SweetBait: Zero-Hour Worm Detection and Containment Using Honeypots

Georgios Portokalidis and Herbert Bos

Vrije Universiteit, Amsterdam, The Netherlands,
{porto,hjb}@few.vu.nl,
WWW home page: http://www.cs.vu.nl/~herbertb

Abstract. As next-generation computer worms may spread within minutes to million of hosts, protection via human intervention is no longer an option. We discuss the implementation of *SweetBait*, an automated protection system that employs low-interaction honeypots to capture suspicious traffic. After discarding whitelisted patterns, it automatically generates worm signatures. To provide a low response time, the signatures may be immediately distributed to network intrusion detection and prevention systems. At the same time the signatures are continuously refined for increased accuracy and lower false identification rates. By monitoring signature activity and predicting ascending or descending trends in worm virulence, we are able to sort signatures in order of urgency. As a result, the set of signatures to be monitored or filtered is managed in such a way that new and very active worms are always included in the set, while the size of the set is bounded. *SweetBait* is deployed on medium sized academic networks across the world and is able to react to zero-day worms within minutes. Furthermore, we demonstrate how globally sharing signatures can help immunise parts of the Internet.

1 Introduction

As new breeds of worms are expected to spread to millions of hosts in minutes, if not seconds, it is imperative to automate both outbreak detection and response [1, 2]. Worse, in order to be effective the automated system should take appropriate counter measures in a fraction of the time that it takes the worm to spread. Previous attempts to develop such detection systems have built on flow-based anomaly detection, honeypots, and end-host detection [3, 4, 5]. Several projects have addressed the problem of automatic signature detection [**?**, 6]. Unfortunately, most existing approaches exhibit one or more of the following problems:

1. *False positives.* For any automated response system holds that misclassifying and blocking *bona fide* traffic may result in unleashing a denial of service attack by the defence mechanism.
2. *Individuals rather than families.* Most existing systems extract the signature of an individual worm with no attempt to check whether this is a variation of a worm that was previously detected.

3. *Presence rather than virulence.* Anyone brave enough to connect an unprotected machine to the Internet will soon discover that there are many different worms out there. In addition to an exhaustive list of what worms have been encountered in various places, a security system would benefit from a information about the worm activity level. Virulent worms may require more drastic and immediate measures than worms that spread slowly.

In *SweetBait* we address the problem of fast-spreading worms by means of honeypots. The system also detects worms that spread more slowly and even other forms of malware, but these are not its focus. We discuss the design and implementation of a system that aims to protect small and medium sized networks from random IP scanning worms. The size of the networks in focus was motivated by a desire to avoid performance issues that arise with systems on backbone links. Our goal is to automate the procedures of: worm signatures generation, and signature distribution to external NID and NIP systems. At the same time we aim to achieve a low reaction time to new outbreaks. The challenging task of identifying new worms is performed by honeypots. For reasons that will be made clear shortly, we emphasise that honeypots have the unique property that almost all traffic directed at the honeypot may be considered suspect. In *SweetBait* the honeypots along with the NID and NIP systems (NIDS and NIPS) will be managed by a *control centre* (CC), which will be able to respond to outbreaks even when untended. Some contributions of this paper are summarised below:

1. reduce false positives by 'whitelisting' benevolent traffic;
2. continuously refine worm signatures to provide automated signature revision;
3. utilise both NIDS and NIPS for detection and containment respectively;
4. predict worm aggressiveness by monitoring a worm's activity level;
5. distribute signatures through a *global control centre* (GCC) to all instances of the system to achieve possible immunisation of parts of the Internet;

We chose honeypots rather than network taps for several reasons. First, network administrators feel more comfortable with handing out chunks of unused IP address space than with systems that snoop on user traffic. Second, while it is true that honeypots by themselves never see hit-list worms, this is technically fairly simple to remedy by directing suspicious traffic to the honeypot. For instance, we are currently working on a NID system that uses anomaly detection as a first, and honeypots as a second tier in the detection process: whenever unusual behaviour is detected, the corresponding traffic is forwarded to honeypots for further analysis. In this way, we preserve the property that all traffic arriving at the honeypot is suspect. This is in contrast to approaches that protect against attacks at the user's machine, which makes the task of separating wanted from unwanted traffic more difficult [5, **?**]. A third consideration may be that random IP scanning worms have been much more popular than hit-list worms and scanning worms are responsible for the fastest spreading behaviour to date. However, we consider this argument much less important as this situation may (and probably will) change when more effective measure against scanning worms are in place.

The remainder of this paper is organised as follows. In Sect. 2 we will give a detailed description of the system's architecture and outline our implementation. We evaluate the system in Sect. 4. Related work is discussed throughout the text and also in Section 5. Finally, we will present our conclusions and future work in Sect. 6.

2 System Overview

SweetBait is comprised of multiple components with distinct roles, which can be roughly classified into two categories: sensors and control elements. Honeypots, intrusion detection and prevention systems are all sensors, while *control centres* and a *global control centre* constitute the control components. The honeypots are set up to receive data destined to nonexistent IP addresses of the corresponding subnet. These data are primarily filtered to exclude any known benevolent traffic patterns, and the remainder is treated as of malicious origin and is processed to generate NID signatures that we claim to belong to malware. The generated signatures are then posted to the CC, where they are compared with the ones already known. Based on the incidence reports from multiple locations, the CC decides which signatures to transmit to the NID and NIP components. NID and NIP sensors return feedback to the CC concerning the number of hits for the signatures they have been monitoring or filtering. Finally, the CC is responsible for exchanging signatures and activity statistics with a GCC. The presence of a GCC enables cooperation of instances of SweetBait globally, which is necessary to achieve worm containment [1].

