ANALYZING VULNERABILITIES IN NETWORK PROTOCOLS USING WIRESHARK: A CASE STUDY ON HTTP AND HTTPS

BY

Idris Olanrewaju Ibraheem, Saka Kamil Kayode & Nurudeen Olawale Ibrahim Al-Hikmah University Ilorin Nigeria

Abstract

In the digital age, securing network communications is critical, with HTTP and HTTPS protocols being integral to web data transfer. Despite HTTPS providing enhanced security through TLS encryption, both protocols exhibit vulnerabilities that can be exploited by attackers. This study explores how Wireshark, a powerful network protocol analyzer, can be utilized to detect and mitigate these vulnerabilities. Through the examination of HTTP traffic and the decryption of HTTPS traffic, the research identifies common weaknesses such as session hijacking, man-in-the-middle attacks, and information leakage. The analysis includes inspecting the TLS handshake process and the security of various cipher suites, highlighting the potential risks associated with outdated or improperly configured protocols. A methodology involving the capture and detailed analysis of network traffic using Wireshark is presented, supported by case studies on several websites. The findings underscore the necessity of robust encryption practices, regular security audits, and proper certificate management to enhance the security of network communications. By adopting these strategies, organizations can significantly reduce their exposure to potential vulnerabilities, ensuring the protection of sensitive information and maintaining the trust of users.

Keywords: Cyber Attacks, Network Security, Wireshark, Transport Layer Security, HTTP, HTTPS

Introduction

In today's digital age, the security of network communications is paramount. HTTP and HTTPS are widely used protocols for data transfer on the web, but they are not without vulnerabilities. Wireshark, a powerful network protocol analyzer, offers robust tools for identifying and mitigating these security issues. This article examines how Wireshark can be employed to analyze vulnerabilities in HTTP and HTTPS protocols, providing insights into the potential risks and the methods to enhance network security (Patel et al, 2021). The widespread use of HTTP as a protocol for web communication has made it a prime target for cyber-attacks. Despite the advent of more secure protocols like HTTPS, HTTP remains prevalent, especially in older systems and legacy applications (Muraleedharan, et al, 2020). Analyzing HTTP traffic to identify vulnerabilities is crucial for understanding potential threats and mitigating risks. Wireshark, a powerful network protocol analyzer, provides extensive capabilities for examining HTTP traffic and uncovering common vulnerabilities Patel et al, (2021). HTTP (Hypertext Transfer Protocol) is the foundation of data communication on the World Wide Web, known for its simplicity and speed but lacking inherent security measures. HTTPS (HTTP Secure) builds on HTTP by adding a layer of security via TLS (Transport Layer Security), encrypting data to prevent eavesdropping and tampering (Danezis, 2009). Despite these protections, both protocols can exhibit vulnerabilities due to improper implementation or configuration, making them attractive targets for attackers. Wireshark allows security professionals to capture and analyze traffic in these protocols, helping to uncover and address potential security flaws (Lavrenovs, & Melón, 2018).

This study aims to:

- i. identify and analyze common vulnerabilities in HTTP traffic using Wireshark.
- ii. examine the TLS handshake process in HTTPS and identify potential weaknesses.
- iii. demonstrate the decryption of HTTPS traffic in Wireshark for vulnerability analysis.
- iv. propose mitigation strategies for the vulnerabilities identified in HTTP and HTTPS protocols.

_ (IJARMS) 737

Literature Review

Recent studies conducted by Banerjee et al, (2010) have demonstrated the effectiveness of Wireshark in network traffic analysis and security assessment. Wireshark's capabilities in detecting security anomalies, such as abnormal port usage and rogue hosts, are critical for network defense. In a study titled Performance and Security Evaluation of TLS, DTLS and QUIC Security Protocols by Gaminara, (2022) highlights Wireshark's role in decrypting HTTPS traffic, revealing potential misconfigurations and cryptographic weaknesses during the TLS handshake process. Research by Toolify.ai underscores the tool's utility in extracting clear text credentials from HTTP traffic, emphasizing the need for secure communication protocols. HTTPS (HyperText Transfer Protocol Secure) is the secure version of HTTP, achieved by layering HTTP on top of the SSL/TLS protocol, providing encrypted communication and secure identification of a network web server. However, vulnerabilities such as misconfigured servers, outdated protocols, and weak encryption can undermine this security (Aslan et al, 2023).

Furthermore, in a study titled Proving the TLS handshake secure (as it is) by Béguelin et al (2014), the TLS (Transport Layer Security) handshake is fundamental to the security of HTTPS, ensuring encrypted communication between clients and servers. This process involves several steps, including the exchange of cryptographic keys and the establishment of a secure connection. However, despite its critical role in secure communications, the TLS handshake has potential weaknesses that need to be understood and mitigated. TLS 1.2, which was widely used before the introduction of TLS 1.3, has several known vulnerabilities primarily due to its flexibility and support for older, less secure cryptographic algorithms. The RSA key exchange algorithm used in TLS 1.2 is a notable example. It lacks Perfect Forward Secrecy (PFS), meaning that if an attacker obtains the server's private key, they can decrypt past sessions.

