CPTR 494 final Project

Part 1: Setting Up Wireshark and Initial Capture

1. **Install Wireshark (if not already installed)**:

   Download and install Wireshark from https://www.wireshark.org/.

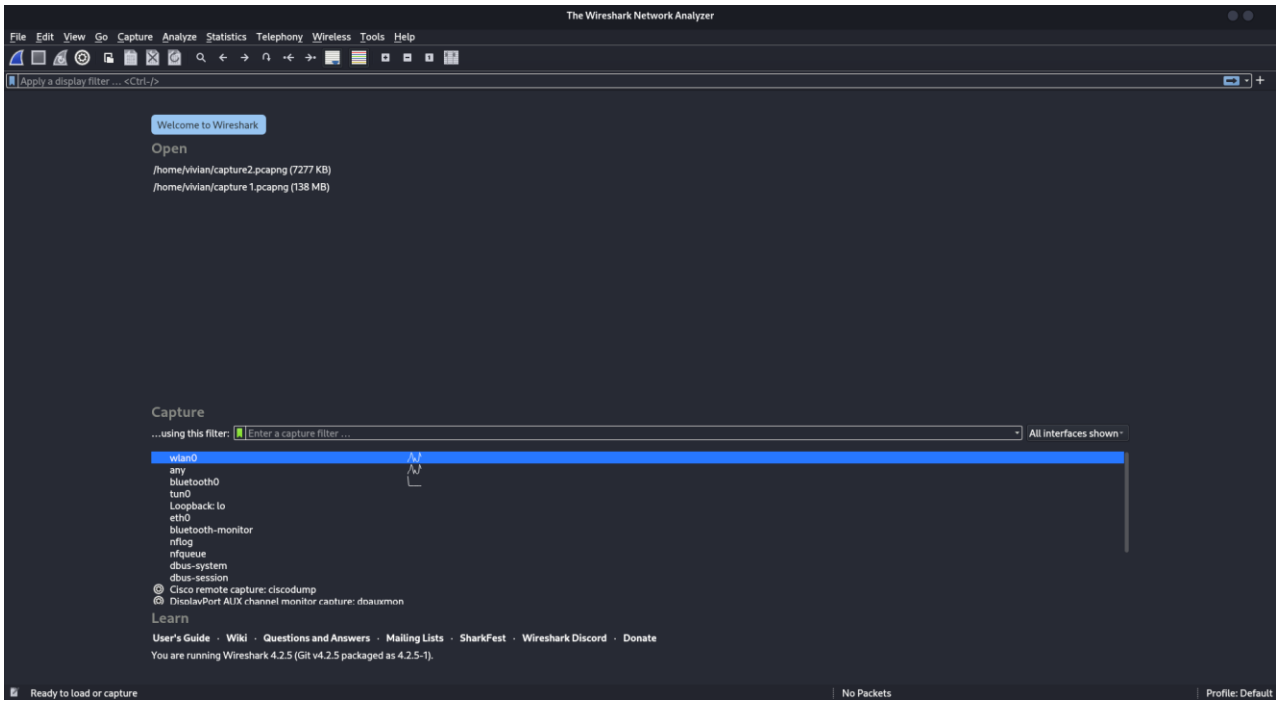
   Ensure that the user has appropriate privileges to capture network traffic (admin/root access may be required).

2. **Start Wireshark**:

   Open Wireshark.

   Select the network interface that you will capture from (e.g., Ethernet, Wi-Fi).

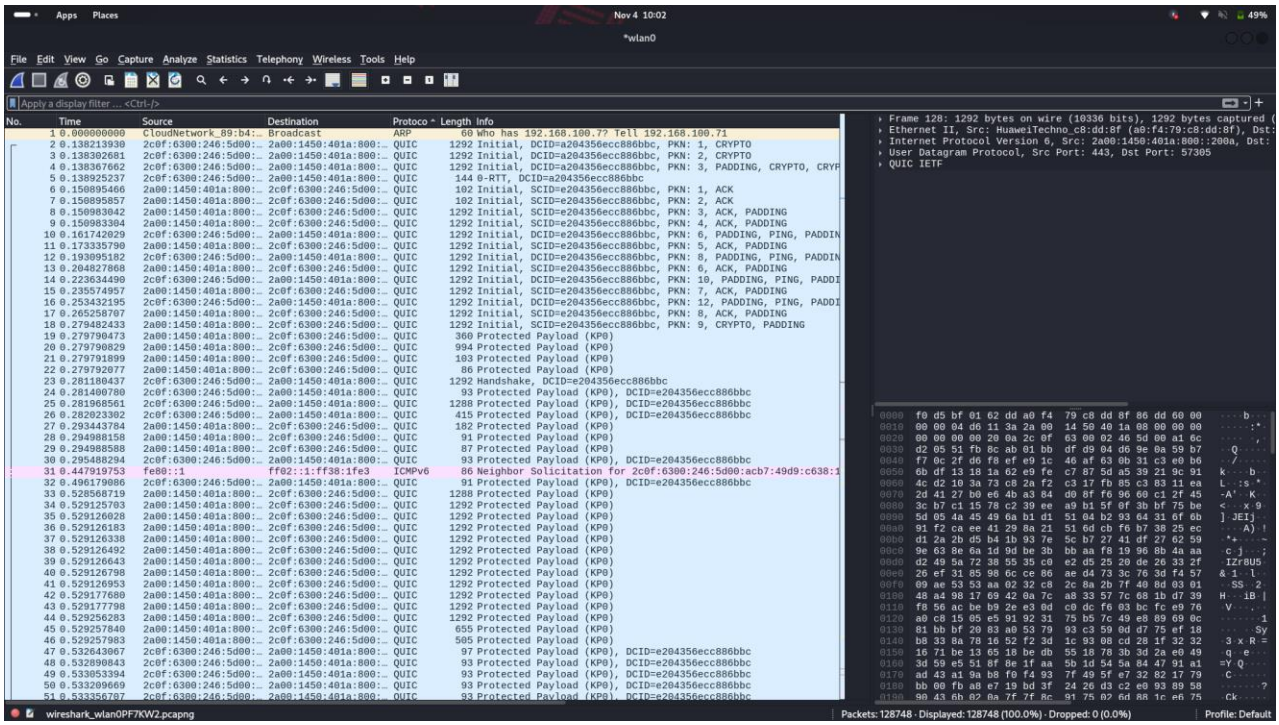
   Click on the **Start Capturing Packets** button (the blue shark fin icon).



3. **Generate Traffic**:

   Open a browser and visit a few websites.

   Perform a file download or stream a video to generate more traffic.



4. **Stop the Capture**:

   After a few minutes, click the **Stop Capturing Packets** button (the red square).

Part 2: Analyzing Network Traffic

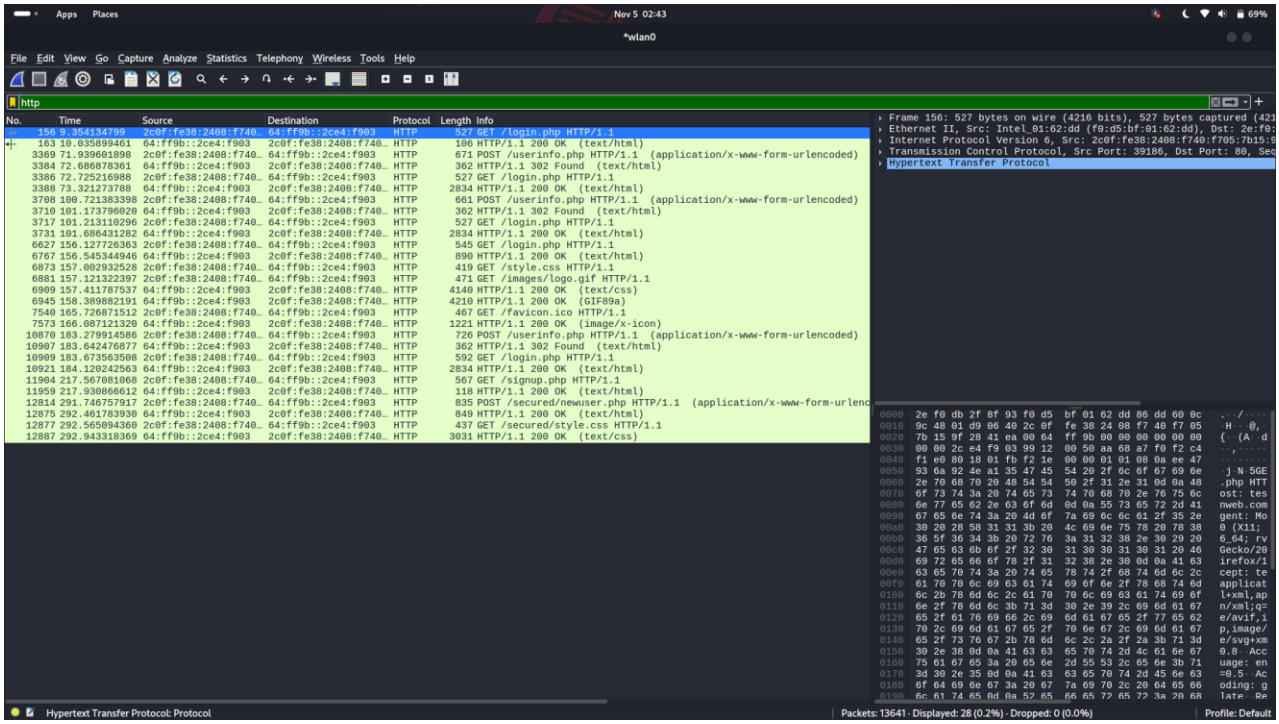
1. **Examine Protocols**:

   In the Wireshark window, notice the **Protocol** column.

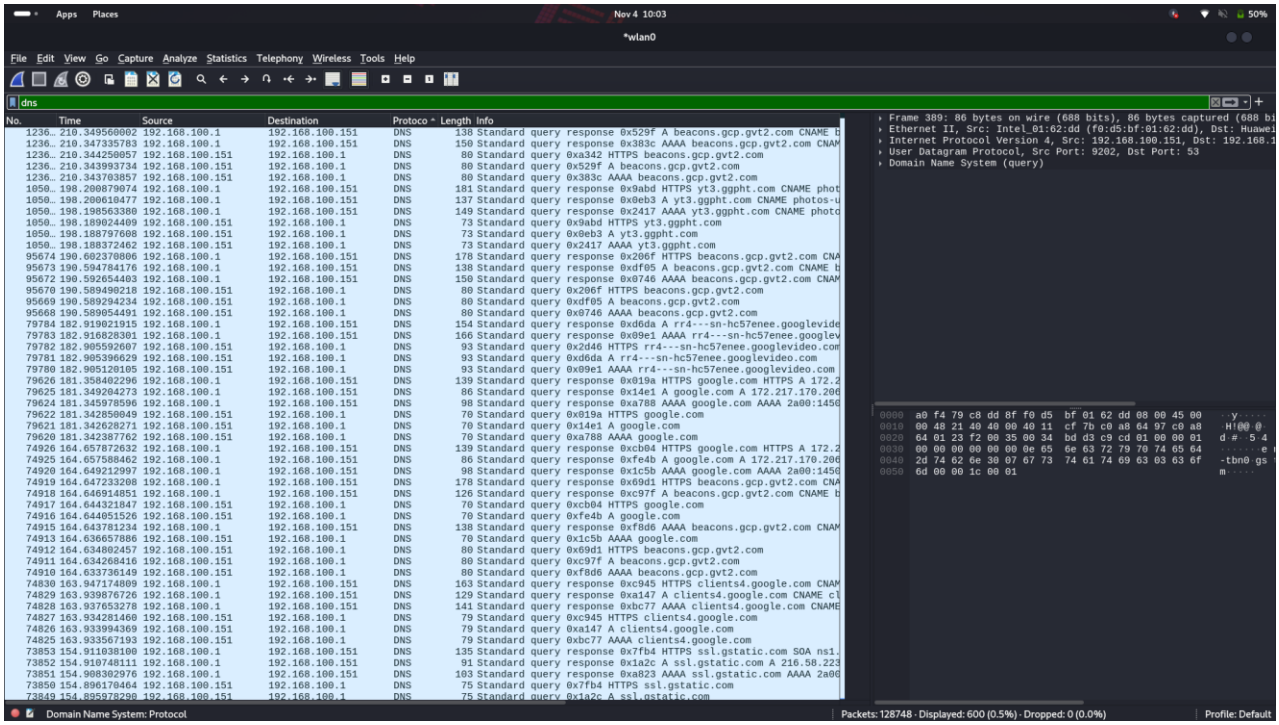
   Identify common protocols like **HTTP**, **HTTPS**, **TCP**, **UDP**, **DNS**, etc.

