Ghost turns Zombie:
Exploring the Life Cycle of Web-based Malware

Michalis Polychronakis[*] *Panayiotis Mavrommatis*[†] *Niels Provos*[†]

Abstract

While the web provides information and services that enrich our lives in many ways, it has also become the primary vehicle for delivering malware. Once infected with web-based malware, an unsuspecting user's machine is converted into a productive member of the Internet underground. In this work, we explore the life cycle of web-based malware by employing light-weight responders to capture the network profile of infected machines. Our results indicate that web-based malware provides a cornerstone for large scale electronic fraud. It is used to exfiltrate address books of compromised machines creating databases of hundred millions of email addresses, to form spamming botnets responsible for a significant fraction of spam currently seen on the Internet, and also to steal login credentials that can be directly monetized or leveraged to turn more web servers into malware delivery vectors.

We support our findings by providing a broad overview of the post-infection network behavior of web-based malware, as well as in-depth examinations of the botnets and leaked information we found during the course of our study.

1 Introduction

A thriving Internet underground [6] has grown up in the past several years, employing the hundreds of thousands of malware infected commodity PCs to provide an infrastructure for conducting a wide range of criminal enterprises. Such activities range from carrying out electronic fraud through phishing, to sending out billions of spam email messages.

In previous work, we examined how vulnerable computer systems become infected with malware simply by browsing the web [10, 11]. Our analysis was based on detecting *drive-by* downloads on billions of web pages. In a drive-by download attack, a malicious web page exploits a vulnerability in a web browser, media player, or other client software to install and run malware on the unsuspecting visitor's computer. By loading suspicious web pages with a web browser inside a virtual machine, we found several million web pages capable of compromising vulnerable computer systems.

While prior research has tried to understand the behavior of individual malware binaries, little work has been done to understand the network-level behavior of a large population of computer systems infected with a diverse set of web-based malware. To better understand this issue, we instrumented our virtual machines with light-weight responders to capture and respond to any network payloads sent to the Internet by malware installed as a result of a successful drive-by download attack. Our responders are capable of emulating HTTP, IRC, SMTP and FTP sessions. For any protocols not directly emulated, we also built a generic responder that captures all other payloads.

Over the course of two months, we collected over $448,000$ responder sessions. We subsequently analyzed these sessions, looking at overall trends and performing several in-depth case studies. Our analysis organizes malware's behavior into three categories: *propagation*, *data exfiltration* and *remote control*. We show how these aspects taken together provide a compelling perspective on the life cycle of web-based malware. The observed malware activities range from capturing email addresses from compromised machines, to joining infected systems into botnets responsible for operating large-scale spam campaigns.

Furthermore, the botnets created by web-based malware are not only controlled via traditional mechanisms such as IRC, but often employ other protocols such as HTTP. Our in-depth examinations turned up a variety of interesting trends, including rich data exfiltration activity and the use of custom protocols for command and con-

[*] FORTH-ICS, Greece, email: `mikepo@ics.forth.gr`

[†] Google Inc., email: `{panayiotis,niels}@google.com`

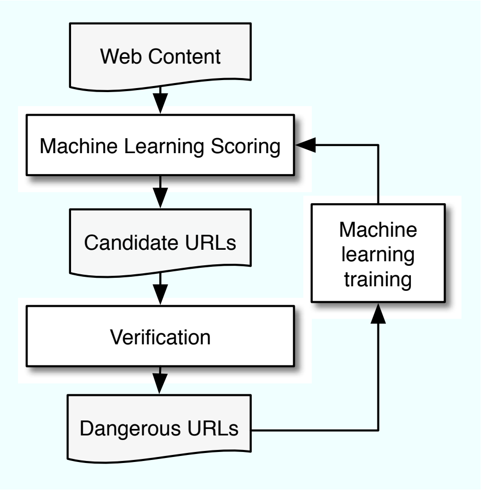

Figure 1: Overall system architecture. Using machine learning techniques, suspicious URLs are selected among billions of web pages for verification in a virtual machine.