The primary weakness in the TLS 1.2 handshake process lies in the use of outdated cryptographic algorithms and insufficient default configurations. For instance, the RSA key exchange, although popular, uses the same key pair for both authentication and encryption of the pre-master secret. This dual usage can lead to significant security risks if the private key is compromised. Attacks such as BEAST (Browser Exploit Against SSL/TLS) and Lucky 13 exploit vulnerabilities in the CBC (Cipher Block Chaining) mode of encryption used in TLS 1.2 Béguelin et al (2014). However, a study titled A cryptographic analysis of the TLS 1.3 handshake protocol by Dowling et al (2021), TLS 1.3, released in 2018, addresses many of these vulnerabilities by removing support for older algorithms and streamlining the handshake process. It reduces the handshake to a single round trip, thereby minimizing the attack surface. TLS 1.3 also mandates the use of forward secrecy, ensuring that session keys cannot be compromised even if the server's long-term key is compromised. This is achieved using ephemeral Diffie-Hellman (DHE) or Elliptic Curve Diffie-Hellman (ECDHE) key exchanges, which generate unique keys for each session. Despite these improvements, TLS 1.3 is not without its potential weaknesses. The complexity of the protocol and the need for widespread adoption can introduce implementation flaws. Additionally, while TLS 1.3 enhances security, it still relies on the security of the underlying public key infrastructure (PKI). If certificate authorities (CAs) are compromised, the entire chain of trust can be undermined.

Moreso, a study by (Dodiya & Singh 2022), titled Malicious Traffic analysis using Wireshark by collection of Indicators of Compromise in the International Journal of Computer Applications highlights the effectiveness of Wireshark in identifying Indicators of Compromise (IoC) within HTTP traffic. The research demonstrates how packet analysis can uncover nefarious activities such as unauthorized data access and malware. Similarly, Silvestre et al, (2023) explores the dual-edged nature of packet sniffing with tools like Wireshark. It underscores the potential for both security enhancement and misuse, emphasizing the need for ethical and legal frameworks to govern the use of such tools. The importance of analyzing HTTP traffic to identify vulnerabilities is underscored by Patel et al (2021), who successfully detected cyberattacks in real-time by monitoring and analyzing network traffic. This is further emphasized by Tran (2018), who developed an application for monitoring and analyzing HTTP communications to detect threats. However, the potential risks of HTTP traffic analysis are highlighted by Danezis (2009) who discussed the information that can be inferred from HTTP transactions over TLS. To address these risks, (Huang et al, 2024)

recommended optimizing the use of web vulnerability standards, such as OWASP Top 10 and CWE, to ensure a higher level of security.

Common Vulnerabilities in HTTP Traffic

HTTP, being a plain text protocol, is susceptible to various attacks that can compromise data integrity, confidentiality, and availability. Key vulnerabilities often identified in HTTP traffic as described by Muraleedharan et al (2020) which include:

Session Hijacking: Attackers can exploit the lack of encryption in HTTP to capture session cookies and hijack user sessions. This vulnerability is particularly prevalent in environments where HTTP is used without proper session management techniques.

Man-in-the-Middle (MitM) Attacks: Without encryption, HTTP traffic can be intercepted and altered by attackers positioned between the client and server. This can lead to data tampering and unauthorized access to sensitive information.

Information Leakage: HTTP headers and URLs can inadvertently expose sensitive information such as software versions, internal IP addresses, and other configuration details that can be exploited by attackers to map out the network.

Cross-Site Scripting (XSS): HTTP-based applications often fail to properly sanitize user inputs, leading to the injection of malicious scripts. These scripts can be executed in the context of the user's browser, leading to data theft and session hijacking.

Methodology

Using Wireshark to Analyze HTTP Vulnerabilities

Wireshark provides a robust platform for capturing and analyzing HTTP traffic. By inspecting packet details, security analysts can identify patterns and signatures indicative of the vulnerabilities. Decrypting HTTPS traffic in Wireshark is a powerful technique for analyzing the security of encrypted communications. HTTPS encrypts data using TLS (Transport Layer Security) to protect it from interception and tampering. However, for security analysis and troubleshooting, it is often necessary to decrypt this traffic. This section provides an overview of the process and potential vulnerabilities identified through this method.

Decrypting HTTPS traffic for vulnerability analysis is a critical task in cybersecurity, allowing analysts to inspect and understand encrypted communications. This process is essential for identifying potential security issues within the traffic, such as weak cipher suites, improper certificate validation, or the exposure of sensitive information. The use of Wireshark to decrypt HTTPS traffic leverages the ability to capture session keys, enabling the decryption of encrypted packets. This method is particularly useful for diagnosing problems and verifying the security of applications that rely on HTTPS.