   Use the filter bar to focus on specific protocols:

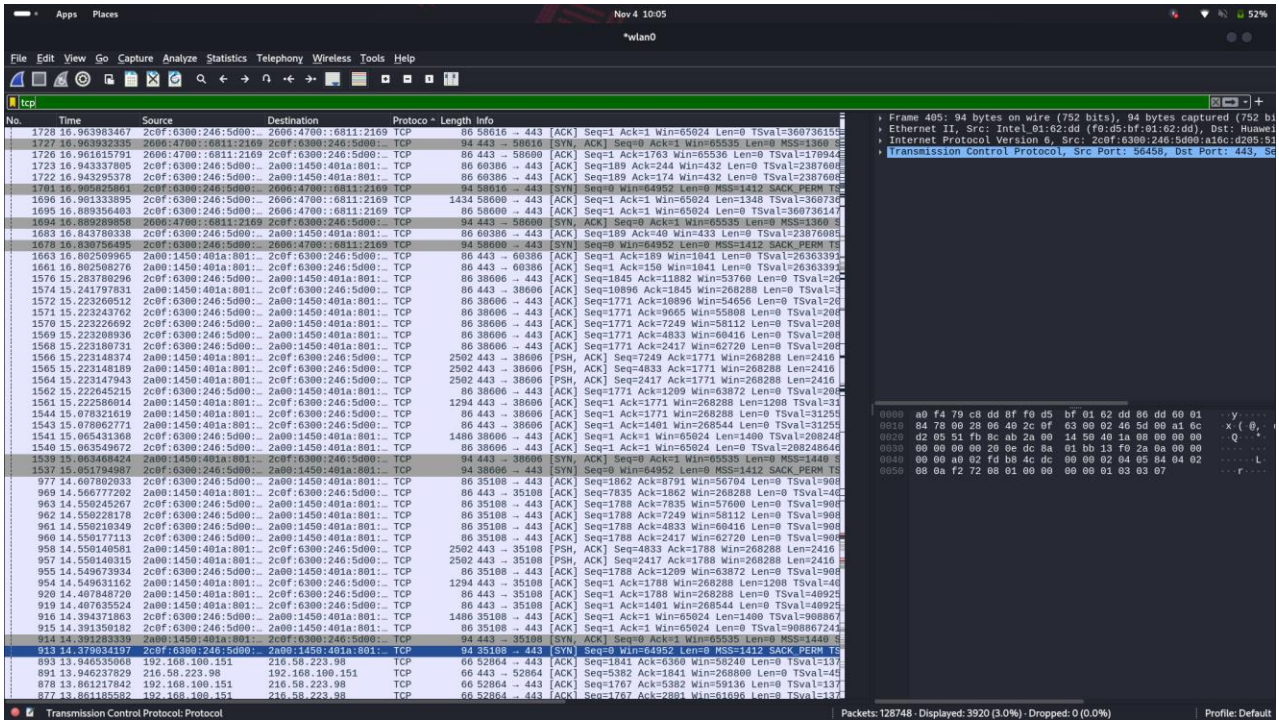
HTTP


Apps Places
Nov 5 02:43
*wlan0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http
No. Time Source Destination Protocol Length Info
156 9.354134799 2c0f:fe38:2408:f740… 64:ff9b::2ce4:f903 HTTP 527 GET /login.php HTTP/1.1
163 10.035899461 64:ff9b::2ce4:f903 2c0f:fe38:2408:f740… HTTP 106 HTTP/1.1 200 OK (text/html)
3369 71.939601898 2c0f:fe38:2408:f740… 64:ff9b::2ce4:f903 HTTP 671 POST /userinfo.php HTTP/1.1 (application/x-www-form-urlencoded)
3384 72.686878361 64:ff9b::2ce4:f903 2c0f:fe38:2408:f740… HTTP 362 HTTP/1.1 302 Found (text/html)
3386 72.725216988 2c0f:fe38:2408:f740… 64:ff9b::2ce4:f903 HTTP 527 GET /login.php HTTP/1.1
3388 73.321273788 64:ff9b::2ce4:f903 2c0f:fe38:2408:f740… HTTP 2834 HTTP/1.1 200 OK (text/html)
3708 100.721383398 2c0f:fe38:2408:f740… 64:ff9b::2ce4:f903 HTTP 661 POST /userinfo.php HTTP/1.1 (application/x-www-form-urlencoded)
3710 101.173796028 64:ff9b::2ce4:f903 2c0f:fe38:2408:f740… HTTP 362 HTTP/1.1 302 Found (text/html)
3717 101.213110296 2c0f:fe38:2408:f740… 64:ff9b::2ce4:f903 HTTP 527 GET /login.php HTTP/1.1
3731 101.686431282 64:ff9b::2ce4:f903 2c0f:fe38:2408:f740… HTTP 2834 HTTP/1.1 200 OK (text/html)
6627 156.127726363 2c0f:fe38:2408:f740… 64:ff9b::2ce4:f903 HTTP 545 GET /login.php HTTP/1.1
6767 156.545344946 64:ff9b::2ce4:f903 2c0f:fe38:2408:f740… HTTP 890 HTTP/1.1 200 OK (text/html)
6873 157.002932528 2c0f:fe38:2408:f740… 64:ff9b::2ce4:f903 HTTP 419 GET /style.css HTTP/1.1
6881 157.121322397 2c0f:fe38:2408:f740… 64:ff9b::2ce4:f903 HTTP 471 GET /images/logo.gif HTTP/1.1
6909 157.411787537 64:ff9b::2ce4:f903 2c0f:fe38:2408:f740… HTTP 4140 HTTP/1.1 200 OK (text/css)
6945 158.389882191 64:ff9b::2ce4:f903 2c0f:fe38:2408:f740… HTTP 4210 HTTP/1.1 200 OK (GIF89a)
7540 165.726871512 2c0f:fe38:2408:f740… 64:ff9b::2ce4:f903 HTTP 467 GET /favicon.ico HTTP/1.1
7573 166.087121320 64:ff9b::2ce4:f903 2c0f:fe38:2408:f740… HTTP 1221 HTTP/1.1 200 OK (image/x-icon)
10870 183.279914586 2c0f:fe38:2408:f740… 64:ff9b::2ce4:f903 HTTP 726 POST /userinfo.php HTTP/1.1 (application/x-www-form-urlencoded)
10907 183.642476877 64:ff9b::2ce4:f903 2c0f:fe38:2408:f740… HTTP 362 HTTP/1.1 302 Found (text/html)
10909 183.673563508 2c0f:fe38:2408:f740… 64:ff9b::2ce4:f903 HTTP 592 GET /login.php HTTP/1.1
10921 184.120242563 64:ff9b::2ce4:f903 2c0f:fe38:2408:f740… HTTP 2834 HTTP/1.1 200 OK (text/html)
11904 217.567081068 2c0f:fe38:2408:f740… 64:ff9b::2ce4:f903 HTTP 567 GET /signup.php HTTP/1.1
11959 217.930866612 64:ff9b::2ce4:f903 2c0f:fe38:2408:f740… HTTP 118 HTTP/1.1 200 OK (text/html)
12814 291.746757917 2c0f:fe38:2408:f740… 64:ff9b::2ce4:f903 HTTP 835 POST /secured/newuser.php HTTP/1.1 (application/x-www-form-urlenc
12875 292.461783930 64:ff9b::2ce4:f903 2c0f:fe38:2408:f740… HTTP 849 HTTP/1.1 200 OK (text/html)
12877 292.565094360 2c0f:fe38:2408:f740… 64:ff9b::2ce4:f903 HTTP 437 GET /secured/style.css HTTP/1.1
12887 292.943318369 64:ff9b::2ce4:f903 2c0f:fe38:2408:f740… HTTP 3031 HTTP/1.1 200 OK (text/css)
Frame 156: 527 bytes on wire (4216 bits), 527 bytes captured (421
Ethernet II, Src: Intel_01:62:dd (f0:d5:bf:01:62:dd), Dst: 2e:f0:
Internet Protocol Version 6, Src: 2c0f:fe38:2408:f740:f705:7b15:9
Transmission Control Protocol, Src Port: 39186, Dst Port: 80, Seq
Hypertext Transfer Protocol
0000 2e f0 db 2f 8f 93 f0 d5 bf 01 62 dd 86 dd 60 0c
0010 9c 48 01 d9 06 40 2c 0f fe 38 24 08 f7 40 f7 05
0020 7b 15 9f 28 41 ea 00 64 ff 9b 00 00 00 00 00 00
0030 00 00 2c e4 f9 03 99 12 00 50 aa 68 a7 f0 f2 c4
0040 f1 e0 80 18 01 fb f2 1e 00 00 01 01 08 0a ee 47
0050 93 6a 92 4e a1 35 47 45 54 20 2f 6c 6f 67 69 6e
0060 2e 70 68 70 20 48 54 54 50 2f 31 2e 31 0d 0a 48
0070 6f 73 74 3a 20 74 65 73 74 70 68 70 2e 76 75 6c
0080 6e 77 65 62 2e 63 6f 6d 0d 0a 55 73 65 72 2d 41
0090 67 65 6e 74 3a 20 4d 6f 7a 69 6c 6c 61 2f 35 2e
00a0 30 20 28 58 31 31 3b 20 4c 69 6e 75 78 20 78 38
00b0 36 5f 36 34 3b 20 72 76 3a 31 32 38 2e 30 29 20
00c0 47 65 63 6b 6f 2f 32 30 31 30 30 31 30 31 20 46
00d0 69 72 65 66 6f 78 2f 31 32 38 2e 30 0d 0a 41 63
00e0 63 65 70 74 3a 20 74 65 78 74 2f 68 74 6d 6c 2c
00f0 61 70 70 6c 69 63 61 74 69 6f 6e 2f 78 68 74 6d
0100 6c 2b 78 6d 6c 2c 61 70 70 6c 69 63 61 74 69 6f
0110 6e 2f 78 6d 6c 3b 71 3d 30 2e 39 2c 69 6d 61 67
0120 65 2f 61 76 69 66 2c 69 6d 61 67 65 2f 77 65 62
0130 70 2c 69 6d 61 67 65 2f 70 6e 67 2c 69 6d 61 67
0140 65 2f 73 76 67 2b 78 6d 6c 2c 2a 2f 2a 3b 71 3d
0150 30 2e 38 0d 0a 41 63 63 65 70 74 2d 4c 61 6e 67
0160 75 61 67 65 3a 20 65 6e 2d 55 53 2c 65 6e 3b 71
0170 3d 30 2e 35 0d 0a 41 63 63 65 70 74 2d 45 6e 63
0180 6f 64 69 6e 67 3a 20 67 7a 69 70 2c 20 64 65 66
0190 6c 61 74 65 0d 0a 52 65 66 65 72 65 72 3a 20 68
Hypertext Transfer Protocol: Protocol
Packets: 13641 · Displayed: 28 (0.2%) · Dropped: 0 (0.0%)
Profile: Default