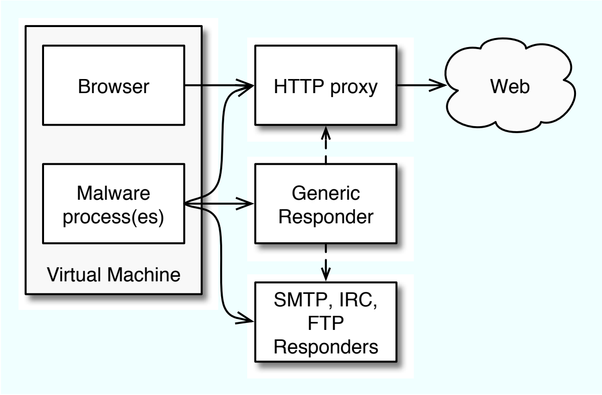

Figure 2: Network flow in the verification component. Outgoing connections to known ports are forwarded directly to the corresponding responder or the HTTP proxy. All other connections are handled by the generic responder, which can potentially identify and hand off connections that use any of the emulated protocols to the appropriate responder.

trol communication. We believe these results provide the first large scale empirical look at web-based malware.

The remainder of the paper is organized as follows. In the next section, we provide a brief overview of our malware analysis and collection architecture. In Section 3, we investigate the life-cycle of web-based malware based on broad trends in our data, and discuss in-depth case studies of botnets and data exfiltration activities. Finally, we discuss related work in Section 4 and conclude in Section 5.

2 System Architecture

We build upon a scalable system developed with the goal of detecting harmful URLs on the web. In this section, we describe our extensions to this system to collect and analyze malicious network activity occurring after infection. To provide context for our research, we first give a brief overview of the overall system described in depth in prior work [10]. This is followed by a detailed description of the light-weight responders which allow us to automatically capture the network-level activity of drive-by downloads.

2.1 Overview

Our system consists of an efficient first-pass filter followed by a verification component. Figure 1 provides an overview of the system components and their interaction. The first-pass filter is essentially a *mapreduce* [5] over billions of web documents. For each web page, we extract several features, including links to known malware “distribution” sites, suspicious HTML elements, or the presence of code obfuscation. The combination of these features is scored by a model trained on a specialized machine-learning system [3]. URLs with a high score are considered potentially malicious, and are submitted to the verification component for further analysis. The URLs that are verified to be malicious are then exported to Google Web Search and, via the Google Safe Browsing API [1], to other clients. The verification results are also used to retrain the machine learning model.

To verify if a candidate URL is indeed malicious, we have deployed a network of client honeypots based on virtual machines running Windows and Internet Explorer. To start the verification of a URL, we load it with Internet Explorer, which is configured to use a web proxy running outside the VM. The proxy records all HTTP requests and scans all HTTP responses using several anti-virus engines. New processes, file system changes, and registry modifications are monitored from within the VM. Upon infection, we typically find abnormal activity in all of the monitored areas. A combination of these signals is used to decide whether the URL is malicious or not.

2.2 Responders

To better understand the purpose and effects of web malware, we extended the verification component with light-weight responders, which provide fabricated responses for commonly used protocols such as SMTP, FTP and IRC. Upon infection of the virtual machine, any traffic to standard ports is forwarded to the appropriate handler, as shown in Figure 2: web traffic is handled by the HTTP proxy, port 25 traffic is redirected to the SMTP responder, and so on.

Since malware may communicate over non-standard ports or using custom protocols, all outgoing traffic towards any port other than the standard ports of the emulated services is redirected to a *generic responder*. The purpose of the generic responder is twofold: i) to hand off connections over non-standard ports that use one of the emulated protocols to the appropriate responder, and ii) to elicit further information about the malware from connections to non-emulated services, or connections that use unknown protocols.

Upon receiving a connection, the generic responder attempts to heuristically identify the application-level protocol used. For client-initiated protocols, in which the client first sends a message to the server (e.g., HTTP, IRC), the generic responder can determine which service it should emulate by looking in the client's message for known protocol keywords and structure.

