files_nsf10/data-wishlist.pdf`.

CHATFIELD, C. 1983. *Statistics for Technology: A Course in Applied Statistics*, 3rd ed. Chapman & Hall/CRC.

COVA, M., LEITA, C., THONNARD, O., KEROMYTIS, A. D., AND DACIER, M. 2010. An analysis of rogue AV campaigns. In *International Symposium on Recent Advances in Intrusion Detection*. Ottawa, Canada, 442–463.

CWE/SANS. 2010. Top 25 most dangerous programming errors.

DEAN, J. AND GHEMAWAT, S. 2004. MapReduce: Simplified data processing on large clusters. In *USENIX Symposium on Operating Systems Design and Implementation*. San Francisco, CA, 137–150.

DEWITT, D. J. 1993. The Wisconsin benchmark: Past, present, and future. In *The Benchmark Handbook for Database and Transaction Systems*, J. Gray, Ed. Morgan Kaufmann.

DHS. 2011a. DETER. `http://www.isi.deterlab.net/`.

DHS. 2011b. PREDICT. `http://www.predict.org/`.

EIDE, E., STOLLER, L., AND LEPREAU, J. 2007. An experimentation workbench for replayable networking research. In *USENIX Symposium on Networked Systems Design and Implementation*. Cambridge, MA.

FREI, S. 2009. Security econometrics: The dynamics of (in)security. Ph.D. thesis, ETH Zürich.

GOOGLE INC. 2011. Google Apps service level agreement. `http://www.google.com/apps/intl/en/terms/sla.html`.

GRIFFIN, K., SCHNEIDER, S., HU, X., AND CHIUEH, T.-C. 2009. Automatic generation of string signatures for malware detection. In *International Symposium on Recent Advances in Intrusion Detection*. Saint-Malo, France, 101–120.

KEETON, K., MEHRA, P., AND WILKES, J. 2009. Do you know your IQ? A research agenda for information quality in systems. *SIGMETRICS Performance Evaluation Review 37*, 26–31.

LEITA, C., BAYER, U., AND KIRDA, E. 2010. Exploiting diverse observation perspectives to get insights on the malware landscape. In *International Conference on Dependable Systems and Networks*. Chicago, IL, 393–402.

LIPPMANN, R. P., FRIED, D. J., GRAF, I., HAINES, J. W., KENDALL, K. R., MCCLUNG, D., WEBER, D., WEBSTER, S. E., WYSCHOGROD, D., CUNNINGHAM, R. K., AND ZISSMAN, M. A. 2000. Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation. *DARPA Information Survivability Conference and Exposition,*, 12–26.

MAXION, R. A. AND TOWNSEND, T. N. 2004. Masquerade detection augmented with error analysis. *IEEE Transactions on Reliability 53,* 1, 124–147.

MCHUGH, J. 2000. Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Transactions on Information and System Security 3,* 4, 262–294.

MILLER, B. P., FREDRIKSEN, L., AND SO, B. 1990. An empirical study of the reliability of UNIX utilities. *Communications of the ACM 33,* 12 (Dec), 32–44.

PAVLO, A., PAULSON, E., RASIN, A., ABADI, D. J., DEWITT, D. J., MADDEN, S., AND STONEBRAKER, M. 2009. A comparison of approaches to large-scale data analysis. In *ACM SIGMOD International Conference on Management of Data*. Providence, RI, 165–178.

PAXSON, V. 2004. Strategies for sound internet measurement. In *Internet Measurement Conference*. Taormina, Italy, 263–271.

PERKINS, J. H., KIM, S., LARSEN, S., AMARASINGHE, S., BACHRACH, J., CARBIN, M., PACHECO, C., SHERWOOD, F., SIDIROGLOU, S., SULLIVAN, G., WONG, W.-F., ZIBIN, Y., ERNST, M. D., AND RINARD, M. 2009. Automatically patching errors in deployed software. In *ACM Symposium on Operating Systems Principles*. Big Sky, Montana, USA, 87–102.

PHAM, N. H., NGUYEN, T. T., NGUYEN, H. A., AND NGUYEN, T. N. 2010. Detection of recurring software vulnerabilities. In *IEEE/ACM International Conference on Automated Software Engineering*. Antwerp, Belgium, 447–456.

ZAHARIA, M., KONWINSKI, A., JOSEPH, A. D., KATZ, R. H., AND STOICA, I. 2008. Improving MapReduce performance in heterogeneous environments. In *USENIX Symposium on Operating Systems Design and Implementation*. San Diego, CA, 29–42.

Legal Issues Associated With Data Collection & Sharing

Jody R. Westby, Esq.*
Global Cyber Risk LLC
5125 MacArthur Blvd, NW; Third Floor
Washington, DC 20016 USA
+ 1.202.255.2700
westby@globalcycberrisk.com

1. The Problem

Cyber security researchers require the use of various types of communications data for problem definition and testing purposes, but they often do not have access to such data, especially that which reflects current traffic patterns and threats. When researchers are able to obtain useful communications data, their organizations often restrict their use of it because the legal issues are complex and present significant risks to the organization and researcher. Thus, cyber security research and development (R&D) initiatives are hampered, the development of effective security solutions is thwarted or impeded, and some threats may not be tested.

The PREDICT[1] project, sponsored by the U.S. Department of Homeland Security (DHS) Science & Technology Directorate's Cyber Security R&D division, has tried to address this problem by making approved data sources available to cyber security researchers. The purpose of PREDICT is to help accelerate the advancement of network-based cyber defense research, product development, and evaluation by providing needed test datasets to the research community. PREDICT's operations include a comprehensive analysis of legal and policy issues associated with each dataset. This provides researchers and their organizations with more certainty that the datasets used in the R&D effort are clear of legal issues.

As cyber security R&D increases and the attacks become more complex, organizations are becoming more concerned about legal and policy considerations associated with R&D projects. Laws governing the interception, disclosure, and use of communications data are strict, yet confusing, and carry criminal penalties. Privacy laws are inconsistent, may apply to both packet headers and content, and present reputational risks as well as civil and/or criminal penalties. Finally, the complexity of research projects that investigate botnet operations spanning the globe raise a host of legal issues, and guidance for structuring projects around these legal landmines is scarce.

Recognizing that further work in this area was needed, DHS's Cyber Security R&D Division funded a project entitled "New Frameworks for Detecting and Minimizing Information Leakage in Anonymized Network Data." One of the goals of the project was to develop a tool that researchers and their organizations could use to help them analyze these legal and policy considerations and understand possible legal protective measures that could be utilized to better manage risks associated with the use of communications data in cyber security R&D. The *Legal & Policy Tool Chest for Cyber Security R&D* (*Tool Chest*) *was* developed by the author to meet this goal.

The *Tool Chest* is a comprehensive set of three documents that may be used both to help analyze the legal and policy implications associated with the use of traffic data in cyber security R&D and to mitigate identified risks:

1. Legal Analysis Tool on Obtaining & Using Network Communications Data (Legal Analysis Tool) focuses on obtaining, using, and disclosing intercepted and stored communications data.
2. Privacy Tool on Using Network Communications Data (Privacy Tool) focuses on the relevant privacy legal considerations with this data.
3. Protection Measures Tool contains sample contract clauses and memoranda of agreement that can be used by researchers and their organizations to mitigate legal risk.

