Chapter 5
Assessing Cybercrime Through the Eyes of the WOMBAT

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

5.1 Foreword

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

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

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

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

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

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

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

5.2 Introduction

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

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

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

5.3 Leurre.com v1.0 Honeyd

5.3.1 Historical background

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

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

5.3.2 Some technical aspects

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

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

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

Data uploading mechanism: Just after the data retrieval, the log files are then uploaded into a large Oracle database by a set of Perl programs. These programs take tcpdump files as input and parse them in order to create different abstraction levels. The lowest one corresponds to the raw tcpdump traffic. The higher level is built on the lower ones and has richer semantics. Due to space constraints, we do not present here all the concepts, but instead we will focus only on the most important notions.

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

Information enrichment

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

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

5.3.3 Generic picture

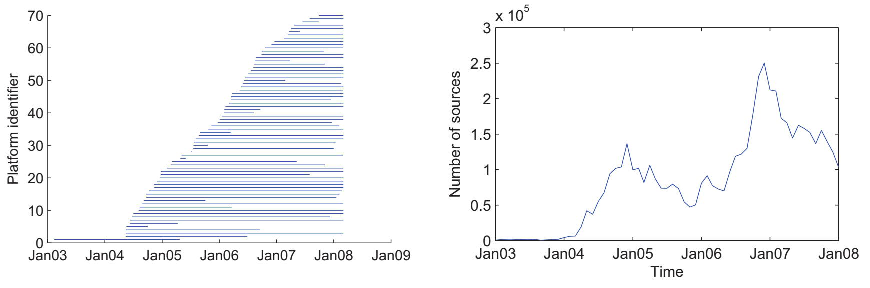


Fig. 5.1 Left: Evolution of platforms, Right: number of sources

Figure 5.1 (left) shows the evolution of platforms. Each curve corresponds to the time life of a platform. As we can see, we started our data collection in January 2003 with one VMware honeypot and we have started to deploy the distributed low interaction honeypots in April 2004. Since then, the number of partners joining us has kept increasing. In total, we have around 50 offical partners and around 20 former partners. These platforms have, in total, covered 37 different /8 networks, locating in 28 different countries in five continents. In total, we have observed 5173624 sources corresponding to 3461745 different IP addresses. Figure 5.1 (right) shows the evolution of the number of sources over time. The variation of the curve is of course influenced by the number of platforms. Note that up to April 2004, the traffic is negligible. After that, the number of sources has increased. It is interesting to observe that the number of sources on the last six months of 2004 is much higher than that of the last six months of 2005 even through, in the second case, we have more platforms. In total, there are 155041 different clusters. Figure 5.2 (left) represents the cumulative distribution function of number of sources per number of cluster. Point (X,Y) on the curve means that Y*100% of the total amount of clusters contain less than X sources. As we can see, most of clusters are very small. There are, in fact, only 15521 clusters containing more than 10 sources each. Interestingly enough, by querying the database one can find that these clusters, ie. around 10% of the total number of clusters, contain in fact 95% of the observed attacks! In other words, the bulk of the attacks is found in a limited number of clusters whereas a very large number of diverse activities originate from a very limited number of sources. In term of attacking machines' OS, according to p0f, almost all attacking machines are Windows ones. This confirms again the results in [11, 12]. Figure 5.3 shows the top ten attacking countries with US in the head, followed by China and Canada. But the surprising thing is that CS (corresponding to former Serbia and Montenegro) is at the fifth position. The reason is that there is one (and only one!) platform which is heavily attacked by this country. In total, it shows up as one of the most attacking countries. Finally, as an example to show the diversity of the attacks over different platforms, Figure 5.2 (right) shows the distribution of the number of different clusters per platform. Each column represents the number of distinct clusters observed on a platform. We have as many columns as number of platforms. As we can see, the attacks are highly diverse. On some platforms, we observe just small number of clusters, but it is not the case for others.

