NMAP Fundamentals

@sidorocs

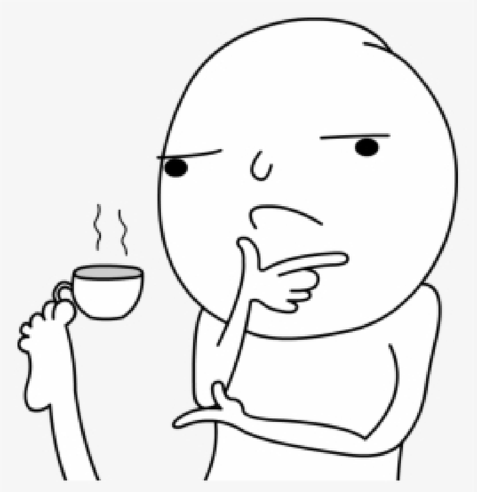
Mathieu

sidorocs@proton.me

Nmap Software LLC

Survey

Who uses nmap regularly ? Who occasionnaly uses it ? who never used it ?



01

What’s nmap

Gives a short introduction about nmap and its basic uses

02

Under the hood

Understanding advanced nmap options, and what is really happenning behind the CLI

03

Scripting

Experiment custom NSE script

Introduction

Nmap = network mapper

- Released as a simple Linux-only port scanner in 1997
- Open source and under constant developpment
- Multiple features released over the time...
- If you are interested, you can go deeper in nmap history : https://nmap.org/book/history-future.html

What are nmap uses ?

Mapping an infrastructure

Gain a clear overview of the network by identifying all connected devices, services, and their relationships.

Identify entry point & insecure configuration

Detect exposed services and open ports that could be leveraged by attackers to gain unauthorized access.

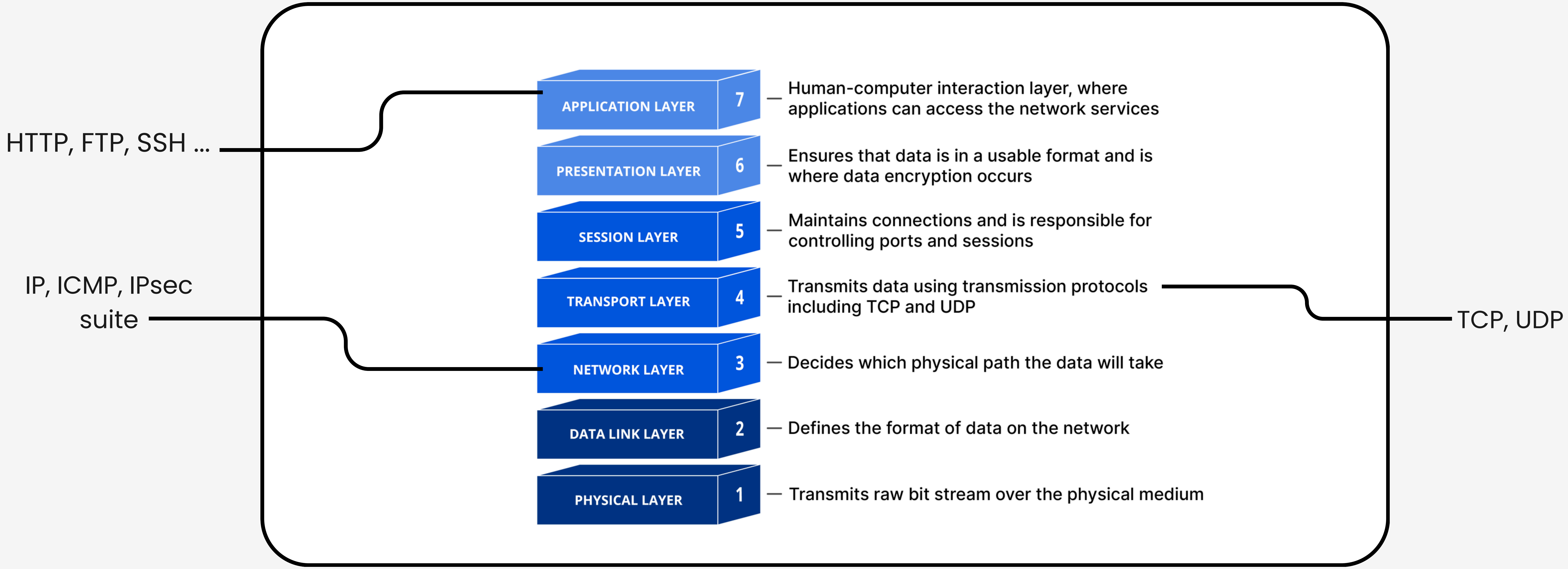
Check firewall and IDS settings

Usefull to test firewall behaviour ans IDS rules using decoys, packets fragmentation ...

Recall – Basic Network notion

OSI model

The Open Systems Intercommunication (OSI) model is a conceptual model that represents how network communications work



HTTP, FTP, SSH ...

APPLICATION LAYER 7 — Human-computer interaction layer, where applications can access the network services

PRESENTATION LAYER 6 — Ensures that data is in a usable format and is where data encryption occurs

SESSION LAYER 5 — Maintains connections and is responsible for controlling ports and sessions

TRANSPORT LAYER 4 — Transmits data using transmission protocols including TCP and UDP

TCP, UDP

IP, ICMP, IPsec suite

NETWORK LAYER 3 — Decides which physical path the data will take

DATA LINK LAYER 2 — Defines the format of data on the network

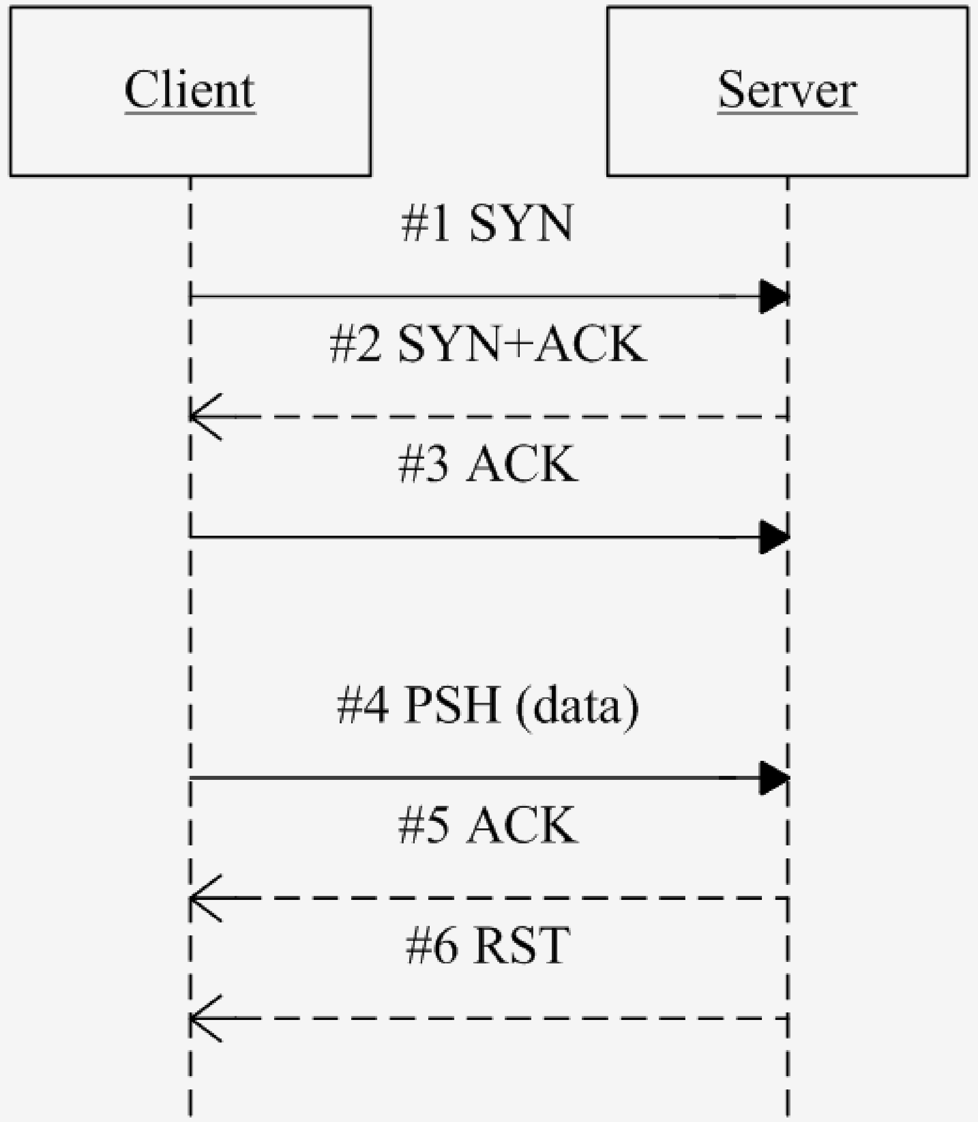
PHYSICAL LAYER 1 — Transmits raw bit stream over the physical medium

Recall – Basic Network notion

TCP Handshake

Transmission Control Protocol

TCP (Transmission Control Protocol) uses a three-way handshake (aka TCP-handshake, three message handshake, and/or SYN-SYN-ACK) to set up a TCP/IP connection.



Recall – Basic Network notion

UDP

User datagram protocol

No handshake, no control on the sequence … I mean almost …

nmap basic uses

nmap functions overview

nmap has several functions for diffrent pruposes

- Host discovery
- Port scanning
- Service enumeration and detection
- OS detection
- Scriptable interaction with the target service (Nmap Scripting Engine)

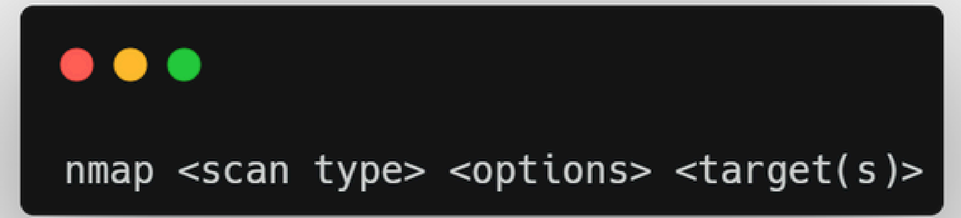
nmap basic uses

nmap functions overview

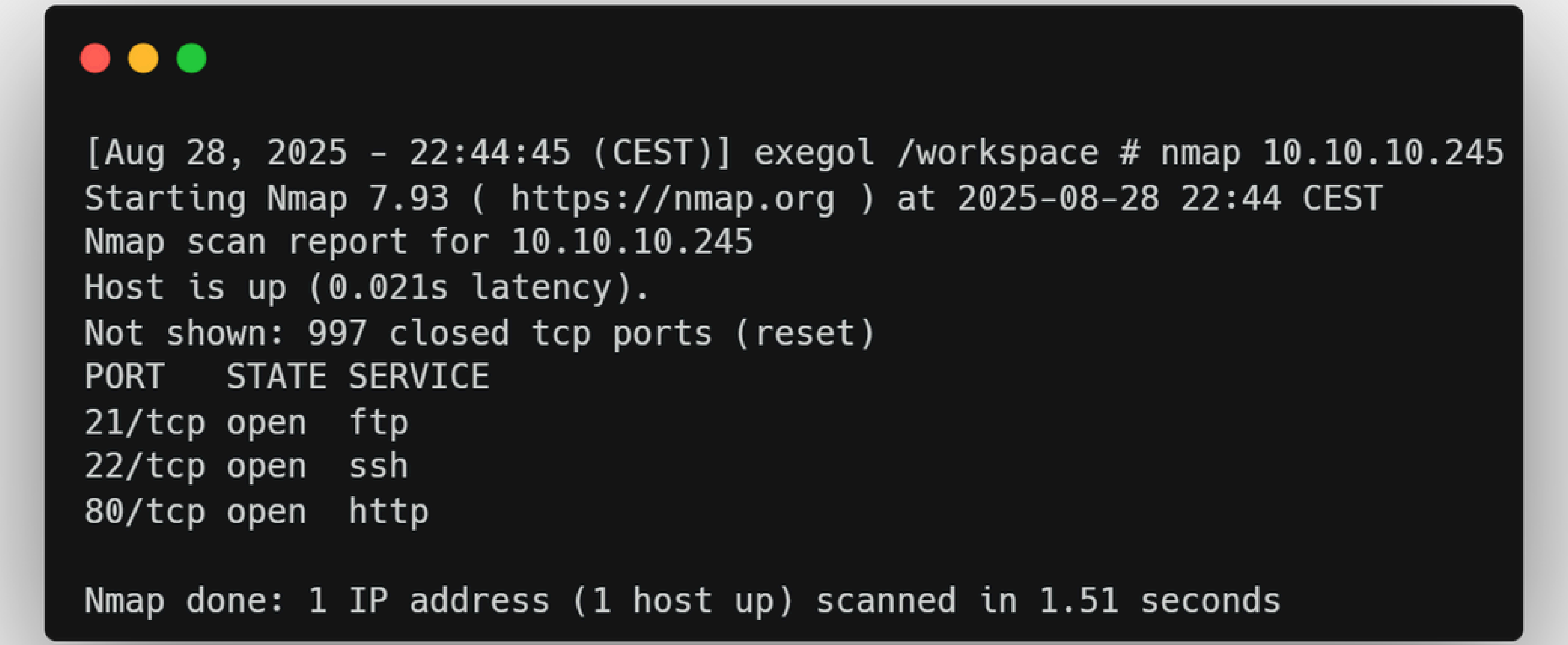
nmap has several functions for diffrent pruposes

