Take a Bite - Finding the Worm in the Apple

Martina Lindorfer, Bernhard Miller, Matthias Neugschwandtner, Christian Platzer
Secure Systems Lab, Vienna University of Technology
{mlindorfer,bernhard,mneug,cplatzer}@iseclab.org

***Abstract*—When it comes to security risks, especially malware, Mac OS X has the questionable reputation of being inherently safe. While there is a substantial body of research and implementations dealing with malware on Windows and, more recently, Android systems, Mac OS X has received little attention so far.**

To amend this shortcoming, we built a Mac OS X based high-interaction honeypot and used it to evaluate over 6,000 blacklisted URLs to estimate how widespread malware for Mac OS X is today. We further built a dynamic analysis environment and analyzed 148 malicious samples to gain insight into the current state of Mac OS X malware. To the best of our knowledge, we are the first to tackle this task.

I. Introduction

The crumbling hegemony of Microsoft Windows-operated computers comes with various side effects. On one hand, this development provides other operating systems with the opportunity to gain a foothold in the ongoing contest for market shares. On the other hand, malware writers are called to attention. The decision if a possible target is worth pursuing is a simple bill of cost-effectiveness [1]. When the market share of an OS and thus the potential for monetization exceeds a certain tipping point, miscreants will invest in developing malware for this platform. So far this trend is evident in most environments, with Google's Android OS being the most popular amongst them. However, while malware analysis and defense strategies for Android-based devices are well under way, the third big player when it comes to consumer electronics is curiously mostly ignored: In 2012, 24% of the estimated 1.07 billion consumer computing devices were shipped by Apple [2]. Further estimates state, that the current market share of Mac OS X amounts to almost 9% [3].

While iOS for mobile devices is arguable an undesirable target for malware writers due to its restricted and confined nature, devices running OS X are vulnerable to all kinds of attacks, including drive-by downloads, scareware or malware-infected executables downloaded from arbitrary sources. Although all major companies aim to confine their PC-systems with their own app stores (e.g. Apple App Store and Microsoft Windows Store), these operating systems are still general-purpose and therefore hard to control completely. Additionally, Mac users are often more gullible because unlike Windows users, they are not sensitized to being the target for malware writers [4]. Still, research in this direction is sparse, even if the need to act has well been established [5], [6]. In this paper, we provide an analysis on the threat level OS X users are currently facing, starting from possible ways of infecting their devices up to a dynamic analysis of currently available OS X malware.

In summary, we make the following contributions:

- We present *iHoneyClient*, a VirtualBox-based high interaction honeypot capable of automatically investigating and downloading OS X binaries.
- We evaluate a set of 6,028 blacklisted URLs and examine their threat level for OS X users.
- We present a dynamic analysis environment, capable of automatically deploying and executing OS X binaries.
- We provide an overview of currently circulating OS X malware and evaluate our approach based on results created from human analysts.

II. Technical Background

In this section we provide a short overview of existing OS X malware and how it infiltrates the target system. Furthermore, we discuss approaches to malware analysis on other platforms.

A. Mac OS X Malware

Targeted attacks aside, the most notorious and wide-spread Apple malware to date is Flashback (also named Flashfake), a Trojan horse specifically targeting OS X. First spotted in the wild in 2011, Flashback imitated an installer for Adobe Flash. In early 2012, the Flashback developers changed the attack vector from social engineering to drive-by downloads, exploiting several Java vulnerabilities. Most important among these is *CVE-2012-0507* [7], which uses insufficient type checking to break out of the Java sandbox. This bug was already fixed by Oracle a month before the Flashback developers started using it. However, Apple had not yet merged the fix into their Java version, leaving OS X users vulnerable. To help spread Flashback, the attackers modified tens of thousands of WordPress blogs to include a malicious script and managed to infect approximately 700,000 computers worldwide [8], [9].