For protocols in which the server initiates the dialog by sending a message to the client (e.g., SMTP, FTP), the responder acts as follows: after it accepts a connection, if no data is received within a few seconds, the responder assumes that the client is waiting for a message from the server. In that case, the responder blindly initiates the dialog by sending a generic welcome banner message. Depending on the client's response, the generic responder can identify the actual protocol being used and hand off the connection to the appropriate emulated service. As discussed in Section 3, the generic responder successfully identified many HTTP and IRC connections over non-standard ports and handled them appropriately.

The generic responder is useful even if the malware uses unsupported or unknown protocols. Without the responder, we would observe just a connection attempt, with no transmitted data. However, by accepting all TCP connections to any port, the generic responder can potentially elicit the first application-level message sent by the malware, which may provide useful insight about the intended activity. Indeed, as discussed in the following section, the generic responder captured several sessions of custom malware communication protocols over arbitrary ports.

3 Life Cycle of Web-based Malware

Once web-based malware infects a computer, it often interacts with other hosts on the Internet to either report information about the compromised system, or to receive instructions for further actions. For example, a newly infected computer is likely to become part of a botnet. Web-based malware may also download other malware components, report stolen information and credentials, or attempt to propagate further.

In the following, we provide a large-scale analysis of post-infection malware activity as observed by our lightweight responders. We aim to provide insight into what happens after a vulnerable system gets infected as a result of visiting a malicious URL, and how it interacts with the Internet. We do not study malware in isolation, but as it behaves after compromising a typical user's system running as a virtual machine. For example, in most drive-by downloads, multiple malware components are being installed at the same time.[1]

Our network-level analysis of malware, i.e., malware's interaction with other hosts and our responders, is organized into three categories: *propagation*, *data exfiltration* and *remote control*. As we explore the post-infection activity, we show how these behaviors taken together provide revealing insights into the life cycle of web-based malware.

3.1 Data Set

Our analysis covers a two-month period, from January 17, 2008 to March 25, 2008. During this period, our virtual machines analyzed URLs from $5,756,000$ unique hostnames—we report on unique hostnames instead of unique URLs, as URLs from the same host usually install the same set of malware. There were $307,000$ hostnames serving at least one harmful URL, while $152,700$ of these sites (49%) had URLs that resulted in HTTP requests initiated from processes other than the web browser. About $18,000$ sites (5%) had URLs that triggered responder sessions.

Across all URLs, the total number of responder sessions with transmitted data exceeded $448,000$. There were many more cases where malware made network connections without transmitting any data. For example, we observed connections to popular SMTP servers without actual data transmission. We speculate that these are attempts to test the victim's firewall configuration or Internet connectivity.

3.2 Network Characteristics

We begin our analysis by providing high-level statistics about the overall post-infection network activity of the analyzed URLs.

Figure 3 presents the destination port distribution of all outgoing connections from the virtual machine upon infection. For each port, we report the number of unique

[1] We hypothesize the existence of malware that relies on the installation of multiple components for becoming fully functional. Its purpose might not be discovered by analyzing individual binaries in isolation.

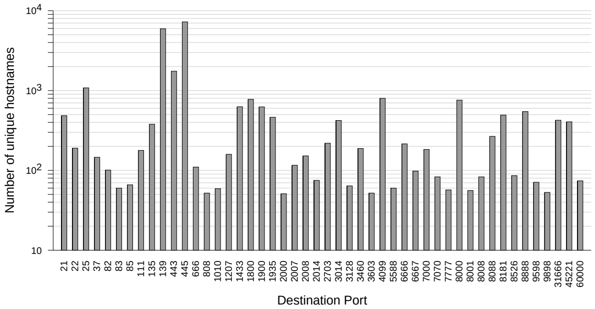

Figure 3: Port distribution of outgoing connections (excluding HTTP/80). For clarity, ports with activity related to less than 50 unique host names have been omitted.

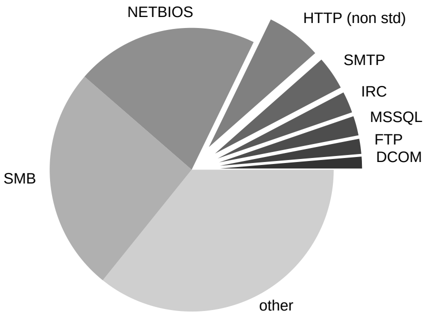

Figure 4: Network protocols used by malware (excluding HTTP/80).

