Collecting Autonomous Spreading Malware Using High-Interaction Honeypots

Jianwei Zhuge[1], Thorsten Holz[2], Xinhui Han[1], Chengyu Song[1], and Wei Zou[1]

[1] Institute of Computer Science and Technology,
Peking University, China
{zhugejianwei|hanxinhui|songchengyu|zouwei}@icst.pku.edu.cn
[2] Laboratory for Dependable Distributed Systems,
University of Mannheim, Germany
holz@informatik.uni-mannheim.de

Abstract. Autonomous spreading malware in the form of worms or bots has become a severe threat in today's Internet. Collecting the sample as early as possible is a necessary precondition for the further treatment of the spreading malware, e.g., to develop antivirus signatures. In this paper, we present an integrated toolkit called *HoneyBow*, which is able to collect autonomous spreading malware in an automated manner using *high-interaction honeypots*. Compared to low-interaction honeypots, HoneyBow has several advantages due to a wider range of captured samples and the capability of collecting malware which propagates by exploiting new vulnerabilities. We validate the properties of HoneyBow with experimental data collected during a period of about nine months, in which we collected thousands of malware binaries. Furthermore, we demonstrate the capability of collecting new malware via a case study of a certain bot.

Keywords: Honeypots, Intrusion Detection Systems, Malware.

1 Introduction

Since the outbreak of the Code Red worm in 2001, malware has become one of the severest threats to the Internet. Especially autonomous spreading malware in the form of worms or bots that propagates over the Internet and infects thousands of computers all over the world in days or even minutes is a problem. In the form of *botnets*, the comprised computers can even be organized into networks that can be remotely controlled by an attacker, and cause lots of harms following the attackers' purposes [14].

In order to deal effectively and efficiently with the threat associated with malware, CERTs, antivirus vendors, and security researchers need to obtain a sample of the actual malware as quickly as possible in the early stage of propagation. This sample can then be analyzed deeply, e.g., to study the propagation and infection mechanism, in order to develop accurate detection signatures or an appropriate treatment strategy. Conventional sample collection approaches include extraction of the binary from an infected machine, reports from customers, exchange between AV vendors, and similar ways. These conventional approaches generally need human interaction. With the increasing

S. Qing, H. Imai, and G. Wang (Eds.): ICICS 2007, LNCS 4861, pp. 438–451, 2007.

birth rate of new malware and the speeding up of the malware propagation, e.g., in the case of the Slammer worm [6], these malware collection approaches with human interaction are always too late for timely incident response. Therefore, we need a completely automated malware collection scheme to catch these trends.

As a new active attack-decoying technology, *honeypots* have been used in the domain of Internet security threats measurement. A honeypot is defined as an information system resource whose value lies in unauthorized or illicit use of that resource [13]. A honeypot has no production usage, therefore, every access launched by the attackers – including automated malware – can be captured and studied in detail. In general, honeypots can be distinguished into two different types: *low-interaction* and *high-interaction* honeypots. Low-interaction honeypots offer limited interaction level to the attackers, commonly through simulation (or emulation) of network services or operation systems. Therefore, they can often only lure automated attacks, and can be identified by a human attackers easily. A popular example of this kind of honeypots is *honeyd* [8]. High-interaction honeypots, on the other hand, use *real* systems for attackers to interact with. This type of honeypots is commonly more complex, furthermore deployment and maintenance often takes more time. In addition, more risks are involved when deploying high-interaction honeypots since an attacker can get complete control of the honeypot and abuse it, e.g., to attack other systems on the Internet. Thus it is necessary to introduce and implement *data control* mechanisms to prevent the abuse of honeypots. The most common used setup for high-interaction honeypots are GenIII honeynets [2].

In this paper, we introduce the *HoneyBow* toolkit, an automated malware collection system based on the high-interaction honeypot principle. The HoneyBow toolkit integrates three malware collection tools called *MwWatcher*, *MwFetcher*, and *MwHunter*. All of them use different techniques and strategies to detect and collect malware samples, in order to achieve a comprehensive collection efficiency. HoneyBow inherits the high degree expressiveness of high-interaction honeypots: it can be constructed upon various customized honeynet deployments, using the true vulnerable services as victims to lure malware infections, but not emulated vulnerable services. Thus HoneyBow is capable of collecting zero-day malware even if the vulnerability exploited during the propagation phase is unknown to the community before the malware outburst. Furthermore, we do not need to investigate the details of the vulnerabilities and implement an emulated version of the vulnerable services, which is commonly required for low-interaction honeypots. Thus the deployment of the HoneyBow toolkit is more flexible and easy. On the other hand, HoneyBow has its limitation in the scalability compared to low-interaction honeypots. Therefore, we combine HoneyBow and the low-interaction honeypot *Nepenthes* [1] to build an integrated malware collection system.