Flashback not only shows the potential for mass malware on OS X, it also shows a level of sophistication comparable to recent Windows outbreaks. It makes use of the latest malware technologies: self-decrypting program code, zero-day vulnerabilities, utilizing publicly available services to manage the botnet, multi-layered domain generation algorithms, self-protection mechanisms, sound authentication procedures and strong cryptography. Furthermore, its authors monetized the infected hosts through click fraud. The incident also shows that Apple's update policy may lead to very serious problems in the future.

B. Malware Analysis

In order to protect users from malware, researchers and anti-malware companies need a thorough understanding of the functionality, capabilities and purpose of malware samples. This insight can be gained by analyzing malware samples either statically or dynamically.

Static analysis examines the malicious program without executing it. The most widely used technique for static analysis is pattern matching, an integral part of most antivirus scanners. Additionally, they use heuristics in order to detect new malware. However, this approach can be circumvented by simple code transformations and binary obfuscation. Moser et al. [10] showed that static analysis is an NP-hard problem, thus rendering static analysis unfeasible in the long run.

In contrast, dynamic analysis relies on monitoring the behavior of a malware sample while it is executed in a controlled environment. Instead of analyzing the program instructions, the behavior of the program under analysis is monitored and evaluated. Several applications have been developed based on dynamic analysis sandboxes to monitor botnet traffic [11], obtain unpacked and unencrypted malware samples [12], [13], automatically obtain malware mitigation procedures [14] and detect unknown evasion techniques [15]. Several such tools exist today for both, research and commercial purposes. However, most of them simulate a Microsoft Windows environment. A comprehensive survey of dynamic analysis tools and techniques was given by Egele et al. [16].

With the possibility to analyze a binary, there is still the need to gather samples in the first place. Honeypots are the tool of choice for that purpose. They exist in a variety of forms: They can simulate services, data, computers, devices, clients or whole networks to attract the attention of miscreants and gather information about attacks.

While many honeypot implementations are passive, waiting for attackers to target them, client honeypots take an active role. They initiate communication with a server, e.g. by opening URLs in a web browser. Client honeypots typically consist of three components:

Queuer: The queuer is responsible for generating a list of targets for the visitor to interact with. Several techniques exist to create URL lists, among them search engines, blacklists, crawlers and feedback from the analyzing component.

Visitor: The visitor's task is to interact with the server. After receiving targets from the queuer it communicates with the given target, e.g. by opening a URL in a web browser. Often, the visitor is contained in a virtual machine to avoid infecting the analysis environment.

Analyzer: After the communication with the server is finished, the analyzer examines the new state of the visitor, respectively the data generated by it. It generates reports and can feed data back to the queuer.

An important characteristic of honeypots is the level of interaction they provide. High-interaction honeypots simulate a whole system, e.g. a host including its operating system and devices. In contrast, low-interaction honeypots simulate only one service, or parts of one service. The level of interaction is not only important for how authentic a honeypot appears to the attacker, but also affects its isolation and containment. Furthermore, the techniques used for analyzing the monitored behavior differs. While high-interaction honeypots tend to analyze state changes, low-interaction honeypots mostly employ pattern matching. This directly affects their ability to detect certain malicious activities like time bombs or changes to their environment [17]. These properties, combined with our prerequisites, drove our decision to establish a high-interaction client honeypot for gathering malware samples.

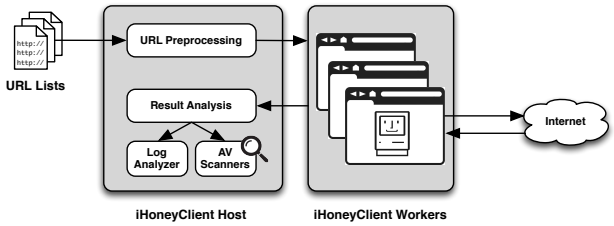

Fig. 1. System overview of the iHoneyClient host (*Queuer* and *Analyzer*) and the iHoneyClient workers (*Visitor*).

