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

WORLDWIDE OBSERVATORY OF MALICIOUS BEHAVIORS AND ATTACK THREATS

D18 (D4.6) Final description of contextual features

Contract No. FP7-ICT-216026-WOMBAT

Workpackage	WP4 - Data Enrichment and Characterization
Author	Davide Balzarotti
Version	1.0
Date of delivery	M36
Actual Date of Delivery	M37
Dissemination level	Public
Responsible	Eurecom

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

SEVENTH FRAMEWORK PROGRAMME

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

The WOMBAT Consortium consists of:

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

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

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

Contents

Abstract

The objective of Workpackage 4 is to develop techniques to characterize the malicious code that is collected in the previous workpackage. The main idea is to enrich the collected code thanks to metadata that might reveal insights into the origin of the code and the intentions of those that created, released or used it.
This deliverable is an extension of D15 (D4.5), and provides a final description of the contextual features collected within the WOMBAT consortium. Furthermore, it presents initial results, statistics, and insights obtained by analyzing the collected contextual features.

1 Introduction

The Internet has become an essential part of the daily lives of many, as more and more people are making use of services that are offered on the Internet. It evolved from a basic communication network to an interconnected set of information sources enabling, among other things, new forms of (social) interactions and market places for the sale of products and services. Online banking or advertising are mere examples of the commercial aspects of the Internet.

Just as in the physical world, there are people on the Internet with malevolent intents that strive to enrich themselves by taking advantage of legitimate users whenever money is involved. Malware (i.e., software of malicious intent) helps these people accomplish their goals.

To protect legitimate users from these threats, security vendors offer tools that aim to identify malicious software components. Typically, these tools apply some sort of signature matching to identify known threats. This technique requires the vendor to provide a database of signatures. Then, these manually created signatures are compared against potential threats. Once the security vendor obtains a sample of a new potential threat to study, the first step for a human analyst is to determine whether this (so far unknown) sample poses a threat to users by *analyzing the sample*.

If so, the analyst attempts to find a pattern that allows to identify this sample (i.e., the signature). This pattern should be generic enough such that it allows matching with variants of the same threat, but not falsely match on legitimate content. The analysis of malware and the successive construction of signatures by human analysts is time consuming and error prone. At the same time, it is trivial for malware authors to automatically generate a multitude of different malicious samples derived from a single malware instance. An antivirus vendor that receives thousands of unknown samples per day is not extraordinary nowadays. This substantial quantity requires an *automated* approach to quickly differentiate between samples that deserve closer (manual) analysis, and those that are a variation of already known threats. This automatic analysis can be performed in two ways. *Dynamic* analysis refers to techniques that execute a sample and verify the actions this sample performs in practice, while *static* analysis performs its task without actually executing the sample.

This deliverable presents the final description of the contextual features adopted by the WOMBAT partners to characterize malware and malware-based threats. The report

expands the preliminary discussion provided in D15 (D4.5), introducing a substantial amount of new material. In the following sections, we describe the specific datasets maintained by the WOMBAT partners. Each description contains the final list of the contextual features that are stored in the dataset, and give some examples of the collected data.

In addition to what was already presented in the intermediate report, this deliverable extends the set of features, provides statistics on the values observed in the collected data, and presents insights on the way certain features can be used to detect malware trends and emerging attack techniques. Finally, each section has been enriched taking into account the lesson learned from the integration and collaboration with the other partners in the WOMBAT consortium.

2 Dataset: SGNET (Eurecom)

SGNET is a distributed honeypot deployment designed for studying server-side attacks and collect insights on the propagation vectors used by malware. As described in D3.2, SGNET aims at minimizing the cost of deployment of new honeypot sensors by leveraging protocol learning techniques and reusing knowledge acquired from previous interactions to handle new instances of a specific network activity. Thanks to these techniques, and through the employment of memory tainting tools developed in the context of the project such as Argos, SGNET honeypots are able to emulate at low cost server-side code injection attacks up to the point of a successful malware download. In designing the system, a specific effort has been devoted to couple each collected malware sample with verbose information on the context in which the malware propagation took place. This includes specifics on the structure and characteristics of the exploit, as well as information on the attacking source and its relationship to the victim.

Table 2.1 lists all the contextual features stored in the SGNET dataset for each malware propagation witnessed by the honeypots. Additionally, Table 2.1 provides a statistical characterization of the "feature space" by analyzing the information collected in the last two years of operation of the deployment, namely between the 1st of January 2008 and the 1st of December 2010. For each feature, we report:

- The number of distinct values observed for the feature throughout the reference dataset.
- The minimum, average and maximum frequency of each value where the frequency is defined as the number of code injection events witnessed by the deployment associated to that specific value.
- In order to characterize the statistical distribution of the features, we compute the standard deviation of the frequencies and the percentage of events associated to the 10 most popular feature values observed in the dataset.

The list of features presented in this deliverable extends and completes the preliminary insights offered by D15. We can distinguish three main feature groups: 1) information on the dynamics of the malware propagation; 2) characterization of each observed source, as well as its relationship to the attacked address; 3) information on the characteristics of the propagation vector.

Name	Type	# values	Min freq.	Mean freq.	Max freq.	Std. Dev.	Top 10 coverage
Propagation dynamics							
Event date	Date	947	1	162.61	635	122.90	0.04
Source and victim characterization							
Source /8 prefix	Integer	157	1	980.83	30477	2692.09	0.50
Source /16 prefix	Integer	10642	1	14.47	27702	271.97	0.24
Source country	String	177	1	624.35	35301	2794.72	0.64
Destination /8 prefix	Integer	25	1	6159.64	39392	10151.32	0.90
p0f label	String	1613	1	48.16	6476	322.44	0.43
Src/dst network distance	Integer	92176	1	1.67	409	6.46	0.02
Src/dst network overlap	Integer	24	1	6416.29	52327	11759.10	0.92
Code injection characteristics							
Extended port sequence	List of strings	13868	1	7.84	23829	221.54	0.48
Compact port sequence	List of strings	3098	1	35.09	29091	728.73	0.74
FSM traversal	String	145	1	778.92	22424	3366.78	0.92
Download protocol	String	6	21	14829.83	26479	10783.63	1.00
Download port	Integer	8714	1	7.80	26455	283.99	0.44
Download filename	String	19894	1	3.49	20148	151.41	0.43
Download repository	Address	341	1	7.27	242	21.33	0.44

Table 2.1: SGNET contextual features

2.1 Propagation dynamics

Honeypot deployments such as SGNET offer valuable information on the temporal evolution of malware propagation. By looking at the localization of the infection sources and of their target, and at their evolution over time, we can get insights on the malware propagation strategy.

For instance, Figure 2.1 shows the observed behavior for different malware samples known to be variant of the same bot family. The different variants are plotted along the x axis, while the y axis represents, from top to bottom:

- The location over the IP space of the hosts seen by SGNET as pushing such malware sample

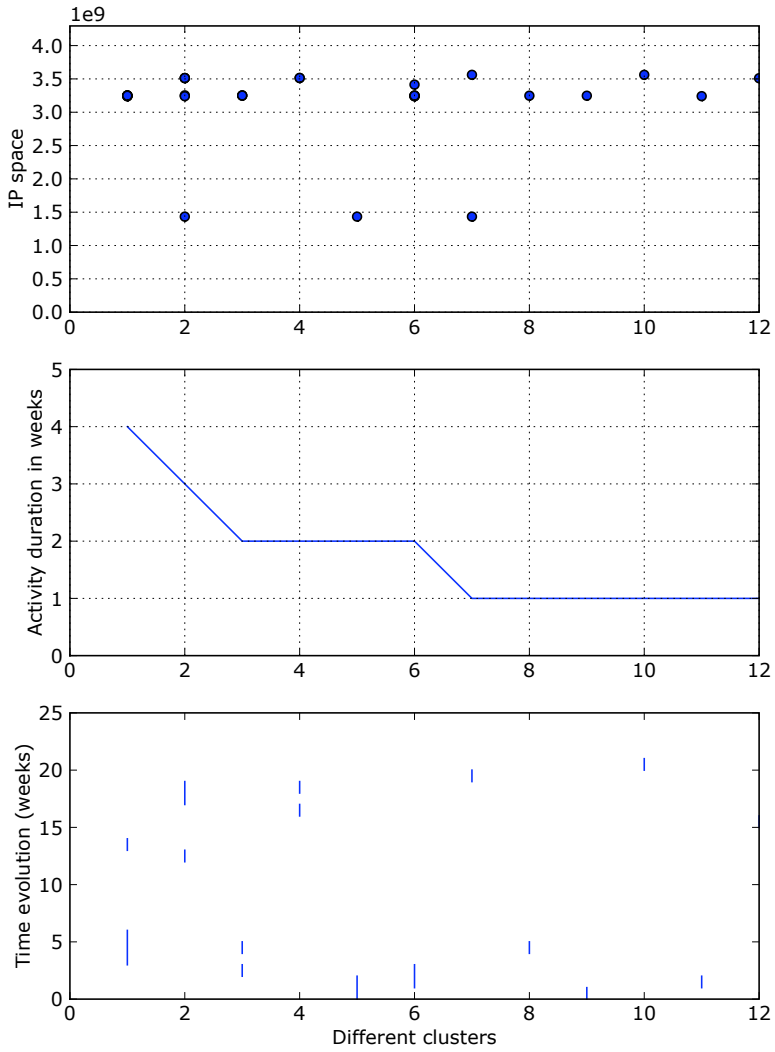

Figure 2.1: Propagation dynamics for different bot variants observed by SGNET honeypots

- The number of weeks in which such malware sample was seen as active by SGNET
- The period of activity of each sample, plotted over a 13 months timeline

The temporal evolution exposes the expected bot-like coordinated behavior. For instance, the variant with longest activity evolves over time in the following way:

- 15/7 - 16/7: observed hitting network location A
- 18/7: observed hitting network location B
- 26/7: observed hitting network location B
- 2/8-4/8: observed hitting network location A
- 27/9: observed hitting network location B

Propagation dynamics is a potentially very interesting source of information on the malware evolution, but is also highly dependent on SGNET's coverage of the IP space. Such challenge is actually clear from the previous example. SGNET honeypots observed the propagation of the different malware variants for relatively small timespans, after which the associated botnet was probably pointed towards an area of the IP space not covered by any SGNET honeypot. For this reason the considered contextual information is limited, and simply consists in tracking the day in which each malware sample has been witnessed propagating to one of the honeypots.

(F.1) Event date. For each code injection event observed by the SGNET deployment, we consider the date in which the event took place.

2.2 Source and destination characterization.

Every network activity carried out by an IP source towards an SGNET honeypot is recorded in the SGNET dataset. The localization of the attacking source, as well as its relationship to the victim address, can provide important insights on the nature of the activity, its scope and the propagation strategy employed by the malware.

When looking at the localization of the attacking source, we consider both the network localization (CIDR block) and the physical location of the IP address by leveraging geolocation databases. There are in fact many reasons for which the spreading malware samples can be affected by geographical boundaries.

- The infection localization can be a result of the malware propagation strategy. For instance, malware samples coordinated by means of a C&C channel might propagate only towards specific subnets specified by the bot herder. In these cases, the localization of the infected hosts might provide information on the modus operandi of the botnet creator.

