


ScanNow DLL Search Order Hijacking Vulnerability and Deprecation

 blog.rapid7.com/2015/12/21/scannow-dll-search-order-hijacking-vulnerability-and-deprecation/

December 21, 2015

Overview

On November 27, 2015, Stefan Kanthak contacted Rapid7 to report a vulnerability in Rapid7's ScanNow tool. Rapid7 takes security issues seriously and this was no exception. In combination with a preexisting compromise or other vulnerabilities, and in the absence of sufficient mitigating measures, a system with ScanNow can allow a malicious party to execute code of their choosing leading to varying levels of additional compromise. In order to protect the small community of users who may still be using ScanNow, Rapid7 has made the decision to remove ScanNow and advises any affected users to remove ScanNow from any system that still has it.

Vulnerability Details

Rapid7's ScanNow is a collection of several separate, standalone executables built over the past several years designed to give users and the community the ability to quickly audit themselves for some higher profile vulnerabilities that have made varying levels of headlines during this time:

The vulnerability disclosed to Rapid7 by Stefan is a generic vulnerability whose most official name is DLL Search Order Hijacking, but is also referred to as DLL side loading, DLL pre-loading, binary planting, binary carpet bombing, or similar names. DLL search order hijacking went more mainstream in 2010 when ACROS Security published extensive information about it here and has affected hundreds of products over the years and continues to do so.

This class of vulnerability occurs when a Windows application attempts to load a DLL or other library and does so with an unqualified search path. When the search path for the DLL is not sufficiently qualified, Windows has a predefined list of places that it will check for the library in question. Included in that list are locations that could be untrusted or insecure, and if the library in question is found in one of those locations, it is possible that the loaded library contains malicious code which could lead to arbitrary code execution when the target executable is launched.

Upon investigating this vulnerability report, Rapid7 discovered that none of code written by Rapid7 for ScanNow utilizes any of the potentially vulnerable Windows API code for loading libraries or executing processes, let alone in such a way that allows the vulnerability to be exploitable in any scenario. Instead, the vulnerability is present in ScanNow because of the way ScanNow is packaged and distributed, part of which includes 7-zip. 7-zip can be used for several things. In the case of ScanNow, it is used to create a standalone self-extracting archive executable (SFX), which is basically just an

executable that, when run, unpacks the actual ScanNow executable along with any resources it needs, and then runs ScanNow itself. The root cause appears to be the same thing that affected the [Mozilla Foundation](#) in 2012 after a [posting to full-disclosure](#). It appears that this vulnerability remains unfixed, and another advisory [posted by Stefan](#) shortly after he contacted us indicates that in addition to 7-zip itself being vulnerable to DLL search order hijacking, all self-extracting archives created with 7-zip are also vulnerable.

At the current time, Rapid7 has assigned a CVSS vector of [\(AV:N/AC:H/Au:N/C:C/I:C/A:C\)](#) with a corresponding score of 7.6 to this vulnerability, but we also realize the particulars around the real risk posed by this vulnerability are complicated and not easily reflected in a CVSS vector.

Exploitation Scenarios

When DLL search order hijacking vulnerabilities are discussed, the topic of whether or not these are remotely exploitable is an almost inevitable point that is raised and quickly detracts from the issue at hand due to the reaction it typically invokes. While we are of the opinion that calling this a remotely exploitable vulnerability is a bit inaccurate, it is important to understand that there are "remote like" characteristics of this as shown below.

Exploiting a DLL search order hijacking vulnerability on Windows is not all that much different than exploiting LD_PRELOAD and similar style vulnerabilities in the UNIX world, namely that to exploit it, *somehow* a malicious library must be placed in a location that will be searched by the target application *before* the correct, expected library is found and loaded by an affected vulnerable application. There are numerous ways this happens, including:

- A malicious `_local_user` or process that can write arbitrary content to a file that is located in a directory that will be searched when trying to locate a DLL for loading by the target application. This could happen as the result of another previously exploited and unrelated vulnerability.
- That malicious DLLs have been somehow delivered to inadequately protected directories using something like a [drive-by download vulnerability](#).
- Social engineering.

Additionally, the target system and application must be free of hardening measures designed to mitigate or prevent exploitation of this style of vulnerability.

