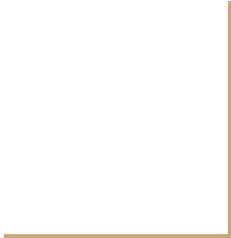


# COMP 4670

## Tutorials

### Network Attacks Part 3

Dr. Sherif Saad



# Agenda

- Network Mapping and Scanning.
- Deep Packet Inspections.
- Network Vulnerability Assessment.

# Network Mapping and Scanning

# NMAP Scanning Options

**NMAP** is a network mapping and scanning tool that could detect **live hosts, open ports, network services, OS** fingerprint and detect **network topology**.

There are several scanning options in NMAP:

1. Intense scan
2. Intense scan plus UDP
3. Intense scan plus all TCP ports
4. Intense scan, no ping
5. Ping scan
6. Quick scan plus
7. Quick traceroute
8. Regular scan
9. Slow Comprehensive

# Options in Nutshell

## Intense Scan

- **Command:** `nmap -T4 -A -v <target>`
- This scan has a reasonable run time and it scans known TCP ports only. It attempts to detect the OS type and the version of the running services.

## Intense Scan plus UDP

- **Command:** `nmap -sS -sU -T4 -A -v <target>`
- Perform regular intense scan plus scanning UDP ports (-sU) and to use SYN packets for TCP ports (-sS)

# Options in Nutshell

## Intense Scan, all TCP ports

- **Command:** `nmap -p 1-65535 -T4 -A -v <target>`
- This scan is a regular TCP intense scan but it scan all the possible ports (1-65535)

## Intense Scan, no ping

- **Command:** `nmap -T4 -A -v -Pn <target>`
- Assume that the target is up and that the target blocks ping request (e.g. firewall or packet filtering)

# Options in Nutshell

## Ping Scan

- **Command:** `nmap -sn <target>`
- Do only a ping scan (ICMP echo request) there is no port scan.

## Quick Scan

- **Command:** `nmap -T4 -F <target>`
- Limit your scan to the top most common TCP ports (e.g. 22, 25, 21, 80, etc). It does not attempt to detect OS

# Options in Nutshell

## Quick Scan Plus

- **Command:** `nmap -sV -T4 -O -F -version-light <target>`
- It is a quick scan but will try to detect OS version and services type.

## Quick traceroute

- **Command:** `nmap -sn -traceroute <target>`
- Use this scan to determine hosts and routers in the target network.

# Options in Nutshell

## Regular Scan

- **Command:** `nmap <target>`
- TCP SYN scan for the 1000 common TCP ports and Ping to detect live host.

# Options in Nutshell

## Slow Comprehensive Scan

- **Command:** `nmap -sS -sU -T4 -A -v -PE -PP -PS80,443 -PA3389 -PU40125 -PY -g 53 -script “default or (discovery and safe)”<target>`
- This is scan s mainly for host detection assuming great effort has been done to hide live hosts. It is an intense scan plus UDP and other options. It tries to use TCP, UDP and SCTP protocols to detect live hosts.
- If the host is live it will try its best to find the host OS and running services. It also claim it is a DNS server (src port 53)

# Some NMAP Scanning Options

- -T4:
  - To control how fast is the scan. It ranges from 0-5, where 5 is the fastest and 0 is the slowest. (0: paranoid 1: sneaky 2: polite 3: normal 4: aggressive 5: insane)
  - 1-2 being used for IDS evasion, 3 is the default and 4-5 is really quick scans.
- -A:
  - Instruct NMAP to try to detect the host OS version, traceroute and services version
- -V:
  - Increased verbosity. This will give your extra information in the data outputted by Nmap.
- -sS:
  - TCP SYN connect scan
- -sU:
  - UDP scan

# Some NMAP Scanning Options

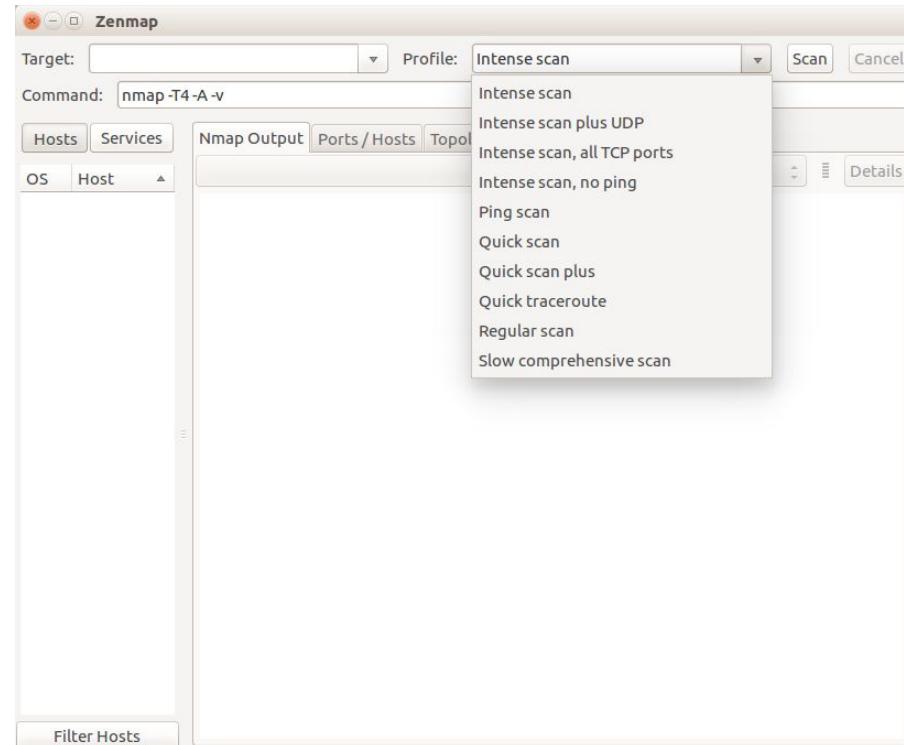
- **-sN:**
  - TCP Null scan, sends TCP packets with none of the TCP flags set in the packet. If the scan is returned a RST packet it means the port is closed, however if nothing is returned it is either filtered or open
- **-sV:**
  - Instruct the scan to try to detect the service version and type by sending additional packets to the open ports.
- **-version-light:**
  - Less intensive service probing. It will try to probe the service using best guesses
- **-p:**
  - To define a set of port to scan and probe
- **-g:**
  - Define what source port you want NMap to use (try to evade IDS and firewall)

# NMAP in actions

You should use the GUI version of Nmap.

The UI allows specifying the target.

Select the scan type and edit the scan command if required.



# NMAP Scan Example

We will scan the subnet **192.168.56.0/24** using an intense scan



# NMAP Scan Example

When NMAP complete the scan. We will get a list of live host on the target subnet

The screenshot shows the Nmap interface with the following details:

- Hosts:** OS Host
- Services:** None
- Scan Command:** nmap -T4 -A -v 192.168.56.0/24
- Scan Output:**
  - WARNING:** OS didn't match until try #2
  - NSE:** Script scanning 192.168.56.101.
  - Initiating NSE at 19:18
  - Completed NSE at 19:18, 0.00s elapsed
  - Initiating NSE at 19:18
  - Completed NSE at 19:18, 0.00s elapsed
  - Nmap scan report for 192.168.56.101
  - Host is up (0.000027s latency).
  - All 1000 scanned ports on 192.168.56.101 are closed
  - Warning:** OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
  - device\_type:** general purpose
  - Running:** Linux 2.4.X|2.6.X
  - OS\_CPE:** cpe:/o:linux:linux\_kernel:2.4.20 cpe:/o:linux:linux\_kernel:2.6
  - OS\_details:** Linux 2.4.20, Linux 2.6.14 - 2.6.34, Linux 2.6.17 (Mandriva), Linux 2.6.23, Linux 2.6.24
  - Network Distance:** 0 hops
  - NSE:** Script Post-scanning.
  - Initiating NSE at 19:18
  - Completed NSE at 19:18, 0.00s elapsed
  - Initiating NSE at 19:18
  - Completed NSE at 19:18, 0.00s elapsed
  - Read data files from:** /usr/bin/../share/nmap
  - OS and Service detection performed. Please report any incorrect results at <https://nmap.org/submit/>.
  - Nmap done:** 256 IP addresses (4 hosts up) scanned in 73.87 seconds
  - Raw packets sent: 7655 (341.456KB) | Rcvd: 6177 (260.364KB)

# NMAP Scan Example

When NMAP complete the scan. We will get a list of running service on every host

The screenshot shows the Nmap interface with a list of hosts on the left and a detailed scan report on the right. The host 192.168.56.102 is selected, highlighted with a blue background.

**Nmap Output:**

```
nmap -T4 -A -v 192.168.56.0/24
Nmap scan report for 192.168.56.102
Host is up (0.00045s latency).
Not shown: 997 closed ports
PORT      STATE SERVICE VERSION
21/tcp    open  ftp      ProFTPD 1.3.3c
22/tcp    open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 d6:01:90:39:2d:8f:46:fb:03:86:73:b3:3c:54:7e:54 (RSA)
|   256 f1:f3:c0:dd:ba:a4:85:f7:13:9a:da:3a:bb:4d:93:04 (ECDSA)
|_  256 12:e2:98:d2:a3:e7:36:4f:be:6b:ce:36:6b:7e:0d:9e (EdDSA)
80/tcp    open  http     Apache httpd 2.4.18 ((Ubuntu))
| http-methods:
|_ Supported Methods: OPTIONS GET HEAD POST
| http-server-header: Apache/2.4.18 (Ubuntu)
| http-title: Site doesn't have a title (text/html).
MAC Address: 08:00:27:14:06:50 (Oracle VirtualBox virtual NIC)
Device type: general purpose
```

# NMAP Scan Example

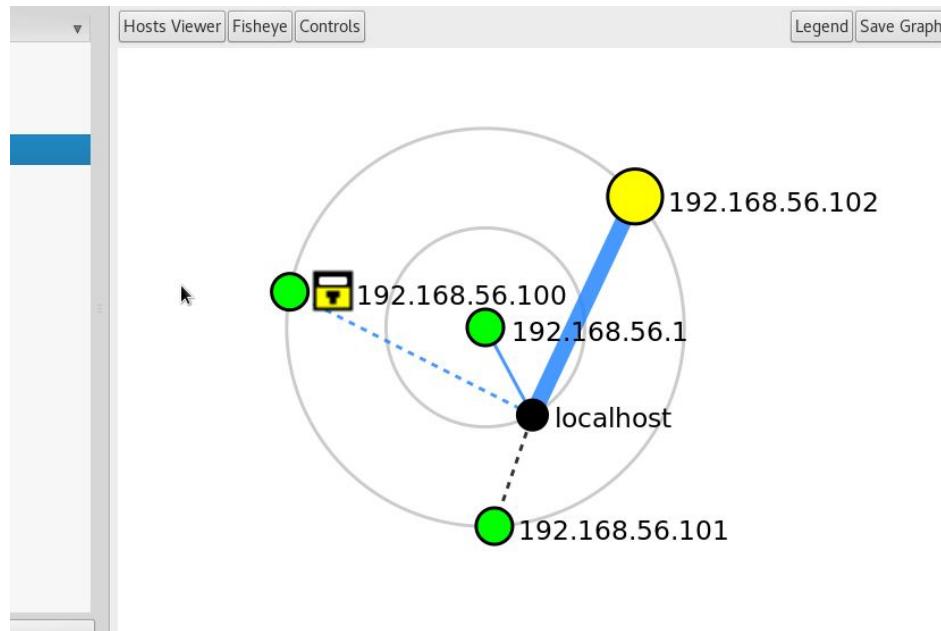
When NMAP complete the scan. We will get a list of running service on every host, in addition the open/filtered/closed ports on every live host

The screenshot shows the Nmap interface with the 'Services' tab selected. On the left, a list of hosts is shown with icons indicating their status: 192.168.56.1 (FTP), 192.168.56.100 (SSH), 192.168.56.101 (HTTP), and 192.168.56.102 (HTTP). The host 192.168.56.102 is currently selected, highlighted with a blue background. The main pane displays the Nmap Output table with the following data:

| Port | Protocol | State | Service | Version  |
|------|----------|-------|---------|--|
| 21   | tcp      | open  | ftp     | ProFTPD 1.3.3c   |
| 22   | tcp      | open  | ssh     | OpenSSH 7.2p2 Ubuntu 4ubuntu2.4 (Ubuntu Linux; protocol 2.0) |
| 80   | tcp      | open  | http    | Apache httpd 2.4.18 ((Ubuntu))                               |

# NMAP Scan Example

When NMAP complete the scan. We will get the network topology (e.g. live host and how many hop they are far from the scanning machine (the machine that run NMAP))



# NMAP Scan Example

In addition if possible we will get OS version for each live node|host in the target network

Closed ports: 997  
Scanned ports: 1000  
Up time: 8846612  
Last boot: Sun Oct 29 10:55:11 2017



▼ Addresses  
IPv4: 192.168.56.102  
IPv6: Not available  
MAC: 08:00:27:14:06:50

▼ Operating System  
Name: Linux 3.2 - 4.8  
Accuracy: 100%

▼ Ports used  
Port-Protocol-State: 21 - tcp - open  
Port-Protocol-State: 1 - tcp - closed  
Port-Protocol-State: 39692 - udp - closed

▼ OS Classes

| Type            | Vendor | OS Family | OS Generation | Accuracy |
|-----------------|--------|-----------|---------------|----------|
| general purpose | Linux  | Linux     | 4.X           | 100%     |

► TCP Sequence

# NMAP Scan Example

We could also check other scanning options like Intense scan all TCP port



# NMAP Scan Example

We could also check other scanning options like Intense scan all TCP port

```
Nmap scan report for 192.168.56.255 [host down]
Initiating SYN Stealth Scan at 19:35
Scanning 3 hosts [65535 ports/host]
Discovered open port 22/tcp on 192.168.56.102
Discovered open port 80/tcp on 192.168.56.102
Increasing send delay for 192.168.56.1 from 0 to 5 due to 14 out of 34 dropped
probes since last increase.
Increasing send delay for 192.168.56.102 from 0 to 5 due to 15 out of 36 dropped
probes since last increase.
Discovered open port 21/tcp on 192.168.56.102
Increasing send delay for 192.168.56.102 from 5 to 10 due to 12 out of 29
dropped probes since last increase.
Increasing send delay for 192.168.56.1 from 5 to 10 due to 48 out of 119 dropped
probes since last increase.
SYN Stealth Scan Timing: About 0.84% done
SYN Stealth Scan Timing: About 1.69% done; ETC: 20:35 (0:59:11 remaining)
SYN Stealth Scan Timing: About 3.64% done; ETC: 20:33 (0:56:06 remaining)
SYN Stealth Scan Timing: About 6.54% done; ETC: 20:31 (0:53:07 remaining)
```

# NMAP Additional Resources

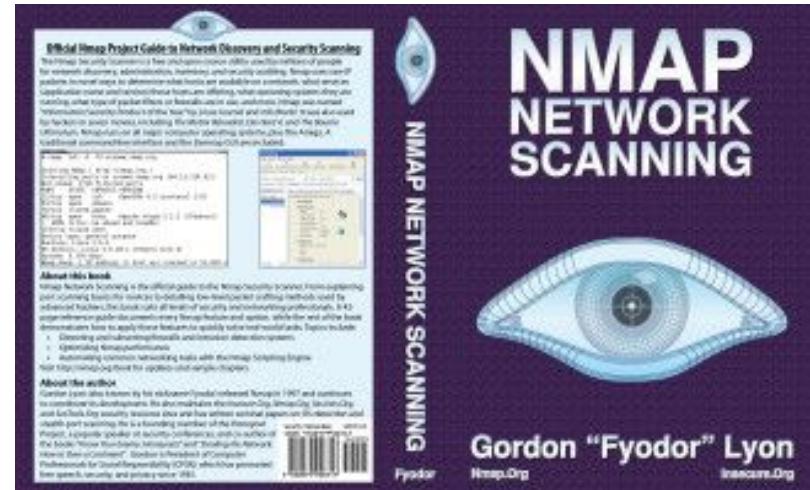
The Official Nmap Project Guide to Network Discovery and Security Scanning

By Gordon “Fyodor” Lyon

Book URL: <http://nmap.org/book/>

ISBN: 978-0-9799587-1-7

ISBN-10: 0-9799587-1-7



# Deep Packet Inspection

# What is Deep Packet Inspection (DPI)?

Deep Packet Inspection is also known as **complete packet inspection**.

A network traffic analysis technique that applies **packet filtering** to inspects the **data part** of the packet in addition to the packet header.

Deep Packet Inspection operates at the **application layer** of the TCP/IP stack to analyze the data in packets to identify protocol noncompliance, intrusions, spam, viruses or other predefined security flags.

Deep Packet Inspection is the combines techniques from network intrusion and prevention systems.

# Why Deep Packet Inspection?

DPI may be used by organizations for **Data Leak Prevention** (DLP). When an employee tries to send an email that contains confidential organization files

Deep Packet Inspection could help in detecting **advanced persistent threat** (APT).