- Malware might be specifically designed to specifically target, or avoid, a specific population. For instance, malware such as Conficker.A is designed to suicide whenever the keyboard language is detected as ukrainian[1].

- Malware propagation might be biased by the characteristics of the exploit and the associated vulnerability. An exploit might be successful only under certain localizations of the vulnerable application, preventing infection of devices located outside a specific geographical area.

Very similar considerations can be applied to the relationship between the source and victim addresses. The propagation strategy, either hard-coded in the malware sample or defined through C&C interaction, influences the relationship between source and victim. We can envisage two different ways to express this similarity: on the one hand, we can define a generic notion of distance between the binary value of two addresses as $d = |inet\_aton(src) - inet\_aton(dst)|$; on the other hand, we can search for specific biases introduced by the propagation strategy. For instance, some malware samples have been witnessed as propagating only within their very same /8 network prefix: measuring the length of the shared prefix between source and destination address could help us in identifying these cases. Figure 2.2 corroborates this intuition by showing the distribution of the shared prefix length on the code injection attacks observed by SGNET. We define as shared prefix length the number of continuous bits starting from the most significant bit that share the same value in both source and destination address. Figure 2.2 shows marked peaks at specific prefix lengths, that seem to reflect specific propagation strategies.

(F.2) Source /8 prefix and (F.3) /16 prefix. Malware infections are sometimes localized to specific portions of the IP space. By looking at the relationship of each source to specific 8-bit and 16-bit long network prefixes we can identify attacks locality and discover, for instance, malware variants that spread within a specific set of network blocks.

[1] `http://mtc.sri.com/Conficker/`

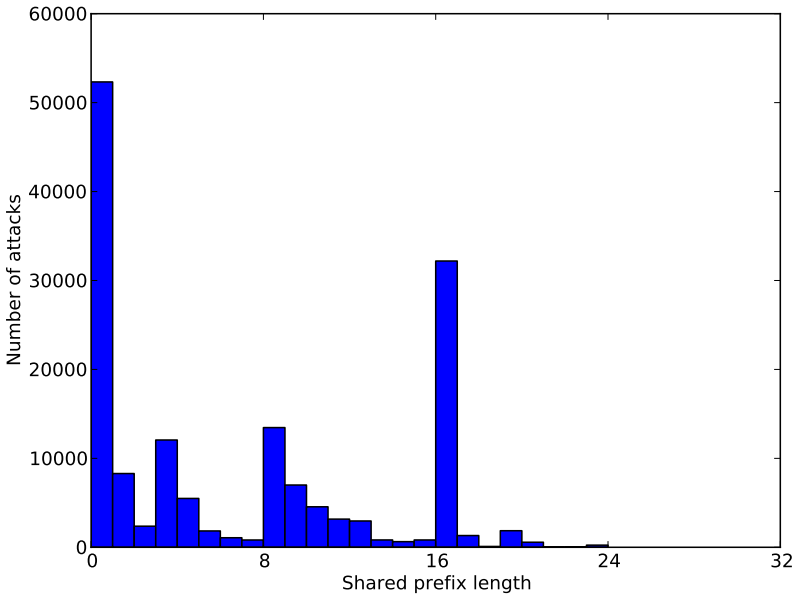

Figure 2.2: Length of the shared prefix between source and victim

(F.4) Source country. Following the previously explained reasoning, we consider geographical boundaries in parallel to network locations. We take advantage of the Maxmind IP geolocation database to associate each source to its country of origin, and detect geographical localization of the attacks.

(F.5) Destination /8 prefix. Similarly to the information on the source localization, the location of the victims targeted by malware propagation provides insights on the targeted nature of Internet attacks.

(F.6) p0f label. While propagating to SGNET honeypots, the network conversation generated by attackers contains useful hints for an approximated identification of the operating system employed by the attacking source. Passive OS fingerprinting tools such as p0f [2] are able to identify specificities in the TCP/IP stack of the source that are indicative of a specific Operating System version. This feature

[2] `http://lcamtuf.coredump.cx/p0f.shtml`

reports the output generated by p0f when applied to the traffic generated by the source in propagating the malware to the victim. This information can be useful when studying the composition of the population affected by a specific malware variant, and in trying to draw conclusions on the ability of the malware sample to infect a specific OS configuration.

(F.7) Src/dst network distance. This feature reports on the distance between a source and destination address, computed as $d = |inet\_aton(src) - inet\_aton(dst)|$.

(F.8) Src/dst network overlap. As previously explained for Figure 2.2, this feature represents the amount of contiguous overlapping bits between source and destination counting from the most significant bit of the address. For instance, the network overlap between source 192.168.0.1 and destination 192.168.130.1 is 16.

2.3 Code injection characteristics

As described in D3.2, the different components of the SGNET deployment generate information on the propagation strategy of the malware samples. Such information provides interesting insights on the modus operandi of malware writers.

(F.9) Extended port sequence. The interaction between an attacking source and a victim is often composed by multiple interactions over TCP sessions, UDP and ICMP exchanges. For each SGNET event, the sequence of network operations that led to the successful code injection is captured in the so call extended port sequence, an ordered list of all the ICMP requests, UDP packets, and TCP sessions established by the attacker in the malware propagation. Each transport type is characterized by simple attributes:

- ICMP requests are characterized by their type and code. For instance, and echo request is represented as '80I' (type=8,code=0).
- UDP packets are characterized by the destination port. For instance, a UDP packet towards the honeypot port 137 is represented as '137U'.
- TCP packets are reassembled into TCP sessions. Each TCP session is characterized by the server port on the honeypot. For instance, a TCP connection towards honeypot port 445 is represented as '445T'

(F.10) Compact port sequence. Compact port sequences are a simplification of extended port sequences. Only TCP packets are taken into consideration, and each

targeted port appears only once when it's first hit by a packet. For instance, the following extended port sequence: $80I|80I|445T|139T|445T$ corresponds to the compact port sequence 445|139.

(F.11) FSM traversal. SGNET models protocol interactions through Finite State Machines that represent the protocol language. These Finite State Machines are the output of the ScriptGen algorithm, that allows the generation of generic protocol models starting from samples of interaction between real hosts. While these models allow the emulation of future instances of the same activities, they also proved their usefulness as a way to categorize in depth network activities. This feature provides a detailed classification of the exploit used to carry on the code injection by providing an identifier of the FSM traversal followed by the attacker to achieve it.

(F.12) Download protocol. For every code injection attack witnessed by the system, the shellcode meant to be executed by the victim is identified and fed to a shellcode handler. The role of the shellcode handler consists in identifying the type of behavior that would be generated by the shellcode if executed, and in emulating this behavior. The download protocol consists in the high level categorization of this behavior, and can consist of standard protocol such as http,ftp and tftp (the shellcode execution would lead to the execution of an http/ftp/tftp download from a remote location) or more ad-hoc protocols expressing shellcode-specific behaviors.

(F.13) Download port. Whenever applicable, this feature describes the server port involved in the shellcode emulation. While some malware variants randomize the port at each propagation attempt, we have seen many cases in which the port is an immutable of a certain propagation vector. For instance, the Allaple worm is known to propagate its content towards a very specific port, TCP port 9988.

(F.14) Download filename. Whenever applicable, this feature describes the file name and path used during the malware download. For instance, the Blaster worm is known to be associated to a specific filename retrieved over a tftp transaction, *msblast.exe*.

(F.15) Download repository. Most of the malware propagation witnessed by the SGNET consisted either of "push" protocols (in which the attacker actively pushes the malware sample towards a port that the victim was forced to open through shellcode execution) or "phone-home" protocols (in which the victim actively connects back to the attacking address to retrieve the malware sample). In a minority of cases, we

have witnessed the existence of a "third party" involved in the malware propagation: an external malware repository, different from the attacker IP address, that hosts the malware code. This feature lists the address of this repository, whenever applicable.

3 Dataset: ANUBIS (TUV)

3.1 The Anubis Malware Behavior Database

As discussed in Deliverable D06 (D3.1) "Infrastructure Design", Anubis is a dynamic analysis environment for malicious code. When analyzing a binary, Anubis executes it in an instrumented sandbox environment and produces a human-readable *Behavioral Analysis Report* describing the system- and network-level actions performed by the binary. This report generally contains enough information for a human expert to establish whether the suspicious binary is in fact malicious, and to get a high-level overview of some of the malicious functionality it implements.

As part of the WOMBAT project, the Anubis service was enhanced by implementing a Malware Behavior Database that is used to store the information available in *Behavioral Analysis Reports* and allow efficient searching, data mining, and analysis of the collected data. The Anubis Malware Behavior Database was briefly discussed in Deliverable D08 (D4.1) "Specification language for code behavior".

The purpose of this database is primarily to store *behavioral* features of the analyzed malware samples. However, some of the malware's behavior, and particularly its network activity, can reveal important information on the context of the malware infection, such as the command and control infrastructure it employs. In some cases, this information can be directly compared and correlated with contextual information available from other sources. As an example, the same HTTP server may be observed serving malware binaries to executables analyzed by Anubis, and to exploits captured by the SGNET distributed honeypot (as reflected in the "Download Source" feature discussed in Section 2.3).

3.2 Anubis and the Malware Life-Cycle

> "Adding to this confusion is the tendency of the botnet owners to frequently change Pushdo's functionality and code. *It is perhaps better to think of Pushdo as a criminal operation, rather than a single piece of malware*"
>
> Trend Micro report on the Pushdo/Cutwail botnet, May 2009 [4].

Malicious software is a crucial component of criminal activity on the internet. Because of the increasing complexity of the techniques employed by cyber-criminals, malware samples cannot simply be analysed in isolation but must be considered as components of the infrastructure deployed by cyber-criminals to support their fraudulent activities. Compared to honeypots such as SGNET (discussed in Chapter 2) or HSN (discussed in Chapter 8, Anubis provides information on different stages of the malware life-cycle. Client-side and server-side Honeypots observe the propagation mechanisms with which hosts are initially infected with malware, such as exploits against network services, and drive-by-download attacks from malicious web servers. Anubis cannot provide this information: Samples are directly submitted by end-users, network administrators, or other security researchers. In most cases, we do not have information on how a binary that is submitted to Anubis for analysis was originally obtained by the submitter. In fact, in some cases the submitter himself may not know: end users sometimes submit a binary to Anubis precisely because they do not know how it found its way into their systems.

On the other hand, Anubis provides a wealth of information on the behavior of malicious code *after* the host has been compromised. While malware behavior has been discussed in D08 (D4.1) "Specification language for code behavior", and is outside the scope of this document, the network-level activity performed by a malicious binary reveals valuable information on the context of the malware infection, such as the command and control infrastructure it relies on, or the modus operandi employed for malware propagation. These network-related contextual features can in many cases directly be compared with other data sources, as well as provide additional insignt into suspicious traffic observed at the network level.

In the following sections, we discuss contextual features available in the Anubis database. Before introducing the network-level features, we discuss contextual features related to Anubis sample submission.

3.3 Submission Features

As discussed in the previous Section, Anubis can provide limited information on the origin of a malware sample. Nonetheless, some contextual data is available that may provide additional insight.

