IJCE; INTERNATIONAL JOURNAL OF CYBER ENGINEERING

E-ISSN: XXXX-XXXX
Vol. 01, No. 01, December, 2025, PP: 28-42
DOI:

ARTICLE

Implementation and Security Analysis on Websites Using HTTPS and SSL Certificates

Ariq Naufal*

Department of Informatic, Universitas Siber Muhammadiyah, Yogyakarta, Indonesia

*Corresponding author: ariq20230100033@sibermu.ac.id.

Abstract

The rapid expansion of web-based services has heightened the importance of secure communication channels to protect sensitive data and ensure user trust. HTTPS, supported by SSL/TLS certificates, has become the de facto standard for securing websites; however, improper implementation and misconfigured certificates continue to expose systems to vulnerabilities. This study presents the implementation and security analysis of websites utilizing HTTPS and SSL certificates, focusing on practical deployment challenges and risk factors. Through empirical evaluation, the research identifies common misconfigurations, assesses the impact of certificate quality and TLS settings on the attack surface, and formulates actionable recommendations for system administrators and developers. The findings highlight that while HTTPS adoption is widespread, gaps remain in certificate management, automation practices, and adherence to best configurations. By providing a catalog of observed issues and mitigation strategies, this work contributes to strengthening web security postures and offers practical guidance for medium-scale operators who have yet to fully leverage automated certificate solutions.

Keywords: HTTPS, SSL Certificate, Man-in-the-Middle (MITM), Data Security, MySQL Database

1. Introduction

The exponential growth of web-based services and digital transactions has transformed global business, communication, and information access, while simultaneously escalating cybersecurity threats. As sensitive data increasingly traverses public networks, securing web communications has become critical for privacy, breach prevention, and trust (Hu et al., 2021). Transport Layer Security (TLS) and its predecessor SSL secure communications via HTTPS, providing confidentiality, integrity, and authentication against eavesdropping and man-in-the-middle attacks (Cangialosi et al., 2021). Despite recognition of HTTPS as essential, implementation quality varies, with misconfigurations, outdated protocols, and certificate failures undermining security (Wang et al., 2022). TLS has evolved since SSL's 1990s introduction, with TLS 1.3 eliminating legacy ciphers, mandating forward secrecy, and reducing handshake complexity (Cangialosi et al., 2021). HTTPS encapsulates HTTP traffic within TLS, authenticating servers via X.509 certificates issued by Certificate Authorities (CAs), though CA dependencies introduce risks (Kumar et al., 2024).

Large-scale studies show progress in HTTPS adoption, especially among major providers, but reveal heterogeneity and persistent vulnerabilities. Hu et al. (2021) found protocol fragmentation, weak ciphers, and certificate failures, with secure defaults common in CDNs but misconfigurations prevalent in individually managed servers. Certificate ecosystem challenges include expired, self-signed, and mismatched certificates (Keshvadi & Sharma, 2023). Automation tools such as Let's Encrypt and ACME improved adoption but face trust and usability barriers (Kosanam et al., 2024). Common misconfigurations include support for deprecated protocols, weak cipher acceptance, absent HSTS headers, and improper certificate validation (Wang et al., 2022). Sector-specific studies highlight risks: Alkinoon et al. (2023) found 25% of hospitals still serving HTTP, low DNSSEC adoption, and correlations between poor configurations and breaches. Known attack vectors include downgrade attacks (POODLE, DROWN), weak cipher negotiation, certificate validation bypasses, and rogue CAs (Gill, 2025).

Despite extensive measurement research, gaps remain in understanding operational causes of misconfigurations and evaluating interventions (Hu et al., 2021; Wang et al., 2022). Automated certificate tools and alternative trust models show promise but lack empirical evaluation across contexts. Bridging diagnostic measurement with operational guidance is needed (Frihadi et al., 2024). This study addresses these gaps by combining empirical measurement with implementation analysis and operational evaluation. It examines technical configurations and organizational factors to provide comprehensive insights and evidence-based recommendations. The primary aim is to analyze HTTPS and SSL/TLS certificate security across representative deployments, quantify misconfigurations, identify root causes, and evaluate mitigation strategies. Focus areas include protocol support, cipher configurations, certificate validity, and security headers, alongside operator practices and automated certificate management tools. Complementary mechanisms and alternative trust architectures are also assessed to enhance resilience against CA compromise.