- Host discovery
- Port scanning
- Service enumeration and detection
- OS detection
- Scriptable interaction with the target service (Nmap Scripting Engine)

Basic nmap command :



```
nmap <scan type> <options> <target(s)>
```

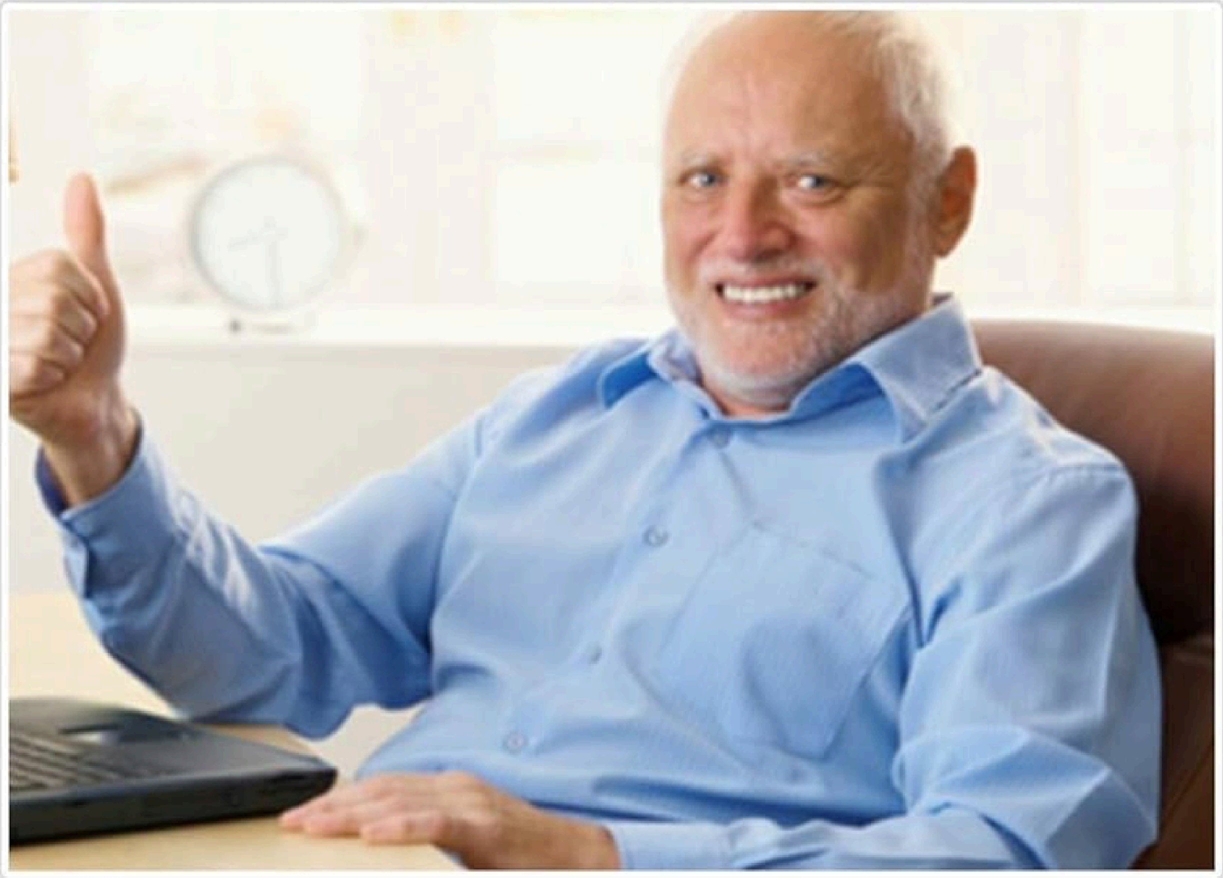


```
[Aug 28, 2025 - 22:44:45 (CEST)] exegol /workspace # nmap 10.10.10.245
Starting Nmap 7.93 ( https://nmap.org ) at 2025-08-28 22:44 CEST
Nmap scan report for 10.10.10.245
Host is up (0.021s latency).
Not shown: 997 closed tcp ports (reset)
PORT   STATE SERVICE
21/tcp open  ftp
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 1.51 seconds
```

EZ nmap

You found your open ports – COOL !

I guess that's all for nmap



Thanks for your attention !

01

What’s nmap

Give a short introduction about nmap and its basic uses

02

Under the hood

Understanding advanced nmap options, and what is really happenning behind the CLI

03

Scripting

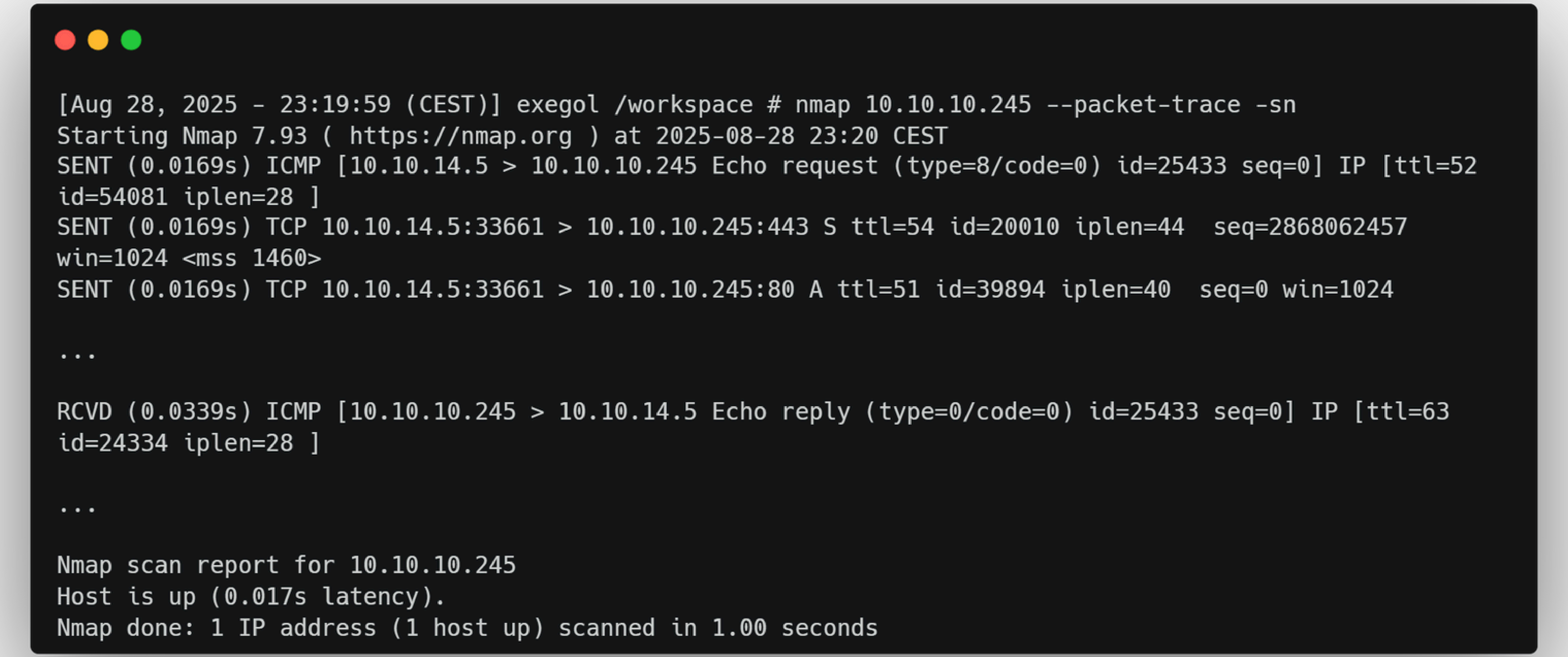
Experiment custom NSE script

Detection (ICMP – arp prob)

What will nmap do ?

Before any port scan, nmap will try to figure out wether the host is up or not

- By default : ICMP echo prob discovery



```
[Aug 28, 2025 - 23:19:59 (CEST)] exegol /workspace # nmap 10.10.10.245 --packet-trace -sn
Starting Nmap 7.93 ( https://nmap.org ) at 2025-08-28 23:20 CEST
SENT (0.0169s) ICMP [10.10.14.5 > 10.10.10.245 Echo request (type=8/code=0) id=25433 seq=0] IP [ttl=52
id=54081 iplen=28 ]
SENT (0.0169s) TCP 10.10.14.5:33661 > 10.10.10.245:443 S ttl=54 id=20010 iplen=44  seq=2868062457
win=1024 <mss 1460>
SENT (0.0169s) TCP 10.10.14.5:33661 > 10.10.10.245:80 A ttl=51 id=39894 iplen=40  seq=0 win=1024

...

RCVD (0.0339s) ICMP [10.10.10.245 > 10.10.14.5 Echo reply (type=0/code=0) id=25433 seq=0] IP [ttl=63
id=24334 iplen=28 ]

...

Nmap scan report for 10.10.10.245
Host is up (0.017s latency).
Nmap done: 1 IP address (1 host up) scanned in 1.00 seconds
```

But he can also perform :

- ARP prob discovery (-PR)
- ICMP timestamp request (-PP)
- TCP/UDP prob discovery (-PS, -PU)

→ read the official documentation !