This paper is organized as follows: Section 2 describes the related work in the area of honeypot and automated malware collection research. Section 3 introduces the HoneyBow toolkit in details, discusses the advantages and limitations of our approach, and shows how to integrate Nepenthes, HoneyBow and the GenIII Honeynet to achieve a fully-automated and efficient distributed malware collection solution. Section 4 compares the malware collection efficiency between Nepenthes and HoneyBow. Finally, we conclude the paper and give the further research directions in Section 5.

2 Related Work

Researchers have developed several methods and tools for malware sample collection based on honeypot techniques, among them the Nepenthes platform [1]. Nepenthes uses the principle of low-interaction honeypots: it *emulates* the vulnerable parts of network services to lure and collect malware samples which attempt to infect the host by exploiting these vulnerable services. As the comparable reference to our HoneyBow toolkit, we compare the advantages, limitations, and the practical effects between them throughout the paper. Using high-interaction honeypots, Levine et al. collected and analyzed rootkits manually [5]. This paper is the first to introduce an automatic malware collection schemes based on high-interaction honeypot principle. Furthermore, we are interested in collecting all types of autonomous spreading malware, i.e., worms, bots, and other kinds of malware, in an automated manner.

Antivirus vendors such as Symantec Inc. developed malware collection tools based on the honeypot technology, and present their measurement status reports on the prevalent malware [12]. However, the actual methods and implementations used for these projects are usually not open to the community due to the commercial benefits.

The Honeynet Project develops the GenIII honeynet [2], which composes the foundation of our deployment. We use a GenIII honeynet as a building block of our system. Portokalidis et al. introduce *Argos* [7], a containment high-interaction honeypot environment to study malware as well as human-generated attacks. Argos is built upon the Qemu x86-emulator and is capable of detecting exploitation with the help of a technique called *taint tracing*. In addition, the tool generates intrusion detection signatures for a detected attack. Argos has not implemented special mechanisms for malware sample collection yet, but the principles behind the HoneyBow toolkit can be integrated into Argos. The Potemkin virtual honeyfarm by Vrable et al. exploits virtual machines, aggressive memory sharing, and late binding of resources to emulate more than 64,000 high-interaction honeypots using ten physical servers [16]. Although the implementation of Potemkin is not publicly available, and there are only preliminary results for the scalability available, it shows a promising approach for improving the scalability limitation of high-interaction honeypots deployment, which can be used by the HoneyBow toolkit to overcome its limitations in the area of scalability.

A study similar to ours own was conducted by Goebel et. al [3]. They collected 2,034 valid, unique malware binaries using Nepenthes listening on about 16,000 IPs within a university environment for a period of eight weeks, and present the measurement and analysis results of autonomous spreading malware. We collect two orders of magnitude more malware binaries by combining Nepenthes and HoneyBow. Furthermore, our study lasts for nine months, thus we can study long-term effects and the temporal changes of malware.

Rajab et al. also use Nepenthes to collect spreading bot instances and focus their work on botnets [10]. To support distributed deployment on the PlanetLab testbed, they deploy a modified version of the Nepenthes platform. We propose a standalone structure integrating Nepenthes, HoneyBow and the GenIII Honeynet for distributed honeynet deployment, and have constructed a widely distributed honeynet on the public Internet of China, which contains up to 50 high-interaction honeypots located at 17 nodes now. The presented results are based on the data collected by such an infrastructure.

3 The HoneyBow Toolkit

In this section, we introduce the HoneyBow toolkit in detail. We present the individual building blocks HoneyBow is based on, and show how the high-interaction honeypot principle can be used to construct an automated malware collection approach, especially for collecting malware samples that use unknown or new vulnerabilities.

A high-interaction honeypot is a conventional computer system, deployed to be probed, attacked, and compromised [13]. Such a system has no production usage in the network and no regularly active users. Thus it should neither have any unusual activities on the system nor generate any network traffic. These assumptions aid in attack detection: every interaction with the honeypot is suspicious by definition. HoneyBow uses this idea and is an approach to collect malware with high-interaction honeypots. Compared to Nepenthes, this has the advantage that we do not need to emulate any vulnerable services: we can use a conventional machine, patch it to an arbitrary patch-level, deploy the honeypot, and wait for successful compromises. The key concept is that a malware binary usually propagates itself through the network and installs a copy of itself into the victim's file system after a successful compromise. If we thus monitor the network flow stream and the changes to the file system, we can detect an infection attempt and also obtain a binary copy of the malware sample.