Figure 1: Methodological Process

The image depicts a flowchart outlining the steps involved in capturing and analyzing HTTPS traffic using Wireshark. The process consists of the following sequential steps.

The use of Wireshark to capture HTTPS traffic from the network, configuration a key log file to capture encryption keys for HTTPS decryption, utilization of the key log file in Wireshark to decrypt the captured HTTPS traffic. Examine the decrypted traffic to understand the data being transmitted and identify for potential security vulnerabilities within the decrypted traffic.

Websites	Security Status	Packets
website 1	Weak cipher	79
website 2	Secure	767
website 3	Secure	236
website 4	Secure	311
website 5	Improved Security	359
website 6	Outdated and less secure	119
website 7	Insecure	197
website 8	Improved Security	401
website 9	Secure	215
website 10	Improved Security	602
website 11	Improved Security	436
website 12	Improved Security	278

Table 1: Websites checked for vulnerabilities

_ (*IJARMS*) 740

website 13	Insecure	327
website 14	Outdated and less secure	76
website 15	Insecure	131
website 16	Outdated and less secure	81
website 17	Weak cipher	83

A total number of 17 websites were analyzed based on SSLv2, SSLv3, TLSv1, TLSv2, TLSv3 protocols with the security status metrics being insecure, weak cipher, outdated and less secure, secure, and improved security, a total number of 4696 packets were captured, the figures of packets captured on each protocol based on the security status of the website are depicted using a bar chart.



Figure 2: SSL Key Log

The SSL key log image shows the content of an SSL/TLS secrets log file generated by NSS (Network Security Services). The file was used for decryption of traffic and further analysis using Wireshark. The key elements of the log file are

CLIENT_HANDSHAKE_TRAFFIC_SECRET: These secrets are used during the TLS handshake to encrypt and authenticate the initial messages between the client and the server.

SERVER_HANDSHAKE_TRAFFIC_SECRET: These secrets are similar to the client handshake traffic secrets but are used by the server.

CLIENT_TRAFFIC_SECRET_0 and SERVER_TRAFFIC_SECRET_0: These secrets are used for the encryption and authentication of the application data after the handshake is complete.

CLIENT_RANDOM: This is a random value generated by the client and is part of the initial handshake process to ensure the security of the session.

EXPORTER_SECRET: This is used to derive additional keys for other purposes during the session.

Each line in the file corresponds to a different secret or random value used in the SSL/TLS session, allowing tools like Wireshark to decrypt the encrypted traffic if the log file is provided. During network traffic analysis, this log file is imported into Wireshark to decrypt and inspect the encrypted data packets.