The *Tool Chest* is not, however, intended to serve as a complete legal reference guide for cyber security R&D. Research activities pertaining to the detection and mitigation of botnets and other malware are often proactive and require some of the most complex legal analysis because research activities can involve numerous bodies of law, including foreign laws and treaties. To date, there are few resources available to assist cyber security researchers, institutional review boards (IRBs), attorneys, and funding organizations to help them determine whether a particular research

*Jody R. Westby is CEO of Global Cyber Risk LLC, located in Washington, DC. Ms. Westby also serves as Adjunct Distinguished Fellow to Carnegie Mellon CyLab. She chairs the American Bar Association's Privacy & Computer Crime Committee (Section of Science & Technology Law) and co-chairs the World Federation of Scientists' Permanent Monitoring Panel on Information Security. She is the author of the *Legal & Policy Tool Chest for Cyber Security R&D* and the *Legal Guide to Cyber Security Research on Botnets*. She has published four books on international issues pertaining to privacy, cybercrime, cyber security and enterprise security programs, as well as numerous articles and papers. She speaks globally on these topics.

[1] PREDICT is an acronym for the Protected Repository for the Defense of Infrastructure Against Cyber Threats. Jody R. Westby © 2011.

project could violate laws or organizational policies or the degree of risk that is involved.

The author developed the *Legal Guide on Cyber Security Research on Botnets (Botnet Legal Guide)* to extend the *Tool Chest's* analysis and examine the myriad of legal issues associated with this particular type of research. The *Botnet Legal Guide* also was funded by DHS's Cyber Security R&D Division and developed as a component of a technical research project led by Georgia Institute of Technology on "Countering Botnets: Anomaly-Based Detection, Comprehensive Analysis, and Efficient Mitigation."

The *Botnet Legal Guide* is intended to be used as a companion to the *Tool Chest.* Botnet research can invoke a number of laws beyond interception, stored communications, and privacy laws. The purpose of the *Guide* is to serve as a central resource and tool with respect to the range of legal and policy issues associated with cyber security research on botnets, but it will also be useful in many other cyber security R&D projects where similar activities are undertaken. In the development of the *Botnet Legal Guide,* nineteen case studies of botnet research projects were analyzed and specific research activities were identified. Laws that may be applicable to these research activities included cybercrime, intellectual property, child pornography, spam, breach notification, identity theft, access device and wire fraud, contract, and tort laws. The *Botnet Legal Guide* maps each R&D activity to the laws that may be applicable. Although it focuses on U.S. law, international legal issues also are discussed.

2. Fact Finding

Some initial fact-finding is a necessary precursor to the legal analysis process. It is important to determine:

- **Who is the provider of the data?**
  - ➢ A provider of communications services "to the public," such as Verizon, AT&T, AOL, Earthlink, etc.?
  - ➢ A private provider of service, such as a university Internet service provider, a non-profit organization, or a private sector business?
  - ➢ A government entity, such as an agency, department, or research entity?
- **Who owns the data?** Does the provider own the data or did they get it from someone else, either directly or indirectly? Many individuals or even organizations may use a particular dataset, but use does not necessarily mean the user owns the data or may allow others to use it.[2]
- **How was it obtained?** Was the data obtained from stored communications or by real-time interception? If it was intercepted, who intercepted it? A provider? Law enforcement? A third party? The researcher?
- **What are the data provider's privacy policies and operating procedures** regarding the collection, handling, storage, and retention of the data and disclosure of the data? What agreement does it have with its users?
- **Who is the researcher?** A student, private individual, private sector employee, or government employee?
- **Who is the organization sponsoring the research and who is the organization that the researcher works for?** A private company, university, national laboratory, non-profit organization, or government entity?
- **What is contained in the data?** Internet Protocol (IP) addresses? Full packet headers? Packet headers with Uniform Resource Locators (URLs)? URLs and/or content? Personally identifiable information (PII)? Sensitive content, such as medical or financial information or data pertaining to students or minors?

Any special characteristics of the data, such as the anonymization of certain fields, the jurisdiction in which it was collected, the age of the data, etc., should be noted. The *Tool Chest* provides a Decisional Framework Worksheet on which to enter the answers to the fact finding questions.

3. Dataset Legal Analysis

Any discussion of legal issues applicable to communications data requires a precise taxonomy. For purposes of this paper, it will be assumed that network communications data generally may consist of:

- **Packet headers**, which may contain IP addresses, port information, and the protocol used; and/or
- **Communications content**, comprised of:
  - ➢ the transmission control protocol (TCP) that transfers the actual content of the communication to the receiver;
  - ➢ the IP Authentication Header (AH) that is used for integrity and data origin authentication of IP packets; and
  - ➢ the actual content of the communication, which may include URLs, commonly referred to as links.

Hypertext Transfer Protocol (HTTP) packet headers may contain the requested URL, which carries an expectation of privacy and may be considered by some to be content. Therefore, for purposes of this paper, HTTP packet headers are distinguished from packets using other protocols (e.g., TCP, IP, User Datagram Protocol (UDP)) and are treated as content. This paper will refer to packet

[2] *See* Marianne Swanson, Joan Hash, and Pauline Bowen, *Guide for Developing Security Plans for Federal Information Systems,"* Feb. 2006 at 5, http://csrc.nist.gov/publications/nistpubs.

headers as "packet headers" or "traffic data" and communications content data (including HTTP headers) as "content."

Legal frameworks with respect to the interception of content and real-time collection of packet headers are usually complicated and carry stiff criminal penalties, including imprisonment and fines. Today, U.S. wiretap and stored communication laws are embodied within the Electronic Communications Privacy Act (ECPA), which governs not only the interception of content and the real-time collection of pen register and trap/trace (packet header) data, but also when such data may be disclosed to or used by third parties. The Stored Communications Act (SCA) has separate provisions governing the disclosure and use of stored communications data, such as that kept by Internet service providers (ISPs).

ECPA presents grave and complex legal risks to cyber security R&D. ECPA now governs:

- The possession, sale, transport, and installation of devices that can be used to intercept content;
- The interception of content (wiretapping) and the disclosure and use of intercepted content;[3]
- The installation and use of pen register and trap and trace devices for the real-time capture of non-content, such as packet headers;[4] and
- The access to and disclosure of stored communications information (content and packet headers) by communications providers "to the public," to governmental entities, and others.[5]

Thus, ECPA sets forth protections for stored communications *and* those in transit *and* the conditions under which intercepted and stored data can be accessed, used, and disclosed. ECPA also provides privacy protections to individuals by limiting what stored data "governmental entities" can obtain and the circumstances under which they can obtain it.

Separate laws protect certain customer data. The Communications Act of 1934 governs the use, disclosure, and access to customer proprietary network information (CPNI) data of regulated carriers, and the Telephone Records and Privacy Protection Act governs obtaining, transferring, receiving, or accessing confidential phone records information (CPRI).

Interception laws contain exceptions that allow providers to capture communications traffic for purposes related to the provisioning of service, protection of their property and the rights of their users, or to record the fact that a communication was completed. The use of intercepted communications, or the disclosure of it to others, however, is restricted and violations carry criminal penalties. There are additional considerations which may come into play, such as whether the researcher is associated with a "governmental entity" and whether the provider is one who services the general public ("provider to the public").

At the outset, it is important that researchers, IRBs, and legal counsel analyze whether the data contemplated for use in a research project:

- Was legally collected (by what entity, by what person, using what device, installed by whom, on what network);
- May be legally disclosed to researchers; and
- May be legally used by researchers.

