Malware distributed collection and pre-classification system using honeypot technology

André R. A. Grégio[1], Isabela L. Oliveira[2], Rafael D. C. Santos[3], Adriano M. Cansian[2], Paulo L. de Geus[1]

[1]Institute of Computing, University of Campinas (UNICAMP), Campinas, SP, Brazil;
[2]UNESP – Universidade Estadual Paulista (Sao Paulo State University), Sao Jose do Rio Preto Campus, Brasil;
[3]Computing and Applied Mathematics Lab., National Institute for Space Research (INPE), São José dos Campos, SP, Brazil

ABSTRACT

Malware has become a major threat in the last years due to the ease of spread through the Internet. Malware detection has become difficult with the use of compression, polymorphic methods and techniques to detect and disable security software. Those and other obfuscation techniques pose a problem for detection and classification schemes that analyze malware behavior. In this paper we propose a distributed architecture to improve malware collection using different honeypot technologies to increase the variety of malware collected. We also present a daemon tool developed to grab malware distributed through spam and a pre-classification technique that uses antivirus technology to separate malware in generic classes.

Keywords: Malicious software, malware collection, honeypots, honeyclients, information systems security.

1. INTRODUCTION

The current major threat to information systems security has been the compromise by malicious software (malware), such as botclients, worms, keyloggers, viruses, trojans, rootkits and a sort of other evil programs. Malware can exploit and compromise a system in many ways, either attacking operating systems vulnerabilities or remote services through Internet or deceiving the user to execute it by clicking on a fake link or opening an e-mail attachment. There are some types of malware that even abuse browsers' capabilities to infect the user's system, exploiting embedded mobile code, performing cross-site scripting or cross-site request forgering, such as javascript malware[1].

Nowadays, companies are more worried about security, improving research and trying to discover vulnerabilities early in order to generate quick patches in a periodic manner to protect their users. Also there has been a significative improvement in areas like software development in order to prevent users from attacks aiming design vulnerabilities, with the development and application of defensive technologies at operating system level to integrate security solutions and restrict damage. Examples of those techniques are the new features of Windows Vista like randomization of address space[2] and stack protection against buffer overflows[3].

Parallel to this, changes in the way software is tested can be noted, as they are submitted to security – and not only functional – tests. Techniques such as "abuse cases" [4], complementing the well-known use cases in order to test what systems are not supposed to do, have been applied together with automated tests such as network and application penetration testing, including protocol and application fuzzying.

On the other hand, all these security efforts are susceptible to the combination of large use of old, outdated or not-licensed systems and careless users, as spam (specially with social engineering tricks) is still an effective way to propagate malware. Malware that are spread over e-mails are small in size but full of functionalities. There are lots of spams with just a downloader attached to it that, once executed, try to fetch the real malicious code or even other kind of code to do extra steps before the malicious program can be installed in the target system, avoiding e-mail antivirus mechanisms.

This paper describes a system to gather malware from basically two sources: malware sent by e-mail, and Internet spreading malware that actively attack networked systems. The system is deployed to operate in active and passive mode at the same time, performing a distributed collection by integrating different honeypot technologies such as active clients technique and passive collection by low and medium interaction sensors. In this paper we describe the deployment of the proposed system, the tools used and/or developed, problems faced and provide some results about distributed collection and the pre-classification technique developed by the authors to separate malware.

This document is divided into 5 sections. Section 2 is a brief review of some tools used to collect or analyze malware. In Section 3, we explain the architecture of the proposed system – how it was designed and implemented, the tools used to perform distributed collection, the tools and scripts developed to accomplish the objectives and the configuration. Section 4 show the results obtained using this system and considerations about the problems, its advantages and malware collection and analysis in general. In Section 5 we acknowledge people who helped this work to be done.

2. MALWARE COLLECTION REVIEW

Malware comes from different sources – they disseminate over e-mail or general Internet traffic, trying to exploit vulnerabilities on Web-faced networks and systems or to compromise a computer by making a user perform a specific action as clicking on an URL or opening an e-mail attachment. There is also malware that consist of malicious code in a server to be executed when a user accesses a web site or loads a frame, for example. Thus, there are several ways to collect malware, either setting up a vulnerable system to wait for attacks, or crawling web pages for malicious code stored on servers. In this section some approaches to collect malware are discussed.