NewHttpvuln.pcapng						
File Edit View Go Capture Analyze Statistics Telephony Wireless Jools Help						
▲ ■ ▲ ② ▲ ◆ ③ ◆ 本 ◆ 雪 ◆ み ■ ■ の の の m						
📕 tis						🗵 📼 🔹 + SSLv2
No. Time		Destination	Protocol	Length Info		
87707 490, 332738	149,154,165,96	192,168,1,109	SSLv2	1294 Encrypted Data		
87708 490.333190	192.168.1.109	52.182.143.215	TLSv1.3	89 Application Data		
87726 490.351674	149.154.165.96	192.168.1.109	SSLv2	1294 Encrypted Data		
87731 490.363960	149.154.165.96	192.168.1.109	SSLv2	1294 Encrypted Data		
87752 490.383195	192.168.1.109	52.182.143.215	TLSv1.3	138 Application Data		
87753 490.383283	192.168.1.109	52.182.143.215	TLSv1.3	910 Application Data		
87760 490.386919	149.154.165.96	192.168.1.109	SSLV2	1294 Encrypted Data		
87778 490 418776	149.154.105.90	192.100.1.109	SSLV2	1294 Encrypted Data		
87800 490 493546	149.154.165.96	192.168.1.109	SSLv2	1294 Encrypted Data		
87814 499, 528827	149,154,165,96	192,168,1,109	SSLv2	1294 Encrypted Data		
87829 490.553103	149.154.165.96	192.168.1.109	SSLv2	1294 Encrypted Data		
87866 490.594127	149.154.165.96	192.168.1.109	SSLv2	1294 Encrypted Data		
87886 490.619425	149.154.165.96	192.168.1.109	SSLv2	1294 Encrypted Data		
87894 490.621948	149.154.165.96	192.168.1.109	SSLv2	1294 Encrypted Data		
87930 490.673653	149.154.165.96	192.168.1.109	SSLv2	1294 Encrypted Data		
87951 490.690384	149.154.165.96	192.168.1.109	SSLV2	1294 Encrypted Data		
07972 490.729570	149.154.105.90	192.100.1.109	SSLV2	1294 Encrypted Data		
87988 490 744900	149.154.165.96	192.168.1.109	SSL v2	1294 Encrypted Data		
87998 498,774549	149,154,165,96	192,168,1,109	SSLv2	1294 Encrypted Data		
88001 490.778559	149.154.165.96	192.168.1.109	SSLv2	1294 Encrypted Data		
88007 490.788957	149.154.165.96	192.168.1.109	SSLv2	1294 Encrypted Data		
88016 490.808581	149.154.165.96	192.168.1.109	SSLv2	1294 Encrypted Data		
88028 490.827170	149.154.165.96	192.168.1.109	SSLv2	1294 Encrypted Data		
88057 490.868445	149.154.165.96	192.168.1.109	SSLv2	1294 Encrypted Data		
88077 490.907132	149.154.167.91	192.168.1.109	SSL	159 Continuation Data		
88885 490.918823	149.154.165.96	192.168.1.109	SSLV2	1294 Encrypted Data		
88100 490.930315	149.154.105.90	192.108.1.109	SSLV2	1294 Encrypted Data		
88121 499 955551	149 154 165 96	192 168 1 109	SSLv2	1294 Encrypted Data		
88148 490,984057	149.154.165.96	192.168.1.109	SSLv2	1294 Encrypted Data		
88150 490.985669	2.18.108.226	192.168.1.109	TLSv1.3	78 [TCP Fast Retransmission] , Application Data		
88152 490.986200	149.154.165.96	192.168.1.109	SSLv2	1294 Encrypted Data		
4						•
Ename 87753: 918 byt	es on wire (7280 bi	ts), 918 bytes cantur	red (7288 bit	ts) on interface \Device\NPE {59953EC4-97E4-46C7-BBED-73AC7BA	3 0000 98 a9 42 2e 75 be 20 1e 88 6f d6 64 08 00 45 00	
Ethernet II, Src: In	tel 6f:d6:64 (20:1e	:88:6f:d6:64), Dst: G	iuangzhouToz	2e:75:be (98:a9:42:2e:75:be)	0010 03 80 57 38 40 00 80 06 00 00 c0 a8 01 6d 34 b6	- W88
Internet Protocol Ve	rsion 4, Src: 192.1	68.1.109, Dst: 52.182	2.143.215		0020 8f d7 d5 b8 01 bb 50 26 8c 97 7b 3a 2c 11 50 18	·····P& ··{:, ·P·
Transmission Control	Protocol, Src Port	: 54712, Dst Port: 44	3, Seq: 716	\$, Ack: 7006, Len: 856	0030 01 fc 8a 15 00 00 17 03 03 03 53 bd ff 2b c9 99	
Transport Layer Secu	rity				0040 45 22 6a ec 48 35 05 a1 8f 66 02 5e 93 94 dd cd	E"j-H5f-^
 TLSv1.3 Record La 	yer: Application Da	ta Protocol: Hypertex	it Transfer F	Protocol	0050 76 c1 f2 0c 7c 02 74 84 02 60 c8 57 3b d2 91 ad	V···· t···· W;····
Opaque Type: A	pplication Data (23)				0000 6C 2a 2C 94 50 15 60 64 5T 03 6T 80 76 8T 33 94	1, 9, -0, V 3
Version: TLS 1	.2 (0x0303)				0070 80 10 70 33 12 05 04 85 00 19 e1 01 50 08 00 64	· · · · · · · · · · · · · · · · · · ·
Encrypted Appl	ication Data (trunca	stadl: bdff2bc0004522	6aec483505a1	#f66825x0394ddcd76c1f28c7c8274848268c8573bd291ad6c2a2c9438156	4 0000 h0 cf ab 78 26 d5 af 67 dc f6 85 66 41 ab fd 97	(gran
[Application D	ata Protocol: Hypert	text Transfer Protoco			00x0 94 86 fd 17 4e 4f d4 75 58 68 cf 4a 88 94 62 1f	····NO-u Xh-1h-
					0050 2d 85 4a 9e f4 54 64 6e d6 16 be 72 03 bd 0a 93	
					00c0 ab f0 86 f0 cb 95 fc fe 4b c0 cc 71 c9 5d bb 1d	····· K··q·]··
					00d0 cc 2b 0d 0c 80 eb 43 8c 57 c9 99 b4 56 1f 04 ae	
					00e0 8f fa 8c ca 9c a3 81 75 cb 6d 49 35 12 ba 91 fb	
					0010 58 5c 06 79 4d e6 93 ad e5 21 96 bt 2c 1b 83 1e	X\.yH
					0100 14 e8 9a 63 /e 6T 19 C2 91 82 53 69 28 96 42 69	C+0 · · · 51(-8
					0120 1a 99 fd dc 21 41 b7 31 Sb 87 c3 6a 25 81 db 95	
					0130 43 b6 16 f9 68 69 b6 24 c2 03 b7 6e 75 ae 26 28	C···hi·\$ ···nu-\$(
					0140 64 c9 c9 a1 73 28 a5 54 be f6 dd 06 59 d1 41 9d	d···s(·T····Y·A
					0150 2b e7 76 df 5c 12 8e 2b 5c cf 9a 82 03 c6 ce 6e	
4					0160 35 57 12 d8 c4 40 97 ef f0 8b 46 0c 54 ab 35 11	