A typical configuration of SweetBait components is shown in Fig. 1. In the remainder of this section we will describe each component in detail.

2.1 Honeypot sensors

Honeypots are powerful devices for capturing random IP scanning worms. These worms discover new victims to attack by randomly generating IP addresses. The IP address of a honeypot is by nature unadvertised on the Internet, and as such it does not exchange any legitimate data with the rest of the network. It is therefore logical to assume that all traffic destined to it is suspect. By populating the unused IP address space of a network with honeypots, there is a high probability of finding a scanning worm in its early stages.

Deploying a honeypot presents us with two possibilities. The first is to sacrifice a real host running *real services* (high-interaction), while the second is to *simulate services* and/or hosts (low-interaction). The former offers high-interactivity with attackers and makes the honeypot almost indistinguishable from other hosts, but it requires additional protection mechanisms such as sandboxing [7, 8] Since no real services are run, the latter offers a lower level of interaction, but requires less maintenance and supplies a greater degree of security [9, 10]. Also,

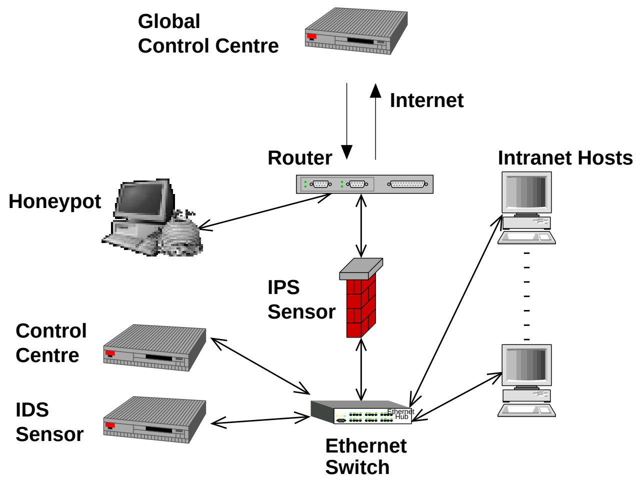

Fig. 1. SweetBait overview

it was shown in NoSEBrEaK that high-interaction honeypot can often be fairly easily discovered as such by intruders [11].

SweetBait currently uses low-interaction honeypots, mainly because of the low maintenance requirements, and security considerations. The honeypot is able to simulate multiple hosts, which allows us to populate unused IP address space easily and maximise the amount of captured traffic. Our choice is also justified by the fact random IP scanning worms rarely interact with a host before attacking it. While we have focused on low interaction, there is nothing in the architecture that precludes connecting high-interaction honeypots.

Even though we consider all traffic received by the honeypot suspect, in practice a small amount of non-worm related, or even legitimate traffic is also captured. Broadcast messages, or attempts to scan the network are both examples of traffic that may be ignored. To tackle this we introduce the notion of a *whitelist*: a list consisting of patterns that are considered benevolent, or inapplicable for generating NID signatures. A filter placed at the honeypot rejects all traffic matching a whitelisted pattern. The filter can be largely auto-generated by training the system on a network when it is unconnected to the larger Internet. We will show that as a result of this simple step the number of false positives drops dramatically.

Consequently, the sensors process the remainder of incoming data, scanning for repeated byte sequences in an attempt to identify worm propagation and generate a signature. For the scanning process to be effective it is necessary to utilise stream reconstruction for sequenced, connection based protocols such as TCP, since the underlying IP layer may deliver packets out of order impeding our capability to identify patterns spanning multiple packets.

Finally, we transmit the generated signatures from the sensor to the *control centre*, where subsequent actions are taken. The transmission is done in a secure manner that guarantees the authentication of both parts, as well as the integrity of the transmitted data.

2.2 Network Intrusion Detection and Prevention Sensors

We use network intrusion detection sensors to passively monitor ingress and egress traffic for worms, based on the signatures generated by the honeypots. Whenever a signature is matched, the NID sensor reports it by issuing an alert that also includes the IP addresses involved in the transaction. This information besides allowing us to quantify a worm's activity it also enables us to:

- populate an Internet map with infected IP addresses;
- block infected remote hosts from accessing our network;
- identify infected hosts in our network, and initiate immunisation procedures.

NIP sensors, besides monitoring and reporting worms by issuing alerts, assume the active role of filtering ingress and egress traffic. If initiated before any host in the network has been compromised, blocking worms from entering the network will lead to immunisation. On the other hand, obstructing worms from leaving the network is tantamount to 'team play' on our part that helps contain the worm, and earns time for other networks to raise their defences.

Most NID and NIP systems today are manually updated each time a new worm appears, while alert reports are being used for purely historical purposes. The SweetBait sensors are in constant communication with the *control centre*, which are responsible for updating the sensors with the signatures that need to be monitored or filtered respectively. Additionally, we immediately post the alerts generated by the sensors to the CC for storage, as well as for estimating worm aggressiveness. Again, the communication channel is secure to ensure only authorised access to the CC.