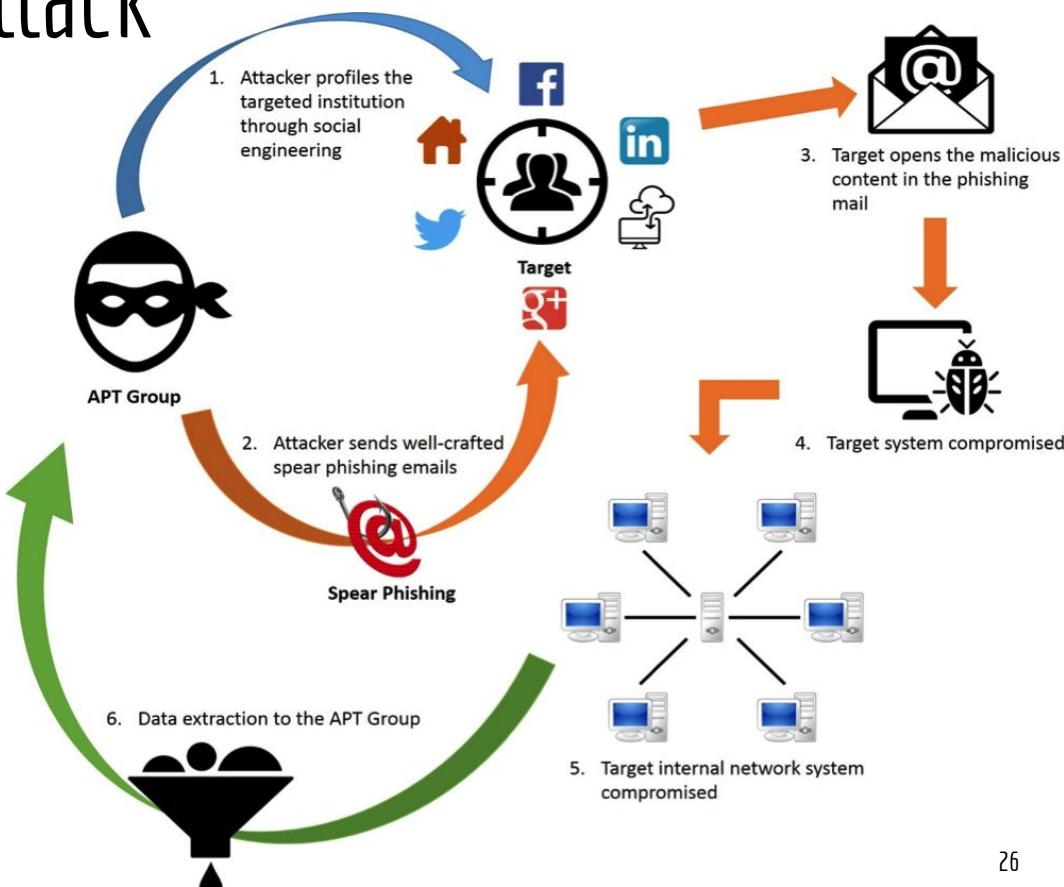
ART or **advanced persistent threat** is a network attack in which an unauthorized person or a group gains access to a network and stays there undetected for a long period of time.

# The Anatomy of APT Attack

The APT attack consist of 6 steps grouped into three main phases:

1. Infiltration
2. Expansion
3. Sabotage

DPI can detect the expansion and the sabotage by looking at the network traffic.



# Deep Packet Inspection with Wireshark

Wireshark is an **open source** tool for analyzing network traffic.

Wireshark enables network traffic **profiling** and **inspection**.

Wireshark is also a **network traffic sniffer** that can capture and record network packets over wired or wireless networks.

The best known packet analyzer and network sniffer that is commonly used by government agencies, enterprises, educational institutions.

Started by Gerald Combs in 1998.



# Why using Wireshark (packet sniffer)?

Provide **network traffic insights** by enabling data in motion recording and monitoring.

Gather and report **network statistics** about network usage

**Debug** and **troubleshooting** network problems and protocol implementation and configuration issue.

Detect **network intrusion** and **misuse** by internal and external users

Help in web filters, spam filters, and provide information to understand and inspect network intrusion and other malicious network activities.

# Wireshark (packet sniffer) Darkside

Could be used to execute [man-in-the-middle](#), and packet injection attacks.

Gather network insights to design and carry out network attacks

[Reverse engineer](#) proprietary protocols used over the network.

[Spy](#) on other network users to collect sensitive information, such as login details or user cookies.

Could be used to [hijack and compromise SSL/TLS](#) communication depending upon encryption being used.

# Wireshark

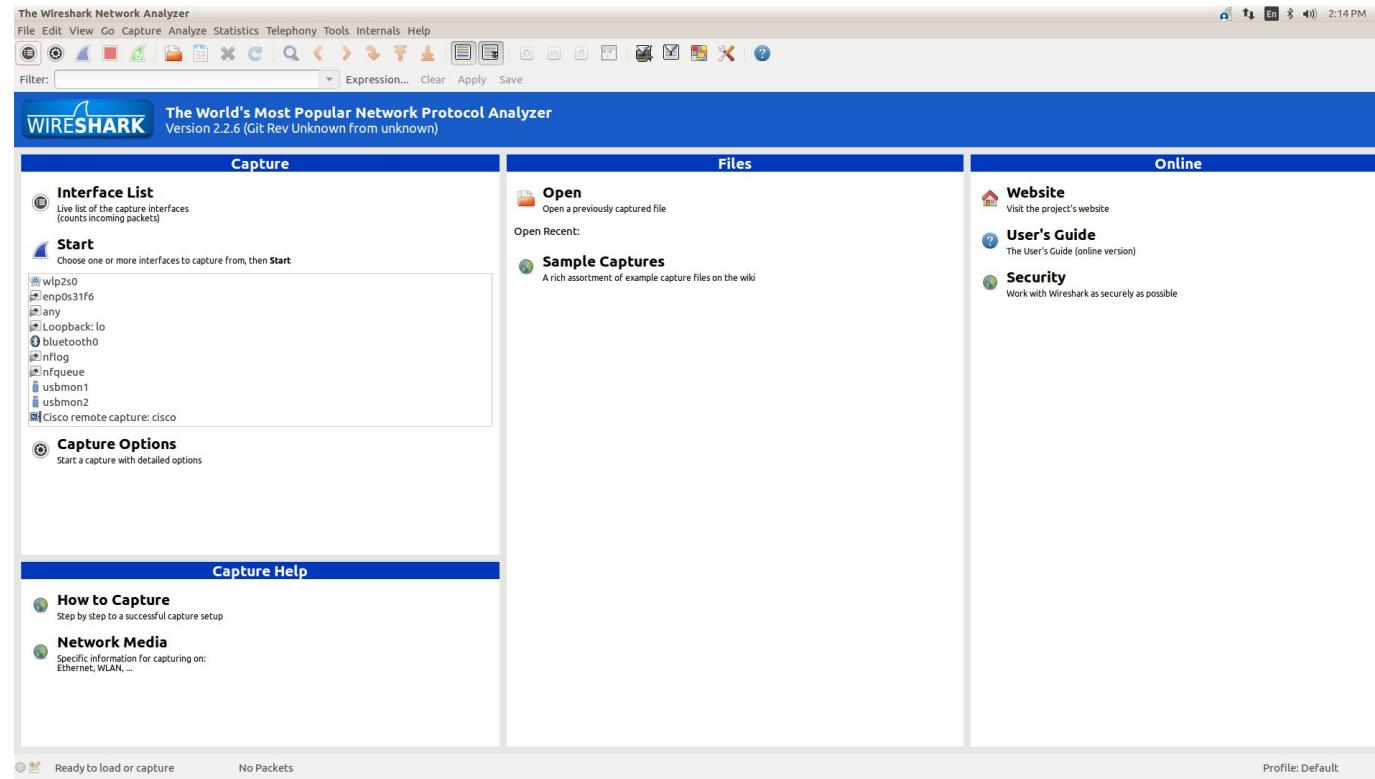
Wireshark runs on almost all popular **platforms**. It runs on Windows, Linux, and macOS.

Wireshark is already installed on **Kali Linux** but if you prefer to use it on any other platform you could easily download the most recent version from <https://www.wireshark.org/>



**Note:** installing wireshark on your machine at work could be a violation to your company or organization policies.

# Starting Wireshark



# Live Packet Capturing with Wireshark

To start capture live packets or live network traffic, you need to select the interface that Wireshark will start listening

If you can not see any interface. Then most likely you are running Wireshark with none-root privilege on Linux like OS or with a none-administrator account on Windows.



## Interface List

Live list of the capture interfaces  
(counts incoming packets)



## Start

Choose one or more interfaces to capture from, then **Start**



wlp2s0

enp0s31f6

any

Loopback: lo

bluetooth0

nflog

nfqueue

usbmon1

usbmon2

Cisco remote capture: cisco

# Live Packet Capturing with Wireshark

The screenshot shows the Wireshark application window. At the top is a toolbar with various icons for file operations, zooming, and analysis. Below the toolbar is a menu bar with 'File', 'Edit', 'View', 'Select', 'Search', 'Statistics', 'Protocols', 'Help', and 'About'. A 'Filter:' dropdown and search fields ('Expression...', 'Clear', 'Apply', 'Save') are located just below the menu. The main area is a table representing the captured network traffic:

| No.     | Time                  | Source         | Destination    | Protocol | Length | Info   |
|---------|-----------------------|----------------|----------------|----------|--------|--|
| 2278909 | 1518124205.983434937  | 74.125.124.189 | 192.168.0.100  | QUIC     | 84     | Payload (Encrypted), PKN: 8452                 |
| 2278910 | 1518124206.008875736  | 192.168.0.100  | 74.125.124.189 | QUIC     | 78     | Payload (Encrypted), PKN: 36, CID: 17132390467 |
| 2278911 | 1518124206.344740281  | 192.168.0.100  | 54.208.108.146 | TLSv1.2  | 129    | Application Data                               |
| 2278912 | 1518124206.381011905  | 54.208.108.146 | 192.168.0.100  | TLSv1.2  | 129    | Application Data                               |
| 2278913 | 1518124206.381032461  | 192.168.0.100  | 54.208.108.146 | TCP      | 66     | 51278 → 443 [ACK] Seq=7120 Ack=11725 Win=1444  |
| 2278914 | 1518124207.361768730  | 137.207.71.197 | 192.168.0.100  | TCP      | 66     | 80 → 60570 [FIN, ACK] Seq=620 Ack=686 Win=5065 |
| 2278915 | 1518124207.402831088  | 192.168.0.100  | 137.207.71.197 | TCP      | 66     | 60570 → 80 [ACK] Seq=686 Ack=621 Win=30331 Len |
| 2278916 | 1518124207.416250631  | 137.207.71.243 | 192.168.0.100  | TCP      | 66     | 80 → 38862 [FIN, ACK] Seq=46810 Ack=6785 Win=1 |
| 2278917 | 1518124207.416560296  | 137.207.71.243 | 192.168.0.100  | TCP      | 66     | 80 → 38860 [FIN, ACK] Seq=72737 Ack=9355 Win=1 |
| 2278918 | 1518124207.416575131  | 137.207.71.243 | 192.168.0.100  | TCP      | 66     | 80 → 38868 [FIN, ACK] Seq=43775 Ack=7632 Win=1 |
| 2278919 | 1518124207.416578823  | 137.207.71.243 | 192.168.0.100  | TCP      | 66     | 80 → 38866 [FIN, ACK] Seq=131983 Ack=6872 Win= |
| 2278920 | 1518124207.416582761  | 137.207.71.243 | 192.168.0.100  | TCP      | 66     | 80 → 38864 [FIN, ACK] Seq=100748 Ack=7642 Win= |
| 2278921 | 1518124207.4587979006 | 192.168.0.100  | 137.207.71.243 | TCP      | 66     | 38862 → 80 [ACK] Seq=6785 Ack=46811 Win=65160  |
| 2278922 | 1518124207.458792892  | 192.168.0.100  | 137.207.71.243 | TCP      | 66     | 38864 → 80 [ACK] Seq=7642 Ack=100749 Win=65168 |
| 2278923 | 1518124207.458794723  | 192.168.0.100  | 137.207.71.243 | TCP      | 66     | 38866 → 80 [ACK] Seq=6872 Ack=131984 Win=65168 |
| 2278924 | 1518124207.458796618  | 192.168.0.100  | 137.207.71.243 | TCP      | 66     | 38868 → 80 [ACK] Seq=7632 Ack=43776 Win=65168  |
| 2278925 | 1518124207.458797702  | 192.168.0.100  | 137.207.71.243 | TCP      | 66     | 38860 → 80 [ACK] Seq=9355 Ack=72738 Win=65160  |
| 2278926 | 1518124208.417617936  | 137.207.71.243 | 192.168.0.100  | TCP      | 66     | 80 → 38870 [FIN, ACK] Seq=176493 Ack=9694 Win= |
| 2278927 | 1518124208.458844464  | 192.168.0.100  | 137.207.71.243 | TCP      | 66     | 38870 → 80 [ACK] Seq=9694 Ack=176494 Win=65535 |
| 2278928 | 1518124208.527391880  | 192.168.0.100  | 74.125.201.189 | QUIC     | 65     | Payload (Encrypted), PKN: 172, CID: 5451951066 |
| 2278929 | 1518124208.582914529  | 74.125.201.189 | 192.168.0.100  | QUIC     | 74     | Payload (Encrypted), PKN: 50432                |
| 2278930 | 1518124209.339620673  | 54.208.108.146 | 192.168.0.100  | TLSv1.2  | 838    | Application Data                               |
| 2278931 | 1518124209.339667512  | 192.168.0.100  | 54.208.108.146 | TCP      | 66     | 51278 → 443 [ACK] Seq=7120 Ack=12497 Win=1444  |
| 2278932 | 1518124209.851145729  | 137.207.71.40  | 192.168.0.100  | TCP      | 66     | 443 → 42332 [FIN, ACK] Seq=1051 Ack=1848 Win=6 |
| 2278933 | 1518124209.894825244  | 192.168.0.100  | 137.207.71.40  | TCP      | 66     | 42332 → 443 [ACK] Seq=1848 Ack=1052 Win=31860  |
| 2278934 | 1518124209.958907031  | 192.168.0.100  | 104.237.191.1  | TCP      | 74     | [TCP Retransmission] 53402 → 443 [SYN] Seq=0 V |

Frame details for the first frame:

- Frame 1: 1392 bytes on wire (11136 bits), 1392 bytes captured (11136 bits) on interface 0
- Ethernet II, Src: D-LinkIn\_68:45:98 (e4:6f:13:68:45:98), Dst: HonHaiPr\_08:71:cf (70:20:84:08:71:cf)
- Internet Protocol Version 4, Src: 172.217.2.161, Dst: 192.168.0.100
- User Datagram Protocol, Src Port: 443, Dst Port: 56492
- QUIC (Quick UDP Internet Connections)

# Wireshark In Nutshell

Simply select the network interface you want to monitor.

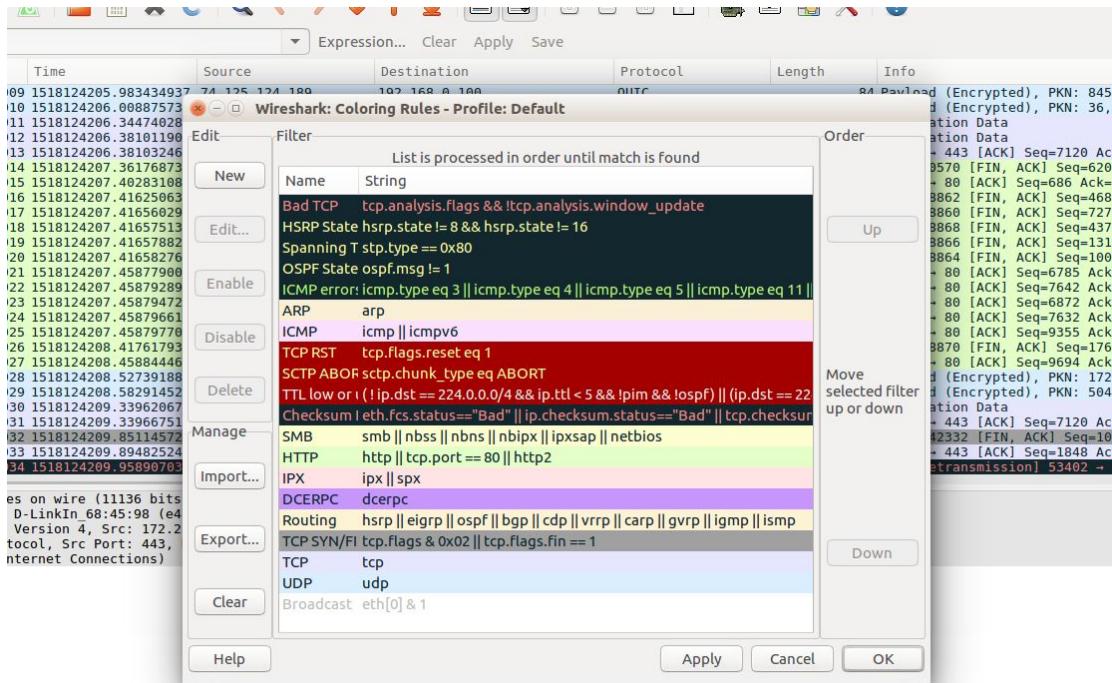
Use the toolbar on the top to start | stop | resume capture

Save the captured packets into a file (pcap file)

Open and load a file with captured packets.

