



Getting started **Wireshark**

Michael Muratov, Darien Ho, Joshua Mazariegos

Wireshark

Cross-platform, open-source Packet Analyzer written in C, C++.

Used for monitoring networks, troubleshooting, and testing security.

Available on many operating systems (including Linux, Mac, & Windows).

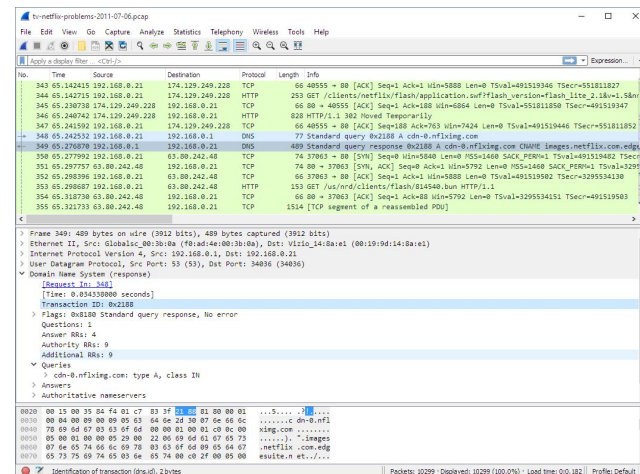
TShark

```
Command Prompt

c:\Program Files\Wireshark>tshark -i 9
Capturing on 'Ethernet'
1 0.800000 192.168.1.102 → 192.168.1.102 Spanning-tree (for-bridges)_00 60 STP Conf. Root = 327
68/0/10:da:43:22:ff:75 Cost = 0 Port = 0x8003
2 0.090870 192.168.1.102 → 239.255.255.250 369 SSDP NOTIFY * HTTP/1.1
3 1.009070 192.168.1.102 → 239.255.255.250 369 SSDP NOTIFY * HTTP/1.1
4 1.203623 192.168.1.102 → 239.255.255.250 378 SSDP NOTIFY * HTTP/1.1
5 1.307914 192.168.1.102 → 239.255.255.250 378 SSDP NOTIFY * HTTP/1.1
6 1.411687 192.168.1.102 → 239.255.255.250 433 SSDP NOTIFY * HTTP/1.1
7 1.515816 192.168.1.102 → 239.255.255.250 433 SSDP NOTIFY * HTTP/1.1
8 1.622650 192.168.1.102 → 239.255.255.250 443 SSDP NOTIFY * HTTP/1.1
9 1.633317 192.168.1.116 → 192.168.1.103 169 AJP13 1 AJP13 Error?
10 1.641753 192.168.1.103 → 192.168.1.116 169 TCP 1 [TCP segment of a reassembled PDU]
11 1.695662 192.168.1.116 → 192.168.1.103 54 TCP 116 51636→8009 [ACK] Seq=116 Ack=116 W
in=254 [TCP CHECKSUM INCORRECT] Len=0
12 1.728618 192.168.1.102 → 239.255.255.250 443 SSDP NOTIFY * HTTP/1.1
13 1.996566 12:da:43:22:ff:75 → Spanning-tree (for-bridges)_00 60 STP Conf. Root = 327
68/0/10:da:43:22:ff:75 Cost = 0 Port = 0x8003
14 3.996472 12:da:43:22:ff:75 → Spanning-tree (for-bridges)_00 60 STP Conf. Root = 327
68/0/10:da:43:22:ff:75 Cost = 0 Port = 0x8003
14 packets captured

c:\Program Files\Wireshark>
```

Wireshark



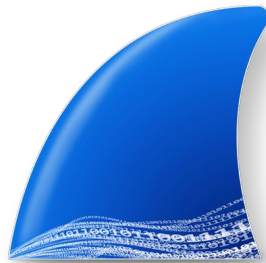
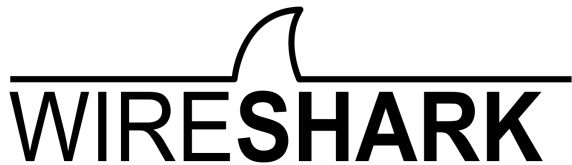
Brief History

Invented by Gerald Combs in the 1990s.

Open sourced.

Released in 1998 as Ethereal.

Heavily extended by the cyber security community.



- v0.99:(2006)
 - TCP, UDP, SSL streams
 - SSL decryption support
 - I/O Graphs
 - AirPcap support
- V1.0 (2009)
 - Usability bug fixes
- V1.2 (2010)
 - Protocol support expansion
- V1.4 (2011)
 - Protocol support expansion
- V1.6 (2013)
 - SSL session key export
- V2.0 (2017)
 - Update to statistics capabilities
- V3.0 (2019)
 - IP Map

Source:

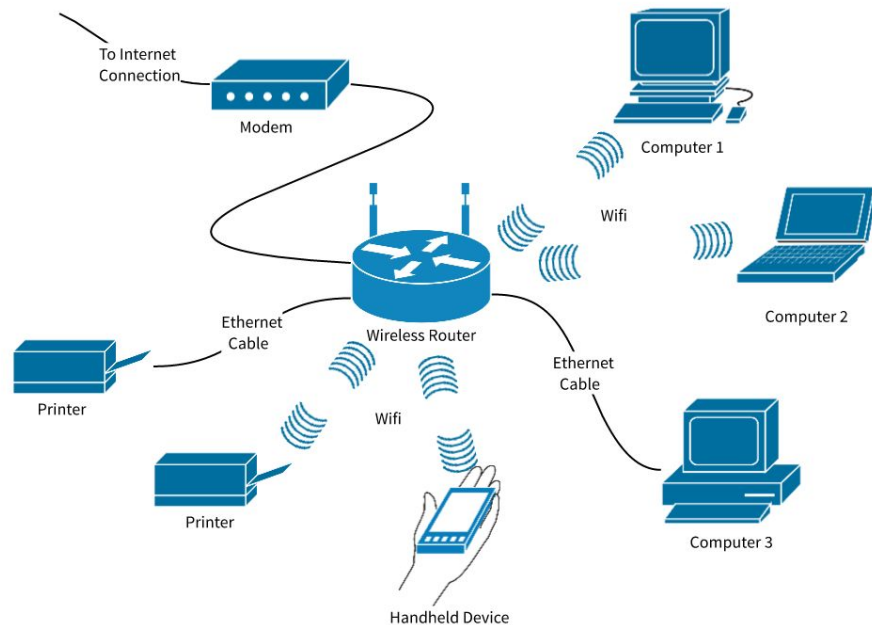
<https://www.wireshark.org/docs/relnotes/>

Wireshark

Captures packets going through a network.

Similar to tcpdump with a GUI and extended toolset:

- Searching/filtering packets
- Promiscuous mode
- Graphs
- Bluetooth traffic
- USB traffic
- Usage statistics
- ...



Source:

<https://www.lucidchart.com/pages/templates/net-work-diagram/wireless-network-diagram-template>



Wireshark

Tools & Features

Tools & Features

Capabilities of Wireshark

Capture Packets

Monitor & promiscuous mode.

Filters

Control which packets are captured or displayed.

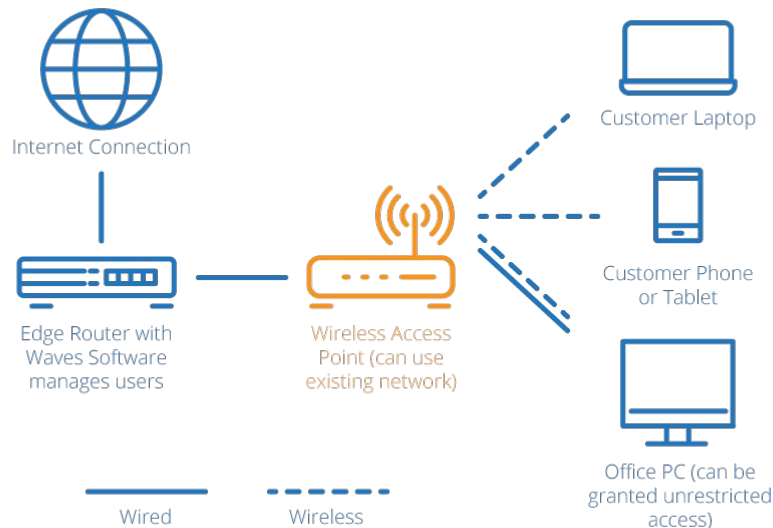
Analysis

On the toolbar, the Statistics menu contain many ways to analyze packets.