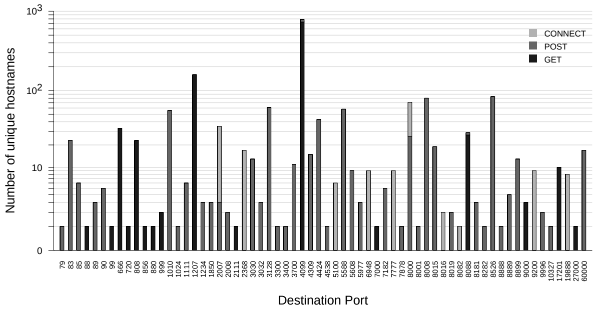

Figure 5: Destination port distribution of HTTP connections destined to non-standard ports, for different HTTP request methods.

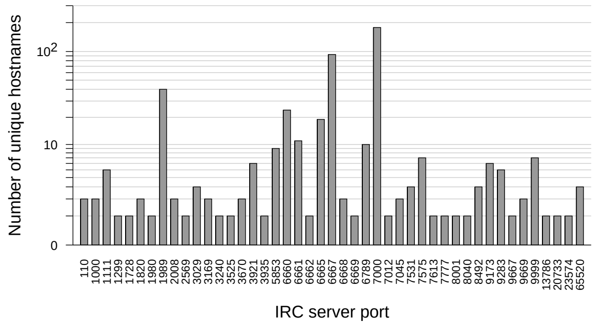

Figure 6: The distribution of IRC server ports contacted by web-based malware.

host names on which we found at least one URL installing malware that transmitted data to that port. This removes any bias from host names for which the system happened to analyze multiple URLs. A total of 416 different destination ports were contacted and is indication of the diverse and obscure nature of malware's post-infection network behavior.

Figure 4 shows the network protocol distribution of the observed outgoing connections. The classification was made using payload inspection, without considering the destination port number. Potential reasons for the increased diversity in destination port numbers are custom protocols as well as standard protocols over non-standard ports — probably for making the purpose of the traffic less obvious. For example, in addition to port 80, we witnessed HTTP connections to 63 other port numbers. Similarly, we found malware communicating with IRC servers on 44 different ports.

The exact distribution of HTTP and IRC connections according to the destination port number is shown in Figures 5 and 6, respectively. Most of the HTTP connections to non-standard ports are either GET or POST requests. Using technologies such as PHP and JSP, malware authors can implement flexible communication mechanisms for sending commands or updates and retrieving information from the infected hosts. We also witnessed some CONNECT requests which probably correspond to probes for open proxies that support the CONNECT method for constructing tunnels. About one third of the connections, corresponding to the *other* category in Figure 4, used less popular or unknown protocols to 317 different destination ports.

3.3 Discovery and Propagation

One of the most common network activities of malware upon infecting a host is to scan for other vulnerable systems, either in the same LAN or the Internet, to further propagate. As shown in Figure 4, we observed a significant number of network connections using common Windows protocols. About half of the connections were to ports 139 (NETBIOS) and 445 (SMB), which are often related to exploiting Windows vulnerabilities. Ports 135 (DCOM) and 1433 (MSSQL) are also commonly associated with exploits against Windows and Microsoft SQL servers, respectively.

As our responders do not emulate these protocols, we observed only the first protocol packet.[2] The majority of probes were to IP addresses on the same network with the virtual machine, which implies that most malware first scan neighboring computers, either for vulnerable services or network shares with unrestricted access.

3.4 Reporting home

Part of the malware life cycle consists of notifying its author upon successful installation. This activity accounted for the majority of the emails captured by our SMTP responders. Table 1 shows the most common email subjects we observed. The subjects of the captured emails signify this type of activity quite clearly: *XP Hacked*, or *Installation Report*. The email bodies usually contain further information about the victim's host, such as its IP address, access credentials, and the port numbers of any installed back doors.