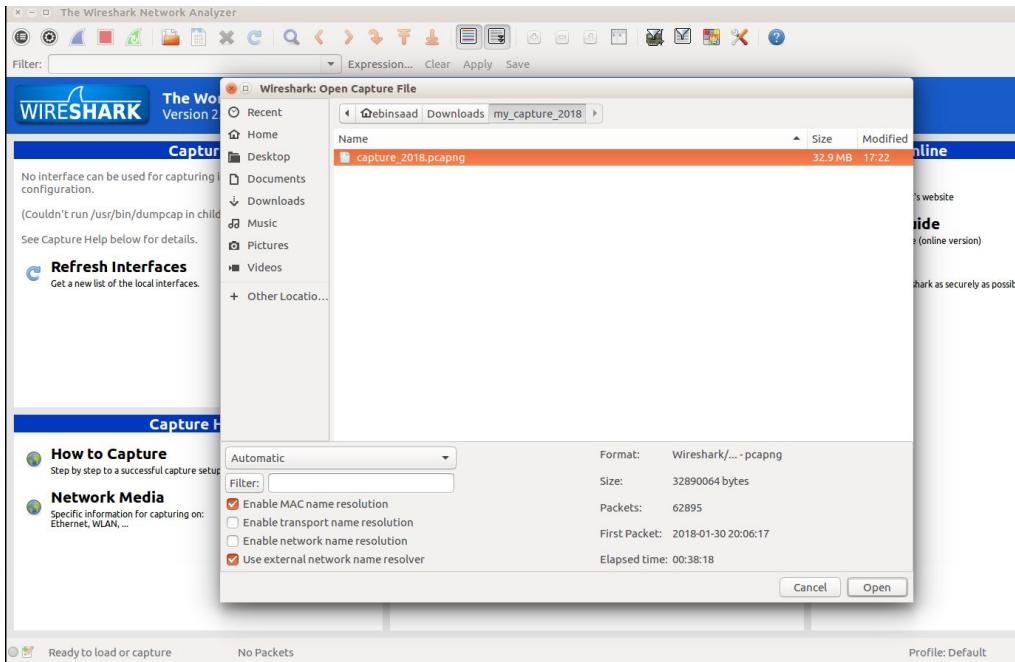
Inspect and analyze packets and traffic using powerful wireshark filters or write Lua script to automate complex analysis tasks.

# Colors in Wireshark



# Open a saved pcap file

From the file menu select open menu item



# Exploring PCAP file with Wireshark

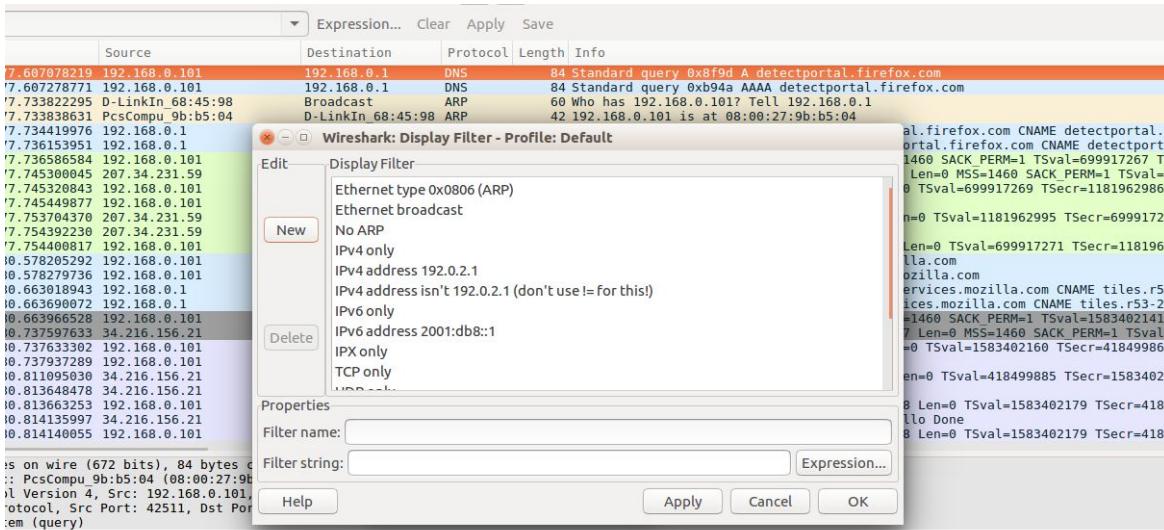
To list of all IP addresses that were seen the network during the capture. Go to **"Statistics -> Endpoints"** and click on the IP tab.

The screenshot shows the Wireshark interface with a list of endpoints. The 'IPv4 Endpoints' tab is selected, displaying 161 entries. Each entry includes the IP address, number of packets, bytes, and various statistics like Tx Packets, Rx Bytes, and Rx Packets. A tooltip for the first entry shows details about the connection to 192.168.0.101. The bottom status bar indicates the file path, packet count, and load time.

| No. | Time                 | Source             | Destination | Protocol   | Length     | Info  |         |                         |                                       |                                    |
|-----|----------------------|--------------------|-------------|------------|------------|---|---------|-------------------------|---------------------------------------|------------------------------------|
| 1   | 1517360777.607078219 | 192.168.0.101      | 192.168.0.1 | DNS        | 84         | Standard query 0x8f9d A detectportal.firefox.com    |         |                         |                                       |                                    |
| 2   | 1517360777.607278771 | 192.168.0.101      | 192.168.0.1 | DNS        | 84         | Standard query 0xb94a AAAA detectportal.firefox.com |         |                         |                                       |                                    |
| 3   | 1517360777.7338      |                    |             |            |            |   |         |                         |                                       |                                    |
| 4   | 1517360777.7338      | 192.168.0.101      | 192.168.0.1 | DNS        | 84         | Standard query 0xb94a AAAA detectportal.firefox.com |         |                         |                                       |                                    |
| 5   | 1517360777.7338      |                    |             |            |            |   |         |                         |                                       |                                    |
| 6   | 1517360777.7361      |                    |             |            |            |   |         |                         |                                       |                                    |
| 7   | 1517360777.7365      |                    |             |            |            |   |         |                         |                                       |                                    |
| 8   | 1517360777.7453      | Address            | Packets     | Bytes      | Tx Packets | Rx Bytes  | Country | AS Number               | Country                               |                                    |
| 9   | 1517360777.7453      | 192.168.0.101      | 35 818      | 12 530 140 | 18 969     | 1 813 478   | 16 849  | 10 716 662              | -                                     | -                                  |
| 10  | 1517360777.7453      | 192.168.0.101      | 11 986      | 3 570 835  | 6 105      | 3 097 662   | 5 881   | 473 173                 | -                                     | -                                  |
| 11  | 1517360777.7537      | 192.168.0.101      | 45          | 4 330      | 21         | 2 162   | 24      | 2 168                   | -                                     | -                                  |
| 12  | 1517360777.7543      | 207.34.231.59      | 45          | 5 932      | 13         | 4 138   | 17      | 1 794                   | -                                     | -                                  |
| 13  | 1517360777.7544      | 34.216.156.21      | 959         | 123 777    | 437        | 67 702  | 522     | 5 025                   | A51513 EdgeCast Network United States | AS5852 TELUS Communications Canada |
| 14  | 1517360780.8134      | 22.19.1.93         | 59          | 1 750      | 1          | 1 750   | 1       | 1                       | -                                     | -                                  |
| 15  | 1517360780.5782      | 192.168.0.101      | 59          | 1 750      | 1          | 1 750   | 1       | 1                       | -                                     | -                                  |
| 16  | 1517360780.6638      | 172.217.2.174      | 960         | 531 383    | 469        | 349 066   | 491     | 183 317                 | AS15169 Google Inc.                   | United States                      |
| 17  | 1517360780.6636      | 172.217.7.14       | 80          | 9 868      | 36         | 5 224   | 44      | 4 644                   | AS15169 Google Inc.                   | United States                      |
| 18  | 1517360780.6639      | 54.191.37.101      | 30          | 6 314      | 13         | 4 262   | 17      | 2 052                   | AS16509 Amazon.com                    | In United States                   |
| 19  | 1517360780.6638      | 151.101.101.184.03 | 232         | 116 006    | 108        | 102 595   | 124     | 13 411                  | AS54113 Fastly                        | United States                      |
| 20  | 1517360780.7376      | 151.201.71.105     | 2 644       | 1 749 206  | 1 244      | 1 613 769   | 1 400   | 135 437                 | AS2914 NTT America, Inc.              | United States                      |
| 21  | 1517360780.7379      | 151.201.71.105     | 32          | 5 764      | 15         | 3 074   | 17      | 2 690                   | AS36351 SoftLayer Techn.              | United States                      |
| 22  | 1517360780.8116      | 50.97.40.233       | 40          | 6 048      | 20         | 2 839   | 20      | 3 209                   | AS16509 Amazon.com                    | In Ireland                         |
| 23  | 1517360780.8136      | 54.154.106.26      | 88          | 29 209     | 41         | 25 480  | 47      | 3 729                   | AS15133 EdgeCast Network              | United States                      |
| 24  | 1517360780.8136      | 72.21.19.113       | 116         | 31 355     | 54         | 24 704  | 62      | 6 651                   | AS16509 Amazon.com                    | In United States                   |
| 25  | 1517360780.8141      | 52.528.89.229      | 46          | 4 392      | 22         | 2 179   | 24      | 2 213                   | -                                     | United States                      |
| 26  | 1517360780.8142      | 34.232.202.198     | 270         | 18 856     | 14         | 14 100  | 39      | 5 346                   | AS3210 BBC                            | United Kingdom                     |
| 27  | 1517360780.8141      | 172.217.1.2        | 1 142       | 303 114    | 554        | 314 800   | 588     | 68 234                  | AS15169 Google Inc.                   | United States                      |
| 28  | 1517360780.8142      | 54.77.65.240       | 85          | 18 051     | 39         | 12 480  | 46      | 5 571                   | AS16509 Amazon.com                    | In Ireland                         |
| 29  | 1517360780.8142      | 184.84.243.35      | 99          | 12 053     | 46         | 5 567   | 53      | 6 486                   | AS20940 Akamai Internal               | United States                      |
| 30  | 1517360780.8142      | 104.31.75.124      | 795         | 687 803    | 445        | 653 016   | 350     | 34 787                  | AS15169 Google Inc.                   | United States                      |
| 31  | 1517360780.8142      | 77.21.18.173       | 45          | 10 366     | 22         | 7 725   | 23      | 2 641                   | AS13335 CloudFlare, Inc.              | United States                      |
| 32  | 1517360780.8142      | 172.217.2.161      | 88          | 25 568     | 43         | 18 487  | 45      | 7 081                   | AS42320 comScore B.V.                 | Netherlands                        |
| 33  | 1517360780.8142      | 23.194.181.179     | 602         | 460 889    | 314        | 432 487   | 288     | 28 402                  | AS15169 Google Inc.                   | United States                      |
| 34  | 1517360780.8142      | 168                | 45 153      | 75         | 35 116     | 93  | 10 037  | AS3320 Deutsche Telekom | Netherlands                           |                                    |
| 35  | 1517360780.8142      | 140                | 60 704      | 60         | 54 270     | 70  | 7 572   | AS16509 Amazon.com      | In United States                      |                                    |

# Wireshark Filters

To list of all IP addresses that were seen the network during the capture. Go to "Analyze -> Display Filters" and click on the IP tab.



# Working with Wireshark Filters

Wireshark filters provide a powerful tool for network traffic inspections and analysis.

We could design simple and complex query to filter the captured traffic and focus our analysis on specific IP address, port, protocol, network interface, etc

| No. | Time                 | Source            | Destination       | Protocol | Length | Info  |
|-----|----------------------|-------------------|-------------------|----------|--------|---|
| 1   | 1517360777.607078219 | 192.168.0.101     | 192.168.0.1       | DNS      | 84     | Standard query 0x8f9d A detectportal.firefox.co |
| 2   | 1517360777.607278771 | 192.168.0.101     | 192.168.0.1       | DNS      | 84     | Standard query 0xb94a AAAA detectportal.firefox |
| 3   | 1517360777.733822295 | D-LinkIn 68:45:98 | Broadcast         | ARP      | 60     | Who has 192.168.0.101? Tell 192.168.0.1         |
| 4   | 1517360777.733838631 | PcsCompu_9b:b5:04 | D-LinkIn_68:45:98 | ARP      | 42     | 192.168.0.101 is at 08:00:27:9b:b5:04           |
| 5   | 1517360777.734419976 | 192.168.0.1       | 192.168.0.101     | DNS      | 405    | Standard query response 0x8f9d A detectportal.f |
| 6   | 1517360777.736153951 | 192.168.0.1       | 192.168.0.101     | DNS      | 223    | Standard query response 0xb94a AAAA detectporta |
| 7   | 1517360777.736586584 | 192.168.0.101     | 207.34.231.59     | TCP      | 74     | 39190 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 |
| 8   | 1517360777.745300045 | 207.34.231.59     | 192.168.0.101     | TCP      | 74     | 80 → 39190 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len |

# Working with Wireshark Filters

Display on TCP Packets, we simply type **tcp** in the filter textbox and hit enter.  
Note that wireshark filter support auto-complete

| No. | Time                 | Source        | Destination   | Protocol | Length | Info   |
|-----|----------------------|---------------|---------------|----------|--------|--|
| 7   | 1517360777.736586584 | 192.168.0.101 | 207.34.231.59 | TCP      | 74     | 39190 → 80 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=699917267 TSecr=699917267 ACK=0 Seq=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1181962986 TSecr=1181962986 |
| 8   | 1517360777.745300045 | 207.34.231.59 | 192.168.0.101 | TCP      | 74     | 80 → 39190 [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=1181962986 TSecr=1181962986 ACK=1 Seq=1 Win=29312 Len=0 TSval=699917269 TSecr=1181962986          |
| 9   | 1517360777.745320843 | 192.168.0.101 | 207.34.231.59 | TCP      | 66     | 39190 → 80 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=699917269 TSecr=1181962986 ACK=0 Seq=2 Win=29336 Len=0 TSval=1181962986 TSecr=1181962986                                    |
| 10  | 1517360777.745449877 | 192.168.0.101 | 207.34.231.59 | HTTP     | 354    | GET /success.txt HTTP/1.1  |
| 11  | 1517360777.753704370 | 207.34.231.59 | 192.168.0.101 | TCP      | 66     | 80 → 39190 [ACK] Seq=1 Ack=289 Win=30048 Len=0 TSval=1181962995 TSecr=699917269 ACK=1 Seq=290 Win=30336 Len=0 TSval=1181962995 TSecr=1181962995                                |
| 12  | 1517360777.754392230 | 207.34.231.59 | 192.168.0.101 | HTTP     | 450    | HTTP/1.1 200 OK (text/plain)   |
| 13  | 1517360777.754400017 | 192.168.0.101 | 207.34.231.59 | TCP      | 66     | 39190 → 80 [ACK] Seq=290 Ack=289 Win=30336 Len=0 TSval=1181962995 TSecr=1181962995 ACK=0 Seq=291 Win=30336 Len=0 TSval=1181962995 TSecr=1181962995                             |

# Working with Wireshark Filters

Let us try another option let us display HTTP traffic contains specific text

**http contains "firefox"**

The screenshot shows the Wireshark interface with a capture file named "capture\_2018.pcapng". The filter bar at the top contains the expression "http contains \"firefox\"". The main window displays a list of network packets. The first five packets in the list are highlighted in green, indicating they match the current filter. These packets are all HTTP GET requests to "/success.txt" from source IP 192.168.0.101 to destination IP 207.34.231.59. The packet at index 49827 is highlighted in red, showing a TCP segment with the status "[TCP Previous segment not captured]". The other two packets in the list are also highlighted in red.

| No.   | Time                 | Source        | Destination   | Protocol | Length | Info                                       |
|-------|----------------------|---------------|---------------|----------|--------|--|
| 10    | 1517360777.745449877 | 192.168.0.101 | 207.34.231.59 | HTTP     | 354    | GET /success.txt HTTP/1.1                  |
| 13068 | 1517360838.820208165 | 192.168.0.101 | 207.34.231.59 | HTTP     | 354    | GET /success.txt HTTP/1.1                  |
| 49827 | 1517362219.323010248 | 192.168.0.105 | 207.34.231.64 | HTTP     | 362    | [TCP Previous segment not captured] GET /s |
| 50436 | 1517362254.445540218 | 192.168.0.105 | 207.34.231.64 | HTTP     | 362    | GET /success.txt HTTP/1.1                  |
| 52305 | 1517362275.338250121 | 192.168.0.105 | 207.34.231.64 | HTTP     | 362    | GET /success.txt HTTP/1.1                  |

# Working with Wireshark Filters

We can also filter based on IP addresses. For example display only the traffic between two IP addresses.