2.1 Nepenthes

Nepenthes is a versatile tool to collect malware. It acts passively by emulating known vulnerabilities and downloading malware trying to exploit these vulnerabilities[5]. It allows the quick collection of a variety of samples and determine certain patterns in known and unknown shellcode as it is sent across the network. In addition, this platform can be used to learn more about attack patterns.

Nepenthes is designed to emulate vulnerabilities that malware use to spread (mainly worms), and to capture them. Since there are many possible ways for worms to spread, Nepenthes is designed to be modular. This platform enables us to efficiently deploy thousands of honeypots in parallel and collect information about malicious network traffic[6].

2.2 Honeytrap

Honeytrap is a network security tool written to observe attacks against network services. As a low-interactive honeypot, it collects information regarding known or unknown network-based attacks and thus can provide early-warning information[7].

The applied model strictly distinguishes between data capture and attack analysis. The process of collecting information related to attacks is completely done within the core system. Further processing like automated analysis can be done with plugins, which can be loaded dynamically during runtime. This guarantees expandability without the need of shutting down the system or recompiling the software.

2.3 Honeybow

HoneyBow sensor is a malware collection honeypot based on the high interaction honeypot principle. It is a high-interaction malware collection toolkit and can be integrated with nepenthes to improve collection.

A HoneyBow[8] sensor consists of the following three components: MwWatcher, MwFetcher and MwSubmitter.

MwWatcher malware collection tool is a program which monitors honeypot file system changes in real time and catches potential malware. It currently only runs on a Win32 guest system. MwFetcher malware collection tool is a program which extracts potential malware from a VMware virtual disk (or physical disk) by comparing the infected file list with the clean file list. MwSubmitter malware submit tool is a program which submits potential malware samples collected by MwWatcher and MwFetcher to an specific collection architecture. MwFetcher and MwSubmitter currently only run on Linux operating systems.

2.4 Honeyclients

Differently of the traditional honeypots behavior, i.e., to wait passively for scans, attacks and compromise, honeyclients[9] or client-side honeypots operate by actively searching for malicious content. The main characteristic of honeyclients is to simulate human behavior like executing e-mail attachments (presentations, pictures, etc.) or following URLs.

Honeyclients also can be divided into the same categories as classical honeypots: low interaction, that simulate some actions in order to download and analyze malicious code, and high interaction, that are fully deployed computers with real operating system and services offered. Some examples of honeyclients are:

- HoneyC[10], a low interaction client-side honeypot that emulate web clients to dynamically request responses from a server to find malicious ones in a network;
- PhoneyC[11], that analyzes JavaScript and VisualBasic Script to decode and extract exploits from browser script codes;
- Capture-HPC[12], a high interaction honeyclient interacting with potentially malicious servers and detecting target system changes by observing it in virtual machine environment.

3. ENVIRONMENT SETUP AND PRE-CLASSIFICATION SCHEME

As shown in previous section, there are some tools being used to collect malware from different sources, as web pages, e-mails or Internet traffic. If we explore the potential of these tools by combining them, it would be possible to improve the collection process and obtain statistical information about the process and the quality of samples collected. In this section, we discuss the system architecture to collect malware from two sources - sent by e-mail and which are searching specific vulnerabilities and attacking systems over the Internet - explaining the modules and the tools that compose it. The system architecture is illustrated in Figure 1.

The system is divided into three subsystems (Distributed collectors, e-mail analyzer, and centralized storage & pre-classification), which are described in the subsections below.

3.1 Distributed Collectors

As honeyd[13] can mimic different operating systems stacks, receive packets destined to any service and protocol, including the establishment of TCP sessions, and answer for multiple IP addresses, it is the best option for deployment of sensors to receive connections. However, for the purpose of malware collection we need a higher level of interaction, i.e., a tool that can be exploited by the malware attack vector, download and store the sample. Nepenthes accomplish this goal, handling the steps of downloading, logging and storing unique malware accordingly to the MD5 hash of the sample. Inspired by Brazilian Distributed Honeypots Consortium[14], we use some honeyd honeypots to receive the initial packets of the attack and then proxy all connections to a Nepenthes sensor, in a way it interacts with the attack vector and download the malware sample. To do this, honeyd is installed in an openBSD operating system and configured in proxy mode. Nepenthes is installed also in a machine running OpenBSD, but in another computer to receive connections forwarded from all honeyd sensors. Figure 2shows the honeyd.conf file parameters used to make honeyd operate in proxy mode, forwarding connections to certain vulnerable services to Nepenthes.