As a trivial example of how this could be exploited, I used the following code, which simply launches the Windows calculator, `calc.exe`, whenever this DLL is loaded:

```
#include <windows.h>
```

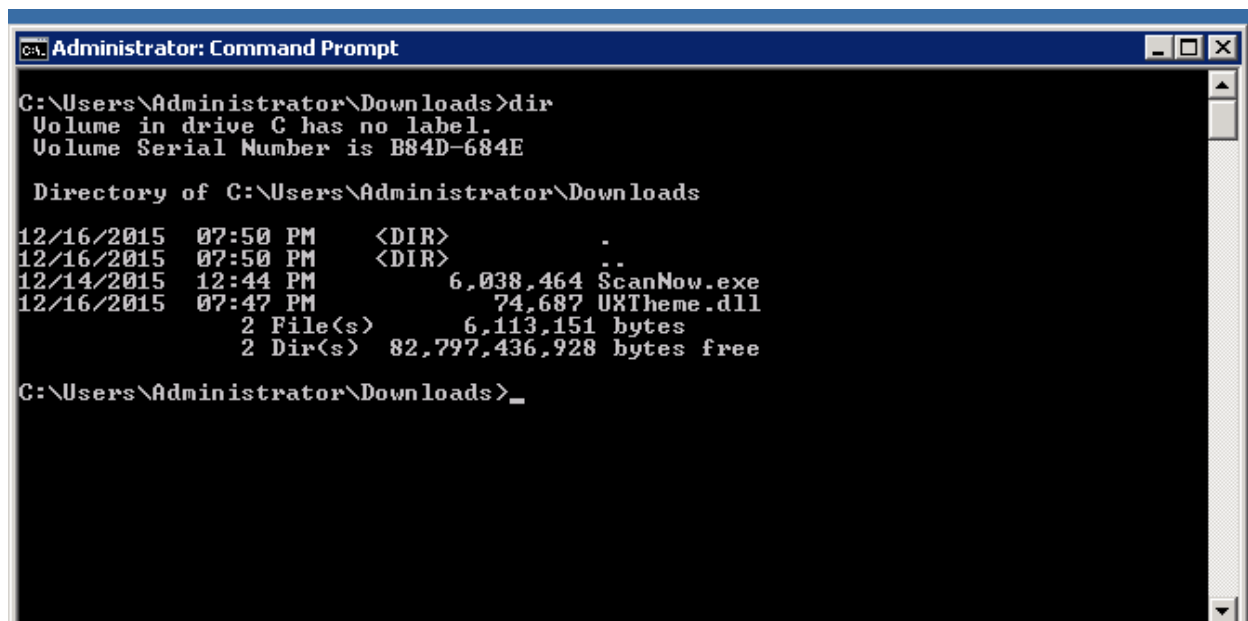
```
int hijack()  
{  
    WinExec("calc", 1);  
    return 0;  
}
```

```
BOOL WINAPI DllMain (HINSTANCE hinstDLL, DWORD fdwReason, LPVOID lpvReserved)  
{  
    if (fdwReason == DLL_PROCESS_ATTACH)  
    {  
        hijack();  
    }  
  
    return TRUE;  
}
```

Then, link and compile as usual. In this case, I used mingw:

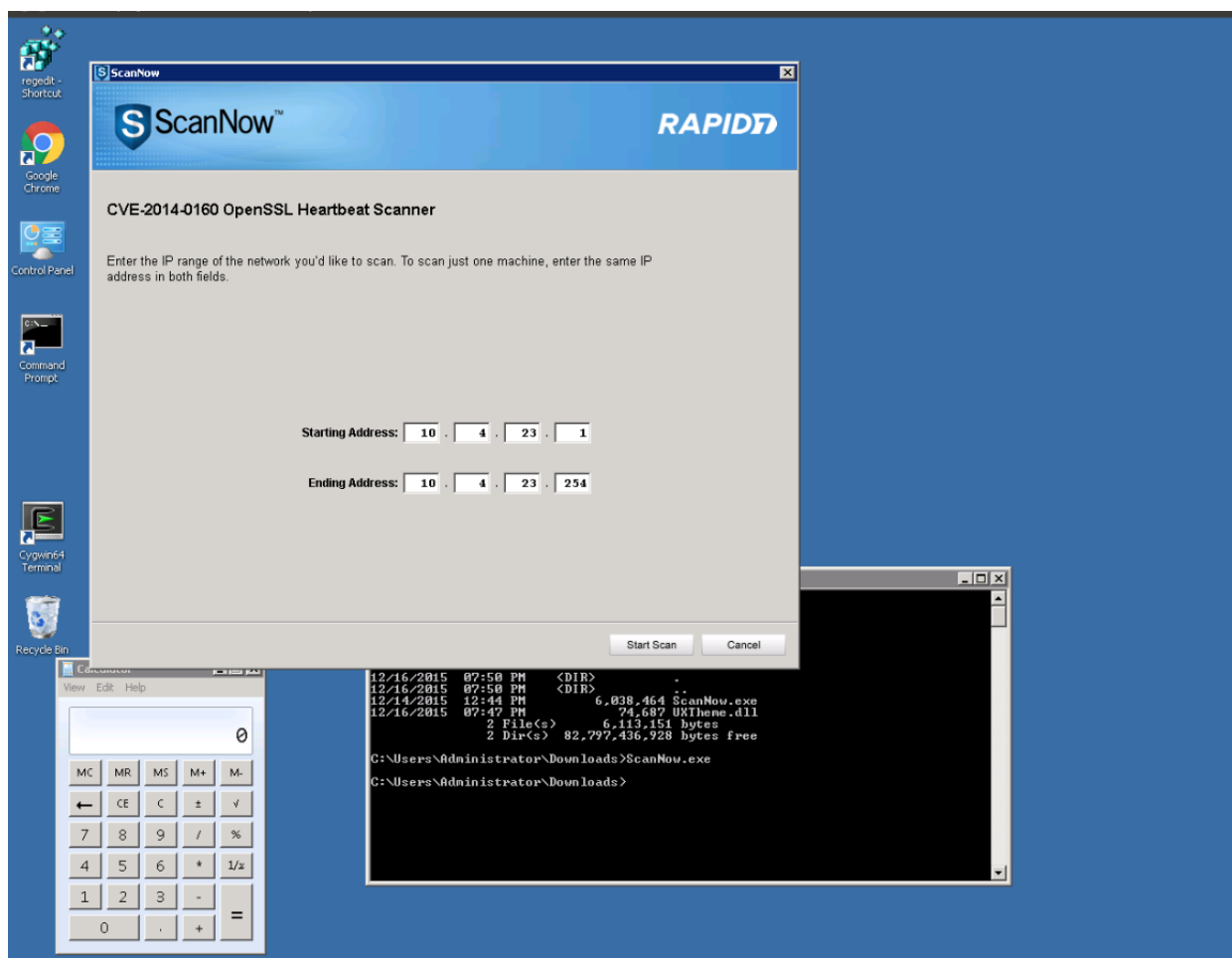
```
$ i686-w64-mingw32-gcc -c -o dll-hijack.o dll-hijack.c  
$ i686-w64-mingw32-gcc -shared -o UXTheme.dll dll-hijack.o -lcomctl32 -Wl,--  
subsystem,windows
```

Then, I copy the malicious DLL (there can be several) to the directory where ScanNow lives or to the directory from which the call to ScanNow is made. In this particular case, I simply deposited the malicious example DLL in my Downloads folder:



```
Administrator: Command Prompt  
C:\Users\Administrator\Downloads>dir  
Volume in drive C has no label.  
Volume Serial Number is B84D-684E  
  
Directory of C:\Users\Administrator\Downloads  
12/16/2015  07:50 PM    <DIR>          .  
12/16/2015  07:50 PM    <DIR>          ..  
12/14/2015  12:44 PM             6,038,464 ScanNow.exe  
12/16/2015  07:47 PM             74,687 UXTheme.dll  
                2 File(s)          6,113,151 bytes  
                2 Dir(s)      82,797,436,928 bytes free  
  
C:\Users\Administrator\Downloads>_
```

Then, when ScanNow is executed, either with an unqualified path when in Downloads or with a fully-qualified path, you can see that calc was executed, proving the existence of this vulnerability:



Obviously, the final part of exploiting this is getting the target system to somehow run ScanNow in a vulnerable environment which could happen with social engineering, exploiting another perhaps unrelated vulnerability or just getting lucky.

Remediation and Mitigation

The nature of how ScanNow is built, distributed and used in combination with the particulars surrounding the vulnerability, its exploitation and the ways to mitigate it complicated Rapid7's response to this vulnerability.