- Conversations—between two endpoints
 - Flow Graph—sequential streams
 - I/O Graph—packets over time
-

Monitor Mode

- In normal operation, devices on a wireless network ignore traffic not directed to it
- In monitor mode, all visible wireless traffic is processed
- Monitor mode is limited to wireless traffic only

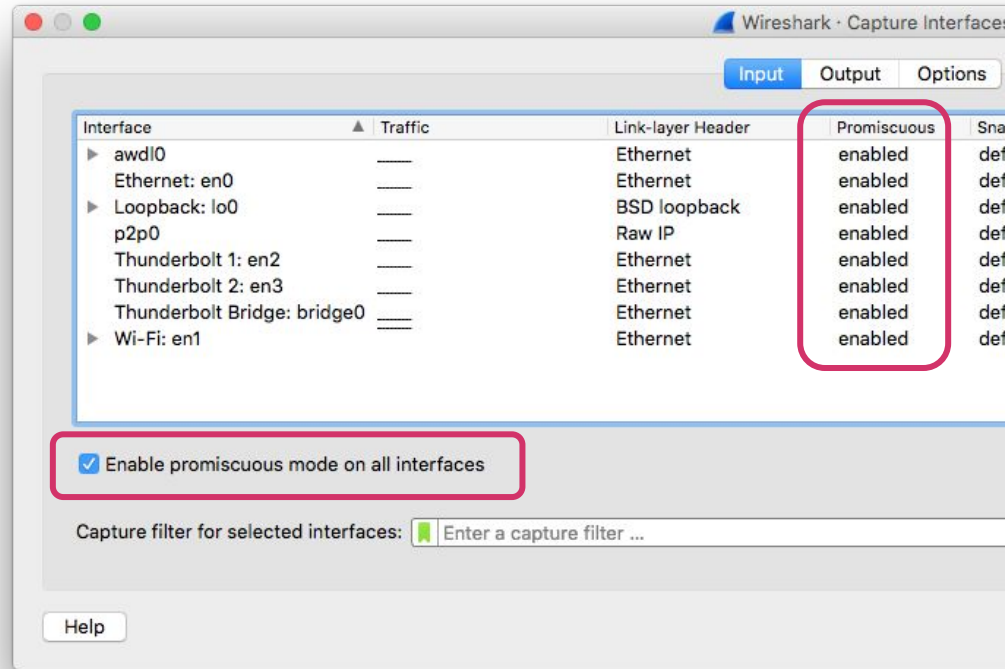


Source:

<https://www.waveswifi.com/how-does-waves-work>

Promiscuous Mode

- Captures all visible packets including those of other devices
- Requires support from your device's network interface card
- In non-promiscuous mode, devices can only read packets directed to them



Source:

<https://medium.com/@debookee/promiscuous-vs-monitoring-mode-d603601f5fa>

Filters

We might only be interested in a small subset of all network traffic. For instance, we might be interested in a particular:

- Source, Destination, Port, ...
- Request method (GET, POST, PUT, ...)
- Protocol (ARP, HTTP, TCP, UDP, ...)
- ...

We can filter packets during *capture* or during *display*.

251000 fields
3000 protocols



Capture Filters

Capture Filters

Only packets that satisfy these filters will be captured.

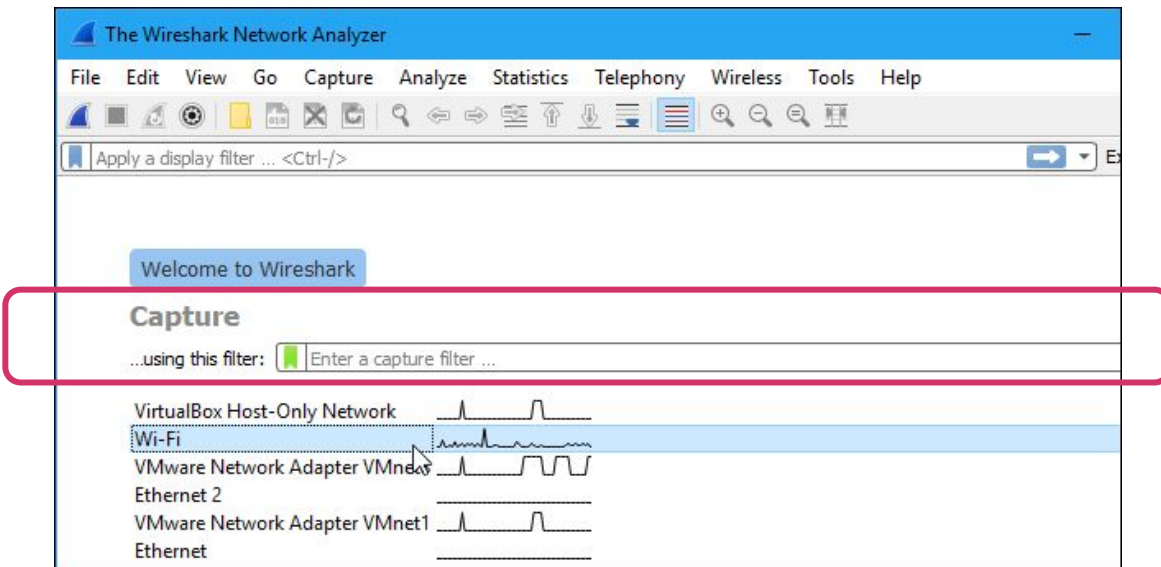
Example

host 192.168.1.1

Only traffic to or from 192.168.1.1

Reference

<https://wiki.wireshark.org/CaptureFilters>



Source:

<https://www.howtogeek.com/104278/how-to-use-wireshark-to-capture-filter-and-inspect-packets/>

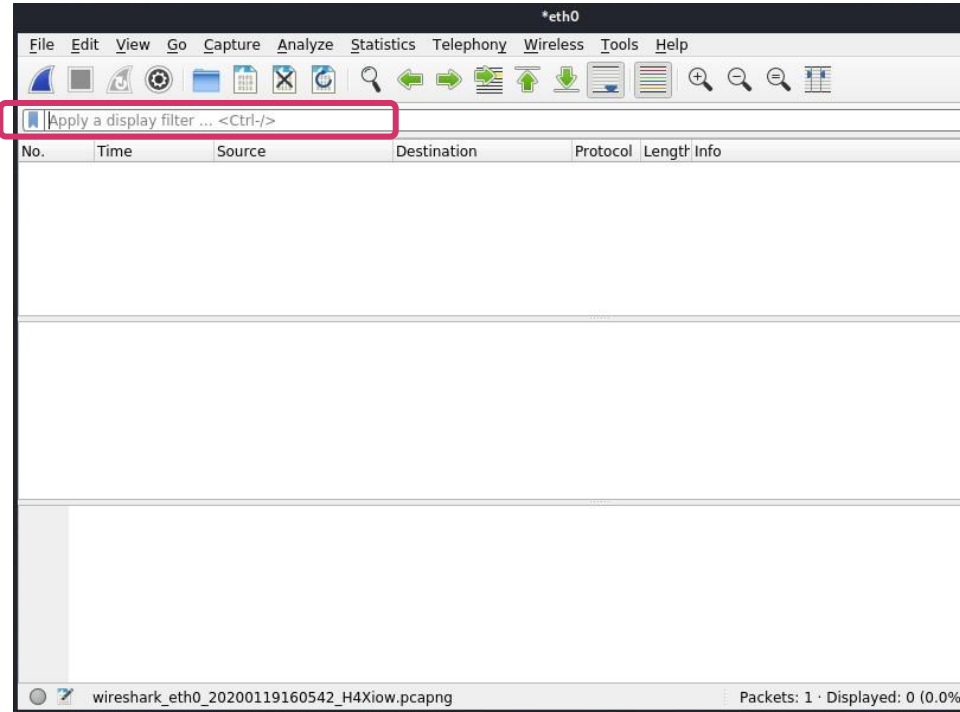
Display Filters

Display Filters

Controls which captured packets are displayed. Will not prevent packets from being captured.

Reference

<https://www.wireshark.org/docs/dfref/>



Statistics

Understanding network activity
and the flow of packets

Acquire insight by visualizing the
flow and relations of packets.

Tools emphasize different
properties of network traffic.

Packet Lengths

Statistics > Packet Lengths

Number of packets belonging to different size ranges.

Provides information about each group, such as:

- Average size
- Minimum/maximum size
- Distribution of packets

Topic / Item	Count	Average
▼ Packet Lengths	28219	1022.08
0-19	0	-
20-39	0	-
40-79	6252	67.22
80-159	5639	96.69
160-319	762	232.59
320-639	261	494.10
640-1279	242	874.24
1280-2559	12836	1502.86
2560-5119	1955	3115.56
5120 and greater	272	7268.69

Graphs

Statistics > I/O Graph

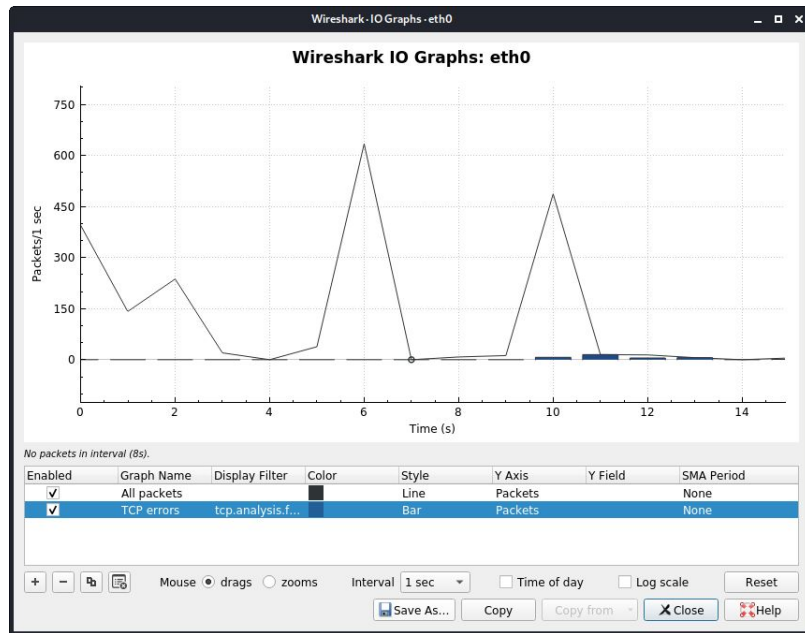
Visualize the flow of packets over time.