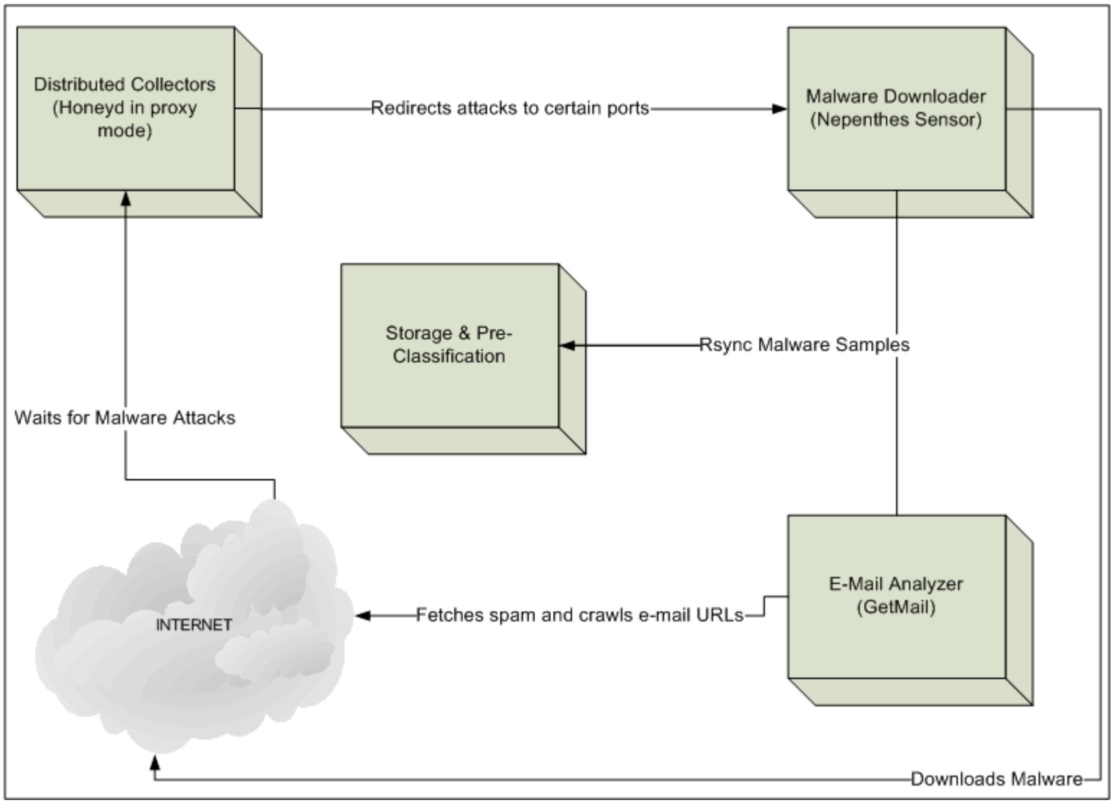
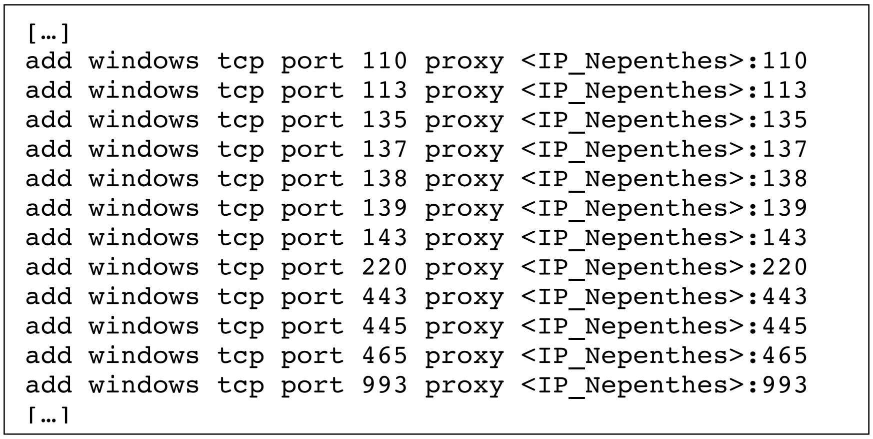