Such analysis is necessary because, beyond the criminal penalties of imprisonment and/or substantial fines, several laws allow persons whose data was wrongfully intercepted, disclosed, or used to bring civil suits against the offender, including the U.S. Government. *Thus, a failure to properly analyze whether network data was legally obtained and whether it may be disclosed to and used by researchers may lead to embarrassment, tarnished reputations, loss of research funding, ruined careers, significant fines, and/or imprisonment.*

Legal analysis is complicated and requires a sequence of questions involving the information gathered in the fact gathering process:

1. Determine whether the data was collected (a) through the real-time interception of content or packet header data, or (b) at the end point of a communication and stored.
2. Determine whether the data involves content and/or packet header data. (Reminder: HTTP packet headers contain URLs and are treated as content).
3. Determine whether the data provider is a provider "to the public" or a private provider.
4. Determine (a) whether the researcher is an individual acting in his/her personal capacity or on behalf of a private organization, (b) whether the researcher is an employee or agent of a "governmental entity," and (c) whether the organization conducting the research is a "governmental entity."
5. Determine whether the data can be disclosed to and used by the researcher.

The *Tool Chest* provides worksheets, decisional flowcharts, definitions, and references to facilitate this analysis.

Once network communications data has been determined to have been legally obtained and may be legally disclosed to researchers and used for research and development (R&D) purposes, the next level of inquiry concerns privacy[6] considerations pertaining to

[3] 18 U.S.C. §§ 2510-22; http://www4.law.cornell.edu/uscode/18/usc_sec_18_00002510----000-.html.
[4] 18 U.S.C. §§ 3121-27, http://www4.law.cornell.edu/uscode/18/3121.html.
[5] 18 U.S.C. §§ 2701-12, http://www4.law.cornell.edu/uscode/18/2701.html.

[6] Europeans and some international audiences often use the term "data protection" instead of privacy. U.S. laws and reference materials generally use the term "privacy" with respect to data that

information in the dataset. Even though a dataset may have cleared the Legal Analysis process, further analysis with respect to privacy laws may reveal that the data cannot be used for R&D purposes – or that certain fields of the dataset will require special anonymization actions or elimination measures to comply with privacy protections or to mitigate risk.

There are several layers of inquiry in analyzing privacy issues in the context of communications data. These include:

- Laws and regulations;
- Legal instruments setting forth compliance obligations to protect the data, such as non-disclosure agreements, contract provisions, terms of service, administrative decisions or directives, etc.; and
- Privacy and other organizational policies, such as codes of conduct, policies governing the use of technology, and data retention and destruction.

The Privacy Analysis Tool in the *Tool Chest* is based upon U.S. laws and regulations, but basic differences between U.S. privacy laws and those in other jurisdictions are discussed. Globally, approximately 55 countries have privacy laws that may impact researchers using communications data. One legal consideration is whether a country's laws extend extraterritorially to a researcher. Due to the nature of packet switching technologies, it is virtually impossible to determine where all the data within a network communications dataset may have originated. Unless there is a clear overseas origination point for the data, such as a device sitting on a network in the Netherlands capturing data pertaining to traffic on that network, data may be viewed as subject to the laws of the country in which it was obtained. It is important to note that this issue has not been directly addressed and there is no clear determination or accepted principles to draw upon.

The Privacy Analysis Tool explains these legal and policy privacy considerations and provides a decisional framework to guide researchers and IRBs through the process of determining (1) whether a dataset has privacy issues associated with it, (2) whether these issues are fatal and may preclude the use of the data, and (3) whether certain privacy issues may be mitigated or eliminated through anonymization or other de-identification techniques.

International Legal Considerations

Every country has its own legal peculiarities, but multinational legal structures, such as the European Union's determination that Internet Protocol addresses are PII, significantly impact cyber security research and must be taken into consideration.

The United Nations (UN) has several international agreements that have been signed by all or most nations and form the basis of international law that can be extended to cyber security R&D. One of the most fundamental documents in international law is the *Universal Declaration of Human Rights*, which was adopted in 1948 and explicitly states that, "No one shall be subjected to arbitrary interference with his privacy, family, home, or correspondence."[7] The UN *International Covenant on Civil and Political Rights* also establishes a right to privacy.[8]

is afforded protection from disclosure. Since this paper is U.S.-centric, it will use the term privacy.

The Council of Europe (CoE), comprised of 47 member countries, laid the foundation in 1950 for Europe's legal framework regarding privacy, the processing of personal data, and cross-border data flows with its *Convention for the Protection of Human Rights and Fundamental Freedoms.*[9] The CoE's *Convention for the Protection of Individuals with Regard to Automatic Processing of Personal Data,* enacted in 1981, carried these concepts forward in an electronic environment.[10] The CoE's subsequent *Additional Protocol on Supervisory Authorities and Transborder Data Flows*, adopted in 2001, incorporated the concept of national supervisory authorities and restrictions on cross-border data flows to the Convention.[11]

European Union Directives

There are two aspects of EU law that are particularly troublesome to researchers: the Data Protection Directive, which prohibits cross-border data flows of information protected by the EU Directive unless the receiving jurisdiction offers equivalent privacy protections,[12] and the Article 29 Working Party's (advisory body to the European Commission) opinions regarding IP addresses constituting PII.[13] The EU Data Protection Directive has a broad definition of PII. It defines "personal data" as:

> [A]ny information relating to an identified or identifiable natural person ("data subject"); an identifiable person is one who can be identified, directly or indirectly, in particular by reference to an identification number or to one or more factors specific to his physical,

[7] *Universal Declaration of Human Rights*, *United Nations, General Assembly, Resolution 217 A (III),* Dec. 10, 1948, http://www.un.org/Overview/rights.html.
[8] *International Covenant on Civil and Political Rights,* Dec. 16, 1966, http://treaties.un.org/Pages/ViewDetails.aspx?src=TREATY&mtdsg_no=IV-4&chapter=4&lang=en.
[9] *CoE Convention for Protection of Human Rights*, http://conventions.coe.int/treaty/en/treaties/html/005.htm.
[10] *CoE Convention on Processing of Personal Data,* Articles 5-6, http://conventions.coe.int/Treaty/en/Treaties/Html/108.htm.
[11] *CoE Additional Protocol*, Articles 2-3.
[12] *Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data,* OFFICIAL JOURNAL L. 281/31, Nov. 23, 1995, http://ec.europa.eu/justice_home/fsj/privacy/law/index_en.htm#directive (hereinafter "EU Directive").
[13] *Opinion 4/2007 on the concept of personal data,* Article 29 Data Protection Working Party, 01248/07/EN WP 136, June 20, 2007 at 16-17, http://ec.europa.eu/justice_home/fsj/privacy/workinggroup/wpdocs/2007_en.htm (hereinafter WP 29 2007 Opinion).

physiological, mental, economic, cultural, or social identity.[14]

The Working Party reaffirmed its position that IP addresses are PII in a 2009 opinion:

> In this respect**, it re-emphasises its earlier Opinion [Opinion 4/2007] that unless the service provider** *"is in a position to distinguish with absolute certainty that the data correspond to users that cannot be identified, it will have to treat all IP information as personal data, to be on the safe side."*[15]

The position was also noted in a June 2010 Working Party opinion concerning online behavioral advertising.[16] The Article 29 Working Party has gone a step farther than contemplating IP addresses as PII. In the same 2007 opinion, the WP also discussed pseudonymised data, determining that:

> Retraceably pseudonymised data may be considered as information on individuals which are *indirectly identifiable.* Indeed, using a pseudonym means that it is possible to backtrack to the individual, so that the individual's identity can be discovered, but then only under predefined circumstances. In that case, although data protection rules apply, the risks at stake for the individuals with regard to the processing of such indirectly identifiable information will most often be low, so that the application of these rules will justifiably be more flexible than if information on directly identifiable individuals were processed.[17]

The EU's expansive definition of PII, coupled with the position that IP addresses are PII, presents a tough compliance issue for cyber security R&D and creates high barriers for international collaboration on cyber security R&D projects. The compliance risks on collaborative R&D projects needs to be carefully evaluated, lest the researcher and his/her organization become ensnarled in a dispute with a national data protection authority, or worse, create a diplomatic issue between countries.

