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.
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 these vulnerabilities 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.

3. PRELIMINARY RESULTS

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