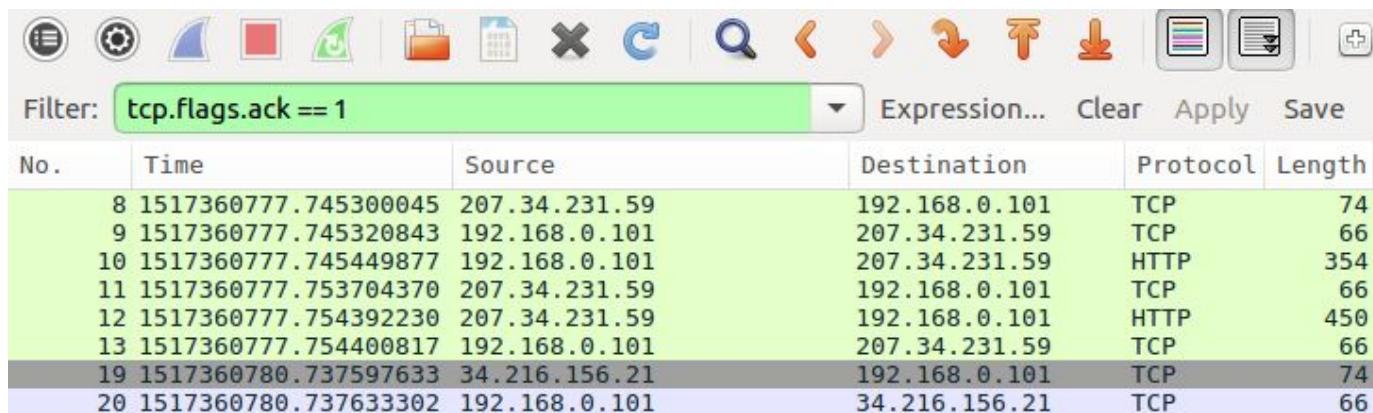
**`ip.src_host==192.168.0.101 and ip.dst_host==207.34.231.59`**

| No.   | Time                 | Source        | Destination   | Protocol | Length | Info                                       |
|-------|----------------------|---------------|---------------|----------|--------|--|
| 10    | 1517360777.745449877 | 192.168.0.101 | 207.34.231.59 | HTTP     | 354    | GET /success.txt HTTP/1.1                  |
| 13068 | 1517360838.820208165 | 192.168.0.101 | 207.34.231.59 | HTTP     | 354    | GET /success.txt HTTP/1.1                  |
| 49827 | 1517362219.323010248 | 192.168.0.105 | 207.34.231.64 | HTTP     | 362    | [TCP Previous segment not captured] GET /s |
| 50436 | 1517362254.445540218 | 192.168.0.105 | 207.34.231.64 | HTTP     | 362    | GET /success.txt HTTP/1.1                  |
| 52305 | 1517362275.338250121 | 192.168.0.105 | 207.34.231.64 | HTTP     | 362    | GET /success.txt HTTP/1.1                  |

# Working with Wireshark Filters

We can also filter based on **protocol options**. For example let us display only the tcp packets where the ack flag is set to 1

**Tcp.flags.ack == 1**

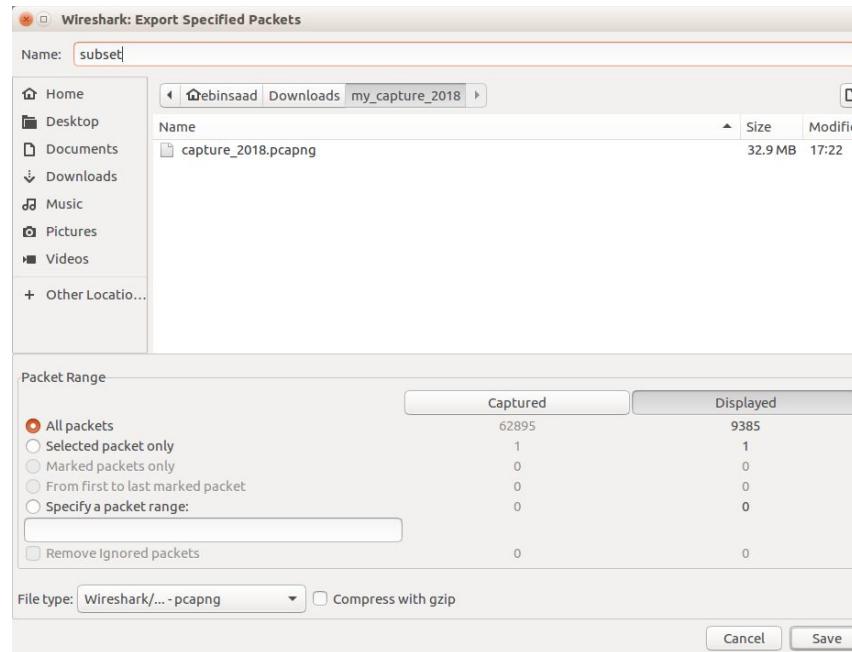


The screenshot shows the Wireshark interface with a green header bar containing the filter expression `Tcp.flags.ack == 1`. Below the header is a toolbar with various icons. The main window displays a table of network traffic. The columns are labeled: No., Time, Source, Destination, Protocol, and Length. The table lists several TCP packets, with rows 8 through 19 highlighted in light green, indicating they match the applied filter. Row 19 is specifically highlighted in dark grey.

| No. | Time                 | Source        | Destination   | Protocol | Length |
|-----|----------------------|---------------|---------------|----------|--------|
| 8   | 1517360777.745300045 | 207.34.231.59 | 192.168.0.101 | TCP      | 74     |
| 9   | 1517360777.745320843 | 192.168.0.101 | 207.34.231.59 | TCP      | 66     |
| 10  | 1517360777.745449877 | 192.168.0.101 | 207.34.231.59 | HTTP     | 354    |
| 11  | 1517360777.753704370 | 207.34.231.59 | 192.168.0.101 | TCP      | 66     |
| 12  | 1517360777.754392230 | 207.34.231.59 | 192.168.0.101 | HTTP     | 450    |
| 13  | 1517360777.754400817 | 192.168.0.101 | 207.34.231.59 | TCP      | 66     |
| 19  | 1517360780.737597633 | 34.216.156.21 | 192.168.0.101 | TCP      | 74     |
| 20  | 1517360780.737633302 | 192.168.0.101 | 34.216.156.21 | TCP      | 66     |

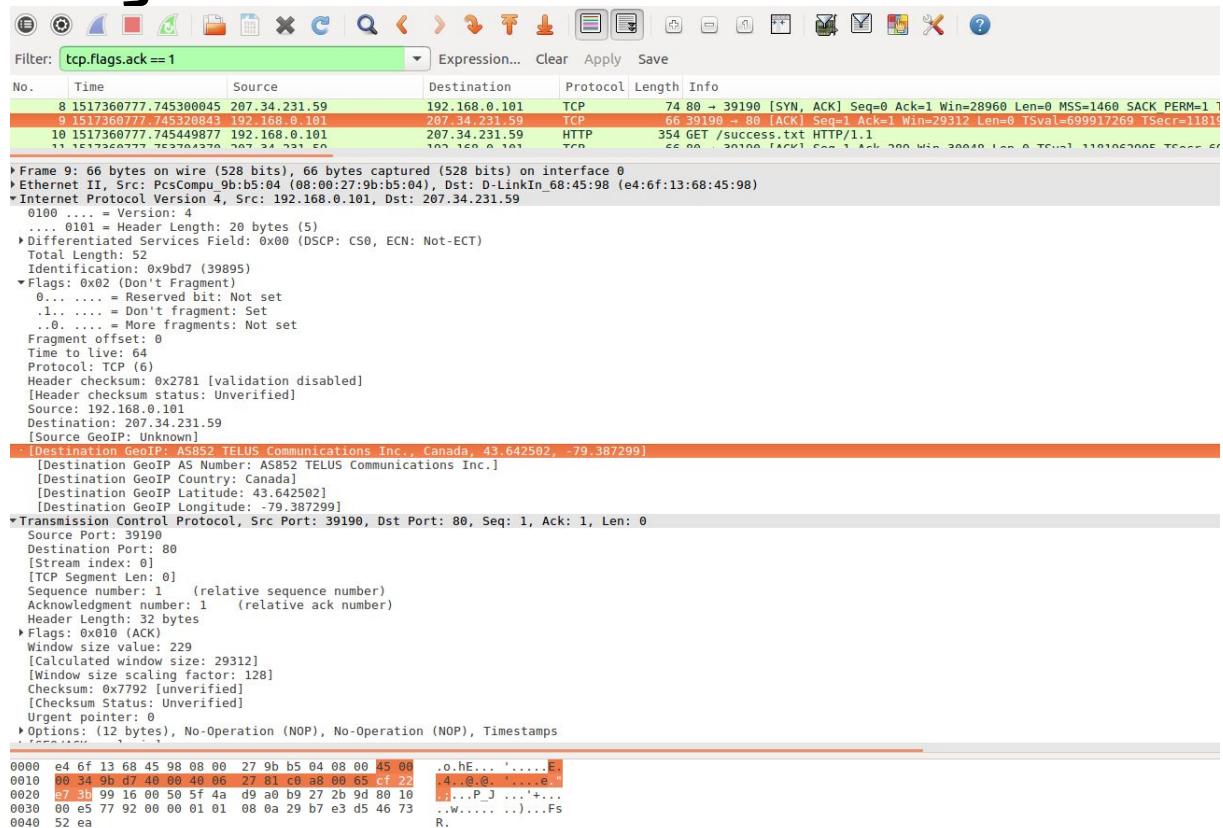
# Working with Wireshark Filters

We could save the filter results into a file. **Select File -> Export Specified packets**



# Wireshark Exploring Packet Details

When you select a **packet** you can see the **details** of the packet .



# Deep Packet Inspection with Wireshark

Download the file [\*\*capture\\_2018.pcapng\*\*](#) from the course site on blackboard.

Open the file using wireshark.

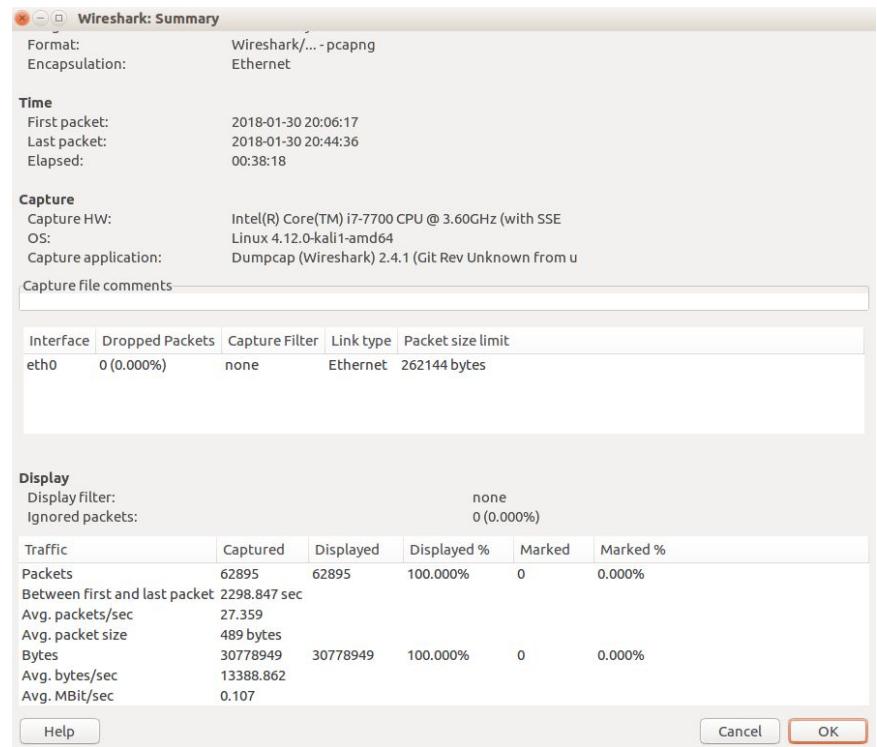
This file contains malicious activities try to:

- Identify the IP addresses involved in the activities.
- Their roles (who is the attacker and who is the victim)
- The type of the attacks, the outcome of the attacks and the severity of the attacks

# DPI with Wireshark: Traffic Statistics

First let us check the traffic summary from the Statistics.

What do you see? What metrics are there?



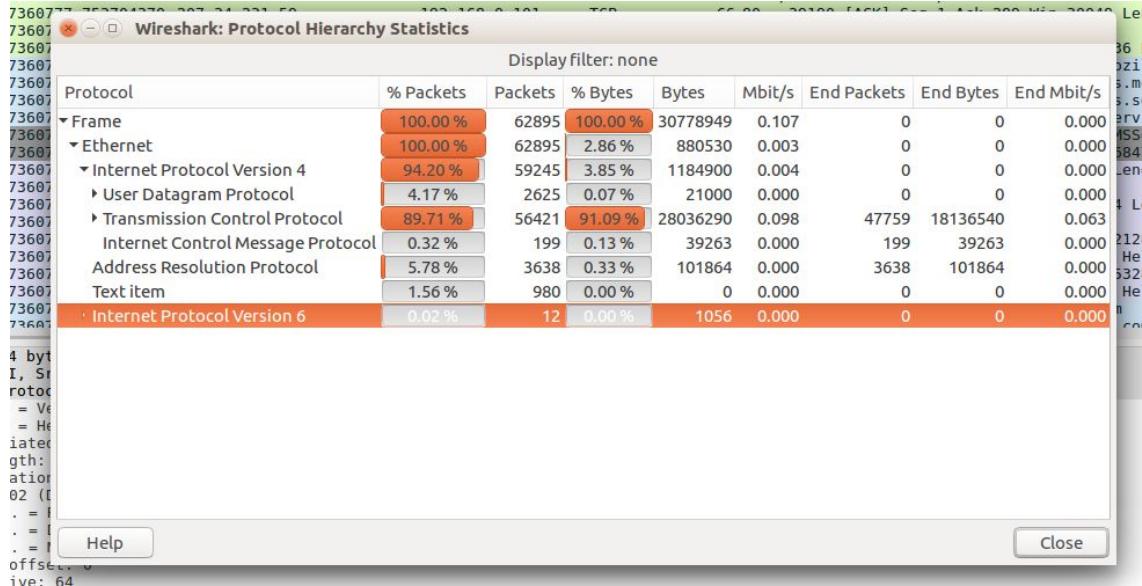
# DPI with Wireshark: Traffic Statistics

| Time          |                     |
|---------------|---------------------|
| First packet: | 2018-01-30 20:06:17 |
| Last packet:  | 2018-01-30 20:44:36 |
| Elapsed:      | 00:38:18            |

| Traffic                                    | Captured  | Displayed | Displayed % | Marked | Marked % |
|--|-----------|-----------|-------------|--------|----------|
| Packets                                    | 62895     | 62895     | 100.000%    | 0      | 0.000%   |
| Between first and last packet 2298.847 sec |           |           |             |        |          |
| Avg. packets/sec                           | 27.359    |           |             |        |          |
| Avg. packet size                           | 489 bytes |           |             |        |          |
| Bytes                                      | 30778949  | 30778949  | 100.000%    | 0      | 0.000%   |
| Avg. bytes/sec                             | 13388.862 |           |             |        |          |
| Avg. MBit/sec                              | 0.107     |           |             |        |          |

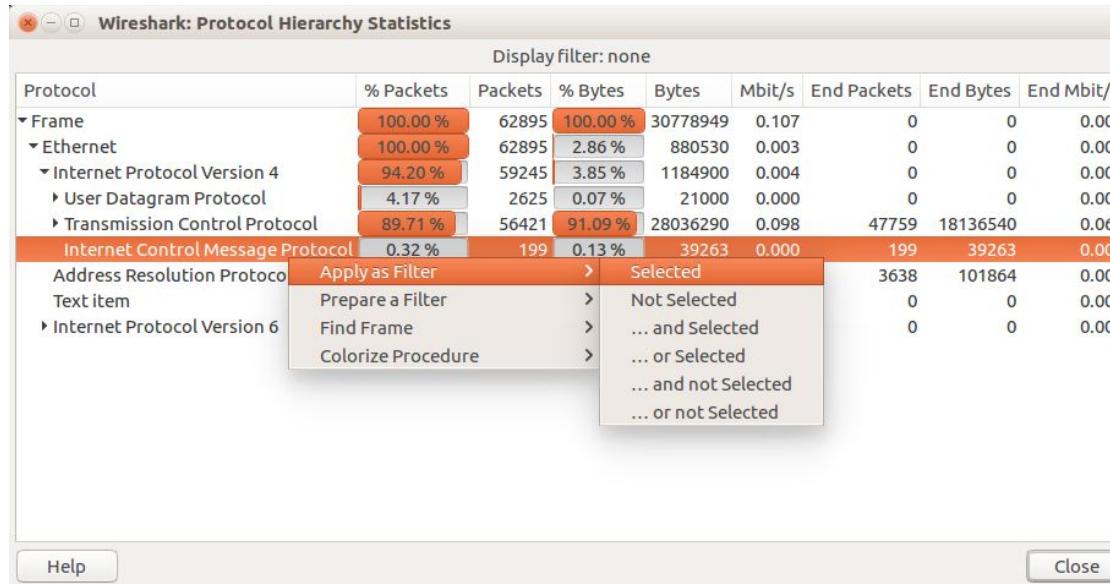
# DPI with Wireshark: Traffic Statistics

Go to **Statistics > Protocol** Hierarchy to see a list of all the protocols present, and a count of the number of packets and bytes for each protocol.



# DPI with Wireshark: Traffic Statistics

You can then right-click on a line and select **Apply As Filter > Selected** to see all the packets of that protocol type.