III. System

As illustrated in Figure 1, iHoneyClient consists of several workers running in virtual machines, orchestrated by the iHoneyClient host. The host compiles a list of URLs to be visited or files to be analyzed, depending on the mode of operation. The workers either visit those URLs to collect malware samples dropped drive-by download attacks or provide the dynamic analysis environment to analyze samples.

To catch new malware, iHoneyClient retrieves URL lists from different services providing malware-related URL blacklists (currently Malware Domain List [18], Malware Patrol [19] and Clean-MX [20]). Before visiting a URL, the host checks if the server is reachable, the HTTP status code and content type returned after a HTTP HEAD request. The first two checks increase the efficiency of iHoneyClient, while the third enforces the focus on drive-by downloads. Only requests returning content of the type "text/html" are analyzed to filter out the numerous entries pointing directly to Windows executables. When malware samples are analyzed directly, their file type is examined before execution. iHoneyClient is able to automatically process Mach-0 binaries, .dmg disk images, .zip archives containing OS X Applications, Java archives and installers in .pkg files. Whenever a URL or malware sample passes these checks, a new VirtualBox instance is created to provide a clean environment for further analysis.

iHoneyClient workers are virtualized instances of Mac OS X 10.6.8 "Snow Leopard". A few modifications to the vanilla Mac OS X installation are necessary to be able to run it in a virtual machine. This stems mostly from the usage of EFI (Extensible Firmware Interface), that is not yet fully supported in VirtualBox. The same modifications are used by the "Hackintosh"community, that strives to run Apple's operating system on PC hardware. This includes a custom bootloader called Chameleon, that contains an EFI emulator capable of providing the basic system information required by the XNU kernel, two kernel modules to provide shutdown and power

management functionality and a kernel module simulating the System Management Controller (SMC) chip used to identify the virtual machine as genuine Apple hardware.

The process of monitoring activities inside the virtual machine is centered around DTrace [21]. The basic building block of this dynamic tracing framework are its instrumentation points, called probes. Whenever the code instrumented by a probe is executed, the probe fires, which results in the execution of a D program specified by the user. Probes can either be statically inserted at key positions by a programmer or dynamically at runtime. DTrace is able to patch running programs as well as the kernel and insert interrupts at arbitrary points, notifying the DTrace user of events like system calls, function calls or CPU performance data. We use this mechanism to react to key system calls like opening files or connecting sockets.

To make full use of DTrace, iHoneyClient requires another kernel module. Apple introduced a new ptrace API allowing a program to prevent DTrace from monitoring it, similar to denying a debugger to attach. The activities of malware using this API would not be detectable with iHoneyClient. Thus, we prohibit such ptrace calls with a kernel module [22].

The software setup of iHoneyClient intentionally provides an easy target for attackers. Aside from the upgrade to OS X 10.6.8, no patches or updates were installed. The software configuration includes Firefox 3.6.8 (released in July 2010) and Java 1.6.0_17 (released in December 2009).

The iHoneyClient virtual machine instances are connected to the host with a VirtualBox NAT network. The hosts are isolated from the Internet with a firewall. We took care to block network traffic known to be malicious while imposing as few restrictions to the malware samples as possible. Thus, we blocked all ports associated with e-mail traffic, ports of services known to be remotely exploitable and non-standard ports used by known mass malware. Additionally, we log all traffic to and from the virtual machine instances.

While the iHoneyClient worker performs its task, it is monitored with DTrace. For analyzing URLs, the focus lies on opened files and newly created processes. After a timeout, the logs and all opened files are copied to the iHoneyClient host. There, they are scanned with antivirus products from Kaspersky and Avira to detect new infections. Furthermore, the process logs are checked for new, non-standard processes. In the dynamic analysis mode, we log all system calls. After copying the system call logs to the host, several scripts analyze the results and create an analysis report.

Figure 2 illustrates one run of the iHoneyClient honeypot. For analyzing stand-alone malware samples, the program flow is analogous. The only difference is, that the queuer compiles a list of files to be analyzed while the visitor performs checks on the file type instead of an HTTP request. Then, the file is executed inside the virtual machine instance.