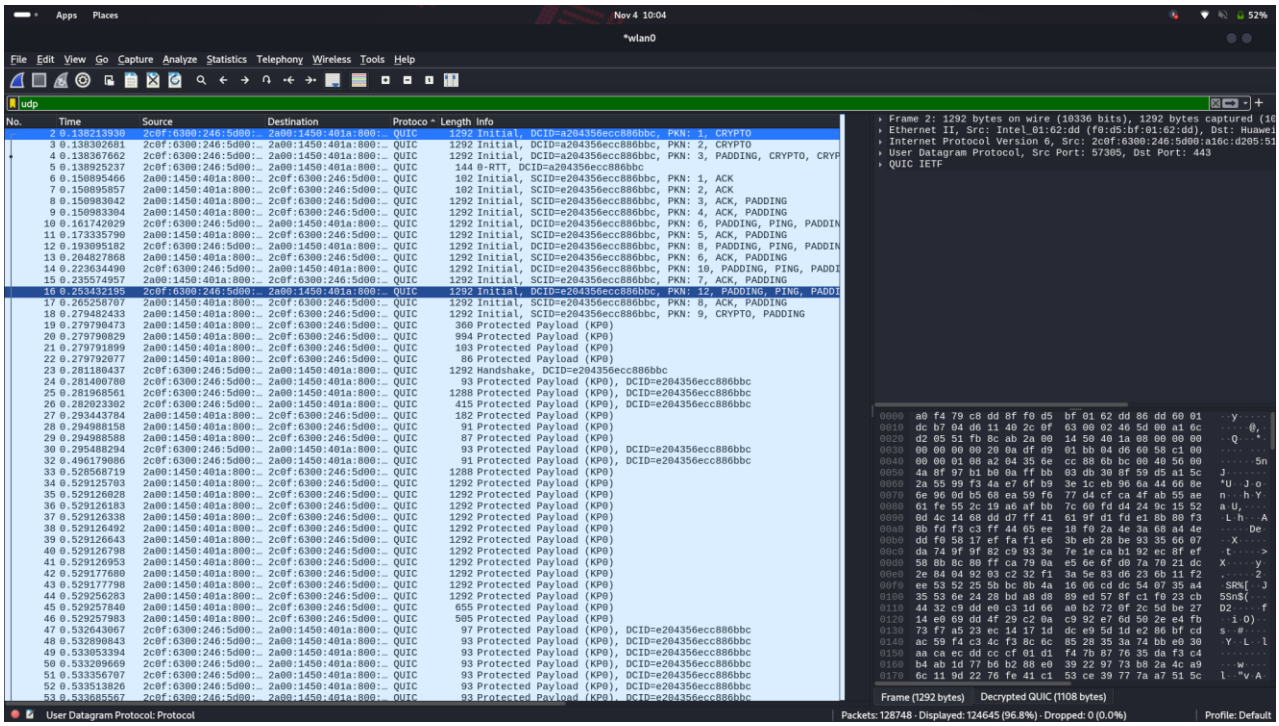
DNS


Apps Places
Nov 4 10:03
*wlan0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
dns
No. Time Source Destination Protocol Length Info
1236… 210.349560002 192.168.100.1 192.168.100.151 DNS 138 Standard query response 0x529f A beacons.gcp.gvt2.com CNAME b
1236… 210.347335783 192.168.100.1 192.168.100.151 DNS 150 Standard query response 0x383c AAAA beacons.gcp.gvt2.com CNAM
1236… 210.344250057 192.168.100.151 192.168.100.1 DNS 80 Standard query 0xa342 HTTPS beacons.gcp.gvt2.com
1236… 210.343993734 192.168.100.151 192.168.100.1 DNS 80 Standard query 0x529f A beacons.gcp.gvt2.com
1236… 210.343703857 192.168.100.151 192.168.100.1 DNS 80 Standard query 0x383c AAAA beacons.gcp.gvt2.com
1050… 198.200879074 192.168.100.1 192.168.100.151 DNS 181 Standard query response 0x9abd HTTPS yt3.ggpht.com CNAME phot
1050… 198.200610477 192.168.100.1 192.168.100.151 DNS 137 Standard query response 0x0eb3 A yt3.ggpht.com CNAME photos-u
1050… 198.198563380 192.168.100.1 192.168.100.151 DNS 149 Standard query response 0x2417 AAAA yt3.ggpht.com CNAME photo
1050… 198.189024409 192.168.100.151 192.168.100.1 DNS 73 Standard query 0x9abd HTTPS yt3.ggpht.com
1050… 198.188797608 192.168.100.151 192.168.100.1 DNS 73 Standard query 0x0eb3 A yt3.ggpht.com
1050… 198.188372462 192.168.100.151 192.168.100.1 DNS 73 Standard query 0x2417 AAAA yt3.ggpht.com
95674 190.602370806 192.168.100.1 192.168.100.151 DNS 178 Standard query response 0x206f HTTPS beacons.gcp.gvt2.com CNA
95673 190.594784176 192.168.100.1 192.168.100.151 DNS 138 Standard query response 0xdf05 A beacons.gcp.gvt2.com CNAME b
95672 190.592654403 192.168.100.1 192.168.100.151 DNS 150 Standard query response 0x0746 AAAA beacons.gcp.gvt2.com CNAM
95670 190.589490218 192.168.100.151 192.168.100.1 DNS 80 Standard query 0x206f HTTPS beacons.gcp.gvt2.com
95669 190.589294234 192.168.100.151 192.168.100.1 DNS 80 Standard query 0xdf05 A beacons.gcp.gvt2.com
95668 190.589054491 192.168.100.151 192.168.100.1 DNS 80 Standard query 0x0746 AAAA beacons.gcp.gvt2.com
79784 182.919021915 192.168.100.1 192.168.100.151 DNS 154 Standard query response 0xd6da A rr4---sn-hc57enee.googlevide
79783 182.916828301 192.168.100.1 192.168.100.151 DNS 166 Standard query response 0x09e1 AAAA rr4---sn-hc57enee.googlev
79782 182.905592607 192.168.100.151 192.168.100.1 DNS 93 Standard query 0x2d46 HTTPS rr4---sn-hc57enee.googlevideo.com
79781 182.905396629 192.168.100.151 192.168.100.1 DNS 93 Standard query 0xd6da A rr4---sn-hc57enee.googlevideo.com
79780 182.905120105 192.168.100.151 192.168.100.1 DNS 93 Standard query 0x09e1 AAAA rr4---sn-hc57enee.googlevideo.com
79626 181.358402296 192.168.100.1 192.168.100.151 DNS 139 Standard query response 0x019a HTTPS google.com HTTPS A 172.2
79625 181.349204273 192.168.100.1 192.168.100.151 DNS 86 Standard query response 0x14e1 A google.com A 172.217.170.206
79624 181.345978596 192.168.100.1 192.168.100.151 DNS 98 Standard query response 0xa788 AAAA google.com AAAA 2a00:1450
79622 181.342850049 192.168.100.151 192.168.100.1 DNS 70 Standard query 0x819a HTTPS google.com
79621 181.342628271 192.168.100.151 192.168.100.1 DNS 70 Standard query 0x14e1 A google.com
79620 181.342387762 192.168.100.151 192.168.100.1 DNS 70 Standard query 0xa788 AAAA google.com
74926 164.657872632 192.168.100.1 192.168.100.151 DNS 139 Standard query response 0xcb04 HTTPS google.com HTTPS A 172.2
74925 164.657588462 192.168.100.1 192.168.100.151 DNS 86 Standard query response 0xfe4b A google.com A 172.217.170.206
74920 164.649212997 192.168.100.1 192.168.100.151 DNS 98 Standard query response 0x1c5b AAAA google.com AAAA 2a00:1450
74919 164.647233208 192.168.100.1 192.168.100.151 DNS 178 Standard query response 0x69d1 HTTPS beacons.gcp.gvt2.com CNA
74918 164.646914851 192.168.100.1 192.168.100.151 DNS 126 Standard query response 0xc97f A beacons.gcp.gvt2.com CNAME b
74917 164.644321847 192.168.100.151 192.168.100.1 DNS 70 Standard query 0xcb04 HTTPS google.com
74916 164.644051526 192.168.100.151 192.168.100.1 DNS 70 Standard query 0xfe4b A google.com
74915 164.643781234 192.168.100.1 192.168.100.151 DNS 138 Standard query response 0xf8d6 AAAA beacons.gcp.gvt2.com CNAM
74913 164.636657886 192.168.100.151 192.168.100.1 DNS 70 Standard query 0x1c5b AAAA google.com
74912 164.634802457 192.168.100.151 192.168.100.1 DNS 80 Standard query 0x69d1 HTTPS beacons.gcp.gvt2.com
74911 164.634268416 192.168.100.151 192.168.100.1 DNS 80 Standard query 0xc97f A beacons.gcp.gvt2.com
74910 164.633736149 192.168.100.151 192.168.100.1 DNS 80 Standard query 0xf8d6 AAAA beacons.gcp.gvt2.com
74830 163.947174809 192.168.100.1 192.168.100.151 DNS 163 Standard query response 0xc945 HTTPS clients4.google.com CNAM
74829 163.939876726 192.168.100.1 192.168.100.151 DNS 129 Standard query response 0xa147 A clients4.google.com CNAME cl
74828 163.937653278 192.168.100.1 192.168.100.151 DNS 141 Standard query response 0xbc77 AAAA clients4.google.com CNAME
74827 163.934281460 192.168.100.151 192.168.100.1 DNS 79 Standard query 0xc945 HTTPS clients4.google.com
74826 163.933994369 192.168.100.151 192.168.100.1 DNS 79 Standard query 0xa147 A clients4.google.com
74825 163.933567193 192.168.100.151 192.168.100.1 DNS 79 Standard query 0xbc77 AAAA clients4.google.com
73853 154.911038100 192.168.100.1 192.168.100.151 DNS 135 Standard query response 0x7fb4 HTTPS ssl.gstatic.com SOA ns1.
73852 154.910748111 192.168.100.1 192.168.100.151 DNS 91 Standard query response 0x1a2c A ssl.gstatic.com A 216.58.223
73851 154.908302976 192.168.100.1 192.168.100.151 DNS 103 Standard query response 0xa823 AAAA ssl.gstatic.com AAAA 2a00
73850 154.896170464 192.168.100.151 192.168.100.1 DNS 75 Standard query 0x7fb4 HTTPS ssl.gstatic.com
73849 154.895978298 192.168.100.151 192.168.100.1 DNS 75 Standard query 0x1a2c A ssl.gstatic.com
Frame 389: 86 bytes on wire (688 bits), 86 bytes captured (688 bi
Ethernet II, Src: Intel_01:62:dd (f0:d5:bf:01:62:dd), Dst: Huawei
Internet Protocol Version 4, Src: 192.168.100.151, Dst: 192.168.1
User Datagram Protocol, Src Port: 9202, Dst Port: 53
Domain Name System (query)
0000 a0 f4 79 c8 dd 8f f0 d5 bf 01 62 dd 08 00 45 00
0010 00 48 21 40 40 00 40 11 cf 7b c0 a8 64 97 c0 a8
0020 64 01 23 f2 00 35 00 34 bd d3 c9 cd 01 00 00 01
0030 00 00 00 00 00 00 0e 65 6e 63 72 79 70 74 65 64
0040 2d 74 62 6e 30 07 67 73 74 61 74 69 63 03 63 6f
0050 6d 00 00 1c 00 01
Domain Name System: Protocol
Packets: 128748 · Displayed: 600 (0.5%) · Dropped: 0 (0.0%)
Profile: Default

TCP



UDP



2. **Follow TCP Stream**

Select any packet from the **TCP** protocol.

Right-click and choose **Follow → TCP Stream**.

This feature will show you the complete conversation between two hosts in a session.
Identify any readable content (especially in unencrypted protocols like HTTP).

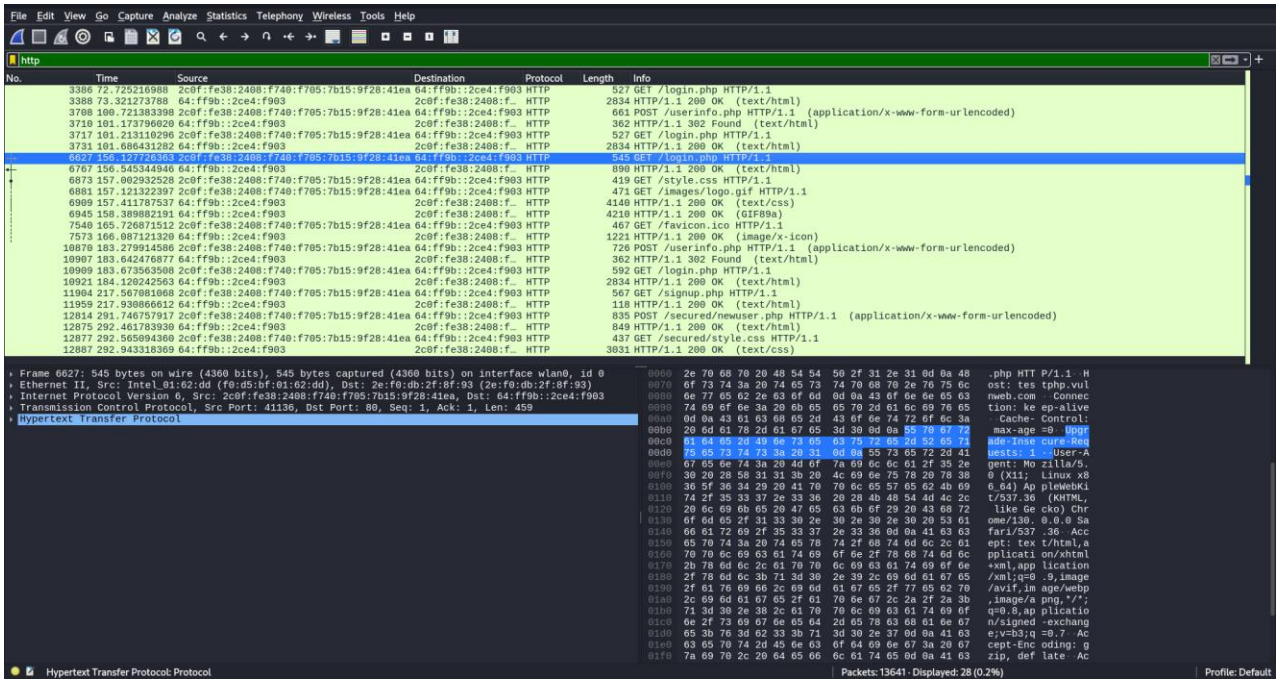
Part 3: Identifying Security Concerns

1. **Unencrypted Data**:

Use the HTTP filter (http) to locate HTTP traffic.

Select a packet and inspect its details in the packet content section.