# DPI with Wireshark

After applying the filter we can see many ICMP request to different hosts on the same local network

| No.   | Time                 | Source        | Destination   | Protocol | Length | Info   |
|-------|----------------------|---------------|---------------|----------|--------|--|
| 7412  | 1517360808.349368375 | 192.168.0.101 | 192.168.0.1   | ICMP     | 182    | Destination unreachable (Port unreachable)     |
| 7560  | 1517360808.859219505 | 192.168.0.101 | 192.168.0.1   | ICMP     | 182    | Destination unreachable (Port unreachable)     |
| 7748  | 1517360809.455023222 | 192.168.0.101 | 192.168.0.1   | ICMP     | 182    | Destination unreachable (Port unreachable)     |
| 35477 | 1517361371.103912001 | 192.168.0.101 | 192.168.0.100 | ICMP     | 162    | Echo (ping) request id=0afb5, seq=295/9985, tt |
| 35478 | 1517361371.104030397 | 192.168.0.100 | 192.168.0.101 | ICMP     | 162    | Echo (ping) reply id=0afb5, seq=295/9985, tt   |
| 35479 | 1517361371.104081317 | 192.168.0.101 | 192.168.0.105 | ICMP     | 162    | Echo (ping) request id=0afb5, seq=295/9985, tt |
| 35480 | 1517361371.104138678 | 192.168.0.101 | 192.168.0.106 | ICMP     | 162    | Echo (ping) request id=0afb5, seq=295/9985, tt |
| 35481 | 1517361371.104425649 | 192.168.0.105 | 192.168.0.101 | ICMP     | 162    | Echo (ping) reply id=0afb5, seq=295/9985, tt   |
| 35482 | 1517361371.125152958 | 192.168.0.101 | 192.168.0.1   | ICMP     | 162    | Echo (ping) request id=0afb5, seq=295/9985, tt |
| 35483 | 1517361371.126148074 | 192.168.0.1   | 192.168.0.101 | ICMP     | 162    | Echo (ping) reply id=0afb5, seq=295/9985, tt   |
| 35485 | 1517361371.132591864 | 192.168.0.101 | 192.168.0.100 | ICMP     | 192    | Echo (ping) request id=0afb6, seq=296/10241, t |
| 35486 | 1517361371.132760255 | 192.168.0.100 | 192.168.0.101 | ICMP     | 192    | Echo (ping) reply id=0afb6, seq=296/10241, t   |
| 35487 | 1517361371.132836156 | 192.168.0.101 | 192.168.0.105 | ICMP     | 192    | Echo (ping) request id=0afb6, seq=296/10241, t |
| 35488 | 1517361371.132939231 | 192.168.0.101 | 192.168.0.106 | ICMP     | 192    | Echo (ping) request id=0afb6, seq=296/10241, t |
| 35489 | 1517361371.133208128 | 192.168.0.105 | 192.168.0.101 | ICMP     | 192    | Echo (ping) reply id=0afb6, seq=296/10241, t   |
| 35492 | 1517361371.150343374 | 192.168.0.101 | 192.168.0.1   | ICMP     | 192    | Echo (ping) request id=0afb6, seq=296/10241, t |
| 35493 | 1517361371.151322074 | 192.168.0.1   | 192.168.0.101 | ICMP     | 192    | Echo (ping) reply id=0afb6, seq=296/10241, t   |
| 35495 | 1517361371.157933058 | 192.168.0.100 | 192.168.0.101 | ICMP     | 370    | Destination unreachable (Port unreachable)     |
| 35498 | 1517361371.158375220 | 192.168.0.105 | 192.168.0.101 | ICMP     | 370    | Destination unreachable (Port unreachable)     |
| 35500 | 1517361371.176641931 | 192.168.0.1   | 192.168.0.101 | ICMP     | 370    | Destination unreachable (Port unreachable)     |
| 35523 | 1517361371.277124930 | 192.168.0.101 | 192.168.0.100 | ICMP     | 162    | Echo (ping) request id=0afb5, seq=295/9985, tt |
| 35524 | 1517361371.277245979 | 192.168.0.100 | 192.168.0.101 | ICMP     | 162    | Echo (ping) reply id=0afb5, seq=295/9985, tt   |
| 35525 | 1517361371.297102124 | 192.168.0.101 | 192.168.0.105 | ICMP     | 162    | Echo (ping) request id=0afb5, seq=295/9985, tt |
| 35526 | 1517361371.297412252 | 192.168.0.105 | 192.168.0.101 | ICMP     | 162    | Echo (ping) reply id=0afb5, seq=295/9985, tt   |
| 35527 | 1517361371.304271054 | 192.168.0.101 | 192.168.0.106 | ICMP     | 162    | Echo (ping) request id=0afb5, seq=295/9985, tt |
| 35529 | 1517361371.306471625 | 192.168.0.101 | 192.168.0.100 | ICMP     | 192    | Echo (ping) request id=0afb6, seq=296/10241, t |
| 35530 | 1517361371.306592997 | 192.168.0.100 | 192.168.0.101 | ICMP     | 192    | Echo (ping) reply id=0afb6, seq=296/10241, t   |
| 35532 | 1517361371.337735864 | 192.168.0.101 | 192.168.0.105 | ICMP     | 192    | Echo (ping) request id=0afb6, seq=296/10241, t |

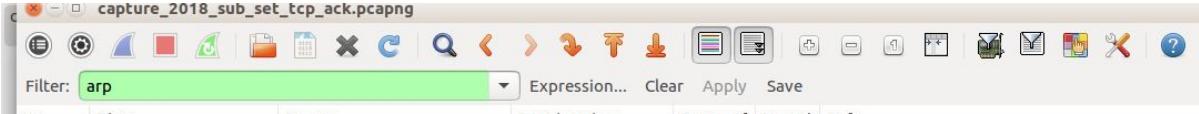
# DPI with Wireshark

The ICMP traffic seems to be **Ping sweep** activities. We could investigate this possibility by applying another filter. `icmp.type==8 or icmp.type==0`

| No.   | Time                  | Source        | Destination   | Protocol | Length | Info   |
|-------|-----------------------|---------------|---------------|----------|--------|--|
| 35478 | 1517361371.104030397  | 192.168.0.100 | 192.168.0.101 | ICMP     | 162    | Echo (ping) reply id=0afb5, seq=295/9985, ttl=64 (request in 35477)      |
| 35479 | 1517361371.1040801317 | 192.168.0.101 | 192.168.0.105 | ICMP     | 162    | Echo (ping) request id=0afb5, seq=295/9985, ttl=58 (reply in 35481)      |
| 35480 | 1517361371.104138678  | 192.168.0.101 | 192.168.0.106 | ICMP     | 162    | Echo (ping) request id=0afb5, seq=295/9985, ttl=58 (no response found!)  |
| 35481 | 1517361371.104425649  | 192.168.0.105 | 192.168.0.101 | ICMP     | 162    | Echo (ping) reply id=0afb5, seq=295/9985, ttl=64 (request in 35479)      |
| 35482 | 1517361371.125152958  | 192.168.0.101 | 192.168.0.1   | ICMP     | 162    | Echo (ping) request id=0afb5, seq=295/9985, ttl=49 (reply in 35483)      |
| 35483 | 1517361371.126148074  | 192.168.0.1   | 192.168.0.101 | ICMP     | 162    | Echo (ping) reply id=0afb5, seq=295/9985, ttl=255 (request in 35482)     |
| 35485 | 1517361371.132591864  | 192.168.0.101 | 192.168.0.100 | ICMP     | 192    | Echo (ping) request id=0afb6, seq=296/10241, ttl=56 (reply in 35486)     |
| 35486 | 1517361371.132760255  | 192.168.0.100 | 192.168.0.101 | ICMP     | 192    | Echo (ping) reply id=0afb6, seq=296/10241, ttl=64 (request in 35485)     |
| 35487 | 1517361371.132836156  | 192.168.0.101 | 192.168.0.105 | ICMP     | 192    | Echo (ping) request id=0afb6, seq=296/10241, ttl=39 (reply in 35489)     |
| 35488 | 1517361371.132939231  | 192.168.0.101 | 192.168.0.106 | ICMP     | 192    | Echo (ping) request id=0afb6, seq=296/10241, ttl=40 (no response found!) |
| 35489 | 1517361371.133208128  | 192.168.0.105 | 192.168.0.101 | ICMP     | 192    | Echo (ping) reply id=0afb6, seq=296/10241, ttl=64 (request in 35487)     |
| 35492 | 1517361371.150343374  | 192.168.0.101 | 192.168.0.1   | ICMP     | 192    | Echo (ping) request id=0afb6, seq=296/10241, ttl=51 (reply in 35493)     |
| 35493 | 1517361371.151322074  | 192.168.0.1   | 192.168.0.101 | ICMP     | 192    | Echo (ping) reply id=0afb6, seq=296/10241, ttl=255 (request in 35492)    |
| 35523 | 1517361371.277124930  | 192.168.0.101 | 192.168.0.100 | ICMP     | 162    | Echo (ping) request id=0afb5, seq=295/9985, ttl=50 (reply in 35524)      |
| 35524 | 1517361371.277245979  | 192.168.0.100 | 192.168.0.101 | ICMP     | 162    | Echo (ping) reply id=0afb5, seq=295/9985, ttl=64 (request in 35523)      |
| 35525 | 1517361371.297102124  | 192.168.0.101 | 192.168.0.105 | ICMP     | 162    | Echo (ping) request id=0afb5, seq=295/9985, ttl=58 (reply in 35526)      |
| 35526 | 1517361371.297412252  | 192.168.0.105 | 192.168.0.101 | ICMP     | 162    | Echo (ping) reply id=0afb5, seq=295/9985, ttl=64 (request in 35525)      |
| 35527 | 1517361371.304271054  | 192.168.0.101 | 192.168.0.106 | ICMP     | 162    | Echo (ping) request id=0afb5, seq=295/9985, ttl=40 (no response found!)  |
| 35529 | 1517361371.306471625  | 192.168.0.101 | 192.168.0.100 | ICMP     | 192    | Echo (ping) request id=0afb6, seq=296/10241, ttl=43 (reply in 35530)     |
| 35530 | 1517361371.306592997  | 192.168.0.100 | 192.168.0.101 | ICMP     | 192    | Echo (ping) reply id=0afb6, seq=296/10241, ttl=64 (request in 35529)     |
| 35532 | 1517361371.337735864  | 192.168.0.101 | 192.168.0.105 | ICMP     | 192    | Echo (ping) request id=0afb6, seq=296/10241, ttl=37 (reply in 35534)     |
| 35533 | 1517361371.337805440  | 192.168.0.101 | 192.168.0.106 | ICMP     | 192    | Echo (ping) request id=0afb6, seq=296/10241, ttl=37 (no response found!) |
| 35534 | 1517361371.337989191  | 192.168.0.105 | 192.168.0.101 | ICMP     | 192    | Echo (ping) reply id=0afb6, seq=296/10241, ttl=64 (request in 35532)     |

# DPI with Wireshark

Let us try to check for **ARP sweep**. Where the attacker send ARP broadcast (for broadcast, destination MAC will be **0xff:ff:ff:ff:ff:ff**)



The screenshot shows the Wireshark interface with a packet list window. A green bar at the top of the list window contains the text "capture\_2018\_sub\_set\_tcp\_ack.pcapng". Below this, there is a toolbar with various icons. The main list area has columns: No., Time, Source, Destination, Protocol, Length, and Info. A green bar at the top of the list area contains the text "Filter: arp". Below this, there are buttons for Expression..., Clear, Apply, and Save. The list itself contains many rows of ARP requests, all originating from "PcsCompu\_9b:b5:04" and directed to "Broadcast". The "Protocol" column shows "ARP" for all entries. The "Info" column provides detailed information about each request, such as the target MAC address being "00:0c:29" and the source IP being "192.168.0.101". The list is very long, indicating a continuous sequence of ARP requests.

| No.   | Time                  | Source            | Destination       | Protocol | Length | Info                                      |
|-------|-----------------------|-------------------|-------------------|----------|--------|---|
| 3     | 1517360777.733822295  | D-LinkIn_68:45:98 | Broadcast         | ARP      | 60     | Who has 192.168.0.101? Tell 192.168.0.1   |
| 4     | 1517360777.733838631  | PcsCompu_9b:b5:04 | D-LinkIn_68:45:98 | ARP      | 42     | 192.168.0.101 is at 08:00:27:9b:b5:04     |
| 13085 | 1517360839.295500558  | D-LinkIn_68:45:98 | PcsCompu_9b:b5:04 | ARP      | 60     | Who has 192.168.0.101? Tell 192.168.0.1   |
| 13086 | 1517360839.295530194  | PcsCompu_9b:b5:04 | D-LinkIn_68:45:98 | ARP      | 42     | 192.168.0.101 is at 08:00:27:9b:b5:04     |
| 13316 | 1517360844.789560871  | D-LinkIn_68:45:98 | Broadcast         | ARP      | 60     | Who has 192.168.0.106? Tell 192.168.0.1   |
| 17970 | 1517360899.195266912  | D-LinkIn_68:45:98 | PcsCompu_9b:b5:04 | ARP      | 60     | Who has 192.168.0.101? Tell 192.168.0.1   |
| 17971 | 1517360899.195280041  | PcsCompu_9b:b5:04 | D-LinkIn_68:45:98 | ARP      | 42     | 192.168.0.101 is at 08:00:27:9b:b5:04     |
| 19404 | 1517360959.245630230  | D-LinkIn_68:45:98 | PcsCompu_9b:b5:04 | ARP      | 60     | Who has 192.168.0.101? Tell 192.168.0.1   |
| 19405 | 1517360959.245660432  | PcsCompu_9b:b5:04 | D-LinkIn_68:45:98 | ARP      | 42     | 192.168.0.101 is at 08:00:27:9b:b5:04     |
| 20027 | 1517361019.775566189  | D-LinkIn_68:45:98 | PcsCompu_9b:b5:04 | ARP      | 60     | Who has 192.168.0.101? Tell 192.168.0.1   |
| 20028 | 1517361019.775579599  | PcsCompu_9b:b5:04 | D-LinkIn_68:45:98 | ARP      | 42     | 192.168.0.101 is at 08:00:27:9b:b5:04     |
| 20262 | 1517361080.105742256  | D-LinkIn_68:45:98 | PcsCompu_9b:b5:04 | ARP      | 60     | Who has 192.168.0.101? Tell 192.168.0.1   |
| 20263 | 1517361080.105772902  | PcsCompu_9b:b5:04 | D-LinkIn_68:45:98 | ARP      | 42     | 192.168.0.101 is at 08:00:27:9b:b5:04     |
| 20385 | 1517361141.645886083  | D-LinkIn_68:45:98 | PcsCompu_9b:b5:04 | ARP      | 60     | Who has 192.168.0.101? Tell 192.168.0.1   |
| 20386 | 1517361141.645916709  | PcsCompu_9b:b5:04 | D-LinkIn_68:45:98 | ARP      | 42     | 192.168.0.101 is at 08:00:27:9b:b5:04     |
| 20395 | 1517361152.982795301  | PcsCompu_9b:b5:04 | Broadcast         | ARP      | 42     | Who has 192.168.0.106? Tell 192.168.0.101 |
| 20396 | 1517361152.983010348  | PcsCompu_98:93:c7 | PcsCompu_9b:b5:04 | ARP      | 60     | 192.168.0.106 is at 08:00:27:98:93:c7     |
| 20417 | 1517361162.2124127690 | PcsCompu_9b:b5:04 | D-LinkIn_68:45:98 | ARP      | 42     | Who has 192.168.0.1? Tell 192.168.0.101   |
| 20418 | 1517361162.212415295  | D-LinkIn_68:45:98 | PcsCompu_9b:b5:04 | ARP      | 60     | 192.168.0.1 is at e4:6f:13:68:45:98       |
| 20456 | 1517361203.085844304  | D-LinkIn_68:45:98 | PcsCompu_9b:b5:04 | ARP      | 60     | Who has 192.168.0.101? Tell 192.168.0.1   |
| 20457 | 1517361203.085859980  | PcsCompu_9b:b5:04 | D-LinkIn_68:45:98 | ARP      | 42     | 192.168.0.101 is at 08:00:27:9b:b5:04     |
| 20463 | 1517361223.652243423  | PcsCompu_9b:b5:04 | D-LinkIn_68:45:98 | ARP      | 42     | Who has 192.168.0.1? Tell 192.168.0.101   |
| 20464 | 1517361223.652281059  | D-LinkIn_68:45:98 | PcsCompu_9b:b5:04 | ARP      | 60     | 192.168.0.1 is at e4:6f:13:68:45:98       |
| 20487 | 1517361296.010251537  | PcsCompu_9b:b5:04 | Broadcast         | ARP      | 42     | Who has 192.168.0.1? Tell 192.168.0.101   |
| 20488 | 1517361296.010343558  | PcsCompu_9b:b5:04 | Broadcast         | ARP      | 42     | Who has 192.168.0.2? Tell 192.168.0.101   |
| 20489 | 1517361296.0104040396 | PcsCompu_9b:b5:04 | Broadcast         | ARP      | 42     | Who has 192.168.0.3? Tell 192.168.0.101   |
| 20490 | 1517361296.010445277  | PcsCompu_9b:b5:04 | Broadcast         | ARP      | 42     | Who has 192.168.0.4? Tell 192.168.0.101   |

# DPI with Wireshark

We can see that the Ip address **192.168.0.101** is performing **ARP sweep** trying to find the MAC address for every possible live host in the network

| Time                        | Source            | Destination | Type | Content                                     |
|-----------------------------|-------------------|-------------|------|---|
| 20500 1517361296.214917463  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.9? Tell 192.168.0.101  |
| 20507 1517361296.411459913  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.10? Tell 192.168.0.101 |
| 20508 1517361296.411687978  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.17? Tell 192.168.0.101 |
| 20509 1517361296.411925239  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.18? Tell 192.168.0.101 |
| 20510 1517361296.412134620  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.19? Tell 192.168.0.101 |
| 20511 1517361296.412305369  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.22? Tell 192.168.0.101 |
| 20512 1517361296.414838427  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.23? Tell 192.168.0.101 |
| 20513 1517361296.415037876  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.26? Tell 192.168.0.101 |
| 20514 1517361296.611755085  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.17? Tell 192.168.0.101 |
| 20515 1517361296.613989810  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.18? Tell 192.168.0.101 |
| 20516 1517361296.614036507  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.19? Tell 192.168.0.101 |
| 20517 1517361296.614062837  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.29? Tell 192.168.0.101 |
| 20518 1517361296.614105951  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.30? Tell 192.168.0.101 |
| 20519 1517361296.616263294  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.31? Tell 192.168.0.101 |
| 20520 1517361296.616313302  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.32? Tell 192.168.0.101 |
| 20521 1517361296.812354334  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.33? Tell 192.168.0.101 |
| 20522 1517361296.815788433  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.34? Tell 192.168.0.101 |
| 20523 1517361296.816145001  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.35? Tell 192.168.0.101 |
| 20524 1517361296.816273620  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.36? Tell 192.168.0.101 |
| 20525 1517361296.816382025  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.37? Tell 192.168.0.101 |
| 20526 1517361296.818933432  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.38? Tell 192.168.0.101 |
| 20527 1517361296.819108928  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.39? Tell 192.168.0.101 |
| 20528 1517361297.012886026  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.40? Tell 192.168.0.101 |
| 20529 1517361297.016300509  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.41? Tell 192.168.0.101 |
| 20530 1517361297.018979822  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.42? Tell 192.168.0.101 |
| 20531 1517361297.019124501  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.43? Tell 192.168.0.101 |
| 20532 1517361297.019203876  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.44? Tell 192.168.0.101 |
| 20533 1517361297.021787021  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.45? Tell 192.168.0.101 |
| 20534 1517361297.022024641  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.46? Tell 192.168.0.101 |
| 20535 1517361297.213202195  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.47? Tell 192.168.0.101 |
| 20536 1517361297.216353495  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.48? Tell 192.168.0.101 |
| 20537 1517361297.219033357  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.49? Tell 192.168.0.101 |
| 20538 1517361297.221323932  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.50? Tell 192.168.0.101 |
| 20539 1517361297.221388223  | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.51? Tell 192.168.0.101 |
| 20540 1517361297.2236666748 | PcsCompu_9b:b5:04 | Broadcast   | ARP  | 42 Who has 192.168.0.52? Tell 192.168.0.101 |

# DPI with Wireshark

When an attacker perform an ARP sweep it is possible that he either want to detect live hosts assuming some firewall block Ping messages (ICMP request)

The other option is the attacker is planning for an ARP poisoning or ARP spoofing attack.