IV. Evaluation

We evaluated our system threefold: First, we performed case studies on two well-known OS X malware samples to verify the completeness of our reports. We then used iHoneypot to crawl blacklisted domains to see whether we can find OS X malware dropped by drive-by download exploits in the wild. Finally, we give insights on the behavior of over a hundred malware samples collected during execution in our sandbox.

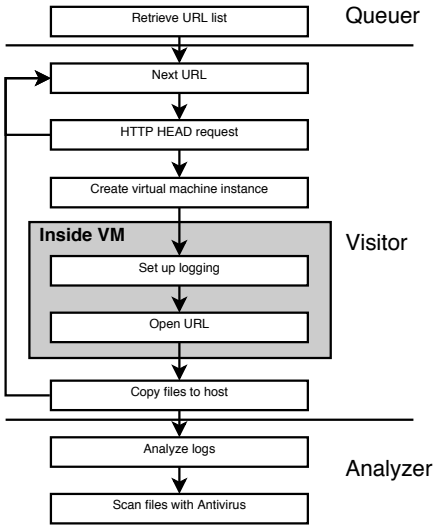

Fig. 2. Flow chart of an iHoneyClient honeypot run.

A. Case Studies

In order to verify, whether the honeypot is able to correctly recognize new malware infections, we tested it in an experimental setup. First, we used the Metasploit framework [23] to create a simple web site exploiting *CVE-2011-3544* [24], a vulnerability in Javas Rhino Scripting Engine. We used the `java/meterpreter/reverse_tcp` payload, which provides a shell-like environment for uploading files or spawning processes. After the iHoneyClient worker visited the prepared website and spawned a new meterpreter session, we uploaded a sample of Flashback and executed it in the virtual machine. The antivirus scanner correctly detected both the malicious Java applet and the Flashback binary.

To verify the correct functionality of the iHoneyClient analyzer we manually compared the analysis reports for two well-known malware samples to the threat descriptions from antivirus companies and show by means of log snippets how the malware's behavior is reflected by the analysis report.

OSX/Olyx.B. Olyx is a backdoor used in targeted attacks on Tibetan NGOs. We used threat descriptions by ESET [25] and Microsoft [26] as a reference.

According to the threat descriptions, Olyx implements a mechanism to start itself automatically when the infected system reboots. To this end, it creates the LaunchAgent script `~/Library/LaunchAgents/com.apple.DockActions.plist` and attempts to copy itself to `/Library/Audio/Plug-Ins/AudioServer`. However, as a default user does not have write permissions to this directory, the call to `open` results in error #13 (permission denied). As described in the ESET threat report, Olyx thus fails to persist itself and does not survive a reboot:

```
open("/Users/xxx/Library/LaunchAgents/com.apple.DockActions
     .plist\0", 0x601, 0x1FF) = 6 0
```

```
open("/Library/Audio/Plug-Ins/AudioServer\0", 0x601, 0x1FF)
     = -1 Err#13
```

The malware then attempts to connect to its C&C server. The contacted domain depends on the version of Olyx, in this case `mail.hiserviceusa.com` is used. The operation fails with error #61 (connection refused) since the server is no longer online:

```
socket(0x2, 0x1, 0x0) = 0 0
      connect to 65.19.141.197:4670 => -1 61
connect(0x0, 0x7FFF5FBFF9D0, 0x10) = -1 Err#61
```

The malware sample then enters an endless loop retrying to connect and does not expose any further activities.

OSX/Flashback. As described in Section II-A, Flashback is the most successful OS X malware to date. We used the threat analyses by Kaspersky [8] and Intego [27] as a reference for its behavior.