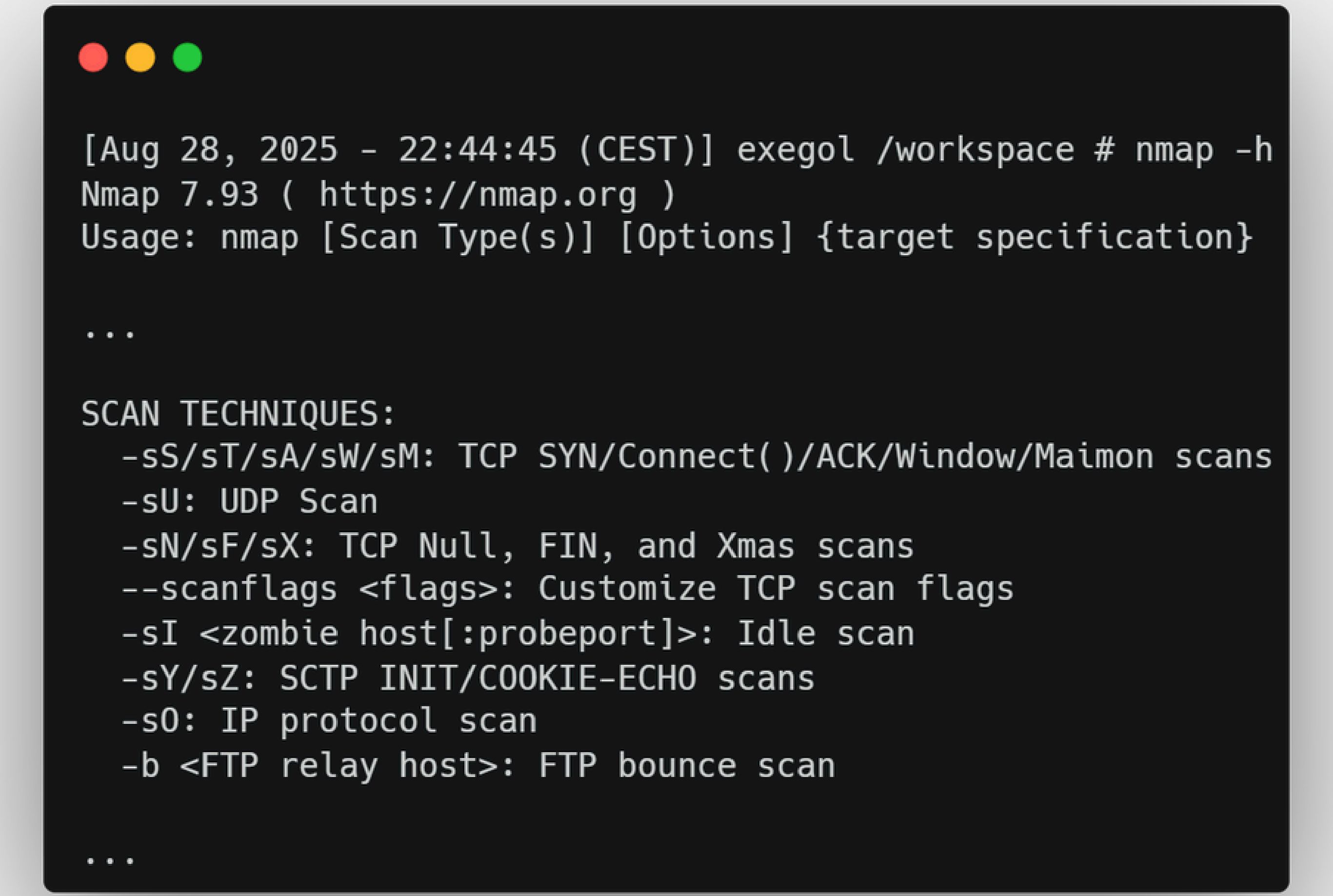
(You can also disable host discovery for your scans)

Scan type

multiple scan type for multiple purposes

TCP, UDP, other stuff ...

- Different TCP scans
- UDP scan
- Weird TCP scans (send weird TCP packet just to see if the server will respond by a RST or not) for firewall bypasses
- zombie scan (WTF)
- Other scan ... (deprecated for some of them)



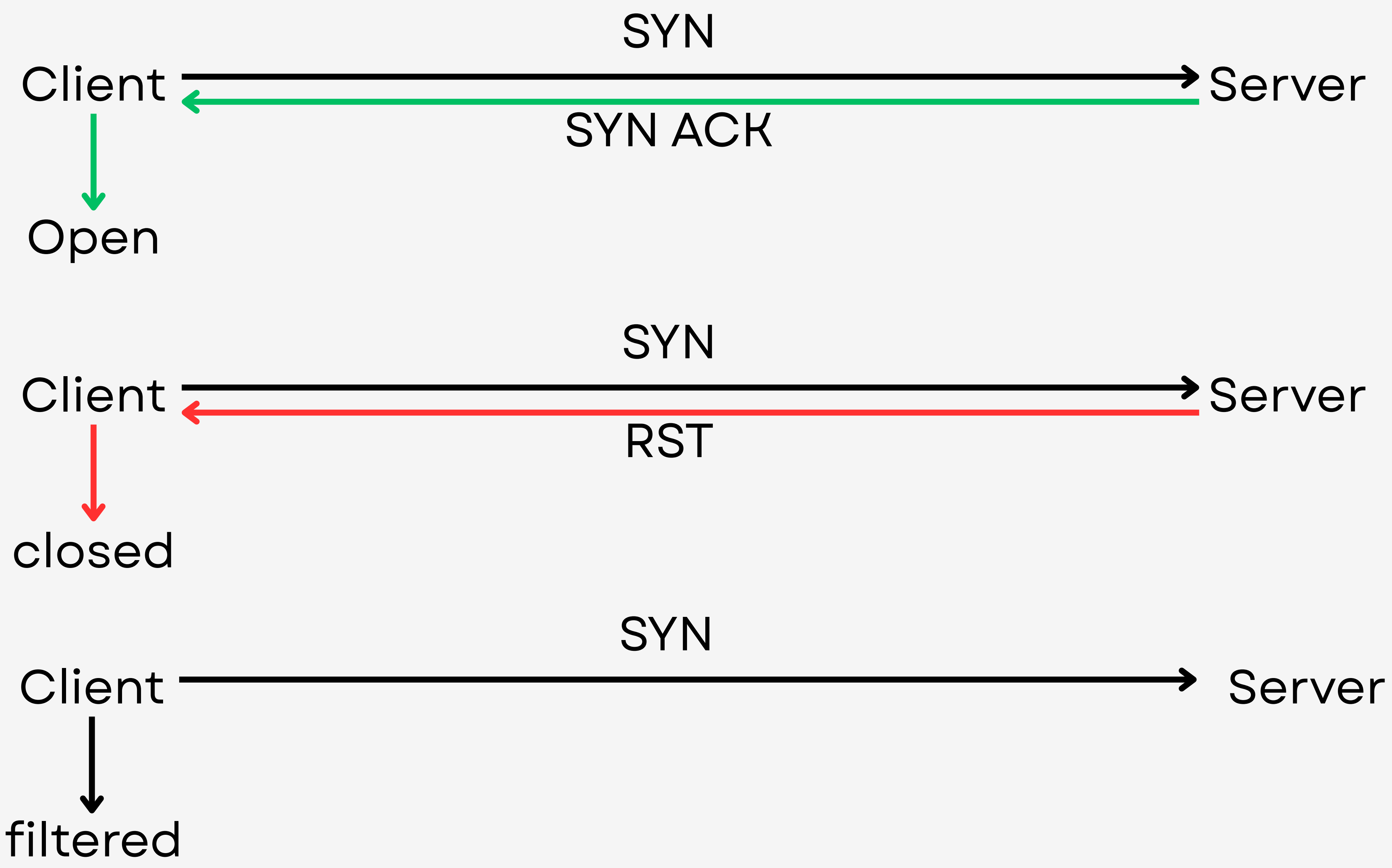
```
[Aug 28, 2025 - 22:44:45 (CEST)] exegol /workspace # nmap -h
Nmap 7.93 ( https://nmap.org )
Usage: nmap [Scan Type(s)] [Options] {target specification}

...

SCAN TECHNIQUES:
  -sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
  -sU: UDP Scan
  -sN/sF/sX: TCP Null, FIN, and Xmas scans
  --scanflags <flags>: Customize TCP scan flags
  -sI <zombie host[:probeport]>: Idle scan
  -sY/sZ: SCTP INIT/COOKIE-ECHO scans
  -sO: IP protocol scan
  -b <FTP relay host>: FTP bounce scan

...
```

TCP Scan

nmap tcp scan in a nutshell : SYN scan

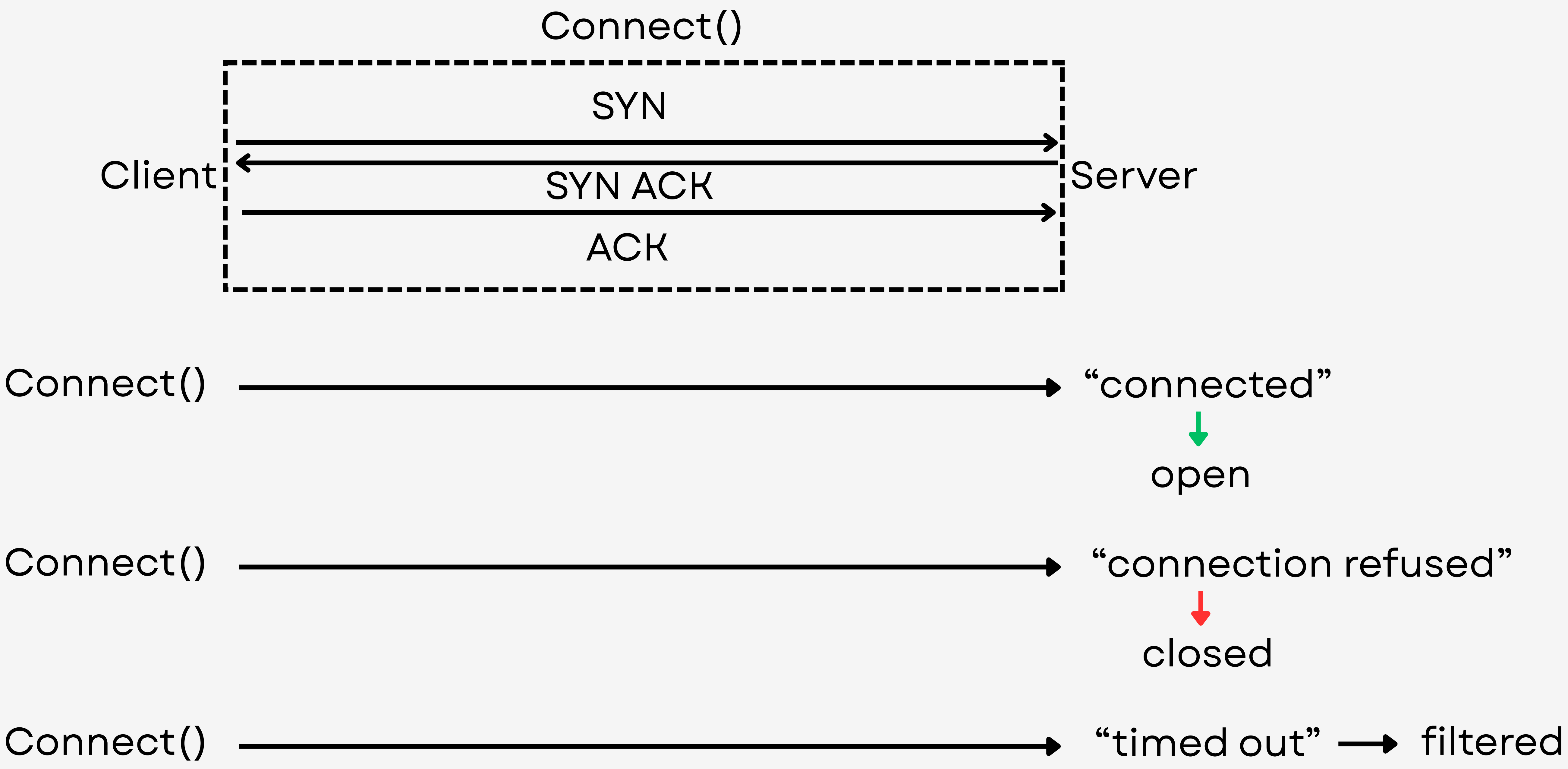
SYN scan : only in root mode

SYN can only be performed with root permissions (or the CAP_NET_RAW capability) because nmap handly craft network packets and read the responses packtes whereas connect scan can be performed as an unprivileged user because it uses the connect() function from the API.