Distributed Collectors (Honeyd in proxy mode)
Redirects attacks to certain ports
Malware Downloader (Nepenthes Sensor)
Storage & Pre-Classification
Rsync Malware Samples
Waits for Malware Attacks
INTERNET
Fetches spam and crawls e-mail URLs
E-Mail Analyzer (GetMail)
Downloads Malware

Fig. 1. Architecture of distribute collection system, where each box represents a subsystem and the arrows represent actions performed by the subsystems.



```
[…]
add windows tcp port 110 proxy <IP_Nepenthes>:110
add windows tcp port 113 proxy <IP_Nepenthes>:113
add windows tcp port 135 proxy <IP_Nepenthes>:135
add windows tcp port 137 proxy <IP_Nepenthes>:137
add windows tcp port 138 proxy <IP_Nepenthes>:138
add windows tcp port 139 proxy <IP_Nepenthes>:139
add windows tcp port 143 proxy <IP_Nepenthes>:143
add windows tcp port 220 proxy <IP_Nepenthes>:220
add windows tcp port 443 proxy <IP_Nepenthes>:443
add windows tcp port 445 proxy <IP_Nepenthes>:445
add windows tcp port 465 proxy <IP_Nepenthes>:465
add windows tcp port 993 proxy <IP_Nepenthes>:993
[…]
```

Fig. 2. Honeyd configuration parameters done in honeyd.conf to act as a proxy for malware connections. Here, the "proxy" directive forwards tcp connections to defined ports to the IP address of the Nepenthes sensor (<IP_Nepenthes>), which downloads the malware, if any available.

3.2 E-mail Analyzer

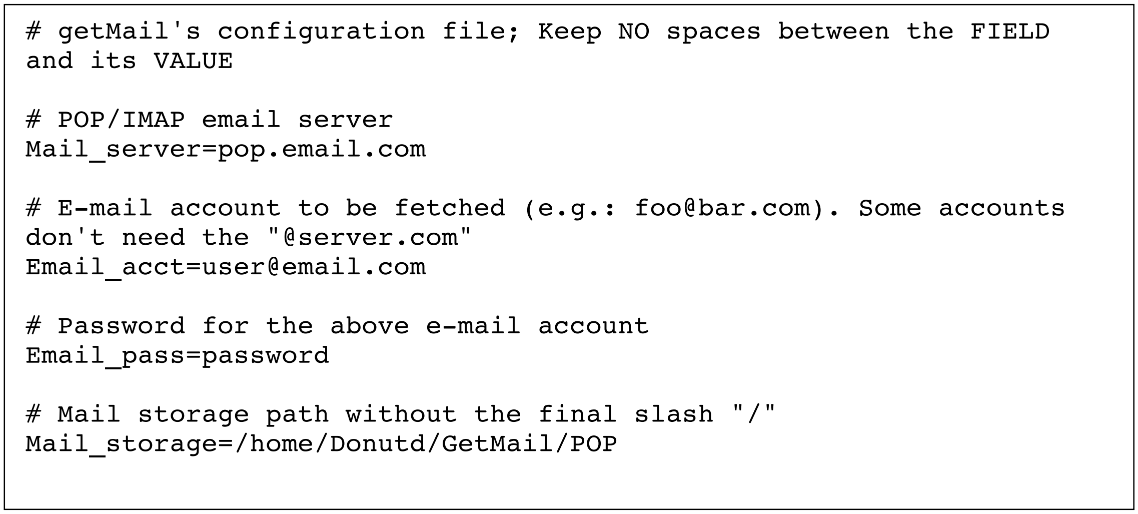
Many attacks from today are accomplished due to users interaction, either executing an attachment, or following interesting links sent by e-mail. In order to get those e-mail malware, we developed *Donut*, composed by a main daemon and the *GetMail* tool. The main daemon, called *Donutd*, is a process that runs in background and has the function of read a configuration file to get all variable values and makes a call to *GetMail* tool, passing the obtained parameters.

