POSTER: Study of Software Plugin-based Malware

Liang Yu[1,2]
[1]Beijing University of Posts and Telecommunications
[2]Institute of Computing Technology, Chinese Academy of Sciences
liangyu@bupt.edu.cn

Li Zhiqiao[1,2]
[1]Beijing University of Posts and Telecommunications
[2]Institute of Computing Technology, Chinese Academy of Sciences
lizhiqiao@software.ict.ac.cn

Cui Xiang[1]
[1]Institute of Computing Technology, Chinese Academy of Sciences
cx@ict.ac.cn

ABSTRACT
Security issues of software plugins are seldom studied in existing researches. The plugin mechanism provides a convenient way to extend an application's functionality. However, it may also introduce susceptibility to new security issues. For example, attackers can create a malicious plugin to accomplish intended goals stealthily. In this poster, we propose a *Software Plugin-based Malware* (SPM) model and implement SPM prototypes for Microsoft Office, Adobe Reader and mainstream browsers, with the aim to study the development feasibility of such malware and illustrate their potential threats.

Categories and Subject Descriptors
D.4.6 [OPERATING SYSTEMS (C)]: Security and Protection

General Terms
Security, Malware.

Keywords
Software plugin, SPM, malware.

1. INTRODUCTION
A plugin, also known as extension or add-on, is a software component that adds a specific feature to existing software (host application). Most of today's popular software solutions support plugins, such as IE's Adobe Flash Player, MS Office and Adobe Reader's translation plugin. Each of these are conceptualized and designed to perform specific functions and simplify our work.

Although, a plugin can bring immense convenience to users, it also introduces some threats [1] [2]. Some viruses such as "Gauss" can use plugin mechanism to compromise the privacy of IE users; "Red October" can download other malware into a computer. A plugin runs within its host application, and therefore, when initiated, it may have the same privileges as its host application.

A SPM is more competent than conventional malware in that it does not need a software bug to exploit and can be initiated as soon as its host application begins. Further, the host application inadvertently serves as a protective umbrella for the SPM, which thereby avoids detection and termination by antivirus software. Moreover, the SPM can also bypass the computer's firewall depending on the host application privileges and can access files opened by the host application to extract information.

In this poster, we propose a SPM model and implement SPM prototypes for MS Office (Word, Excel, Outlook and PowerPoint), Adobe Reader and mainstream browsers (IE, Firefox and Google Chrome). These SPM prototypes can communicate with a remote server, download executable files, and execute them upon initiation. In addition, they can also execute CMD commands and create working threads, the browser extension in SPM can even be used to collect sensitive information about the user. All these actions can successfully bypass Avira and Kaspersky antivirus software.

CCS'14, November 3-7, 2014, Scottsdale, AZ, USA.
ACM 978-1-4503-2957-6/14/11.
http://dx.doi.org/10.1145/2660267.2662381

2. THE PROPOSED SOFTWARE PLUGIN-BASED MALWARE MODEL

2.1 Definitions and Modules
Figure 1 illustrates the proposed SPM model, which can be divided into three main parts: a Dropper, a plugin, and a command and control (C&C) server.

The **Dropper** is a wrapper program for a plugin, which can install the plugin automatically and stealthily without the user's consent in such a manner that the process does not trigger antivirus software, system firewall and User Account Control (UAC). A Dropper is always necessary for an attacker because a plugin's installation procedures may be complex or they may need administrator rights for installation. Therefore, a Dropper can simplify installation procedures greatly. Once the dropper has performed its task, it deletes itself, and the plugin will be installed successfully.

The **Plugin** is the key part of the SPM model. Different software imposes different types and levels of limitations on plugins' privileges. This determines the levels of security that the attacker needs to bypass or break for completing its malicious actions. In addition, the methods of plugins' development vary as well. One kind of software may support a few methods to develop plugins, and depending on the development environment used, the plugin can have different privileges.