3.1 Architecture of the HoneyBow Toolkit

The HoneyBow toolkit uses similar concept as the GenIII honeynet architecture, the most common setup for high-interaction honeypots used nowadays. In the honeynet research area, there are two well-known methods to deploy high-interaction honeynets: the first is called *physical honeynets* and the second one is called *virtual honeynets* [9]. Physical honeynets use normal machines for deploying high-interaction honeypots, and use an actual networking device to link them together into a honeynet. In contrast to this, virtual honeynets use virtual machines (VMs) like VMware or Virtual PC to set up virtual honeypots. Obviously, virtual honeypots have advantages due to lower deployment costs and easier management compared to physical honeynet. On the other hand, this kind of honeypots also has several disadvantages, e.g., in the area of performance degradation, single point of failure, and higher risk of fingerprinting.

The HoneyBow toolkit supports both methods of high-interaction honeynet deployment. As depicted in Figure 1, the HoneyBow toolkit consists of three malware collection tools: *MwWatcher*, *MwFetcher*, and *MwHunter*, all of which implement different malware collection strategies. Additionally, two tool called *MwSubmitter* and *MwCollector* support distributed deployment and malware collection.

The individual building blocks of HoneyBow perform the following tasks:

– MwWatcher is one of the three malware collection tools implemented in the HoneyBow toolkit. It is based on the essential feature of honeypot – no production activity – and watches the file system for suspicious activity caused by malware infections in real time. The tool is executed on a high-interaction honeypot and exploits a characteristic feature of propagating malware: when some malware successfully exploits a vulnerable service and infects the honeypot, the malware sample will

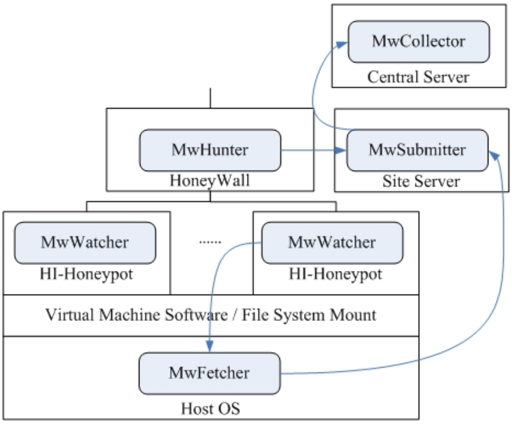


Fig. 1. Schematic Overview of the HoneyBow architecture

commonly transfer a copy of itself to the victim and stored it in the file system. MwWatcher will then detect this change of the filesystem and catch a binary copy of the malware sample. This sample is moved to a hidden directory and waits for further collection by another tool called MwFetcher.

- MwFetcher is the second malware collection tool in the toolkit. This tool runs periodically on the host OS, issues a command to shutdown the honeypot OS, and generates a listing of all files from the hard disk image of the honeypot system. Then this listing is compared to a file list generated formerly from the clean system, and all added or modified files are extracted since they could be artifacts of successful infections. The samples collected by MwWatcher are also extracted and aggregated with the MwFetcher results. After sample extracting, MwFetcher will activate a restore procedure which reverts the honeypot OS to a clean state.
- MwHunter is the third malware collection tool in the toolkit and it is based on the PE Hunter [18] tool. MwHunter is implemented as a dynamic preprocessor plugin for *Snort*, an open source network intrusion detection system, and can be integrated into the Snort instance running at inline mode on the Honeywall of a standard GenIII honeynet [15]. MwHunter relies on the `stream4` and `stream_reassembly` preprocessor build in the Snort daemon: it extracts Windows executables in PE format from the reassembled network stream and dumps them to the disk. The tool tries to find a PE header based on the DOS header magic `MZ` and PE header magic `PE|00|`, and then uses a simple heuristic to calculate the file length. Starting at the position of the header, the resulting number of bytes is then dumped to a file. When an executable has been successfully identified, MwHunter will treat the captured binary as a malware sample due to the properties of the honeynet environment. MwHunter generates an alert including the five tuple *(source IP, source port, IP protocol, destination IP, destination port)* of the network stream, timestamp, and MD5sum of the captured sample.

In a virtual honeynet deployment, apparently, the host OS refers to the operation system where the virtual machine software is installed, and the restore procedure can be easily implemented using the revert functionality that almost all of the virtual machines such as VMware support. But in a physical honeypot deployment, generally, we need to manually reinstall the operation system or restore the file system using system management software such as Norton Ghost. To achieve automated malware collection and honeypot operation, we introduce a full-automatic system restore procedure for physical honeypots based on the *IPMI* (Intelligent Platform Management Interface[1]) and *PXE* (Preboot Execution Environment [4]) protocol. A schematic overview of the system is given in Figure 2. In a physical honeynet deployment, the host OS refers to the little customized Linux kernel which is downloaded and activated via the PXE protocol. MwFetcher operates after step 4 (load base OS) and before step 5 (download the backup honeypot OS image).