GetMail is a tool written in python to download spam and aggregate data to the malware collection architecture defined in this paper. The purpose of this tool is, in a practical manner, analyze spam, obtain malicious code and store in the storage module messages, attachments and files fetched from suspicious URLs from a specific e-mail account.

This tool was developed modularly, in a way that we can easily add functionalities and improve it. It is divided into some scripts and initially one configuration file, as listed below:

- email.config;
- getMailPOP.py;
- getMailIMAP.py;
- unpack.py.

For the proper working of the tool, some details of the email account that receives spam should be defined in the email.config file. The template for this configuration file must be strictly followed as seen in Figure 3.



```
# getMail's configuration file; Keep NO spaces between the FIELD
and its VALUE

# POP/IMAP email server
Mail_server=pop.email.com

# E-mail account to be fetched (e.g.: foo@bar.com). Some accounts
don't need the "@server.com"
Email_acct=user@email.com

# Password for the above e-mail account
Email_pass=password

# Mail storage path without the final slash "/"
Mail_storage=/home/Donutd/GetMail/POP
```

Fig. 3. Parameters of GetMail tool configuration file.

The two main modules of GetMail are *getMailPOP* and *getMailIMAP*, which one being responsible for connecting to an e-mail server using POP3 or IMAP under an SSL connection. Each message obtained is stored in a separate directory, using the following standard: *Mail_storage/msgNUM1/attNUM2* where *Mail_storage* is the local machine path information defined in the configuration file, *NUM1* is the message number for accounting purposes and *NUM2* is the attachment number from that message, in the case of several attachments.

Finally, the *unpack.py* module contains the functions responsible for separating the attachments of the message body and stores them for further analysis. This module search for links in the message body and if some is found, it accesses the link and obtains the file available. It also obtains the type of the file and the possible packer used to compress it.

3.3 Storage & Pre-classification

After all malware is collected in a Nepenthes collector, it is sent to another computer to be separated and stored. All malware from Nepenthes is sent to this subsystem via rsync and analyzed initially with ClamAV solution. ClamAV identifies it as a malicious code or not and accordingly to these scan results, malware is copied into a directory containing a label that preclassifies it. To accomplish it, a python script was written to parse the output of ClamAV and identify some predefined classes, such as Bankers, Trojans, Bots, Downloaders, Worms, Others and Unidentified. Once ClamAV scan is done and the results are logged, if the file contains a malware within the classes determined by the script, it is copied to its respective directory, forming a kind of database of different malware families. This module is deployed in the same machine that downloads spam and analyzes it, serving as a point of convergence to all obtained malware samples.

Thus, the system combines active and passive malware collection, where spams are analyzed and can have attachments or URLs that lead to new malware specimens, with the e-mail analyzer doing the active part and the distributed collectors passively wait for attacks and redirect attacks to some ports to a the malware downloader sensor. At the end, all collected malware is sent to a storage to pass through an antivirus scanning and then be separated under classes.

4. RESULTS AND FINAL CONSIDERATIONS

In this section some results from the test of the system are described, as well as information on the environmentused for tests. In order to do the comparisons, we used initially a single Nepenthes sensor to collect malware in an IP range of a network "A". After one-month period, we put a honeyd sensor in an IP range of a network "B", with approximately 70 IP addresses emulating Windows profiles and proxying connections to the previously mentioned Nepenthes as proposed in this paper. The collection results are shown in Figure 4.