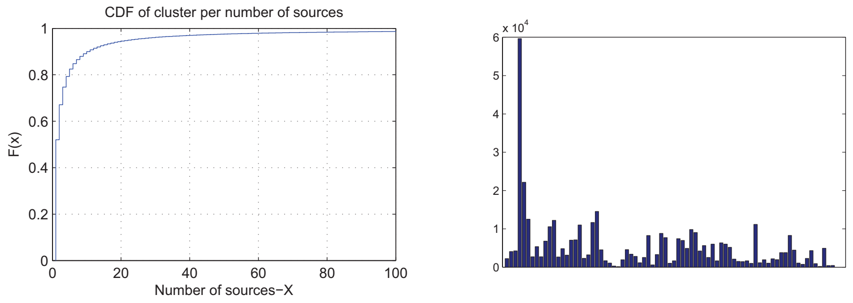


Fig. 5.2 Left:Cumulative distribution function of number of source per cluster; Right:Distribution of number of clusters per platform.

5.3.4 *Some illustrative examples*

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

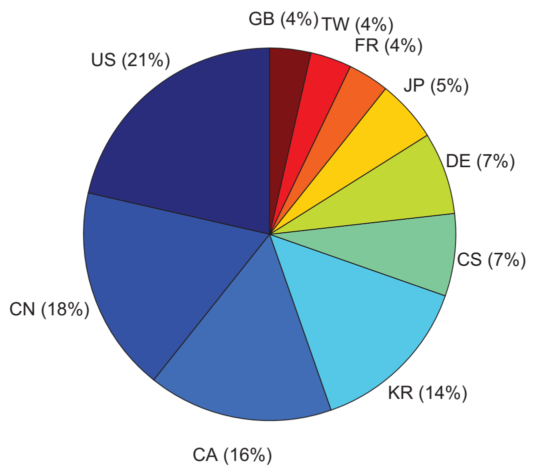


Fig. 5.3 Top ten attacking countries

5.3.4.1 Temporal Evolution of Attack Clusters

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

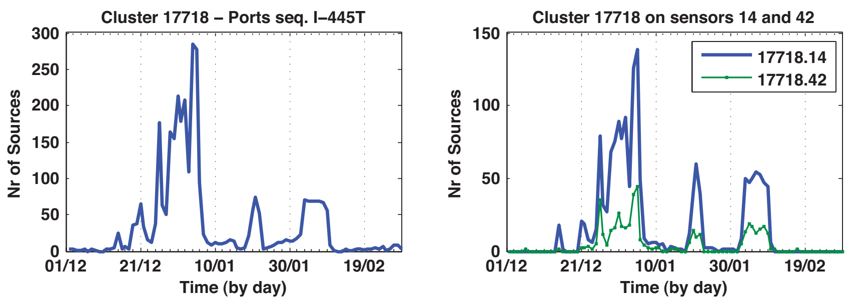


Fig. 5.4 Left: global time evolution of attack cluster 17718, with the sources aggregated by day. Right: time evolution of the same attack cluster for the platforms 14 and 42 separately.

5.3.4.2 Geographical Location of Attackers

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

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

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

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

5.3.4.3 Attackers Subnets Information

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

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

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

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

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

5.3.4.4 Targeted Platforms or Subnets

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

Table 5.3 Distribution of targeted platforms for attack cluster 17718

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

5.4 Leurre.com v2.0: SGNET

5.4.1 Increasing the level of interaction

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

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

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

5.4.2 ScriptGen

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

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

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

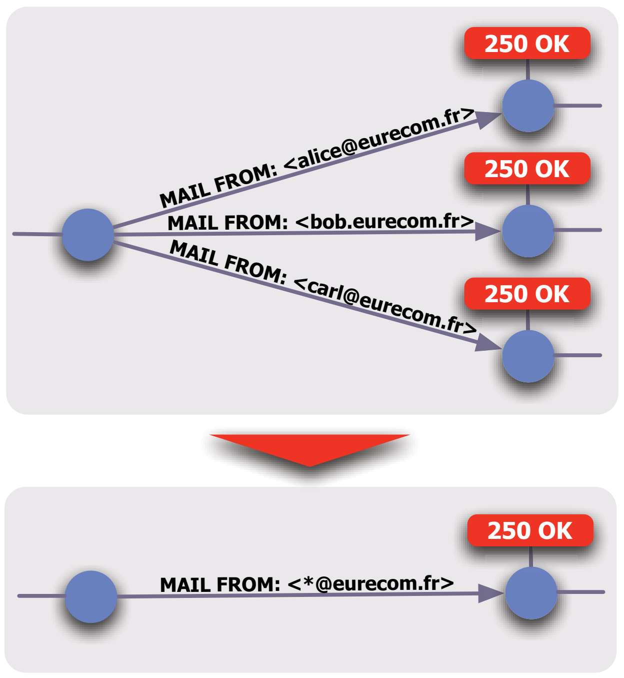

250 OK
MAIL FROM: <alice@eurecom.fr>
250 OK
MAIL FROM: <bob.eurecom.fr>
MAIL FROM: <carl@eurecom.fr>
250 OK
250 OK
MAIL FROM: <*@eurecom.fr>