Customizable

- Plot type (line, bar, dot, ...)
- Use filters to choose which packets to plot
- Show plots of different filters in a single graph

Example: Load Balancer

Plot lines corresponding to traffic going to each server. This could expose unequal distribution of traffic.



Endpoints

Endpoints

The end of a specific protocol layer.
For example, an IP endpoint sends and receives traffic to/from a specific IP address.

Reference

<https://wiki.wireshark.org/Endpoint>

Statistics > Endpoints

Lists endpoints encountered in the captured traffic.

Example

- IPv4
- IPv6
- Ethernet
- 802.11

Conversations

Statistics > Conversations

Lists “conversations” between pairs of endpoints within the captured traffic.

Displays the amount of data sent between the two hosts and how long they were communicating.

Statistics > Flow Graph

Useful for visualizing the interaction between two hosts (e.g., handshake protocols).

Example Usage:

1. Right click a packet
2. Follow > TCP Stream
3. Statistics > Flow Graph
4. Flow Type: TCP Flows
5. Limit to display filter

The Big Picture

Drawing conclusions about
network activity

It can be difficult to make sense of
hundreds of thousands of packets.

Wireshark offers many tools for
filtering and analyzing important or
relevant traffic.



Wireshark **Applications**

Applications

Uses of Wireshark

Use Cases:

- Troubleshoot networks
- Analyze network traffic
- Develop network protocols
- Debug programs that uses networks
- Education - learning how networks work

Used By:

- System administrators
 - Security professionals
 - Software developers/engineers
-

Capture Network Traffic

- Type of request or response can be seen in the info tab
- Also shows the type of data that was sent or received
- Any data sent or received can be seen as well through the last tab that matches the type of data

Type of Request

URL for Request

Type of Sent / Received Data

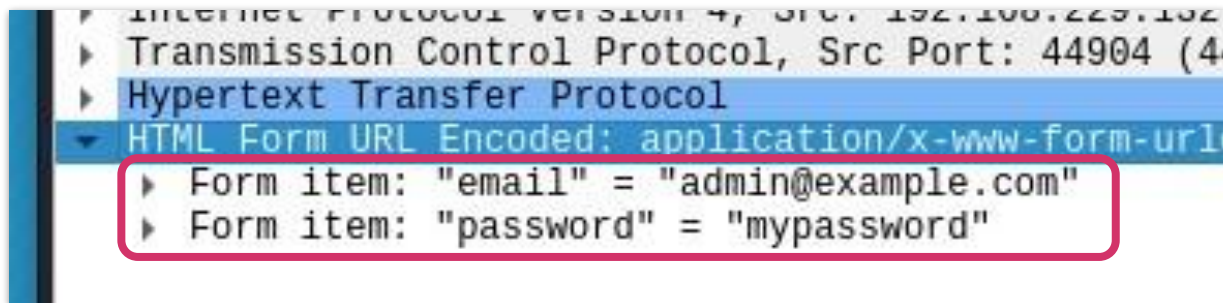
Tab that data is in

```
POST /index.php HTTP/1.1 (application/x-www-form-urlencoded)
HTTP/1.1 200 OK (text/html)
HTTP/1.1 200 OK
GET /R/A1MKIDcxM0FBQzk3Q0VD0DQxNTI4QjFBQTczM0FF0DEzM0Q2EgQCFwEgC
```

```
Frame 399: 742 bytes on wire (5936 bits), 742 bytes captured (5936 bits) on interface en0, id 0
Ethernet II, Src: Apple_52:c8:59 (a4:83:e7:52:c8:59), Dst: CASwell_ee:d9:c9 (08:35:71:ee:d9:c9)
Internet Protocol Version 4, Src: 138.51.154.125, Dst: 72.52.251.71
Transmission Control Protocol, Src Port: 60311, Dst Port: 80, Seq: 1, Ack: 1, Len: 676
Hypertext Transfer Protocol
HTML Form URL Encoded: application/x-www-form-urlencoded
```

HTTP Traffic

- No TLS/SSL layer used to encrypt traffic
- Packets might be transmitted as plaintext
- Eavesdroppers can capture these packets and view sensitive information



Plaintext credentials captured by Wireshark

Filter for HTTP/HTTPS Traffic

Display HTTP/HTTPS Traffic

- Filter for `http` to show traffic using HTTP
- Filter for specific ports:
 - `tcp.port==80` (Internet traffic)
 - `tcp.port==443` (HTTPS traffic)
- Filter for `ssl` or `tls` to find secure traffic

```
tcp.port == 443
```

```
tcp.port == 80
```

```
http
```

Filter by Domain Name

```
http.host=="example.com"
```

Filter by Request Method

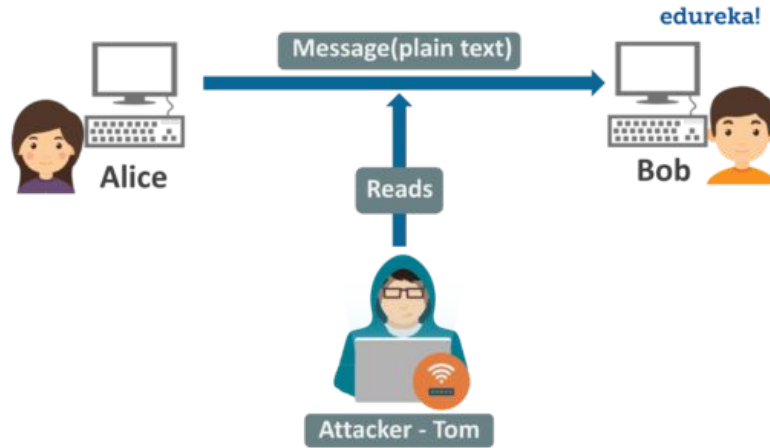
```
http.request.method=="GET"
```

Reference

https://www.wireshark.org/docs/wsug_html_chunked/ChWorkBuildDisplayFilterSection.html

Insecure Traffic

- Insecure packets are not properly encrypted
- Plaintext sent over a network can be intercepted and read
- Problematic for websites possessing private/sensitive user data



Source: <https://www.edureka.co/blog/what-is-network-security/>

Secure Traffic

- Secure packets contain encrypted data
- Data in secure traffic is protected from network analyzers
- HTTPS is used to provide a secure communication channel
 - Data is encrypted before transmission
 - Secure as long as the decryption key is not made available to the public

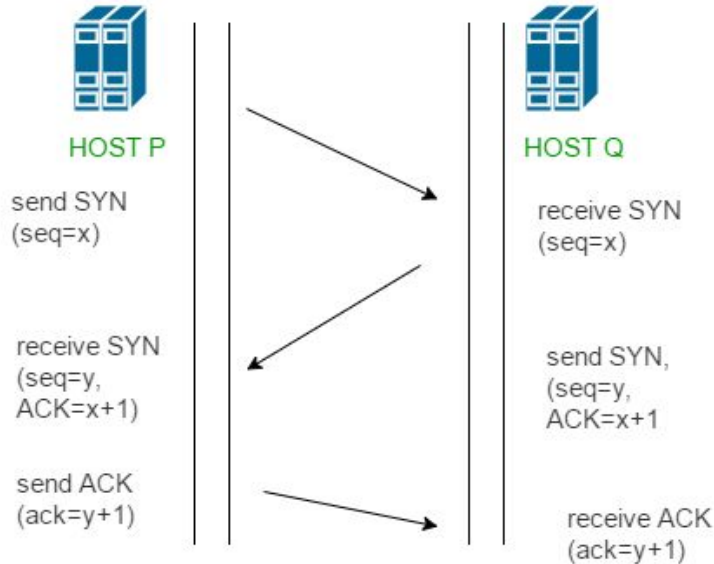
Decrypting Secure Traffic

When users want to see
human-readable data

HTTPS Decryption

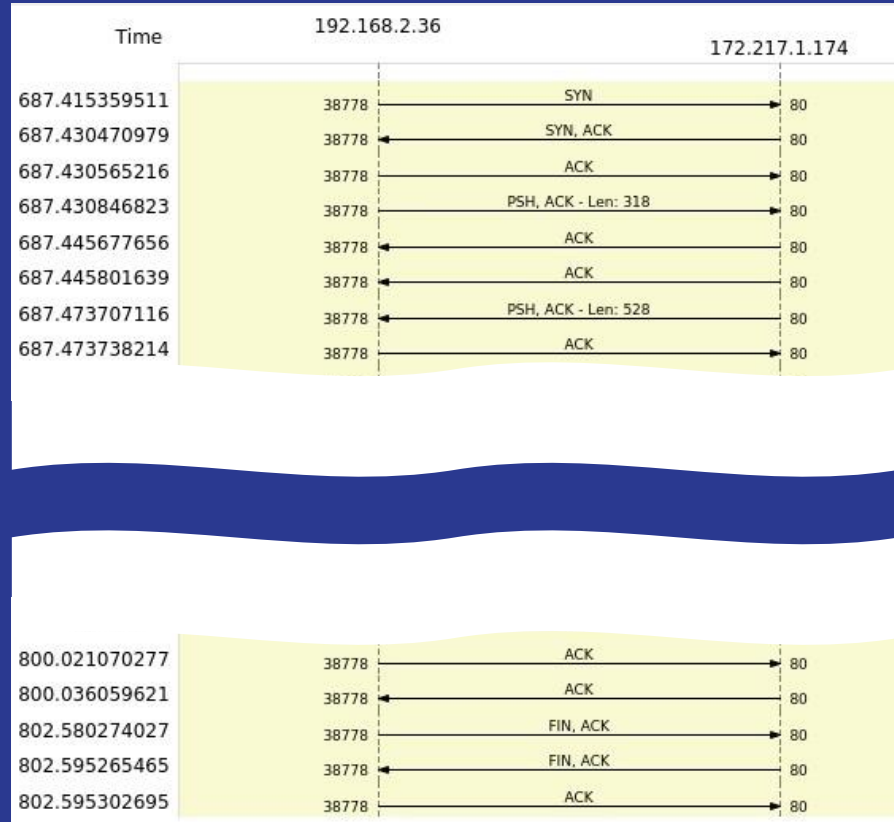
Cannot be done without the
corresponding private key. Those
who possess the decryption key
can provide it to Wireshark.