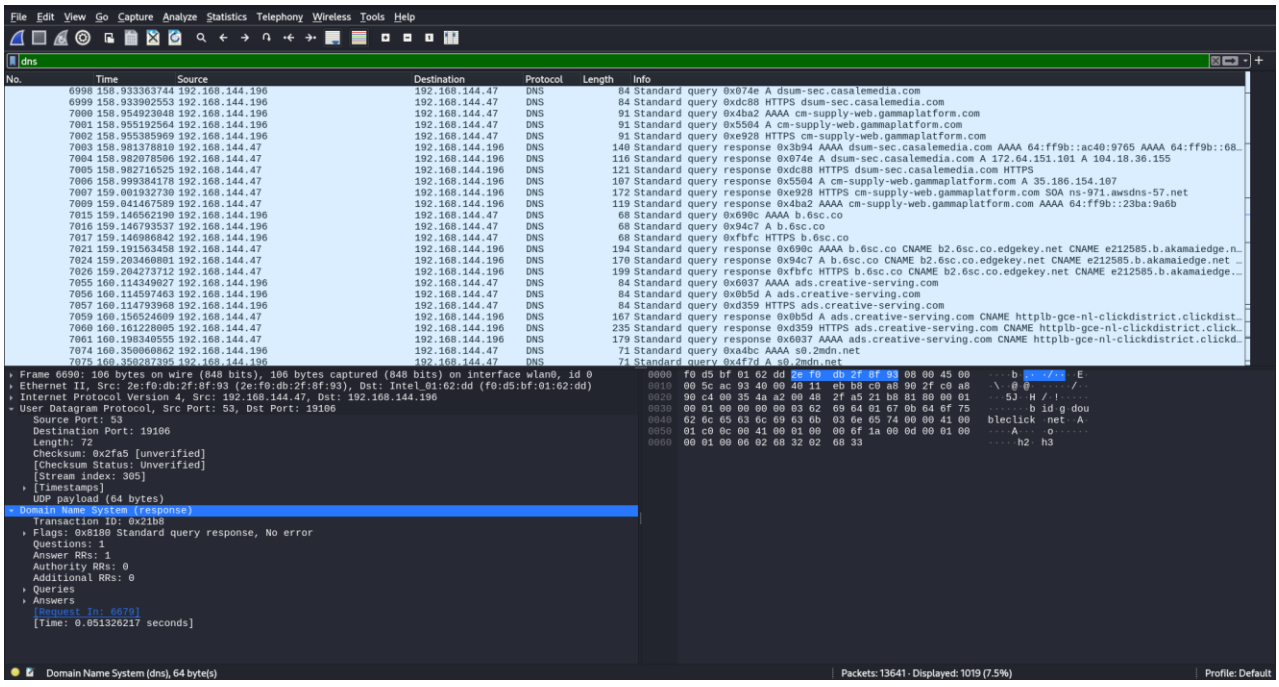
You should be able to see the **request** and **response** sent over HTTP. Look for plain-text data like usernames, passwords, or sensitive information.



2. **DNS Queries**:

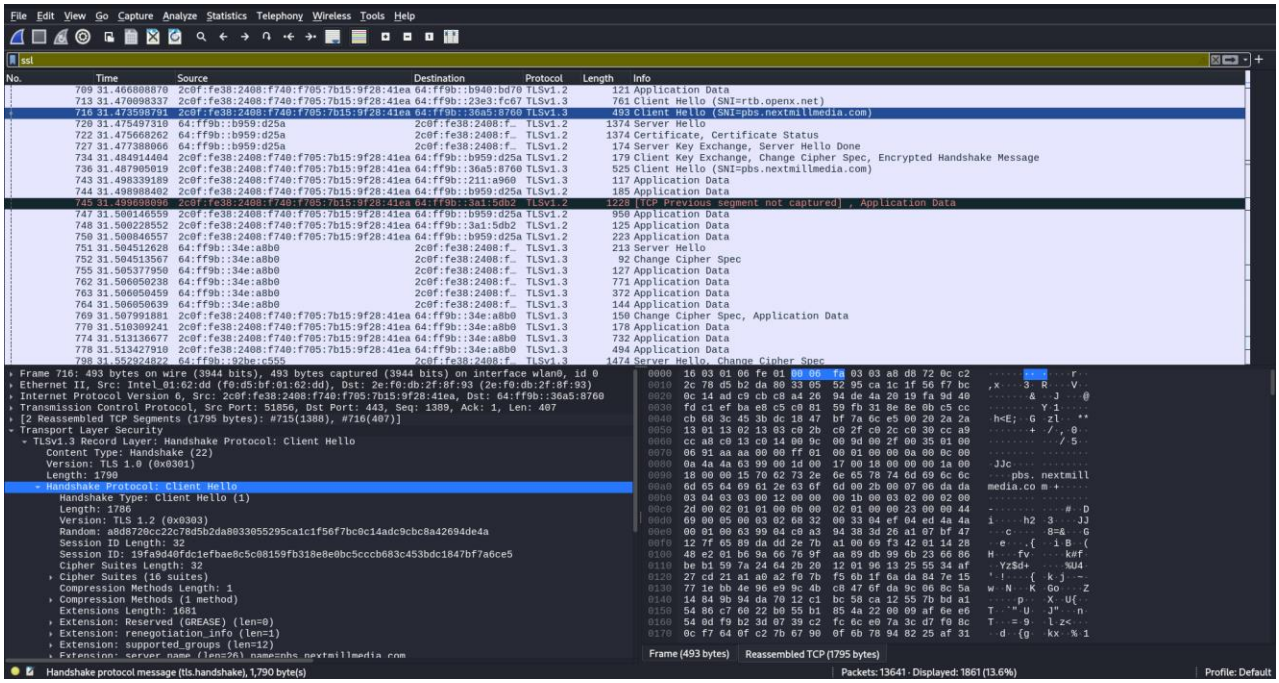
Use the DNS filter (dns) to view DNS query and response traffic.

Analyze what domain names are being requested and to which IP addresses they are resolved.



Use the HTTPS filter (ssl) to find encrypted HTTPS traffic.

Notice the difference from HTTP traffic — you cannot see the actual content of encrypted communication, but you can still analyze the structure of the handshake.



4. **Man-in-the-Middle Vulnerabilities (Discussion)**:

How unencrypted traffic (like HTTP) can be intercepted by attackers.

Unencrypted traffic, like HTTP, is sent in plain text without encryption or security mechanisms, making it vulnerable to interception by attackers. Attackers can exploit this lack of encryption to:

- Read the text in requests and responses to monitor and capture sensitive information.
- Intercept private communications, like login credentials, credit card information, and other sensitive data.
- Record and monitor user interactions with an application, gaining insights into user behavior.
- Use the application as a platform for attacks against users and third-party websites.
- Trick users into entering credentials on unencrypted websites, exposing them to phishing attacks.

Methods of Interception

plain text, sensitive data can be directly read from captured packets.

Man-in-the-Middle (MITM) Attacks - Attackers position themselves between the user and the server, intercepting or altering data as it flows between them. Public Wi-Fi networks are especially vulnerable to MITM attacks, where attackers can capture HTTP data without users knowing.

SSL Stripping - Attackers can downgrade a secure HTTPS connection to an unencrypted HTTP connection by taking advantage of protocol vulnerabilities. This is done by intercepting HTTPS requests and redirecting users to HTTP versions of the site. Non-HTTPS links, insecure redirections, and mixed content on websites can facilitate these attacks.

DNS Spoofing - The attacker manipulates DNS records to redirect users to a malicious site. Since HTTP doesn't authenticate the server, users can unknowingly enter sensitive information on a fake website.

Session Hijacking - Attackers can capture HTTP cookies over unencrypted connections, allowing them to hijack user sessions. This grants unauthorized access to the user's account without needing a password.

Public Wi-Fi Networks - In unsecured public networks, attackers can intercept HTTP traffic with relative ease. Without encryption, all transmitted data, such as usernames and passwords, is accessible to the attacker.

Shared Networks - On a shared network, such as in corporate or home settings, a compromised device can capture unencrypted traffic, potentially exposing other users' sensitive data.

Local and Remote Interception - Malicious software can run locally on the user's device to capture HTTP traffic, while remote interception involves capturing traffic along the network path, often by compromised routers or network nodes.

Mitigation

To prevent these risks, encryption protocols like HTTPS should be used. HTTPS secures traffic using SSL/TLS, encrypting data between the client and server and making it challenging for attackers to intercept or manipulate information.

How HTTPS prevents eavesdropping but also highlight the risk of SSL/TLS attacks (e.g., SSL stripping).

HTTPS prevents eavesdropping by using encryption protocols (SSL/TLS) to secure communication between a client (e.g., a web browser) and a server. Here's how HTTPS works and how it guards against eavesdropping, along with some risks associated with SSL/TLS attacks:

How HTTPS Prevents Eavesdropping

Encryption - When a user connects to a website via HTTPS, SSL/TLS encrypts the data being transmitted. This means that even if an attacker intercepts the data packets, they cannot read the contents because it appears as scrambled, unreadable text without the decryption keys.

Authentication - SSL/TLS uses digital certificates issued by trusted certificate authorities (CAs) to verify the authenticity of the server. This prevents attackers from impersonating a legitimate website, as users can check the certificate to confirm they're connecting to the correct server.

Integrity - HTTPS uses hashing techniques to ensure data integrity. Even if an attacker intercepts the data, any alteration would invalidate the hash, alerting both the client and the server to potential tampering.

Risks of SSL/TLS Attacks

While HTTPS is robust, certain SSL/TLS attacks can weaken or bypass its protections

SSL Stripping - This attack downgrades an HTTPS connection to HTTP. If a user initially connects via HTTP (such as by typing "example.com" without specifying HTTPS), the attacker intercepts the HTTP request, strips the HTTPS requirement, and serves an unencrypted HTTP connection instead. Users then communicate over an insecure connection without realizing, allowing the attacker to eavesdrop on and modify the data. This is why many websites now use HTTP Strict Transport Security (HSTS) to

enforce HTTPS connections.

Man-in-the-Middle (MITM) Attacks on SSL - In some cases, attackers use fake SSL certificates to impersonate legitimate sites. If the user's browser doesn't properly validate the certificate, it may connect to the attacker's server, enabling the attacker to decrypt and read the traffic. This risk is mitigated by keeping browsers and systems updated to detect invalid or revoked certificates.

Protocol Vulnerabilities - SSL/TLS protocols have evolved, and older versions (like SSL 2.0, SSL 3.0, and even TLS 1.0) are known to have vulnerabilities (such as POODLE and BEAST attacks). Attackers exploiting these vulnerabilities can potentially decrypt sensitive data. Modern systems should use TLS 1.2 or higher, as newer versions offer stronger encryption and more robust protections.

Misconfigured Certificates - Insecure configurations, such as using weak ciphers or failing to properly configure HSTS, can leave HTTPS vulnerable to attacks. An attacker could exploit these weaknesses to intercept or modify data.

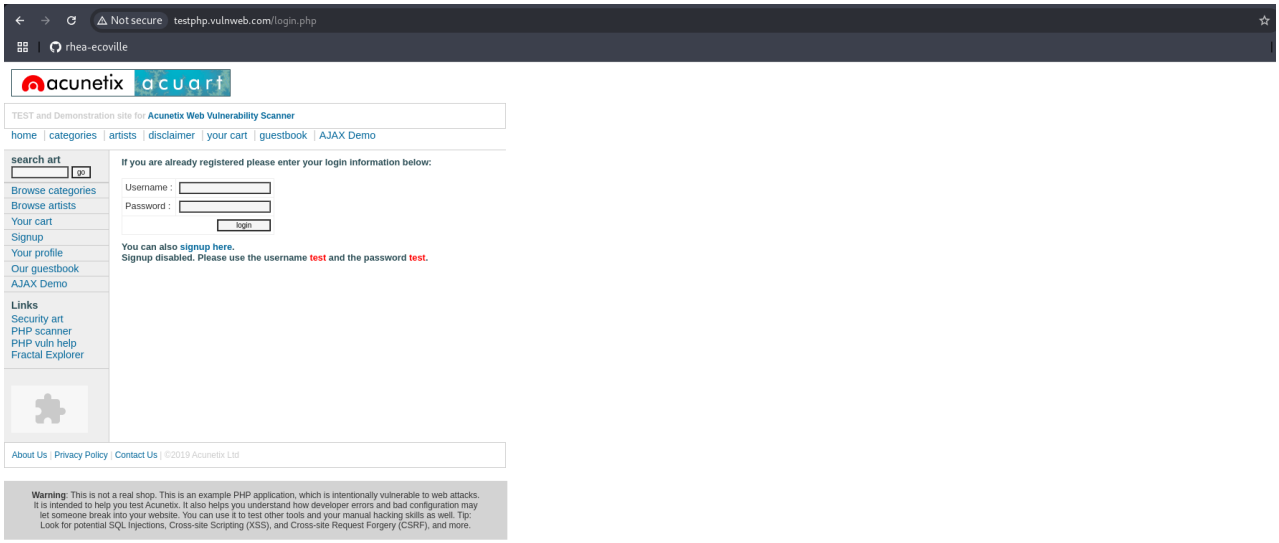
Certificate Authority (CA) Compromise - If a CA is compromised, attackers can issue fraudulent certificates, allowing them to impersonate legitimate sites without detection. This risk is reduced by implementing Certificate Transparency logs and other checks to monitor and flag suspicious certificate.