Fig. 5.5 ScriptGen FSM generalization

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

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

5.4.3 SGNET: a ScriptGen-based honeypot deployment

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

SGNET and code injections. SGNET is a scalable framework that offers almost the same amount of information than real high interaction systems for a specific class of attacks, namely server-based code injection attacks generated by deterministic scripts. We are aware of the fact that they correspond only to a subset of the possible attack scenarios. However, as of today, they are considered to be responsible for the creation of large botnets [32] and the preferred propagation mechanisms of a large number of different malware.

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

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

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

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

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

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

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

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

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

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

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

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

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

SGNET Architecture.

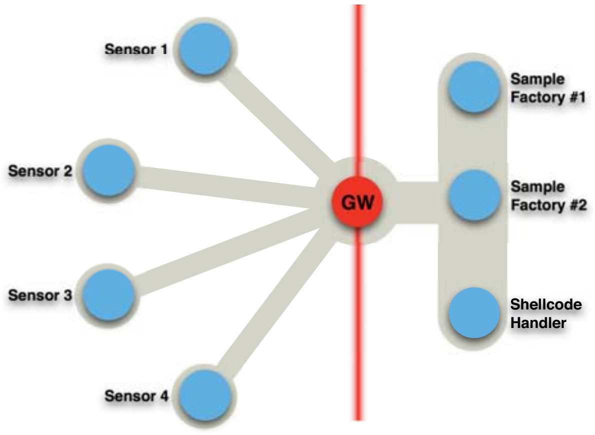


Fig. 5.6 SGNET architecture

The general architecture of the SGNET is presented in Figure 5.6. All the SGNET entities communicate through an ad-hoc HTTP like protocol called Peiros [18]. The Peiros protocol allows communication under the form of a set of service requests, allowing for instance a sensor to require the instantiation of a sample factory. The sensors, distributed over the IP space and hosted by partners of the project, are connected to a central entity called SGNET gateway, that acts as an application-level proxy for the Peiros protocol. The gateway receives service requests from the sensors and dispatches them to a free internal entity, performing a very simple load balancing. The architecture offers a clean separation between the sensors, relatively simple daemons running over inexpensive hosts, and the internal entities, having a higher complexity and higher resource requirement.

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

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

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

5.5 Analysis of Attack Events

5.5.1 Identification of Attack Events

5.5.1.1 Attack Event Definition

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

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

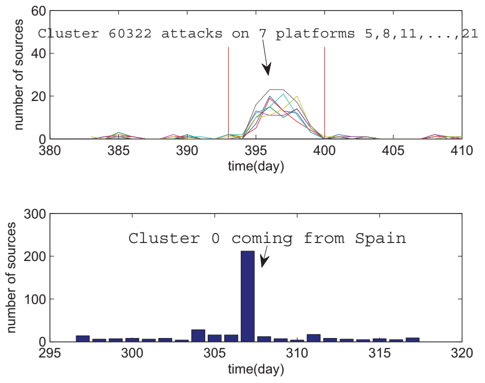


Fig. 5.7 On the top plot, cluster 60232 attacks seven platforms from day 393 to day 400. On the bottom plot, peak of activities of cluster 0 from Spain on day 307.

5.5.1.2 Dataset Description

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

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

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

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

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

5.5.1.3 Results on Attack Event Detection

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

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

Table 5.5 Result on Attack Event Detection

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

No.AEs: amount of attack events

5.5.2 *Armies of Zombies*

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

5.5.2.1 Identification of the armies

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

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

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

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

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

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

5.5.2.2 Main Characteristics of the Zombie armies

Lifetime of Zombie Army.