TCP Scan

nmap tcp scan in a nutshell : connect scan



Connect()

Client

SYN

SYN ACK

ACK

Server

Connect() → “connected” → open

Connect() → “connection refused” → closed

Connect() → “timed out” → filtered

SYN scan : only in root mode

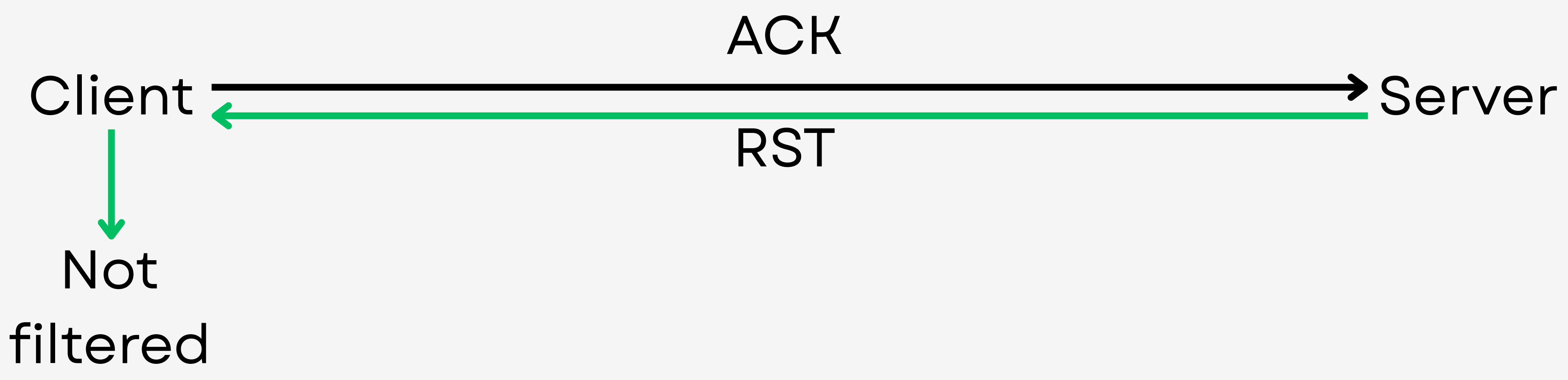
SYN can only be performed with root permissions because nmap handly craft network packets and read the responses packtes whereas connect scan can be performed as an unprivileged user because it uses the connect() function from the API.

TCP Scan

nmap tcp scan in a nutshell : ACK scan

ACK scan

ACK scan is a scan type used to identify filters or firewall. If a port respond RST to an ACK packet with no prior TCP handshake, it will considered as not filtered. In case there is no response from the server, the port will be marked as filtered.



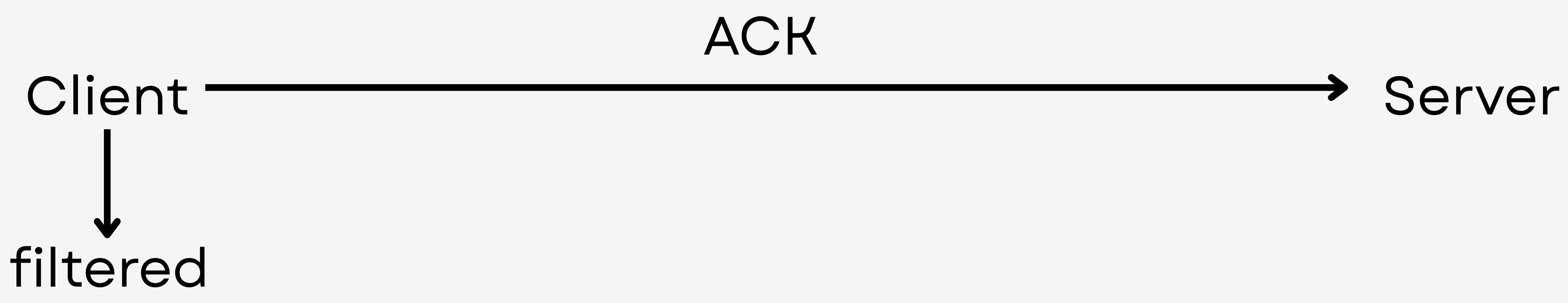
ACK

Client

Server

RST

Not filtered



ACK

Client

Server

filtered

TCP Scan

nmap tcp scan in a nutshell : Other scans ...

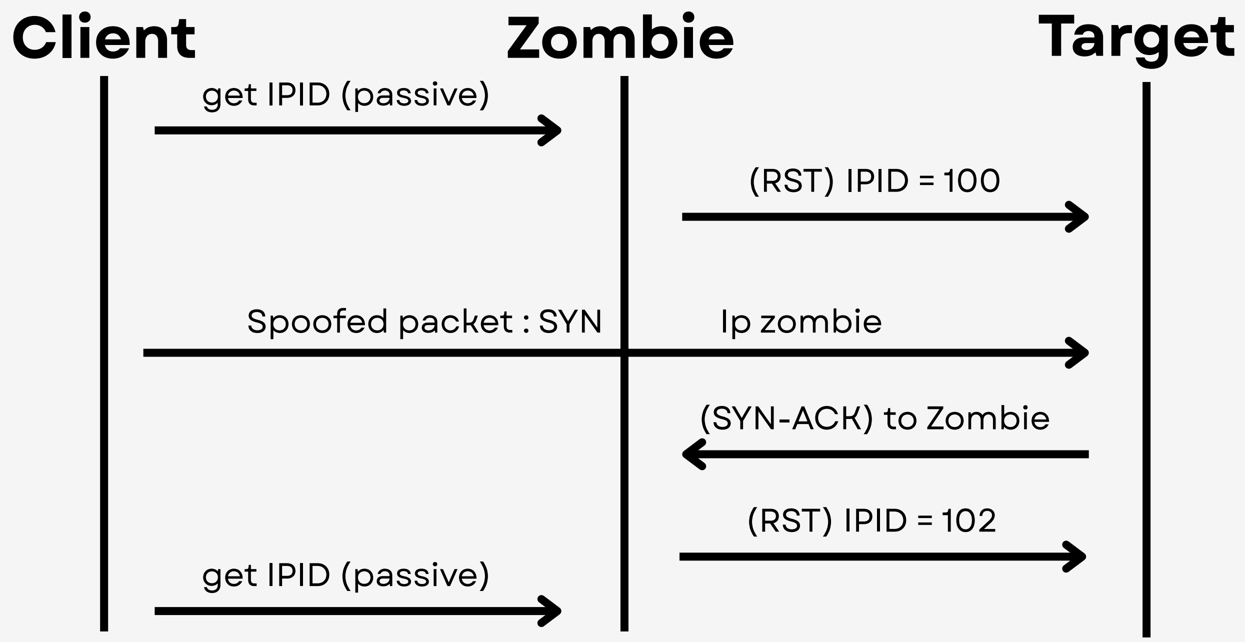
Window scan

Window scan will send an ACK and check the Window field number. if it is 0 → closed, if it is a non-zero value → open

Null - FIN - Xmas scan

No TCP flag will be sent (SYN, FIN, whatever ...), no response → open, Response (RST) → closed ... Stealth but deprecated because modern systems will reply RST. Same for the FIN scan, but we send a FIN flag. XMAS scan will send FIN + URG + PSH flags in the same packet.

Zombie scan



Client

Zombie

Target

get IPID (passive)

(RST) IPID = 100

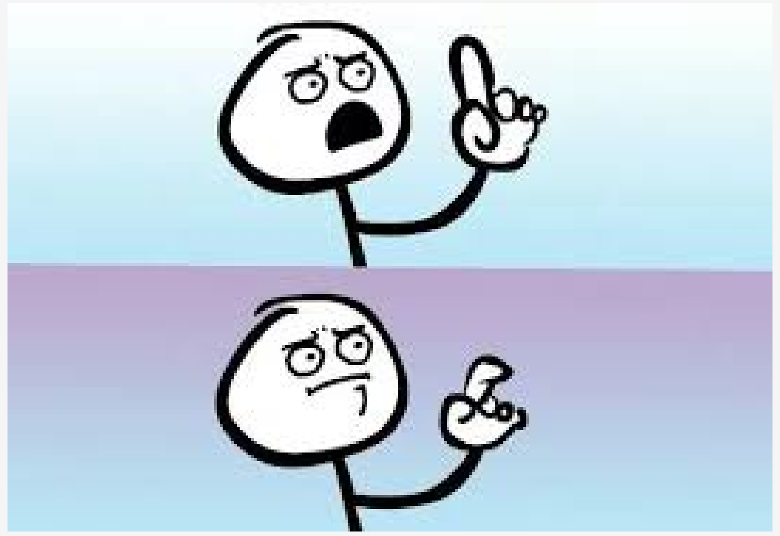
Spoofed packet : SYN

Ip zombie

(SYN-ACK) to Zombie

(RST) IPID = 102

get IPID (passive)



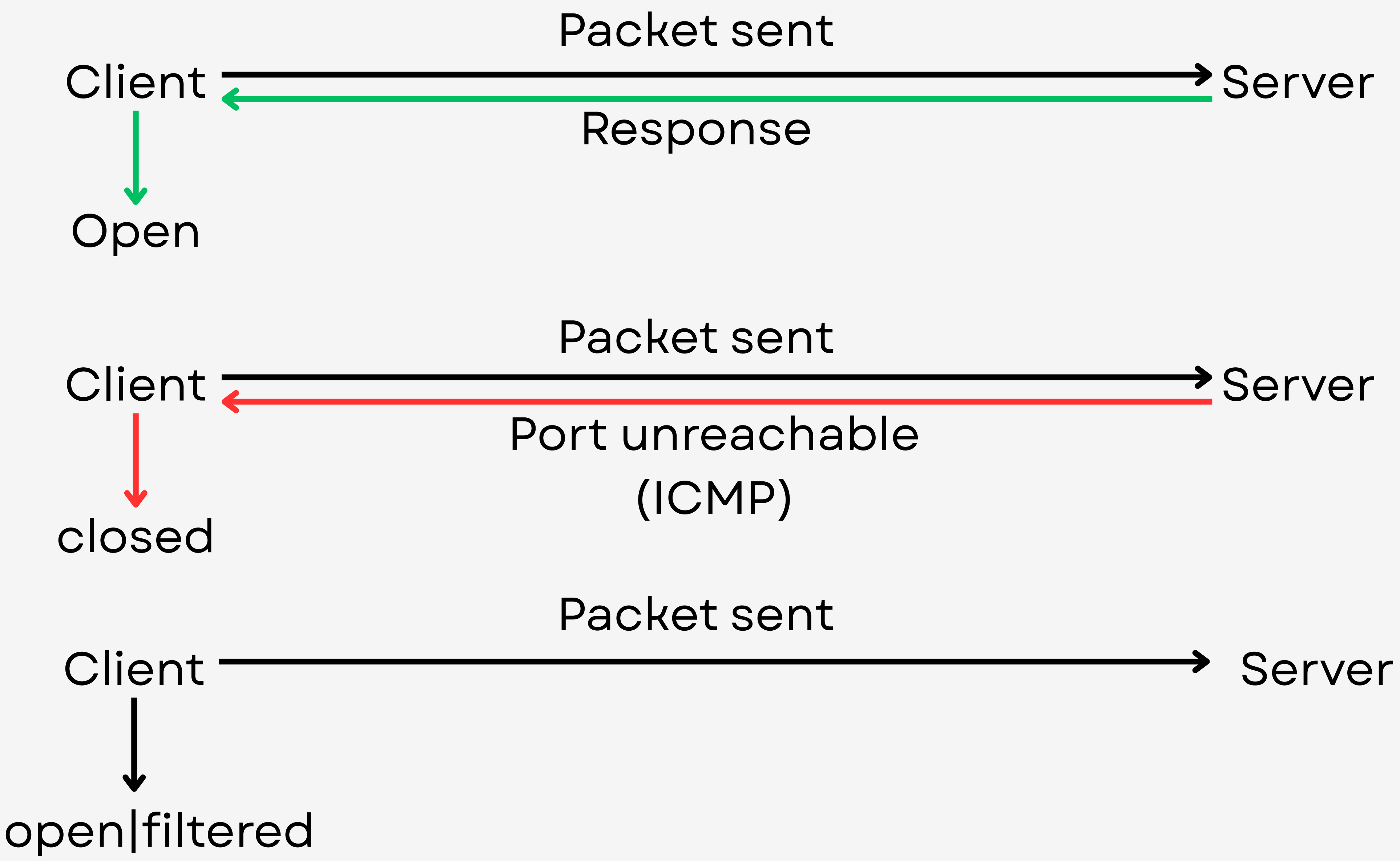
We perform a difference between the first measure and the last measure, in order to see if the target responded to the zombie. This scan has a lot of limitations since IDS can checks for spoofed packets, IPID randomized, if the zombie is not idle and sent a lot of packets ...