Table 2 shows the SMTP domains most frequently used by malware to send installation reports. In most cases, the emails were sent to drop box accounts on popular free web mail services, as well as ISP mail services, usually employing several different SMTP servers for each service.

Subject	**# Messages**
`XP Hacked`	390
`ProRat [...]`	162
`Vip Passw0rds`	98
`Log file from ...`	82
`Installation report`	76
`Perfect Keylogger [...]`	47
`Installation on XP succeeded`	12
`E g y S p y KeyLogger [...]`	12
`INFECTADO`	6
`Mais 1:  XP`	3
`AVSXP`	3
`C-h-e-c-k-i-n-g:XP`	2
`...:Noticia quentinha de:...  XP`	2

Table 1: Top malware email subjects.

SMTP Server	**# Messages**
yahoo.com	436
google.com	118
tvm.com.tr	98
aol.com	82
hotmail.com	19
outblaze.com	8
globo.com	6

Table 2: Top second-level domains of the SMTP servers employed by malware.

The HTTP protocol is also frequently used to inform malware authors about infections. The following example shows a GET request made by the *DoDoLook* trojan that uses the MAC address (here sanitized) of the infected computer as an identifier for retrieving targeted updates:

```
GET /geturl.php?version=1.1.2&fid=7493&mac=00-00-00-00-
00-00&lversion=&wversion=&day=0&name=dodolook&recent=0
HTTP/1.1
Accept: */*
User-Agent: Mozilla/4.0 (compatible; )
Host: loader.51edm.net:1207
Cache-Control: no-cache
```

We also found malware utilizing custom communication protocols for reporting home. In some cases, the malware reported successful infections using a custom XML-like format. Specifically, the captured connections of this particular protocol were destined to 129 remote hosts using 29 different destination ports. Two examples of the transmitted data[3] are shown below:

```
HGZ5.<FT>2008-01-28 12:55:30</FT><IM>80</IM><GR>_&</GR>
<SYS>Windows XP 5.1</SYS>
<NE>XP</NE><pid>488</PID><VER>Ver1.22-0624</VER>
<BZ></BZ><P>1</P><V>0</V><IP>0.0.0.0</IP>

000......<LC></LC><GR>-</GR><IM>25</IM><NA>XP</NA>
<CS>English (United States)</CS><OS>Windows XP</OS>
<MEM>1024MB</MEM><CPU>2200 MHz</CPU>
<NET>LAN</NET><video>0</video><BZ>-</BZ>
```

The leaked information includes the IP address, OS version, country, and other machine properties. We observed several different instances in the above format with only slight variations in the field types and order. Another example of custom infection notification, observed only on port 6346, looked as follows:

```
105|OnConnect|United States|SYSTEM|XP - SYSTEM
|0.0.0.0|Not Detected|4.0.4 (BAZ)
|United States|OnConnect|
```

In this case, different fields are separated using pipe characters. There were also cases in which most of the contents of the captured stream consisted of binary data, with some interspersed ASCII strings representing machine information.

3.5 Data Exfiltration

Moving from reporting successful installations to exfiltrating more sensitive information is the next logical step in the malware life cycle. Many responder sessions contained signs of data exfiltration, including browser history files and stored passwords, usually captured by keyboard loggers or browser hooks. As one of the email subjects — `Vip Passw0rds` — indicates, SMTP is one method of achieving this goal. We observed several emails sending back stored passwords from the compromised machine.

[2] We plan on emulating more protocols as part of future work.

[3] Non-printable characters are represented as dots, and some of the information was masked.

The large number of POST requests in Figure 5 suggests that HTTP is also employed for sending sensitive information back to data collection servers. Moreover, almost all of the observed FTP sessions corresponded to uploads of harvested data. The malware connected to our FTP responder, supplying a login and password, and started uploading data. We were able to analyze the stolen information by accessing some of the FTP servers that were still operational using the malware's credentials.