Handshake Protocol



Source: <https://www.geeksforgeeks.org/tcp-3-way-handshake-process/>

Statistics > Flow Graph



Network Troubleshooting

Understanding a network

Isolating Traffic

Statistics > Conversations

Details about traffic between two endpoints.

Troubleshooting

Analyze content and flow of packets

- HTTP headers, methods, parameters, etc.
 - HTML responses
 - TCP Errors
-

HTTPS Overview

HTTP and HTTPS

HyperText Transfer Protocol (HTTP)

A request-response protocol where client sends requests and receives responses from a server. Data could be sent as plaintext over the wire.

HyperText Transfer Protocol Secure (HTTPS)

A request-response protocol where TLS or SSL is layered over HTTP. Traffic is encrypted before transmission so that plaintext will not be intercepted.



Source: <https://serp.co/blog/http-vs-https/>

TLS and SSL

Transport Layer Security (TLS)

- Asymmetric encryption used to establish trust between client and server
- TLS Handshake generates a shared secret for symmetric encryption
- Trusted certificates contain public key to the host

Secure Sockets Layer (SSL)

- Predecessor of TLS
- Weaker encryption algorithms
- Susceptible to man-in-the-middle attacks
 - POODLE vulnerability - attackers were able to decrypt SSL traffic

Certificates

Certificates provide information about the organization and the certificate itself.

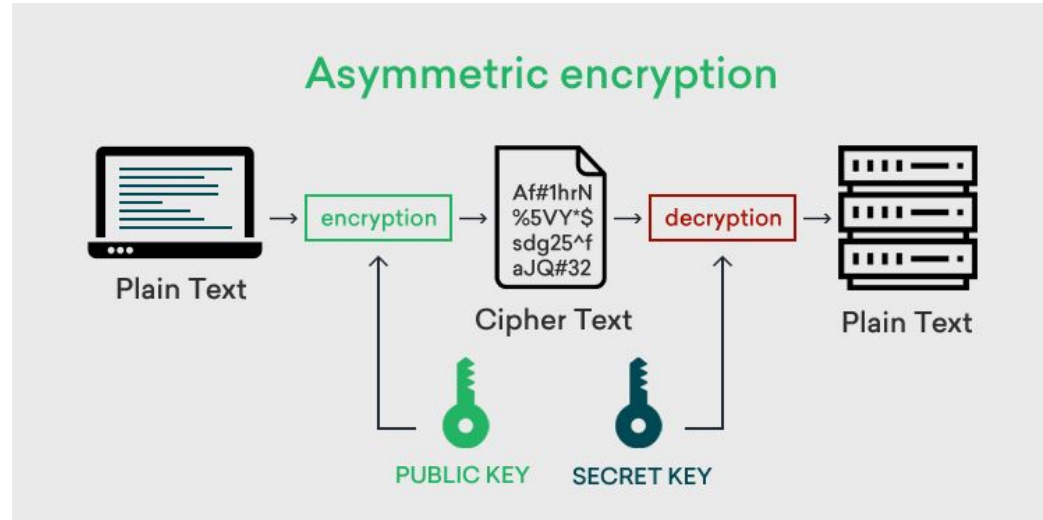
Users can use trusted certificates to establish a secure connection.



Source: <https://www.cloudflare.com/learning/ssl/why-is-http-not-secure/>

Certificates

- Issued by trusted certificate authorities (CA)
 - Browsers are often preinstalled with known CAs
- Contains a public key for encrypting traffic
- Server possesses the corresponding private key
- Eavesdropper can only capture encrypted traffic

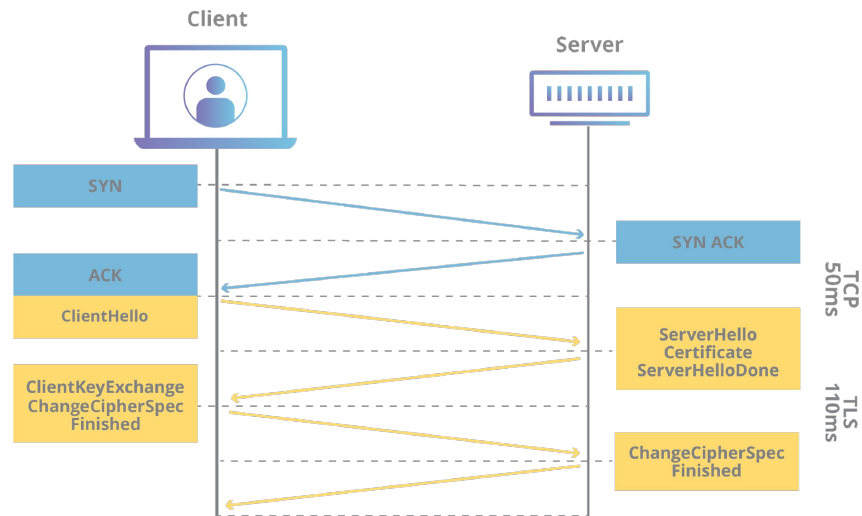


Source:

<https://www.clickssl.net/blog/symmetric-encryption-vs-asymmetric-encryption>

TLS Handshake

- First request-response messages act as “hello” to ensure they can communicate
- Then the next is for sending any information needed about the client’s or server’s TLS and the certificate
- Lastly it authenticates the certificate using the issuer and both do a test send to make sure that they are who they say they are then finish



Source:

<https://www.cloudflare.com/learning/ssl/what-happens-in-a-tls-handshake/>

HTTPS Traffic

- Wireshark can see HTTPS traffic but cannot expose plaintext on its own
- If the decryption key is provided, Wireshark can decrypt HTTPS traffic
 - Useful for developers or system administrators who want to see plaintext and not pseudorandom strings when debugging or troubleshooting
- Wireshark can still see parts of HTTPS packets like source/destination IP



Source: <https://medium.com/@kasunpdh/ssl-handshake-explained-4dabb87cdce>

Network Monitoring

Network Monitoring

Other Devices' Traffic

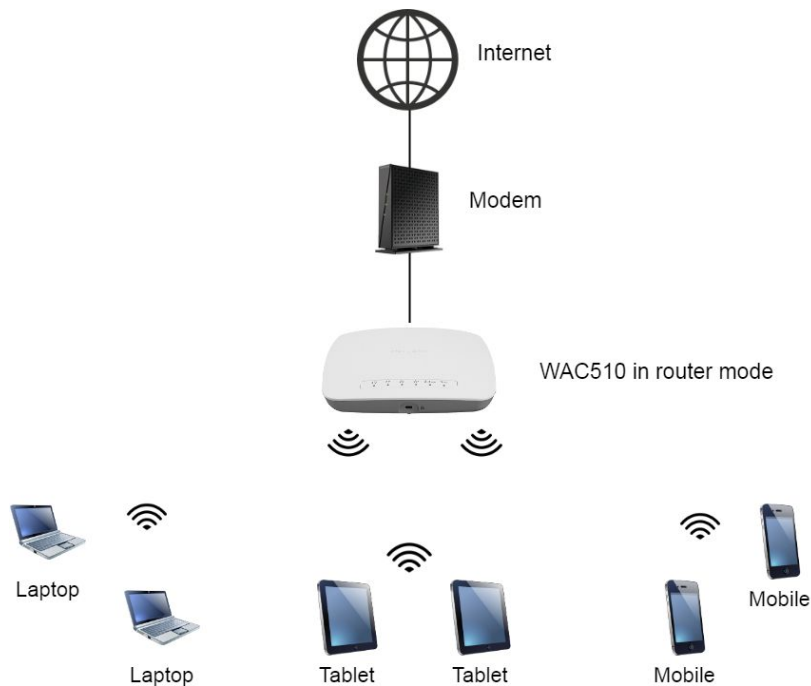
Capturing only your own device's traffic is not very interesting.

Promiscuous Mode

Capture traffic from other devices depending on device and network configuration.