UDP Scan

ZzZzZzZ

UDP scan

Nmap will send several UDP packet with different payloads (some are adapted to the application that usually runs on the port).



Packet sent

Client → Server

Response

Open

Packet sent

Client → Server

Port unreachable (ICMP)

closed

Packet sent

Client → Server

open|filtered

Version scan

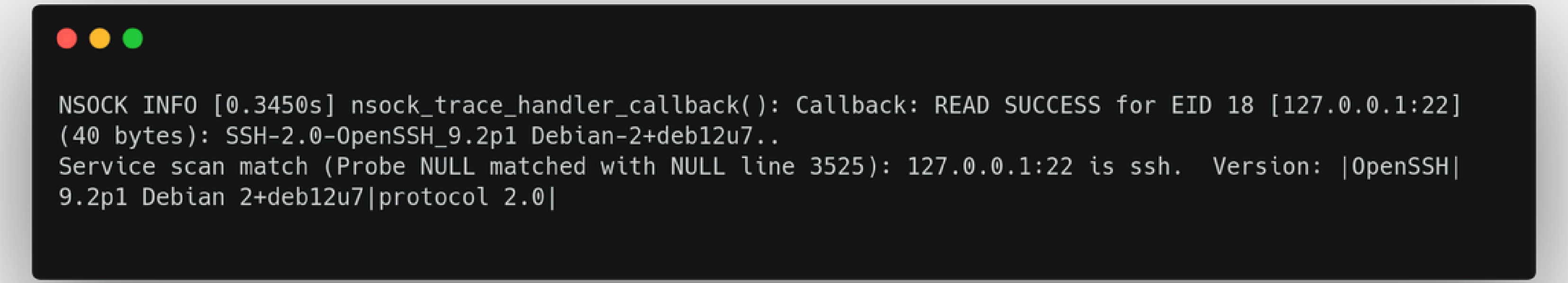
Fingerprinting as much as possible

- For HTTP, sending HTTP request and try to read the server version in header, in 404 response ...
- For SSH, the version is in the TCP banner etc.

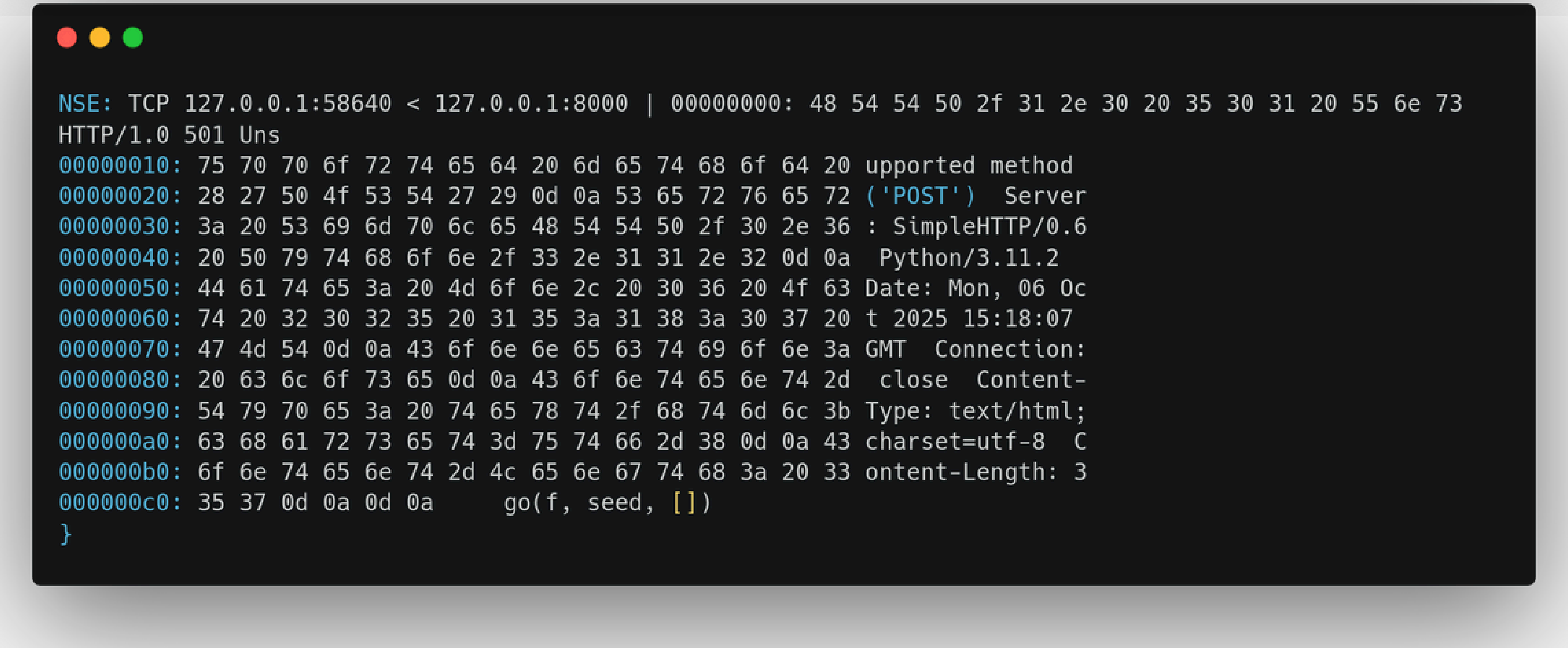
For the OS scan, nmap will focus his guess on a lot of parameters :

- TCP option values (window size, MSS, timestamps, window scale)
- behavior in response to abnormal packets
- ICMP responses and their fields.

→ Signature of each OS stored in /usr/share/nmap/nmap-os-db

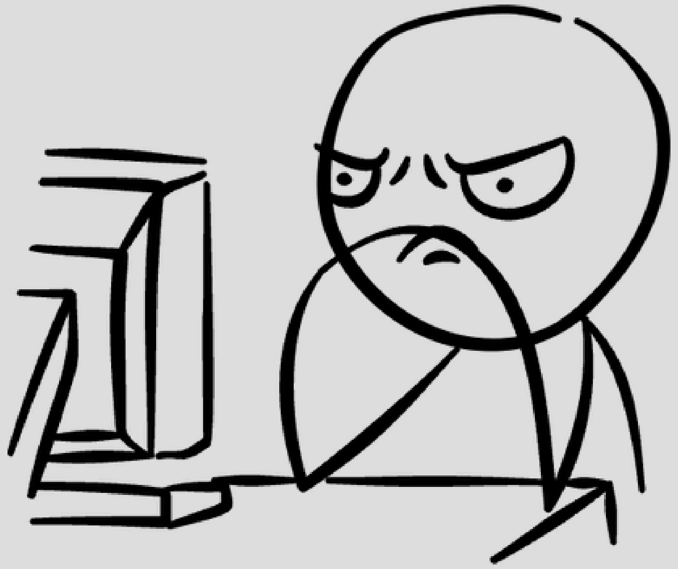


```
NSOCK INFO [0.3450s] nsock_trace_handler_callback(): Callback: READ SUCCESS for EID 18 [127.0.0.1:22]
(40 bytes): SSH-2.0-OpenSSH_9.2p1 Debian-2+deb12u7..
Service scan match (Probe NULL matched with NULL line 3525): 127.0.0.1:22 is ssh.  Version: |OpenSSH|
9.2p1 Debian 2+deb12u7|protocol 2.0|
```



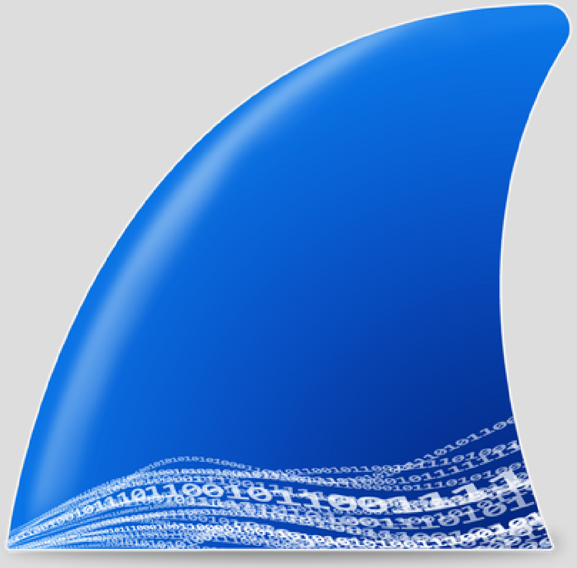
```
NSE: TCP 127.0.0.1:58640 < 127.0.0.1:8000 | 00000000: 48 54 54 50 2f 31 2e 30 20 35 30 31 20 55 6e 73
HTTP/1.0 501 Uns
00000010: 75 70 70 6f 72 74 65 64 20 6d 65 74 68 6f 64 20 upported method
00000020: 28 27 50 4f 53 54 27 29 0d 0a 53 65 72 76 65 72 ('POST')  Server
00000030: 3a 20 53 69 6d 70 6c 65 48 54 54 50 2f 30 2e 36 : SimpleHTTP/0.6
00000040: 20 50 79 74 68 6f 6e 2f 33 2e 31 31 2e 32 0d 0a  Python/3.11.2
00000050: 44 61 74 65 3a 20 4d 6f 6e 2c 20 30 36 20 4f 63 Date: Mon, 06 Oc
00000060: 74 20 32 30 32 35 20 31 35 3a 31 38 3a 30 37 20 t 2025 15:18:07
00000070: 47 4d 54 0d 0a 43 6f 6e 6e 65 63 74 69 6f 6e 3a GMT  Connection:
00000080: 20 63 6c 6f 73 65 0d 0a 43 6f 6e 74 65 6e 74 2d  close  Content-
00000090: 54 79 70 65 3a 20 74 65 78 74 2f 68 74 6d 6c 3b Type: text/html;
000000a0: 63 68 61 72 73 65 74 3d 75 74 66 2d 38 0d 0a 43 charset=utf-8  C
000000b0: 6f 6e 74 65 6e 74 2d 4c 65 6e 67 74 68 3a 20 33 ontent-Length: 3
000000c0: 35 37 0d 0a 0d 0a     go(f, seed, [])
}
```

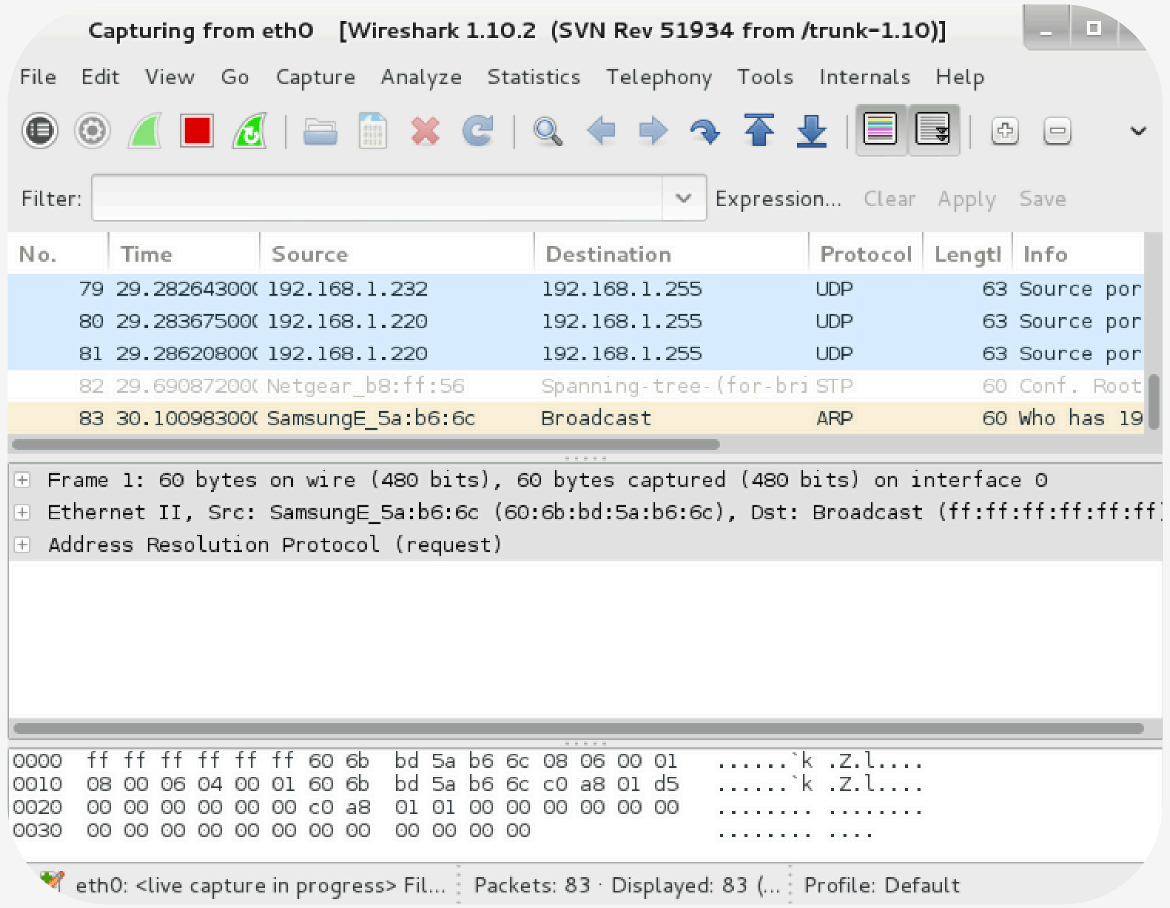
Trouble shouting with nmap



- --packet-trace : will show all packets send and received and their content

- Wireshark





Capturing from eth0 [Wireshark 1.10.2 (SVN Rev 51934 from /trunk-1.10)]