Figure 3: SSLv2 Without Cipher Suite

The figure 3, showcases the process of decrypting and analyzing the captured HTTP/HTTPS traffic within Wireshark. The view emphasizes the encrypted nature of the traffic, consistent with the goal of examining network security and decrypting HTTPS data for further analysis. The analysis further shows the selected packet has no cipher suite which makes it an unsecure packet which can be intercepted and redirected for an attack.

d apture2.pagng					
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Hep					
	≞⊠ ≌ < ← ⇒	• 🗟 🕯 🕹 📃 📃 '	x q q II		
📕 tis					
No. Time	Source	Destination	Protocol Length Info		
3901 6.362735	108.159.102.27	10.0.0.8	TLSv1.3 1494 Continuation Data		
3907 6.363317	108.159.102.27	10.0.0.8	TLSv1.3 1494 [TCP Previous segment not captured] , Continuation Data		
3915 6.364116	108.159.102.27	10.0.0.8	TLSv1.3 1494 Continuation Data		
3916 6.364116	108.159.102.27	10.0.0.8	TLSV1.3 469 Continuation Data		
3920 6.364702	102.132.101.10	10.0.0.8	HTTP2 1446 DATA[1]		
3921 6,364702	102.132.101.10	10.0.0.8	TLSv1.3 1446 [TLS segment of a reassembled PDU]		
3935 6.369403	102.132.101.10	10.0.0.8	TLSv1.3 1446 [TLS segment of a reassembled PDU]		
3936 6.369403	102.132.101.10	10.0.0.8	HTTP2 1446 DATA[1]		
3938 6.370213	102.132.101.10	10.0.0.8	TLSv1.3 1446 [TLS segment of a reassembled PDU]		
3956 6.379634	102.132.101.10	10.0.0.8	TLSv1.3 1446 [TLS segment of a reassembled PDU]		
3959 6.380193	102.132.101.10	10.0.0.8	HTTP2 1446 DATA[1]		
3960 6.380867	102.132.101.10	10.0.0.8	ILSV1.3 1446 [ILS segment of a reassembled PDU]		
3967 6.382188	102.132.101.10	10.0.0.8	HITP2 1257 DATA[1] (application/x-javascript)		
3981 6 412872	10.0.0.0	210.30.223.230	OUTC 1292 Initial, DCID=cfc4284891102905, PKN: 1, CRIPIO OUTC 1292 Initial DCID=cfc4284891fbe9d3 PKN: 2 CPVPTO CPVPTO	PTING PADDING DING DING CRYPTO PADDING CRYPTO PADDING DING CRYPTO	
3988 6 432422	216.58.223.238	10.0.0.8	OUTC 1292 Initial, SCID=efc4284091fbe9d3, PKN: 2, CRVPTO, PADDING	FING, FADDING, FING, FING, CKIFIG, FADDING, CKIFIG, FADDING, FING, CKIFIG	
3989 6.432422	216.58.223.238	10.0.0.8	OUTC 1292 Initial, SCID=efc4284091fbe9d3, PKN: 3, CRYPTO, PADDING		
4018 6,487011	10.0.0.8	216,58,223,238	TLSv1.3 1875 Client Hello (SNI=www.voutube.com)		
4020 6.487722	18.165.160.7	10.0.0.8	HTTP2 1494 HEADERS[1]: 200 OK, DATA[1]		
4034 6.499472	18.165.160.7	10.0.0.8	HTTP2 1494 DATA[1], DATA[1], DATA[1]		
4060 6.527610	216.58.223.238	10.0.0.8	QUIC 1292 Handshake, SCID=efc4284091fbe9d3, PKN: 4, CRYPTO		
4061 6.527610	216.58.223.238	10.0.0.8	QUIC 1292 Handshake, SCID=efc4284091fbe9d3, PKN: 5, CRYPTO		
4062 6.527610	216.58.223.238	10.0.0.8	QUIC 1292 Handshake, SCID=efc4284091fbe9d3, PKN: 6, CRYPTO		
4065 6.528118	216.58.223.238	10.0.0.8	HTTP3 887 Protected Payload (KP0), PKN: 8, STREAM(3), SETTINGS		
4067 6.529382	10.0.0.8	216.58.223.238	HIIPS 205 Protected Payload (KPO), DCID=etC4284091TDe903, PKN: 10	, ACK, SIREAM(2), SEILINGS	