In the following, we give a few examples of the types of information we found. Some of the servers functioned as drop boxes for exfiltrated email addresses from users' address books, organized in separate files according to the name of the computer from which they were harvested. One server had $4,729$ files containing more than $250,000$ addresses, all dated within two days of our inspection. This indicates that the server's administrators collect and remove the information regularly. However, it also means that malware authors have a supply of valid email addresses and even their social context readily available.

More sensitive information was found in extensive logs periodically uploaded by malware, containing the victim's *IP address*, *DNS server*, *gateway*, *MAC address*, *username*, as well as the *URL* and intercepted *form and password fields* of any HTTP request made by the user's machine. We analyzed over 250 MB of logs from a single server, extracting user names and passwords of 500 users for over 250 unique sites, including `myspace.com`, `yahoo.com`, `live.com`, `google.com` as well as many online banking sites.[4] Banking credentials can be monetized easily, while even the recently introduced two-factor authentication that relies on secure cookies cannot provide complete protection, since the adversary may decide to steal the cookie file, too. On the other hand, credentials to web content management systems can be used to turn more web servers into malware infection vectors.

3.6 Joining Botnets

Self propagation, reporting, and data exfiltration are disconcerting, but a more troubling aspect of malware lies in its ability to connect a compromised system to a network of bots that can be collectively controlled by a single entity. Command and control channels can be implemented using either well-known application-level protocols, such as HTTP and IRC, or through custom communication mechanisms. In the following, we give an overview of the different botnets we encountered.

IRC Server	# Sessions
irc.dal.net	109
undernet.irc.justedge.net	43
72.9.146.134.tailormadeservers.com	40
oslo.no.eu.undernet.org	35
42.32.1343.static.theplanet.com	14
dns2.labinahost.com	10
undernet.xs4all.nl	7
irc2.saunalahti.fi	7
Tampa.FL.US.Undernet.org	6
primescratchcards.com	6
w0rm.UnionIRC.Net	6

Table 3: Top IRC servers used by malware.

3.6.1 IRC Botnets

Internet Relay Chat provides the basis for the most commonly studied type of C&C communication. By joining a predefined channel, each bot can receive commands from its author and send back collected information. We observed IRC sessions to 90 servers, utilizing 1587 different nicknames in 95 channels. Table 3 presents the most frequently contacted IRC servers. As we can see from Figure 6, most of the IRC sessions use servers bound to well-known IRC ports, although there is a considerable number of IRC connections to non-standard ports.

We observed that some malware families use seemingly regular nicknames and channels on public and sometimes popular IRC servers. This saves the burden of running a dedicated IRC server, while at the same time offers some degree of concealment among legitimate users. On the other hand, we found cases using artificial nicknames, e.g., `[0]USA|XP[P]152102` or `Inject-2l087876`, that are usually unique to the infected host, and sometimes provide further information, such as the victim's IP address and OS.

3.6.2 HTTP Botnets

Among the HTTP-based botnets we observed, a case of particular interest involved a botnet that was used for orchestrating large-scale spam campaigns. Each bot periodically downloaded ZIP-archives with instructions on participating in spam campaigns using HTTP requests like the following:

```
GET /g/FA3521-9EE5C0-69ED87 HTTP/1.1
Host: 208.72.169.153
X-Flags: 0
X-TM: 34
[...]
```

Each response contained a ZIP-archive containing nine files with detailed instructions on how to participate in the spam campaign: `000_data22` - a list of domains and their authoritative name severs used to form the sender's email address, `001_ncommall`

[4] We reported the stolen credentials to Security Science who collaborate with the FTC and FBI as well as banks to protect customers whose credentials were exfiltrated.

Email Domain	Frequency
yahoo.com	28899
sbcglobal.net	14417
yahoo.co.uk	8939
shaw.ca	8321
hotmail.com	6985
korea.com	6041
yahoo.co.jp	5215
striker.ottawa.on.ca	4415
web.de	4276
yahoo.co.in	4200

Table 4: Top domains out of $700,000$ email addresses collected from a spam-sending botnet.