No.	Time	Source	Destination	Protocol	Lengtl	Info
79	29.28264300(	192.168.1.232	192.168.1.255	UDP	63	Source por
80	29.28367500(	192.168.1.220	192.168.1.255	UDP	63	Source por
81	29.28620800(	192.168.1.220	192.168.1.255	UDP	63	Source por
82	29.69087200(	Netgear_b8:ff:56	Spanning-tree-(for-bri	STP	60	Conf. Root
83	30.10098300(	SamsungE_5a:b6:6c	Broadcast	ARP	60	Who has 19

```
⊞ Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
⊞ Ethernet II, Src: SamsungE_5a:b6:6c (60:6b:bd:5a:b6:6c), Dst: Broadcast (ff:ff:ff:ff:ff:ff
⊞ Address Resolution Protocol (request)
```

```
0000  ff ff ff ff ff ff 60 6b  bd 5a b6 6c 08 06 00 01   ......`k .Z.l....
0010  08 00 06 04 00 01 60 6b  bd 5a b6 6c c0 a8 01 d5   ......`k .Z.l....
0020  00 00 00 00 00 00 c0 a8  01 01 00 00 00 00 00 00   ........ ........
0030  00 00 00 00 00 00 00 00  00 00 00 00              ........ ....
```

eth0: <live capture in progress> Fil... Packets: 83 · Displayed: 83 (... Profile: Default

Firewall

IDS, IPS, Firewall ...

- **Firewalls**

A firewall monitors network traffic and enforces rules to allow, ignore, or block connections, preventing unauthorized or potentially dangerous access.

- **IDS/IPS**

An IDS detects and reports potential attacks by analyzing network traffic for known patterns, while an IPS takes active measures to block or prevent detected threats.

Defender

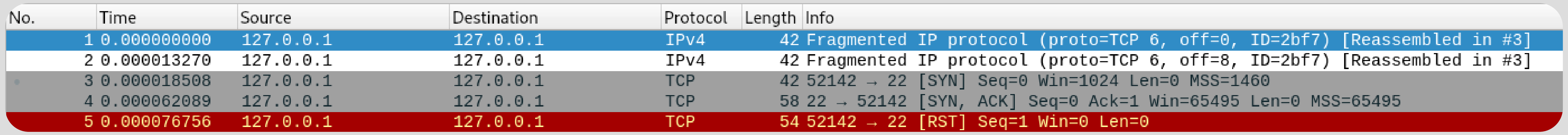
: Wall of Fire gets +1/+0 until end of turn.

Conjured from the bowels of hell, the fiery wall forms an impassable barrier, searing the soul of any creature attempting to pass through its terrible bursts of flame.

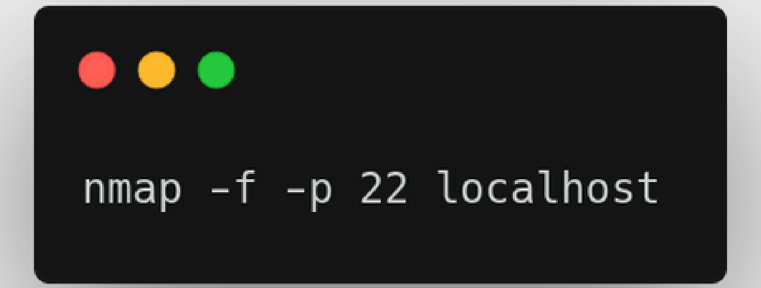
Bypassing firewall ?

IDS, IPS, Firewall ...
Fragmented packets

- ACK scan : used to detect firewall rules
- IP spoofing
- MAC Address spoofing
- Packet fragmentation
- Decoys : Nmap will send multiple packets with different (and existing) IP addresses and insert attacker's IP address into this set
- Using other source port (like 53 - DNS port)



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	IPv4	42	Fragmented IP protocol (proto=TCP 6, off=0, ID=2bf7) [Reassembled in #3]
2	0.000013270	127.0.0.1	127.0.0.1	IPv4	42	Fragmented IP protocol (proto=TCP 6, off=8, ID=2bf7) [Reassembled in #3]
3	0.000018508	127.0.0.1	127.0.0.1	TCP	42	52142 → 22 [SYN] Seq=0 Win=1024 Len=0 MSS=1460
4	0.000062089	127.0.0.1	127.0.0.1	TCP	58	22 → 52142 [SYN, ACK] Seq=0 Ack=1 Win=65495 Len=0 MSS=65495
5	0.000076756	127.0.0.1	127.0.0.1	TCP	54	52142 → 22 [RST] Seq=1 Win=0 Len=0

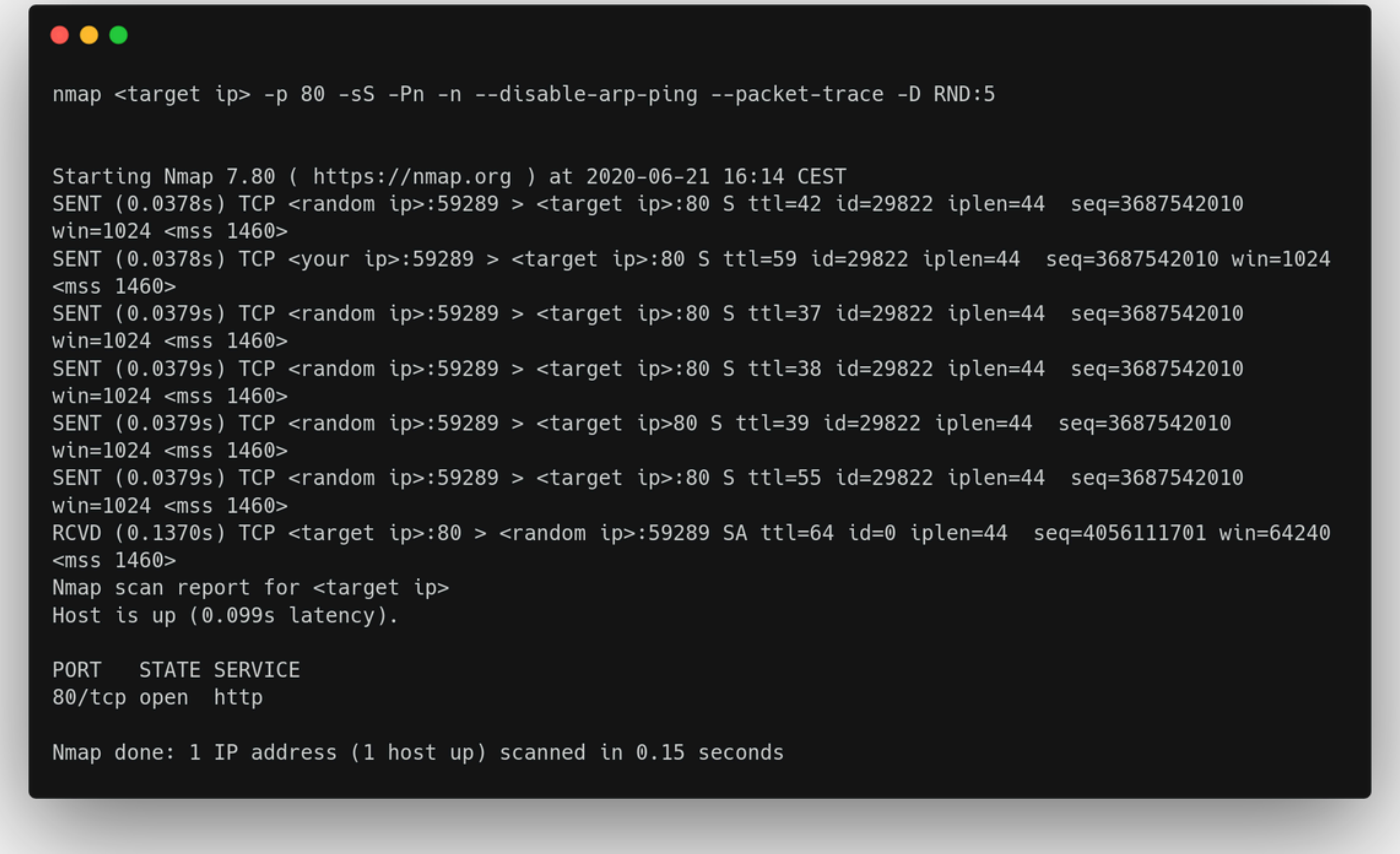


```
nmap -f -p 22 localhost
```

Bypassing firewall ?

IDS, IPS, Firewall ... Decoys

- ACK scan : used to detect firewall rules
- IP spoofing
- MAC Address spoofing
- Packet fragmentation
- Decoys : Nmap will send multiple packets with different (and existing) IP addresses and insert attacker's IP address into this set
- Using other source port (like 53 - DNS port)



```
nmap <target ip> -p 80 -sS -Pn -n --disable-arp-ping --packet-trace -D RND:5


Starting Nmap 7.80 ( https://nmap.org ) at 2020-06-21 16:14 CEST
SENT (0.0378s) TCP <random ip>:59289 > <target ip>:80 S ttl=42 id=29822 iplen=44  seq=3687542010
win=1024 <mss 1460>
SENT (0.0378s) TCP <your ip>:59289 > <target ip>:80 S ttl=59 id=29822 iplen=44  seq=3687542010 win=1024
<mss 1460>
SENT (0.0379s) TCP <random ip>:59289 > <target ip>:80 S ttl=37 id=29822 iplen=44  seq=3687542010
win=1024 <mss 1460>
SENT (0.0379s) TCP <random ip>:59289 > <target ip>:80 S ttl=38 id=29822 iplen=44  seq=3687542010
win=1024 <mss 1460>
SENT (0.0379s) TCP <random ip>:59289 > <target ip>80 S ttl=39 id=29822 iplen=44  seq=3687542010
win=1024 <mss 1460>
SENT (0.0379s) TCP <random ip>:59289 > <target ip>:80 S ttl=55 id=29822 iplen=44  seq=3687542010
win=1024 <mss 1460>
RCVD (0.1370s) TCP <target ip>:80 > <random ip>:59289 SA ttl=64 id=0 iplen=44  seq=4056111701 win=64240
<mss 1460>
Nmap scan report for <target ip>
Host is up (0.099s latency).

PORT   STATE SERVICE
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 0.15 seconds
```

01

What’s nmap

Give a short introduction about nmap and its basic uses

02

Under the hood

Understanding advanced nmap options, and what is really happenning behind the CLI

03

Scripting

Experiment custom NSE script

NSE introduction

Nmap script engine

Nmap Script Engine

The Nmap Scripting Engine (NSE) is an extensible framework that allows Nmap to run scripts to automate a wide range of network tasks, from service discovery to vulnerability detection.
Because scripts can perform intrusive actions, they range from non-destructive probes to potentially disruptive tests, so they should only be run against authorized targets. NSE greatly expands Nmap’s flexibility by enabling custom checks, automation, and integration with external data sources.