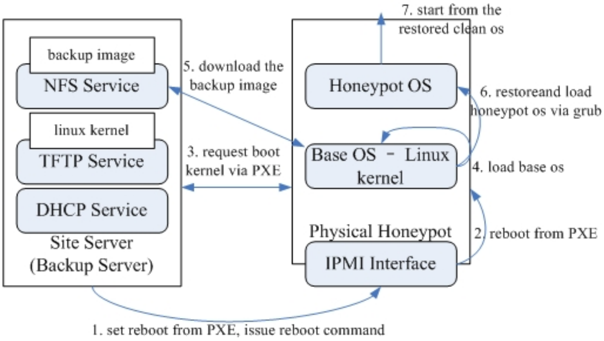

7. start from the restored clean os
backup image
NFS Service
5. download the backup image
Honeypot OS
6. restoreand load honeypot os via grub
linux kernel
TFTP Service
3. request boot kernel via PXE
Base OS - Linux kernel
4. load base os
DHCP Service
Site Server (Backup Server)
Physical Honeypot
2. reboot from PXE
IPMI Interface
1. set reboot from PXE, issue reboot command

Fig. 2. Full-automatic system restore procedure for physical honeypots

MwSubmitter and MwCollector support a distributed deployment: multiple MwFetcher instances can be deployed in a distributed honeynet and each instance sends the collected information to MwSubmitter. This tool monitors the capture logs of the different malware collection tools and the collected binaries, and submits new collected samples to MwCollector. MwCollector is a network daemon at a central malware collection server, accepting MwSubmitter's sample submissions, and storing all collected information and samples in a central database.

Because malware for the Windows operating system constitutes the vast majority of malware in the wild, we implemented the HoneyBow toolkit for now only for Windows. For other platforms such as Linux or FreeBSD, the mechanism of real-time file system monitoring behind MwWatcher, and executables identification and extraction behind MwHunter, can also be implemented. The implementation details differ, but the principle remains the same.

3.2 Comparison of Advantages and Limitations

The HoneyBow toolkit integrates three malware collection tools using different malware identification and collection techniques: MwWatcher runs on the honeypot and

[1] www.intel.com/design/servers/ipmi/

adopts real-time file system monitoring to detect and collect the changed files as malware samples. MwFetcher is executed periodically on the host OS and uses cross-view file system list comparing technique to extract added/modified files. MwHunter is intended to sit inline at the network level in front of high-interaction honeypots, and it can identify and extract Windows executables from the network stream. Due to the nature of honeynet environments, the resulting files collected by these three tools can be treated as malware samples with a low false negative rate.

Although these three tools achieve the same objective, each has their own advantages and limitations when comparing them with one another. We summarize and list the comparison results in Table 1. MwWatcher can be easily detected and bypassed if the malware implements some evading detection mechanisms. In contrast, MwFetcher and MwHunter operate outside the honeypot box and are thus hard to detect by malware. As MwWatcher and MwHunter monitor the file system and the network, respectively, in real time, they can both deal with temporary files which delete themselves after execution. MwHunter can even capture some forms of memory-only malware samples which do not store a copy of themselves on the permanent storage. MwFetcher can not collect temporary files because they have been already eliminated when MwFetcher compares the listing after a certain period. However, MwFetcher has its advantages on detecting concealed malware, including rootkits, which protect themselves from exposing to the application level APIs and tools. MwHunter relies on the signatures of Windows executables during the transmission through the network: if the executable is compressed, encrypted, or encoded, then they can not be detected by MwHunter.

Table 1. Comparison of advantages and limitations among the three different HoneyBow tools

	Collection technique	Advantages	Limitations
MwWatcher	Real-time file system monitoring	Can deal with temporary files	Can be easily detected by malware
MwFetcher	Cross-view file system list comparing	Can deal with concealed malware, such as rootkits; Hard to be detected by malware	Can not collect temporary files; Loss of exact time and attacker information
MwHunter	Identification and extraction from network streams	Can deal with temporary files and some memory-only samples; Passive, hard to be detected by malware	Can not deal with some specially crafted binaries, e.g., self-extracting archives

Since these three tools have their unique advantages and limitations, we integrate them into the HoneyBow toolkit, and hope to achieve better coverage of collecting autonomous spreading malware.

Compared with the Nepenthes platform based on the low-interaction honeypot principle, the HoneyBow toolkit has several advantages. First, HoneyBow is capable of collecting zero-day malware samples which exploit unknown vulnerabilities. This feature is significant for CERTs and AV vendors, since they can then obtain a malware sample in the early stage of their propagation. For example, we could capture samples of bots that use new attack vectors (e.g., MocBot which uses MS06-040 for propagation [11], see Section 4.2) which were not caught by Nepenthes. Second, the high-interaction approach taken by HoneyBow does not need any signature of the malware, including no detailed information about the exploited vulnerability. Thus we do not need to investigate the specific vulnerability and implement an emulated version of the vulnerable service. The deployment and maintenance of the HoneyBow toolkit is quite easy. Third, we can customize the patch level, installed network services, and existing vulnerabilities of the deployed high-interaction honeypots, to satisfy the different requirements of malware collection. Such a customization does not need to modify or re-configure the HoneyBow toolkit and demonstrates the flexibility and easy-of-use of the tool. Fourth, HoneyBow has the capability of collecting the second-stage samples (and possibly even more stages) downloaded by the initial malware.