Number of submissions. A single binary sample may be submitted to Anubis by several users over time. The number of submission provides an indication of the prevalence of the binary in the wild. Furthermore, polymorphic malware generally produces unique binaries for each infection. Therefore, binaries that are submitted multiple times by

Table 3.1: Submissions by TLD

edu	at	org	net	com	info
29%	27%	13%	7%	2%	1%

different users are extremely unlikely to be polymorphic.

Of all submitted samples, 7% were submitted at least from two different users.

Time of submission. For each submission of a binary, the date and time that it was submitted. This can provide some information on the period of time over which this malicious binary was active in the wild. In particular, the first time of submission for a binary provides a useful lower bound for the "age" of the malware sample.

Submitter. The Anubis user(s) that submitted the binary to Anubis. Since registration is not required to submit samples to Anubis, this information is not always available. At the moment, the 1692 registered users account for 89% of the submitted samples.

Submitter IP. The IP address(es) from which the binary was submitted to Anubis. If possible, the hostname is resolved via a reverse DNS lookup. This provides some information on the submitters of samples sent to Anubis by anonymous users. Based on the IP address we count 165722 anonymous users. Table 3.1 lists top level domains of the submitter hostnames ranked by their contribution.

Captcha solved. Whether the submitter solved the Anubis CAPTCHA when sending in the sample. This indicates whether the submitter is a human being that manually submits individual samples or an automated program. Users are encouraged to solve the CAPTCHA, since it causes the submitted samples to be analyzed with higher priority. On average, the CAPTCHA is solved for three percent of the submissions to Anubis.

Submitted file name. The name of the submitted file. A single binary may be submitted by different users under different names. The name is often the name under which the malware appears in the wild. Sometimes, users give sample meaningful names, such as "keylogger.exe", before submitting them to Anubis. Finally, honeypots typically give conventional file names to binaries they collect. For instance, the nepenthes honeypot typically stores collected files with names starting with "nepenthes-".

User comments. Anubis users can leave comments on the files analyzed. This is sometimes used to inform us of issues with Anubis, such as binaries that detect the Anubis environment and do not perform their intended behavior. Additionally, some users inform us of the origin of the sample, or share information on the malware's purpose.

So far, we have received comments on 440 submissions. Over seven percent of the comments mention that the submitted sample employs dynamic analysis evasion mechanisms. This information is extremely useful to us because identifying samples that evade Anubis analysis is the first step to developing effective countermeasures to the specific sandbox-detection techniques that are in use in the wild. Additionally, four percent of comments concern the network features of the submitted sample, nine percent are on trojan activities and ten percent on generic viral activity. Overall 91% of the comments are from unregistered users.

3.4 Network Features

Anubis collects all the network traffic generated by the analyzed samples. Information from the IP/TCP/UDP headers and from DNS packets is stored into the Malware Behavior Database. Furthermore, scan detection heuristics are employed, and a number of application-layer protocols that are frequently used by malicious code are detected and analyzed. Specifically, features related to the HTTP, IRC, SMTP and FTP protocols are stored.

UDP Traffic Features For each UDP conversation, the destination IP address and port, as well as the number of packets sent and received.

TCP Traffic Features For each TCP connection, the destination IP address and port, as well as the number of packets sent and received. Additionally, information is available on whether the TCP connection was successfully established and terminated.

Queried Domains For each DNS query, the queried domain, the query type, and the result of the query. While malware also performs queries for legitimate domains, many of the queried domains are related to botnet command and control. Furthermore, the result of the query at the time of analysis is useful because malicious infrastructure is frequenly hosted using fast-flux hosting techniques. Therefore, resolving the domain name at a later date will lead to different IP addresses.

Opened Ports TCP and UDP ports that the binary opened for listening. Opening ports is an indication of backdoor functionality. The use of certain port numbers may be characteristic of certain malware variants. This information can therefore allow attribution of suspicious traffic observed at the network level to a specific malware.

Port Scans The target IP address and number of scanned ports for each host that is scanned. Patterns in scanning behavior can be used to correlate malware analyzed in Anubis with attacks observed by honeypots such as SGNET.

Address Scans The scanned subnet and the number of scanned hosts.

HTTP Traffic Features These include the type of HTTP request (GET,POST,..), the requested Host and URL, the client and server headers, the User-Agent, and the content type of the requested page. Furthermore, the entire body of POST requests and of HTTP replies is available. The IP address, DNS domain name and TCP port used by the contacted server are of course also available, as they are for all TCP connections. HTTP is currently the main protocol used by malware for Command and Control, therefore many of the HTTP servers contacted by Anubis malware are malicious.

IRC Traffic Features These include the name, password and topic of IRC channels joined by the malware, as well as the nickname and username it employs. IRC servers contacted during Anubis analysis are invariably used for Command and Control, but they may in fact be legitimate, publicly accessible servers. Malware authors create their own channels on these servers and use them to control their botnets.

SMTP Traffic Features Emails sent by malware inside Anubis are invariably SPAM or phishing emails. SMTP features include the recipient name and email address and the subject, sender address, and full content of the email. Furthermore, the name and file type of all attachments is available. This data can help identify the origin of spam campains that send messages similar to those observer in Anubis.

FTP Traffic Features These include the username and password used to connect to the FTP server.

3.5 Command and Control Communication

Modern malware uses a command and control (C&C) infrastructure that allows the malware operators to remote-control the infected machines (also known as bots). It also lets them update the bots' software to adapt to the changing environment in which they operate and to the changing goals of the botnet owners. While some botnets employ peer-to-peer protocols for C&C, most employ client-server architectures and rely on redundancy and fallback mechanisms to provide robustness. The C&C servers are therefore a weak point of a botnet's operation, making information on their domain names or IP addresses extremely valuable to security practitioners. Such information has been used for coordinated takedowns of a botnet's C&C servers, that in some cases have succeeded in completely shutting down a botnet [3]. In a few cases, C&C server information has even led to the networks of malicious internet service providers being depeered from the internet [9]. Even if the malicious servers cannot be taken down, blacklists of C&C servers such as the one provided by FIRE [12] can be used by network administrators as an additional layer of defense. A C&C blacklist provides two benefits: on the one hand, it prevents infected hosts from receiving commands that would lead them to engage in harmful behavior; On the other hand, it can alert a network administrator to the presence of infected hosts on his network.

To assist in the task of identifying and blacklisting C&C servers, we measure the number of potential C&C servers contacted during dynamic analysis. The features relevant for C&C server detection are those indicating network communication with an IP address, and those indicating a DNS request for resolving a domain. A simple way to measure the number of potential C&C servers would be to count the number of distinct network endpoints (IP addresses and DNS names) contacted by the analyzed samples. However, not all network traffic observed is related to C&C, or to malware's auto-update functionality (which can be seen as a type of C&C where commands are delivered in the form of executable code). Therefore, we employ a number of additional filters:

- *fast-flux*: C&C infrastructure is sometimes hosted on fast-flux networks, where the same domain name resolves to a rapidly-changing set of IPs. These IP typically belong to infected hosts that temporarily serve as C&C servers. In this case, the information on the contacted IP addresses is of little value. Therefore, we consider only IP addresses that the malware did not obtain from a DNS query.

- *portscan*: Many malware samples include self-propagation components that scan the internet for targets before attempting to compromise them. Clearly, we do not want to include these hosts in the network endpoints score. For this, we first discard connections to a small number of ports that are known to be typical exploit targets, such as TCP

ports 139 and 445, used by Windows file and directory sharing services. In fact, our analysis sandbox is already configured to redirect traffic on these ports to a honeypot to prevent propagation. Furthermore, we use the Bro IDS [10] to detect port and address scans, and discard connections that are part of detected scans.

- *liveness*: This heuristic aims to filter out endpoints that could not be successfully contacted by the malware, such as C&C servers that are no longer active. For this, we use protocol-specific heuristics for a few protocols commonly used for C&C, and fall back to a default heuristic for other kinds of traffic:
  - HTTP: we only consider servers that responded with a successful status code (that is, 200-299) to at least one request.
  - IRC: we only consider a server if the malware sent or received a private message or if it successfully joined a channel and sent or received a message on that channel.
  - FTP: we only consider a server if the malware successfully logged in.
  - Other: we only consider endpoints where a connection was successfully established (for TCP) and where the server sent back some data.
- *clickbots*: Clickbots are malware samples that include functionality to automatically visit advertisement links on target websites. Their goal is to fraudulently generate advertisement revenue for these websites. Due to the dynamic and tiered nature of advertisement networks, this can lead to a significant number of network endpoints being contacted during analysis. Since these endpoints are not related to command and control, we filter most of them out by using an existing list of ad-related domains that is manually maintained for the purpose of blocking advertisement [5].

In the following we will show results on a dataset consisting all samples that have been analyzed by Anubis between July and September 2010 – 643,212 samples in total. Table 3.2 shows to which extent filtering reduced the total number of network endpoints in the dataset to potential C&C endpoints.

The samples were then clustered using the Anubis behavioral clustering techniques described in deliverables D4.1 and D4.2 resulting in 79368 clusters of which 15575 contain more than one sample. Table 3.3 shows measurement results of potential C&C servers in these clusters. The clusters are ranked based on the feature count to cluster size ratio.

Indeed, almost all top-ranked clusters belong to remote controlled bots or trojans. Furthermore, several of these are associated with malware families that are known to use highly dynamic C&C infrastructure. One example is the pushdo/cutwail botnet (discussed extensively in [7]), found at rank three. Pushdo binaries contain a frequently-updated list of IP addresses. The malware contacts these addresses over HTTP to

Table 3.2: Network endpoints in samples from July to September 2010

Network Protocol	total IPs	filtered IPs	total Domains	filtered Domains
FTP	30	19	431	360
HTTP	1644	1207	15369	11545
IRC	36	13	546	233
other	279011	1551	1914	1882

Table 3.3: Clusters from July to September 2010 ranked by network endpoint to size ratio

Rank	Size	Network Endpoints	Malware Family
1	36	84	Unknown Bot
2	20	20	Harebot
3	29	26	Cutwail
⋮			
6	31	24	Adware Adrotator
15	67	32	Bredolab
28	141	51	Koobface
36	82	27	Swizzor
67	173	33	zBot
273	189285	2596	Unknown Downloader
⋮			
Last	16370	0	Allaple/Rahack
Last	28179	0	No activity

download a binary payload: typically an instance of the cutwail spam engine. Cutwail then proceeds to obtain templates and instructions for spamming from other C&C servers over a custom binary protocol. Likewise, the koobface botnet, at rank 28, is known to use compromised web pages as part of its C&C infrastructure. Bredolab at rank 15, is a downloader similar to pushdo [11].

Vundo/Zlob at rank eight, is a Fake-AV downloader. These samples download binaries from a number of different servers.

In a few cases, clusters that we are not necessarily interested in are ranked high. The top non-relevant cluster is the Adrotator cluster at rank six. The samples in this cluster are clickbots: They visit advertisement links to fraudulently generate advertisement

revenue. In this case, some of the advertisement servers in questions are not in our adblock blacklist, therefore they contribute to the feature to size ratio.

Among the clusters with no potential C&C servers an Allaple cluster can be found, which performs network scans that are filtered. Also placed at the bottom is another cluster performing no network activity at all.