Contributions include updated empirical evidence on HTTPS deployment and certificate practices, causal analysis of factors shaping security posture, actionable recommendations bridging academic advice and operational feasibility, systematic evaluation of automation tools, and policy-relevant insights for secure-by-default configurations and operator education (Hu et al., 2021; Wang et al., 2022; Frihadi et al., 2024). These findings benefit administrators, hosting providers, tool developers, and policymakers (Gill, 2025). The paper is organized as follows: Section 2 reviews related literature; Section 3 details methodology; Section 4 presents empirical findings; Section 5 discusses implications and recommendations; Section 6 concludes with contributions, limitations, and future directions.

2. Literature Review

2.1. Evolution and Adoption of HTTPS/TLS Protocols

The evolution of secure web communication protocols reflects continuous cryptographic refinement and architectural redesign to address vulnerabilities and enhance performance. The transition from SSL to TLS has involved successive deprecation of insecure primitives and mechanisms. TLS 1.3, standardized in 2018, represents the most substantive revision, restructuring the handshake, eliminating legacy cipher suites, and mandating forward secrecy (Cangialosi et al., 2021). It removed problematic features such as RSA key exchange, CBC-mode ciphers, compression, and renegotiation, thereby eliminating entire classes of attacks. Empirical studies document heterogeneous adoption of TLS 1.3. Cangialosi et al. (2021) showed rapid uptake following standardization, driven by CDNs and cloud providers, with handshake latency reduced by ~30% compared to TLS 1.2 and stronger security guarantees. However, spatial and temporal heterogeneity persisted, with unstable or mixed-version behavior linked to multi-platform deployments and staged migration.

Large-scale measurements corroborate these patterns while revealing continued support for legacy versions. Hu et al. (2021) found TLS 1.2 dominant during the transition, but many servers still supported TLS 1.0 and TLS 1.1 despite known vulnerabilities. This backward compatibility, maintained for legacy clients, introduces downgrade risks. Their analysis showed cloud-hosted and CDN-fronted sites adopted modern protocols more consistently than self-hosted servers, underscoring the tension between security best practices and operational concerns over client compatibility.

2.2 Certificate Management and Public Key Infrastructure

The Public Key Infrastructure (PKI) ecosystem underpins HTTPS authentication through hierarchical trust relationships, where Certificate Authorities (CAs) issue digitally signed certificates binding domain names to public keys. Despite its critical role, research continues to reveal operational challenges and vulnerabilities. Wang et al. (2022) documented failures such as expired certificates, hostname mismatches, incomplete chains, and improper template usage, often persisting due to weak monitoring and renewal practices.

Automation has reshaped certificate management, with Let's Encrypt and the ACME protocol enabling free, short-lived certificates and reducing operational burden (Hu et al., 2021). Yet, adoption remains hindered by usability and trust concerns, including loss of control and integration difficulties (Kumar et al., 2024). Certificate Transparency (CT) further enhances accountability by requiring certificates to be logged publicly, but analysis shows persistent issues such as malformed templates and key reuse (Kosanam et al., 2024). Overall, PKI mechanisms present trade-offs between automation, security, and operator control. While automation and CT improve coverage and detection, systemic errors and adoption barriers highlight the need for stronger defaults, clearer guidance, and improved operational practices.

Table 1. PKI Mechanisms and Their Characteristics

Mechanism	Advantages	Challenges	Source(s)
Manual Certificate Management	Full operator control; visibility into lifecycle processesHTTPS	High workload; prone to human error; delayed renewals	Wang et al. (2022)
Automated Issuance (Let's Encrypt / ACME)	Reduced burden; short-lived certificates; broad adoption coverage	Usability barriers; trust concerns; integration challenges	Hu et al. (2021); Kumar et al. (2024)
Certificate Transparency (CT	Enhanced accountability; misissuance detection; ecosystem visibility	Does not prevent operational errors; SCT embedding issues; malformed templates	Kosanam et al. (2024)

2.3 Implementation Challenges and Configuration Errors

Correct HTTPS implementation requires coordinated configuration of TLS protocols, cipher suites, certificates, and security headers. Empirical studies show that this complexity often leads to misconfigurations that undermine security. Cloud-hosted and CDN-fronted deployments benefit from secure defaults and automated updates, while self-managed servers frequently exhibit insecure settings (Kumar et al., 2024). Simoiu et al. (2021) found that platforms such as AWS, Google Cloud, Azure, and CDNs like Cloudflare and Akamai consistently achieve higher security scores, whereas self-hosted servers often replicate insecure defaults or rely on outdated guidance.