On the other hand, HoneyBow has several limitations: First, the scalability of HoneyBow is limited. Although we can assign several IP addresses to a high-interaction honeypot to enlarge the measurement scope and improve the malware collection effect, HoneyBow lacks a large scalability compared with Nepenthes, which can emulate more than 16,000 different IP addresses on a single physical machine. With techniques similar to the ones used by Potemkin [16], this could be addressed in the future. Second, HoneyBow relies on special hardware conditions (IPMI-enabled motherboard) when deployed in the physical honeynet mode, and the cost of such a hardware is relative high. When deployed in the virtual honeynet mode, the malware sample can detect the virtual environment (e.g. VMware) and the presence of MwWatcher in order to evade the collection and analysis. Third, HoneyBow can only collect malware samples that remotely exploit security vulnerabilities and infect the honeypot successfully by sending a binary to the victim. Malware that propagates via e-mail or via drive-by downloads can not be captured with such an approach.

Since both malware collection tools have their own advantages and limitations, we should combine these two different malware collection methods adequately, exploiting their advantages while restraining their limitations, to achieve the best malware collection efficiency and coverage.

3.3 Integration of HoneyBow, Nepenthes, and the GenIII Honeynet

To measure security threats on the Internet, we have constructed a distributed honeynet based on the architecture shown in Figure 3. One of the most important objectives of the distributed honeynet is to collect autonomous spreading malware samples in the early stage of their propagation. Furthermore, we want to measure the prevalence of specific malware samples. To achieve these objectives, we integrate HoneyBow, Nepenthes, and the GenIII Honeynet into one architecture. Each honeynet site contains two physical machines: one is used to deploy a standard GenIII virtual honeynet setup based on VMware, and the other takes the role of a *Site Server*. This machine is responsible for the storage, upload, and analysis of the collected samples and attack data.

The HoneyBow tools are installed at different components of the honeynet site: MwWatcher runs on the honeypot guest OS. We use both Windows 2000 and Windows XP as guest OS, in order to cover the two common OS installed on end-user machines. MwFetcher is executed on the host machine of the virtual honeynet, and MwHunter is placed on the Honeywall in front of the honeypots. In order to integrate

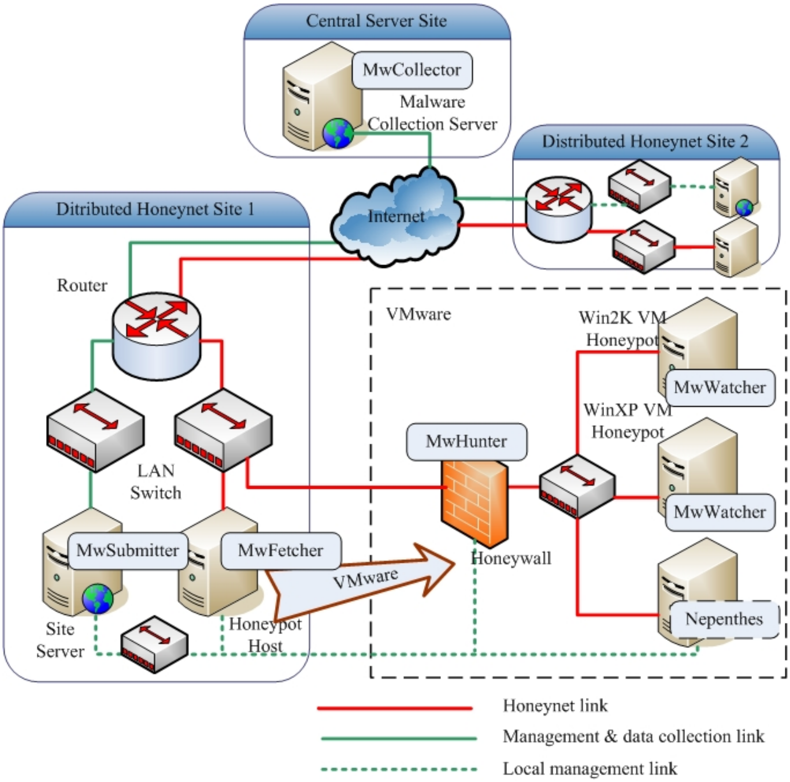


Fig. 3. Integration of HoneyBow, Nepenthes and GenIII Honeynet

malware collection methods based on the low-interaction honeypot principle, we install Nepenthes in a Linux VM and place it behind the Honeywall. All of the malware samples collected by MwWatcher, MwFetcher, MwHunter, and Nepenthes are aggregated to an NFS-mounted directory on the Site Server. From there, all samples are submitted by MwSubmitter to the MwCollector located at a central server site.