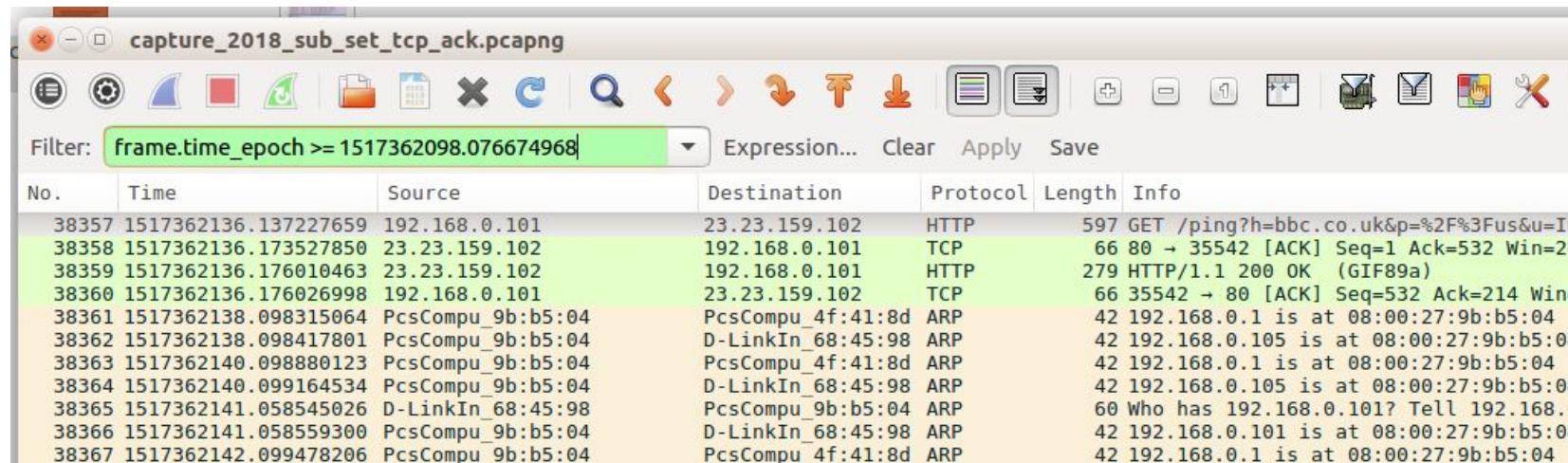
To check for **ARP spoofing** we will continue inspecting the ARP packets.

|       |                      |                   |                       |  |
|-------|----------------------|-------------------|-----------------------|--|
| 62399 | 1517362920.389754503 | PcsCompu_9b:b5:04 | PcsCompu_4f:41:8d ARP | 42 192.168.0.1 is at 08:00:27:9b:b5:04   |
| 62400 | 1517362920.389864332 | PcsCompu_9b:b5:04 | D-LinkIn_68:45:98 ARP | 42 192.168.0.105 is at 08:00:27:9b:b5:04 |
| 62407 | 1517362922.390289030 | PcsCompu_9b:b5:04 | PcsCompu_4f:41:8d ARP | 42 192.168.0.1 is at 08:00:27:9b:b5:04   |
| 62408 | 1517362922.390499816 | PcsCompu_9b:b5:04 | D-LinkIn_68:45:98 ARP | 42 192.168.0.105 is at 08:00:27:9b:b5:04 |
| 62409 | 1517362924.390821698 | PcsCompu_9b:b5:04 | PcsCompu_4f:41:8d ARP | 42 192.168.0.1 is at 08:00:27:9b:b5:04   |
| 62410 | 1517362924.390891989 | PcsCompu_9b:b5:04 | D-LinkIn_68:45:98 ARP | 42 192.168.0.105 is at 08:00:27:9b:b5:04 |
| 62411 | 1517362926.391326614 | PcsCompu_9b:b5:04 | PcsCompu_4f:41:8d ARP | 42 192.168.0.1 is at 08:00:27:9b:b5:04   |
| 62412 | 1517362926.391563288 | PcsCompu_9b:b5:04 | D-LinkIn_68:45:98 ARP | 42 192.168.0.105 is at 08:00:27:9b:b5:04 |
| 62413 | 1517362928.392064961 | PcsCompu_9b:b5:04 | PcsCompu_4f:41:8d ARP | 42 192.168.0.1 is at 08:00:27:9b:b5:04   |

# DPI with Wireshark

We want to investigate what the attacker did after he executed the ARP spoofing. We could use the following filter

**frame.time\_epoch >= 1517362098.076674968**



The screenshot shows the Wireshark interface with a packet capture titled "capture\_2018\_sub\_set\_tcp\_ack.pcapng". The filter bar at the top contains the expression "frame.time\_epoch >= 1517362098.076674968". The main window displays a list of network packets. The first few rows are highlighted in green, indicating they match the applied filter. The columns shown are No., Time, Source, Destination, Protocol, Length, and Info.

| No.   | Time                 | Source            | Destination       | Protocol | Length | Info                                    |
|-------|----------------------|-------------------|-------------------|----------|--------|---|
| 38357 | 1517362136.137227659 | 192.168.0.101     | 23.23.159.102     | HTTP     | 597    | GET /ping?h=bbc.co.uk&p=%2F%3Fus&u=I%   |
| 38358 | 1517362136.173527850 | 23.23.159.102     | 192.168.0.101     | TCP      | 66     | 80 → 35542 [ACK] Seq=1 Ack=532 Win=28   |
| 38359 | 1517362136.176010463 | 23.23.159.102     | 192.168.0.101     | HTTP     | 279    | HTTP/1.1 200 OK (GIF89a)                |
| 38360 | 1517362136.176026998 | 192.168.0.101     | 23.23.159.102     | TCP      | 66     | 35542 → 80 [ACK] Seq=532 Ack=214 Win=   |
| 38361 | 1517362138.098315064 | PcsCompu_9b:b5:04 | PcsCompu_4f:41:8d | ARP      | 42     | 192.168.0.1 is at 08:00:27:9b:b5:04     |
| 38362 | 1517362138.098417801 | PcsCompu_9b:b5:04 | D-LinkIn_68:45:98 | ARP      | 42     | 192.168.0.105 is at 08:00:27:9b:b5:04   |
| 38363 | 1517362140.098880123 | PcsCompu_9b:b5:04 | PcsCompu_4f:41:8d | ARP      | 42     | 192.168.0.1 is at 08:00:27:9b:b5:04     |
| 38364 | 1517362140.099164534 | PcsCompu_9b:b5:04 | D-LinkIn_68:45:98 | ARP      | 42     | 192.168.0.105 is at 08:00:27:9b:b5:04   |
| 38365 | 1517362141.058545026 | D-LinkIn_68:45:98 | PcsCompu_9b:b5:04 | ARP      | 60     | Who has 192.168.0.101? Tell 192.168.0.1 |
| 38366 | 1517362141.058559300 | PcsCompu_9b:b5:04 | D-LinkIn_68:45:98 | ARP      | 42     | 192.168.0.101 is at 08:00:27:9b:b5:04   |
| 38367 | 1517362142.099478206 | PcsCompu_9b:b5:04 | PcsCompu_4f:41:8d | ARP      | 42     | 192.168.0.1 is at 08:00:27:9b:b5:04     |

# DPI with Wireshark

We can see some HTTP traffic (unencrypted HTTP), to focus on those traffic we could update the filter by appending to it and http

**frame.time\_epoch >= 1517362098.076674968 and http**

| No.   | Time                 | Source        | Destination   | Protocol | Length | Info                           |
|-------|----------------------|---------------|---------------|----------|--------|--------------------------------|
| 38357 | 1517362136.137227659 | 192.168.0.101 | 23.23.159.102 | HTTP     | 597    | GET /ping?h=bbc.co.uk&p=%2F%3F |
| 38359 | 1517362136.176010463 | 23.23.159.102 | 192.168.0.101 | HTTP     | 279    | HTTP/1.1 200 OK (GIF89a)       |
| 38545 | 1517362165.928511425 | 192.168.0.105 | 72.21.91.29   | OCSP     | 505    | Request                        |
| 38547 | 1517362165.928565505 | 192.168.0.105 | 72.21.91.29   | OCSP     | 505    | Request                        |
| 38561 | 1517362165.928995430 | 192.168.0.105 | 72.21.91.29   | OCSP     | 505    | Request                        |
| 38563 | 1517362165.929037731 | 192.168.0.105 | 72.21.91.29   | OCSP     | 505    | Request                        |
| 38573 | 1517362165.971132612 | 72.21.91.29   | 192.168.0.105 | OCSP     | 854    | Response                       |
| 38577 | 1517362165.971606756 | 72.21.91.29   | 192.168.0.105 | OCSP     | 854    | Response                       |
| 38581 | 1517362165.972445156 | 72.21.91.29   | 192.168.0.105 | OCSP     | 854    | Response                       |
| 38585 | 1517362165.973028734 | 72.21.91.29   | 192.168.0.105 | OCSP     | 854    | Response                       |
| 40672 | 1517362177.765466226 | 192.168.0.105 | 172.217.0.238 | OCSP     | 503    | Request                        |
| 40686 | 1517362177.769884169 | 192.168.0.105 | 172.217.0.238 | OCSP     | 503    | Request                        |
| 40688 | 1517362177.769900051 | 192.168.0.105 | 172.217.0.238 | OCSP     | 503    | Request                        |
| 40689 | 1517362177.769900050 | 192.168.0.105 | 172.217.0.238 | OCSP     | 503    | Request                        |

# DPI with Wireshark

Is it possible that the victim send sensitive data over http. We could extend our filter to investigate the http payload for some keywords.

**frame.time\_epoch >= 1517362098.076674968 and http contains "login"**

| Filter: i >= 1517362098.076674968 and http contains "login" ▾ Expression... Clear Apply Save |                      |               |               |          |        |   |
|--|----------------------|---------------|---------------|----------|--------|---|
| No.  | Time                 | Source        | Destination   | Protocol | Length | Info  |
| 50703  | 1517362255.667692956 | 192.168.0.1   | 192.168.0.105 | HTTP     | 1299   | HTTP/1.1 200 OK (text/plain)  |
| 51029  | 1517362255.931557349 | 192.168.0.1   | 192.168.0.105 | HTTP     | 838    | HTTP/1.1 200 OK (text/plain)  |
| 51077  | 1517362256.167388501 | 192.168.0.1   | 192.168.0.105 | HTTP     | 351    | HTTP/1.1 200 OK (text/html)   |
| 51973  | 1517362256.655918629 | 192.168.0.1   | 192.168.0.105 | HTTP     | 838    | HTTP/1.1 200 OK (text/plain)  |
| 52295  | 1517362275.303168495 | 192.168.0.105 | 192.168.0.1   | HTTP     | 629    | POST /goform/formLogin HTTP/1.1 (application/x-www-form-urlencoded) |
| 52357  | 1517362275.553080538 | 192.168.0.1   | 192.168.0.105 | HTTP     | 351    | HTTP/1.1 200 OK (text/html)   |
| 52441  | 1517362276.347162631 | 192.168.0.105 | 192.168.0.1   | HTTP     | 438    | GET /login_fail.asp HTTP/1.1  |
| 52457  | 1517362276.663696374 | 192.168.0.1   | 192.168.0.105 | HTTP     | 1014   | HTTP/1.1 200 OK (text/html)   |
| 52471  | 1517362276.706856473 | 192.168.0.105 | 192.168.0.1   | HTTP     | 397    | GET /comm.asp HTTP/1.1  |
| 52501  | 1517362277.026984322 | 192.168.0.105 | 192.168.0.1   | HTTP     | 397    | GET /comm.asp HTTP/1.1  |
| 52503  | 1517362277.027128556 | 192.168.0.105 | 192.168.0.1   | HTTP     | 355    | GET /language_pack.js HTTP/1.1                                      |

# DPI with Wireshark

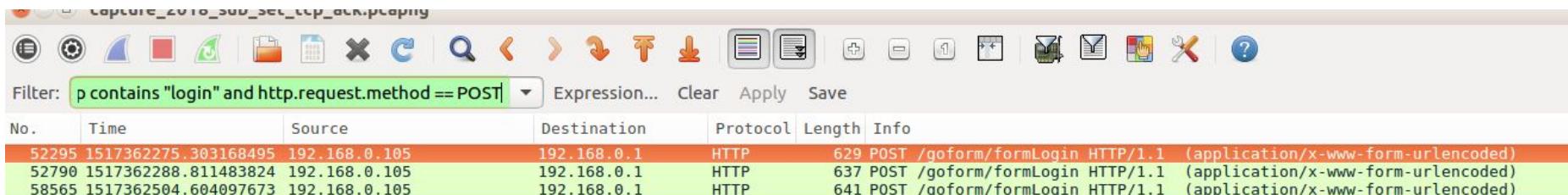
Since the filter return a nonempty results, then the HTTP contains a login keyword. But which packet??

| Filter: i >= 1517362098.076674968 and http contains "login" ▾ Expression... Clear Apply Save |                      |               |               |          |        |   |
|--|----------------------|---------------|---------------|----------|--------|---|
| No.  | Time                 | Source        | Destination   | Protocol | Length | Info  |
| 50703  | 1517362255.667692956 | 192.168.0.1   | 192.168.0.105 | HTTP     | 1299   | HTTP/1.1 200 OK (text/plain)  |
| 51029  | 1517362255.931557349 | 192.168.0.1   | 192.168.0.105 | HTTP     | 838    | HTTP/1.1 200 OK (text/plain)  |
| 51077  | 1517362256.167388501 | 192.168.0.1   | 192.168.0.105 | HTTP     | 351    | HTTP/1.1 200 OK (text/html)   |
| 51973  | 1517362256.655918629 | 192.168.0.1   | 192.168.0.105 | HTTP     | 838    | HTTP/1.1 200 OK (text/plain)  |
| 52295  | 1517362275.303168495 | 192.168.0.105 | 192.168.0.1   | HTTP     | 629    | POST /goform/formLogin HTTP/1.1 (application/x-www-form-urlencoded) |
| 52357  | 1517362275.553080538 | 192.168.0.1   | 192.168.0.105 | HTTP     | 351    | HTTP/1.1 200 OK (text/html)   |
| 52441  | 1517362276.347162631 | 192.168.0.105 | 192.168.0.1   | HTTP     | 438    | GET /login_fail.asp HTTP/1.1  |
| 52457  | 1517362276.663696374 | 192.168.0.1   | 192.168.0.105 | HTTP     | 1014   | HTTP/1.1 200 OK (text/html)   |
| 52471  | 1517362276.706856473 | 192.168.0.105 | 192.168.0.1   | HTTP     | 397    | GET /comm.asp HTTP/1.1  |
| 52501  | 1517362277.026984322 | 192.168.0.105 | 192.168.0.1   | HTTP     | 397    | GET /comm.asp HTTP/1.1  |
| 52503  | 1517362277.027128556 | 192.168.0.105 | 192.168.0.1   | HTTP     | 355    | GET /language_pack.js HTTP/1.1                                      |

# DPI with Wireshark

Most likely login data are sent over HTTP post method. So maybe we could update the filter to look for HTTP post

frame.time\_epoch >= 1517362098.076674968 and http contains "login" and  
http.request.method == POST



A screenshot of the Wireshark interface. The title bar reads "Capture\_2018\_08\_26\_1108\_00.pcapng". The toolbar has various icons for file operations, zoom, and analysis. The main window shows a list of network captures. A green box highlights the "Filter:" field which contains the expression "p contains \"login\" and http.request.method == POST". Below the filter bar are buttons for "Expression...", "Clear", "Apply", and "Save". The table below has columns: No., Time, Source, Destination, Protocol, Length, and Info. Three rows of traffic are listed, all of which have their "Info" column highlighted in red. The first row shows a POST request to "192.168.0.1" with length 629 and content "(application/x-www-form-urlencoded)". The second row shows a POST request to "192.168.0.1" with length 637 and content "(application/x-www-form-urlencoded)". The third row shows a POST request to "192.168.0.1" with length 641 and content "(application/x-www-form-urlencoded)".

| No.   | Time                 | Source        | Destination | Protocol | Length | Info  |
|-------|----------------------|---------------|-------------|----------|--------|---|
| 52295 | 1517362275.303168495 | 192.168.0.105 | 192.168.0.1 | HTTP     | 629    | POST /goform/formLogin HTTP/1.1 (application/x-www-form-urlencoded) |
| 52790 | 1517362288.811483824 | 192.168.0.105 | 192.168.0.1 | HTTP     | 637    | POST /goform/formLogin HTTP/1.1 (application/x-www-form-urlencoded) |
| 58565 | 1517362504.604097673 | 192.168.0.105 | 192.168.0.1 | HTTP     | 641    | POST /goform/formLogin HTTP/1.1 (application/x-www-form-urlencoded) |

# DPI with Wireshark

If we investigate the filtered packets we can see that the victim was trying to login to the network router with an admin account

```
[Full request URI: http://192.168.0.1/goform/formLogin]
[HTTP request 1/1]
[Response in frame: 52325]
File Data: 118 bytes
▼ HTML Form URL Encoded: application/x-www-form-urlencoded
▶ Form item: "login_name" = ""
▶ Form item: "curTime" = "1517362275195"
▶ Form item: "FILECODE" = ""
▶ Form item: "VERIFICATION_CODE" = ""
▶ Form item: "login_n" = "admin"
▶ Form item: "login_pass" = "aGVsbG93b3JsZA=="
▶ Form item: "VER_CODE" = ""

0000  08 00 27 9b b5 04 08 00  27 4f 41 8d 08 00 45 00 .
0010  02 67 5a 8e 40 00 40 06  5c 48 c0 a8 00 69 c0 a8 .|
0020  00 01 c6 5c 00 50 b0 ed  3b 34 c1 62 8e 38 80 18 .
0030  00 e5 8e 20 00 00 01 01  08 0a b1 ff a0 2f 03 58 .
0040  a9 9f 50 4f 53 54 20 2f  67 6f 66 6f 72 6d 2f 66 .
0050  6f 72 6d 4c 6f 67 69 6e  20 48 54 54 50 2f 31 2e o
0060  .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. .. ..
```

# Deep Packet Inspection

What did we learn?