2.3 Control Centre

If honeypots, network intrusion detection and prevention systems are the arms and legs of SweetBait, the control centre is the brain. The CC collects information from the sensors, processes it and instructs the sensors on future actions. Information exchange between the CC and connected sensors is performed based on a well-defined protocol, decoupling the exchanged data from specific sensor types. The CC gathers two types of incidence reports: network intrusion signatures and alerts.

Signatures are compared with every known signature to detect overlaps. When significantly sized overlaps exist, the received signature is considered to be a *specialisation* of the stored one, and it is stored as a new version. Besides the actual network intrusion signature, the stored data consist of accompanying meta-information. Such data include the timestamp to indicate the time of generation, the source of the signature (e.g., the IP of the honeypot), and various flags to indicate whether it has been verified by an expert to be a valid or a false signature. The latter is essential information that permits humans to affect the decision making process that is described later on to increase the effectiveness of the system.

The CC also collects alerts generated by NID and NIP components when network traffic is found to match one of the known signatures. Each alert contains information that identifies the signature, as well as the source and destination IP addresses involved in the data exchange. The information is stored in a database both for auditing and the reasons mentioned in Sect. 2.2. Furthermore, the number of occurrences of a worm in the network is an indication of its aggressiveness, and will be used to classify the signatures based on the threat they pose.

Besides *gathering* these incidence reports, the CC also *pushes* information to NID and NID sensors. A sensor that identifies itself as able to perform either intrusion detection or prevention abilities receives periodic reports about what packets it should monitor or block. This automates the deployment of NID signatures, and consists a significant step to zero-hour detection and containment.

In most implementations, the number and size of signatures that each NID or NIP is looking for determines its maximum throughput, therefore we enforce a limit on the size in bytes of the signatures that are pushed to the sensors. We adopt the notion of a *signature budget*, signatures are sorted based on their virulence and we push as many as the budget permits. Additionally, a policy dictates which of these signature are to be filtered. When a signature's activity exceeds a configurable threshold, the sensors will be instructed to filter the offending traffic. Whether a signature is verified to be valid or false, can also be used to exclude false or even unverified signatures from being filtered, preventing in this way the system from accidentally stopping legitimate traffic.

Finally, the CC periodically exchanges information with the global control centre (GCC). The information include newly generated signatures, as well as activity statistics of known signatures. The statistics received by the GCC are accumulated with the ones generated locally to determine a worm's aggressiveness. This accumulation ensures that the CC is able to react on a planetary outbreak, even if it has not yet been attacked itself, achieving immunisation of the protected network.

Vulnerability

2.4 Global Control Centre

The detailed specification of a planetary scale centre for worm control is beyond the scope of our work, nevertheless we briefly mention the aspects of such a centre that are necessary for SweetBait. The GCC collects signatures and statistics in a similar way to a CC, with the main difference being the lack of a signature budget when pushing signatures to CCs. Additionally, because of performance, as well as privacy concerns only the number of alerts is exchanged discarding the source and destination IPs information. As a single GCC obviously is susceptible to attacks and could become itself a liability, a distributed solution is preferable both for performance and security reasons.

3 System Implementation

This section discusses the implementation of sensors and CC. We have employed already available solutions wherever possible in an attempt to seamlessly integrate already deployed systems in SweetBait.

3.1 Honeypot Sensor

For the honeypot sensor we use *honeyd*, a virtual honeypot framework that provides multiple low-interaction virtual honeypots on a single host [4]. It captures traffic destined to unused IP addresses on the deployed network and supports third party plug-ins that can access and process the captured traffic. Every honeypot is attached to an operating system (OS) profile that results in the simulation of its TCP stack on established connections. This approach protects the host from tools like xprobe [12] and nmap [13] that fingerprint TCP packets to identify its implementation and expose the host's OS. Besides simulating operating systems, honeyd supports scripts that emulate services such as a web server or a telnet daemon.

For automatically generating NID signatures from the captured traffic, we employ *honeycomb*, a *honeyd* plug-in that scans incoming traffic and detects repeating patterns using a longest common substring (LCS)

algorithm [6]. In addition, *honeycomb* performs flow reconstruction, and is able to detect patterns even when they are segmented in multiple IP packets. Signatures are periodically written out to a log file in pseudo-snort rule format along with a timestamp that can be later read and distributed to the CC.

To utilise a filter for *whitelisted* patterns, we developed a new *honeyd* plug-in named *honeybounce*. This plug-in supports a list of rules that specify in snort rule format the patterns to be excluded from the NID signature generation. This is achieved by loading *honeybounce* prior to *honeycomb* and rejecting the matching packets. Since *honeyd* plug-ins do not support packet rejection, we have developed a patch that installs this functionality. Currently, *honeybounce* does not perform flow reconstruction, because of difficulties imposed by the *honeyd* plug-in architecture and to conserve CPU time for pattern detection by *honeycomb*. We do not expect this to become an inconvenience, because of the nature of *whitelisted* traffic, which is benevolent by definition and consists mainly of broadcast and multicast messages originating from the subnet. These messages are mostly small enough to be contained in a single packet, and in all other cases we observed in practice that fragmentation is predictable, being the result of regular fragmentation of a stream into IP packets.

Honeybounce supports filtering of TCP and UDP packets based on exact byte sequences and perl regular expressions. To accelerate the filtering procedure the filters are classified in three categories based on protocol: TCP, UDP or ANY. For each of these classes the filters are hashed based on their destination port number(s), which can also be ANY. Filters are applied sequentially, when a match occurs the procedure is terminated by signalling *honeyd* to reject the packet averting its processing by *honeycomb* and subsequent plug-ins.