4 Collection Results of Autonomous Spreading Malware

In this section, we present the results of collecting and analyzing autonomous spreading malware with the help of a widely distributed honeynet deployment containing 17 sites and up to 50 honeypots around the public Internet of China. Each site is constructed based on the topological structure shown in Figure 3, except the MwHunter tool because it was integrated in the architecture just recently. Our collection and analysis results are based on nine months in-the-wild measurements, which took place during October 2006 and June 2007.

4.1 Statistical Results

With the help of the distributed honeynet setup integrating Nepenthes, HoneyBow and GenIII Honeynet, we had a *hit count* of about 800,000. The hit count specifies the total number of downloaded samples, i.e., how often we successfully captured a binary, disregarding multiple copies of the same binary. As a metric for uniqueness we use the MD5sum. While this has same problems, e.g., small changes in a binary result in a completely different MD5sum, it allows us to quickly determine whether or not we have seen a particular binary before. Using this metric, we collected nearly 100,000 unique sample binaries during the measurement period of nine months.

This means that we have on average about 2,800 collected and 360 new unique binaries per day. The large amount of collected binaries is to some degree due to our weak measurement of uniqueness: by using MD5 hash values, even slight differences in two binaries cause a completely different hash value. This implies that if we capture a polymorphic worm, we can not efficiently differentiate different versions of the same binary. As part of our future work, we plan to develop better metrics to differentiate between malware binaries.

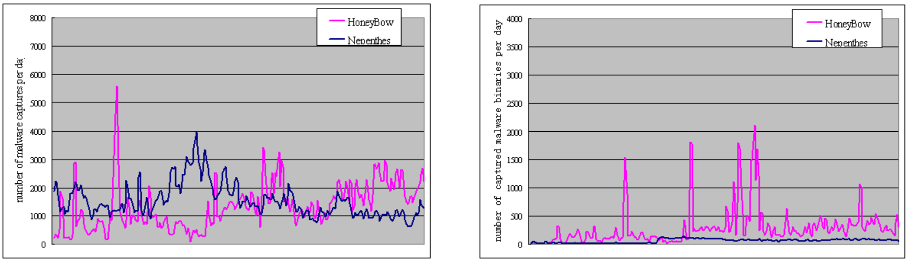
All collected binaries were analyzed with *MwScanner*, a tool that combines nine common antivirus (AV) engines, to identify the known malware variations and families, and to examine the detection rates of these AV engines. Using MwScanner, each collected sample is scheduled to be scanned several times: immediately after collection, after 1 day, after 3 days, after 2 weeks, and finally after 1 month. These results allow us to study the response rates of common AV engines to the threat brought by autonomous spreading malware. In general, the detection rates are rather low. The detection rates vary between 50.4% and 92.8% for the nine engines in the first scan. Even the best engine in our test detected only 93.7% of the samples in the last scan one whole month after the samples was collected.

Table 2 summarizes the comparison of collected malware samples for both Nepenthes and HoneyBow. On average, Nepenthes collects 1,539 samples per day and HoneyBow 1,359, thus Nepenthes captures slightly more samples per day. Nepenthes has predominance on the number of captures because of their capacity to capture unsuccessful infections and some forms of exclusive samples. However, the situation changes when comparing the number of unique samples per day: We collect about 63.7 unique samples with Nepenthes and 296 unique samples with HoneyBow per day. HoneyBow thus yields a higher number of unique malware samples than Nepenthes, mainly because it does not rely on known vulnerabilities. With the help of MwScanner, we are able to compare the numbers of collected malware variations and families between Nepenthes and HoneyBow: We use the output of an AV-engine to assign a given malware sample to a malware family and malware variant. For example, if binary *A* has the AV-label *Trojan.Delf-1470* and binary *B* the label *Trojan.Delf-142*, both belong to the same family, but are different variants. As shown in Table 2, during the measurement period, Nepenthes collected 467 different malware variations of 64 families, but HoneyBow achieved 1,011 variations of 171 families.