4. Botnet Research Legal Analysis

Botnet research brings several additional bodies of law into play, such as those pertaining to the protection of communications equipment, cybercrime, child pornography, privacy, breach notification, identity theft, spam, intellectual property, access device and wire fraud, torts, and contracts. For example, botnet research may involve various data elements, including the content of communications, PII, and sensitive information obtained through the compromised computer or victim. Researchers who allow botnets to run over live networks, especially if they have infiltrated them and are involved in command and control functions, may be deemed to be aiding and abetting activities that violate cybercrime laws or wilfully causing these acts to be done. Researchers who infiltrate a botnet and passively observe spam-related commands could be viewed by some enforcement authorities as aiding and abetting a crime. If the researchers change a link in a botnet's spam message to one under the researchers' control in the belief they are reducing harm, they are actually bringing more risk upon themselves and their organizations because they are actively involved in perpetrating an online fraud, directing a spam operation, and sending commercial email messages directed toward a site they control. Researchers who establish websites that mimic those used by botnets may be infringing legitimate copyrights or removing or altering copyright management material, which could lead to suits by the legitimate owners of the work. These are but a few of the examples that the *Botnet Legal Guide* analyzes based upon research activities undertaken in the case studies.

Since botnet legal analysis cannot be neatly broken into a set of questions and guided by decisional flowcharts, the *Botnet Legal Guide* sets forth a number of tables and charts to facilitate the process. One table lists the various research activities undertaken in the case studies, indicates the laws that may be triggered, and notes actions that the researcher may take to mitigate risks. Another tables lists penalties associated with each of the laws. A Botnet Legal Research Template sets forth the key provisions of the laws and notes how the *Tool Chest* or *Botnet Legal Guide* may assist.

International Considerations

To complicate matters further, the laws of more than one jurisdiction may need to be considered, depending upon where the research is being performed and/or the jurisdictions that may be impacted. For example, botnet and malware research projects may involve communications sent by compromised computers around the globe, with drop zones located in multiple jurisdictions, domain names registered in several countries, and botmasters controlling the operations from more than one location.

Given the global nature of botnet research, it is important to note that researchers based outside the United States may be subject to criminal penalties for violating domestic U.S. laws. Likewise, researchers in the U.S. might be subject to the extraterritorial reach of foreign laws. In the U.S., laws are generally applicable only within the United States, however Congress has the authority to enforce its laws beyond the territorial boundaries of the United State. For example, in 2001, as part of the USA PATRIOT Act, Congress amended the U.S. cybercrime law, the Computer Fraud and Abuse Act, to apply to a computer "which is used in interstate or foreign commerce, including a computer located outside the United States that is used in a manner that affects interstate or

[14] EU Directive.
[15] Article 29 Working Party, Opinion 1/2009 on the proposals amending Directive 2002/58/EC on privacy and electronic communications (e-Privacy Directive), 00350/09/EN WP 159, Feb. 10, 2009, http://ec.europa.eu/justice/policies/privacy/workinggroup/wpdocs/2009_en.htm (emphasis in original).
[16] Article 29 Working Party, Opinion 2/2010 on online behavioural advertising, 00909/10/EN WP 171, June 22, 2010, http://ec.europa.eu/justice/policies/privacy/docs/wpdocs/2010/wp171_en.pdf.
[17] WP 29 2007 Opinion at 18.

foreign commerce or communication of the United States."[18] Thus, researchers who are conducting activities that may impact computers in another country may find themselves (a) the target of legal action or extradition by a foreign government for their actions that impacted the country's population, or (b) the target of a U.S. legal action for affecting a foreign computer involved in U.S. commerce.

Without a specific statutory reach, the general rule for applying laws extraterritorially revolves around whether someone had the requisite minimum connection with persons in the other jurisdiction. For example, a website operator may be based outside the United States, but his activities may be said to fall within the U.S. if the website is accessed or intended to be accessed by a single person within the United States. The U.S. Supreme Court set forth this long-standing principle in *International Shoe Co. v. Washington* with its declaration that there must be "certain minimum contacts" with the jurisdiction so it does not offend "traditional notions of fair play and substantial justice."[19] Intention may also be a factor.[20]

5. Relationship of Legal Analysis to Ethical Considerations

With little to guide them regarding the legal issues pertaining to botnet R&D, researchers have increasingly looked to whether their research was "ethical" or within accepted "principles" to determine whether specific activities would be acceptable conduct. Ethical determinations are frequently based upon whether (1) the benefits of the research outweigh any potential harms that may occur, or (2) the research activity is "doing no harm" (e.g., the activity would have occurred anyway by the bot). The problem with this analysis is that activities viewed as "beneficial" or "not harmful" are also assumed to be legal. Unfortunately, the laws are not administered and enforced through such a simple prism.

Many activities that are considered by researchers to be ethical are, in fact, illegal. For example, a researcher may justify infiltrating a botnet and allowing it to send spam because the spam would have been sent anyway, so no harm was done. Or, the researcher may justify changing links in the botnet's spam message to an innocuous site that he/she controls because it reduces harm to the person who otherwise would have received a malicious spam message. This reasoning seems logical, but it ignores the fact that sending the spam, especially with the researcher's involvement from its own system, violates anti-spamming laws and raises a number of other legal issues.

The intent here is not to point fingers or blame researchers for conducting illegal research; until now, guidance has been limited. That said, the lack of legal consideration given most botnet research projects is deeply concerning and indicates a reluctance on the part of researchers to seek out competent legal assistance. This concern is compounded when researchers include questionable activities in their research simply because another research team has undertaken similar activities and concluded they were all right. In addition, these decisions seem to be made with complete disregard for jurisdictional differences in legal frameworks, such as between the U.S. and EU, even though researchers are analyzing botnets that span the globe and have hosts, drop zones, and victims scattered across various countries.

It is important to understand the connection between "ethical" and legal research. Generally, conduct that is illegal is not viewed as ethical. Numerous corporate codes of conduct prohibit conduct that is unlawful or inconsistent with their compliance requirements. *Therefore, it is important that researchers first undertake a legal analysis of their project and, after ensuring that the research activities are within the bounds of the law, then proceed to examine ethical considerations.*

6. Conclusion

The *Tool Chest* and *Botnet Legal Guide* are companion publications that provide the cyber security research community with a central repository of definitions, descriptions of the laws, worksheets, decisional frameworks, tables simplifying privacy provisions and penalties, and conclusions regarding how U.S. laws apply to datasets to be used in research projects and impact research activities. International considerations, especially with respect to privacy and cybercrime laws, present challenges for researchers that require careful analysis. The *Tool Chest* and *Botnet Legal Guide* offer a positive step toward helping researchers, IRBs, legal counsel and management better understand the legal issues associated with research projects and the data used in them. The need for collaboration between the legal and technical communities is great, particularly with respect to exploring the extraterritorial reach of laws and inconsistencies in legal frameworks. Researchers particularly need to better understand critical jurisdictional differences in the global legal framework for interception, privacy, and cybercrime. Programs such as PREDICT that include the legal analysis of datasets that are offered to researchers help build confidence that data used in research efforts will not run afoul of the law, but they do not address the legality of the activities undertaken by researchers when using the data. The development of best practices with respect to certain research activities would make a significant difference toward encouraging legal conduct in R&D projects.