What are the challenges in DPI?

Could we automate the DPI process?

Could we use machine learning to improve DPI?

# Network Vulnerability Assessment

# What is Network Vulnerability Assessment?

The process of identifying and classifying any security vulnerability in computer networks or other communication systems.

It is the process of scanning, identifying and analyzing potential security holes.

It is an iterative process that should usually be conducted on a regular basis.

It is required when new software or hardware added to the network. Even with a software update, it is recommended to apply vulnerability assessment.

Network vulnerability could be an **active** or **passive** process.

# Passive Scanners

Passive Scanners will scan the network under investigation to collect and gather information about network nodes, running services, and network topology.

The scanner will analyze the collected data to identify potential security weakness.

Passive scanners use one or more network vulnerability database to identify the potential vulnerability.

It is essential to update network vulnerability databases otherwise the scanner will be outdated and provide inaccurate scanning results.

# Active Scanners

Active Scanners just like passive scanner they will scan the network under investigation to collect and gather information about network nodes, running services, and network topology.

The scanner will analyze the collected data to identify potential security weakness.

Based on the identified weakness the active scanner will try to exploit the identified weakness by attacking the network and monitor the network status.

Active scanners not only report potential vulnerability but also attempt to verify the vulnerability by trying to exploit discovered vulnerability.

# Limitations and Issues with Network Scanners

Both active and passive scanner could produce false positive and false negative scan due to many reasons and they **can not** detect **zero-day exploit**

A **false positive** scan means a nonvulnerable network asset( server, router, etc.) identified as vulnerable

A **false negative** scan means a vulnerable network asset ( server, router, etc.) identified as secure

Both passive and active scan affect the network performance. In addition, the active scan could result in denial of services, disrupt regular network operations.

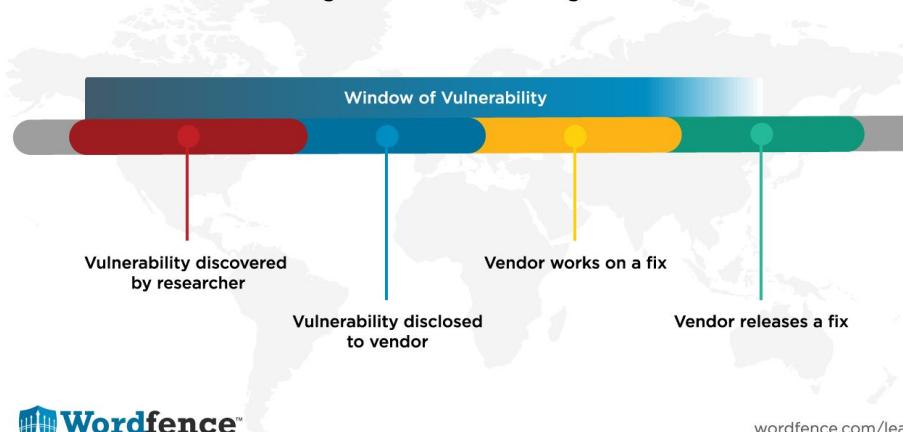
Qualified network **security engineers** need to review scanning results to detect potential false positive and false negative

# Zero-Day Exploit | Vulnerability

Is an attack that exploits a previously **unknown** security vulnerability.

A zero-day attack is also sometimes defined as an attack that takes advantage of a security vulnerability on the same day that the vulnerability becomes known.

**Zero Day Vulnerability Timeline**



# Nessus

Nessus is a **proprietary** vulnerability scanner developed by Tenable Network Security. It is **free of charge for personal use** in a non-enterprise environment.

Nessus is generally a **passive** vulnerability scanner

Nessus **scan** for:

1. Vulnerabilities that allow a remote hacker to control or access sensitive data on a system (e.g. weak and default password)
2. Misconfiguration
3. Denials of service against the TCP/IP stack by using malformed packets
4. Security compliance (e.g PCI DSS = Payment Card Industry Data Security Standard)

You could use **Nessus Attack Scripting Language (NASL)** to write exploit and attacks

# OpenVAS

OpenVAS is a framework of several services and tools offering a comprehensive and powerful vulnerability scanning.

OpenVAS originally was a fork of the last free version of Nessus (2005)

OpenVAS is by default an **active** network vulnerability scanner even with the **none destructive scan** option. It is possible to crash a network services.

OpenVAS known vulnerabilities and mis-configuration using a large database  
**Network Vulnerability Tests** (NVTs)

# Using Nessus with Kali Linux

Go to <https://www.tenable.com/products/nessus-home> to register and download Nessus, you will receive an activation code by email

The screenshot shows the Tenable website at https://www.tenable.com/products/nessus-home. The page has a teal header with the Tenable logo and navigation links for Cyber Exposure, Products, Services, Company, Partners, and Blog. A 'Login' and 'Try/Buy' button are also visible. The main content area features a large teal background with a wavy pattern and the text 'Nessus Home'. Below this, there is descriptive text about Nessus Home, a note about support availability, and a note about commercial use. On the right side, there is a form titled 'Register for an Activation Code' with fields for First Name, Last Name, Email, and a checkbox for receiving updates. A 'Register' button is at the bottom of the form.

Nessus® Home allows you to scan your personal home network (up to 16 IP addresses per scanner) with the same high-speed, in-depth assessments and agentless scanning convenience that Nessus subscribers enjoy.

Please note that Nessus Home does not provide access to support, allow you to perform compliance checks or content audits, or allow you to use the Nessus virtual appliance. If you require support and these [additional features](#), please purchase a [Nessus](#) subscription.

Nessus Home is available for personal use in a home environment only. It is not for use by any commercial organization.

**Register for an Activation Code**

First Name \*  Last Name \*   
Email \*   
 Check to receive updates from Tenable

**Register**

# Download the version that match your OS

## Nessus - 7.0.1

### Release Date

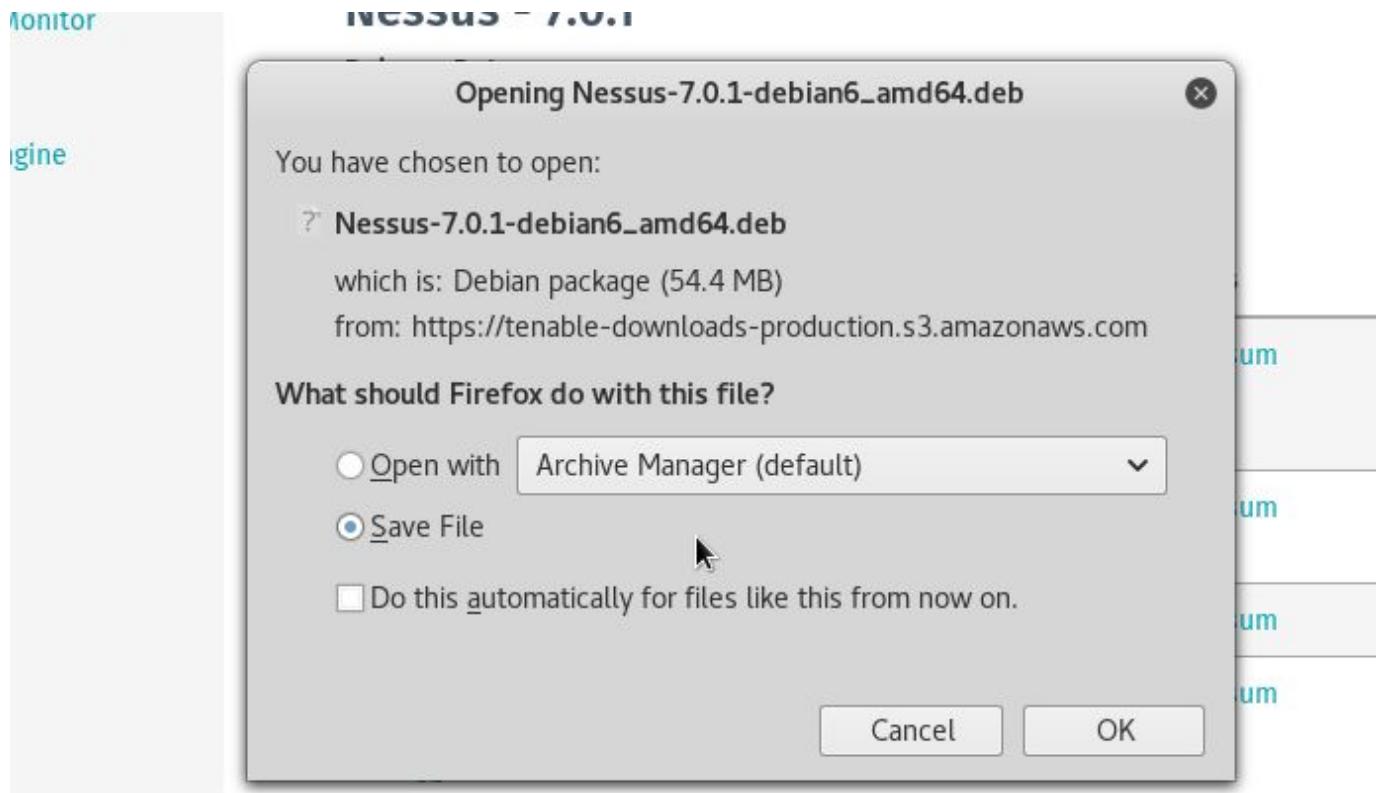
01/11/2018

### Release Notes:

[Nessus 7.0.1](#)

| Name   | Description   | Details                  |
|--|---|--------------------------|
| <a href="#"> Nessus-7.0.1-x64.msi</a>           | Windows Server 2008, Server 2008 R2*, Server 2012, Server 2012 R2, 7, 8, 10, Server 2016 (64-bit) | <a href="#">Checksum</a> |
| <a href="#"> Nessus-7.0.1-Win32.msi</a>         | Windows 7, 8, 10 (32-bit)   | <a href="#">Checksum</a> |
| <a href="#"> Nessus-7.0.1.dmg</a>               | macOS (10.8 - 10.13)  | <a href="#">Checksum</a> |
| <a href="#"> Nessus-7.0.1-debian6_amd64.deb</a> | Debian 6, 7, 8 / Kali Linux 1 AMD64   | <a href="#">Checksum</a> |

# Save the deb file to your disk



# Make sure you received the activation code

Tenable Nessus Home Activation Code

 Nessus Registration <noreply@nessus.org> 5:51 PM (21 minutes ago)  
to shsaad Show details Inbox Reply More

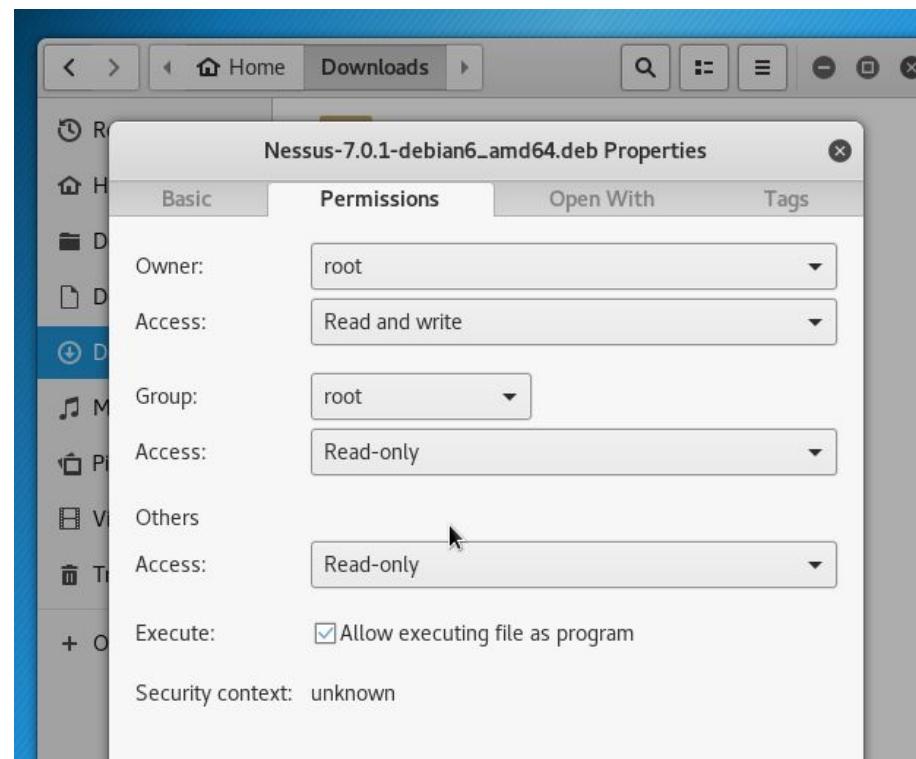
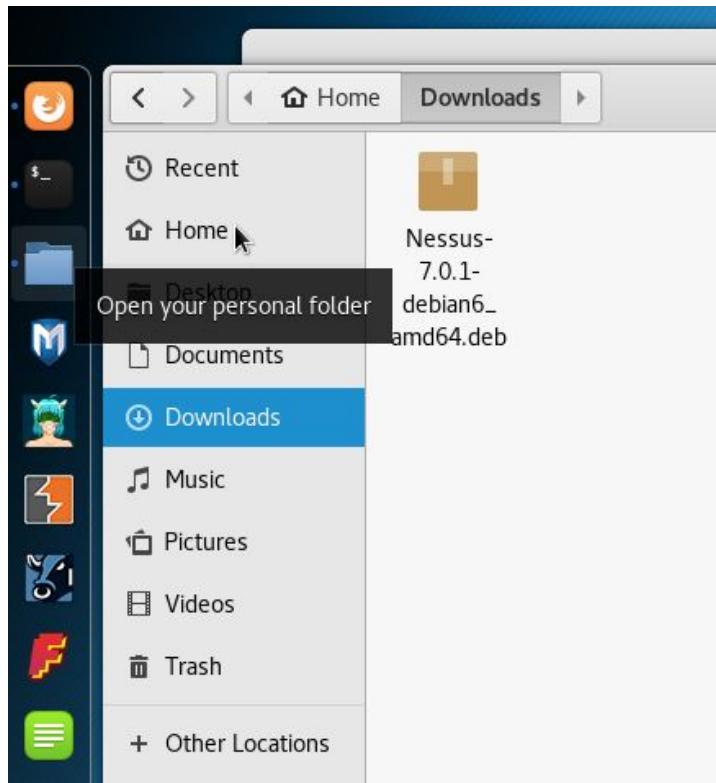
Thank you for registering your Nessus scanner with Tenable. The Nessus Home subscription will keep your Nessus scanner up to date with the latest plugins for vulnerability scanning.

(Note: If you use Nessus in a professional capacity, you need a Nessus subscription.)

Your activation code for the Nessus Home is  
B69F-A33B-23D9-5372-FB75

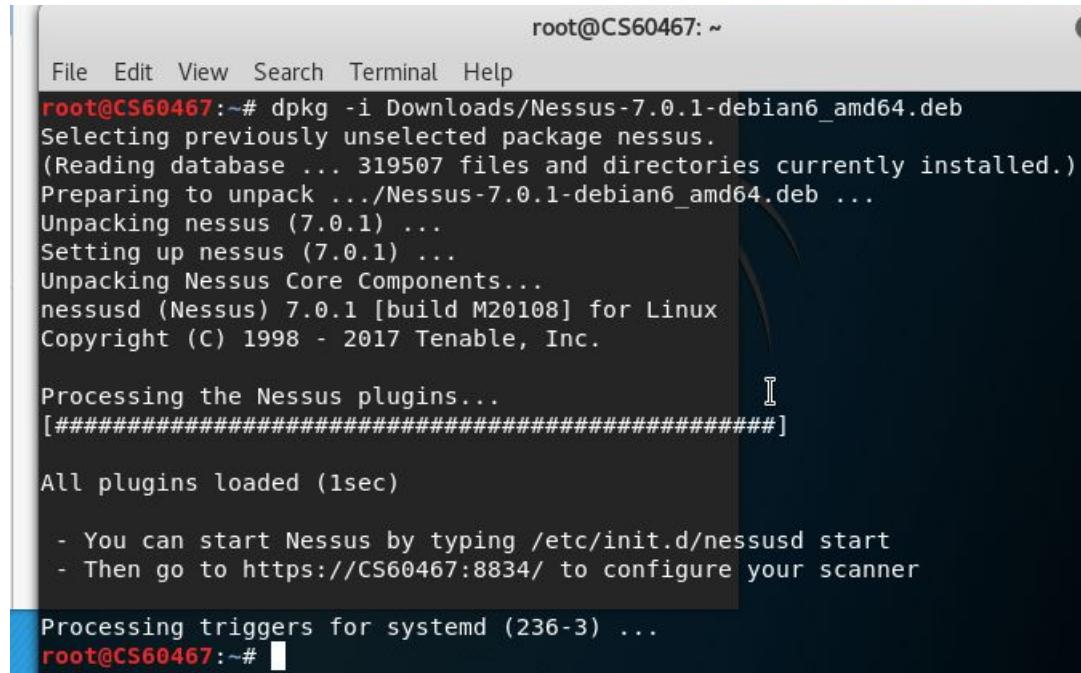
This is a one-time code. If you un-install and then re-install Nessus, you will need to register the scanner again and receive another Activation Code.