The signatures generated by *honeycomb* are read by a signature distribution process that transmits them immediately to the CC. The signatures are first examined to ensure that the content is valid and that they are not older than the last signature received from the CC. This is accomplished by requesting the timestamp of the last signature received from the CC, when first establishing the connection. Such an approach is necessary, because *honeycomb* generates signatures even for obscure protocol flags combinations, and additionally dumps all generated signatures periodically. Finally, SSL is used between the honeypot and the CC to perform authentication between the two using public and private keys, and to ensure that the exchanged data are secure from eves dropping.

3.2 Network Intrusion Detection and Prevention Sensor

Snort [14] is one of the most popular open source NIDS, and is deployed in many networks. This along with the fact that *honeycomb* generates signatures in snort rule format motivated the adoption of snort as the base of NID sensors. Snort scans received traffic for a set of rules and generates an alert each time a match occurs. These alerts are logged in a file and subsequently transmitted to the CC. The information contained in an alert includes a custom annotation, which we use to identify the rule that caused it, and the involved IP addresses. Such an alert is shown in Example 1. Snort can also react (in a passive way) when a TCP flow has been found to match a rule by using control packets to terminate it. This is accomplished by transmitting a TCP FIN packet to both ends of a TCP flow, when a corresponding rule is matched. Unfortunately, in the case of worms such a mechanism is not sufficient, since the original packets containing the worm have already reached their destination, and have probably infected it.

*Example 1 (*Snort alert*).*

```
[**] [1:2003:4] MS-SQL Worm propagation attempt [**]
[Classification: Misc Attack] [Priority: 2]
08/24-16:03:13.805589 192.168.2.24:1178 -> 192.168.24.16:1434
UDP TTL:108 TOS:0x0 ID:30134 IpLen:20 DgmLen:404
Len: 376
[Xref => http://vil.nai.com/vil/content/v_99992.htm]
[Xref => http://www.securityfocus.com/bid/5311]
[Xref => http://www.securityfocus.com/bid/5310]
```

Since snort is not deployed in-line and does not offer an efficient protection mechanism, along with the lack of open NIP alternatives, we implemented a very simple NIP system based on Linux netfilter (`http://www.netfilter.org`). Netfilter on a Linux router permits us to intercept packets before being routed, and thus filter them at the point of entry in the network.

We developed a Linux kernel module, named `CBFilter`, based on Netfilter to perform content based filtering. The module scans for exact byte sequences in packets' payload using the well-known Aho-Corasick algorithm [15]. If a packet's payload matches a target pattern corresponding to one of the rules and the packet's protocol and destination port number also match, the packet is dropped. We have chosen to scan first the payload of a packet and then check the protocol and port number to avoid instantiating multiple search trees for each protocol and port number pair. Such an approach would diminish the benefits of using the aho-corasick algorithm, reducing performance to that of a serial search algorithm. Note that even though the algorithm is the most effective we could use, this procedure remains computationally expensive.

`CBFilter` is controlled from user-space over a device file. A process can instruct the module to load new filters, start and stop filtering, as well as recover statistics. Statistics include the number of packets filtered using a specific rule, but not the source and destination addresses involved due to performance restrictions. Each time statistics are retrieved, the corresponding counter is reset.

The alerts generated by snort and `CBFilter` are collected by a signature distribution process that transmits them to the CC. The same process also listens for signature updates from the CC, and applies them to snort and `CBFilter`. To minimise data transmission, the process supports alerts caching: aggregation of the alerts generated by each rule, and periodic transmission of the number of hits incurred during ine each period. This is useful when the number alerts is sufficiently large to approach saturation of the connection with the CC, or the CC itself. The aggregation occurs at the expense of detailed information that may be used for auditing, as the IP addresses are not included with the aggregate alerts. As in all case, we use SSL on this connection for authentication and security purposes.

3.3 Control Centre

We implemented CC as a multi-threaded server that handles multiple concurrently connected sensors, and uses a PostgreSQL database for storing its data (`http://www.postgresql.org`). system in *SweetBait* is The CC collects two types of information: signatures and alerts. When a new signature N is received it is first compared with every stored signature S, sharing the same protocol, using the LCS algorithm. If an overlap O exists, then the following apply:

1. **Specialisation:** the length of O is at least $X\%$ of the length of N.
   O will be treated as a new version of the stored signature S exists. On practice we found an initial value of $X = 85$ to perform well, as it leaves space for the generation of more specialised signatures, while protecting the system from misclassifying a new worm signature N as a new version of a stored signature S.

2. **Generalisation:** the length of O is at least $Y\%$ of the length of S;
   This rule was introduced to keep the system consistent when a new honeypot sensor is introduced or an already running one is restarted. The honeypot sensors do not hold persistent information regarding previously generated signatures; when restarted in an attempt to generate signatures as soon as possible they will generate signatures more generic than the ones already stored at the CC. These signatures will be large enough to evade the first rule, but will be captured by this one. The value of Y should be just below 100, or even 100 to completely eliminate the possibility of missing valid new signatures. In practice, we found a value of 95 to be sensible and effective.