4 Dataset: Virustotal (HISPASEC)

The VirusTotal WAPI Dataset was conceived as a means of querying information on files uploaded to the free online service (http://www.virustotal.com) and retrieving metadata about the submissions themselves (number of times a file was sent, names with which the files were uploaded, etc.).

VirusTotal was initially developed as an online multiantivirus scanning tool, as such, the obvious piece of information that can be included in its dataset are the antivirus engine results themselves. Having said this, it is not the sole data that is present in the VirusTotal database. Soon, the web application started to integrate file characterization tools such as PEiD (identification of packed files), Sigcheck (identification of files that are digitally signed), PDFiD (characterization of PDF files), etc. The output of these tools is part of the dataset, hence, this dataset can be described as a static file characterization database. The word file must be emphasized, many of the other WAPI datasets work only with confirmed malware, given the public nature of VirusTotal, all kinds of files are submitted on a daily basis. Since many of its massive users are malware research related teams (universities, antivirus labs, honeypot and honeyclient deployments, etc.), a noticeable portion of the received files are indeed malware. Nonetheless, also freeware, shareware, all kinds of software, legitimate music files, documents and other innocuous files are present in the VirusTotal store and can be queried via its dataset.

As of July 2010 VirusTotal was extended with an online community that allows its users to place comments on samples and tag them with different labels: goodware, malware, spam attachment/link, P2P download, propagating via IM, network worm and drive-by-download. This information further enhances the dataset, allowing a collaborative classification of files, for example, it has proven itself to be extremely useful in flagging false positives (goodware or inocuous software that has been incorrectly flagged as malicious by one or more antivirus engines).

4.1 VirusTotal dataset contextual features

The VirusTotal dataset is made up of three basic objects: source, file and analysis, the file object being the keystone of the dataset. A brief description of each type is provided below:

- Source: represents a given VirusTotal user. In the context of VirusTotal a user is considered to be a unique IP, email or VT Community account. The fact that IPs might be reused is ignored.
- File: represents a file submitted to VirusTotal. The reader should note that the term malware is not used since files of any nature are received at VirusTotal. The number of antivirus engines that identify a file as malicious can be used to try to classify a given file as malware or not, nonetheless, this is not a definitive approach due to the problem of false positives.
- Analysis: is associated to a particular scan request on a given file by a specific user, in other words one file may be subjected to several analyses, either because they have been performed at different moments in time or because different users have requested it. The analysis displays all the antivirus engine information returned by VirusTotal on a particular file, the output of other tools is considered to be static and are introduced in the dataset as an intrinsic property of the file object, so are the comments of VT Community users.

All of these objects have several attributes, methods and reference methods that provide useful information to the user and allow for data correlation. A deeper glance at the different types is now given, justifying, where appropriate, why certain pieces of information were included in the dataset.

4.2 Source

Currently this object only has only one attribute, source_type, which identifies, whenever it is known, the nature of the source that sent the file to VirusTotal. A (growing) discrete number of category labels are used: security company, antivirus vendor, telco company, anonymous, etc. Under particular circumstances this feature could be interesting for law enforcement (determination of a malicious origin), yet it may proof itself even more useful when trying to determine whether repeated submissions of a same file are product of the research activity of a particular user or due to the predominance of a given sample in the wild.

Certain users repeatedly submit a same set of samples to VirusTotal in order to detect signature updates and measure antivirus industry reaction times. For example, the file with md5 hash dceb3b61d06e4a96ea13cbd0b0225dfe was sent 100 times, yet only three different sources account for these submissions, one of them being the SGNET deployment with 97 uploads. This user would appear in the dataset with the label *Honeypot/Honeyclient*. This metadata is extremely useful for antivirus and malware research

companies, since it allows them to prioritize analyses according to real prevalance (i.e. multiple sources and multiple submissions rather than one unique source and multiple submissions). Thus, it can be used as a notion of an analysis priorization matrix.

Similarly, sometimes, even though a sample appears to be sent from many different sources, source labelling enables researchers to realize that a high number of submissions is not so much due to a noticiable prevalence in the wild but due to a same malware collection setup. This can be exemplified by the Dionaea honeypot (http://dionaea.carnivore.it/), nepenthes successor.

In 2010 Dionaea included a module that automatically submits the samples it captures to VirusTotal and places the following VT Community comment on them: *This sample was captured in the wild and uploaded by the dionaea honeypot.* Dionaea VT Community accounts are labeled as *Honeypot/Honeyclient*, hence, even though several Dionaea deployments may submit a given sample accounting for many different submission sources, source labeling allows us to understand that this may not be true in-the-wild predominance. In other words, given the ratio of total submissions to honeypot submissions we might deduce that end-users are no longer really exposed to a given threat because their systems are conveniently patched. For example, the malware sample with md5 266a9f6af8d98d5f8ae9b4929d769006 - (which is a variant of the Allaple worm) - has been submitted 8 times to VirusTotal, all of them through a Dionaea setup.

The source object also includes an attribute identifying source's country of origin, making use of tools like Maxmind this is a simple task. This information has proven itself to be useful in identifying country based targeted attacks. For example, we have noticed that certain banker samples directed against Peruvian banks are uploaded to VirusTotal mostly by Peruvian users (Table 4.1). This observation can surely be extended to other kind of targeted attacks.

Moreover, the country information can be combined with other pieces of data present in the dataset to try to build attack detection heuristics. Let us run through a very simple and quick scenario so as to understand how this can be done. The whole set of samples in Table 4.1 were sent to VirusTotal with the following file names: *Postal-Amor-exe.txt, Postal-Amor.amor.exe, Postal-Amor.exe, VideoPostal-AmorSecreto.exe, VideoPostal-Animada.mpg.exe, VideoPostal-exe.txt, VideoPostal-exe.txt, VideoPostal.exe.* Hence, a very simple initial heuristic to find Peruvian banking trojans within VirusTotal's live flux would be to flag those samples where more than 50% of its distinct sources are from Peru and at least one file name has the string *postal* in it. Later on, when looking at VirusTotal's service related metadata we will see that the file naming, in this particular case, can also be used to identify the baiting technique used to infect end-users.

Table 4.1: Submissions for a set of banking trojans targeting Peruvian banks

Sample md5	**Distinct sources**	**Peruvian sources**
d8aa4e787c53debd16d87e76879bf910	4	3
28cf4c1a25ba90116822568fade55a7f	2	2
64b62b8cd506d3ffa2b7bb1e0e3809b3	1	1
7d6bc9f9a2fcd80bb742cb63addb1402	8	7
e9d46df89a83826db26d47687973d265	4	3
4a4b2908a0f0a00ad5d4e66bfca99bbb	20	15

4.3 File

This is the keystone object of the dataset, its attributes can be grouped according to the goal they pursue:

- File identification: md5, sha1, sha256. These are the main hashes currently being used by the malware research community, they are the means for data correlation with other databases and datasets.
- Preliminary file characterization: size (file size in bytes), file_description (magic number descriptor) , portable_executable (determines whether the file is a portable executable), dll (determines whether the file is a dynamically linked library). These attributes provide a quick overview of the type of file that is being dealt with.
- Portable executable immediate information: whenever a file is a Windows Portable Executable, there are a series of file properties that can be immediately retrieved from the database at low cost. These attributes are entry_point (initial execution offset), pe_timestamp (date-time indicating the compilation time of the executable, can be faked), ep_surroundings (first 256 bytes that follow the entry point), machine_type (target architecture of the executable). All these attributes allow for a preliminary morphological characterization of an executable file, it is information that might be used for clustering of malware based on static properties rather than behavioural analysis. This is information is extended by certain object methods that will be later described.
- VirusTotal service related metadata: there are a number of interesting parameters that can be extracted from the submissions to VirusTotal that are not dependant on the binary properties of the file. first_seen and last_seen represent the first and last

submission date-times of the file to VirusTotal, it allows the analyst to know how old, at least, a file is and may give a notion of whether it is still found in the wild. last_detection_ratio gives the fraction of antivirus engines that detected the file in its most recent analysis. num_submissions and num_diff_submission_sources are, respectively, the number of times that the file was sent to VirusTotal and the number of these that were submitted by different sources. Finally, num_hash_requests indicates the number of times a particular file was queried for analysis information via the hash request feature in VirusTotals service web site. These last pieces give an idea of how prevalent a given sample is in the wild.

This set of data categories is not only useful in studying files, it can also be used to build filters when feeding other malware characerization setups such as sandboxes. Sandboxes spend considerably more time than VirusTotal in studying a given sample (execution of the sample), hence, unless a powerful cluster is available, time will be an important constraint. Thus, it makes absolutely no sense to send to the sandbox setup files that it cannot execute. This might be the case of sandboxes that exclusively work with Windows executable files. Figure 4.1 shows that approximately 70% of the files sent to VirusTotal are Windows executable files, therefore, it would have beem a considerable waste of resources to send all files received at VirusTotal to our hypothetical sandbox.

Moreover, we might only be interested in executing samples that we had never executed before, or only files with at least a detection ratio of one engine out of the whole number of engines. Figure 4.2 gives us an idea of how the load on our hypothetial sandbox could be lowered further. This will proof itself very useful when building a banking attacks early warning system in WP5.

With this in mind, the file object then has a series of methods that may also be categorized:

- Tool analysis: get_packers returns the packer signatures that match a given file according to tools like PEiD or the antivirus engines themselves, this allows the analyst to know whether the sample is packed and what unpacker (if any) he should use to get to the original entry point. get_tools_information returns the results of the rest of tools integrated in VirusTotal, for example, thanks to TRiD a WAPI user can get to know what is the most probable file type of a given sample.

- Portable executable advanced information: whenever a file is not packed, get_pe_imports and get_pe_exports are methods that may proof themselves useful in order to have a very top level idea of what a sample is doing based on static inference. On the other hand, get_pe_sections lists the sections of the executable along with basic information about them (entropy, offsets, size), this information can be used when

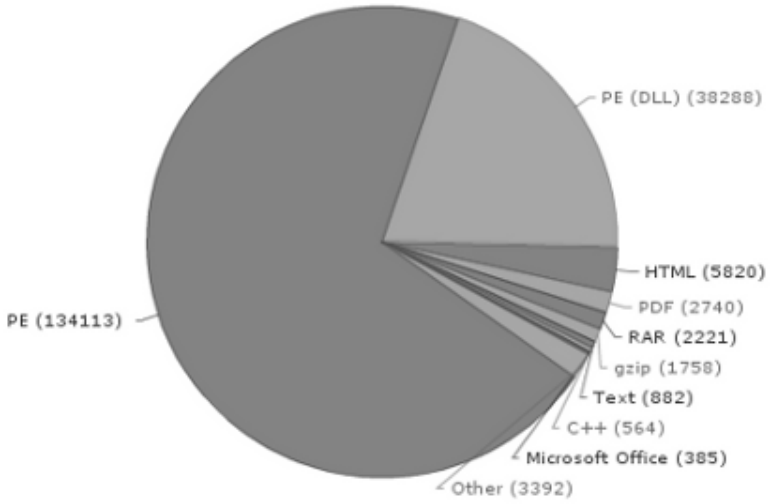

Figure 4.1: Nature of the files sent to VirusTotal on a typical day

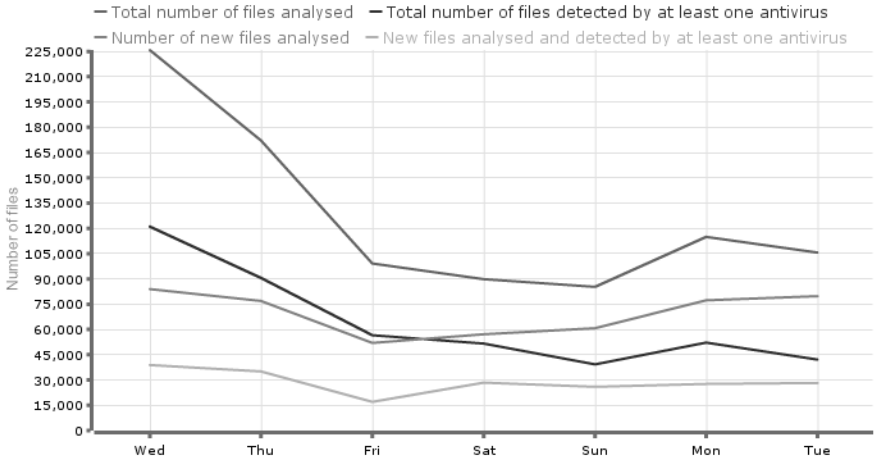

Figure 4.2: File submissions over a typical week

clustering samples according to portable executable properties and to determine whether a sample is potentially packed.