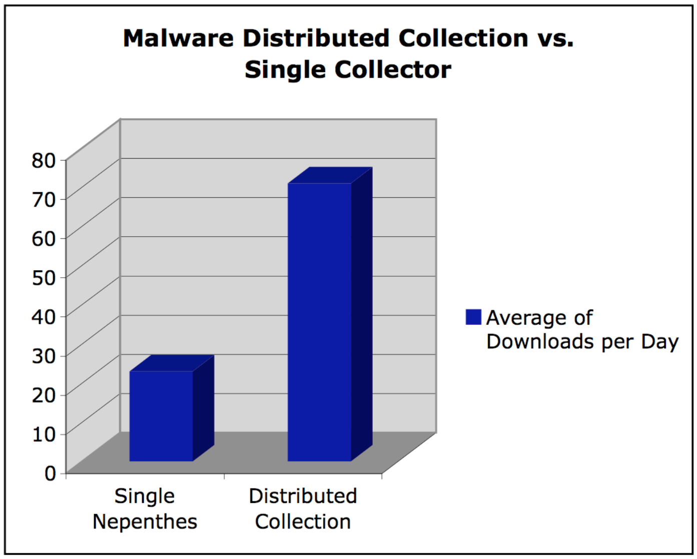


Fig. 4. Average of malware collected using a single nepenthes sensor and an architecture of distributed sensors forwarding attacks to this same single nepenthes sensor.

Considering that Nepenthes technology was developed to avoid downloading of repeated malware, using the distributed collection improved the collection of unique samples almost three times. To verify it, we get 100 (a hundred) malware collected and analyzed by our system and the results are in the Figure 5.

Those labeled "Identified" are malware recognized by ClamAV, while the other ones were unidentified. Those unidentified were manually analyzed and classified as "Unknown" malware, which didn't have a signature in the antivirus at the moment of analysis, or "Corrupted", which are files that couldn't be completely downloaded by the system. Corrupted ones were discarded from the storage.

Identified malware was separated in different classes by the pre-classification script. Based on Nepenthes mechanism, they must all be unique samples. Verifying the samples stored in "Backdoor" directory, it was observed that four samples, referred respectively by their MD5 – "5865-e732-663d-75b5-01ff-d7d9-8bc4-9005"; "6525-8e79-ee78-ac97-dc33-dd3a-7cd4-82ff"; "9a11-9463-e60c-ed44-88e4-309f-37dd-2d02" and "c819-d774-6a18-cbca-ca2f-f066-a310-86e9" – were identified as "*Trojan horse BackDoor.RBot.BI*". When those samples were executed, it was stated that they are exactly the same malware, differing by the IP addresses they tried to connect.

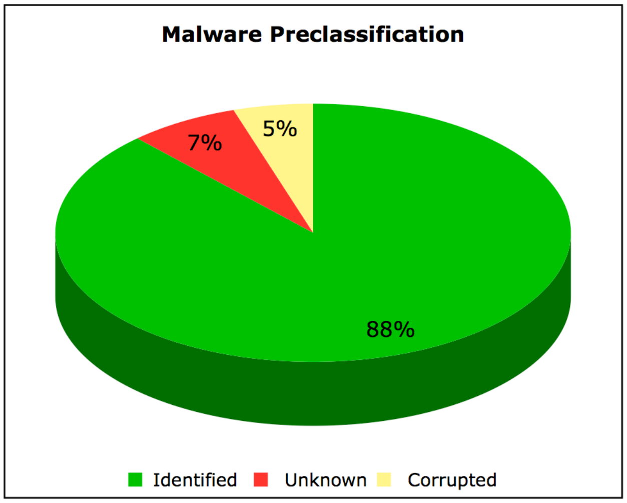


Fig. 5. Status of a sample of malware pre-classified under the proposed system scheme, where "Identified" means malware that has an antivirus signature, "Unknown" means files don't recognized as malware by the antivirus and "Corrupted" means broken binaries.

Existence of similar malware with different MD5s poses initially a problem, related to uniqueness. This one is just a problem in a statistical way as it can result in a wrong counts if we want to measure unique samples collected based on the MD5 attributed to them. This problem is easily solvable if the amount of unique malware is counted based on the antivirus identification (and ignoring misidentifications). Another issue is caused by the lack of a standard in antivirus classification. Different antivirus can identify the same sample as a different class, for example, based on ClamAV identification, the previously mentioned malware instances were copied into three different locations: Trojan, Backdoor and Bot directories. One way to try to solve it is to deploy more antivirus brands and separate it based on the recurrent identification, i.e., as more antivirus programs classify the sample in the same class, this sample will then be copied to this class directory and the other possible classes will be ignored.