Part 4: Practical Task – Password Capture (Simulated)

Note: This section should be conducted in a controlled environment to avoid capturing sensitive real-world data.

1. **Simulated Login Capture**:

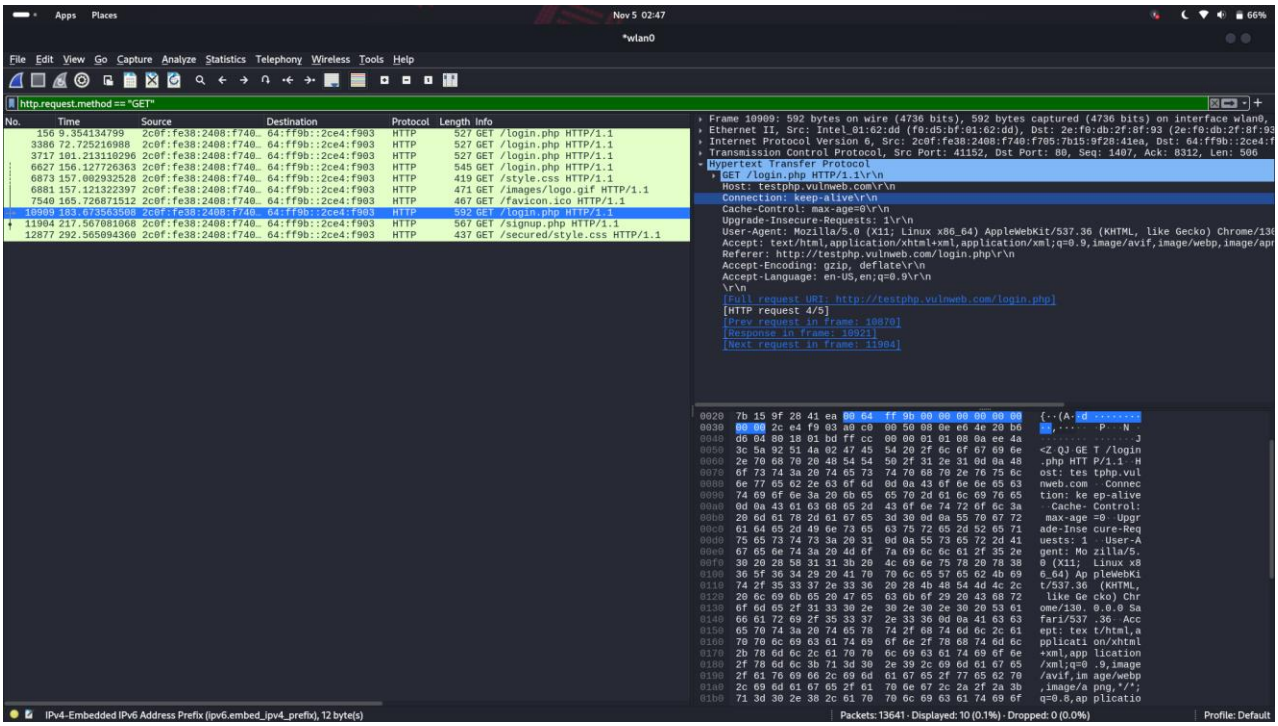
   Open a website or local service that uses plain HTTP for login (set up in a local environment, e.g., a demo website using HTTP).

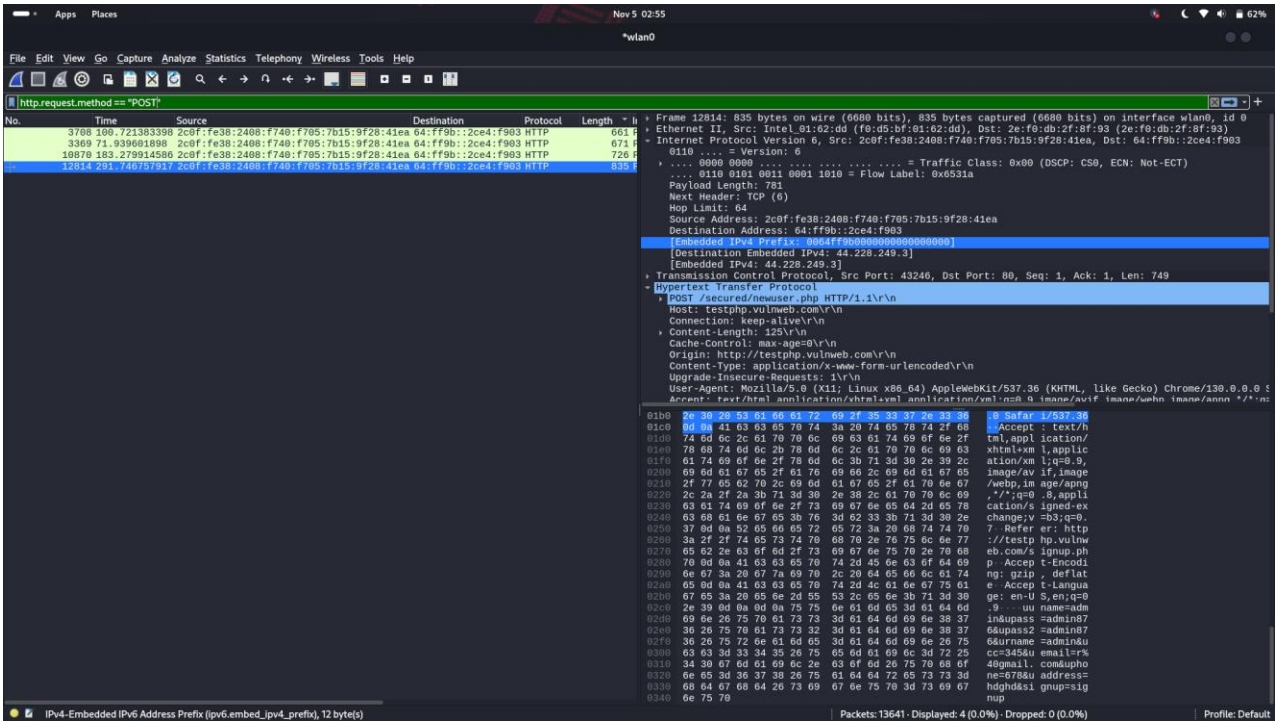


2. **Capture and Analyze the Traffic**:

   In Wireshark, use the filter http to locate the HTTP POST request.

   Look into the packet details and identify the credentials sent in plain text.





Implications of sending credentials over unencrypted channels.

Sending credentials over unencrypted channels, such as HTTP, poses serious security risks, as the lack of encryption leaves sensitive data exposed to interception, modification, and unauthorized access. Here are the key implications:

1. Interception

Credential Theft – Attackers can easily intercept unencrypted traffic and capture sensitive information, including usernames, passwords, and session cookies. This is especially prevalent on unsecured networks like public Wi-Fi or shared corporate networks, where attackers can use packet-sniffing tools to view plain text data directly.

Session Hijacking -Unencrypted transmission of session cookies allows attackers to capture these cookies and impersonate users without needing their login credentials. This grants unauthorized access to users' accounts and any resources tied to those sessions.

2. Modification

Data Tampering - Attackers can intercept and modify the data exchanged between the client and server. This can lead to altered or malicious content being injected into a web session, potentially resulting in compromised functionality or misleading information.

Phishing and Redirects - Attackers may modify responses to redirect users to phishing sites, collecting credentials and sensitive information. Without encryption, users have no way of verifying if they are communicating with the legitimate server.

3. Impersonation

Unauthorized Access - Capturing session cookies or login credentials allows attackers to impersonate legitimate users, gaining full control over accounts and systems. This unauthorized access can result in data breaches, identity theft, and fraud.

Lateral Movement - For users who reuse passwords across multiple accounts, attackers can attempt to use the stolen credentials elsewhere, compromising other systems and applications.

4. Compromised Audit Trails

Obscured Investigation Pathways - When credentials are intercepted and used without authorization, tracing the exact cause of a breach becomes challenging. Obscured audit trails make it difficult to understand the scope of the compromise, complicating remediation and forensic analysis efforts.

5. Risk to Other Users

Reused Passwords - Even if an application only handles non-sensitive data, exposing passwords over unencrypted channels can endanger users who reuse these passwords across different accounts. Attackers can use intercepted credentials to access more critical accounts, amplifying the impact of a breach.

Platform for Further Attacks - Once an attacker gains access to even one user’s credentials, they may leverage that access for phishing attacks, social engineering, or

6. Legal and Compliance Implications

Regulatory Non-Compliance - Many data protection laws and standards (e.g., GDPR, HIPAA, PCI DSS) mandate secure handling of user data, especially credentials. Transmitting credentials over unencrypted channels can result in fines, legal repercussions, and reputational damage.

Breach of Trust and Loss of Reputation - Users expect their data to be secure. If credentials are intercepted and accounts compromised, user trust in the application or organization can erode, potentially leading to customer attrition and negative publicity.

7. Protecting Against These Risks

Implement HTTPS - Enforcing HTTPS with robust TLS configurations encrypts credentials and sensitive data, preventing interception and modification. Using HTTP Strict Transport Security (HSTS) helps ensure that all connections to the server are securely encrypted.

Monitoring for Cleartext Credential Transmission - Regularly monitor traffic for unencrypted credentials, particularly in complex environments, to detect and eliminate any cleartext transmission risks before they can be exploited.

Part 5: Exporting and Reporting

1. **Export Packet Capture**:

   Go to **File → Export Specified Packets** to save the capture file for further analysis or report writing.

2. **Lab Report**:

Network Traffic Analysis and Security Concerns

1. Protocols Captured

In this capture session, the primary protocols identified were:

HTTP - Used for web traffic, including unencrypted login requests.
TCP - A core transport layer protocol providing reliable, ordered delivery of data packets.
UDP - A connectionless transport layer protocol used for applications needing low latency.
DNS - Responsible for translating domain names into IP addresses.

2. Security Concerns Observed

A significant security issue observed in this capture was the transmission of login credentials over plain HTTP. Since HTTP is an unencrypted protocol, all data transmitted, including sensitive information such as usernames and passwords, is sent in clear text. By using the HTTP filter, we were able to locate an HTTP POST request that contained these credentials in plain text, making them easily visible to anyone with access to the network traffic.

This poses several risks:

Credential Theft - An attacker intercepting this traffic could capture and read sensitive login information directly.
Session Hijacking - With access to the login credentials, an attacker could impersonate the user and gain unauthorized access to the website or system.
Data Integrity Risks - Without encryption, attackers could intercept and alter the transmitted data, leading to potential data corruption or manipulation.

3. Importance of Encryption in Network Security

Encryption plays a crucial role in protecting data during transmission by making intercepted data unreadable without proper decryption keys. Protocols like HTTPS (HTTP over TLS/SSL) ensure that sensitive information, such as login credentials, is encrypted, significantly reducing the risk of unauthorized access and data tampering. Implementing HTTPS instead of HTTP is vital for securing login pages, as it protects user data from eavesdropping, MITM attacks, and credential theft.

Conclusion

The use of plain HTTP for transmitting credentials highlights a critical vulnerability. To enhance security, it’s essential to use encrypted protocols, such as HTTPS, for all sensitive data transmissions. Encrypting network traffic helps ensure data confidentiality, integrity, and authenticity, which are fundamental components of network security.

Lab Wrap-Up and Discussion

1. Reflection:

The Importance of Using Tools like Wireshark to Monitor Network Traffic and Risks of Insecure Communications

The Role of Wireshark in Network Monitoring and Security

Monitoring network traffic is vital for maintaining both the security and performance of a network, and Wireshark is one of the most powerful tools for this purpose. Here's why Wireshark is essential:

Visibility into Network Activity

Packet Inspection - Wireshark provides deep visibility into each packet traveling across the network, including details like source and destination IP addresses, ports, and protocols. This allows network administrators to analyze communication patterns and detect unusual activity.

Traffic Analysis - By revealing the types and volume of traffic on the network, Wireshark helps identify if any specific application is consuming excessive bandwidth or if there are spikes that may need investigation.

Troubleshooting Network Issues

Identifying Bottlenecks - By examining packet flows and timing, Wireshark can pinpoint network delays or congestion points, allowing teams to optimize performance.