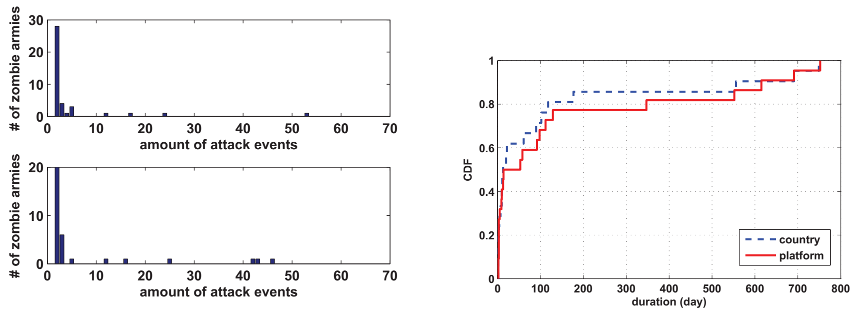


Fig. 5.8 Left: Zombie Army Size. Right: Cumulative Distribution Function (CDF) of the durations of zombie armies.

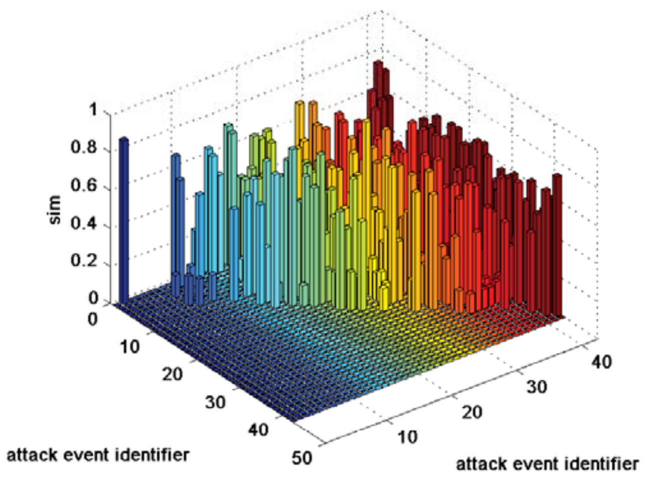
Figure 5.8 (Right) represents the cumulative distribution of minimum lifetime of zombie armies obtained from $TS_{platform}$ and $TS_{country}$ (see Section 5.5.1). According to the plot, around 20% of zombie armies have existed for more than 200 days. In the extreme case, two armies seems to have survived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

Lifetime of Infected Host in Zombie Armies

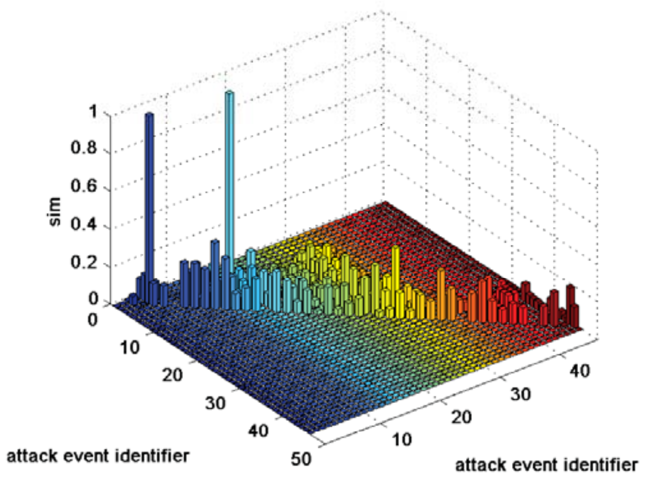
We can classify the armies into two classes as mentioned in the previous Section. For instance, Figure 5.9a represents the similarity matrix of zombie army 33, ZA33. To build this matrix, we first order its 42 attack events according to their occurred time. Then, we represent their similarity relation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

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

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



(a)



(b)

Fig. 5.9 Renewal rate of zombie armies

5.5.3 Impact of Observation View Point

5.5.3.1 Analysis

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

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

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

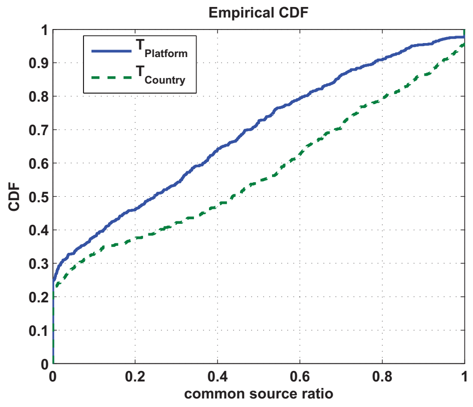


Fig. 5.10 CDF common source ratio.