- Script in Lua
- Different categories (vuln, safe, discovery ...)
- Can be very intrusive and be responsible of crash or denial of service

(so be careful using it on authorized targets)

NSE introduction

Nmap script engine

Exemple of well known (and useful) scripts

- vuln : script group that will test CVEs related tests on the target in order to determine wether the target is vulnerable or not
- http-brute : script that will bruteforce http authentication (basic, digest and NTML auth are handled), very basic creds
- FTP anon : will test if you can connect to a FTP server as anonymous
- SMB OS discovery : will guess the operating system of the SMB server
- Other protocol specific scripts ...

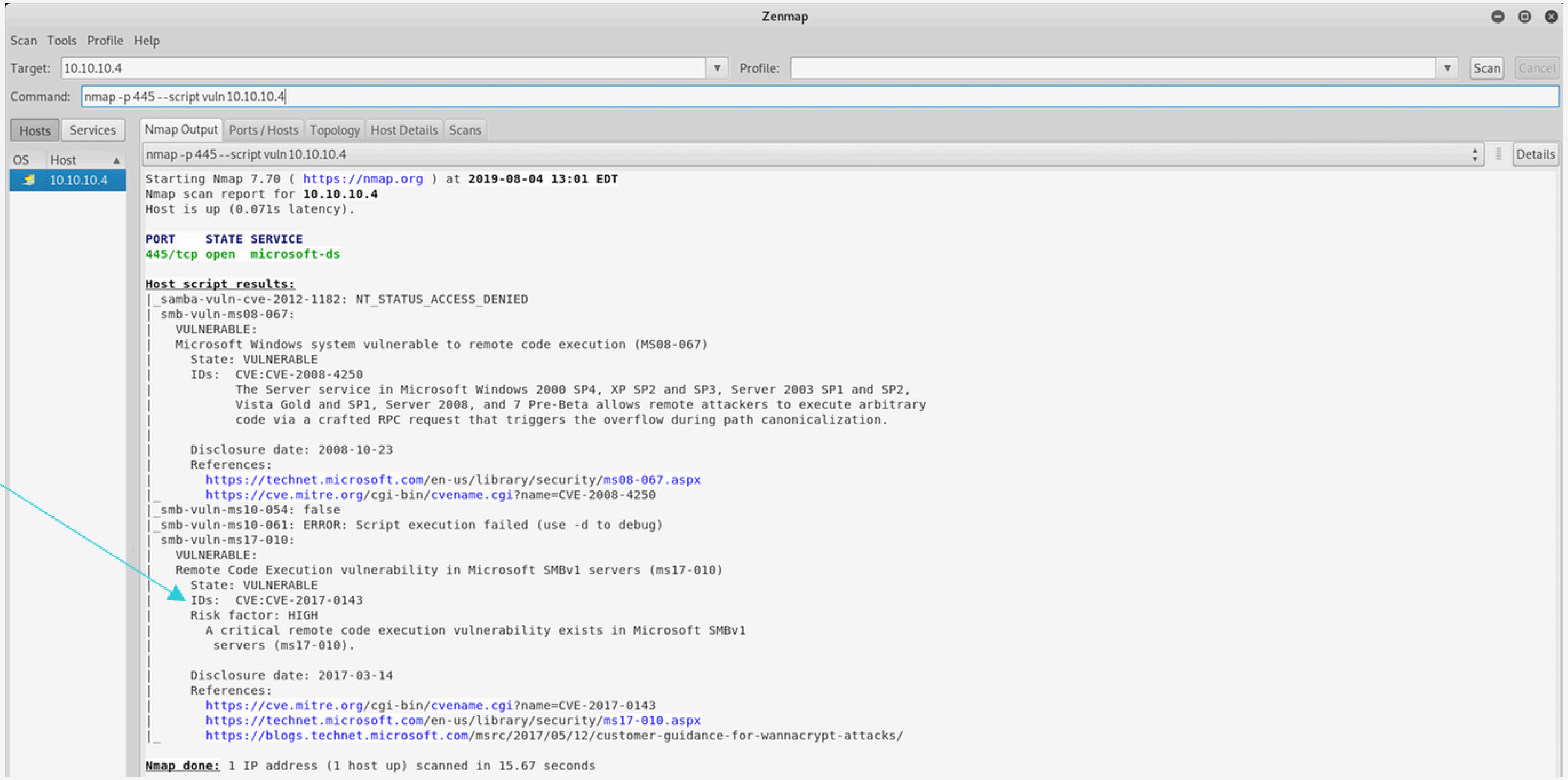
EternalBlue

NSE introduction

Nmap script engine

Example :

Eternal Blue detection



Target: 10.10.10.4

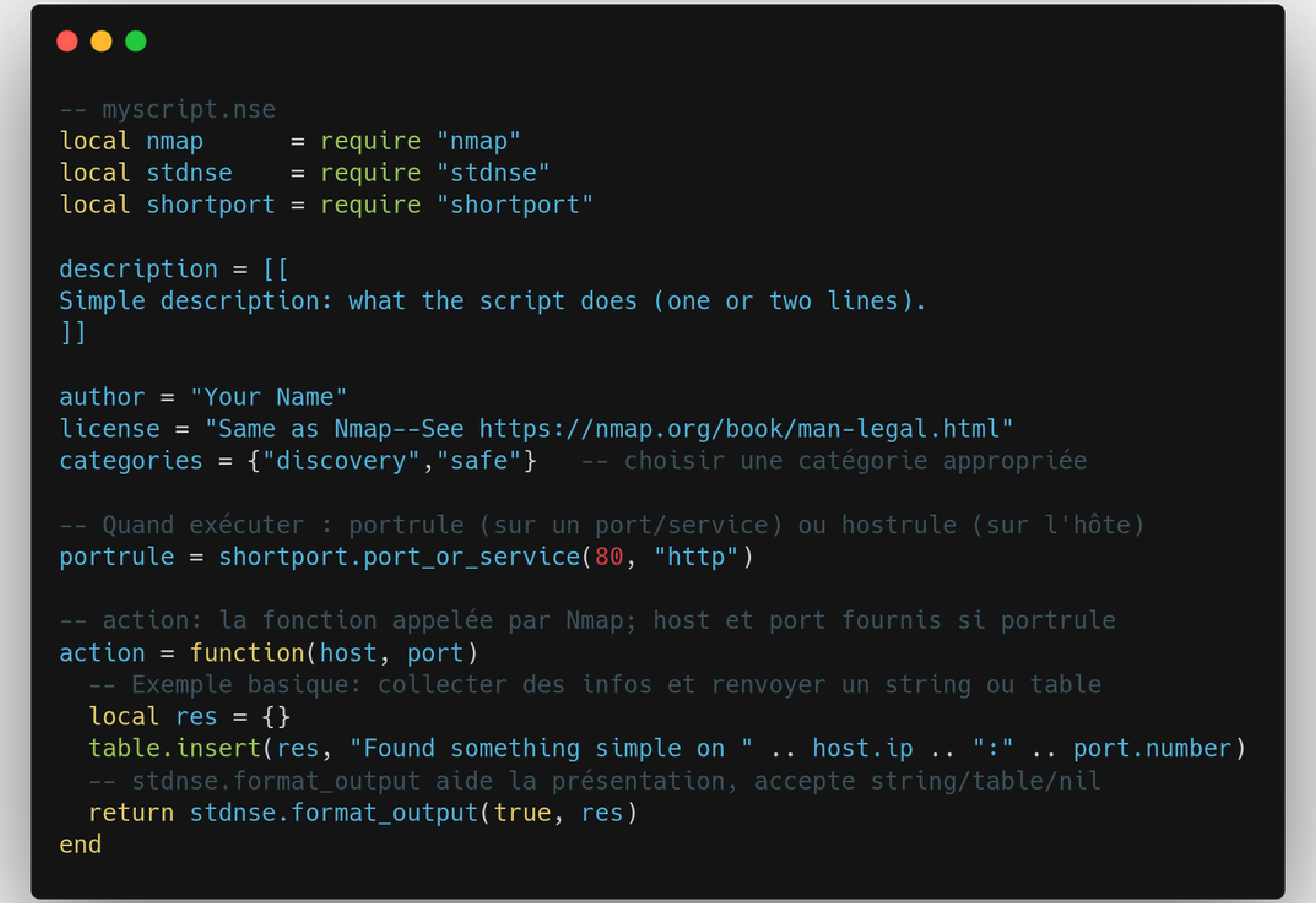
Command: nmap -p 445 --script vuln 10.10.10.4

10.10.10.4

```
nmap -p 445 --script vuln 10.10.10.4
Starting Nmap 7.70 ( https://nmap.org ) at 2019-08-04 13:01 EDT
Nmap scan report for 10.10.10.4
Host is up (0.071s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
|_samba-vuln-cve-2012-1182: NT_STATUS_ACCESS_DENIED
| smb-vuln-ms08-067:
|   VULNERABLE:
|   Microsoft Windows system vulnerable to remote code execution (MS08-067)
|     State: VULNERABLE
|     IDs:  CVE:CVE-2008-4250
|           The Server service in Microsoft Windows 2000 SP4, XP SP2 and SP3, Server 2003 SP1 and SP2,
|           Vista Gold and SP1, Server 2008, and 7 Pre-Beta allows remote attackers to execute arbitrary
|           code via a crafted RPC request that triggers the overflow during path canonicalization.
|
|     Disclosure date: 2008-10-23
|     References:
|       https://technet.microsoft.com/en-us/library/security/ms08-067.aspx
|_      https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-4250
|_smb-vuln-ms10-054: false
|_smb-vuln-ms10-061: ERROR: Script execution failed (use -d to debug)
| smb-vuln-ms17-010:
|   VULNERABLE:
|   Remote Code Execution vulnerability in Microsoft SMBv1 servers (ms17-010)
|     State: VULNERABLE
|     IDs:  CVE:CVE-2017-0143
|     Risk factor: HIGH
|       A critical remote code execution vulnerability exists in Microsoft SMBv1
|        servers (ms17-010).
|
|     Disclosure date: 2017-03-14
|     References:
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-0143
|       https://technet.microsoft.com/en-us/library/security/ms17-010.aspx
|_      https://blogs.technet.microsoft.com/msrc/2017/05/12/customer-guidance-for-wannacrypt-attacks/

Nmap done: 1 IP address (1 host up) scanned in 15.67 seconds
```