Other Options

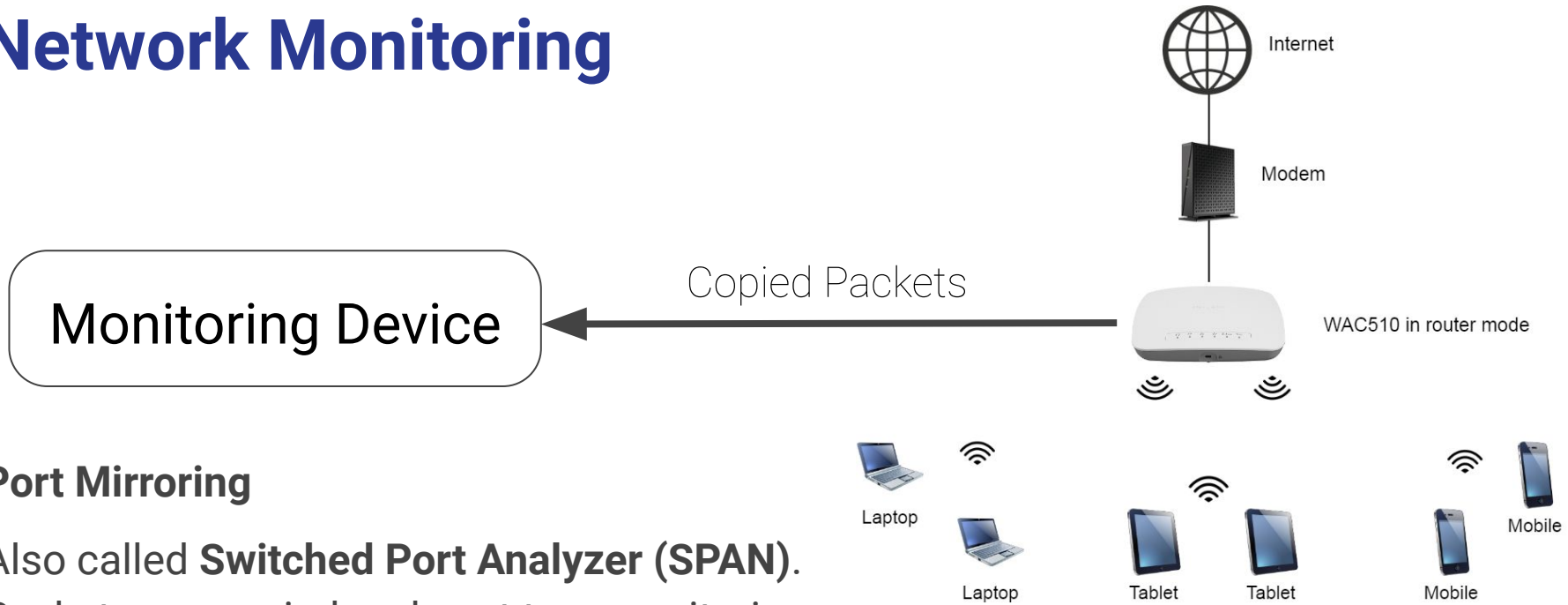
Send traffic to a monitoring device.



Source:

<https://kb.netgear.com/000058849/What-do-I-need-to-know-about-using-my-NETGEAR-WAC510-access-point-in-router-mode>

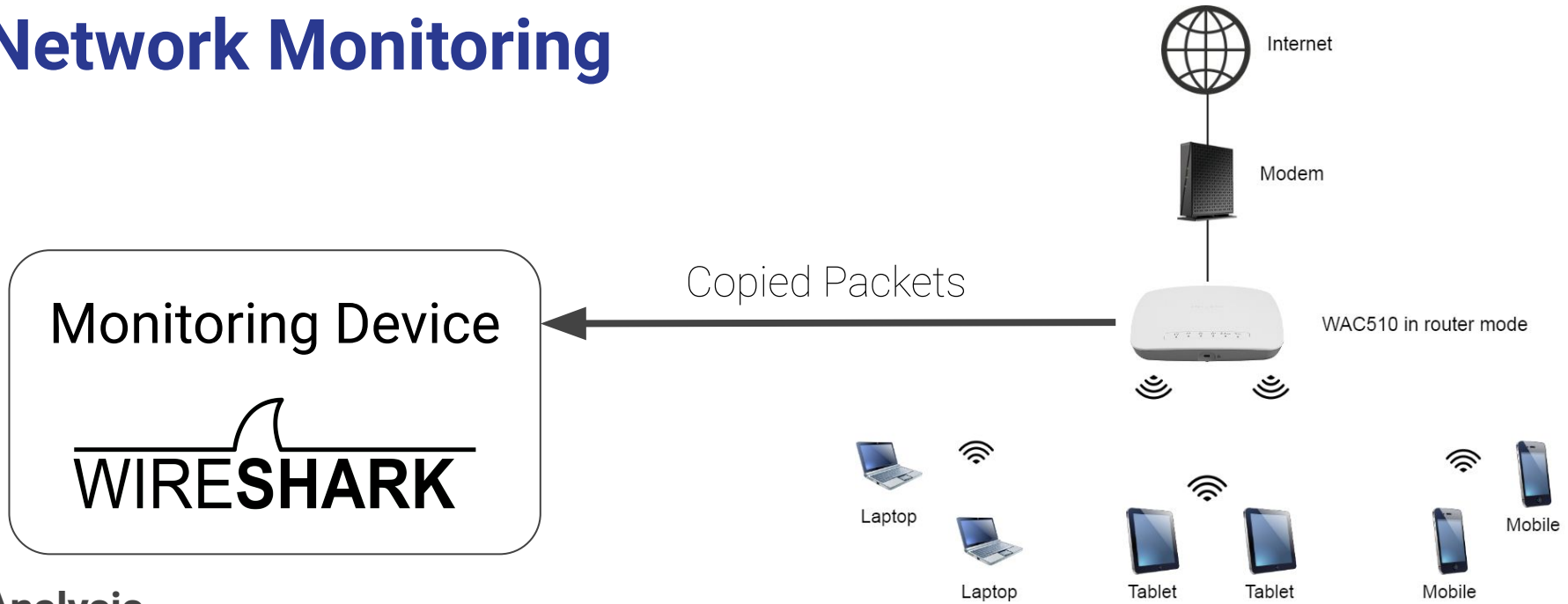
Network Monitoring



Port Mirroring

Also called **Switched Port Analyzer (SPAN)**.
Packets are copied and sent to a monitoring device.

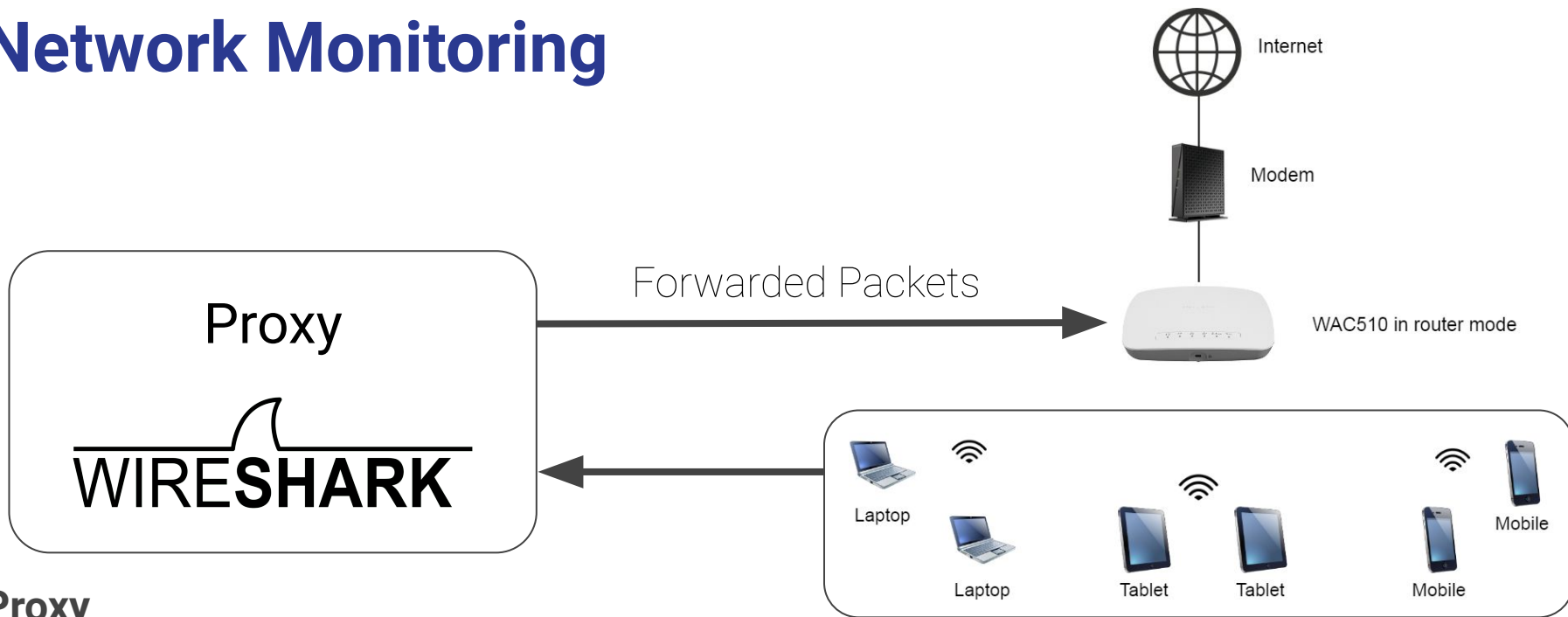
Network Monitoring



Analysis

Use tools provided by Wireshark to analyze the traffic.