# Change the permission on the deb file



# Open the command line and run the installer

**dpkg -i Downloads/Nessus-7.0.1-debian6\_amd64.deb**



The screenshot shows a terminal window with a light gray header bar containing the text "root@CS60467: ~". Below the header is a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The main area of the terminal displays the output of the command "dpkg -i Downloads/Nessus-7.0.1-debian6\_amd64.deb". The output includes:

```
root@CS60467:~# dpkg -i Downloads/Nessus-7.0.1-debian6_amd64.deb
Selecting previously unselected package nessus.
(Reading database ... 319507 files and directories currently installed.)
Preparing to unpack .../Nessus-7.0.1-debian6_amd64.deb ...
Unpacking nessus (7.0.1) ...
Setting up nessus (7.0.1) ...
Unpacking Nessus Core Components...
nessusd (Nessus) 7.0.1 [build M20108] for Linux
Copyright (C) 1998 - 2017 Tenable, Inc.

Processing the Nessus plugins...
[#####
All plugins loaded (1sec)

- You can start Nessus by typing /etc/init.d/nessusd start
- Then go to https://CS60467:8834/ to configure your scanner

Processing triggers for systemd (236-3) ...
root@CS60467:~#
```

# Start the Nessus Server

**/etc/init.d/nessusd start**

**Then open your process and go to <https://<hostname>:8834>**

```
Processing the Nessus plugins...
[#####
All plugins loaded (1sec)

- You can start Nessus by typing /etc/init.d/nessusd start
- Then go to https://CS60467:8834/ to configure your scanner

Processing triggers for systemd (236-3) ...
root@CS60467:~# ^C
root@CS60467:~# /etc/init.d/nessusd start
Starting Nessus : .
root@CS60467:~#
```

Nessus Home | Nessus™

Trump to have "new America" moment

DAI NEWS

# Create and Account , Register, and Activate

The screenshot shows a web-based account creation interface for the Nessus scanner. At the top left, it says "STEP 1 OF 3". In the top right corner is the Nessus logo, which consists of a green hexagon with a white stylized 'N' inside. Below the logo, the word "Nessus" is written in a green sans-serif font. The main title "Create an account" is centered above a descriptive text block. This text explains that to use the scanner, an account must be created, and that this account can execute commands on remote targets and should be treated as a root user. Below this text are two input fields: one for "Username \*" and one for "Password \*". Both fields have a light gray placeholder text inside them. To the right of the password field is a small icon of an eye, which typically indicates a "show password" or "password visibility" function. At the bottom center of the form is a large green button labeled "Continue". At the very bottom of the page, outside the main form area, is a copyright notice: "© 2018 Tenable™, Inc."

STEP 1 OF 3

Nessus™

Create an account

To use this scanner, an account must be created.  
This account can execute commands on remote  
targets and should be treated as a root user.

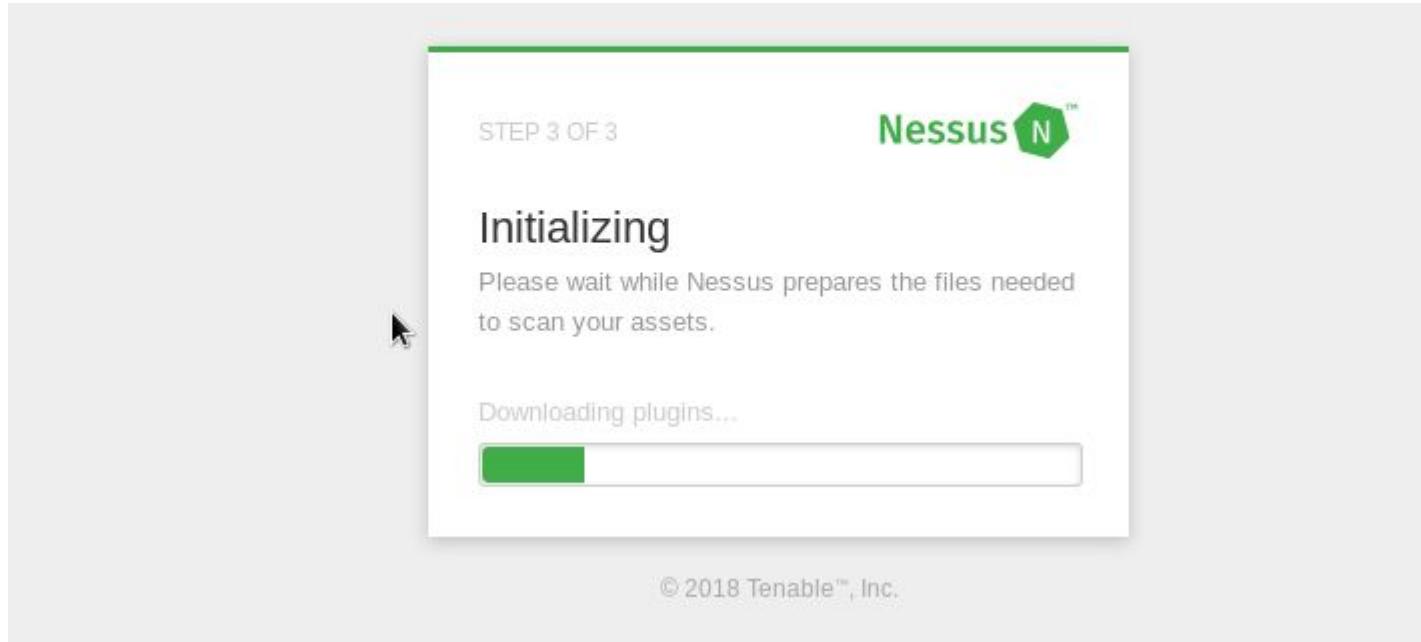
Username \*

Password \*

Continue

© 2018 Tenable™, Inc.

# Create and Account , Register, and Activate

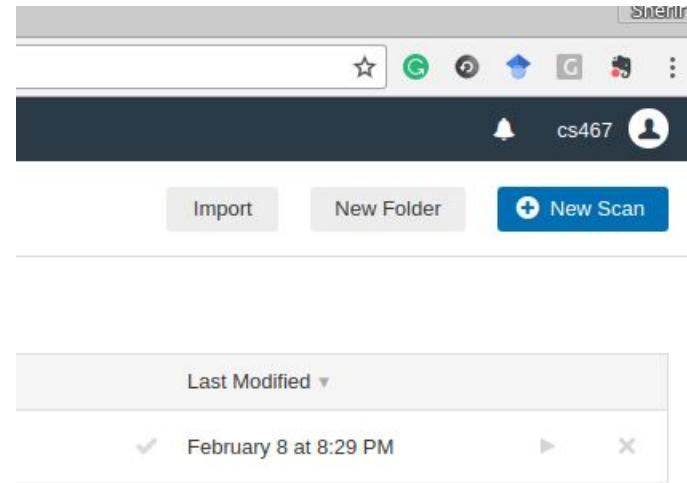


# Nessus Dashboard

The screenshot shows the Nessus Dashboard interface. The top navigation bar includes the Nessus logo, 'Scans' and 'Settings' links, and a user profile icon with the identifier 'cs467'. The main content area is titled 'My Scans' and displays a table of one scan entry. The table columns are 'Name', 'Schedule', and 'Last Modified'. The single entry is 'my\_basic\_scan', scheduled 'On Demand', and last modified 'February 8 at 8:29 PM'. The left sidebar contains sections for 'FOLDERS' (My Scans, All Scans, Trash) and 'RESOURCES' (Policies, Plugin Rules, Scanners).

| Name          | Schedule  | Last Modified           |
|---------------|-----------|-------------------------|
| my_basic_scan | On Demand | ✓ February 8 at 8:29 PM |

# Nessus Create a Scan



# Select the Type of the Scan

Scanner Search Library

|   |  |   |   |   |   |   |
|---|--|---|---|---|---|---|
| <br><b>Advanced Scan</b><br>Configure a scan without using any recommendations.  | <br><b>Audit Cloud Infrastructure</b><br>Audit the configuration of third-party cloud services.<br><small>UPGRADE</small> | <br><b>Badlock Detection</b><br>Remote and local checks for CVE-2016-2118 and CVE-2016-0128.                               | <br><b>Bash Shellshock Detection</b><br>Remote and local checks for CVE-2014-6271 and CVE-2014-7169. | <br><b>Basic Network Scan</b><br>A full system scan suitable for any host.                                     | <br><b>Credentialed Patch Audit</b><br>Authenticate to hosts and enumerate missing updates.                        | <br><b>DROWN Detection</b><br>Remote checks for CVE-2016-0800.   |
| <br><b>Host Discovery</b><br>A simple scan to discover live hosts and open ports.  | <br><b>Intel AMT Security Bypass</b><br>Remote and local checks for CVE-2017-5689.  | <br><b>Internal PCI Network Scan</b><br>Perform an internal PCI DSS (11.2.1) vulnerability scan.<br><small>UPGRADE</small> | <br><b>Malware Scan</b><br>Scan for malware on Windows and Unix systems.                             | <br><b>MDM Config Audit</b><br>Audit the configuration of mobile device managers.<br><small>UPGRADE</small>    | <br><b>Mobile Device Scan</b><br>Assess mobile devices via Microsoft Exchange or an MDM.<br><small>UPGRADE</small> | <br><b>Offline Config Audit</b><br>Audit the configuration of network devices.<br><small>UPGRADE</small> |
| <br><b>PCI Quarterly External Scan</b><br>Approved for quarterly external scanning as required by PCI.<br><small>UPGRADE</small> | <br><b>Policy Compliance Auditing</b><br>Audit system configurations against a known baseline.<br><small>UPGRADE</small>  | <br><b>SCAP and OVAL Auditing</b><br>Audit systems using SCAP and OVAL definitions.<br><small>UPGRADE</small>              | <br><b>Shadow Brokers Scan</b><br>Scan for vulnerabilities disclosed in the Shadow Brokers leaks.    | <br><b>Spectre and Meltdown</b><br>Remote and local checks for CVE-2017-5753, CVE-2017-5715, and CVE-2017-5754 | <br><b>WannaCry Ransomware</b><br>Remote and local checks for MS17-010.  | <br><b>Web Application Tests</b><br>Scan for published and unknown web vulnerabilities.                  |

# Select Basic Network Scan

## New Scan / Basic Network Scan

[Back to Scan Templates](#)

Basic Network Scan

A full system scan suitable for any host.

CI

€

UPGRADE

Basic

General

Schedule

Notifications

Discovery

Assessment

Report

Advanced

Name

Description

Folder

My Scans

Targets

Example: 192.168.1.1-192.168.1.5, 192.168.2.0/24, test.com

Upload Targets

Add File

Save

Cancel

# Edit your Scan info and save it

The screenshot shows the Nessus web interface with the following details:

- Header:** Nessus N™, Scans, Settings
- Left Sidebar (Folders):** My Scans, All Scans, Trash
- Left Sidebar (Resources):** Policies, Plugin Rules, Scanners
- Page Title:** cs scan / Configuration
- Back Link:** < Back to Scan Report
- Tab Bar:** Settings (selected), Credentials, Plugins
- Form Fields (Basic):**
  - Name: cs scan
  - Description: This a simple test scan
  - Folder: My Scans
  - Targets: 137.207.140.164
- Buttons:** Save, Cancel

# Select and Launch your Scan

The screenshot shows the Nessus web interface with the following details:

- Header:** Nessus N, Scans, Settings, cs467, Notifications icon.
- Folders:** My Scans (selected), All Scans, Trash.
- Resources:** Policies, Plugin Rules, Scanners.
- Search Bar:** Search Scans, 2 Scans (1 Selected) Clear Selected Item.
- Table:** My Scans

| Name          | Schedule  | Last Modified         | Actions                       |
|---------------|-----------|-----------------------|-------------------------------|
| cs scan       | On Demand | Today at 11:30 AM     | <span>▶</span> <span>X</span> |
| my_basic_scan | On Demand | February 8 at 8:29 PM | <span>▶</span> <span>X</span> |

# Wait until the scan complete

my\_basic\_scan  
[Back to My Scans](#)

Hosts 1   Vulnerabilities 17   History 2

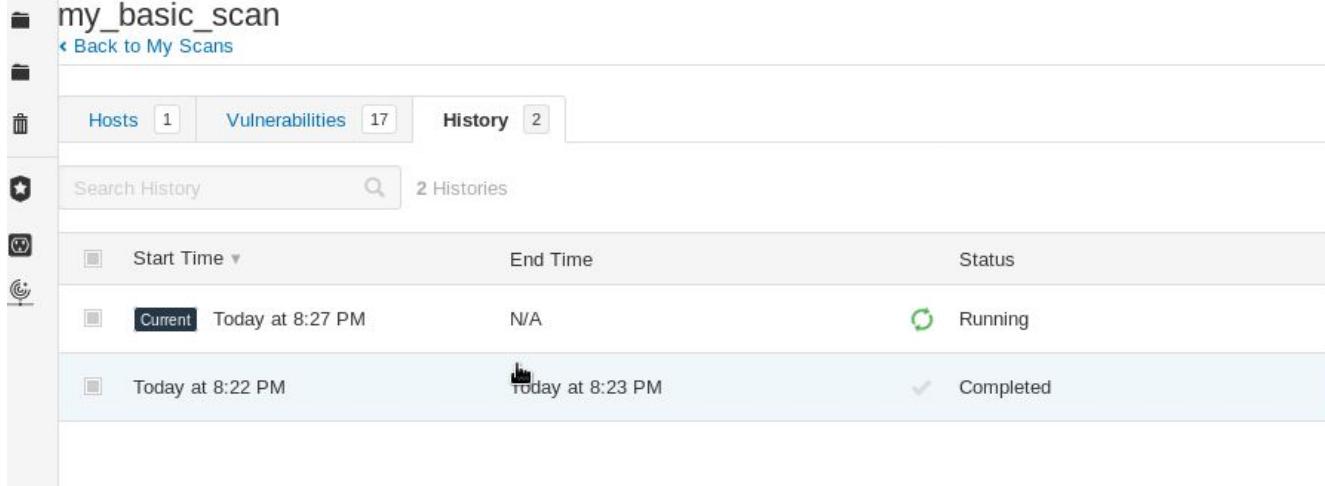
Search History  2 Histories

| Start Time               | End Time         | Status  |
|--------------------------|------------------|---|
| Current Today at 8:27 PM | N/A              |  Running   |
| Today at 8:22 PM         | Today at 8:23 PM |  Completed |

**Scan Details**

Name:  
Status:  
Policy:  
Scanner:  
Start:

**Vulnerabilities**

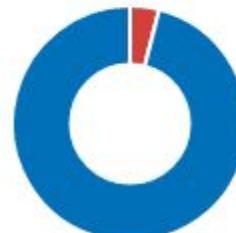


# When the Nessus Complete the Scan

| Scan Details |                    |
|--------------|--------------------|
| Name:        | my_basic_scan      |
| Status:      | Completed          |
| Policy:      | Basic Network Scan |
| Scanner:     | Local Scanner      |
| Start:       | Today at 8:27 PM   |
| End:         | Today at 8:29 PM   |
| Elapsed:     | 2 minutes          |



## Vulnerabilities



- Critical
- High
- Medium
- Low
- Info

# You can view the scanning result or export it

my\_basic\_scan

[Back to My Scans](#)

Configure Audit Tr...

Hosts 1 Vulnerabilities 24 History 2

Filter ▾ Search Vulnerabilities 24 Vulnerabilities

| Sev ▾    | Name ▲  | Plugin ID: 50989  | Family ▲ | Count ▾ |
|----------|---|-------------------|----------|---------|
| CRITICAL | ProFTPD Compromised Source Packages Trojaned Distribution | FTP               | 1        |         |
| INFO     | Nessus SYN scanner  | Port scanners     | 3        |         |
| INFO     | Service Detection   | Service detection | 3        |         |
| INFO     | Apache Banner Linux Distribution Disclosure               | Web Servers       | 1        |         |
| INFO     | Apache HTTP Server Version                                | Web Servers       | 1        |         |
| INFO     | Backported Security Patch Detection (SSH)                 | General           | 1        |         |
| INFO     | Backported Security Patch Detection (WWW)                 | General           | 1        |         |

Scan De

Name: Status: Policy: Scanner: Start: End: Elapsed:

Vulneral



# Nessus report the problem and suggest solution

Hosts 1    Vulnerabilities 24    History 2

**CRITICAL** ProFTPD Compromised Source Packages Trojaned Distribution >

**Description**  
The remote host is using ProFTPD, a free FTP server for Unix and Linux.

The version of ProFTPD installed on the remote host has been compiled with a backdoor in 'src/help.c', apparently related to a compromise of the main distribution server for the ProFTPD project on the 28th of November 2010 around 20:00 UTC and not addressed until the 2nd of December 2010.

By sending a special HELP command, an unauthenticated, remote attacker can gain a shell and execute arbitrary commands with system privileges.

Note that the compromised distribution file also contained code that ran as part of the initial configuration step and sent a special HTTP request to a server in Saudi Arabia. If this install was built from source, you should assume that the author of the backdoor is already aware of it.

**Solution**  
Reinstall the host from known, good sources.

**See Also**  
[http://www.theregister.co.uk/2010/12/02/proftpd\\_backdoored/](http://www.theregister.co.uk/2010/12/02/proftpd_backdoored/)  
<http://x0r.wordpress.com/2010/12/02/news-proftpd-owned-and-backdoored/>

**Plugin Details**

Severity:  
ID:  
Version:  
Type:  
Family:  
Published:  
Modified:

**Risk Information**

Risk Factor: C  
CVSS Base Score:  
CVSS Temporal Score:  
CVSS Vector: /I:C/A:C  
CVSS Temporal Vector:

# Questions