[18] *See* 18 U.S.C. § 1030(e)(2)(B). Even prior to the 2001 amendment, however, at least one court held that the plain language of 18 U.S.C. § 1030 was a clear manifestation of congressional intent to apply that section extraterritorially. See *United States v. Ivanov*, 175 F.Supp.2d 367, 374-75 (D. Conn. 2001).

[19] *International Shoe Co. v. State of Washington*, 326 U.S. 310 (1945).

[20] "The intent to cause effects within the United States ... makes it reasonable to apply to persons outside United States territory a statute which is not extraterritorial in scope." *United States v. Muench*, 694 F.2d 28, 33 (2d Cir. 1982).

An Architectural Solution for Data Exchange in Cooperative Network Security Research

Brian Trammell
ETH Zurich
Zurich, Switzerland
trammell@tik.ee.ethz.ch

Jan Seedorf
NEC Laboratories Europe
Heidelberg, Germany
seedorf@neclab.eu

Giuseppe Bianchi
Uni. Roma Tor Vergata
Rome, Italy
giuseppe.bianchi@uniroma2.it

1. INTRODUCTION

Science can be seen as a cycle of hypothesis, collection of experimental results, and analysis to refine or refute the original hypothesis. The desire to increase the rigor of large-scale computer and network security studies extends to all three of these steps: an improved understanding of the situation leads to better hypotheses, improved collection and sharing of data increases the scope of the studies that can be done, and improved analysis techniques lead to deeper insights and more useful results.

However, seeing this cycle as a set of discrete steps has disadvantages in the real world, especially when it comes to the collection step. Most computer security research requires data which have a significant impact on the privacy of the users of the studied systems. Protection of privacy in such studies is a matter of complying with legal obligations to protect the rights of individuals in such studies [1, 2]. Enhanced collection and centralization would indeed seem to violate the principle in European data protection legislation that "only the kind and amount of data that are functional and necessary to the specific processing purpose that is pursued" should be collected.

The more useful a data set is, the more detail it tends to have; and the more detail it has, the larger a privacy threat it represents. It is very difficult to separate utility for researchers with a legitimate use for a data set from utility for miscreants. Anonymization techniques can help here but must be used with care; for example, in the area of network traffic traces, it has been shown that anonymization on most useful data sets can be compromised by the asymmetric ease of traffic injection [3]. Other side channels may exist for similar situations. This makes data sharing as a basis for collaboration risky.

The EU FP7 integrated research project DEMONS [1] proposes an architectural solution to this problem. Instead of the centralization and dissemination of access to large-scale data sets, analysis code can be distributed to local data sets, which remain in the control of their original owners. Access control at each of these local repositories is applied to the incoming code (to ensure it is safe to run) as well as to the outgoing data (to ensure its identifiability is below an acceptable threshold). To further reduce the release of privacy-sensitive results in multi-domain scenarios, we apply secure multiparty computation to generate low-risk aggregates from multiple high-risk single-domain results. This paper gives some background on this architectural proposal, and applications to the security research use case.

[1]This work was partially supported by DEMONS, a research project supported by the European Commission under its 7th Framework Program (contract no. 257315). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the DEMONS project or the European Commission.

2. FOUNDATIONS AND ARCHITECTURE

Our proposal is inspired by previous work in the area of programmable measurement. SC2D [7] is an exploration of the properties of such "code mobility" architectures, envisioning a modular architecture built around a standard data model. In SC2D, the smallest unit of processing is a module, and SC2D programs are built by chaining modules together. Module security is handled out of band, via code signing. Trol [6] specifically adds the concept of privacy protection to this architectural reversal. Here, instead of modules, it provides a declarative query language similar to SQL, tailored to network measurement. This restricted language ensures it is possible to measure the privacy risk on data returned from the query, and to prevent the release of data with too much identifiable information, assuming a secure implementation of the interpreter.

This last is an important point: when considering code mobility, the security of the implementation is much more important than in traditional centralize-and-process architectures. Here, implementation faults can lead not only to data disclosure but to complete compromise of the hosts providing the analysis services, which could have impacts on the hosting organization beyond the compromised data sharing application. Scriptroute[8], applied to active network measurement by untrusted parties, illustrates some design choices which can minimize security risk in running potentially untrusted code in a restricted interpreter. It provides a language in which a small set of safe, high-level primitives can be combined, followed by a sandbox built around the execution environment to independently place limits on traffic sent.

In DEMONS, we apply these concepts to a distributed measurement system which allows dynamic composition of *blocks*, by chaining them together by their *gates*, or well defined interfaces, on a network of processing nodes. Processing nodes with packet capturing hardware or other raw data sources take the place of probes in traditional monitoring infrastructures. The functionality of the blocks covers a variety of granularities, from simple and generic primitives (e.g. "count elements") to whole algorithms (e.g. "find DNS servers which are used in botnet control based on this set of reply packets"). New blocks can be dynamically added to nodes, as well, though we rely on access control, trusted peers and signed code to secure the implementation of the blocks. Access control with awareness of the semantics of each of the blocks is also applied to compositions before they are sent to the nodes, to evaluate the risk that a given composition would result in too much data being exported for a given identity, role, and purpose.

Code mobility can be applied to aggressive data reduction. Moving analysis closer to the "edge" at which data is initially collected tends to reduce both the total resource demand as well as the privacy risk of a given analysis. Computational, storage, and bandwidth demand is reduced by throwing away irrelevant data as soon as possible, and data which is not discarded cannot be used to infer identifying information.

In some cases, however, code mobility is not enough. Many common large-scale queries in network or security measurement deal with aggregates across multiple administrative domains. Here, the aggregate is not privacy-sensitive, but the intermediate results from each domain are. Here we can apply secure multiparty computation (MPC), which has recently emerged as a viable, scalable approach to secure sharing of computing tasks [4, 5]. In MPC as implemented by the SEPIA framework used by DEMONS, *input peers* at each domain compute shared secrets, which represent the source data from a given domain without allowing recovery of the original data. These are then processed together with secrets from other domains by *privacy peers* to produce an aggregated result. The protocols presently supported by SEPIA include privacy-preserving aggregate counts, set union and intersection operations, and top-N lists.

3. APPLICATION TO LARGE-SCALE SECURITY DATA SHARING

While the main focus of the DEMONS project is the development of a network monitoring environment for operational use, we also intend to develop a platform suitable to collect large-scale data for research applications: The architecture illustrates several principles, and we intend to apply the flexibility of the primitives provided on the DEMONS nodes to collaborative security measurement research problems.

The principles advanced within DEMONS we see as applicable to the this application area are as follows:

- **Mobile analysis of immobile data**: Composition of analysis primitives allows safe execution of external code, which in turn allows data to be secured in a single location, reducing the risks associated with distribution of raw data sets.
- **Aggressive data reduction**: At each layer in a given analysis, data no longer necessary for the analysis can be aggregated down or discarded, benefiting both privacy and scalability.
- **Cryptographically-protected interdomain data sharing**: By applying MPC to appropriate intermediate results per-domain, interdomain aggregation is possible without requiring raw data set distribution.