In our experiments, in terms of MS Office and Adobe Reader, first, an attacker can create a plugin in the form of a DLL for MS Office and an API file(file extension is API) for Adobe Reader, using C/C++. Subsequently, the attacker can add malicious code into MS Office's plugin's OnConnection(…) function and Adobe Reader's plugin's PluginInit(void) function. While the former function will be called when the MS Office software loads a plugin, the latter function is responsible for the Adobe Reader plugin's initial work. After these two steps, both plugins will be loaded, and the attacker's malicious code will be executed every

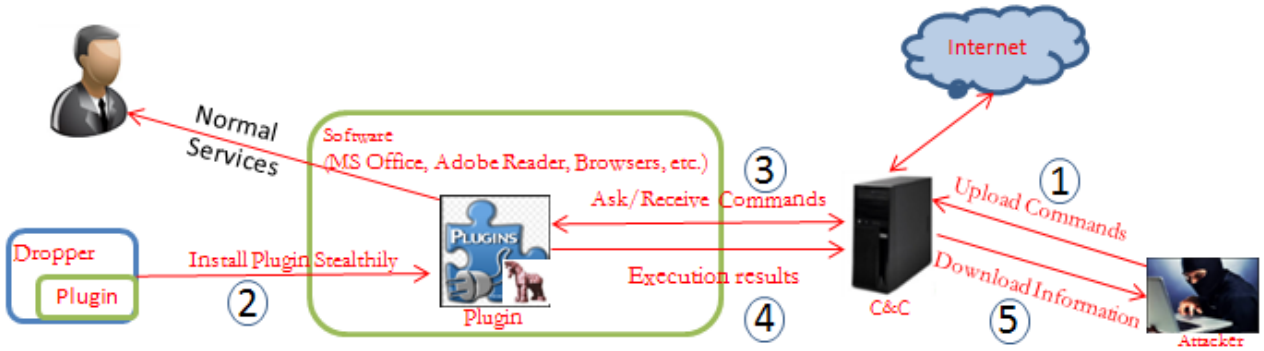

Figure 1: Software Plugin-based Malware Model

time the corresponding host application starts. As for browsers, an attacker can create a plugin in the form of browser extensions using JavaScript. With the browser extension's ability to travel across domains and its privileged access to DOM elements, an attacker can add malicious code into browser extension's Load_Event callback function using powerful API functions like XPCOM or AcitveX to implement downloading and information stealing tasks.

The C&C server is responsible for distributing commands to the plugin and storing execution results sent back from the plugin.

2.2 Working Procedures

The attacker uploads some commands to a prepared C&C server, which is waiting for a malicious plugin to connect.

The attacker creates a Dropper for the plugins and induces users to run it, i.e., the Dropper may be delivered to a user in the form of an attachment of a fishing email, it may be disguised as an image, or may be presented through other forms of social media. Once the user triggers it, the dropper installs plugins stealthily and deletes itself. In the case of MS Office, the Dropper will register the DLL (MS Office's plugin format) as its plugin. On another hand, in the case of Adobe Reader, the Dropper will release the API file (Adobe Reader's plugin format) into its "plug-ins" directory. As for browsers, the Dropper will release extension files into extension folder in the browser's profile directory and then rewrite the browser extension preference file. Then each plugin will be loaded as and when its host application starts.

The plugin will send some registration information, such as IP, OS, Username and etc. to the C&C server when it loads for the first time. Subsequently, it will ask for new commands every time it is loaded again.

When the plugin receives new commands, it will execute them and send the execution results back to the C&C server. Take browser extension for example, when a victim accesses his or her email or bank account, the browser extension will capture this event and then produce a copy of the login information from input fields, even if this sensitive information is encrypted. After capturing login information, the browser extension returns it to the C&C server through secure socket layers, ensuring that the malicious extension remains stealthy and reliable.

Accordingly, the attacker will extract information from a compromised computer successfully.

To a user it seems like the SPM is performing the advertised function; however, in the background it stealthily communicates with the C&C server and executes malicious tasks.

3. PRELIMINAY RESULTS AND EXPLANATIONS

3.1 Preliminary Results

Preliminarily, we have implemented SPM prototypes for MS Office, Adobe Reader and three mainstream browsers. All the SPM prototypes can download executable files from C&C servers, execute them upon loading, and execute remote CMD commands. Figure 2 show SPM prototypes' downloading, executing and information stealing functions for MS Office, Adobe Reader and Google Chrome. It is necessary to point out that some browser vendors have devised measures to protect users from malicious extensions. For example, Google allows extensions to be installed only if they are hosted on its Chrome web store.; Internet Explorer (IE) versions 8–11 will display security warnings and prevent scripts or ActiveX controls from running natively, and etc. Although these restrictions are strong, an attacker can still find and exploit vulnerabilities in these commercial browsers to bypass those restrictions.