In Figure 4, we illustrate the temporal distribution of hit counts and number of unique samples captured over the period of nine months. The hit count in Figure 4(a) shows that both tools collect a comparable amount of binary samples per day, disregarding

Table 2. Comparison of total / average number of collected malware samples for number of captures, binaries, variants, and families between Nepenthes and HoneyBow

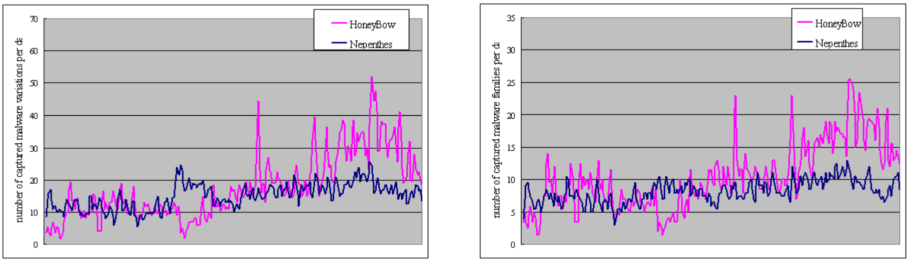
	Captures (hit count)	Binaries	Variants	Families
Nepenthes (Total)	427,829	17,722	467	64
HoneyBow (Total)	376,456	82,137	1,011	171
Nepenthes (Average per day)	1,539	63.7	15.0	8.2
HoneyBow (Average per day)	1,359	296.0	17.8	10.6



(a) Number of malware samples captured (*hit count*) per day

(b) Number of unique malware binaries captured per day

Fig. 4. Comparison of malware collection effects between Nepenthes and HoneyBow



(a) Number of different malware variants captured per day

(b) Number of different malware families captured per day

Fig. 5. Comparison of malware collection effects between Nepenthes and HoneyBow

duplicate copies. Figure 4(b) shows clearly the advantages of HoneyBow for collecting unique binaries: on almost all days, we collect more unique binaries with HoneyBow than with Nepenthes.

The spikes in Figure 4(b) (and also Figure 4(a)) are mainly caused by polymorphic worms: in each iteration, such a worm changes certain parts of itself and thus the MD5 hash value is different. Due to our metric of uniqueness, a polymorphic worm thus generates many hits and a large amount of unique binaries. In the wild, we commonly see polymorphic worms like *All.Aple* which cause such spikes.

In Figure 5, we illustrate the temporal distribution of number of different malware variants and families captured over the period of nine months. In both areas, HoneyBow usually outperform Nepenthes. This is mainly due to the fact that Nepenthes relies on static signatures of how to respond to an incoming attack. If a malware binary uses a vulnerability that Nepenthes does not know how to emulate (or sometimes even a slight variation of a known vulnerability), the tool can not capture a copy of this particular binary. On the other hand, HoneyBow follows the high-interaction principle and uses a real system, thus the actual system replies to an incoming attack and we do not need to emulate a vulnerability.

4.2 MocBot Case

With the help of an anecdotal report, we want to show how HoneyBow is also able to capture malware samples that use an unknown or recent vulnerability.

During the MocBot outbreak in August 2006 [11], our distributed honeynet deployment and the HoneyBow toolkit played an important role. In the early stages of the MocBot outbreak, our HoneyBow system captured the sample for the first time at 03:54 pm of August 13 (Beijing time - CST). After the MocBot sample was downloaded and executed on the high-interaction honeypot, it connected to an IRC-based botnet. The Command and Control (C&C) server used the domain `bniu.househot.com`, and the bot joined an obfuscated channel to accept the botherder's commands. After several hours, the MocBot sample received an obfuscated command which could be decoded as `e http://media.pixpond.com/[removed].jpg`. The bot was thus instructed to download and execute a file from a remote location. This command installed a second-stage infection Trojan named *Ranky*. As shown in Table 3, HoneyBow collected both samples in a very early time. MwScanner was executed by schedule at 04:10 am with latest signature base: none of the AV engines was able to identify the MocBot sample.

Table 3. MocBot samples captured by the HoneyBow toolkit in its early stages of propagation

Sample MD5	Family	Timestamp (CST)	Honeypot
9928a1e6601cf00d0b7826d13fb556f0	IRCBot	2006-08-13 03:54	vmpot.2k
4e618ca11b22732f412bafdac9028b19	Ranky	2006-08-13 11:14	vmpot.2k

Since the exploited vulnerability (MS06-040) was not implemented in Nepenthes (and is not implemented as of today), this tool can not deal with malware that exploits this particular vulnerability. After a deep analysis of the captured sample binary, CNCERT/CC took appropriate strategies, announced the situation and treatment mechanisms to the public. With the help of this information, the botnet constructed by MocBot was then taken down and its propagation was restrained.

5 Conclusion and Future Work

In this paper, we presented an integrated toolkit called HoneyBow to collect samples of autonomous spreading malware. HoneyBow is based on the high-interaction honeypot principle and can collect malware in an automated manner. The HoneyBow toolkit contains MwWatcher, MwFetcher, and MwHunter, each of them using a different malware collection strategy. Compared with the Nepenthes platform based on the low-interaction honeypot principle, HoneyBow has its advantages due to a larger range of captured samples and the capability of collecting malware samples that use new vulnerabilities. The toolkit has its limitation mainly in the area of scalability. Thus we introduced a topological structure which integrates Nepenthes, HoneyBow, and GenIII honeynets, to achieve an even better malware collection coverage. Measurement results of a nine-month period and the MocBot case validated that HoneyBow has better collection coverage compared to Nepenthes and that it is capable of capturing unknown malware samples.