Error Detection - Wireshark identifies issues like malformed packets, retransmissions, and other network errors, helping to quickly resolve connectivity issues.

Protocol Analysis - Wireshark supports detailed analysis of hundreds of protocols, making it invaluable for troubleshooting protocol-specific problems.

Security Monitoring

Intrusion Detection - With filters and alerts, Wireshark can help detect unauthorized or unusual activity that may indicate a security breach.

Suspicious Patterns - Wireshark can spot signs of malware or unauthorized communication with

Data Exfiltration Detection - By monitoring for unusual outbound data transfers, Wireshark helps prevent unauthorized data from leaving the network.

Compliance and Auditing

Data Flow Documentation - Wireshark assists in documenting data flows and network interactions, which is essential for audits and compliance with security standards.

Policy Enforcement - By analyzing traffic, Wireshark helps ensure that policies (e.g., blocking certain types of traffic) are enforced, enhancing security and adherence to organizational guidelines.

Risks of Insecure Network Communications

If network communications are insecure, attackers can exploit various vulnerabilities. Here are specific ways attackers can target unencrypted or poorly secured network traffic:

Eavesdropping (Sniffing)

Unencrypted Data Interception - Attackers can use packet sniffing tools to capture unencrypted data. Sensitive information such as login credentials or personal information transmitted without encryption is fully exposed, allowing attackers to read it easily.

Credential Theft - Login credentials, session cookies, and other sensitive data can be intercepted. Attackers can use this information to gain unauthorized access, posing serious risks to users and networks.

Man-in-the-Middle (MITM) Attacks

Session Hijacking - Attackers intercept and alter communications between two parties without their knowledge. This allows the attacker to steal information, modify data, or inject malicious content into the communication.

SSL/TLS Stripping - Attackers can downgrade HTTPS connections to unencrypted HTTP, enabling them to view and alter data in transit. SSL stripping is particularly dangerous because it can go unnoticed by users, who may think they are on a secure site.

Token Theft - By capturing session tokens, attackers can hijack user sessions and gain unauthorized access without knowing the user's credentials.

Cookie Poisoning - Attackers may modify session cookies, allowing them to gain unauthorized privileges or impersonate users, leading to data breaches or other security incidents.

Data Manipulation

Packet Injection - Attackers can inject malicious packets into a data stream, which may alter application behavior or corrupt data.

Data Tampering - During transmission, data can be intercepted and modified, potentially leading to financial losses, data breaches, or operational disruptions.

Network Reconnaissance

Mapping the Network - By capturing and analyzing network traffic, attackers can identify network topology, active devices, and the operating systems or services running on them. This information can be used to find vulnerabilities or plan further attacks.

Service Identification - Attackers can identify open ports and services, giving them a blueprint for launching attacks on exposed services.

Importance of Encryption in Network Security

Encryption is critical for protecting data in transit and maintaining network security. Encrypted protocols like HTTPS use SSL/TLS to prevent unauthorized parties from reading or altering data. Here's why encryption is essential:

Data Confidentiality - Encryption ensures that only authorized users can read transmitted data, making it harder for attackers to extract sensitive information.

Data Integrity - By protecting data from tampering, encryption helps maintain data integrity, preventing attackers from modifying or corrupting information.

Authentication and Trust - SSL/TLS certificates provide identity verification, helping users

Wireshark is a powerful tool for monitoring network traffic, troubleshooting issues, and detecting security threats, but it also reveals the importance of secure communications. Insecure protocols expose sensitive data to attackers, making encryption essential for safeguarding information in transit. Using secure protocols like HTTPS and employing network monitoring tools are foundational practices in a robust security strategy.

Resource: https://www.youtube.com/watch?v=a_4MjV_-7Sw

Cisco Packet Tracer

To create a beginner lab in Cisco Packet Tracer that focuses on IPv6 and ICMP, you can configure devices to use IPv6, and then analyze how ICMP works for network diagnostics. The video link is a tutorial on Cisco Packet Tracer, explaining the software's functions, features, and how to use it for networking simulations. It covers setting up basic network devices, configuring routers and switches, creating topologies, and using the simulation mode to visualize packet flows. This video is ideal for beginners looking to understand networking and practice network configurations in a virtual environment.

You can watch it here: https://www.youtube.com/watch?v=qZB_biPOBwA

Part 1: Network Topology Design

Cisco Packet Tracer installed.

Basic understanding of networking concepts like IP addressing, routers, and pings.

Familiarity with the Cisco Packet Tracer environment.

Step 1: Create the Network

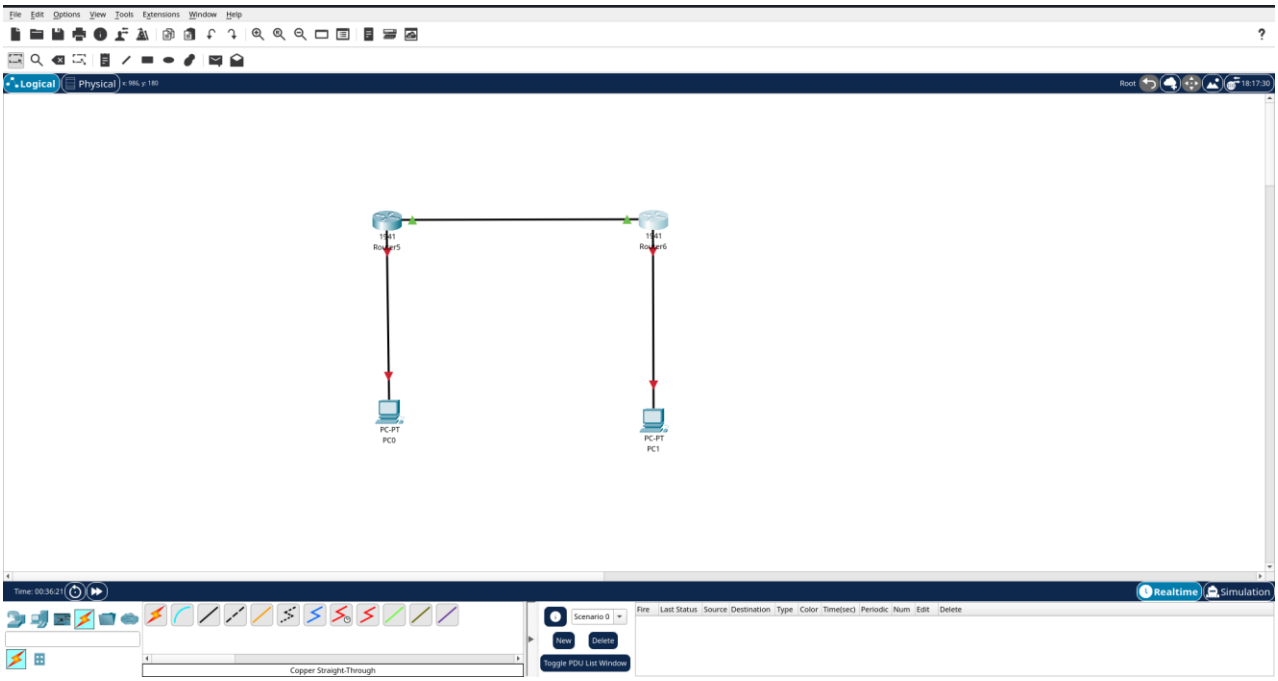
Use two **PCs** (PC0 and PC1) and two **Routers** (Router0 and Router1).

Interconnect the devices with **Ethernet cables**.

PC0 should connect to **Router0**.

PC1 should connect to **Router1**.

Router0 and **Router1** should be connected to each other via a **Serial Connection** (if using older router models) or via **Ethernet** for newer models (GigabitEthernet).



Step 2: Configure Network Settings

Assign IP addresses to the routers and PCs using IPv6.

Part 2: Configuring IPv6 on Routers and PCs

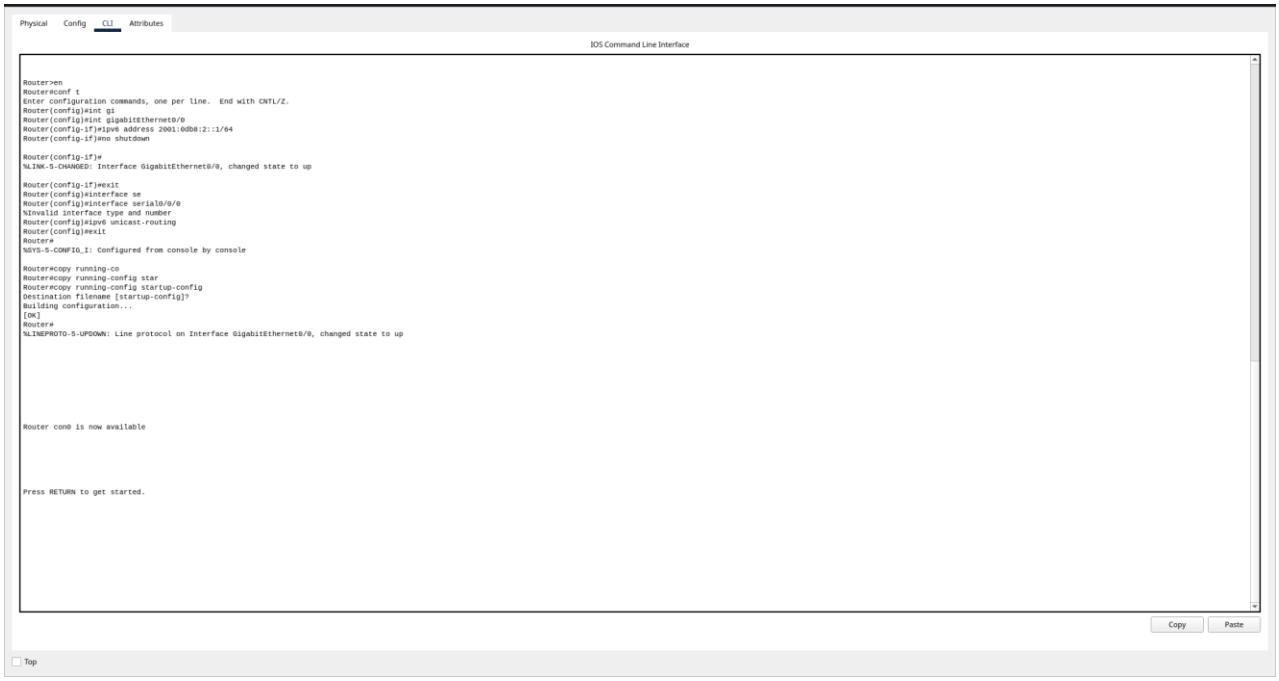
Step 1: Configure Router0 and Router1

1. **Configure Router0:**

   Click **Router0** > **CLI**.

Enter the following commands:

```
Router> enable
Router# configure terminal
Router(config)# interface gigabitethernet0/0
Router(config-if)# ipv6 address 2001:0db8:1::1/64
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# interface serial0/0/0 (for serial interface)
Router(config-if)# ipv6 address 2001:0db8:2::1/64
Router(config-if)# no shutdown
Router(config-if)# exit
Router(config)# ipv6 unicast-routing
Router(config)# exit
Router# copy running-config startup-config
```



1. **Configure Router1:**

Click **Router1** > **CLI**.