One notable characteristic of Flashback is that it checks its host for several applications such as XCode, antivirus scanners or network monitoring tools:

```
stat("/Library/Little Snitch\0", 0xBFFFE62C, 0xBFFFE57C) =
     -1 Err#2
stat("/Developer/Applications/Xcode.app/Contents/MacOS/
     Xcode\0", 0xBFFFE62C, 0xBFFFE57C) = -1 Err#2
[ ... ]
stat("/Applications/HTTPScoop.app\0", 0xBFFFE62C, 0
     xBFFFE57C) = -1 Err#2
stat("/Applications/Packet Peeper.app\0", 0xBFFFE62C, 0
     xBFFFE57C) = -1 Err#2
```

If any of these files are present Flashback deletes itself and terminates. If all checks return a negative result, Flashback continues to connect to its C&C server. The IP address of this server varies in most samples from our test set, but no sample uses DNS to resolve a name. The server refuses the connection by sending a TCP RST packet. This causes Flashback to overwrite itself with zeroes, delete itself and terminate execution:

```
open_nocancel("/samples/flashback\0", 0x601, 0x1B6) = 8 0
fstat64(0x8, 0x7FFF5FBFE240, 0x7FFF5FBFE30C) = 0 0
write_nocancel(0x8,"\0",0x1000) = 4096 0
[ ... ]
write_nocancel(0x8,"\0", 0x800) = 2048 0
close_nocancel(0x8) = 0 0
unlink("/samples/flashback\0", 0x0, 0x0) = 0 0
```

With the appearance of newer variants, Flashback has changed and extended its functionality. For example, OSX/Flashback.S deletes all files and folders in `~/Library/Caches/Java/cache` to delete the applet responsible for infecting the host in the first place. This complicates the process of sample recovery:

```
posix_spawn(0xBFFFE5FC, 0x94D8600C, 0xBFFFE538) = 0 0
 exec-success( sh -c rm -rf /Users/xxx/Library/Caches/Java/
     cache ) = 1 0
```

B. Honeypot Results

To evaluate out how widespread OS X malware is in the wild, we used three different domain blacklists (Malware Patrol, Malware Domain List and Clean MX) to compose an input feed of drive-by download URLs for iHoneyClient. In several runs during January 2013, we obtained a total of 6,028 malicious URLs. After applying the URL filters described in Section III, 2,844 URLs remained to be visited.

TABLE I
ALERTS BY KASPERSKY AFTER CRAWLING KNOWN MALWARE DOMAINS.

Type	Amount
JavaScript	386
HTML	23
Windows binary	12
VBS	3
Flash	2

TABLE II
PERCENTAGE OF SAMPLES EXHIBITING NETWORK ACTIVITY.

Activity	Percentage
Any network activity	30 %
DNS queries	11 %
Valid DNS answers	7 %
Dynamic DNS services	4 %
Successful connections to a server	21 %
Unsuccessful connection attempts	9 %
Connections to two or more servers	5 %
Binding a socket	1 %

A total of 288 of those sites caused one or more antivirus alerts. Table I details the type of alert, showing a predominance of malicious JavaScripts. Only five domains successfully used drive-by downloads, dropping twelve malicious Windows binaries. To our surprise, we were not infected by any kind of OS X or cross-platform malware.

C. Dynamic Analysis

To get an insight into the behavior of current Mac malware, we obtained 148 recent malware samples for OS X from VirusTotal [28] in January 2013 and examined them using our dynamic analysis environment. To this end, we selected all samples in VirusTotal's database that matched a signature for OS X by at least one antivirus scanner. After filtering out invalid OS X file types (e.g. Windows executables matching signatures for OS X malware), the sample set was comprised of 111 binary Mach-O executables, 27 OS X applications (.app folders), six Java archives and four .dmg disk images. We focused our analysis on file modification events, network activity and process-related activity.

Table III shows statistics about the files created by the malware samples. More than half the samples did not attempt to open a file with write access, 12% of the samples created hidden files, while only 6% attempted to create a LaunchAgent entry, responsible for starting automatically after a user login or system reboot.