Network Monitoring



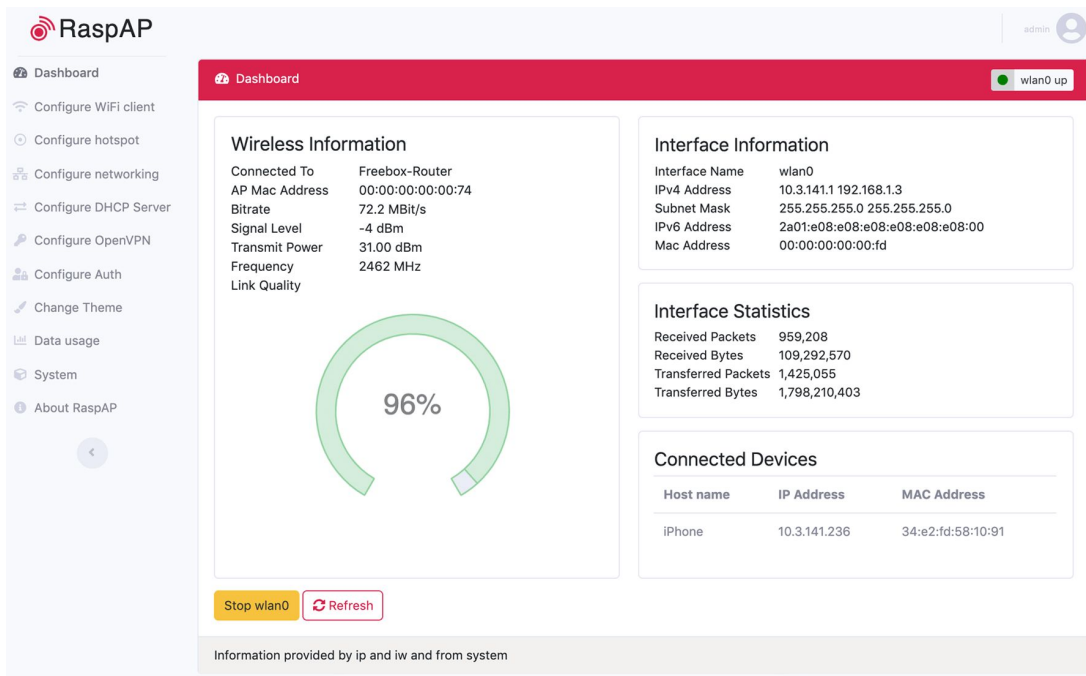
Proxy

Wireshark can see all traffic that goes through a device.

Raspberry Pi Proxy

RaspAP

WiFi Configuration Portal



WiFi proxy forwarding

Collects usage statistics

Enables the capturing of packets from connected devices

Source: <https://raspap.com/>

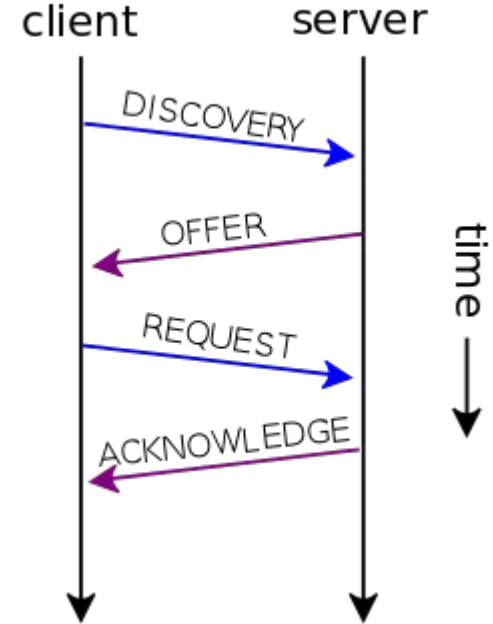
DHCP

Dynamic Host Configuration Protocol

Centralized protocol for assigning IP addresses to devices connecting to the network

Four Phases

1. Discovery
2. Offer
3. Request
4. Acknowledgement



Source:

https://en.wikipedia.org/wiki/Dynamic_Host_Configuration_Protocol

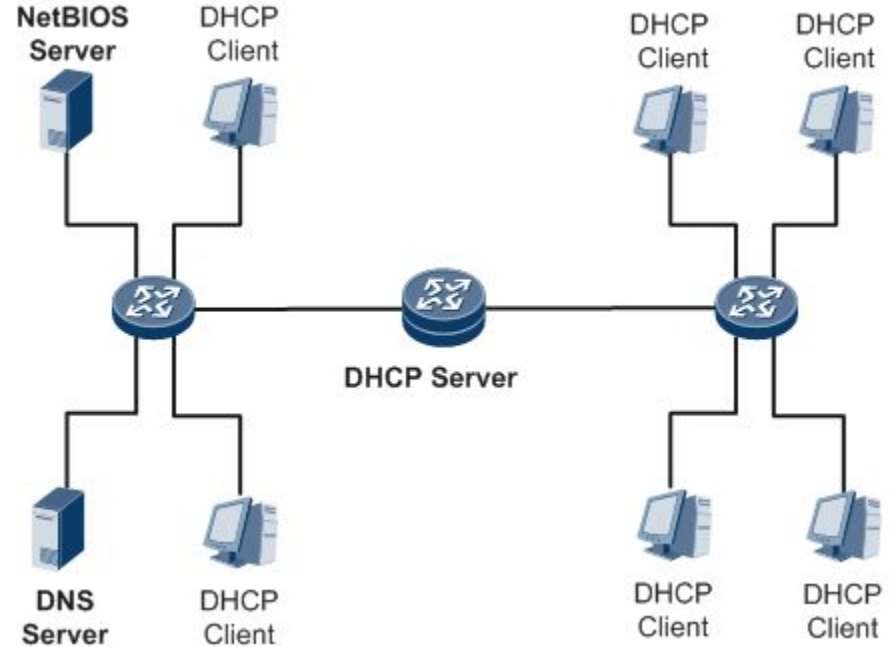
DHCP

Dynamic Host Configuration Protocol

Clients send a broadcast to request information from the DHCP server

Server responds with available IP addresses and client configuration:

- Default gateway
- Domain name
- Name servers
- Time servers



Source:

<https://support.huawei.com/enterprise/en/doc/EDO1100059438/93af6f6a/configuring-a-dhcp-server>

iptables

```
sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state RELATED,ESTABLISHED -j ACCEPT
sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
sudo iptables -t nat -A PREROUTING -i wlan0 -p tcp -m tcp --dport 80 -j REDIRECT --to-ports 8080
sudo iptables -t nat -A PREROUTING -i wlan0 -p tcp -m tcp --dport 443 -j REDIRECT --to-ports 8080
```

Send ethernet network traffic to our wireless network

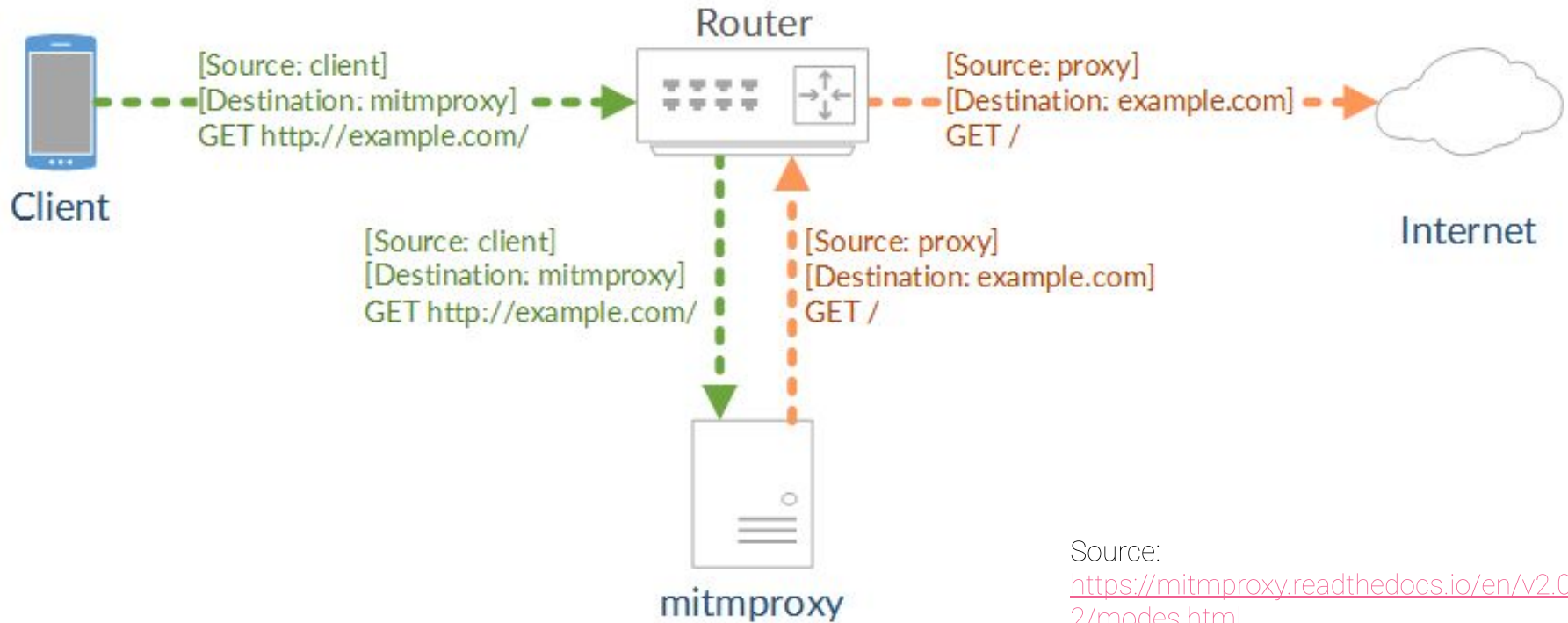
Send the wireless network traffic back to ethernet