4. CONCLUSION

Our proposed architecture differs somewhat from the customary workflow on large shared data sets. However, most any study that can be done in the traditional way can be translated to the model proposed by DEMONS. For example, sequential experiments on a given data set used to verify or compare two given approaches to a given analysis could instead be run simultaneously on a data stream, using the same nodes for observation but different nodes for computation and aggregation. In any consideration of the best methods for advancing the science of network and host security data analysis through collaboration, "how can we improve data collection and sharing" is not necessarily the only question to be answered – architectural approaches such as those advanced by DEMONS are important to consider as well.

5. REFERENCES

[1] Directive 2002/58/EC of 12 July 2002, concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communication), O.J. L 201/37, 31 July 2002.

[2] Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data; O. J. L. 281, 23 November 1995.

[3] Burkhart, M., Schatzmann, D., Trammell, B., Boschi, E., and Plattner, B., "The Role of Network Trace Anonymisation under Attack", in *ACM Computer Communications Review*, 40(1) pp. 6–11, January 2010.

[4] Burkhart, M., Strasser, M., Many, D., and Dimitropoulos, X., "SEPIA: Privacy-Preserving Aggregation of Multi0Domain Network Events and Statistics" in *Proceedings of the 19th USENIX Security Symposium*, Washington, DC, August 2010.

[5] Duan, Y., Canny, J., Zhan, J., "P4P: Practical Large-Scale Privacy-Preserving Distributed Computation Robust against Malicious Users" in *Proceedings of the 19th USENIX Security Symposium*, Washington, DC, August 2010.

[6] Mirkovic, J., "Privacy-safe network trace sharing via secure queries", in *NDA '08: Proceedings of the 1st ACM workshop on Network data anonymization*, ACM, pp. 3–10.

[7] Mogul, J. C., and Arlitt, M. "SC2D: An Alternative to Trace Anonymization", in *MineNet '06: Proceedings of the 2006 SIGCOMM workshop on mining network data*, ACM, pp. 323–328.

[8] Spring, N., Wetherall, D., and Anderson, T. "Scriptroute: A public internet measurement facility.", in *USITS'03: 4th USENIX Symposium on Internet Technologies and Systems*, pp. 225–238.

PREDICT: A TRUSTED FRAMEWORK FOR SHARING DATA FOR CYBER SECURITY RESEARCH

Charlotte Scheper
RTI International
3040 Cornwallis Road
Durham, NC 37705
1-919-485-5587
cscheper@rti.org

Susanna Cantor
RTI International
3040 Cornwallis Road
Durham, NC 37705
1-919-541-7323
scantor@rti.org

Dr. Douglas Maughan
DHS Science & Technology Directorate
Washington, DC
dmaughan@dhs.gov

ABSTRACT

In this paper, we describe the formatting guidelines for ACM SIG The Protected Repository for Defense of Infrastructure against Cyber Threats (PREDICT) has established a trusted framework for sharing real-world security-related datasets for cyber security research. In establishing PREDICT, a set of key issues for sharing these data has been addressed: providing secure, centralized access to multiple sources of data; assuring confidentiality to protect the privacy of the individuals and the security of the networks from which the data are collected; assuring data integrity to protect access to the data and ensure its proper use; and protecting proprietary information and reducing legal risks. PREDICT continues to address issues in producing and sharing datasets as it enters its second phase of development, providing more controversial data, adding data providers, and initiating international participation.

Categories and Subject Descriptors

H.2.7 [**Database Administration**]: Data Warehouse and Repository.

H.3.4 [**Systems and Software]:** Distributed Systems, Information Networks.

General Terms

Management, Legal Aspects. Standardization.

Keywords

Distributed Repository, Cyber Security, Internet, PREDICT.

1. INTRODUCTION

Defensive cyber security technologies have to be improved to address the rapidly changing cyber security threat landscape. However, researchers have insufficient access to data to test their research prototypes and technology decision-makers have no data to evaluate competing products. The White House Cyberspace Policy Review [1] action plan called for providing data to the research community to use to develop tools, test theories, and identify workable solutions for cyber security. The Protected Repository for Defense of Infrastructure against Cyber Threats (PREDICT) is directly addressing this call to action.

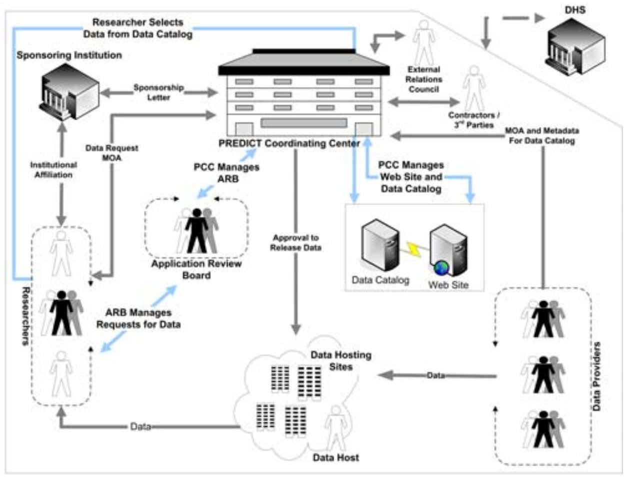

Researcher Selects Data from Data Catalog
Sponsoring Institution
Sponsorship Letter
DHS
External Relations Council
Contractors / 3rd Parties
MOA and Metadata For Data Catalog
PREDICT Coordinating Center
Data Request MOA
Institutional Affiliation
PCC Manages ARB
PCC Manages Web Site and Data Catalog
Application Review Board
Approval to Release Data
Data Catalog
Web Site
Researchers
ARB Manages Requests for Data
Data Hosting Sites
Data
Data Providers
Data
Data Host

Figure 1 PREDICT Repository Framework

To provide security-related datasets, PREDICT has had to address a set of key issues: providing secure, centralized access to multiple sources of data; assuring confidentiality to protect the privacy of the individuals and the security of the networks from which the data are collected; assuring data integrity to protect access to the data and ensure its proper use; and protecting proprietary information and reducing legal risks. To address these issues, PREDICT followed a three-pronged approach: develop a framework based on the data sharing models used in other domains that share sensitive data; conduct a full review of the legal context in which data collection and sharing occur; and reach out to the privacy community.

2. PREDICT REPOSITORY FRAMEWORK

Following the model of multi-site research networks [2], PREDICT is a distributed repository where multiple data providers collect and prepare data for sharing, multiple data hosts provide computing infrastructure to store the datasets and provide mechanisms to access them, and a central coordinating center (the PCC) provides a unified view of and portal [3] into the repository collection and manages the repository processes for accepting datasets and authorizing access. The PREDICT process includes sensitivity assessments of datasets to determine conditions of use; Memoranda of Agreement between the PCC and the providers,

Badgers'11, 10-APR-2011, Salzburg, Austria Copyright 2011 ACM /11/04…$10.00.

hosts, and researchers containing legally binding terms and conditions for providing and accessing data; and expert review of data requests. As shown in the framework illustration in Figure 1, the PCC provides information (metadata) about the data collected by the Data Providers, Data Providers work with Data Hosts to store data, approved Researchers browse for datasets of interest and apply for access through the PCC, and once their dataset requests are approved by the PCC, Researchers work directly with the Data Hosts to obtain the datasets.

In determining whether a dataset is suitable for inclusion in the repository, the following factors are considered: Who is the provider of the data? Who owns the data? How was the data obtained (i.e., was it intercepted or is it stored data?) What are the Data Provider's privacy policies and operating procedures? What is contained in the data? The answers to these questions determine the legal risks and impact the conditions of use. In accepting a user into the PREDICT community and granting access to data, the following factors are considered: Who is the Researcher and does he/she have a legitimate cyber security research role? What organization is the researcher affiliated with and will that organization sponsor the researcher? Are the requested datasets suitable for the proposed research?

