POSTER: Cross-Platform Malware: Write Once, Infect Everywhere

Martina Lindorfer*, Matthias Neumayr*, Juan Caballero†, Christian Platzer*
*Vienna University of Technology †IMDEA Software Institute
{mlindorfer, mneumayr, cplatzer}@iseclab.org juan.caballero@imdea.org

ABSTRACT

In this ongoing work we perform the first systematic investigation of cross-platform (X-platform) malware. As a first step, this paper presents an exploration into existing X-platform malware families and X-platform vulnerabilities used to distribute them. Our exploration shows that X-platform malware uses a wealth of methods to achieve portability. It also shows that exploits for X-platform vulnerabilities are X-platform indeed and readily available in commercial exploit kits, making them an inexpensive distribution vector for X-platform malware.

Categories and Subject Descriptors

K.6.5 [**Security and Protection**]: Invasive software

Keywords

Cross-Platform Software; Malware; Vulnerabilities.

1. INTRODUCTION

A desired capability by many programmers is writing a program once and then using it on different computing platforms without modifications. This capability has been sold by programming languages using catchphrases such as "write once, run anywhere" or "write once, compile anywhere" and brings benefits such as reduced development time, code reuse, and easier maintenance.

Although these paradigms are fairly extended in benign software, they are not prevalent in malware. Nowadays, the very large majority of malware runs on a single platform: most malware targets Windows, with Android malware recently growing, and few instances of Mac OS and Linux malware. While cross-platform (X-platform) malware has been around for a long time, e.g., the Morris worm in 1988 and macro viruses in the 90's [11], it still constitutes a very small minority.

However, the advent of mobile computing and the increase in X-platform malware in the last two years raises the question of whether the tide is rising. In this context, security vendors like Websense [24] and Fortinet [14] have forecasted X-platform malware as a 2013 trend. While it is unclear how fast (if) the tide will rise, in this work we proactively perform the first systematic investigation of X-platform malware, to be prepared when (if) it happens.

For a malware developer, supporting a new platform is a decision that boils down to a cost-benefit analysis. Since the goal of most malware families (targeted attacks apart) is monetizing the infected computers, the malware developer needs to weigh the additional income obtained by reaching targets of the new platform, against the required additional investment in software development and distribution.

In this ongoing work we strive to understand such a cost-benefit tradeoff. As a first step, we perform an exploration of existing X-platform malware. This helps us understand what monetization vectors work across platforms and how much development effort is needed to make the malware X-platform. For example, we observe malware families focusing on click fraud (e.g., LilyJade [18]) and information-stealing (e.g., ZeuS-in-the-Mobile [26]), two monetization vectors that extrapolate well to mobile platforms. We also observe families that reuse the same Java code on all platforms (e.g., jRAT [16]) and others that prefer platform-specific components (e.g., Badbunny [3]).

Malware developers also require cost-effective ways of distributing their X-platform malware. Our examination shows that they favor distribution vectors that support all platforms such as social engineering and vulnerabilities on programs that run on multiple platforms, i.e., X-platform vulnerabilities. Given the predominance of drive-by downloads as a malware distribution vector [6], we analyze the presence of exploits for X-platform vulnerabilities in commercial exploit kits. Our exploration shows that most X-platform vulnerabilities have been weaponized in at least one exploit kit. It also shows that most exploits for X-platform vulnerabilities indeed work on different platforms, without any modifications. Thus, malware owners using drive-by download specialization services, can essentially distribute their malware to multiple platforms at no extra cost.

2. OVERVIEW

This section defines X-platform programs and presents an examination of X-platform malware and vulnerabilities.

A *computing platform* can refer to an operating system family (e.g., Windows, Linux, OS X, Android), a computer architecture (e.g., x86, AMD64, ARM), or a combination of both (e.g., Windows on x86). In this work, a platform corresponds to an OS family and we consider two OS families different if they do not share significant amounts of code. In particular, we consider different versions of the same OS

CCS'13, November 4–8, 2013, Berlin, Germany.
ACM 978-1-4503-2477-9/13/11.
http://dx.doi.org/10.1145/2508859.2512517.