In both cases, if O is identical with S, it is discarded. Furthermore, to avoid over-specialisation of signatures and unreasonable signature lengths, we also introduce a minimum and maximum. If the length of N or O do not fall within these limits it is also discarded. As for the alerts, their process is quite straightforward: whenever an alert is received the activity counter of the corresponding signature is increased, and the involved IPs are stored in the database.

The CC periodically updates the NID and NIP sensors with the set of signatures that should be monitored and filtered respectively. Because we enforce a budget on the maximum size of the signature set deployed on these sensors, we need to sort them based on their expected aggressiveness. To quantify this, we selected the exponentially weighted moving average of the number of alerts generated by each signature on each period. It is defined as

$$m' = w \times a + (1 - w) \times m \tag{1}$$

where: m' is the new value, m is the previous value, a is the number of alerts this period, and w is the weight (a configurable parameter). Selecting a value for the weight $0 < w \leq 1$ configures m to follow more or less aggressively the recent changes in activity levels. In practice, values less then 0.5 are not very useful. The value m is used to predict future values of a worm's aggressiveness. The value of a signature's activity A that is eventually used by *SweetBait* is biased towards specific destination ports and protocols:

$$A = m \times \mathit{portbias} \times \mathit{protocolbias} \tag{2}$$

This approach allows us, for instance, to react more aggressively to UDP than TCP worms, and with caution to signatures involving web or mail services. This is also useful as some ports (e.g., ports 139 and 80) and protocols are much more frequently attacked (or scanned) than others [16].

The signatures are subsequently transmitted to the NID and NIP sensors, starting with new signatures, and proceeding with signatures that have the largest activity value, going as far as the signature budget allows. Signatures with a value of A larger than the filtering threshold are transmitted to the sensors with the indication that they should be filtered, unless the administrator of the system has requested that only verified ones should be filtered.

Finally, the CC periodically contacts the GCC to exchange signatures and global activity statistics. The received statistics are aggregated with the local ones to provide new values of a and A. Again, SSL is used for communication between CC and GCC.

3.4 Global Control Centre

The global control centre is a stripped-down version of the CC described above. It is a multi-threaded server that handles multiple connections from CCs, and exchanges signatures and statistics. Signature specialisation is done as described in Sect. 3.3. Activity statistics received by the CCs are aggregated, and are periodically cleared to avoid stale values from inhibiting the ability of detecting new outbreaks.

4 Experimental Evaluation

To evaluate SweetBait we deployed it in four different sites: Vrije Universiteit in Amsterdam, ICS FORTH in Heraklio, UNINET in Oslo, and University of Pennsylvania in the US. In all cases we deployed a honeypot sensor, but as it was unlikely that we would obtain permission to setup a NID or NIP system for the entire network, we resorted to deploying the NID system on the same host as the honeypot. The goal of our evaluation is to prove the ability of SweetBait to generate valid worm signatures, and achieve a low reaction time.

The size of unused IP address space varied from 32 IPs in ICS FORTH to two class C subnets in Vrije Universiteit. As expected, the larger the address space, the more traffic was captured by the honeypot and consequently more signatures were generated. Additionally, we noted increased activity in our University of Pennsylvania honeypot, which may be caused by the higher density of IP addresses in this area.

Initially, we ran SweetBait for 24 hour lapses, to get a first glimpse on the generated signatures, and tune the system to achieve the best possible results. We setup the honeypot to emulate hosts running the following operating systems: Linux kernel 2.4.20-2.5.20, Windows XP Professional RC+1, and MS Windows Professional Advance Server Beta3. Additionally, the hosts emulated services such as FTP, POP3, and IIS application server, while accepting connections on all ports. Obviously, such a choice is not suggested for a production system, since it would expose the honeypot, but it is ideal for maximising the captured traffic during evaluation.

4.1 Signature Generation

Exceeding our expectations we collected a significant number of signatures in just a couple hours. Using the values given in section 3.3, signature specialisation reduced the tens of thousands generated by *honeycomb* to tens. Table 1 depicts this for five of our experiments. SweetBait also generated signatures that could

Table 1. Specialisation results

Usable signatures *(honeycomb log)*	**Unique signatures** *(honeycomb log)*	**CC entries** *(database)*
23400	12356	14
2861	592	9
6030	2402	11
35500	9269	20
43470	22504	21
323237	4042	27

not be applied to a NIP sensor, since it would not be able to discriminate between legitimate and malicious traffic, resulting in the *whitelisted* signatures shown in Table 2.

After applying the *whitelist* at the honeypot the results where further improved. The generated signatures consisted of well known older worms such as CodeRedII [17], Slammer, MSBlaster [18] and Nimda [19], as well as many exploit attempts including the more recent Veritas backup exec [20] and Microsoft WINS [21] vulnerabilities. A significant number of signatures is indirectly connected with worm propagation, since it involves traffic targeting backdoors created by worms on infected hosts, such as MyDoom [22] and Sasser [23]. A detailed list of all the generated signatures can be found on-line at `http://www.liacs.nl/~gportoka/signatures.html`.

Almost 100% of the generated signatures concern malicious activity, regardless whether they are the result of a computer worm, or a human exploiting vulnerabilities. In the latter case, it is possible that generated signatures are not applicable to NIP sensors, since it might also hinder access to legitimate users.