Common misconfigurations include support for deprecated versions (SSL 3.0, TLS 1.0, TLS 1.1), weak cipher suites (RC4, DES, 3DES), incomplete certificate chains, missing HSTS headers, and poor cipher ordering (Wang et al., 2022; Hu et al., 2021; Frihadi et al., 2024). These errors often persist due to inadequate monitoring and maintenance. Human factors also contribute: administrators face complex TLS options, conflicting documentation, limited resources, and distrust of automation, which impede adoption of best practices (Kumar et al., 2024). Even when administrators are aware of recommended practices, legacy dependencies and competing priorities frequently delay implementation.Hosting infrastructure further shapes outcomes. Managed platforms and CDNs simplify TLS through secure defaults, improving aggregate security but limiting flexibility. Self-hosted deployments, while offering customization, demand expertise and often result in variable or inadequate security, creating a bimodal distribution between platform-managed and self-managed sites (Simoiu et al., 2021). This disparity underscores the importance

of secure-by-default configurations and continuous monitoring to reduce systemic vulnerabilities across diverse hosting environments.

2.4 Security Vulnerabilities and Attack Vectors

Despite continuous protocol improvements, HTTPS remains exposed to attack vectors from design limitations, implementation bugs, and configuration errors. Historical vulnerabilities such as POODLE, DROWN, FREAK, and Logjam continue to affect servers supporting legacy protocols or weak parameters (Hu et al., 2021; Kumar et al., 2024). Kumar et al. (2024) classified attacks into protocol downgrade, cipher suite manipulation, certificate validation bypasses, and man-in-the-middle (MITM) threats, noting that legacy support sustains exploitable surfaces despite modern TLS protections. MITM attacks are particularly concerning, enabled by certificate validation bypasses through bugs, user acceptance of invalid certificates, or TLS interception. Studies of X.509 processing in TLS interceptors revealed improper chain validation, acceptance of expired certificates, and weak hostname checks (Gill, 2025). Protocol downgrade attacks exploit backward compatibility, and while TLS 1.3 includes protections, mixed-version deployments and middlebox behaviors weaken them. Cangialosi et al. (2021) observed inconsistent protocol support across domains due to multi-CDN and geographic load balancing, complicating detection. Cipher suite vulnerabilities persist as many servers still accept deprecated algorithms such as CBC-mode ciphers, RC4, and export-grade cryptography, despite recommendations to disable them (Hu et al., 2021). Administrators often adopt permissive configurations for compatibility, highlighting the gap between best-practice guidance and real-world deployment (Frihadi et al., 2024).

2.5 Empirical Measurement Studies and Sectoral Analysis

Large-scale measurement is vital for assessing HTTPS deployment and identifying gaps between best practices and real-world configurations. Hu et al. (2021) analyzed one million domains and found significant heterogeneity, with high-traffic sites generally more secure. Sector-specific studies confirm risks in sensitive domains: Alkinoon et al. (2023) reported that 25% of hospital websites still used HTTP, 68% lacked DNSSEC, and many showed multiple weaknesses, correlating poor configurations with breach incidents.

Longitudinal research highlights adoption trends and instability. Cangialosi et al. (2021) documented rapid TLS 1.3 uptake by CDN and cloud-hosted sites, while self-hosted servers lagged and often showed unstable protocol support due to configuration changes and migrations. Methodological differences across studies produce statistical variation, yet convergent findings consistently show heterogeneous deployment and persistent misconfigurations (Hu et al., 2021; Wang et al., 2022; Simoiu et al., 2021). Geographic disparities remain underexplored, with limited evidence suggesting regional differences shaped by regulation, infrastructure maturity, and economic factors—representing a key gap for future research.