Save traffic from ports 80 and 443 to port 8080

Our proxy will read from port 8080

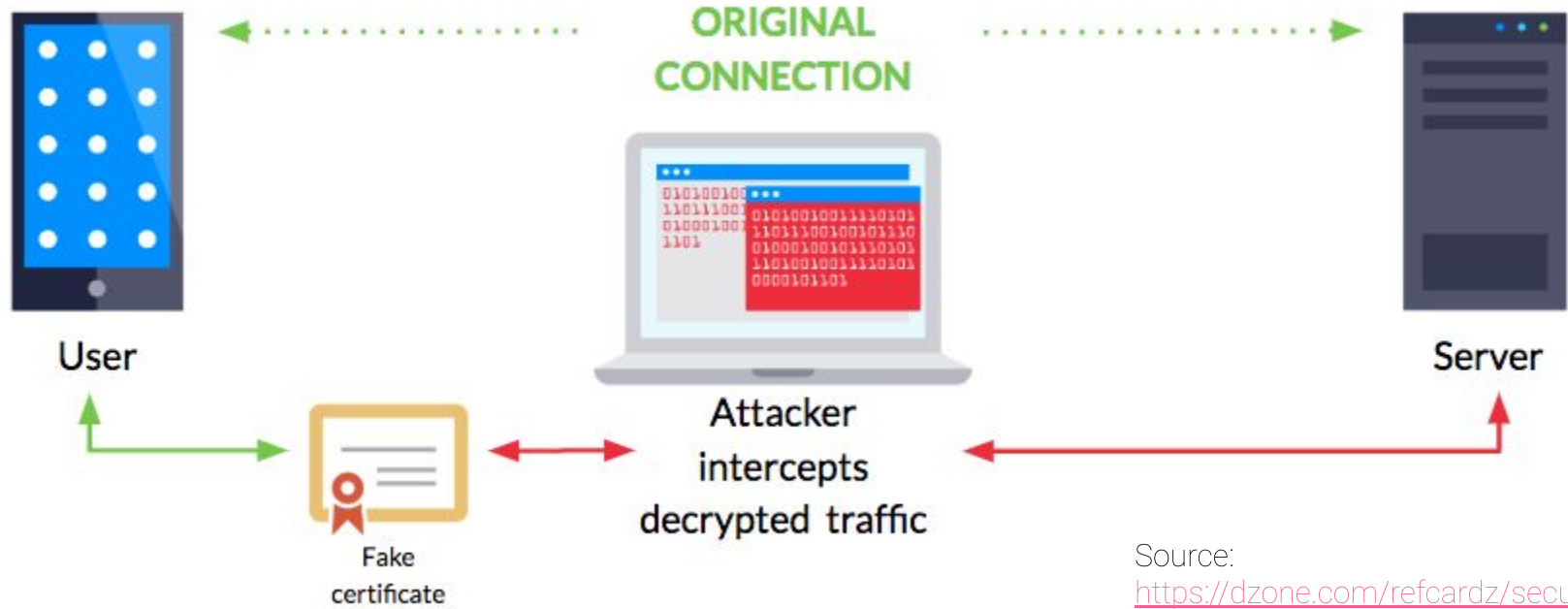


mitmproxy



Source:
<https://mitmproxy.readthedocs.io/en/v2.0.2/modes.html>

Certificate Authority Replacement



Source:

<https://dzone.com/refcardz/securing-mobile-applications-with-cert-pinning>

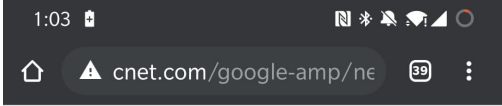
Built In Protection

Browsers will check a list of trusted authorities to see if the website certificate came from a trusted source

If the source is not trusted (public key not in its database) the website will be flagged

User have the option to disregard the message and continue to the site

This leaves the users vulnerable to the man in the middle attack



Your connection is not private

Attackers might be trying to steal your information from **www.cnet.com** (for example, passwords, messages, or credit cards). [Learn more](#)

NET::ERR_CERT_AUTHORITY_INVALID

☐ Help improve Chrome security by sending URLs of some pages you visit, limited system information, and some page content to Google. [Privacy policy](#).

Reload

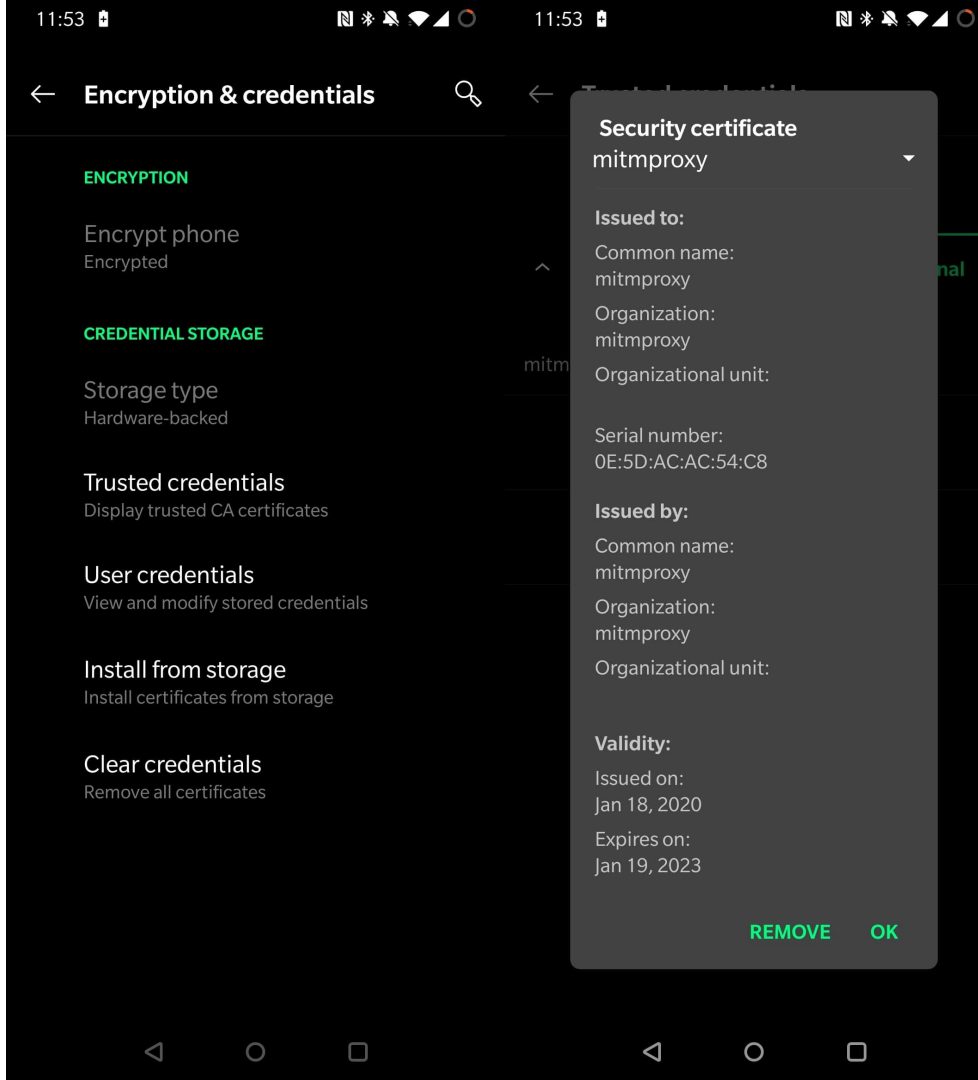
Advanced

Compromising the User

Add MITM proxy
authority to list of
trusted authorities

On page visit the
browser will check
the certificate

Certificate will be
deemed acceptable



Secure Website without Proxy

pi@raspberrypi: ~

Messenger - Chromiu... *wlan0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ssll