Table II summarizes the network activity of the analyzed malware samples. Out of the total of 148 samples, 43 showed some network activity. Only 17 of those required a DNS query to initiate a connection, six used a dynamic name resolution service like DynDNS [29]. In total, the analyzed samples attempted to open 44 TCP connections, 31 of which were successfully established. Three malware samples used erroneous IP addresses due to insufficient validation of the DNS reply. Although eight samples connected to two or more servers, all samples failing to connect to their server did not try another IP address. Two samples bound a socket, but did not actively initiate any connections.

Table IV shows the most commonly used ports. The majority of connections used the ports for HTTP(S), other ports are only used in one or two samples each. Analysis of the

TABLE III
PERCENTAGE OF SAMPLES WRITING FILES.

File write activity	Percentage
Any file	43 %
Outside user home	26 %
LaunchAgent entries	6 %
Hidden files	12 %

TABLE IV
USED TCP PORTS.

Port	Percentage
80	16 %
443	1 %
3360	2 %
4141	1 %
4670	1 %
8080	2 %
25565	2 %
49474	1 %

TABLE V
PERCENTAGE OF SAMPLES SPAWNING PROCESSES.

Activity	Percentage
Samples calling `fork`	9 %
Samples calling `execve`	5 %
Samples with erroneous `execve` calls	3 %

traffic showed that only nine samples actually transmitted data via HTTP. Two samples used an SSL connection to transfer encrypted payload.

Another important aspect of malware samples is whether or not they spawn new processes. Table V gives an overview of system calls to `fork` and `execve`. 13 samples made a system call to `fork`, eight called `execve`. Interestingly, not all `execve` calls are preceded by a call to `fork`. Furthermore, only 50% of the calls to `execve` were successful.

V. CONCLUSION

In this paper, we introduced iHoneyClient, a high interaction, VirtualBox-based OS X honeypot. Using this honeypot, we examined a set of 6,028 URLs retrieved from malware blacklists to gain insight on the current threat level for OS X users. We found that only five websites dropped binaries through drive-by downloads, all of them targeted at Microsoft Windows.

Additionally, we presented our dynamic analysis environment and verified it by manually comparing our analysis logs to threat reports issued by antivirus companies for two malware samples. Furthermore, we compiled an overview of the behavior of current OS X malware. Based on 148 malware samples, we created statistics on file creation, network activity and process management.

Our findings paint an ambivalent picture of OS X malware. While some malware families use sophisticated techniques and pose a significant threat to OS X users, several others fail to perform simple but important tasks like persisting after a reboot. Moreover, while computers running Microsoft Windows face a consistently high risk to become infected with malicious software, our results suggest that infecting a Mac by simply browsing the Web is highly unlikely.

Based on the work presented in this paper, we plan to perform a larger-scale analysis by extending the malware blacklists used as input for iHoneyClient. We also plan to crawl the URLs provided by those blacklists regularly. This will hopefully allow us to monitor a successful OS X malware outbreak like Flashback. Furthermore, we plan to include the analysis of OS X executables in a publicly available dynamic malware analysis tool Anubis [30].

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Union Seventh Framework Programme under grant agreement n. 257007 (SysSec) and from the FFG – Austrian Research Promotion under grant COMET K1.