Family	Date	DV	Windows	Linux	OS X	Windows Phone	Symbian	Android	BlackBerry
Badbunny [3]	07/2009	SE	JavaScript	Perl	Ruby				
Boonana [4, 5]	10/2010	SE	Java		Java				
ZitMo [25, 26]	09/2010	SE				Java*	Java	Java*	Java
Olyx [21]	06/2011	Exploit	PE		MACH-O				
Tibet [23]	03/2012	Exploit	PE		MACH-O				
Flsplysc [13]	04/2012	Exploit	PE		Python				
Crisis [17]	04/2012	SE	PE		MACH-O	PE*			
LilyJade [18]	05/2012	Exploit	JavaScript	JavaScript	JavaScript				
GetShell [15]	07/2012	SE	Java	Java	Java				
Netweirdrc [20]	08/2012	SE	PE	ELF	MACH-O				
jRAT [16]	10/2012	SE	Java	Java	Java				
Ssucl [22]	01/2013	SE	PE					Java	
MinecraftHack [19]	03/2013	SE	Java		Java				
Janicab [2]	07/2013	Exploit/SE	VB Script		Python				
Clt10 [7] (PoC)	2006	-	ASM	ASM					
Yakizake [11] (PoC)	08/2007	-	.NET	.NET					
Clapzok [12] (PoC)	05/2013	-	ASM	ASM	ASM				

Table 1: X-platform malware, earliest date reported, and distribution form used for each supported platform. Stars mark platforms supported later than the earliest reported date.

(e.g., Windows XP, Vista, 7) and different distributions of the same OS (e.g., Ubuntu, Fedora) the same platform.

We say that a program is X-platform if it is portable across (i.e., runs on) different OS families. A program can become portable by using programming languages that compile to bytecode (e.g., Java, .NET), at the source code level using standardized interfaces like POSIX or interpreted languages, and by running on top of other X-platform programs such as web browsers or office applications.

2.1 X-Platform Malware

As a first step we perform an investigation of existing X-platform malware. Table 1 shows malware families we found, the earliest date they were reported, their distribution vector (DV), and the target platforms they support. The top of the table comprises 14 families observed in the wild, while the bottom three are proof-of-concept malware.

Out of the 14 families in the wild, four use exploits to get installed on the target hosts, while the remaining nine convince users to install them through social engineering (SE) and one uses both depending on the platform. The four families that solely rely on exploitation leverage X-platform vulnerabilities that enable installation on the different target platforms (see Section 2.2).

For each supported platform, Table 1 shows in which form the malware is distributed, which is tightly linked to how the malware achieves portability. It shows that X-platform malware can be distributed as source code (Python, JavaScript, Perl), binary code (PE, ELF, MACH-O), or bytecode (Java, .NET). When the malware is not distributed as binary code, it can fail to run if the infected host does not have an interpreter for the scripts or a runtime for the bytecode. When distributed as binary code, it needs to match the executable file format used by the platform. Note that distribution as source code or bytecode makes it significantly easier for analysts to reverse-engineer the malware. So far, the use of external tools to obfuscate the source code and bytecode is not prevalent among X-platform malware.

An important question is to what degree X-platform malware reuses code across platforms. Our preliminary examination indicates a mixture of approaches. Three families use the same code across platforms: Minecraft and jRAT (both Java) and LilyJade, a JavaScript browser plugin that uses the Crossrider API [9] to run on Internet Explorer, Chrome, and Safari without modifications. In addition, Boonana and Getshell distribute as a Java JAR file, but later download platform-specific modules. Another four families use different malware for each platform. In particular, Badbunny distributes as an OpenOffice document containing a macro and executables for each platform, Sscul distributes as an Android APK containing a PE executable, Crisis as a Java JAR file containing PE and MACH-O executables, and Flsplysc uses Java code during exploitation to select which platform-specific module to install. Finally, there are four families for which further analysis is needed. Olyx, Tibet, and Netweirdrc distribute as executables, and although Zitmo is written in Java for all platforms, its Android version is significantly simpler [25], pointing to the use of separate code bases. To refine this preliminary examination we plan to use code analysis and similarity techniques.

2.2 X-Platform Vulnerabilities

A X-platform vulnerability is a software defect on a X-platform program. The vulnerable program may comprise both platform-specific and platform-independent code, or be fully platform-independent by running on top of an above-OS runtime that enables portability. In both cases, a X-platform vulnerability is present in the platform-independent code of the vulnerable application.