Enter the following commands:

Router> enable

Router# configure terminal

Router(config)# interface gigabitethernet0/0

Router(config-if)# ipv6 address 2001:0db8:3::1/64

Router(config-if)# no shutdown

Router(config-if)# exit

Router(config)# interface serial0/0/0 (for serial interface)

Router(config-if)# ipv6 address 2001:0db8:2::2/64

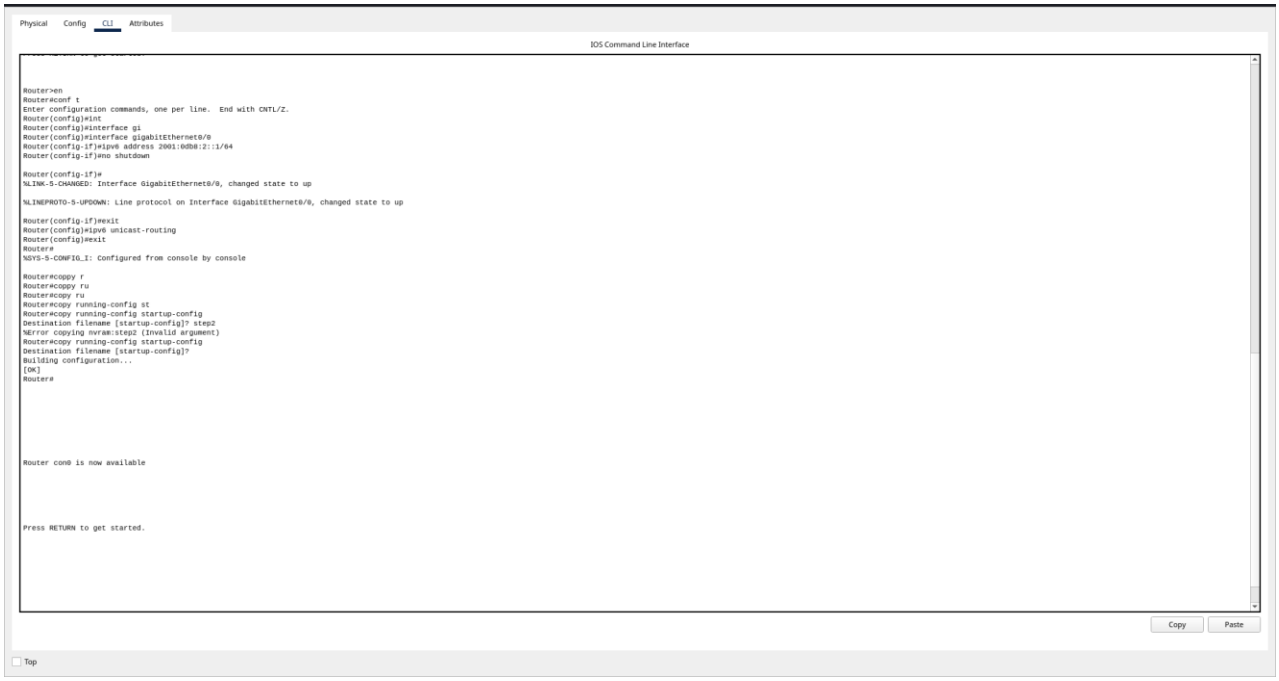
Router(config-if)# no shutdown

Router(config-if)# exit

Router(config)# ipv6 unicast-routing

Router(config)# exit

Router# copy running-config startup-config



Step 2: Configure PCs

1. **Configure PC0:**

   Click **PC0** > **Desktop** > **IP Configuration**.

2001:0db8:1::1.

2. **Configure PC1:**

Click **PC1** > **Desktop** > **IP Configuration**.

Set the IPv6 address to 2001:0db8:3::2/64 and the default gateway to 2001:0db8:3::1.

Part 3: Testing Connectivity with ICMP (Ping)

Step 1: Ping between PCs

Go to **PC0** > **Desktop** > **Command Prompt** and use the following command to ping **PC1**:

ping 2001:0db8:3::2

```
Cisco Packet Tracer PC Command Line 1.0
C:\>ping 2001:0db8:3::2

Pinging 2001:0db8:3::2 with 32 bytes of data:

Reply from 2001:DB8:3::2: bytes=32 time=16ms TTL=128
Reply from 2001:DB8:3::2: bytes=32 time=7ms TTL=128
Reply from 2001:DB8:3::2: bytes=32 time=7ms TTL=128
Reply from 2001:DB8:3::2: bytes=32 time=7ms TTL=128

Ping statistics for 2001:DB8:3::2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 7ms, Maximum = 16ms, Average = 9ms
```

Go to **PC1** and ping **PC0**:

ping 2001:0db8:1::2

```
C:\>ping 2001:0db8:1::2

Pinging 2001:0db8:1::2 with 32 bytes of data:

Reply from 2001:DB8:1::2: bytes=32 time=21ms TTL=128
Reply from 2001:DB8:1::2: bytes=32 time=7ms TTL=128
Reply from 2001:DB8:1::2: bytes=32 time=6ms TTL=128
Reply from 2001:DB8:1::2: bytes=32 time=6ms TTL=128

Ping statistics for 2001:DB8:1::2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 6ms, Maximum = 21ms, Average = 10ms

C:\>
```

Expected Result: The ping should succeed if the routing and interfaces are correctly configured.

Task:

Test the ping to verify connectivity between the devices.

Troubleshoot any issues that might occur (e.g., no connection due to incorrect IP addresses or interfaces not being enabled).

Part 4: Viewing and Analyzing ICMPv6 Traffic

Step 1: Capture ICMPv6 Traffic

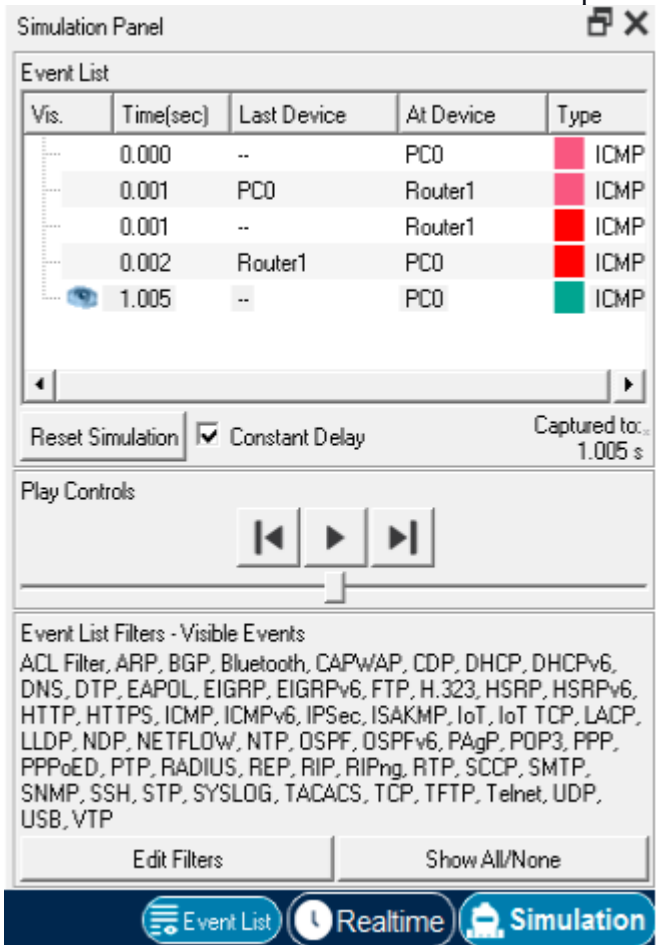
Cisco Packet Tracer allows you to capture and view packets.

Click on the **Simulation** tab at the bottom.

Choose **ICMP** from the filters on the right side.

Start a new ping from **PC0** to **PC1** or vice versa.

Watch the simulation and observe the ICMP packets.



Step 2: Analyze the Packet

Click on one of the captured ICMP packets to open it.

Review the packet contents, specifically the ICMP message types (128 for Echo Request, 129 for Echo Reply).

Task:

Fields in the ICMPv6 packet (Type, Code, Checksum).

In an ICMPv6 packet, the following fields are fundamental:

1. Type - This field identifies the specific ICMPv6 message type. Different types correspond to various functions within ICMPv6. For example:

   128: Echo Request, used to initiate a diagnostic ping.

   129: Echo Reply, used to respond to an Echo Request.

2. Code - The Code field provides additional context for the message type. It helps to specify the exact nature or purpose of the message within the type. For Echo Request and Echo Reply messages, the Code field is always set to 0, meaning there are no subtypes or extra conditions within these specific message types.

3. Checksum - The Checksum field is used for error-checking. This field contains a calculated value based on the ICMPv6 header and data payload, which allows the receiving device to detect any errors that might have occurred in transmission. The sender calculates this checksum and includes it in the packet. The receiver recalculates it upon receipt; if the values match, the packet is considered error-free.

Function of an ICMP Echo Request and Reply.

The ICMP Echo Request (Type 128) and Echo Reply (Type 129) messages are used primarily for diagnostic purposes, particularly in

network tools like ping.

Echo Request (Type 128)

The Echo Request message is sent by a device (the sender) to another device on the network to check connectivity. The message includes information such as an identifier and sequence number to help match requests with replies.

When a device (like a router, server, or PC) receives an Echo Request, it understands that another device is asking if it's reachable and, if it is, it should respond with an Echo Reply.

Echo Reply (Type 129)

When a device receives an Echo Request, it sends back an Echo Reply message to confirm receipt. The Echo Reply contains the same identifier and sequence number as the request, which helps the original sender match the reply to the request.

This reply signals that the target device is accessible and that packets can be sent to it successfully, confirming network connectivity.

Together, these messages help diagnose network issues by:

- Confirming whether a device is reachable (if the Echo Reply is received).
- Allowing measurement of round-trip time (by calculating the delay between sending the Echo Request and receiving the Echo Reply)

- Identifying packet loss (if no reply is received, it may indicate connectivity issues).

This functionality makes ICMPv6 Echo Request and Reply messages crucial for network troubleshooting and monitoring.

Part 5: Traceroute and ICMP Error Messages (Optional)

Step 1: Traceroute

On **PC0**, run the tracert command to trace the path to **PC1**:

tracert 2001:0db8:3::2

Observe how ICMP Time Exceeded messages are generated at each hop.

Task:

How traceroute uses ICMP messages to map the route between devices. Traceroute is a network diagnostic tool used to trace the path packets take to reach a destination, identifying each “hop” along the route and measuring the time taken at each step. Here’s a detailed look at how it does this using ICMP (Internet Control Message Protocol) messages:

Initial Packet Sending

Traceroute initiates the process by sending out a series of ICMP Echo Request messages to the target device, similar to the "ping" command. These messages request responses from each router along the path to the destination, allowing traceroute to map the route

Time-To-Live (TTL) Field

Each packet has a TTL (Time-To-Live) field in its IP header, which represents the maximum number of hops (routers or nodes) the packet is allowed to traverse before being discarded.

For the first set of packets, the TTL is set to 1, meaning it will expire after just one hop. With each subsequent set of packets, traceroute increments the TTL by 1, exploring each new hop along the path one step further.

Hop-by-Hop Exploration

When a router receives a packet, it decrements the TTL by 1. If the TTL reaches zero, the router discards the packet and sends back an ICMP Time Exceeded message to the sender.

This ICMP Time Exceeded message includes the IP address of the router, which allows traceroute to record it as one of the hops in the path to the destination.

Traceroute also measures the round-trip time (RTT) by calculating the time it takes to receive the Time Exceeded message. This helps identify latency at each step.

Incrementing TTL

After each hop, traceroute sends a new set of packets with an increased TTL, allowing the packets to reach one hop further than before. This continues until traceroute either reaches the destination or the maximum TTL value.

At each hop, the router sends back an ICMP Time Exceeded message, allowing traceroute to map out the network route sequentially.

Destination Reached

When the packet finally reaches the destination, the device does not return an

device responds with an ICMP Echo Reply message, confirming it as the endpoint.

Traceroute records this as the final hop, completing the path mapping.

Information Provided by Traceroute

For each hop, traceroute displays:

The IP address (or hostname) of the router or device at each hop.

The round-trip time (RTT) for each packet.

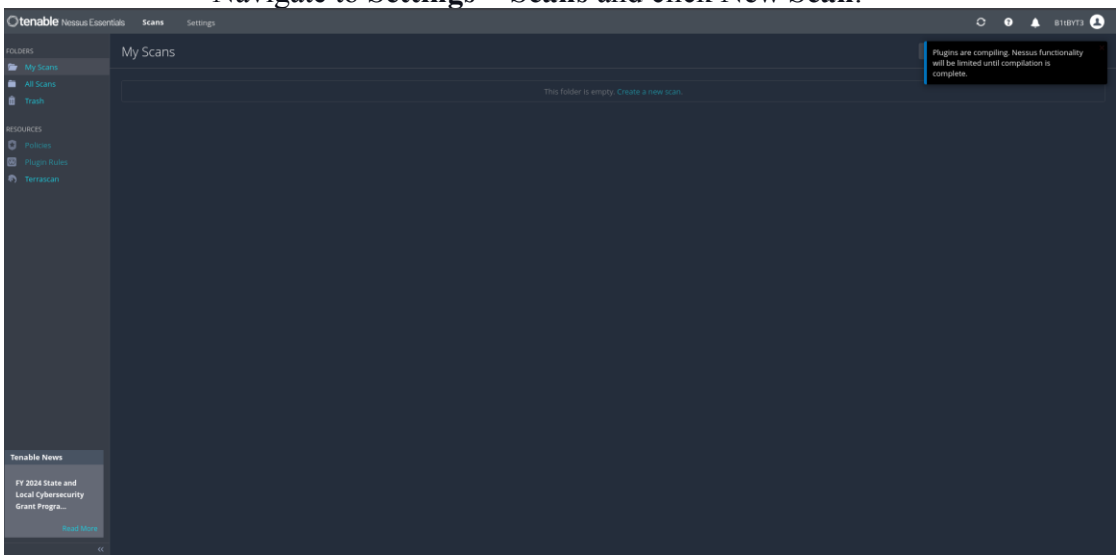
This information helps network administrators diagnose issues, understand latency across each hop, and identify any failing or slow routers along the path.