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

5.5.3.2 Explanation

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

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

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

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

5.6 Multi-Dimensional Analysis of Attack Events

5.6.1 Methodology

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

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

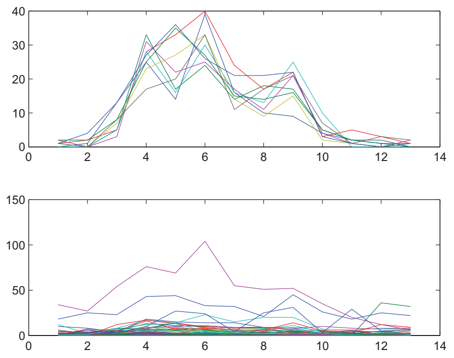


Fig. 5.11 Top plot represents the attack event 79 related to cluster 17309 on 9 platforms. The bottom plot represents the evolution of this cluster by country. Noise of the attacks to other platforms decrease significantly the correlation of observed cluster time series when split by country.

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

5.6.2 Clique-based Clustering

5.6.2.1 Principles

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

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

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

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

Finally, we take advantage of those similarity measures to group all pattack events whose patterns look very similar. We simply use an unsupervised graph-theoretic approach to formulate the problem: the vertices of the graph represent the patterns (or feature vectors) of each attack event, and the edges (or the arcs) express the similarity relationships between those vertices. Then, the clustering is performed by extracting so-called *maximal weighted cliques* (MWC) from the graph, where a maximal *clique* is defined as an induced sub-graph in which the vertices are fully connected and it is not contained within any other clique. Since it is a NP-hard problem [4], several approximate algorithms for solving the MWC problem have been developed. We refer the interested reader to [38, 39] for a more detailed description of this clique-based clustering technique applied to honeynet traces.

5.6.2.2 Some Experimental Results.

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

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

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

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

5.6.2.3 Visualizing Cliques of Attackers.

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

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

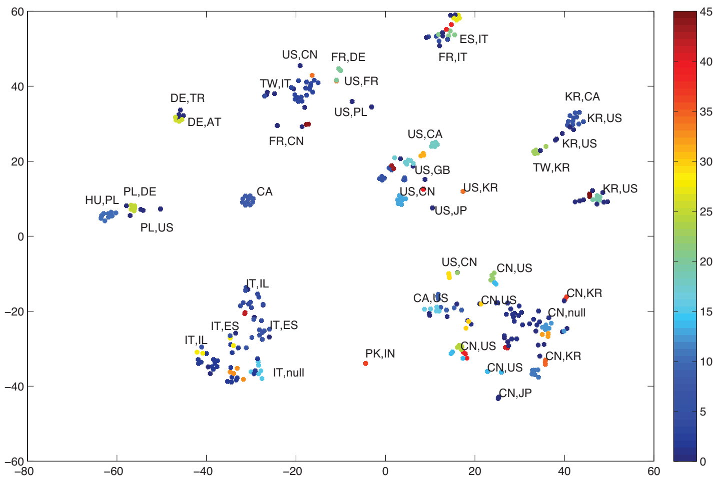


Fig. 5.12 Visualization of geographical cliques of attackers. The coloring refers to the different clique Id's. The superposed text labels show the two top attacking countries for some of the data points.