4075 0.545145	210.50.225.250	10.0.0.0	TISH1 2 1404 Application Data		
4145 6.600470	18 165 160 7	10.0.0.0	TLSv1.3 1494 Application Data		
4165 6.611878	216,58,223,238	10.0.0.8	TLSv1.3 1466 Server Hello, Change Cipher Spec		
4185 6.620021	18.165.160.7	10.0.0.8	TLSv1.3 1494 Application Data		
4					
Transport Laver Se	curity			A 0080 90 78 13 01 00 04 66 00 33 04 64 63 99 04 60 at .x	
TLSv1.3 Record	Laver: Handshake Pro	otocol: Server Hello		0090 a1 b0 01 2d da 64 4d b3 35 59 e2 46 3b 23 05 d6dM. 5Y.F:#	
Content Type	: Handshake (22)			00a0 33 2a df 63 fd 4c 39 d8 7f 0a d5 18 60 df 02 b8 3*·c·L9· ····	
Version: TLS	1.2 (0x0303)			00b0 79 06 91 13 df 21 5d 8f ec 19 18 02 ed 70 ba 92 y····!]· ····p··	
Length: 1210				00c0 5d 8c 53 ef 4b 9a 1c 9e d6 68 57 cc ad 70 58 c6	
 Handshake Pr 	otocol: Server Hello			00d0 06 37 f1 60 70 1d e9 64 f2 e3 be a8 09 6c f0 5d ·7·`p··d ·····l·]	
Handshake	Type: Server Hello			00e0 71 6a d1 7b 1e 93 98 e4 7e ad 7e 77 89 8f 3a 91 qj.{ ~.~w:	
Length: 1	206			0010 11 22 4c 12 04 6c 16 11 3a 01 b2 86 tb 76 40 a6 ."L.1	
Version: TLS 1.2 (0x0303)				0100 c1 18 c0 /8 ac 00 c5 9c 22 d2 36 b3 ac 35 ac 35x	
Kalluba: u2004a0/10054c0ca0u04a54c009453aa010654/051140/55cC505C102001a05				0110 U1 60 /1 52 36 04 60 45 01 35 80 /2 60 65 CC 46 (477) 1 85(177)	
Session TD: 2-BAPAJAF88F8PF30P3d91565ec126fc980de95fd0e60ed1f0cfh123dfdd9078				0130 b6 b2 43 ba a2 c8 35 61 83 52 06 68 24 a1 65 18 cC 5a	
Cipher Suite: TIS AES 128 GCN SH4256 (9X1301)				0140 4e 7f ee 86 11 04 aa 69 55 ad bd e5 0c 53 f1 2c Ni US.	
Compression Method: null (0)				0150 fc bc 69 35 be 75 f7 50 7e a9 61 13 3b a2 10 4di5.u.P ~.a.;M	
Extensions Length: 1134				0160 fc f5 f3 7a d4 d7 83 bf a7 cc 65 cf 17 83 f4 a9z	
Extension	: key_share (len=112	24) X25519Kyber768Draft	90	0170 8c 05 a0 51 0e 66 62 d3 78 a3 32 48 86 18 9d 28 ···Q·fb· x·2H···(
 Extension: supported versions (len=2) TLS 1.3 				0180 51 4b 03 21 b7 56 1e 54 84 0a 3d 70 07 04 d6 7c QK ·!·V·T ··=p···	
[JAS5 Fullstring: //I,4865,51-43]				0190 e3 27 22 06 c3 28 bb 7a ca 38 c4 7d 7c 1e 28 98 ·/"··(·z ·8·)]·(·	
[JASS: ebid94dba3/eb34459/e75ba1fbbe/054] * TIS: 2 Record Lynon, Charge Charge Rectargl, Charge Cichen Spor				0180 21 27 28 97 60 28 71 50 Ca a5 C7 50 00 19 58 a3 ['(·m(·] ······X·	
TLSVIJS RECORD Layer: Change Cipher Spec Protocol: Change Cipher Spec Contant Tune: Change Cipher Spec (20)				0100 55 00 22 73 07 67 54 52 60 47 68 50 24 50 02 50 1.0.48 (0.=\$[.]	
Version: TIS 1.2 (9x8983)				01d0 9c d5 86 61 30 42 ab 66 51 29 5e c6 5f e9 bd fe	
Length: 1				01e0 40 ff e9 0b 9a cc ad 97 1c 5f d7 68 34 c4 08 a3 0	
Change Ciphe	r Spec Message		01f0 6c 43 e4 b6 48 b9 ba 33 61 50 12 fe a9 54 b1 ae 1C-H3 aPT		
TLS segment dat	a (191 bytes)			0200 1b 0c 6e e8 65 cf 28 0f 7f 57 9b d8 34 31 0d 2d ••••••(•••W••41••	
4				0210 29 ef a0 38 73 fa b8 3e 1d be 90 05 ee 66 0b 9c)…8s…>f…	
🥚 🕅 🛛 Cinher Suite (tlr	handchake cinhercuite) 2	hyter		Packets: 44633 . Displayed: 4445 (10.0%)	