More detailed experimental results are listed in Table 1.

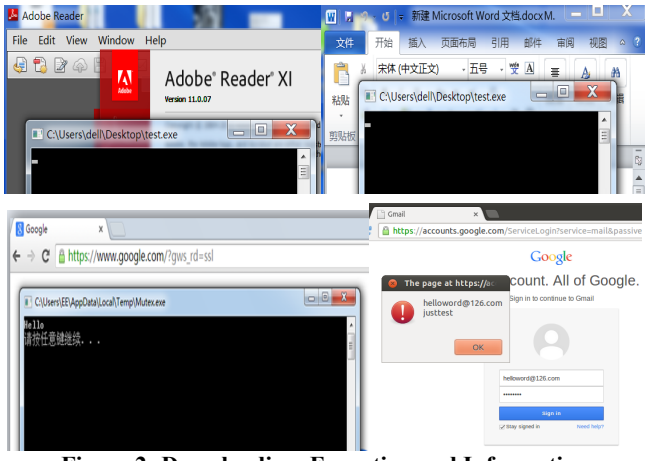

Figure 2: Downloading, Executing and Information stealing by SPM prototypes

Table 1. Functions of SPM Prototypes.

Privileges / Software	Download executable file	Execute executable file	Create threads	Execute CMD commands	Bypass Avira & Kaspersky	Online information stealing	Platform supported
Microsoft Office 2013	√	√	√	√	√	×	Win8 Pro Win7 SP1
Microsoft Office 2010	√	√	√	√	√	×	Win8 Pro Win7 SP1
Microsoft Office 2007	√	√	√	√	√	×	Win8 Pro Win7 SP1
Adobe Reader XI	√	√	×	√	√	×	Win8 Pro Win7 SP1
Adobe Reader X	√	√	×	√	√	×	Win8 Pro Win7 SP1
Adobe Reader 9	√	√	×	√	√	×	Win8 Pro Win7 SP1
IE 8-11	√	√	√	√	√	√	Win8 Pro Win7 SP1
Chrome 28–35	√	√	√	√	√	√	Win8 Pro Win7 SP1 Ubuntu 12 Mac OS X 10.8
Firefox 13–30.0	√	√	√	√	√	√	Win8 Pro Win7 SP1 Ubuntu 12 Mac OS X 10.8

3.2 Explanations

In our experiments, we were able to download and execute executable files, create threads, and execute CMD commands with SPM prototypes for MS Office and browsers as the applications impose minimalistic restrictions on their plugins. But we are not able to create threads with the Adobe Reader SPM prototype because the Adobe Reader's plugin needs to be activated and the activation of the plugin requires a key, which has to be bought from Adobe Systems. As soon as Adobe Reader detects the plugin is inactive or invalid, it discontinues the plugin's loading process. However, our experiments show that Adobe Reader can execute malicious code that exists inside the plugin, and an attacker can still make use of the plugin without activation.

4. CONLUSIONS AND FUTURE WORKS

In this poster, we present the Software Plugin-based Malware model and present an experimental implementation of the SPM prototype, proving that it is possible to infect the most popular software: Microsoft Office, Adobe Reader and mainstream browsers, with SPM. Therefore, it is necessary to consider countermeasures in advance.

In the future, we will develop an application to monitor plugins' behaviors, which can detect such kinds of SPMs.

5. ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their helpful comments for improving this paper. This work is supported by the National Natural Science Foundation of China under grant (No. 61202409) and the National High Technology Research and Development Program (863 Program) of China under grant (No. 2012AA012902).

6. REFERENCES

[1] http://www.freebuf.com/articles/system/15065.html

[2] Utakrit, Nattakant. (2009,2009). Review of Browser Extensions, a Man-in-the-Browser Phishing Techniques Targeting Bank Customers. In 7th Australian Information Security Management Conference, 2009.

.