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

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

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

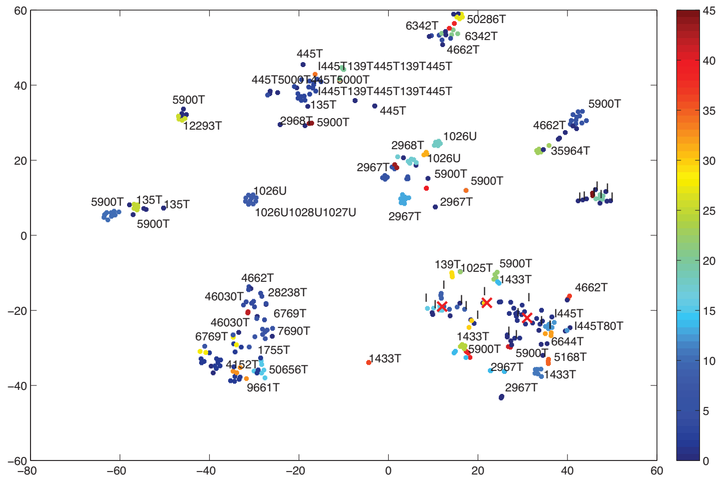


Fig. 5.13 Same visualization of the geographical cliques of attackers as Fig 5.12, but here the superposed text labels indicate the port sequences made by the attackers.

5.6.3 Combining Cliques of Attackers

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

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

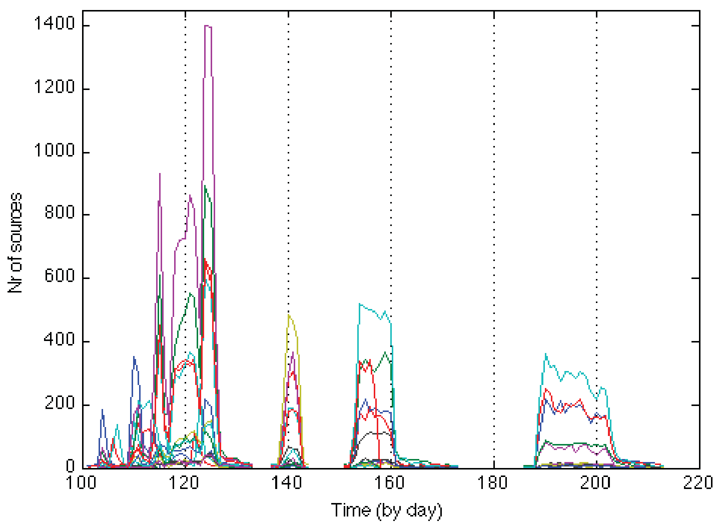


Fig. 5.14 Time series (i.e., nr of distinct sources by day) of a large scale phenomenon related to a botnet activity, made of 67 attack events observed from Dec 06 until April 07.

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

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

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

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

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

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

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

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

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

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

Table 5.7 Information taken into account

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

5.7.1 Degrees of freedom

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

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

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

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

5.7.2 Interesting cases

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

5.7.2.1 ASN.1 vulnerability

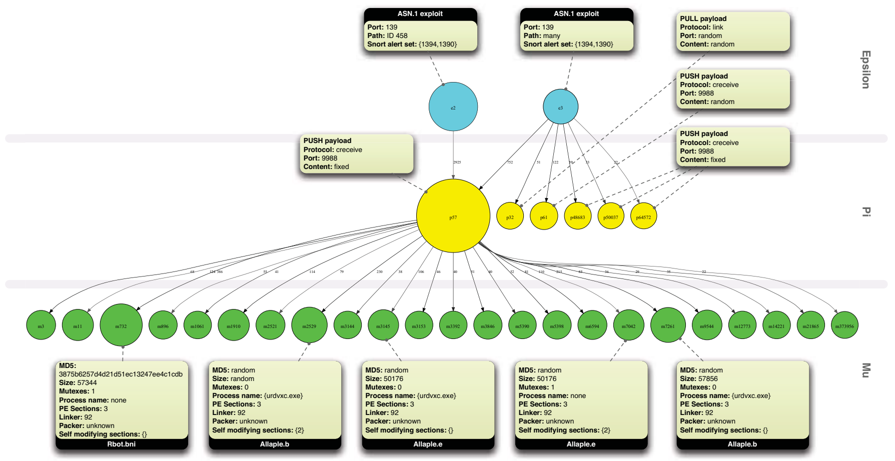


Fig. 5.15 The ASN.1 exploit (port 139)