- a list of common first names used as part of the sender's email address, `002_otkogo_r` - a list of possible "from" names related to the subject of the spam campaign, `003_subj_rep` - a list of possible email subjects, `004_outlook` - the template of the spam email, `config` - a configuration file that instructs the bot how to construct emails from the data files, how many emails to sent in total, and how many connections are allowed at a given time, `message` - the message body of the spam campaign, `mlist` - a list of email addresses to which to send the spam, and `mxdata` - a binary file containing information about the mail-exchange servers for the email addresses in `mlist`.

We downloaded about 700 such ZIP-files, amounting to approximately $700,000$ different email addresses. Table 4 shows the most frequently occurring email domains. We reported our findings to another researcher[5] who provided us with a set of 250 million email addresses captured from the same botnet in only 24 hours. We noticed that the most frequent domains captured by us within an hour did not completely overlap with the larger data set. This indicates that the email addresses are not handed out at random.

As part of infecting the system, the malware attempted to install a malicious kernel driver named `ntio922.sys` but failed. To help the malware authors debug their software, the installer attempted to upload a *small memory dump* file containing a stack trace in case of a failed driver installation. To us, this indicates a high-level of sophistication on part of the malware authors.

3.7 Summary

Taking a step back, we outline how these individual pieces might fit into a much larger puzzle. Figure 7 shows what we deem to be the life cycle of web-based malware. The malware is seeded to millions of users from compromised web servers that infect new visitors. The infected PCs are transformed into a platform for conducting large-scale electronic fraud.

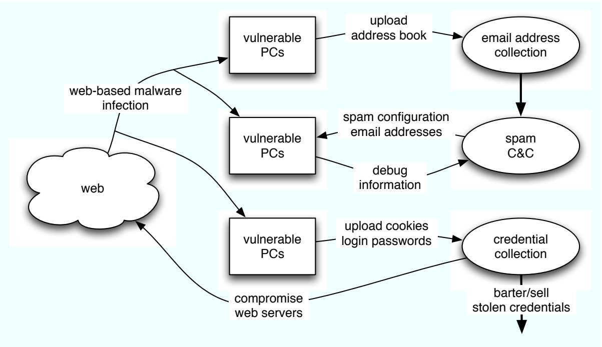

Figure 7: An overview of the malware life cycle based on observations from light-weight responders.

Stolen address books are used to create databases containing hundred millions of email addresses. This information, together with the computational resources provided by the compromised machines, is currently being used for sending a significant fraction of the email spam currently encountered on the Internet. With access to login credentials, the malware can potentially gain access to web servers which can be turned into additional malware delivery vectors. From this point of view, the life cycle of web-based malware may be self perpetuating.

4 Related Work

Virtual machines, virtual honeypots, and lightweight responders have been used by several researchers to capture and better understand attacks [2, 9, 12, 14]. Pang et al. [8] used active responders emulating protocols associated with commonly exploited services to elicit attack payloads from darkspace traffic. Our light-weight responders are similar in that they respond to network connections initiated by adversaries, in our case malware running inside a virtual machine. However, we use them to analyze the post-infection behavior of malware, rather than to capture new attacks.

Our previous work [10,11] analyzed the maliciousness of a large collection of web pages. Although we provided some details on the prevalence of malware, we did not give any insights on the network activities of malware once installed on a system. While there is already significant research on malware analysis [4, 7, 15], our analysis focuses on a large collection of web-based malware and provides insights into the activities of currently deployed malware. Our approach is much less sophisticated than analysis systems that employ whole-system emulation and information flow tracking, nonetheless using a very simple approach based on light-weight responders provides interesting insights when applied to a large

[5] The researcher prefers to remain anonymous.

collection of malware. CWSandbox uses a similar approach [13] but unlike our system it hooks the malware and emulates protocols inside the virtual machine.

5 Conclusion

Although malware analysis has developed into its own research area, resulting in increasingly sophisticated analysis techniques, we showed that simple approaches motivated by low-interaction honeypots can yield a surprising amount of information on malware's activities. We explored the life cycle of web-based malware by employing light-weight responders to capture the network profile of infected machines. Our responders are capable of emulating protocols such FTP, HTTP, IRC and SMTP as well as capturing payloads from any protocols not directly emulated.