3. PREDICT LEGAL PROCESS

A thorough review of applicable laws and regulations, both federal and state, was conducted in setting up PREDICT. As part of the process of approving datasets and data requests, a legal consultant reviews the policies and procedures and other available documents from providers, identifies legal relationships and agreements needed between PREDICT participants, prepares a risk chart for every dataset that identifies high risk data fields and/or datasets and establishes requirements for high risk fields, and works with the participants and the PCC to prepare Memoranda of Agreement (MOAs), which are legally binding within U.S. jurisdiction. While necessary to protect privacy rights and reduce the legal risk to data providers and researchers, the MOA process can be a hurdle for many researchers. Increasing researchers' understanding of the legal risks involved, planned revisions to the MOAs, and the upcoming provision of less readily available datasets will increase the value of the return on the effort required.

3.1 Privacy and Legal Outreach

During the design of the framework, the PREDICT program conducted a number of outreach activities to the legal and privacy communities. Privacy advocates, including the ACLU, the Electronic Frontier Foundation (EFF), and the Center for Democracy and Technology (CDT), were briefed and their input obtained. Working with the DHS Privacy Office, a Privacy Impact Assessment (PIA) [4] was prepared and government officials, including the DHS S&T General Counsel, the DHS General Counsel, and the Department of Justice, were briefed. This outreach successfully allayed concerns and identified key issues that had to be addressed.

3.2 2010 and Beyond

PREDICT currently houses 140 datasets from five data providers. The types of data include BGP Routing Data, Blackhole Address Space Data, Internet Topology Data, IP Packet Headers, Traffic Flow Data, and VOIP Measurement Data. Collection periods for the datasets vary from hours to days to months; the size of the datasets, from Bytes to Terabytes. In 2010, researchers from 24 academic, 1 government, and 31 private sector organizations joined the PREDICT community, resulting in a total community of 65 academic, 12 government, and 48 private sector organizations. PREDICT continues to address issues in producing and sharing datasets, developing a draft report on guidelines for ethical principles in networking and security research similar to the Belmont Report for human subject research, and holding workshops on disclosure control. In Phase II, currently scheduled to being operation in April 2011, PREDICT will expand datasets to include more controversial data such as unsolicited bulk email, DNS data, web logs, infrastructure data, and IDS and firewall data. New data providers will be added and international participation will be piloted through affiliation with research centers that will be responsible for vetting their researchers.

In summary, PREDICT is addressing an acknowledged need by providing large-scale, real-world security-related datasets for cyber security research. Significant policy and legal issues exist in collecting and sharing security-related data: many of these have been addressed by PREDICT but many still remain to provide usable data across the entire spectrum of information security R&D activities.

4. AUTHORS

The following authors represent RTI International, 3040 Cornwallis Road, Durham, NC 37705: Charlotte Scheper, Susanna Cantor, Renee Karlsen, Sandhya Bikmal, Roger Osborn, Gary Franceschini, Craig Hollingsworth, Al-Nisa Berry,

The following author represents DHS Science & Technology (S&T) Directorate: Dr. Douglas Maughan.

5. ACKNOWLEDGEMENT

PREDICT is funded by The Department of Homeland Security (DHS) Science and Technology (S&T) Directorate under contract number NBCHC070131.

6. REFERENCES

[1] Cyberspace Policy Review: Assuring a Trusted and Resilient Information and Communications Infrastructure. May, 2009. http://www.whitehouse.gov/assets/documents/Cyberspace_Policy_Review_final.pdf.

[2] Scheper, C. O., Cantor, S., & Karlsen, R. (2009, March). Trusted distributed repository of internet usage data for use in cyber security research. Proceedings of the Cybersecurity Applications and Technologies Conference for Homeland Security (CATCH).

[3] PREDICT Portal. https://www.predict.org.

[4] DHS Privacy Impact Assessment. February, 2008. http://www.dhs.gov/xlibrary/assets/privacy/privacy_pia_st_predict.pdf

A social-engineering-centric data collection initiative to study phishing

Federico Maggi
Dip. di Elettronica e Informazione
Politecnico di Milano, Italy
fmaggi@elet.polimi.it

Alessandro Sisto
Dip. di Elettronica e Informazione
Politecnico di Milano, Italy
alessandro.sst@gmail.com

Stefano Zanero
Dip. di Elettronica e Informazione
Politecnico di Milano, Italy
zanero@elet.polimi.it

ABSTRACT

Phishers nowadays rely on a variety of channels, ranging from old-fashioned emails to instant messages, social networks, and the phone system (with both calls and text messages), with the goal of reaching more victims. As a consequence, modern phishing became a multi-faceted, even more pervasive threat that is inherently more difficult to study than traditional, email-based phishing.

This short paper describes the status of a data collection system we are developing to capture different aspects of phishing campaigns, with a particular focus on the emerging use of the voice channel. The general approach is to record inbound calls received on decoy phone lines, place outbound calls to the same caller identifiers (when available) and also to telephone numbers obtained from different sources. Specifically, our system analyzes instant messages (e.g., automated social engineering attempts) and suspicious emails (e.g., spam, phishing), and extracts telephone numbers, URLs and popular words from the content. In addition, users can voluntarily submit voice phishing (vishing) attempts through a public website. Extracted telephone numbers, URLs and popular words will be correlated to recognize campaigns by means of cross-channel relationships between messages.

1 Introduction

Modern cyber criminals are widely recognized to be well-organized and profit-driven, as opposed to the reputation-driven underground which was prevalent years ago [2]. As a part of their arsenal, the miscreants have learned to streamline their campaigns also by leveraging automated social engineering attacks over several channels including emails, instant messaging, social networks [3], and the phone system (with both calls and text messages), with the common goal of expanding their "business" beyond email users. However, traditional *a-lá-Mitnick* scams are based on pure social engineering techniques and, despite their effectiveness, they are relatively slow. To make this a viable business, modern scammers have begun to take advantage of the customers' familiarity with "new technologies" such as Internet-based telephony, text-messages [4], and automated telephone services. Another example is the use of instant messaging (e.g., Windows Live Messenger, Skype, the FaceBook chat), which involves some form of conversation with computer programs that leverages natural language processing and artificial intelligence techniques to mimic a real person [6].

A particular variant of phishing, known as *vishing* (i.e., voice phishing), was popular in the U.S. in 2006–2009 [5], and is now slowly gaining ground in Europe. Notably, an experiment conducted in 2010 by the United Nations Interregional Crime and Justice Research Institute revealed that the 25.9% of Italians (on a sample comprising 800 randomly-selected citizens) were successfully tricked by phone scammers. In a previous work [7] we analyzed this type of scams, based on a selection of about 400 user-submitted reports, including the caller identifier (e.g., source phone number), (parts of) the transcribed conversation, general subject of the conversation, and spoken language. Besides confirming that vishing was popular in the U.S. at that time, our experience suggests that phishers rely on automated responders, and not only on live calls, with the goal of reaching a broader spectrum of victims. Reports were filed between 2009 and 2010 through a publicly-available web site where anyone can submit anonymous reports of vishing.