Nepenthes and HoneyBow are both only intended for malware sample collection: they ignore some valuable information about malware propagation including information about the attackers, targeted services, and exploited vulnerabilities. As an improvement, we extend these tools to support the collection of more detailed information. Even with the combination of Nepenthes and HoneyBow, we can not collect malware that uses other propagation vectors like e-mails or exploitation of browsers. We plan to extend our system with client-side honeypots [17] which can be used to fill this gap.

Acknowledgments

This work was supported in part by the 863 High-Tech Research and Development Program of China under Grant No. 2006AA01Z445, Chinese Information Security Research Plan under Grant No. 2006A30, and the Electronic Development Fund of Ministry of Information Industry of China under Grant No. [2006]634. The first author Jianwei Zhuge was supported by a IBM Ph. D. Fellowship Plan.

We would like to thank the anonymous reviewers for valuable comments on a previous version of this paper.

Availability

The HoneyBow toolkit is released under the GNU General Public License (GPL). The software is available for download at `http://honeybow.mwcollect.org/`.

References

1. Baecher, P., Koetter, M., Holz, T., Dornseif, M., Freiling, F.C.: The nepenthes platform: An efficient approach to collect malware. In: Zamboni, D., Kruegel, C. (eds.) RAID 2006. LNCS, vol. 4219, pp. 165–184. Springer, Heidelberg (2006)
2. Balas, E., Viecco, C.: Towards a Third Generation Data Capture Architecture for Honeynets. In: Proceeedings of the 6th IEEE Information Assurance Workshop, IEEE Computer Society Press, Los Alamitos (2005)
3. Goebel, J., Holz, T., Willems, C.: Measurement and Analysis of Autonomous Spreading Malware in a University Environment. In: Proceeding of 4th Conference on Detection of Intrusions & Malware, and Vulnerability Assessment (DIMVA 2007) (2007)

4. Intel Corporation and SystemSoft. The preboot execution environment specification v2.1 (September 1999), http://www.pix.net/software/pxeboot/archive/pxespec.pdf
5. Levine, J., Grizzard, J., Owen, H.: Application of a methodology to characterize rootkits retrieved from honeynets. In: Proceedings of the 5th Information Assurance Workshop, pp. 15–21 (2004)
6. Moore, D., Paxson, V., Savage, S., Shannon, C., Staniford, S., Weaver, N.: Inside the slammer worm. IEEE Security and Privacy 1(4), 33–39 (2003)
7. Portokalidis, G., Slowinska, A., Bos, H.: Argos: an emulator for fingerprinting zero-day attacks for advertised honeypots with automatic signature generation. SIGOPS Oper. Syst. Rev. 40(4), 15–27 (2006)
8. Provos, N.: A virtual honeypot framework. In: Proceedings of the 13th USENIX Security Symposium (August 2004)
9. Provos, N., Holz, T.: Virtual Honeypots: From Botnet Tracking to Intrusion Detection. Addison-Wesley Professional, Reading (2007)
10. Rajab, M.A., Zarfoss, J., Monrose, F., Terzis, A.: A multifaceted approach to understanding the botnet phenomenon. In: Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement, pp. 41–52. ACM Press, New York (2006)
11. Stewart, J.: Mocbot/MS06-040 IRC bot analysis, (August 2006), http://www.secureworks.com/research/threats/mocbot-ms06040/
12. Symantec Inc. Symantec Internet security threat report: Trends for January - June 2007, (2007), http://www.symantec.com/business/theme.jsp?themeid=threatreport
13. The Honeynet Project. Know Your Enemy, http://honeynet.org/
14. The Honeynet Project. Know Your Enemy: Tracking Botnets (March 2005), http://www.honeynet.org/papers/bots/
15. The Honeynet Project. Honeywall CDROM, (March 2007), http://honeynet.org/tools/cdrom/
16. Vrable, M., Ma, J., Chen, J., Moore, D., Vandekieft, E., Snoeren, A.C., Voelker, G.M., Savage, S.: Scalability, fidelity, and containment in the potemkin virtual honeyfarm. SIGOPS Oper. Syst. Rev. 39(5), 148–162 (2005)
17. Wang, Y.-M., Beck, D., Jiang, X., Roussev, R., Verbowski, C., Chen, S., King, S.T.: Automated web patrol with strider honeymonkeys: Finding web sites that exploit browser vulnerabilities. In: NDSS (2006)
18. Werner, T.: honeytrap: Ein Meta-Honeypot zur Identifikation und Analyse neuer Angriffstechniken. In: Proceedings of the 14th DFN-CERT Workshop Sicherheit in vernetzten Systemen (2007), http://honeytrap.mwcollect.org