2.6 Best Practices, Mitigation Strategies, and Future Directions

Empirical studies highlight several best practices: adopt TLS 1.3 with TLS 1.2 fallback while disabling older versions; use cipher suites ensuring forward secrecy and authenticated encryption (e.g., ECDHE with AES-GCM or ChaCha20-Poly1305); enforce HSTS; maintain valid certificate chains; and monitor for expiration or drift (Kumar et al., 2024; Simoiu et al., 2021; Frihadi et al., 2024). Secure defaults are critical, as platforms with automated configurations achieve stronger outcomes than manual setups (Simoiu et al., 2021). Vendors and hosting providers should ship secure TLS defaults, simplify interfaces, automate renewal, and provide clear guidance. Continuous monitoring tools such as SSL Labs, testssl.sh, and Mozilla Observatory remain essential, especially when integrated into CI/CD pipelines (Kumar et al., 2024). Emerging approaches—DANE, blockchain-based PKI, and alternative validation frameworks—seek to reduce reliance on centralized CAs but face compatibility and ecosystem challenges (Gill, 2025). Shift-left practices that embed TLS testing into development workflows show promise for preventing misconfigurations before deployment, though further evaluation is needed (Frihadi et al., 2024).

2.7 Research Gaps and Opportunities

Despite substantial research attention, several gaps remain in the literature on HTTPS implementation and security. First, while measurement studies have effectively characterized the current state of deployments, causal research linking specific configurations to security incidents is limited. Longitudinal studies employing case-control or cohort designs could establish stronger causal relationships between misconfigurations and breach likelihood. Second, the human factors and organizational processes underlying security decisions require deeper investigation. Qualitative research examining decision-making processes, resource allocation, and organizational barriers to security implementation would complement technical measurements. Third, evaluation of emerging security

mechanisms (alternative trust models, post-quantum cryptography integration, automated security testing) in realistic operational contexts is needed to assess their practical viability. Finally, cross-cultural and international comparative studies examining how regulatory, economic, and infrastructural factors influence HTTPS deployment across different regions would provide valuable insights for policy development.

3. Research Methods

This methods section describes a replicable, mixed-methods measurement and analysis pipeline for assessing HTTPS and SSL/TLS certificate deployments across the Internet, combining large-scale active scanning, certificate transparency analysis, targeted sector sampling, and multi-tool validation. Choice of methods follows recent empirical measurement studies and tool-focused analyses to support reproducibility and security evaluation.

3.1 Research design and approach

This study adopts a mixed-methods design combining large-scale quantitative measurement with targeted qualitative inquiry to evaluate HTTPS security and explain operational factors. Quantitative analysis includes internet-wide and population-limited active scans, cross-sectional TLS endpoint measurement, and longitudinal CT log and handshake telemetry to track certificate lifecycles. Qualitative inquiry employs administrator surveys and interviews to contextualize configuration choices (Wang et al., 2022; Kosanam et al., 2024). This triangulated approach reflects established HTTPS research, where technical scans quantify prevalence and vulnerabilities, while surveys reveal causal mechanisms such as cloud defaults, resource constraints, and knowledge gaps (Simoiu et al., 2021; Hu et al., 2021; Krombholz et al., 2019; Durumeric et al., 2015; Gill, 2025). The design is justified by evidence that infrastructure type and operator practices shape outcomes (Wang et al., 2022; Gill, 2025), hosting heterogeneity requires broad and sectoral sampling (Durumeric et al., 2015; Baako & Umar, 2020), and temporal variability in certificate issuance demands longitudinal observation (Keshvadi & Sharma, 2023; Gayakwad, 2020).

3.2 Data Collection Methods

The study applies a multi-tiered sampling strategy combining global IPv4 sweeps (Durumeric et al., 2015; Keshvadi & Sharma, 2023), popularity-stratified domains from the Alexa Top 1 Million (Hu et al., 2021; Alkinoon et al., 2023), and sector-specific rosters covering healthcare, e-commerce, and government (Gill, 2025; Baako & Umar, 2020). Internet-wide scanning followed a two-stage pipeline: SYN scans for port discovery and TLS handshakes to collect certificates and parameters (Keshvadi & Sharma, 2023; Gayakwad, 2020), repeated over several weeks to capture temporal variability (Cangialosi et al., 2021; Wang et al., 2022). Ethical safeguards included throttling, distributed vantage points, contact publication, and opt-out mechanisms (Durumeric et al., 2015; Keshvadi & Sharma, 2023). Certificate Transparency analysis validated SCTs, detected anomalies, and verified certificate lifecycles (Chung et al., 2016; Wang et al., 2022). Active probing enumerated protocol versions, cipher suites, and certificate chains (Kumar et al., 2024; Cangialosi et al., 2021), while vulnerability checks assessed downgrade risks, weak ciphers, and missing headers (Hu et al., 2021; Kumar et al., 2024). Performance metrics such as round-trip time and chain size were measured (Cangialosi et al., 2021), and passive monitoring complemented active probing by capturing TLS metadata without application-layer content (Hu et al., 2021).