Most of the signatures are of less than 200 bytes long. Small signatures focus on the exploit used by a worm, and permit us to deploy more of them on the NID and NIP sensors. The distribution of the size of the generated signatures is shown in Fig. 2. The fact that the length of most signatures is smaller than an IP packet does not imply that the worms used a single packet to propagate. *Honeycomb* employs flow reconstruction, and can identify patterns fragmented across multiple packets.

4.2 Performance

To complete the evaluation of our system, we conducted measurements regarding the performance of the CC. It has to be able to process all the received information in a reasonable amount of time to achieve a low reaction time to worm outbreaks. Because of the nature of the honeypot sensor, the amount of traffic sent to the CC is negligible, while the number of alerts generated during an outbreak could be overwhelming.

Table 2. Whitelist

```
alert udp any any -> any 137 (msg: "NetBIOS Name Service Wildcard Query";
pcre: "\x00*\x0F*\x00*\x01\x00*\s*CKAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\x00*!
\x00*\x01*\$"; )

alert tcp any any -> any any ( msg: "NULL packets"; pcre: "^\x00+$"; )

alert tcp any any -> any 80 ( msg: "Web Server DoS"; pcre: "^GET / HTTP/
1\.1\r\nAccept\x3A image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
\*/\*\r\nUser-Agent: Mozilla/4\.0 (compatible; MSIE 5\.5; Windows 98)\r\n
Host\x3A .+\r\nConnection\x3A Keep-Alive\r\n\r\n$"; )

alert tcp any any -> any 139 ( msg: "Session Request to SMBSERVER";
pcre: "\x81\x00\x00D [A-Z]{32}\x00 [A-Z]{32}\x00$"; )

alert tcp any any -> any 139 ( msg: "Session Request to SMBSERVER";
pcre: "A\x00 [A-Z]{32}"; )

alert tcp any any -> any 80 ( msg: "IIS WebDAV request"; pcre: "^OPTIONS
 / HTTP/1\.1\r\ntranslate\x3A f\r\nUser-Agent\x3A Microsoft-WebDAV-
MiniRedir/5\.1\.2600\r\nHost\x3A .+\r\nContent-Length\x3A 0\r\n
Connection\x3A Keep-Alive\r\n\r\n$" );
```

We conducted experiments with a NID sensor generating false alerts to test the throughput of the CC, and locate possible bottlenecks. We set up the NID sensor to continuously transmit alerts, and measured the number of alerts that were processed every second at the CC. To achieve more realistic results, a honeypot sensor was also connected and was sending signatures. Initially, the NID sensor did not use alert caching, which resulted on a mere 15 alerts per second on average. Investigating the cause of such poor performance, we discovered that the database needed approximately 70 msec to store a single alert. Using faster hardware to host the control centre would definitely improve throughput, since we just used a PC with 256MB of memory running at 1.2Ghz. Switching to alert caching overcomes this limitation by achieving an astounding 140,000 alerts per second.

Another aspect of performance is the time it takes for the control centre to initiate monitoring, and consequently filter a new worm signature. As we have mentioned before, the generation of signatures depends on repeated byte patterns being identified by *honeycomb*. This implies that the speed in which a signature is generated,depends on the speed of the worm itself. As soon as the signature is generated, it is send to the CC within seconds. The time needed to initiate monitoring, depends solely on the period T that the CC updates NID and NIP sensors. Exchange with the GCC is also performed periodically,

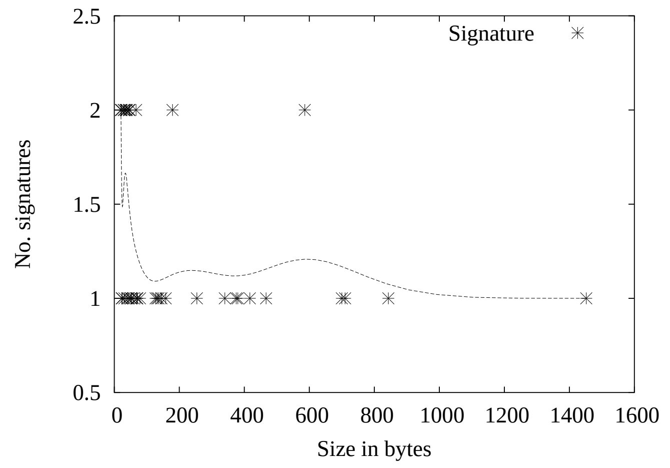

Fig. 2. *Signature size distribution*

so the maximum time needed for a signature to be distributed globally is $2 \times T$, assuming all deployed systems have the same period. $2 \times T$ has to elapse as well, before filtering a signature, assuming that its activity has exceeded the defined threshold. To conclude, a short update period leads to fast reaction times against new worms, but would overwhelm the GCC. A distributed GCC would overcome such issues, but is beyond the scope of this paper. During our evaluation, T was set to 2 minutes and we were pleased to observe that no unexpected performance degradation occurred.

5 Related Work

Various types of honeypots have been used for worm detection. A network of high-interaction honeypots are used to capture worms in the honeynet project [7]. By analysing network traffic and the honeypot's state it is possible to produce detailed descriptions of worm behaviour. Successful application of low-interaction virtual honeypots was demonstrated by Laurent Oudot in capturing and counter-attacking the MSBlaster worm [24]. . LaBrea [25], is a honeypot used as a tarpit: it slows down scanning worms, by keeping TCP connections open indefinitely period. While effective for some worms, it would be powerless in the face of a UDP worm like Slammer. Sombria [26] is yet another honeypot system that has been setup for research purposes in Japan. All of these projects differ form *SweetBait* in that the focus is on capturing worms, rather than on automated response based on automatically extracted and refined signatures.