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

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

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

5.7.2.2 Rbot.bni malware family

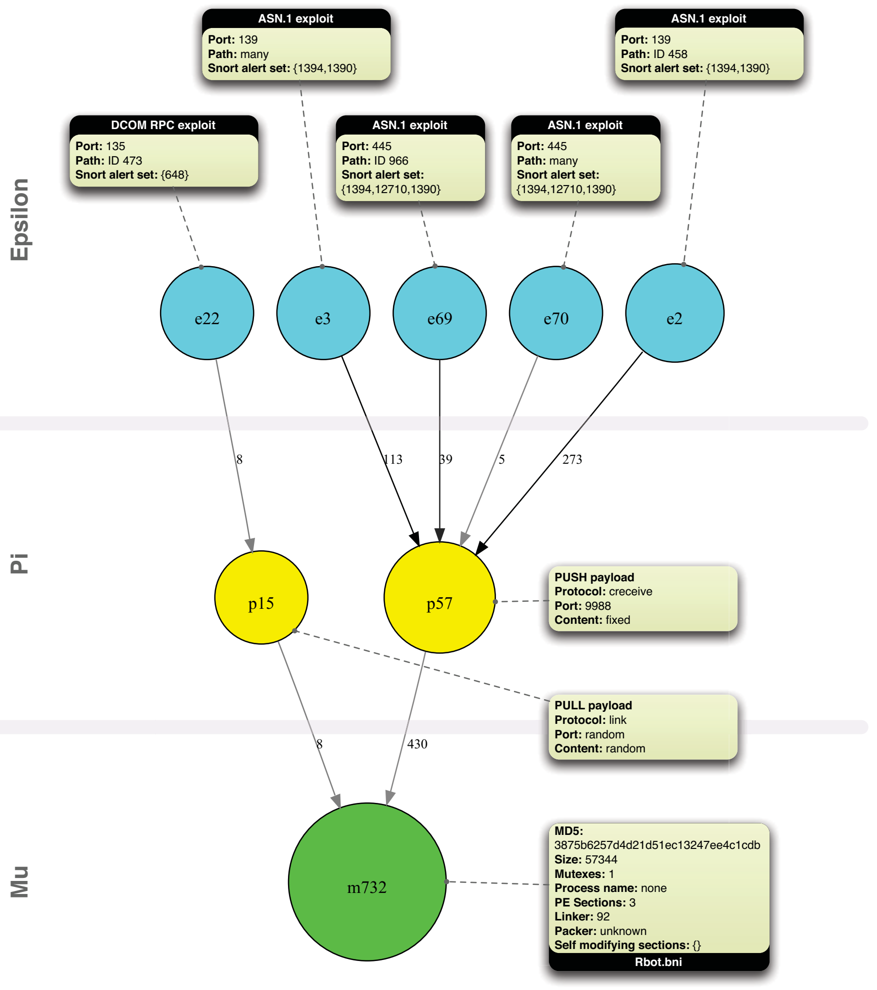


Fig. 5.16 Propagation ability for mu group m732

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

5.7.2.3 DCOM RPC vulnerability

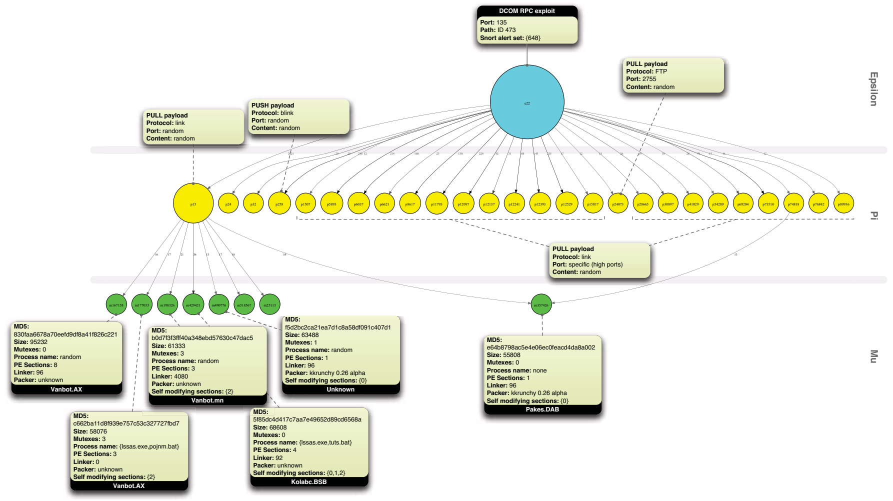


Fig. 5.17 The DCOM RPC exploit

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

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

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

5.8 Conclusions

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

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

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

References

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

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

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