- VirusTotal service related metadata: the dataset stores the file names with which samples are submitted for analysis, sometimes these names will simply be the md5 the samples, this gives no interesting information. However, very often the submission names will be the actual file names with which a given sample is executed in the wild, hence, the get_submission_names method may proof itself very useful when undertaking forensic investigations. In the context of other projects related with VirusTotal, over 3000 spanish home user PCs are scanned for malware, the wild paths and wild names of any files that are present in the VirusTotal database is also part of the dataset and can be queried via the get_wild_names and get_wild_paths methods. Finally, another useful element for forensic analyses is the names with which a file is found hosted in web servers whenever its propagation strategy is via drive-by-downloads or web-based social engineering. Thanks to a honeyclient setup these names are available through the get_wild_names method.

- Antivirus engine analyses: get_detection_ratio_evolution retrieves the detection ratio of all analysis performed on the given file. This may allow researchers to detect signature improvements and signature failures. If something interesting is observed, the dataset user may then get a deeper understanding of what happened using the WR_get_most_recent_analyses method, which returns the latest 100 analyses performed on the sample. Similarly, WR_get_last_analysis and WR_get_first_analysis return, respectively, references to the last and first analysis of the sample.

It is obvious that the information provided by these methods can be used to spot macroscopic malware trends: prevalence of certain malware families, changes in particular banking trojans to avoid antivirus detection, social engineering tactics being used in the wild, etc.

When studying the files referenced in Table 4.1 we mentioned that all of them had been sent with file names of the form *videopostal-{..}* *or postal{..}*, this is the spanish equivalent for postcard and videopostcard. With this infomation, we can easily deduce that the attackers are sending the trojans either as links or attachments in mails pretending to be postcards or videopostcards. Moreover, most of the file names also included the spanish word *amor* (love), hence, they are pretending to be love postcards. File naming also allows us to discover another technique being used to confuse the victims, double extensions. Files are named as *.txt.exe* or similar patterns with the aim of making them think they are inoquous text files.

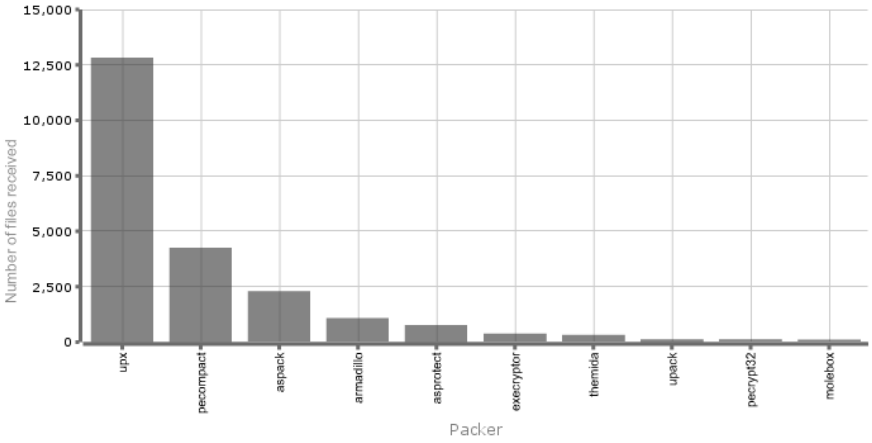

Figure 4.3: Prevalence of packers for submissions on a typical day

Propagation strategies are not the only macroscopic feature that can be observed, Figure 4.3 shows the top10 packers for submissions on a typical average day. The most popular packer is, by far, UPX. This is because UPX is a simple compressor that is used not only in malware but also in all sorts of software. It can also be an indication of the fact that most attackers are just interested in reducing the size of their specimens and providing a first layer of obfuscation of the strings and other metadata in their files, rather than in building complex custom crypters that could be automatically detected by antivirus engines as malicious without having to seek into the real executable behind the protection layer.

This macroscopic view can also be very interesting in detecting new attack techniques. Imagine suddenly a completely new packer appears in the top20, it is going to get the focus of malware researchers and perhaps new techniques for detecting virtual machines or avoided antivirus detection are discovered when studying the particular crypter.

4.4 Analysis

As already mentioned, the analysis entity only includes antivirus engine results, this gives a hint about the malicious nature of a file. Redundant data such as the detection ratio of the analysis is also offered as an attribute. The analysis allows identifying the source that requested it and the moment in time in which it ended.

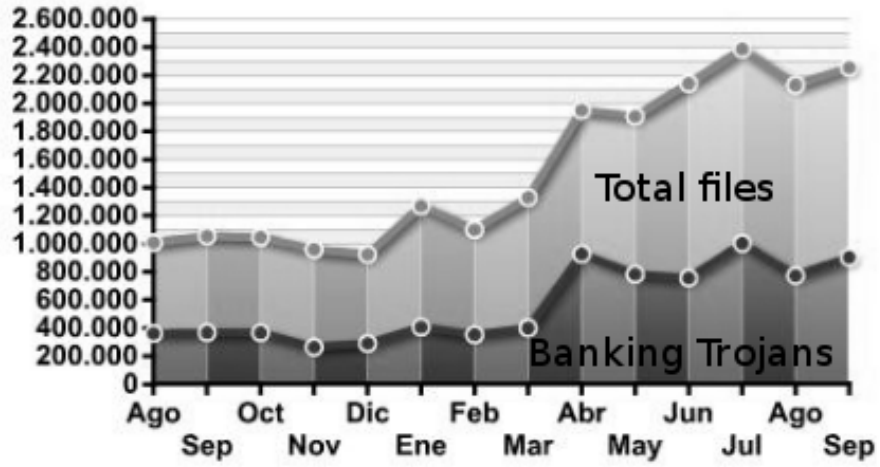

Figure 4.4: Banking trojans received at VirusTotal August08 - September09

Analyses can also be used in order to build macroscopic indicators that enable us to spot trends. For example, by considering typical banking trojan antivirus signatures and weighting the different individual detections Figure 4.4 was produced. Graphs of this nature allow us to understand the growing lucrative goal of attackers, malware production is no longer a question of fame or intellectual challenge, money is moving the market. Similar charts could be drawn for adware, adclickers, online gaming trojans, etc. tu further backup this claim.

As it was previously mentioned, these features are a useful resource to statically characterize samples, but as we have prooved, they are equally interesting for producing macroscoping indicators that enable us to better understand the current landscape regarding malware development.

In conjunction with the rest of the datasets that make up the WOMBAT api it might be a very valuable resource for forensics investigations and all kinds of malware research. The main difficulty with VirusTotal's dataset is its size, with over 57 million samples at the present moment in time (January 2011) and over 100 million analyses, not all pieces of information on a given sample can be returned at once since it would be very costly in terms of api response time. For example, when querying for all the analyses performed on a given sample, only the latest 100 are returned, if this was not the case,

certain samples could have thousands of analysis that would end up in relatively high network transfer times.

5 Dataset: Shelia (VU)

Shelia is a Windows-based honeypot for the client side. The current version is available from `http://www.cs.vu.nl/~herbertb/misc/shelia`. It will detect exploits of client applications and consists of two main parts: a management/control part and an intrusion detection engine. Among other things, the management/control part feeds the detection engine with things to check - URLs, potentially malicious attachments, etc. It allows for different ways to get these items to the detection engine.

A particularly interesting feeder is what we call the *client emulator*. The emulator connects to an IMAP server and emulates a naive user reading email. It will read the email in a folder, dig out all the links and attachments from emails and feeds these to the detection engine. The detection engine blindly follows all the links, opens all attachments, etc.

The intrusion detection engine determines whether the website, or the attachment, is malicious and if so, it raises an alert. Unlike most other systems, Shelia creates virtually no false positives (although there may be false negatives).

The contextual and other features collected by Shelia include the set listed below. We divide the features in categories, specify how they can be useful, and for some them, print an example.

1. Target context
   a) *Target application.* Which application was exploited? For analysis, we need as much information as possible about the victim process. For instance, if we want to be able to say that attackers move from server applications to client applications, or from web browsers to instant messengers, we need to know what sort of process was exploited. In addition, we may be interested in more detailed questions, such as: "Which version of an IE running on what type of host is attacked most". For this purpose, we need more information (e.g., the version number). Shelia currently provides fairly limited information about the victim process - the name of the binary and the path. Example:

      ```
      Target application = C:\Program Files\Internet Explorer\iexplore.exe
      ```

   b) *System address width.* We collect a flag that indicates whether the system on which the malware was collected was 32 or 64 bits. This will help us assess

the vulnerability on different platforms (does it work on 32 *and* 64 bits, or only one of the two. In addition, it will help us repeat the experiment.

c) *Pid.* The process identifier of the victim process. The pid is not directly useful in most security analyses (except if the process that is attacked is one with a well-known pid, which seems unlikely). However, it may be helpful locally to perform further forensics.