Figure 4: TLSv1.3 Cipher Decryption

As seen in figure 3, which showcases the process of decrypting and analyzing the captured HTTP/HTTPS traffic within Wireshark. The view emphasizes the encrypted nature of the traffic, consistent with the goal of examining network security and decrypting HTTPS data for further analysis. The analysis further shows the selected packet has cipher suite which makes it a secure packet which will be hard to be intercepted and redirected for an attack. It further shows the communication between the server and client and the changing of cipher specification, which shows it is a secured connection.



Figure 5: HTTP/HTTPS security vulnerability chart

The image as depicted in figure 5 compares the number of packets associated with different security protocols which are SSLv3, SSLv2, TLSv1, TLSv1.2, and TLSv1.3. The increasing number of packets from SSLv3 to TLSv1.3 can be interpreted as an indicator of the evolving landscape of HTTP/HTTPS security, where newer protocols, despite being more secure, also attract more attention and scrutiny from security researchers. The older protocols like SSLv2 and SSLv3 have fewer packets, likely because they are less used today due to their known vulnerabilities and have been largely replaced by more secure versions of TLS. The significant jump in packets from TLSv1.2 to TLSv1.3 reflects the adoption and ongoing analysis of the latest TLS protocol.

The bar chart highlights the security vulnerabilities associated with different HTTP/HTTPS protocols, showing an upward trend in the number of packets as the protocols evolve. This could reflect both the increasing security measures and the extensive use and examination of these protocols in modern secure communications.



Figure 6: Packets based on security status

The image provided in figure 6, categorizes packets according to their security status from insecure, outdated and less secure, weak cipher, secure, and improved security. The chart clearly shows a progression in the number of packets from insecure to improved security statuses, with the highest counts in the more secure categories. This distribution suggests that most of the network traffic is adopting secure and improved security measures, with fewer packets remaining in the less secure or outdated categories. The presence of packets in the insecure and outdated categories indicates that while security is improving, there are still areas that require attention and updates.

The bar chart provides an insightful representation of the distribution of packets based on their security status. It highlights the ongoing trend towards improved security in network communications, while also indicating the necessity to address the remaining instances of insecure and outdated protocols to enhance overall security.

Mitigating HTTPS Traffic Vulnerabilities

To effectively mitigate vulnerabilities in HTTPS traffic, it is essential to use robust encryption practices and proper configuration of security protocols. One of the primary steps is to ensure the use of strong, modern cipher suites. Outdated or weak cipher suites can be easily compromised, so configuring your servers to support only the latest, most secure cipher suites is critical. Regularly updating your TLS configurations and disabling support for deprecated versions of protocols like SSL and older TLS versions is also necessary. Additionally, implementing HTTP Strict Transport Security (HSTS) ensures that browsers always connect via HTTPS, preventing protocol downgrade attacks and enhancing overall security. Another key aspect of mitigation involves proper certificate management. This includes ensuring that all certificates are issued by trusted Certificate Authorities (CAs) and that they are regularly monitored and renewed before expiration. Implementing Certificate Pinning can further enhance security by binding a certificate or public key to specific servers, which helps to prevent MITM attacks. Clients should always validate server certificates, and organizations should use Online Certificate Status Protocol (OCSP) and Certificate Revocation Lists (CRL) to check the validity of certificates in real-time. By focusing on these strategies, you can significantly reduce the risk of vulnerabilities in HTTPS traffic.

Summary

The study examines the vulnerabilities present in HTTP and HTTPS protocols, emphasizing the critical role of network security in the digital age. It utilizes Wireshark, a network protocol analyzer, to identify and analyze common security issues such as session hijacking, man-in-the-middle attacks, and information leakage. The study highlights the importance of the TLS handshake process, and the risks associated with outdated or misconfigured cipher suites in HTTPS. Through traffic capture and analysis, the research underscores the necessity of robust encryption practices, regular security audits, and proper certificate management to enhance the security of network communications. Ultimately, the findings advocate for proactive measures to protect sensitive information and maintain user trust in online environments.

Conclusion

Ensuring the security of HTTPS traffic is crucial for protecting sensitive information transmitted over the internet. By adopting strong encryption practices, properly managing certificates, and staying vigilant with regular security audits and updates, organizations can significantly reduce their exposure to potential vulnerabilities. Educating all stakeholders about the importance of security measures and keeping systems up to date are equally important. By following these recommendations and implementing comprehensive security strategies, you can create a robust defense against the myriad of threats targeting HTTPS communications, safeguarding your data and maintaining the trust of your users.