As observed in this paper, there are many issues to be considered about the subject of malware collection. It has been noted that collection directions are pointing to active ones, i.e., tools that search for malware such as our GetMail tool. Passive collection in general has less groundbreaking results, as most of tools are just able to collect known pieces of code while active collection can provide some 0-day malware. Otherwise, it is important to have known malware in order to have enough samples of similar species to derive the main features that identifies each class. In the system proposed in this paper, we combine the two collection methods to have better results, distributing the sensors in a way we can get more samples at the same time we can observe malware comproníse trends. The advantages of distributed collection that could be seen in this paper are mainly:

- Better use of computer resources, as we can use low interaction honeypots to redirect attacks and gather more value from them by the download of the malware in the collection sensor.

- Trend analysis of spreading and compromise of malware, as we have more sensors geographically distributed. This way, it is possible to verify if a collected malware that attacked some services on a specific network is the same sample that attacked the same services on another network or is a variant.

Considering that this work is one of the pieces of a bigger project aiming malware classification, the system can provide resources to this further classification. This can be done by the extraction of features from the samples of each labeled directory, in order to have behavior patterns to classify the samples stored in the "Unidentified" directory.

5. ACKNOWLEDGEMENTS

Many thanks to Guilherme Paulovic for the initial coding of GetMail tool and to Dr. Antonio Montes from Information Technology Center Renato Archer (CTI) for the samples and access provided to their systems.

REFERENCES

[1] Grossman, J and Niedzialkowski, T.C. "Hacking Intranet Websites from the Outside – Javascript malware just got a lot more dangerous". Black Hat (USA) – Las Vegas, 2006. Available at: http://www.blackhat.com/presentations/bh-usa-06/BH-US-06-Grossman.pdf

[2] Whitehouse, O. "An Analysis of Address Space Layout Randomization on Windows Vista". Symantec Advanced Threat Research, 2007. White paper available at: http://www.symantec.com/avcenter/reference/Address_Space_Layout_Randomization.pdf

[3] Whitehouse, O. "Analysis of GS Protections in Microsoft Windows Vista". Symantec Advanced Threat Research, 2007. White paper available at: http://www.symantec.com/avcenter/reference/GS_Protections_in_Vista.pdf

[4] McDermott, J.; Fox, C. "Using abuse cases models for security requirement analysis". Proceedings of the 15th Annual Computer Security Applications Conference, p.55, 1999, IEEE Computer Society, ISBN:0-7695-0346-2.

[5] Nepenthes – Finest Collection. Available at: http://nepenthes.carnivore.it. (Accessed on January 2009).

[6] Baecher, P. et al. "The Nepenthes Platform: An Efficient Approach to Collect Malware", Recent Advances in Intrusion Detection, Springer Berlin / Heidelberg. Pages 165-184. 2006.

[7] Honeytrap. Available at: http://honeytrap.mwcollect.org/. (Accessed on January 2009).

[8] Zhuge, J.; Holz, T.; Han, X.; Song, C. and Zou, W. "Collecting Autonomous Spreading Malware Using High-interaction Honeypots". Proceedings of 9th International Conference on Information and Communications Security (ICICS'07), Zhengzhou, China, December 2007.

[9] Provos, N and Holz, T. "Virtual Honeypots: From Botnet Tracking to Intrusion Detection". Addison Wesley, 2007. ISBN: 0-321-33632-1.

[10] Seifert, C., Welch, I. and Komisarczuk, P. "HoneyC - The Low-Interaction Client Honeypot". Proceedings of the 2007 NZCSRCS, Waikato University, Hamilton, New Zealand, April 2007.

[11] Nazario, J. "Phoneyc". http://svn.carnivore.it/browser/phoneyc. (Accessed on January 2009).

[12] Seifert, C.; Steenson, R.; Holz, T.; Yuan, B. and Davis, M.A. "Know Your Enemy: Malicious Web Servers". Available at: http://www.honeynet.org/papers/mws. (Accessed on January 2009).

[13] Spitzner, L. "Honeypots: Tracking Hackers". Addison Wesley, 2002. ISBN: 0-321-10895-1

[14] Brazilian Honeypots Alliance – Distributed Honeypots Project. Available at: http://www.honeypots-alliance.org.br/. (Accessed on January 2009).