We supported our findings by analyzing more than $448,000$ responder sessions collected over a period of two months. Our analysis divided malware's behavior in three different categories: propagation, data exfiltration and remote control. Our in-depth investigation of these areas allowed us to explore several different aspects of malware's life-cycle.

Besides notifying adversaries about compromised systems and exfiltrating sensitive data, web-based malware often joins compromised hosts into botnets. These botnets make use of traditional C&C communication via IRC, but are also using other protocols such as HTTP to establish communication to a C&C server. One of the botnets we analyzed is currently responsible for a significant fraction of spam on the Internet and demonstrated surprising sophistication, even going as far as providing malware developers memory dumps of failed installations.

In future work, we plan on extending the protocol emulation to more services and hope to increase our understanding of currently unclassified network communication. Furthermore, the light-weight responders may provide additional signals for determining whether a URL is indeed malicious, especially for cases where the process activity and malware scanning provide insufficient information.

Acknowledgments

The work of Michalis Polychronakis was supported in part by the project CyberScope, funded by the Greek General Secretariat for Research and Technology under contract number PENED 03ED440. M. Polychronakis is also with the University of Crete. We would like to thank Oliver Fisher, Fabian Monrose, Moheeb Rajab, and the anonymous reviewers for their valuable feedback.

References

[1] Google Safe Browsing API. `http://code.google.com/apis/safebrowsing/`.

[2] K. G. Anagnostakis, S. Sidiroglou, P. Akritidis, K. Xinidis, E. Markatos, and A. D. Keromytis. Detecting Targeted Attacks Using Shadow Honeypots. August 2005.

[3] J. Bem, G. Harik, J. Levenberg, N. Shazeer, and S. Tong. Large scale machine learning and methods. US Patent: 7222127.

[4] M. Christodorescu, S. Jha, S. Seshia, D. Song, and R. Bryant. Semantics-aware malware detection. *Security and Privacy, 2005 IEEE Symposium on*, pages 32–46, 2005.

[5] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Proceedings of the Sixth Symposium on Operating System Design and Implementation*, Dec 2004.

[6] J. Franklin, V. Paxson, A. Perrig, and S. Savage. An Inquiry into the Nature and Causes of the Wealth of Internet Miscreants. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, October 2007.

[7] A. Moser, C. Kruegel, and E. Kirda. Exploring multiple execution paths for malware analysis. *Proc. IEEE Symposium on Security and Privacy*, pages 231–245, 2007.

[8] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson. Characteristics of Internet background radiation. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement (IMC)*, pages 27–40, 2004.

[9] N. Provos. A virtual honeypot framework. In *Proceedings of the 12th USENIX Security Symposium*, pages 1–14, August 2004.

[10] N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose. All Your iFrames Point To Us. Technical Report provos-2008a, Google Inc, 2008. `http://research.google.com/archive/provos-2008a.pdf`.

[11] N. Provos, D. McNamee, P. Mavrommatis, K. Wang, and N. Modadugu. The ghost in the browser analysis of web-based malware. In *Proceedings of the First Workshop on Hot Topics in Understanding Botnets (HotBots)*, 2007.

[12] M. A. Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A Multifaceted Approach to Understanding the Botnet Phenomenon. In *Proceedings of ACM SIGCOMM/USENIX Internet Measurement Conference (IMC)*, pages 41–52, Oct., 2006.

[13] C. Willems, T. Holz, and F. Freiling. Toward Automated Dynamic Malware Analysis Using CWSandbox. *Security & Privacy Magazine, IEEE*, 5(2):32–39, 2007.

[14] V. Yegneswaran, P. Barford, and D. Plonka. On the Design and Use of Internet Sinks for Network Abuse Monitoring. In *Proceedings of the 7th International Symposium on Recent Advances In Intrusion Detection (RAID)*, September 2004.

[15] H. Yin, D. Song, M. Egele, C. Kruegel, and E. Kirda. Panorama: Capturing System-wide Information Flow for Malware Detection and Analysis. In *Proceedings of the 14th ACM Conference of Computer and Communication Security*, October 2007.