Table 2 summarizes our investigation on X-platform vulnerabilities. The left side of the table shows, for each vulnerability, its CVE identifier [10], the vulnerable program, whether there exists a publicly available exploit for the vulnerability, and whether any exploit kit contains an exploit for it [8]. As shown, X-platform vulnerabilities exist in browser plugins (Java, PDF, Flash), web browsers (Firefox, WebKit), and prevalent desktop applications like Microsoft Word. The majority of these applications are written in C/C++, although by far the most vulnerable application is the Java runtime, for which we plan to evaluate which parts of the code contain the vulnerabilities.

Nearly all of these vulnerabilities have publicly available exploits, the exception being recent zero-day vulnerabilities (marked with Z) for which we expect a public exploit soon. It is worrisome to observe that for most of these vulnerabilities an exploit is included in commercial exploit kits. If these exploits are truly X-platform (examined in Section 3), then for an attacker using drive-by download specialization services [6], distributing its malware to multiple platforms essentially incurs no extra cost.

3. PRELIMINARY RESULTS

Our exploration of X-platform vulnerabilities shows that exploits are widely available for them. But, it is unclear whether those exploits indeed work across platforms without modifications, given particular exploit payloads and OS-

CVE	Program	Exploit	Kit	Metasploit			
				XP	7	Linux	X
2009-0563	MS Word	✓					
2009-3867	Oracle Java	✓	✓	✓			
2010-3333	MS Word	✓					
2011-1774	WebKit	✓		✓			
2011-3544	Oracle Java	✓	✓	✓	✓	✓	✓
2012-0507	Oracle Java	✓	✓	✓	✓	✓	✓
2012-0779	Adobe Flash	✓	✓	✓			
2012-1723	Oracle Java	✓	✓	✓	✓	✓	
2012-4681	Oracle Java	✓	✓	✓	✓	✓	✓
2012-5076	Oracle Java	✓	✓	✓	✓	✓	✓
2013-0422	Oracle Java	✓	✓	✓	✓	✓	✓
2013-0431	Oracle Java	✓	✓	✓	✓	✓	✓
2013-0640	Adobe Reader	Z		-	-	-	-
2013-0641	Adobe Reader	Z		-	-	-	-
2013-0758	Firefox	✓		✓	✓	✓	✓
2013-1488	Oracle Java	✓		✓	✓	✓	✓
2013-1491	Oracle Java	Z		-	-	-	-
2013-2423	Oracle Java	✓	✓	✓	✓	✓	✓

Table 2: X-platform vulnerabilities, whether an exploit is publicly available and included in an exploit kit, and whether the exploit works in each platform.

specific defenses such as ASLR and $W \oplus X$. To investigate this, we test exploits for X-platform vulnerabilities, from the Metasploit framework [1], against different platforms. Our setup includes Windows XP, Windows 7, Debian Linux, Mac OS X 10.6 "Snow Leopard", and Mac OS X 10.8 "Mountain Lion". To allow running vulnerable Java versions on recent browsers we disable extensions.blocklist in Firefox and XProtect in Safari. For Java exploits we use the `java/meterpreter/reverse_tcp` payload and the `generic/shell_reverse_tcp` payload for others, verifying exploitation by executing uploaded files.

The results on the right side of Table 2 show that the majority (9 out of 15) of the public exploits for X-platform vulnerabilities are truly X-platform, exploiting different platforms with no modifications. Four other exploits fail to run in at least one vulnerable platform, and the two Microsoft Word exploits crash in all platforms. Note that the failing exploits are for the oldest vulnerabilities; exploit writers are becoming more effective over time.

4. FUTURE WORK

In this paper we have introduced our ongoing work to understand the X-platform malware and vulnerabilities landscape. Much remains to be done, including collecting samples of the identified X-platform malware families, measuring the amount of code reuse across platform-specific executables, examining exploits that do not work on all platforms, and analyzing X-platform exploits and malware in the wild. For the latter, we are deploying multi-platform honeyclients to monitor what platforms are exploited by drive-by downloads and what malware is dropped on each platform.

5. ACKNOWLEDGMENTS

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

6. REFERENCES

[1] Metasploit. `http://www.metasploit.com`.

