THM : OVERPASS 2 – HACKERD

https://tryhackme.com/room/overpass2hacked

This is a beginner friendly room simulating an Incident Response team. Apparently this is a trilogy but in order to proceed with this room, no prior knowledge of the previous room is required. Some labs to go through in order to make this room slightly easier to go through is TryHackMe's *"Wireshark: The Basics"* (https://tryhackme.com/room/wiresharkthebasics) and Hack The Box's *"Intro to Network Analysis"* room.

Objective: The organization called Overpass has its production server hacked. We have a traffic capture file and we are supposed to analyze it in order to understand the attackers actions and be able to hack back in and take back control.

I am going to be cliche and use Wireshark as many write-ups have simply because using it to analyze the PCAP file is easier to analyze due to the rich features the tool provides.

In a basic network traffic analysis, the general steps with a captured packet are

>Identify clients, servers, ports and protocols used
> reduce the noise by filtering what looks normal
> Look at specific data, protocols used, ports,
> Look at what is irregular/ suspicious and understand what is happening.

Generally, a server uses a well known port while the client will connect on a random high level port. Looking at the PCAP, we see that 192.168.170.145 is connecting to port 80 of 192.168.170.159 using a high level port number. We now know the host and the server. Maybe they are on the same network as this IP address is a private class.

*In Wireshark, the **display filter** and **capture filter** affect the data which is shown to us but their implementation is the difference. The display filter will only show packets that match a requirement, the other packets are not discarded, they are merely not shown. A capture filter determines what packets will be captured and which will be dropped. Capture filters are set before a packet capture.* As we know Port 80 usually deals with HTTP we can filter looking for HTTP traffic.

We are using the display filter and simply enter *http* in order to apply the filter. Our assumptions concerning the client and server are further confirmed by looking at the responses. Clients send requests to the server using methods such as GET. While servers, respond with status codes such as 200 OK. We see a directory and we see a POST request that uploads a php file to the server.

Following the TCP stream for the uploaded file, as it is HTTP - therefore in clear text – we can see that a PHP reverse shell was uploaded to the server and it is using port 4242. Again, confirmation on the client and server as IP being connected is *.145 . We can start filtering again for traffic on Port 4242.
tcp.port == 4242

blackgrease

Following the TCP stream we can see that the attacker was successful in getting a reverse connection back to his machine. We can see the commands that we carried out.

> He upgrade his shell to make it interactive
> He listed the files in the web directory and found a hidden file.
> He read the file and found some scrambled text which he managed to unscramble and find a password for the user James which he used to escalate his privileges vertically.
> He saw that the user James has root privileges.
> He used this to read the *cat*/shadow file and see the password hashes for other users.
> A ssh backdoor script was downloaded from GitHub where he installed a SSH backdoor on Port 2222

The user passwords are hashed in HMAC-SHA1 format and can be cracked using the fasttrack word-list.

john --wordlist=/usr/src/fasttrack/wordlist.txt users.txt

By this stage, 1/3 of the guided questions should be answered.

For the 2/3 of the questions, we need to clone the git repository -that was used to install the backdoor – on our machine. Looking at how to run the script using *./backdoor -h* we see the default hash that is used. Looking at the source code of backdoor, we see that at the bottom, the salt is hard-coded into the script. The has that is used the one that we see in the PCAP analysis.

Using the found out results, we can use this to crack the hash and crack the password. It is important to remember that the hash is salted. I used John the Ripper as I was having configuration issues with Hashcat. In JTR, in order to crack a salted password an syntax is required.

john -form=dynamic='sha512($p.$s)' hash.txt --wordlist=/usr/share/wordlists/rockyou.txt

> -form=dynamic=’sha512($p.$s)’ tells John that the hash is sha512 and the format is password_hash:salt
> hash.txt = the hash and the salt in the format hash:salt

The default password file used in JTR will suffice.

We should now have the password and 2/3 of the rooms questions answered.

For the final third of the questions. We run a nmap scan on the machine in order to see what we have.

nmap -Pn -n -v –disable-arp-ping -A $IP_ADDRESS -oN Overpasss_Nmap

> -Pn : disable host discovery, assume the host is up
> -n : do not perform name resolution
> -v : verbose; provide more information
> --disable-arp-ping : part of disabling the host-discovery process

> -A perform an aggressive scan that consists of service version detection, os detection and default NSE scripts.

> -oN : normal output/save to 'Overpass_Nmap'

```
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 e4:3a:be:ed:ff:a7:02:d2:6a:d6:d0:bb:7f:38:5e:cb (RSA)
|   256 fc:6f:22:c2:13:4f:9c:62:4f:90:c9:3a:7e:77:d6:d4 (ECDSA)
|_  256 15:fd:40:0a:65:59:a9:b5:0e:57:1b:23:0a:96:63:05 (ED25519)
80/tcp   open  http    Apache httpd 2.4.29 ((Ubuntu))
| http-methods:
|_  Supported Methods: OPTIONS HEAD GET POST
|_http-server-header: Apache/2.4.29 (Ubuntu)
|_http-title: LOL Hacked
2222/tcp open  ssh     OpenSSH 8.2p1 Debian 4 (protocol 2.0)
| ssh-hostkey:
|_  2048 a2:a6:d2:18:79:e3:b0:20:a2:4f:aa:b6:ac:2e:6b:f2 (RSA)
```

The default SSH port and the SSH Backdoor on port 2222 are both open. Attempting to SSH into the default SSH port using the user of James and both passwords, fails. It succeeds – using the user James - when we SSH into the backdoor with the password that we have cracked. We have hacked our way back in.

Moving back one directory we find the *user.txt* flag.

Running sudo -l using any of the passwords that we have fails. Looking in the home directory of James we run ls -la in order to find any hidden files/directories. We find that there is a .suid_bash script that has the SUID bit set.

On GTFOBins there is an exploit for the bash command if it has the SUID bit set.

[SUID bit= a permission that allows other users to run a binary/ script with the same privileges as the owner]

Snippet from GTFOBins

If the binary has the SUID bit set, it does not drop the elevated privileges and may be abused to access the file system, escalate or maintain privileged access as a SUID backdoor. If it is used to run sh -p, omit the -p argument on systems like Debian (<= Stretch) that allow the default sh shell to run with SUID privileges.

Running

./suid_bash -p

on the system gives us a root shell and allows us to cd /root and read the *root.txt* flag. Having grabbed our flags and running as root, we can kick the attacker out and secure our system from within.

Room Complete.

blackgrease