REFERENCES

[1] A. J. O'Donnell, "When Malware Attacks (Anything but Windows)," *IEEE Security and Privacy*, vol. 6, no. 3, May 2008.
[2] S. J. Vaughan-Nichols, "Windows has fallen behind Apple iOS and Google Android," http://www.zdnet.com/windows-has-fallen-behind-apple-ios-and-google-android-7000008699, 2012.
[3] "OS Statistics," http://www.w3schools.com/browsers/browsers_os.asp.
[4] T. Bradley, "Is MacDefender Malware a Sign of the Macpocalypse?" http://www.pcworld.com/article/228832/is_macdefender_malware_a_sign_of_the_macpocalypse.html, 2011.
[5] S. Evans, "Apple '10 years' behind Microsoft on security: Kaspersky," http://malware.cbronline.com/news/apple-10-years-behind-microsoft-on-security-kaspersky-250412, 2012.
[6] Z. Whittaker, "Latest Mac malware adds to 'troubling trend,' says security expert," http://www.zdnet.com/latest-mac-malware-adds-to-troubling-trend-says-security-expert-7000008814, 2012.
[7] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2012-0507.
[8] A. Gostev, "The anatomy of Flashfake. Part 1," http://www.securelist.com/en/analysis/204792227/The_anatomy_of_Flashfake_Part_1, 2012.
[9] M.-E. M. Léveillé, "OSX/Flashback," http://go.eset.com/us/resources/white-papers/osx_flashback.pdf, 2012.
[10] A. Moser, C. Kruegel, and E. Kirda, "Limits of Static Analysis for Malware Detection," in *Annual Computer Security Applications Conference (ACSAC)*, 2007.
[11] B. Stone-Gross, A. Moser, C. Kruegel, K. Almaroth, and E. Kirda, "FIRE: FInding Rogue nEtworks," in *Annual Computer Security Applications Conference (ACSAC)*, 2009.
[12] M. G. Kang, P. Poosankam, and H. Yin, "Renovo: A Hidden Code Extractor for Packed Executables," in *ACM Workshop on Recurring Malcode (WORM)*, 2007.
[13] L. Martignoni, M. Christodorescu, and S. Jha, "OmniUnpack: Fast, Generic, and Safe Unpacking of Malware," in *Annual Computer Security Applications Conference (ACSAC)*, 2007.
[14] R. Paleari, L. Martignoni, E. Passerini, D. Davidson, M. Fredrikson, J. Giffin, and S. Jha, "Automatic Generation of Remediation Procedures for Malware Infections," in *USENIX Security Symposium*, 2010.
[15] M. Lindorfer, C. Kolbitsch, and P. Milani Comparetti, "Detecting Environment-Sensitive Malware," in *International Symposium on Recent Advances in Intrusion Detection (RAID)*, 2011.
[16] M. Egele, T. Scholte, E. Kirda, and C. Kruegel, "A Survey on Automated Dynamic Malware Analysis Techniques and Tools," *ACM Computing Surveys Journal*, vol. 44, no. 2, 2012.
[17] M. Qassrawi and H. Zhang, "Client Honeypots: Approaches and Challenges," in *International Conference on New Trends in Information Science and Service Science (NISS)*, 2010.
[18] "Malware Domain List," http://www.malwaredomainlist.com.
[19] "Malware Patrol," http://www.malware.com.br.
[20] "CLEAN MX," http://support.clean-mx.de/clean-mx/viruses.php.
[21] B. Gregg and J. Mauro, *DTrace: Dynamic Tracing in Oracle Solaris, Mac OS X and FreeBSD*. Prentice Hall, 2011.
[22] L. Fuller, "Fixing ptrace(pt_deny_attach, ...) on Mac OS X 10.5 Leopard," http://landonf.bikemonkey.org/code/macosx/Leopard_PT_DENY_ATTACH.20080122.html, 2008.
[23] "Metasploit," http://www.metasploit.com.
[24] http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-3544.
[25] A. Dorais-Joncas, "OSX/Lamadai.A: The Mac Payload," http://blog.eset.com/2012/03/28/osxlamadai-a-the-mac-payload, 2012.
[26] Microsoft Malware Protection Center, "Backdoor:MacOS_X/Olyx.B," http://www.microsoft.com/security/portal/threat/encyclopedia/entry.aspx?Name=Backdoor:MacOS_X/Olyx.B, 2012.
[27] P. James, "New Flashback Variant Continues Java Attack," http://www.intego.com/mac-security-blog/new-flashback-variant-continues-java-attack-installs-without-password/, 2012.
[28] "VirusTotal," http://www.virustotal.com.
[29] "DynDNS," http://www.dyndns.com.
[30] "Anubis," http://anubis.iseclab.org.