2. Capture context

   a) *Timestamp.* When was the alert generated? Time stamps are an important bit of contextual information. They allow us to correlate attacks at different sites, evaluate attack intensity in time periods, and spot attackers' patterns in time (e.g., do attack campaigns start at specific time of the day?). Example:

   ```
   Timestamp = 2009-08-24 10:51:53
   ```

3. Exploit context

   a) *Attack object.* The object identifies the URL or filename that was used to compromise the target. It is important to identify URLs, so as to correlate the URLs with other databases that black list URLs and to assess the lifetime of a malicious site. Example:

   ```
   Object = http://azadars.com
   ```

   b) *Payload.* As accurately as possible, we store the exploit (in binary). The exploit may need further analyses (e.g., to see what methods are used to get the value of the program counter, or to classify the exploit according to the system/API calls that it makes).

   c) *Payload address.* Shelia stores the location of the malware in the victim's memory. The address may help us identify the attack in broad terms. For instance, we may be able to classify the attack as heap overflow. Example:

   ```
   Address = 202571238
   ```

4. Malware identifiers and attributes

   a) *File name.* Shelia stores the name of the malware file that is downloaded on the victim system. While file names may vary as malware spreads, sometimes the filename provides a hint. Example:

   ```
   File name = C:\DOCUME~1\user\LOCALS~1\Temp\update.exe
   ```

b) MD5 and SHA1 and sha-256 hashes: uniquely identify a specific malware binary. They allows to check whether different samples are exactly the same, and to look up the malware in other repositories. Example:

```
Md5    = 00b23b08657a153fcde4e0891e2484bb
Sha1   = 522674387e1a8e2d3ab5f7c11ecd9db7e5904dc4
Sha256 = b851756487f055bb746cae506e5ffc016f88a07177ab7bfc5b8be7208cbc8156
```

c) *Length.* The length of the malware binary in bytes. Often, the hashes of a binary are different, because the malware samples different in a few bytes. In that case, the length of the binary may be a first useful hint to see if binaries 'look' similar.

d) *Binary.* The binary can be analysed, run, tested, etc.

5. Activity context

a) *Calls and call arguments.* For the victim process, we store all calls to a carefully selected subset of the Windows API. This is interesting, as it allows us to assess the attack's *behaviour*. In addition, we may use the calls and the call arguments to whether certain files or registry key values have been modified, etc. As such, they can also be classified as context. Example:

```
Call = 'ReadFile', Args = 'T Y P E L I B'
```

b) *Description.* To further help the security analysis, we store a description of the function that is called. Besides helping out non expert analysts, the description does not play a further role in the contextual analysis.

5.1 Limitations, improvements and redesigns

We discussed Shelia, a Windows-based honeypot for client-side applications introduced in D15[1]. Shelia detects exploits of applications like browsers, mail clients, photo viewers, etc. We have seen that s far as client-side honeypots are concerned, Shelia is interesting, as it is raises virtually no false alarms. In other words, if Shelia claims that something is malicious, you can be sure that it really is malicious. For incident management purposes, this is a very desirable property.

On the other hand, Shelia also has some disadvantages:

[1] `http://www.cs.vu.nl/~herbertb/misc/shelia`

1. Limited contextual information. We designed Shelia originally as an research tool, and never put much effort in really maximizing the contextual information that can be gathered. Specifically, Shelia is limited to:
   a) target application (which program is attacked?);
   b) system address width (is it 32 or 64 bit system?);
   c) process identifier;
   d) timestamp;
   e) attack object (e.g., the file or URL used to compromise the target);
   f) payload (the exploit - but not very precise);
   g) payload address (address of the payload in the victim's memory);
   h) malware (name, length, hashes and binary);
   i) activity context (API calls and arguments of victim under attack).

   There is much more contextual information that can be gathered quite easily (e.g., information about the domain). The information would be quite useful to security analysis.

2. Shelia is Windows-specific (and needs continuous updating with new versions of the OS).

3. The shellcode analysis in Shelia, while interesting for a human analyst, is not powerful enough for automated analysis. This is an example of the Wombat feedback loop, where the actual use of the dataset showed that the information was not quite good enough. The main problem is that the recorded shellcode is quite imprecise and shellcode and unpackers are not clearly distinguished.

To remedy the above issues, we decided to pursue two parallel tracks. First, as an immediate measure, we improved Shelia to make it record more contextual information (this deliverable). Second, we designed a completely new client-side honeypot, based on our popular Argos honeypot technology that is much more accurate in detecting unpackers, and shellcode (deliverable D21). While new, we desiged it such that it can almost serve as a drop in replacement for Shelia. In other words, the architecture itself need not change: we use the same scripts and almost the same databases that drive the Shelia deployment.

The need for more accurate shellcode extraction emerged when VU, TUV, Eurecom and other partners started work on shellcode analysis. It is a good example of the

feedback loop that was explicitly added in the Wombat project. As it does not constitute an extension of the contextual features of D15, we decided to push the description of the client-side Argos to D21. In the remainder of this chapter, we discuss the additional contextual information gather by Shelia.

5.1.1 More contextual information in Shelia

We augmented Shelia to gather more contextual information using a variety of sources. The enhancement operates as a python program that improves the default contextual information that is gathered for each new alert. The Shelia extension was driven by user demand and pragmatism. For instance, we considered how often certain information was needed in the scenarios specified by researchers, how expensive it is to have Shelia obtain the additional information immediately, how difficult/expensive it would be for a researcher to obtain the information later (when needed), etc. Based on this, Shelia now also gathers the following information:

1. domain,
2. domain registrant (with email address);
3. ASN number,
4. geographical information (where the server is located)
5. other suspect IP addresses in that domain (by crosslinking with the NoAH and Harmur databases);
6. port numbers on which the clients contacted the malicious server,
7. HARMUR classification of threat id and threat class (crosslinking with HARMUR);
8. operating system of the malicious server;
9. for malware, Shelia stores
    - the VirusTotal description,
    - the detecton ratio of various AV products at the moment of attack,
    - the meta-names,
    - the linked DLLs.

6 Dataset: NoAH (VU and FORTH)

As described earlier in WOMBAT deliverable D3.2 ("Design and prototypes of new sensors") NoAH is an architecture for large-scale threat monitoring. In a nutshell, NoAH clients, namely Honey@home, forward traffic to unused IP addresses or unused ports to a honeypot farm and forward the replies provided by the honeypots back to the attackers. Honey@home is designed to be simple and lightweight, as its main target audience is home users or administrators unfamiliar with honeypot technologies. The software package that comes with it needs no special configuration to run, and is ported to run on both Linux and Windows platforms. Also, it is non-intrusive as it runs in the background with minimal CPU, memory, and network overhead.

The contextual features that make sense to maintain in the NoAH infrastructure are the IP addresses, timestamp, TCP ports, TCP flags and payload size. Although these contextual features are the same with the features that were presented in the intermediate report, we decided to maintain the current infrastructure, to make sure it is stable and works without problems, and not to add some additional material.

6.1 Geolocation

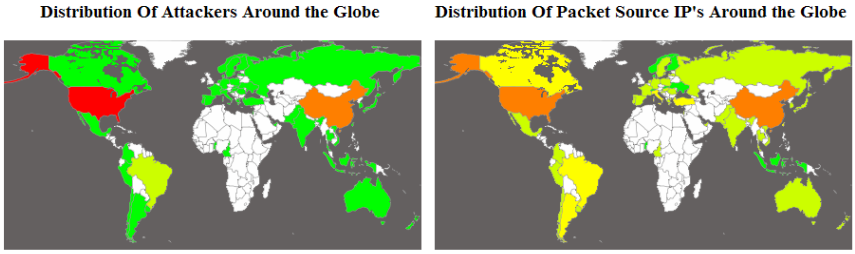

Figure 6.1: Distribution of Attackers Around the Globe

The geolocation of IP addresses provide us information about the attacker's topology. The geographic location is retrieved by querying a local MaxMind database. Based on

this we can categorize the attackers according to their geographical position. That led us to the creation of *Attack maps*. These maps display the geographic distribution of distinct source IP addresses. Each country is colored based on how many IP addresses are hosted in that country. Countries that host no attackers are colored as white, low activity countries are colored as green while countries that host lots of attacking IP addresses are red (Figure 6.1).

6.2 Packet Characteristics

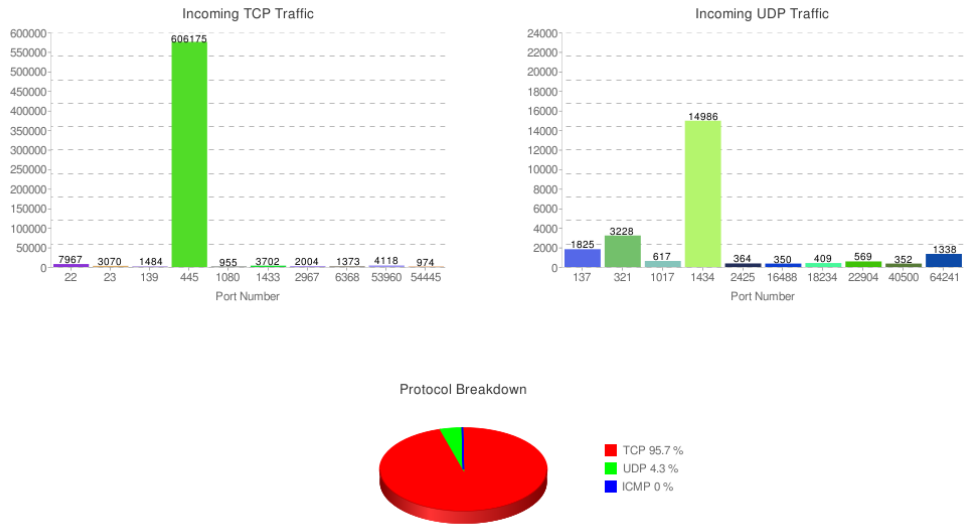

Figure 6.2: Attack graphs

The timestamp the packet was captured is important for our analysis. Based on the packet timestamps, we can identify attack patterns such as diurnal cycles in malware propagation and host scanning. Furthermore, we can gather information about the attacking behavior by examining the inter-arrival of packets. For example, many scanning tools send constant number of packets per second. Timing analysis can reveal such information.

Furthermore, we hold information about TCP ports. TCP and UDP port information help us understand what services are targeted the most so as to configure the NoAH honeypots. By emulating heavily attacked ports, we are able to capture more attack

instances and malware. For example, over the last year we observed a significant increase in sniffing activity on TCP Port 445 that signaled an impending mass malicious code attack targeting a vulnerability over the SMB protocol (Figure 6.2).

Finally, each NoAH sensor receives unsolicited traffic, that is traffic that comes in response to spoofed attacks. It is trivial to identify such traffic by inspecting the TCP flags of each packet.

6.3 Malware Characteristics

The honeypot farm maintained by the NoAH infrastructure runs multiple instances of medium and high-interaction honeypots. These honeypots emulate/run a number of services and are able to capture exploitation attempts, shellcode injections and malware samples.

For the shellcode and malware samples, the information stored in the NoAH database is:

- *Timestamp* is essential to record the date and time the exploitation attempt was made or the malware sample was downloaded. Timestamp allows us to identify the lifetime of a malware sample and the persistence of attack vectors.
- *IP addresses and TCP/UDP ports.* This is the basic information required to identify the attacker source as well as the targeted honeypot and service.
- *Shellcode type.* Based on the stages of the exploitation attempt, we are able to identify what kind of exploit was sent by the attackers. For example, in the case of port 445 we can identify various types of shellcodes, such as DCOM, ASN1, PNP or LSASS vulnerabilities that were triggered.
- *Shellcode* itself is also recorded. The hexdump of the shellcode can be later used for post-analysis.
- *Malware sample.* If the shellcode vector leads to a malware fetch, we download it. For the malware we store the binary data itself as well as the md5 sum of the sample.