Honeycomb is a system that automatically creates intrusion detection signatures [6]. It uses honeyd [4], a popular low-interaction virtual honeypot, to capture suspicious traffic and uses a longest common substring algorithm to compare received packets and consequently generate NID signatures. Honeycomb managed to generate accurate signatures for the Slammer and Code Red II worms; nevertheless it can be fooled by long sequences of bytes repeated by certain protocols such as NetBIOS or by malevolent service availability scans, and create signatures for otherwise legitimate traffic. Another system that automatically generates signatures for TCP worms is AutoGraph [27]. It operates by analysing prevalence of portions of flow payloads and exhibits a fairly low false positives rate. Like *SweetBait* it operates better in a distributed environment. However, it is not aimed at finding refinements or generalisation of signatures, nor is it currently coupled to an automated response system.

A worm containment environment for enterprise scale networks was proposed by Joukov and Chiueh [28]. Their design combines anomaly detection, egress filters and honeypots to generate worm signatures and filter them at the enterprise firewall. It is a system quite similar to the one we propose. Its major weakness can be identified in that entire TCP sessions destined to a non-existent host are treated as worm signatures and no dedicated host is acting as a honeypot. The ease by which even an unsophisticated polymorphic worm could evade detection and flood the system with new worm signatures is apparent. Furthermore, the system is isolated from the rest of the Internet unable to take advantage of worm epidemic alerts generated by other parts of the Internet.

A similar system also addressing the issue of an Internet wide centre to correlate warnings and share information is described by Changchun Zou et al [29]. Ingress and egress scan monitors are distributed in different parts of the network and submit their warnings to a *malware* warning centre. The monitors are using a Kalman filter to identify the propagation of a worm based on observed illegitimated scan traffic. Such an approach aims to detect zero-day worms at their early stage, but is vulnerable to background noise that could cause a high rate of false alarms.

A fast containment system is presented in [30]. It differs from most projects described here, including ours, in that it takes a network-centric and aims at implementation in hardware.

A cooperative immunisation system against worms is described by Anagnostakis et al [31]. The system consists of distributed nodes exchanging information about on going threats and the actions necessary to defend. Each node hosts an agent which scans incoming traffic for worm and virus signatures. The set of scanned signatures and the intensity of scanning is determined by the observed virulence of each worm/virus and the available resources of each node. Node agents periodically poll each other to update their state. To solve the issue of trust, each agent polls multiple nodes to compose a global state used to validate the node's claims. This method of validation can be susceptible to attacks if a large number of nodes is compromised. Also, the assumption made that routers and switches are less likely to be attacked leaves these network components exposed to attacks.

A more aggressive approach is adopted by Sidiroglou and Keromytis [32]. Similarly to the architecture we propose, diverse sensors are employed for monitoring including network monitors and honeypots. The honeypots used are highly-interactive, running real versions of popular applications to be protected. To avoid the compromise of the honeypots the applications run on a sandbox environment or on a high performance virtual machine. The applications are monitored for illegal behaviour, and when such is detected the error that caused it is located and a patch is automatically generated and distributed through a software update service. Such active measures cannot be easily trusted though. An automatically generated patch could do more harm than good and it leaves the possibility of gaming by hackers; carefully crafted input to the honeypots could cause the generation of patches that create weaknesses.

Besides Honeypots there are various other approaches to intrusion detection and prevention. Anomaly detection systems (ADS) In-band content inspection [**?**]. Intrusion-tolerant systems [33]. Bro [34]. Early warning [29]

6 Conclusions and Future Work

In this paper we discussed the design and implementation of SweetBait, a system that is an amalgam of network intrusion detection and preventions technical. It was shown that SweetBait is able to automatically generate signatures for random IP address space scanning worms without any prior knowledge. We also demonstrated how this information can be distributed and deployed without any human intervention minimising reaction time to zero-day worms. Furthermore, the signature version-ing, activity prediction and automatic deployment techniques introduced provide an invaluable administration tool, which condenses the information that need auditing by administrators, while self-adapting to ensure a high throughput of the monitoring nodes.

Our future plans are to further extend SweetBait with new sensors, as well as to improve existing ones. New sensor types will be introduced to capture 'hit-list' worms, which currently evade our honeypot sensor. Improvements will be explored in the section of signature generation by moving to algorithms employed in Bioinformatics, which are able to detect all common substrings between two sequences. Technical modifications are also scheduled, communications between sensors will be changed from periodical to asynchronous, to even more lower reaction time to new worms. Additionally, we are also considering distributing the planetary scale control centre to achieve fault tolerance and higher performance. Finally, we will also be involved in worm detection in encrypted network traffic.