Recommendations

Conduct frequent security audits and penetration tests to identify and address vulnerabilities in your HTTPS configurations and the use of automated tools for initial assessments and manual testing for more in-depth analysis. Implementation of robust patch management process to ensure that all software, including web servers, libraries, and dependencies, are kept up to date with the latest security patches. Employing the IDPS to monitor network traffic and detect potential threats. Employ anomaly-based detection to identify unusual patterns indicative of a security breach.

References

- Aslan Ö, Aktuğ SS, Ozkan-Okay M, Yilmaz AA, Akin E. (2023): A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions. Electronics. 2023 Mar 11;12(6):1333. https://doi.org/10.3390/electronics12061333
- Banerjee, U., Vashishtha, A., & Saxena, M. (2010): Evaluation of the Capabilities of WireShark as a tool for Intrusion Detection. International Journal of computer applications, 6(7), 1-5. https://doi.org/10.5120/1092-1427
- Bhargavan, K., Fournet, C., Kohlweiss, M., Pironti, A., Strub, P. Y., & Zanella-Béguelin, S. (2014): Proving the TLS handshake secure (as it is). In Advances in Cryptology–CRYPTO 2014: 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part II 34 (pp. 235-255). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-44381-1_14
- Danezis, G. (2023). Traffic Analysis of the HTTP Protocol over TLS.
- Dodiya, B., & Singh, U. K. (2022): Malicious Traffic analysis using Wireshark by collection of Indicators of Compromise. International Journal of Computer Applications, 183(53), 1-6. https://doi.org/10.5120/ijca2022921876
- Dowling, B., Fischlin, M., Günther, F., & Stebila, D. (2021): A cryptographic analysis of the TLS 1.3 handshake protocol. Journal of Cryptology, 34(4), 37. https://doi.org/10.1007/s00145-021-09384-1
- Gaminara, A. (2022): Performance and Security Evaluation of TLS, DTLS and QUIC Security Protocols (Doctoral dissertation, Politecnico di Torino). http://webthesis.biblio.polito.it/id/eprint/25561
- Huang, J., Zhang, J., Wang, Q., Han, W., & Zhang, Y. (2024): Exploring Advanced Methodologies in Security Evaluation for LLMs. arXiv preprint arXiv:2402.17970. https://doi.org/10.48550/arXiv.2402.17970
- Lavrenovs, A., & Melón, F. J. R. (2018): HTTP security headers analysis of top one million websites. In 2018 10th International Conference on Cyber Conflict (CyCon) (pp. 345-370). IEEE. https://doi.org/10.23919/CYCON.2018.8405025
- Muraleedharan, N., Thomas, A., Indu, S., & Bindhumadhava, B. S. (2020): A Traffic Monitoring and Policy Enforcement Framework for HTTP. In 2020 Third ISEA Conference on Security and Privacy (ISEA-ISAP) (pp. 81-86): IEEE. https://doi.org/10.1109/ISEA-ISAP49340.2020.235004
- Muraleedharan, N., Thomas, A., Indu, S., & Bindhumadhava, B. S. (2020): A Traffic Monitoring and Policy Enforcement Framework for HTTP. In 2020 Third ISEA Conference on Security and Privacy (ISEA-ISAP) (pp. 81-86). IEEE.
- Patel, A., Schenk, T., Knorn, S., Patzlaff, H., Obradovic, D., & Halblaub, A. B. (2021): Real-time, simulation-based identification of cyber-security attacks of industrial plants. In 2021 IEEE International Conference on Cyber Security and Resilience (CSR) (pp. 267-272). IEEE. http://dx.doi.org/10.1109/CSR51186.2021.9527938
- Patel, M., Prabhu, S. R., & Agrawal, A. K. (2021): Network Traffic Analysis for Real-Time Detection of Cyber Attacks. In 2021 8th International Conference on Computing for Sustainable Global Development (INDIACom) (pp. 642-646). IEEE. https://doi.org/10.1109/INDIACom51348.2021.00113.
- Silvestre, A. B., & De Ocampo, J. R. D. (2023): Packet Sniffing in the Cyber Threat Landscape: Examining Wireshark Capabilities, Misuse, and Policy Options in the Philippines. International Journal of Research and Innovation in Social Science, 7(8), 778-786. https://dx.doi.org/10.47772/JJRISS.2023.7856
- Tran, M. C., Nguyen, M. H., & Nguyen, T. Q. (2018): An application for monitoring and analysis of HTTP communications. Journal of Communications, 13(8), 456-462. <u>http://dx.doi.org/10.12720/jcm.13.8.456-462</u>