The system described in [7] focuses solely on vishing and, in addition, it has two main limitations. First, we trust submitters and, second, the effectiveness of vishing attacks could not be determined (evidently, people reporting suspicious calls are less prone to falling prey to them). To overcome these limitations, we propose to correlate the evidence on vishing scams with other forms of phishing. To this end, the new approach is to collect suspicious emails from spam-traps, instant messages from dedicated honeypots (e.g., based on myMSNhoneypot [1]) and content published by spammers on social networks (leveraging the @spamdetector service [9]). Our approach is content-driven. In particular, the first goal is to thoroughly quantify the popularity of voice-based scams. Secondly, we want to understand whether there are relationships between voice-based campaigns and text-based campaigns. Third, we strive to recognize evidence that suggest the use of social engineering techniques.

2 System overview

Our system has four modules, each tackling a different aspect of phishing. The *phone* module is an automated phone bot that places outbound calls, receives inbound ones, and records resulting conversations. The *email* module is a spam bot that receives spam and phishing email messages, and *IM* module is an instant messaging honeypot that collects unsolicited chat messages. The *social network* module will be implemented as a web crawler that to monitor suspicious accounts, known for sending spam (according to @spamdetector).

2.1 Text processing and correlation

The collected corpus (e.g., body of email messages, transcribed phone conversations, instant messages) is stored and analyzed using simple natural language processing techniques to extract popular sentences and words. Specifically, the stemming algorithm described in [8] is first applied to reduce words to stems. Secondly, stop words such as "the", "an", "this" are removed.

Regular expressions are then used to extract (possibly new) phone numbers and URLs. The former, core part of our approach, are sent to the phone module, while the latter will be shared for external analysis. Numbers, URLs and popular stems are used as a preliminary set of *features* to correlate messages across channels

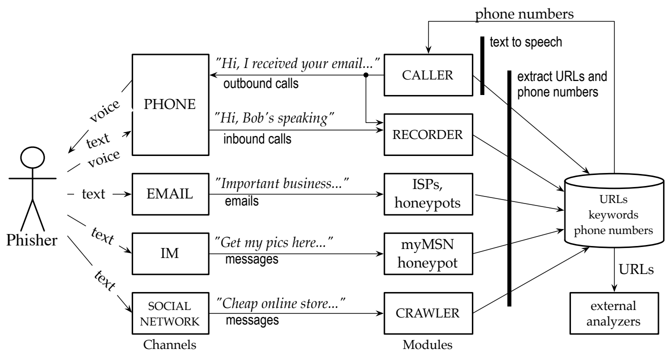


Figure 1: Overview of the dataflow of our collection system.

and find groups of different campaigns. Since shortened URLs are often used to evade filters (or simply to trick users), these are first resolved with the `long-shore.com` API, a service that mimics a real browser and records the redirection chain from a short URL to the target URL. Instead of the URL itself, the whole chain is retained and used as a similarity feature: it is indeed common for spammers to use multiple redirections to the same phishing site, to increase the lifespan of their campaigns.

2.2 Phone channel

The core of our collection system is divided into two sub-modules, both based on Asterisk. The *caller* sub-module periodically calls a feed of numbers. Whenever someone answers, a pre-recorded prompt mimics a hypothetical victim, supposedly tricked by the reverse vishing scam (e.g., *"Hi, this is Bob, I received your email and I am curious to know more about it"*) and waits for 30 seconds. The resulting audio is recorded along with simple metadata such as date, time, and number. The *recorder* module is leveraged to answer inbound calls on a series of decoy numbers that we plan to make available deliberately on social network profiles, blog posts and forums.

Audio recorded from both inbound and outbound calls is retained in a database, and is transcribed using the Sphinx speech-to-text engine. The resulting text, if any, is then processed as described above.

2.3 Email channel

This module is implemented as a distributed client, meant to be deployed at ISPs and other institutions (e.g., universities and research centers). The client analyzes spam databases and collects emails that are likely to contain a phone number. At the moment, attachments that may contain scanned documents (used by scammers that attempt to evade basic filters) are not considered. Found messages are sent back to a bot, publicly reachable via SMTP at `bot@phonephishing.info`. Contributors are invited to submit suspicious emails directly to this address.

2.4 Instant messaging channel

This module is implemented as a set of instant messaging accounts (i.e., Yahoo! Messenger, Windows Live Messenger and Google Talk), all registered on myMSNhoneypot, a honeypot that monitors such accounts for any activity. Since the accounts all have empty buddy lists, any message or friendship request received on those accounts is considered as malicious. Only instant messages that contain phone numbers are retained.

3 Collected data

As of February 2011, the email module has been working for 2 months, and the phone module is ready for deployment. To bootstrap the system, we gathered data from the email module and from `phonephishing.info`. We selected 551 vishing reports out from about a thousand of reports submitted by users in the first two years of activity. Discarded reports are mostly about telemarketing calls. This may appear a limited amount of data, but it must be considered that people typically do not voluntarily give out information, especially when falling victims. Nevertheless, this module collected 532 *unique* numbers. We observed that a good share of the vishers resort to automated responders. In such calls, popular terms such as "press", "credit", "account", are more frequent on automated calls with respect to calls made by live operators.

The email module has been processing spam emails provided from an ISP located in Southern California. In less than one month, the system selected 16,750 emails containing at least one telephone number, which amount to the 0.047% of the total number of spam emails collected by the ISP. Overall, this module collected 152 unique phone numbers as the time of writing.

With the support of a large telecommunication provider, the *phone* module is being deployed on a number of DSL lines to begin calling our initial list of 685 numbers.

4 Limitations and technical challenges

The main limitation of our approach lies in phone numbers collected by user-submitted reports, that could be very well spoofed identifiers. In fact, based on a few probing calls we placed manually, a good share of numbers (a rough 10%) are either deactivated or non-existing; unfortunately, it is difficult if not impossible to tell spoofed, blacklisted or deactivated numbers apart.

The main technical challenge of our system lies in the phone module. Specifically, even accurate speech-to-text software are far from being able of transcribing an entire conversation. We plan to workaround this obstacle by recognizing only a finite set of known (key)words extracted from reverse-vishing emails.

5 References

[1] S. Antonatos, I. Polakis, T. Petsas, and E. P. Markatos. A systematic characterization of im threats using honeypots. In *NDSS*, 2010.

[2] T. Cymru. the underground economy: priceless. `http://www.usenix.org/publications/login/2006-12/openpdfs/cymru.pdf`, December 2006.

[3] C. Grier, K. Thomas, V. Paxson, and M. Zhang. @spam: the underground on 140 characters or less. In *Proc. of the 17th ACM conf. on Computer and Communications Security*, CCS '10, pages 27–37, New York, NY, USA, 2010. ACM.

[4] M. Hofman. There is some smishing going on in the eu. `http://isc.sans.org/diary.html?storyid=6076`, March 2009.

[5] Internet Identity (IID). Phishing trends report: First quarter 2010. Technical report, 2010.

[6] T. Lauinger, V. Pankakoski, D. Balzarotti, and E. Kirda. Honeybot, your man in the middle for automated social engineering. In *Proceedings of the 3rd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more*, LEET'10, pages 11–11, Berkeley, CA, USA, 2010. USENIX Association.

[7] F. Maggi. Are the con artists back? a preliminary analysis of modern phone frauds. In *Proc. of the 10th IEEE Intl. Conf. on Computer and Information Technology*, pages 824–831, 2010.

[8] M. Porter. An algorithm for suffix stripping. *Program: electronic library and information systems*, 40(3):211–218, 2006.

[9] G. Stringhini, C. Kruegel, and G. Vigna. Detecting spammers on social networks. In *Proc. of the 26th Annual Computer Security Applications Conf.*, ACSAC '10, pages 1–9, New York, NY, USA, 2010. ACM.