References

[1] Stuart Staniford, Vern Paxson and Nicolas Weaver: How to 0wn the internet in your spare time. In: Proceedings of the 11th USENIX Security Symposium. (2002)
[2] Staniford, S., Moore, D., Paxson, V., Weaver, N.: The top speed of flash worms. In: Proc. of the 2004 ACM Workshop on Rapid malcode (WORM'04), New York, NY, USA, ACM Press (2004) 33–42

[3] Systems, C.: Cisco secure intrusion detection system version 2.2.0 user guide (netranger) (2003)

[4] Provos, N.: A virtual honeypot framework. Technical Report 03-1, CITI (2003)

[5] Manuel Costa, Jon Crowcroft, M.C., Rowstron, A.: Can we contain internet worms? In: Third Workshop on Hot Topics in Networks (HOTNETS-III), San Diego, CA (2004)

[6] Kreibich, C., Crowcroft, J.: Honeycomb - Creating Intrusion Detection Signatures Using Honeypots. ACM SIGCOMM Computer Communication Review **34** (2004) 51–56

[7] Levine, J., Grizzard, J., Owen, H.: Using honeynets to protect large enterprise networks. IEEE Security & Privacy **2** (2004) 73–75

[8] Jiang, X., Xu, D.: Collapsar: A vm-based architecture for network attack detention center. In: USENIX Security Symposium, USENIX (2004) 15–28

[9] Provos, N.: A virtual honeypot framework. In: 13th USENIX Security Symposium, San Diego, CA (2004)

[10] Vanderavero, N., Brouckaert, X., Bonaventure, O., Charlier, B.L.: The honeytank: a scalable approach to collect malicious Internet traffic. In: Proc. of IISW04. (2004)

[11] Dornseif, M., Holz, T., Klein, C.: Nosebreak - attacking honeynets. In: Procceedings of the 5th Annual IEEE Information Assurance Workshop. (2004) Honored with Best paper award "in recognition of valuable contributions to the advancement of the Information Assurance field through research and publication".

[12] Ofir Arkin and Fyodor Yarochkin: Xprobe v2.0: A "Fuzzy" Approach to Remote Active Operating Systems Fingerprinting. `http://www.xprobe2.org` (2002)

[13] Fyodor Yarochkin: Remote OS Detection via TCP/IP Stack Fingerprinting. `http://www.nmap.prg/nmap/nmap-fingerprinting-article.html` (1998)

[14] Roesch, M.: Snort Lightweigt Intrusion Detection for Networks. In: Proceedings of USENIX LISA '99: 13th Systems Administration Conference. (1999)

[15] Aho, A.V., Corasick, M.J.: Efficient string matching:an aid to bibliographic search. In Manacher, G., ed.: Communications of the ACM. Volume 18. (1975)

[16] Dacier, M., Pouget, F., Debar, H.: 10th ieee pacific rim international symposium on dependable computing (prdc 2004), Papeete, Tahiti (2004)

[17] Symantec: CodeRedII. `http://www.symantec.com/avcenter/venc/data/codered.ii.html` (2001)

[18] Symantec: w32.blaster.worm. `http://securityresponse.symantec.com/avcenter/venc/data/w32.blaster.wor%m.html` (2003)

[19] CERT: CERT Advisory CA-2001-26 Nimda Worm. `http://www.cert.org/advisories/CA-2001-26.html` (2001)

[20] SecuriTeam: Veritas backup exec agent browser registration request exploit. `http://www.securiteam.com/exploits/5ZP0G0KEKQ.html` (2005)

[21] K-Otik: Microsoft wins remote code execution exploit. `http://www.k-otik.com/exploits/20041231.ZUC-WINShit.c.php` (2004)

[22] Symantec: W32.mydoom.a@mm. `http://securityresponse.symantec.com/avcenter/venc/data/w32.novarg.a@mm%.html` (2004)

[23] Symantec: w32.sasser.worm. `http://securityresponse.symantec.com/avcenter/venc/data/w32.sasser.worm%.html` (2004)

[24] Oudot, L.: Fighting internet worms with honeypots. `http://www.securityfocus.com/infocus/1740` (2003)

[25] Liston, T.: Welcome To My Tarpit: The Tactical and Strategic Use of LaBrea. `http://www.threenorth.com/LaBrea/LaBrea.txt` (2001)

[26] Service, S.S.: A Walk Through "Sombria": A Network Surveillance System. `http://www.lac.co.jp/business/sns/intelligence/sombria_e/smbr_1.pdf` (2003)

[27] Kim, H.A., Karp, B.: Autograph: Toward Automated, Distributed Worm Signature Detection. In: Proc. of the 13th USENIX Security Symposium, San Diego, CA (2004)

[28] Joukov, N., cker Chiueh, T.: Internet worms as internet-wide threat. Technical Report TR-143, Experimental Computer Systems Lab (2003)

[29] Cliff Changchun Zou, Lixin Gao, Welbo Gong and Don Townsley: Monitoring and early warning for internet worms. In: Proceedings of the 10th ACM conference on Computer and communication security. (2003) 190–199

[30] Weaver, N., Staniford, S., Paxson, V.: Very fast containment of scanning worms. In: 13th USENIX Security Symposium, San Diego (2004) 29–44

[31] K. G. Anagnostakis, M. B. Greenwald, S. Ioannidis, A. D. Keromytis and D. Li: A Cooperative Immunization System for an Untrusting Internet. In: Proceedings of the 11th IEEE Internation Conference on Networking (ICON). (2003)

[32] Sidiroglou, S., Keromytis, A.D.: A network worm vaccine architecture. In: 12th International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises. (2003)

[33] Just, J.E., Reynolds, J.C., Clough, L.A., Danforth, M., Levitt, K.N., Maglich, R., Rowe, J.: Learning unknown attacks - a start. In: RAID. (2002) 158–176

[34] Paxson, V.: Bro: A System for Detecting Network Intruders in Real-Time. Computer Networks **31** (1998) 23–24