Name	Type	# values	Min freq.	Mean freq.	Max freq.	Std. Dev.	Top 10 coverage
Naming information							
DNS A records	List of addrs	1360788	1	1.86	15430	25.22	0.03
DNS NS records	List of addrs	308036	1	6.79	37448	151.07	0.10
Registration date	Date	199904	1	2.51	322	10.21	0.00
Whois registrar	String	8315	1	127.41	352846	4758.00	0.84
Whois registrant	String	394954	1	2.98	587949	936.34	0.55
Server information							
Location	List of tuples	2393	1	1012.21	1182385	24968.93	0.83
Autonomous System	List of int	72938	1	34.52	127396	882.35	0.24
Server version	String	42151	1	62.45	525248	3130.01	0.46

Table 7.1: HARMUR contextual features

7 Dataset: HARMUR (Symantec)

As described in Deliverable 3.2, HARMUR is a repository of information on the evolution of client side threats, with special focus on their temporal dimension. Differently from HoneySpider or Shelia, HARMUR is not, per se, a honeyclient, and does not crawl the web searching for vulnerabilities. HARMUR is an aggregator for information generated by other security services, information that is classified by HARMUR into two orthogonal categories:

- Site feeds. Lists of sites that have been judged by a honeyclient as containing malicious or suspicious content.
- Analysis feeds. Information on the current state of each site, on its security state (e.g. type of threats associated to its content) and on its current context (e.g. nature of the web server hosting the content).

HARMUR specifically focuses on the security and network properties of sites rather than malware samples. Still, whenever a malware sample can be mapped back to a set of domains responsible for its propagation, HARMUR can provide information which is instrumental to the definition of the malware propagation context. Throughout the design of HARMUR, a special attention was devoted to modeling the temporal aspect of these threats. Every information stored in the HARMUR dataset is associated to a timestamp, and this allows to study its evolution over time (e.g. the movement of a specific domain among different hosting providers). However, early attempts to automatically inspect threat dynamics have been unsuccessful due to the coarseness of the data at our disposal. While the information can still be valuable for the manual forensic analysis of specific domains, in the context of this deliverable we have decided to "flatten" the definition of the contextual features leaving out the temporal dimension.

Table 7.1 provides a statistical characterization of the HARMUR contextual features initially defined in D15 and finalized in this deliverable. Similarly to what was done for the SGNET data, we report in Table 7.1 information on the statistical distribution of each feature: number of distinct values, minimum, maximum and average frequency, standard deviation of the value frequency, and coverage of the 10 most frequent values. We define as feature value frequency as the number of domains associated to that value.

7.1 Naming information

For each domain name, HARMUR tries to collect 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 known hostnames for such 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).

(F.1) DNS A records. This feature associates each domain tracked by HARMUR to a list of IP addresses. This list is generated through the union of all the DNS A records that have been seen associating a hostname belonging to the domain to an IP address. The list therefore represents the address of all the servers that have ever been associated to the domain, and includes also associations that may be no longer valid. For instance, if the hostname *www.wombat.org* has been associated to the host 1.1.1.1 first, and then moved to a new hosting provider on 2.2.2.2 at a

certain time in the past, this feature will associate the domain *wombat.org* to the list [1.1.1.1, 2.2.2.2].

(F.2) DNS NS records. Similarly to F.1, this feature associates a domain name to all the nameserver addresses that have ever been advertised as authoritative for the given domain.

On top of the basic DNS information, HARMUR takes advantage of the whois protocol to retrieve information on the registration information for each domain name. Such information can be extremely valuable in understanding the modus operandi of the attackers. For instance, a single registrant registered on ONLINENIC 71 distinct domains exactly on the same day. The domain names were the result of the permutation of a few dictionary words associated to 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 AV software.

As already pointed out in D15, 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. We have therefore implemented parsers for the most common registrars, but it has been impossible for us to fully cover this feature based due to the previously explained challenges. We estimate the current coverage to be sufficient to our purposes, and more information can be easily generated by extending the numbers of implemented parsers.

(F.3) Registration date. This feature represents the time in which a given domain has been registered into the registration authority. This information can be particularly useful to detect bulk registrations of a large number of domains.

(F.4) Registrar. This feature associates each domain to the name of the organization or commercial entity through which the domain was reserved.

(F.5) Registrant. This feature associates each domain to the email address that was provided as contact point during the registration of the domain. Nowadays many registrars offer specific services to anonymize this information: if these services are being used, the real registrant email address is replaced by a stub address.

7.2 Server information

HARMUR extracts a variety of different information on each web server known to be hosting one or more of the tracked sites.

(F.6) Location. Similarly to many other WOMBAT datasets, HARMUR tries to take advantage of geolocation databases such as Maxmind to collect information on the geographical location of the servers hosting the tracked content. This feature lists the country of origin of all the servers known to have been hosting content associated to a given domain (building upon F.1).

(F.7) Autonomous system. In parallel to the physical location of the servers, this feature reports the localization of the servers hosting the content of each domain with respect to the autonomous system they are in. As shown by FIRE [12], online criminals have been often masquerdading behind disreputable Internet Service Providers known to have very loose control on the activity of their customers. Collection of AS information in HARMUR is possible thanks to Team Cymru's IP to ASN mapping project[1]. Interestingly, while using this service we have detected a limited number of cases in which the server was mapping a single IP address to multiple AS numbers. To disambiguate such cases, we integrate the Team Cymru's information with that provided by the Route Views project (`routeviews.org`).

(F.8) Server version. HARMUR was designed to be able to scale to an extremely large number of domains, and is therefore unable to perform expensive analyses on the tracked domains. Still, it is possible to retrieve from web servers very valuable information through very simple HTTP interactions. HARMUR probes each server with an HTTP HEAD request for the root page of the site. By parsing the associated answer, we collect 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.

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

8 Dataset: HoneySpider Network (NASK)

8.1 Introduction

The HoneySpider Network (HSN) is a hybrid client honeypot aimed at fast and accurate classification of web-pages. The main goal of HSN is to identify whether the visited web-page is malicious or not in the shortest time possible. For this purpose, the system design employs two types of client honeypots. Low-Interaction client honeypot is used to filter out malicious and suspicious web-pages and is the first line of detection. To confirm its results a High-Interaction client honeypot is used. The layered approach is used to improve the detection rate of the system, as the scope of low and high interaction honeypots is different.

As HSN is concerned primarily with URLs, not malware, we understand context in terms of the circumstances that surround the collection and assessment of a URL and thus determine its role in an event. This context can then contribute to a better understanding of the attack infrastructures and players behind client-side attacks i.e. improving the "threats intelligence" and "attack attribution". We have identified a number of contextual features behind the usage of a URL for malicious infection purposes. Not all of these are collected by HSN, but will be in the future. What follows is a list of features, along with their short explanation, whether they are currently being collected or not, and their importance and usefulness for the tasks of attack attribution.

For the purposes of enriching the contextual data with additional information a correlation software was developed. Its main purpose is to aggregate, store and re-factor data so it will present new and useful information to aid operations of Incident Handling Teams (see Table 8.1). Below is the list of contextual features as described in the Intermediate report expanded with additional information delivered by the correlation software.

8.2 Features identified

URL name The naming scheme behind URLs can uncover possible links between URLs, demonstrating that they can be part of the same attack. TLD usage (e.g. .cn or .com) can also contribute to additional information about an attack. HSN stores all the URL

Table 8.1: Data enrichment produced by the correlation software

Available data	Data enrichment
• Multiple scan results by different instances of HSN	Data gathering to produce one clear image of observed threats available for further analysis.
• URL name, classification and requests generated during scan • IP addresses of the web-servers • Geolocation information	Various relation graphs between compromised, redirecting and malware websites and servers.
• Geolocation information • Fast flux assessment • URL source • URL categorization	Priority assigned to each node on a graph describing the importance of a case. The value is higher for servers in CERT constituency.
• Threat map of analyzed websites	Presented in easy-to-use GUI allowing filtering, restructuring and adding information based on expert knowledge.

names discovered, including those that are generated as requests when visiting the initial parent URL. The URL name gives us insight about the domain and host name used

to lure users in and infect. The domain and host name extracted from the URL by correlation software are used to place it on the graph of redirections between particular domains. This kind of information can help in assessing the severity of a threat carried by a discovered compromised website.

Geolocation information The origin of an attack is a useful piece of contextual meta-data. This includes information like country code (i.e. the country where the malicious web page is hosted), autonomous system number, operator name. HSN collects all this information, not just for the initial URL but for all the further requests. This is done in real-time, by querying the Team Cymru Whois database through DNS requests. The geo-location information together with classification results of URLs determined to be located in a particular region give an overview of current situation and may indicate new threat rising if previously no malicious activity was observed. The correlation software uses this information to build a list of most actively used AS Numbers and their locations.

IP characteristics IPs that the URL resolves to can also provide insight: is there a limited group of IPs being used, are these on a similar subnet or network class, is there overlap between seemingly unrelated URLs in terms of IP usage etc. HSN stores all IPs associated with a URL when the URL is being checked. The IP addresses of the domain names infecting users can be used to cluster the URLs into one single entity representing malicious server. The new map of relations between IP addresses can be more precise and better represent malicious network. The correlation software is able to make such a representation and mark whether the server is just redirecting further or serves malware by itself.

Fast flux assessment Whether a URL is fast-flux or not is a useful piece of information that can show relationships between collected URLs as well as give additional insight into an attack infrastructure. HSN makes such an assessment, based on an algorithm that takes into account the TTL of a domain, amount of A records returned, network distance between the A records, and the existence of certain keywords in reverse DNS records for the A records returned (a sort of an "individual subscriber" count).

Whois information WHOIS database information about a domain name can supply information such as DNS Registrant (ie. the entity that owns the domain), DNS registrar (ie. the entity – organization – through which the domain was registered), date of

registration. Information about the registrants e-mail address is also an example of information that can be gained. Currently, HSN does not collect whois information.

Referer used The referer that was used when visiting a web site is an important piece of contextual information. Many malicious web sites behave differently depending on how they are accessed. For instance, a URL may serve malicious code when accessed from the Google Search results page and differently when accessed directly. HSN stores the referer headers used.

Browser profile Browser headers, such as the User-Agent, Accept, Accept-Language, Accept-Encoding headers etc, and their ordering can also influence the behavior of a URL. Basic browser headers are supported by HSN.

Server type information Information about the HTTP server version behind a URL can also be a useful feature. This information is stored by HSN.

URL source URLs can be obtained in different ways. They can be collected from spamtraps (e-mail, instant messenger), dynamically created from Google queries, obtained from proxy logs, or reported by some external party. How they are obtained can also give insight into how a malware group searches for possible targets. It is not only the source type that is of interest, but the specific source as well. This may help to infer targeted attacks. Information relating to the origin of URL submission is stored in HSN.

URL categorization Initial URLs that are obtained for checking are most often content serving URLs. They normally do not serve exploits directly or serve malware. Usually, they have iframes injected into them that serve as redirects to the actual exploit or malware download site. There can be multiple redirect, exploit and malware download URLs involved, forming an attack tree. Recreating such attack trees of URLs and finding relationships between them can be very useful in identifying different groups of attackers. The correlation software was built to serve this and other purposes. It tries to determine whether the URL acts as a intermediate node in path of redirections or is the malware-serving website. Then the URL is placed on a graph of redirections forming a tree of relations. The information obtained from such classification can help in taking proper steps towards mitigating the threat.