It should be noted that there are various hardening techniques provided by Microsoft and others to help mitigate or prevent this class of vulnerabilities, however they are far from fool-proof and Rapid7 has experienced limited success when utilizing them to mitigate this particular vulnerability. Some of these were:

1. [Microsoft Security Advisory 2269637](#) is not especially relevant in this particular case. This is essentially Microsoft's response to when this class of vulnerability first really went like wildfire back in 2010, and in it they list all vulnerabilities to date in Microsoft products that were of the same basic class (DLL Search Order Hijacking), of which there have been 28 MS advisories and a handful of generic MS KnowledgeBase or TechNet items since its initial writing in 2010. That is quite a few. However, only the solutions in two of them are even potentially relevant when discussing this vulnerability, and, as I describe below, most of these were also found to be problematic.

2. Dynamic-Link Library Security (Windows) is not really relevant in our case because we don't control the code that is making the insecure calls. Otherwise this would be a fine option.
3. Dynamic-Link Library Search Order (Windows) sounds like it should work. It basically helps when an application doesn't follow the previous recommendation, and the lock-down needs to happen on the client running the application rather than when it is built. We found this unable to prevent against exploitation of this vulnerability (in fact, the screenshots above were done on a system configured as suggested in this link). Why is currently unknown and may be an area for future research.
4. CAPEC - CAPEC-471: DLL Search Order Hijacking (Version 2.8) does an OK job at explaining what a DLL search order hijacking vulnerability is, how it is exploited and suggests CAPEC - CAPEC-159: Redirect Access to Libraries (Version 2.8) as a solution. That solution actually gives 3 options. The first two are essentially identical to the previous two bullet points (source code modifications, which are, again, not applicable in our case and client-side hardening, which we found to be ineffective in this instance). The third and final suggestion is to sign system DLLs, the responsibility for which is Microsoft's in this case (right?), and would only be applicable if the affected application was only insecurely loading system DLLs (as opposed to non-system DLLs). In the case of ScanNow, both Stefan's POC and ours utilize UXTheme.dll, which normally exists as uxtheme.dll on most Windows operating systems since XP in the usual locations buried under C:\Windows. Had a signing solution been available, it is assumed that the vulnerability would be mitigated to some extent, but signing DLLs is a relatively easy process and there are ways to attempt to bypass signing checks like this. In other words, signing helps, but perhaps not when facing a super determined adversary. Furthermore, Stefan's POC which he has been using for several years in the security community, called sentinel, is signed, and it was reported that there were other avenues to exploit this class of vulnerability in ScanNow beyond UXTheme.dll which may or may not be signed by default.
5. CAPEC - CAPEC-159: Redirect Access to Libraries (Version 2.8), is a similar but slightly different vulnerability, and unfortunately its solutions mimic what others have suggested above which were shown to be not relevant or ineffective for various reasons.
6. Both CWE - CWE-426: Untrusted Search Path (2.9) and CWE - CWE-427: Uncontrolled Search Path Element (2.9) cover vulnerabilities and solutions that are practically identical to everything above with the aforementioned flaws.
7. Oftentimes, suggestions are made to advise against or prevent the execution of executables that live in unprotected locations and to avoid running executables when the current working directory is untrusted. Download and other temporary directories are great targets for this for this vulnerability, and while following this

advice would help, it also makes the system a bit of a pain for the end user. The result is basically finding a compromise between security and usability, and usability may win in many environments in this regard

8. UAC Group Policy Settings and Registry Key Settings shows that there are some settings that exist on UAC enabled systems to control elevation to administrator by installers automatically and, when *disabled* require that the user can authenticate successfully as an administrator before allowing the installer to utilize administrative privileges. When this setting is disabled, all it does is prevent the immediate and easy jump from exploiting a DLL search order hijacking vulnerability in an installer to obtaining administrative privileges. In other words, the vulnerability was still exploited but the current damage was stopped before further elevation of privilege. While this setting is *disabled* on all all Windows enterprise operating systems, Windows home users systems have this setting *enabled* by default and are subsequently at risk for this turning into a quicker EOP.

In short, there appear to be very few workable options for the occurrence of this vulnerability in ScanNow, and it seems like this is a predicament many will have to contend with when faced with a DLL search order hijacking vulnerability. Both of these areas may be ripe for future research.

Anyone who has downloaded ScanNow is advised to locate and remove the affected executables. For any users who still have a need for ScanNow, both Metasploit and Nexpose have better coverage for the vulnerabilities that ScanNow previously covered. For anyone who registered when downloading ScanNow over the years, Rapid7 will also be attempting to reach these users to advise them of the situation.

Rapid7 would like to again thank Stefan Kanthak for responsibly disclosing this vulnerability.