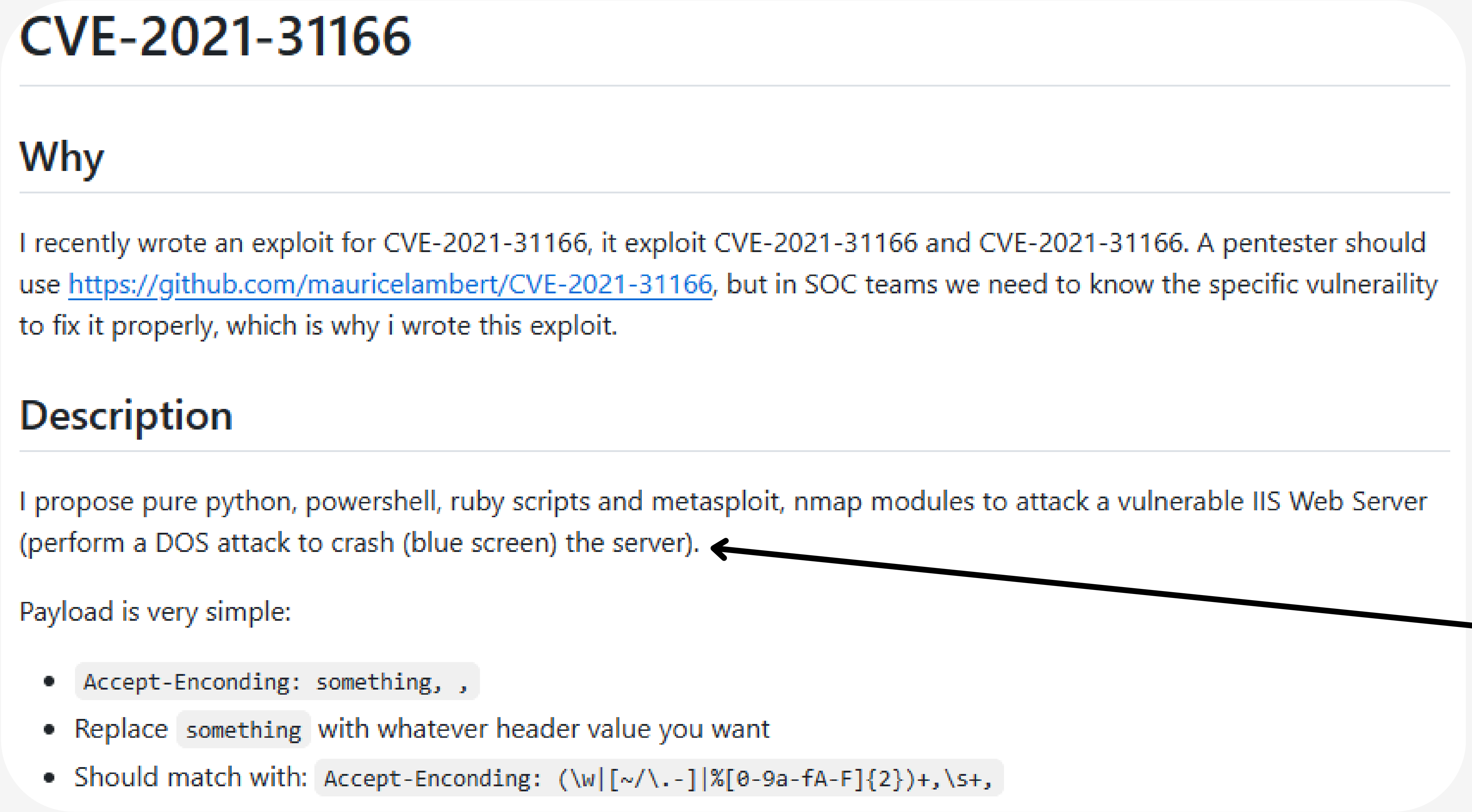
Basic NSE script

Lua lua land



```lua
-- myscript.nse
local nmap      = require "nmap"
local stdnse    = require "stdnse"
local shortport = require "shortport"

description = [[
Simple description: what the script does (one or two lines).
]]

author = "Your Name"
license = "Same as Nmap--See https://nmap.org/book/man-legal.html"
categories = {"discovery","safe"}   -- choisir une catégorie appropriée

-- Quand exécuter : portrule (sur un port/service) ou hostrule (sur l'hôte)
portrule = shortport.port_or_service(80, "http")

-- action: la fonction appelée par Nmap; host et port fournis si portrule
action = function(host, port)
  -- Exemple basique: collecter des infos et renvoyer un string ou table
  local res = {}
  table.insert(res, "Found something simple on " .. host.ip .. ":" .. port.number)
  -- stdnse.format_output aide la présentation, accepte string/table/nil
  return stdnse.format_output(true, res)
end
```

Basic NSE script



CVE-2021-31166

Why

I recently wrote an exploit for CVE-2021-31166, it exploit CVE-2021-31166 and CVE-2021-31166. A pentester should use https://github.com/mauricelambert/CVE-2021-31166, but in SOC teams we need to know the specific vulneraility to fix it properly, which is why i wrote this exploit.

Description

I propose pure python, powershell, ruby scripts and metasploit, nmap modules to attack a vulnerable IIS Web Server (perform a DOS attack to crash (blue screen) the server).

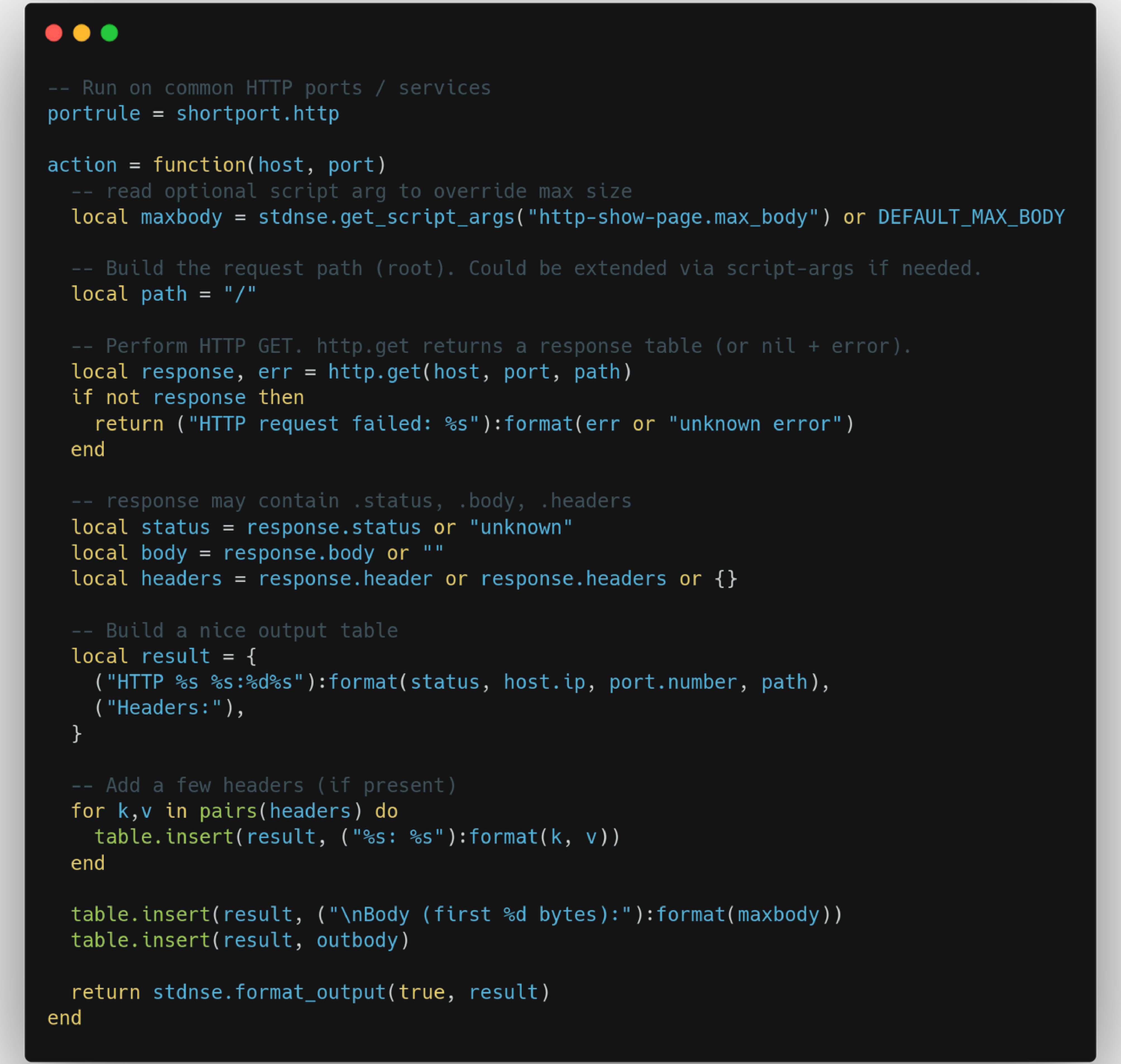
Payload is very simple:

- `Accept-Enconding: something, ,`
- Replace `something` with whatever header value you want
- Should match with: `Accept-Enconding: (\w|[~/\.-]|%[0-9a-fA-F]{2})+,\s+,`

Vulnerability that allows an attacker to perform a buffer overflow over the "Accept-Encoding" HTTP field. Sending specific payloads lead to a crash of the service and hence, a BSOD :(

Some remarks

- Do not forget to include HTTP library
- Output are sent to the stdnse with table



```lua
-- Run on common HTTP ports / services
portrule = shortport.http

action = function(host, port)
  -- read optional script arg to override max size
  local maxbody = stdnse.get_script_args("http-show-page.max_body") or DEFAULT_MAX_BODY

  -- Build the request path (root). Could be extended via script-args if needed.
  local path = "/"

  -- Perform HTTP GET. http.get returns a response table (or nil + error).
  local response, err = http.get(host, port, path)
  if not response then
    return ("HTTP request failed: %s"):format(err or "unknown error")
  end

  -- response may contain .status, .body, .headers
  local status = response.status or "unknown"
  local body = response.body or ""
  local headers = response.header or response.headers or {}

  -- Build a nice output table
  local result = {
    ("HTTP %s %s:%d%s"):format(status, host.ip, port.number, path),
    ("Headers:"),
  }

  -- Add a few headers (if present)
  for k,v in pairs(headers) do
    table.insert(result, ("%s: %s"):format(k, v))
  end

  table.insert(result, ("\nBody (first %d bytes):"):format(maxbody))
  table.insert(result, outbody)

  return stdnse.format_output(true, result)
end
```

Your turn – DEMO

You have to use your own script to get the flag on the following network protocol

Goal : *when a client connect to the server, the server will choose one index from 1 to 10. The client can request the target asking what is the value of an index. The client can do it as much as possible.*

You need to automate that !

The protocol is quite simple, when you connect to the server in TCP, you can perform the following functions :

- "**INFO**" : will give you informations on the protocole (same as what I am explaining right now)
- "**DISCOVER**" : will give you a session UUID
- "**QUERY <UUID> <ID>**" : will return you 1 or 0, the ID will go from 1 to 10, the UUID is the UUID session previously obtained
- "**VERIFY <UUID> <ID>**" : will give you the flag if the ID submitted is the ID set to 1

04

Bonus – Legal consideration

To scan or not to scan ?

⚠️ Disclaimer
I'm not a legal expert, and this section is based on my understanding of publicly available information. It's possible that I may have misunderstood or oversimplified certain legal aspects. Please consult a qualified professional or legal advisor for authoritative guidance.

Laws – rules

From nmap’s documentation

What can we do ?

“When used properly, Nmap helps protect your network from invaders.
But when used improperly, Nmap can (in rare cases) get you sued, fired, expelled, jailed, or banned by your ISP.”

nmap.org

Laws – rules

From nmap's documentation

Law ? Depends where ...

US : Port scanning as such is not criminalized under federal law.
The Computer Fraud and Abuse Act criminalizes unauthorized access to a protected computer. Since port scanning is not considered an intrusion, it is *not penalized per se*.
The determining factors are intent and subsequent actions (intrusion, denial of service, etc.).

France : There is no specific law targeting port scanning ... BUT, scanning can be reinterpreted as an attempt at unauthorized access or a malicious reconnaissance act, under the Criminal Code (e.g., Articles 323-1 and following: fraudulent access, interference or alteration).
In particular, using scanning to gather information about possible vulnerabilities or to enable unauthorized access gives rise to a presumption of malicious intent.

Laws – rules

From nmap's documentation

Solutions ?

- **Obtain explicit written authorization covering the scope, purpose, etc.**
- Perform minimal scanning (e.g., targeting only one port if the goal is to debug a single service).
- Use non-intrusive scan techniques (e.g., SYN scan, polite/stealth mode), and avoid using --script vuln.
- Do not attempt obfuscation techniques.
- Always justify the scan with professional or technical objectives.

References

- OSI model explanation : https://www.cloudflare.com/fr-fr/learning/ddos/glossary/open-systems-interconnection-model-osi
- https://www.freecodecamp.org/news/keep-calm-and-hack-the-box-legacy/
- https://nmap.org
- https://www.youtube.com/watch?v=nX9JXI4l3-E (conf DEF CON 22 Mass scanning the internet)
- https://github.com/mauricelambert/CVE-2021-31166
- https://svn.nmap.org/nmap/scripts/smb-vuln-ms17-010.nse

Thank You

FOR YOUR TIME

https://sidorocs.github.io