[2] Multisystem Trojan Janicab attacks Windows and MacOSX via scripts. `http://blog.avast.com/2013/07/22/multisystem-trojan-janicab-attacks-windows-and-macosx-via-scripts/`, July 2013.

[3] SB/BadBunny-A, July 2009. `http://www.sophos.com/en-us/threat-center/threat-analyses/viruses-and-spyware/SB~BadBunny-A/detailed-analysis.aspx`.

[4] Boonana Trojan Horse, October 2010. `http://www.securemac.com/boonana-bulletin.php`.

[5] Boonana trojan linked to malware for Windows, Mac OS X or Linux, October 2010. `http://nakedsecurity.sophos.com/2010/10/28/cross-platform-worm-targets-facebook-users/`.

[6] C. Grier et al. Manufacturing Compromise: The Emergence of Exploit-as-a-Service. In *Proceedings of the 19th ACM Conference on Computer and Communication Security (CCS)*, 2012.

[7] CAPZLOQ TEKNIQ v1.0, 2006. `http://spth.virii.lu/rrlf7/sources/clt.htm`.

[8] An Overview of Exploit Packs (Update 19.1), April 2013. `http://contagiodump.blogspot.com/2010/06/overview-of-exploit-packs-update.html`.

[9] Crossrider. `http://crossrider.com`.

[10] CVE Database. `http://cve.mitre.org/cve/`.

[11] P. Ferrie. Something Smells Fishy. *Virus Bulletin*, November 2007. `http://pferrie.host22.com/papers/yakizake.pdf`.

[12] P. Ferrie. MultiPlatform Madness. *Virus Bulletin*, June 2013. `http://pferrie.host22.com/papers/clapzok.pdf`.

[13] Python-based malware attack targets Macs. Windows PCs also under fire, April 2012. `http://nakedsecurity.sophos.com/2012/04/27/python-malware-mac/`.

[14] Fortinet's FortiGuard Labs Reveals 2013 Threat Predictions, December 2012. `http://www.fortinet.com/press_releases/121210.html`.

[15] Multi-platform Backdoor Lurks in Colombian Transport Site, July 2012. `http://www.f-secure.com/weblog/archives/00002397.html`.

[16] New Multiplatform Backdoor Jacksbot Discovered, October 2012. `http://www.intego.com/mac-security-blog/new-multiplatform-backdoor-jacksbot-discovered/`.

[17] T. Katsuki. Crisis: The Advanced Malware, November 2012. `http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/crisis_the_advanced_malware.pdf`.

[18] Worm 2.0, or LilyJade in action, May 2012. `http://www.securelist.com/en/blog/706/Worm_2_0_or_LilyJade_in_action/`.

[19] Intego Discovers a New Multi-Platform Minecraft Password Stealer, March 2013. `http://www.intego.com/mac-security-blog/intego-discovers-a-new-multi-platform-minecraft-password-stealer/`.

[20] An Analysis of the Cross-Platform Backdoor NetWeirdRC, August 2012. `http://www.intego.com/mac-security-blog/an-analysis-of-the-cross-platform-backdoor-netweirdrc/`.

[21] Backdoor Olyx - is it malware on a mission for Mac?, July 2011. `http://blogs.technet.com/b/mmpc/archive/2011/07/25/backdoor-olyx-is-it-malware-on-a-mission-for-mac.aspx`.

[22] Mobile attacks!, January 2013. `http://www.securelist.com/en/blog/805/Mobile_attacks/`.

[23] Tibet.C Malware Delivered by Poisoned Word Documents Installs Backdoors on Macs, March 2012. `http://www.intego.com/mac-security-blog/tibet-c-malware-delivered-by-poisoned-word-documents-installs-backdoors-on-macs/`.

[24] 7 for 13: 2013 Security Predictions from Websense, December 2012. `http://www.websense.com/content/websense-2013-security-predictions.html`.

[25] ZeuS-in-the-Mobile: Facts and Theories, October 2011. `http://www.securelist.com/en/analysis/204792194/`.

[26] ZeuS Mitmo: Man-in-the-mobile, September 2010. `http://securityblog.s21sec.com/2010/09/zeus-mitmo-man-in-mobile-i.html`.