

Open Security Research

Sponsored by Foundstone

Tuesday, January 7, 2014

Unsafe DLL Loading Vulnerabilities

By Muralidharan Vadivel.

A common issue we see in applications is the order in which they import DLLs at runtime. This is referred to as a Load Order Vulnerability that can result in local privilege escalation. It became popular a few years ago after the release of a **Microsoft Advisory** for a number of Microsoft products. In this blog post we'll dissect the vulnerability, exploitation scenarios, and how to fix it.

First let us try to understand the two different types of unsafe DLL loading vulnerabilities i.e. DLL hijacking and Component Resolution Failure:

What is DLL Hijacking?

A **Microsoft article** explains it as *"When an application dynamically loads a dynamic-link library without specifying a fully qualified path name, Windows attempts to locate the DLL by searching a well-defined set of directories in a particular order, as described in Dynamic-Link Library Search Order. If an attacker gains control of one of the directories on the DLL search path, it can place a malicious copy of the DLL in that directory. This is sometimes called a DLL preloading attack or a binary planting attack. If the system does not find a legitimate copy of the DLL before it searches the compromised directory, it loads the malicious DLL. If the application is running with administrator privileges, the attacker may succeed in local privilege elevation"*

In simple terms if an application (e.g. *Test.exe*) loads a DLL (e.g. *foo.dll*) by just the name, Windows follows a specific search order depending upon whether **"SafeDllSearchMode"** is enabled or disabled to locate the legitimate DLL. If an attacker has knowledge of this application, he can place a malicious DLL in its search path with the same name as the legitimate DLL forcing the application to load the malicious DLL, thus leading to remote code execution. SafeDllSearchMode places the user's current working directory later in the search order.


Assuming that SafeDllSearchmode is enabled, system searches the directories in the following order:

1. The directory from which the application loaded.
2. System directory (C:\Windows\System32).
3. The 16-bit system directory (C:\Windows\System).
4. The Windows directory (C:\Windows).
5. The Current Directory.
6. Directories that are listed in the PATH variables.

This issue had not been considered a serious threat because it requires local file system access on the victim's host for successful exploitation. Following section describes some of the realistic attack scenarios:

1. Combining **carpet bombing** with unsafe DLL loading: When the victim visits a malicious web page, attackers can make the browser automatically download arbitrary files. This is referred to as Carpet bomb attack. This flaw leads to remote code execution if the vulnerable application checks in the desktop directory first for resolving the DLL. For example, **Safari Web Browser** was vulnerable to carpet bomb attack. **Internet explorer 7** loads *sqmapi.dll* when it runs, suppose this DLL gets downloaded in the victim's desktop directory by a carpet bomb attack IE7 loads the malicious DLL and executes arbitrary code.
2. Sending the victim an archive file containing the shortcut to vulnerable application along with the malicious DLL . Since many vulnerable applications resolve the missing DLL in the startup directory this can be used to load up the malicious DLL upon clicking the shortcut to the vulnerable application. This can also be combined with carpet bombing attack.
3. Opening a document can load certain files placed in the same directory as the document. Attacker can send an archive containing the document along with a malicious DLL to exploit this kind of behavior.

Component Resolution Failure



Our Regular Authors

Brad Antoniewicz
Tony Lee
Gursev Singh Kalra
Robert Portvliet
Melissa Augustine
Paul Ambrosini
Tushar Dalvi

Popular Posts

Getting Started with GNU Radio and RTL-SDR (on Backtrack)

Deconstructing a Credit Card's Data

Using Mimikatz to Dump Passwords!

Windows DLL Injection Basics

Identifying Malware Traffic with Bro and the Collective Intelligence Framework (CIF)

Deobfuscating Potentially Malicious URLs - Part 1

Top 10 Oracle Steps to a Secure Oracle Database Server

Comcast and DOCSIS 3.0 - Worth the upgrade?

Manually Exploiting Tomcat Manager

Hacking KeyLoggers

Blog Archive

▼ 2014 (27)

► November (1)

► September (2)

► August (2)

► July (2)

► June (4)

► May (2)

► April (4)

► March (3)

► February (3)

▼ January (4)

Y U Phish Me? [Part 2]

Y U Phish Me? [Part 1]

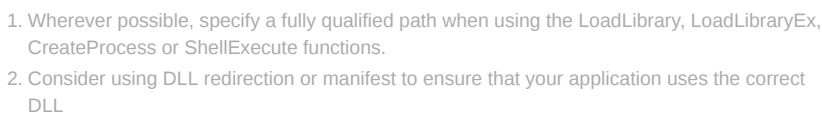
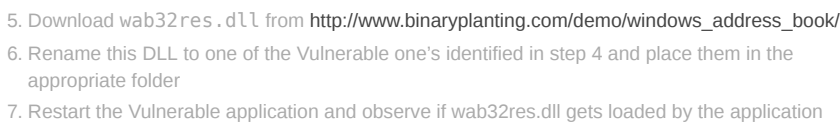
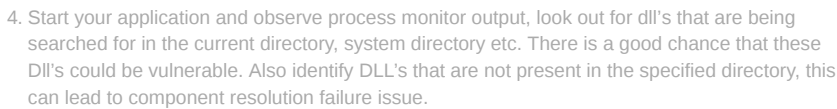
Creating Custom Peach Fuzzer Publishers

Unsafe DLL Loading Vulnerabilities

- ▶ 2013 (40)
- ▶ 2012 (60)
- ▶ 2011 (15)

We can identify these issues with the help of process monitor. To use process monitor to examine unsafe DLL loading issues:

3. Exclude the following filters
 - Process Name begins with "Name of the process"
 - Operation is RegQueryValue
 - Operation is RegOpenKey



- 3. When using the standard search order, make sure that safe DLL search mode is enabled. This places the user's current directory later in the search order, increasing the chances that Windows will find a legitimate copy of the DLL before a malicious copy
- 4. Consider removing the current directory from the standard search path by calling SetDllDirectory with an empty string (""). This should be done once early in process initialization, not before and after calls to LoadLibrary. Be aware that SetDllDirectory affects the entire process and that multiple threads calling SetDllDirectory with different values can cause undefined behavior. If your application loads third-party DLLs, test carefully to identify any incompatibilities
- 5. Do not use the SearchPath function to retrieve a path to a DLL for a subsequent LoadLibrary call unless safe process search mode is enabled

Posted by [OpenSecurity Research](#) at [11:14 AM](#)
Labels: [application security](#)

1 comment:

iphone hookup February 14, 2014 at 8:54 AM
Nice post. Thanks for sharing
Reply

Enter your comment...

Comment as: [Google Account](#)

Publish

Preview

[Newer Post](#)

[Home](#)

[Older Post](#)

Subscribe to: [Post Comments \(Atom\)](#)

All content provided here is purely for educational purposes only. Review state and local laws before partaking in any activity. The views and statements here have not been reviewed, approved, or endorsed by Foundstone, McAfee, or Intel.

Awesome Inc. theme. Powered by [Blogger](#).