No.	Time	Source	Destination	Protocol	Length	Info
58	42.558417788	192.168.1.139	192.168.42.10	TLSv1.2	382	Application Data
71	42.608223105	192.168.42.10	172.217.165.4	TLSv1.3	624	Client Hello
83	42.649600555	192.168.42.10	172.253.122.188	TLSv1.3	599	Client Hello
88	42.674721030	172.253.122.188	192.168.42.10	TLSv1.3	269	Server Hello, Change Cipher Spec, Application Data
91	42.677010577	192.168.42.10	172.253.122.188	TLSv1.3	139	Change Cipher Spec, Application Data
92	42.683143472	192.168.42.10	172.217.165.10	TLSv1.3	583	Client Hello
100	42.708706325	192.168.42.10	172.253.122.188	TLSv1.3	458	Application Data
102	42.744784952	172.253.122.188	192.168.42.10	TLSv1.3	595	Application Data, Application Data
103	42.759856167	172.253.122.188	192.168.42.10	TLSv1.3	173	Application Data
104	42.760114847	172.253.122.188	192.168.42.10	TLSv1.3	184	Application Data
107	42.775142877	172.217.165.4	192.168.42.10	TLSv1.3	1514	Server Hello, Change Cipher Spec, Application Data
108	42.775173228	172.217.165.4	192.168.42.10	TLSv1.3	923	Application Data, Application Data, Application Data
111	42.785701066	192.168.42.10	172.217.165.4	TLSv1.3	96	Change Cipher Spec, Application Data
121	42.884073738	172.217.165.10	192.168.42.10	TLSv1.3	1514	Server Hello, Change Cipher Spec, Application Data
122	42.884121607	172.217.165.10	192.168.42.10	TLSv1.3	1400	Application Data, Application Data, Application Data
125	42.897730018	192.168.42.10	172.217.165.10	TLSv1.3	96	Change Cipher Spec, Application Data
140	43.830618833	192.168.42.10	172.217.165.10	TLSv1.3	583	Client Hello
142	43.943735572	172.217.165.10	192.168.42.10	TLSv1.3	1514	Server Hello, Change Cipher Spec, Application Data
143	43.943782201	172.217.165.10	192.168.42.10	TLSv1.3	1400	Application Data, Application Data, Application Data
146	43.951993684	192.168.42.10	172.217.165.10	TLSv1.3	96	Change Cipher Spec, Application Data
158	43.984420642	192.168.42.10	172.217.165.4	TLSv1.3	624	Client Hello
168	44.171888421	172.217.165.4	192.168.42.10	TLSv1.3	1514	Server Hello, Change Cipher Spec, Application Data
169	44.171914124	172.217.165.4	192.168.42.10	TLSv1.3	923	Application Data, Application Data, Application Data
172	44.182807977	192.168.42.10	172.217.165.4	TLSv1.3	96	Change Cipher Spec, Application Data
190	45.269552501	192.168.42.10	172.217.165.10	TLSv1.3	583	Client Hello
192	45.409281539	172.217.165.10	192.168.42.10	TLSv1.3	1514	Server Hello, Change Cipher Spec, Application Data
193	45.409395816	172.217.165.10	192.168.42.10	TLSv1.3	1400	Application Data, Application Data, Application Data
196	45.427609714	192.168.42.10	172.217.165.10	TLSv1.3	96	Change Cipher Spec, Application Data
211	46.397554978	192.168.42.10	172.217.165.4	TLSv1.3	624	Client Hello
218	46.589014311	172.217.165.4	192.168.42.10	TLSv1.3	1514	Server Hello, Change Cipher Spec, Application Data
219	46.589126531	172.217.165.4	192.168.42.10	TLSv1.3	923	Application Data, Application Data, Application Data
222	46.596911170	192.168.42.10	172.217.165.4	TLSv1.3	96	Change Cipher Spec, Application Data
243	47.052934458	192.168.42.10	172.217.165.10	TLSv1.3	583	Client Hello
247	47.756389544	172.217.165.10	192.168.42.10	TLSv1.3	1514	Server Hello, Change Cipher Spec, Application Data
248	47.756425321	172.217.165.10	192.168.42.10	TLSv1.3	1400	Application Data, Application Data, Application Data
253	47.846873724	192.168.42.10	172.217.165.10	TLSv1.3	96	Change Cipher Spec, Application Data
273	50.860667479	192.168.42.10	172.217.165.4	TLSv1.3	624	Client Hello
281	51.058085976	172.217.165.4	192.168.42.10	TLSv1.3	1514	Server Hello, Change Cipher Spec, Application Data
282	51.058105709	172.217.165.4	192.168.42.10	TLSv1.3	923	Application Data, Application Data, Application Data
285	51.088647527	192.168.42.10	172.217.165.4	TLSv1.3	96	Change Cipher Spec, Application Data
332	52.187787556	192.168.42.10	96.6.23.156	TLSv1.3	583	Client Hello
336	52.188560022	192.168.42.10	172.217.1.10	TLSv1.3	583	Client Hello
338	52.188626502	192.168.42.10	172.217.164.226	TLSv1.3	583	Client Hello
349	52.193970205	192.168.42.10	172.217.1.172	TLSv1.3	583	Client Hello
356	52.203880047	192.168.42.10	52.206.200.225	TLSv1.3	583	Client Hello
367	52.303228762	192.168.42.10	172.217.165.10	TLSv1.3	583	Client Hello

Frame 103: 173 bytes on wire (1384 bits), 173 bytes captured (1384 bits) on interface 0

Ethernet II, Src: Raspberr1a:6a:60 (dc:a6:32:1a:6a:60), Dst: 5a:31:bd:37:c6:ab (5a:31:bd:37:c6:ab)

Internet Protocol Version 4, Src: 172.253.122.188, Dst: 192.168.42.10

Transmission Control Protocol, Src Port: 5228, Dst Port: 47290, Seq: 733, Ack: 990, Len: 107

Secure Sockets Layer

TLSv1.3 Record Layer: Application Data Protocol: Application Data

Opaque Type: Application Data (23)

Version: TLS 1.2 (0x0303)

Length: 102

Encrypted Application Data: 552c9c2039f0ea95dc691218a9c198531d376cbbd1f1b...

Secure Website with Proxy

More information is made available

MITM decrypts ciphertext encrypted with its certificate public key

Request	Response	Details
GET https://opensource.com/article/20/1/wireshark-linux-tshark HTTP/2.0		
:authority	opensource.com	
cache-control	max-age=0	
save-data	on	
upgrade-insecure-requests	1	
user-agent	Mozilla/5.0 (Linux; Android 10; ONEPLUS A6013) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.3945.116 Mobile Safari/537.36	
sec-fetch-user	?1	
accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9	
sec-fetch-site	none	
sec-fetch-mode	navigate	
referrer	https://www.googleapis.com/auth/chrome-content-suggestions	
accept-encoding	gzip, deflate, br	
accept-language	en-US,en;q=0.9,ru;q=0.8	
cookie	has_js=1; _ga=GA1.2.32918463.1579543743; _gid=GA1.2.729685603.1579543743; check=true; AMCVS_945D02BE532957400A490D4C%40AdobeOrg=1; rh_omni_tc=701f2000010H7nAAG; sat_prevInternalCampaign=; sat_prevExtCmp=no%20value; scCidHist=701f20000010H7nAAG; s_cc=true; mbox=session#bia9fd8ecb44491ea809e51c2036cc78%1579549244 PC#bia9fd8ecb44491ea809e51c2036cc78.17_0%1642792183; dtm_prevURL=https%3A%2F%2Fopensource.com%2Farticle%2F20%2F1%2FWireshark-linux-tshark%2520my; sat_ppv=54; _CT_RS_=Recording; WRUICD=2612870918439262; _CT_Data=gpv=2&ckp=tl&dm=opensource.com&apv_25612_www14=4&cpv_25612_www14=2&pv_25612_www14=1; AMCV_945D02BE532957400A490D4C%40AdobeOrg=10750059558%7CMCIDTS%7C18282%7CMCMID%7C23417381787794176430122423514770256766%7CMCAAMLH-1580152184%7C7%7CMCAAMB-1580152184%7CRKhpRz8krq2tL06pguXwp5olkAcUniQYPHaMwgdJ3xzPWQmdj0y%7CMCOPTOUT-1579554584s%7CNCONE%7CMCAID%7CNCONE%7CvVersion%7C4.4.1%7CMCCIDH%7C1332280493; _cs_ex=1; _cs_c=1; rh_elqCustomerGUID=4e25e161-4417-42ac-8fb0-1c4003e4368d; ctm=eydwZ3Yn0jE4NzIwMTEzMtG4NzEzOTB8J3ZzdCc6Njg3NDM1NDI3MjEzMTIwM2H0HndnN0cic6NDU1Njg4NDEwMzQ5NTExMHNhW50cic6MTU3OTU0NzUyODM5OXwmdic6MX0=	

Request content missing.

Wireshark and SSL/TLS Master Secrets

The SSL/TLS master keys can be logged by mitmproxy so that external programs can decrypt SSL/TLS connections both from and to the proxy. Recent versions of Wireshark can use these log files to decrypt packets. See the [Wireshark wiki](#) for more information.

Key logging is enabled by setting the environment variable `SSLKEYLOGFILE` so that it points to a writable text file:

```
SSLKEYLOGFILE="$PWD/.mitmproxy/sslkeylogfile.txt" mitmproxy
```

You can also `export` this environment variable to make it persistent for all applications started from your current shell session.

You can specify the key file path in Wireshark via `Edit -> Preferences -> Protocols -> SSL -> (Pre)-Master-Secret log filename`. If your `SSLKEYLOGFILE` does not exist yet, just create an empty text file, so you can select it in Wireshark (or run mitmproxy to create and collect master secrets).

Note that `SSLKEYLOGFILE` is respected by other programs as well, e.g., Firefox and Chrome. If this creates any issues, you can use `MITMPROXY_SSLKEYLOGFILE` instead without affecting other applications.

Source: <https://docs.mitmproxy.org/stable/howto-wireshark-tls/>

Man-in-the-Middle

Raspberry Pi

Takeaways

Wireshark offers a suite of tools for capturing and analyzing network traffic

HTTP is not secure - plaintext can be captured by an eavesdropper

HTTPS is secure - the traffic is visible but the data is encrypted

On an unsecure network connection, even HTTPS can be not enough to protect your credentials

If a user's devices can be physically compromised, the user can be made unaware of a surveyed network

Questions?