Exploits identified Exploit information about a URL can be useful in understanding an attack infrastructure. This can be information relating not only to a specific exploit,

but to a more generic identifier, such as whether the URL is a drive by download or not. HSN does not currently provide information about the exact exploits being used, but will do so in future versions.

Malware collected Malware collected from a URL can also give an insight about the relationships between sites. HSN can collect malware from URLs. However, this is not the main area of operation of the system. Malware is stored by the system along with MD5/SHA1 hashes, which can be used to further identify the roles and relationships of URLs being investigated.

URL neighborhood assessment An URL neighborhood can be assessed by correlating information about the URL placement in the relation's graph. Three types of neighborhood were defined: a set of URLs belonging to the same domain also classified as malicious but not necessarily serving malware; a set of URLs redirecting to the same malicious site from many different servers viewed globally; a set of URLs that were redirecting to our particular URL also from the global perspective. Having the possibility to extract such types of information gives a way to asses the scale of observed threat and devise proper means to mitigate it or limit its impact.

URL priority marking The correlation software tries to prioritize the URLs based on the information gathered from different scans and in the future from different sources. The priority is assigned based on the constituency of the scanned URL, the number of occurrences as malicious in different scans from different IPs and the type assigned to URL (compromised site, redirection or malicious site). The priority should mark an URL as possibly the easiest to deal with regarding to the effort needed to stop the threat. For example value will be higher when malicious website resides in the CERT's constituency so time needed to take it down will be considerably shorter than dealing with website located in different country.

URL labeling URLs and trees of redirects can be marked with a label which correspond to a short description representing some characteristics of the threat. For example when a SPAM campaign spreading malicious URLs is launched all found URLs can be marked with the campaign name and later with the name of a group responsible for it. Also the type of malware used and attacked resources (e.g. banking software) can be one of the labels. This will lead to building a repository of observed threats which in later research will prove invaluable in improving the correlation software and using it in even more automated manner.

Statistical data The correlation software provides a set of statistical data useful for assessing the current state of the threats based on scanned URLs. The statistics are: top sources of infections based on Country Codes, Autonomous Systems identifying ISPs, versions of HTTP servers and the most active malicious domains. Those statistics are created in two basic contexts: global and constituency, global being the overall view of monitored threats and constituency defined as the country code and AS numbers the IRT operates within.

In summary, there is a wealth of contextual information that can be obtained about a URL and its nature. Most of the above features are currently part of the HSN dataset. Key to the dataset is the "path of exploitation" from the compromised server (ie. the content site or the landing site) to the one serving malware. Building those types of relation-graphs is the goal of the threat intelligence WP of the WOMBAT project. The goal, from the HSN perspective, is to discover global network of relations which will allow for the recognition of the servers of main interest to the CERT teams. Those will not just be exploit servers or malware-download servers but also servers which only redirect further and are located in the CERT's constituency.

8.3 The correlation software.

The main purpose of creating the software was to aid operations of the IRT at CERT Polska by reprocessing information gathered by the HSN project. Many instances of HSN produce a vast amount of information which cannot be easily cross-checked. The software tries to do it for the user and present the processed data in the most suitable form allowing to quickly launch an investigation if the threat is serious.

The core element of the application is the correlating module. It connects to various instances of the HSN and monitors for new data as the classification process of an URL finishes. The gathered data is then restructured to better fit the application needs and stored in a separate database. The occurrences of the same URL are compared and new information is added if available (see Figure 8.1). Moreover a map of redirections between malicious sites is compiled based on information delivered by the Low-Interaction Component of the HSN project. This map allows to quickly asses the severity of a threat and to have a better understanding of its mechanisms. Based on information like place of origin (e.g. the AS number), the source of URL (e.g. spam, proxy logs, incident report) and number of occurrences in unrelated scans, a priority is assigned to mark the importance of a case. The software tries to differentiate observed URLs into three groups: compromised sites, redirections and malicious servers. Usually those three groups of URLs are independent from each other in terms of location and

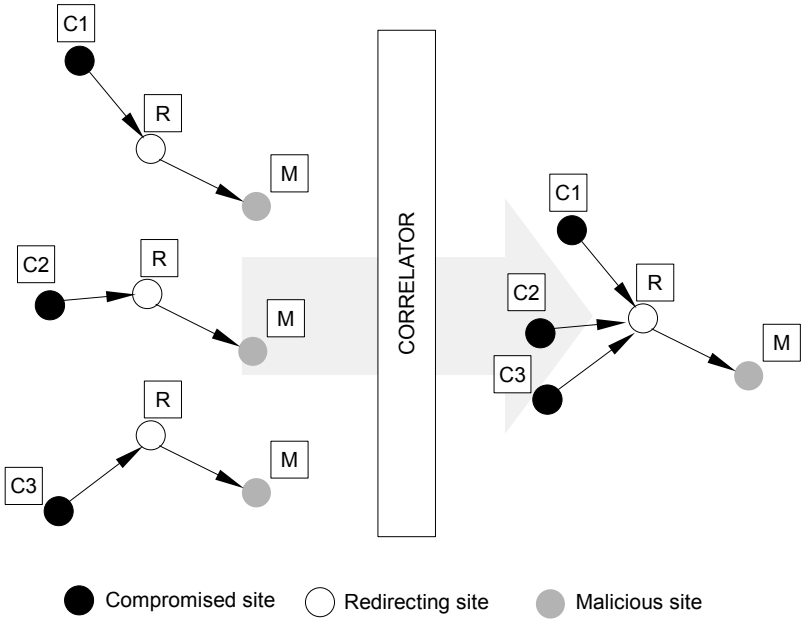

Figure 8.1: Scan correlation as performed by the correlation software

purpose. URLs located within the IRT constituency are marked with higher priority, to be taken care of as soon as possible.

The information produced by the correlating software would be useless if it could not be presented in proper way to an operator. This purpose is served by the GUI module. It's main functionality is to present the map of relations between compromised sites, the redirecting sites and the malware servers. Also, it allows user interaction with the map by giving the possibility of assigning labels to URLs and redirection trees. If some additional information is available for the case, e.g. the type of exploit used, the user is allowed to input such and store it in the database. The information about each URL is displayed in the side panel. The GUI allows to search through the map for a specific URL, domain or IP and to change the view of the map, e.g. the URLs view can change to IPs view representing redirections between servers. The application is written in Java and independent of the underlaying operating system.

9 Dataset: Bluebat (Polimi)

BlueBat is an ad hoc device based on the GNU/Linux OS, which is primarily designed to collect samples of the latter type of malware, using a set of software based on the pybluez framework [2].

We remind to the reader that BlueBat is an experimental Bluetooth honeypot sensor, which we described in earlier WOMBAT deliverables D06 and D13, as well as in a publication [8]. As discussed in previous deliverable D06, Bluetooth exhibits a number of security issues in various specific implementations of the stack. Such attacks are well known, and allow different degrees of data access, communication interception, up to and including running any AT command taking full control of the phone. However, viruses for mobile device primarily rely on simple *social engineering* to propagate, sending copies of themselves to any device which comes into range through an OBEX push connection.

Given this specific setting, contextual features are both peculiar and limited.

9.1 Features collected

Basic features Basically, the entries logged by a BlueBat sensor contain a description of the interaction with the peer, the address of the peer, and if the transaction pushed a file (likely, a malware file), the file itself and its MD5 hash. The latter feature can be used to cross-search for the same binary on other datasets that may likely contain it (specifically, Shelia and Virustotal).

The address of the peer and the filename of the pushed file are simple to gather and they can be considered contextual features. Their usefulness is similar to what happens in TCP/IP based exploit collection, so we will not need to elaborate on this, except to note that while an IP address is logical, the MAC address of a Bluetooth peer is a physical and unique identifier of a device. Also, since most of the other datasets use IP addresses as contextual information, this information cannot be effectively cross-linked.

Geolocation Things become more interesting when considering geolocation, as this is an extremely important parameter in our collection setting. Geolocation for Bluetooth devices does not need to be inferred by databases (as it happens for IP addresses), but is directly inferred by their proximity to a specific honeypot. Honeypots in our current

deployments are static, so the collected data can be very simply geographically tagged. For mobile honeypots, we have developed a connection with the gpsd [1] GPS daemon for tagging with location data.

Blueprinting In parallel to the data collection, we perform a continuous scanning for devices, and we fingerprint the ones we find, using the process called "blueprinting", which is described in [6]. Basically, and similarly to what happens with hosts on the Internet, by mapping the different services exposed by a device and several Bluetooth stack parameters, it is sometimes possible to determine remotely the type of hardware and software of the device.

This is actually difficult to do in extremely open settings, as device scanning is very slow. So even if we use extremely powerful antennas for running the OBEX server, we need to scale down to less powerful ones for additional scanning. This means that we will not always be able to have the fingerprinting results for a given device. Additionally, Blueprinting has less and less reliable databases as the number of variations of hardware and software on Bluetooth enabled devices grows, and this is a shortcoming that cannot be addressed.

10 Conclusion

The objective of Workpackage 4 is to develop techniques to characterize the malicious code that is collected in the previous workpackage. The main idea is to enrich the collected code thanks to metadata that might reveal insights into the origin of the code and the intentions of those that created, released or used it.

This deliverable discusses the contextual features of malware and malware-based threats in the datasets that are maintained by the WOMBAT partners. The document contains a description of each feature, and some statistics about the values observed in the collected data.

Bibliography

[1] Gpsd website. `http://gpsd.berlios.de/`.

[2] Pybluez website. `http://org.csail.mit.edu/pybluez/`.

[3] Smashing the Mega-d/Ozdok botnet in 24 hours. `http://blog.fireeye.com/research/2009/11/smashing-the-ozdok.html`, 2009.

[4] A study of the pushdo/cutwail botnet. `http://us.trendmicro.com/imperia/md/content/us/pdf/threats/securitylibrary/study_of_pushdo.pdf`, 2009.

[5] Adblock list. `http://www.fanboy.co.nz`, September 2010.

[6] L. Carettoni, C. Merloni, and S. Zanero. Studying bluetooth malware propagation: The bluebag project. *Security & Privacy, IEEE*, 5(2):17–25, March-April 2007.

[7] A. Decker, D. Sancho, L. Kharouni, M. Goncharov, and R. McArdle. A study of the Pushdo / Cutwail Botnet. http://us.trendmicro.com/imperia/md/content/us/pdf/threats/securitylibrary/study_of_pushdo.pdf, 2009.

[8] A. Galante, A. Kokos, and S. Zanero. Bluebat: Towards practical bluetooth honeypots. In *Communications, 2009. ICC '09. IEEE International Conference on*, pages 1–6, June 2009.

[9] B. Krebs. Takedowns: The shuns and stuns that take the fight to the enemy. *McAfee Security Jouranl*, (6), 2010.

[10] V. Paxson. Bro: a system for detecting network intruders in real-time. In *USENIX Security Symposium*, 1998.

[11] D. Sancho. You Scratch My BackBREDOLABs Sudden Rise in Prominence. http://us.trendmicro.com/imperia/md/content/us/trendwatch/researchandanalysis/bredolab_final.pdf, 2009.

[12] 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.