Lab Steps: Vulnerability scanning

Part 1: Setup Nessus

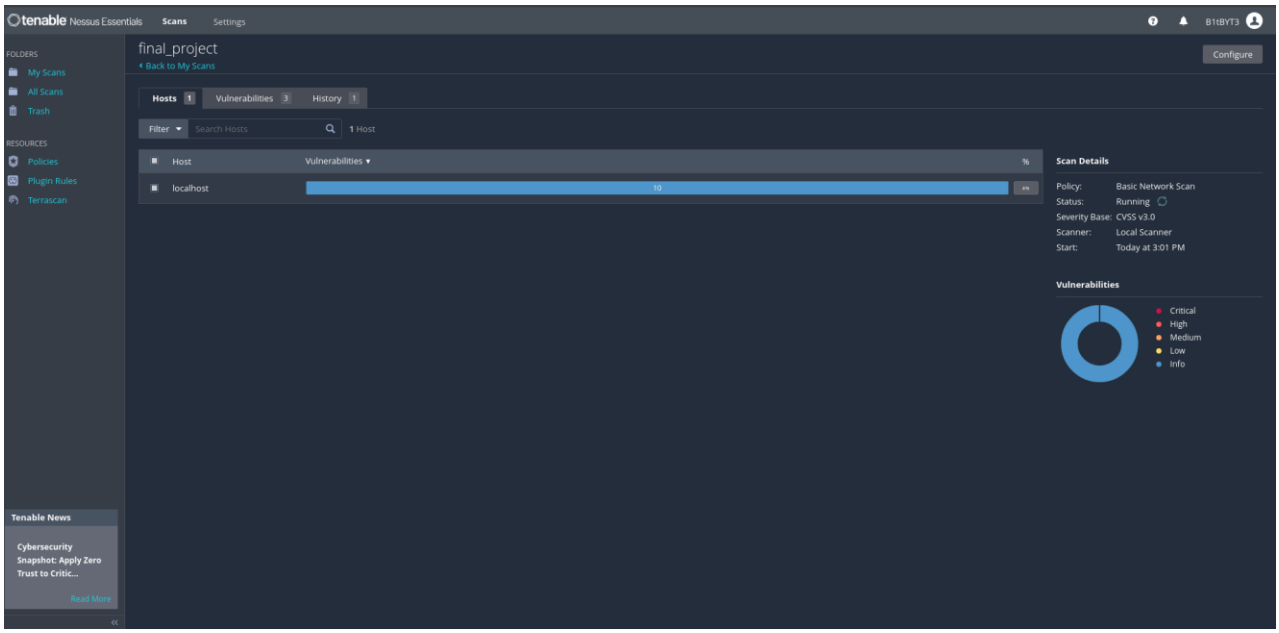
1. **Install Nessus: https://www.youtube.com/watch?v=hRQRg7SB1Aw**
    - Download and install Nessus from the Tenable website.

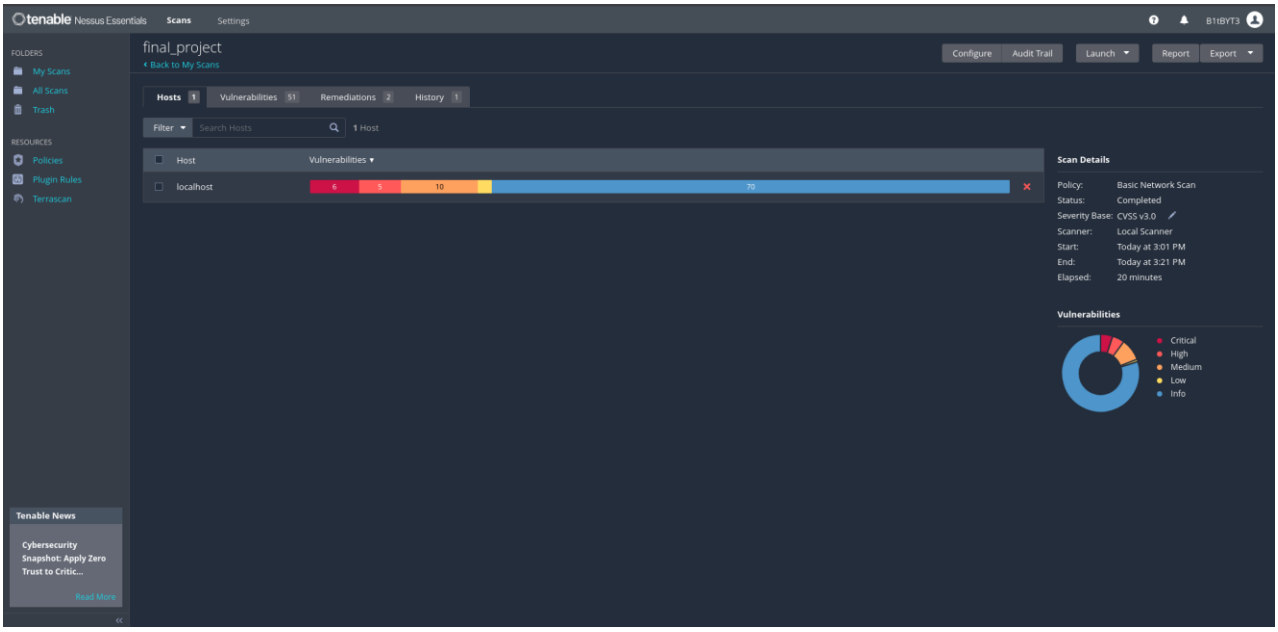
2. **Configure Nessus:**
    - Start Nessus and log into the web interface.
    - Navigate to **Settings** > **Scans** and click **New Scan**.



Part 2: Run a Scan

1. In Nessus, set up a **Basic Network Scan**.
    - Target: (localhost or specific individual IPs).
    - Leave default settings, or choose specific scan types (e.g., vulnerability scan).
2. **Start the Scan** and monitor progress.





Part 3: Analyze Results

1. **Review Vulnerabilities:**

    - Once the scan completes, review the report.
    - Categorize findings (Critical, High, Medium, Low).

localhost

4	4	7	1	60
CRITICAL	HIGH	MEDIUM	LOW	INFO

Vulnerabilities — Total: 76

SEVERITY	CVSS V3.0	VPR SCORE	EPSS SCORE	PLUGIN	NAME
CRITICAL	9.8	7.4	0.0046	182874	libcurl 7.69 < 8.4.0 Heap Buffer Overflow
CRITICAL	9.1	6.0	0.0004	201086	OpenSSL 1.0.2 < 1.0.2zk Vulnerability
CRITICAL	9.1	6.0	0.0004	201085	OpenSSL 3.0.0 < 3.0.15 Vulnerability
CRITICAL	10.0	-	-	182259	OpenSSL SEoL (1.0.2.x)
HIGH	8.8	6.7	0.0004	209152	OpenSSL 1.0.2 < 1.0.2zl Vulnerability
HIGH	8.8	6.7	0.0004	209150	OpenSSL 3.0.0 < 3.0.16 Vulnerability
HIGH	7.8	6.7	0.0004	181289	OpenSSL 3.0.0 < 3.0.11 Vulnerability
HIGH	7.5	5.0	0.0011	183891	OpenSSL 3.0.0 < 3.0.12 Multiple Vulnerabilities
MEDIUM	6.5	5.0	0.0023	185160	OpenSSL 3.0.0 < 3.0.13 Multiple Vulnerabilities
MEDIUM	6.5	-	-	51192	SSL Certificate Cannot Be Trusted
MEDIUM	6.5	4.4	0.0006	205024	libcurl 7.32.0 < 8.9.1 DoS (CVE-2024-7264)
MEDIUM	5.9	3.6	0.0004	192966	OpenSSL 3.0.0 < 3.0.14 Multiple Vulnerabilities
MEDIUM	5.5	4.4	0.0023	184812	OpenSSL 1.0.2 < 1.0.2zj Multiple Vulnerabilities
MEDIUM	5.3	2.2	0.0014	178476	OpenSSL 1.0.2 < 1.0.2zi Multiple Vulnerabilities
MEDIUM	4.8	4.4	0.0004	197900	Intel Media SDK Multiple Vulnerabilities (INTEL-SA-00935)
LOW	3.7	2.2	0.0012	182873	libcurl 7.9.1 < 8.4.0 Cookie Injection
INFO	N/A	-	-	141394	Apache HTTP Server Installed (Linux)
INFO	N/A	-	-	156000	Apache Log4j Installed (Linux / Unix)
INFO	N/A	-	-	45590	Common Platform Enumeration (CPE)

localhost 4

2. **Document Vulnerabilities**:

Critical Vulnerabilities

libcurl 7.69 < 8.4.0 Heap Buffer Overflow

CVE ID - Not listed

Description - A heap buffer overflow vulnerability in libcurl affecting versions from 7.69 to below 8.4.0.

Solution - Update libcurl to version 8.4.0 or later.

OpenSSL 1.0.2 < 1.0.2zk Vulnerability

CVE ID - Not listed

Description - Multiple vulnerabilities in OpenSSL version 1.0.2 up to (but not including) 1.0.2zk.

OpenSSL 3.0.0 < 3.0.15 Vulnerability

CVE ID - Not listed

Description - Multiple vulnerabilities in OpenSSL versions 3.0.0 up to (but not including) 3.0.15.

Solution - Upgrade OpenSSL to version 3.0.15 or later.

OpenSSL SEOL (1.0.2.x)

CVE ID - Not listed

Description - OpenSSL 1.0.2 has reached end of life, meaning it no longer receives updates or patches.

Solution - Upgrade to a supported version of OpenSSL.

High Vulnerabilities

OpenSSL 1.0.2 < 1.0.2zl Vulnerability

CVE ID - Not listed

Description - Security vulnerabilities in OpenSSL versions below 1.0.2zl.

Solution - Update to OpenSSL version 1.0.2zl or later.

OpenSSL 3.0.0 < 3.0.16 Vulnerability

CVE ID - Not listed

Description - Security vulnerabilities in OpenSSL versions below 3.0.16.

Solution - Update to OpenSSL version 3.0.16 or later.

OpenSSL 3.0.0 < 3.0.11 Vulnerability

CVE ID - Not listed

Description - Multiple vulnerabilities in OpenSSL versions below 3.0.11.

Solution - Update to OpenSSL version 3.0.11 or later.

OpenSSL 3.0.0 < 3.0.12 Multiple Vulnerabilities

CVE ID - Not listed

Description - Multiple vulnerabilities in OpenSSL versions below 3.0.12.

Solution - Update to OpenSSL version 3.0.12 or later.

Medium Vulnerabilities

OpenSSL 3.0.0 < 3.0.13 Multiple Vulnerabilities

CVE ID - Not listed

Description - Multiple vulnerabilities in OpenSSL versions below 3.0.13.

Solution - Update to OpenSSL version 3.0.13 or later.

SSL Certificate Cannot Be Trusted

CVE ID - Not listed

Description - The SSL certificate is untrusted, potentially compromising secure connections.

Solution - Replace the SSL certificate with one from a trusted certificate authority.

libcurl 7.32.0 < 8.9.1 DoS (CVE-2024-7264)

CVE ID - CVE-2024-7264

Description - A Denial of Service (DoS) vulnerability affecting libcurl versions below 8.9.1.

Solution - Update libcurl to version 8.9.1 or later.

OpenSSL 3.0.0 < 3.0.14 Multiple Vulnerabilities

CVE ID - Not listed

Description - Multiple vulnerabilities in OpenSSL versions below 3.0.14.

Solution - Update to OpenSSL version 3.0.14 or later.

OpenSSL 1.0.2 < 1.0.2z Multiple Vulnerabilities

CVE ID - Not listed

Description - Multiple vulnerabilities in OpenSSL versions below 1.0.2z.

Solution - Update to OpenSSL version 1.0.2z or later.

OpenSSL 1.0.2 < 1.0.2zi Multiple Vulnerabilities

CVE ID - Not listed

Description - Vulnerabilities in OpenSSL versions below 1.0.2zi.

Solution - Update to OpenSSL version 1.0.2zi or later.

Intel Media SDK Multiple Vulnerabilities (INTEL-SA-00935)

CVE ID - Not listed

INTEL-SA-00935.

Solution - Apply Intel-provided patches or updates for this SDK version.

Low Vulnerabilities

libcurl 7.9.1 < 8.4.0 Cookie Injection

CVE ID - Not listed

Description - Vulnerability in libcurl versions below 8.4.0 that could allow cookie injection.

Solution - Update libcurl to version 8.4.0 or later.

Informational

Apache HTTP Server Installed (Linux)

CVE ID - Not applicable

Description - Detection of Apache HTTP Server installed on Linux, which may need updates or configuration checks for security.

Solution - Ensure Apache is up-to-date and configured securely.

Apache Log4j Installed (Linux / Unix)

CVE ID - Not applicable

Description - Log4j is installed, and prior versions (below 2.17.1) had significant vulnerabilities.

Solution - Check the version of Log4j and update to the latest secure version if needed.