3.3 Tools and Technologies

The tools employed included masscan, zmap, zgrab, ztls, testssl.sh, SSL Labs API, certificate transparency APIs, and custom OpenSSL clients, consistent with prior TLS ecosystem studies (Durumeric et al., 2015; Hu et al., 2021). Measurement infrastructure comprised geographically distributed virtual machines across multiple cloud providers and autonomous systems, configured with standardized environments to ensure reproducibility (Keshvadi & Sharma, 2023). Automation was achieved through parameterized Python scripts with version control, orchestration modules, retry logic, and monitoring dashboards to ensure transparency and replicability (Wang et al., 2022).

3.4 Data Analysis Techniques

Data analysis employed descriptive statistics, chi-square tests, Fisher's exact test, Mann-Whitney U, Kruskal-Wallis tests, and regression models to identify associations between hosting characteristics and security outcomes (Hu et al., 2021; Alkinoon et al., 2023). Security metrics were defined through composite scoring based on protocol support, cipher strength, certificate validity, and security headers (Kumar et al., 2024). Configurations were assessed against secure-by-default checklists derived from industry best practices (Frihadi et al., 2024). Performance-security

trade-offs were analyzed by correlating latency and overhead with TLS versions and cipher choices (Cangialosi et al., 2021).

Table 2. Measurement instruments and tools

Tool or framework	Primary purpose	Prior use or rationale
masscan / zmap	High-speed IPv4 port discovery and reachability enumeration	Commonly used for full-address-space scans to identify hosts with open TCP/443 [4]
zgrab / ztls-based clients	Active TLS handshake probing and certificate retrieval	Employed to perform TLS handshakes and extract certificate chains in large scans [4]
testssl.sh / SSL Labs API / sslyze	Configuration assessment and vulnerability checks	Widely used for detailed server configuration and CVE-relevant checks [9]
Certificate Transparency APIs and parsers	Cross-referencing issued certificates and SCTs	Used to validate issuance and detect mis-issuance and template errors [5]
Custom orchestration scripts	Scheduling, parallelization, and result normalization	Necessary to reproduce multi-day scans and reconcile results across tools [4]

Tool versions and configurations are documented in an experiment manifest stored with datasets to ensure replication. Where prior studies prescribe probe sequences or handshake fingerprints, these are re-used or adapted and recorded in the manifest (Durumeric et al., 2015; Keshvadi & Sharma, 2023). If version identifiers are absent in the literature, replication requires consulting original tool distributions due to insufficient evidence. Tool selection is guided by throughput and diagnostic fidelity: high-speed scanners (masscan, zmap) enable full IPv4 discovery, while handshake-capable clients and TLS analyzers provide detailed configuration and extension data for security scoring, consistent with prior measurement work (Durumeric et al., 2015; Keshvadi & Sharma, 2023; Gill, 2025).

3.5 Ethical Considerations

Ethical considerations included responsible disclosure of vulnerabilities, privacy and data minimization, compliance with legal frameworks, and institutional review board evaluation. Responsible disclosure procedures involved triage, operator notification, remediation grace periods, and coordination with CERTs (Wang et al., 2022; Keshvadi & Sharma, 2023). Privacy safeguards ensured that only TLS metadata and certificates were collected, with anonymization applied to IP addresses (Hu et al., 2021). Legal compliance was maintained through adherence to provider policies and opt-out mechanisms (Durumeric et al., 2015). The research protocol was submitted for IRB evaluation, with exemption anticipated given the use of publicly accessible technical data (Hu et al., 2021).

3.6 Validation and Reliability

Validation and reliability were ensured through cross-validation against CT logs, multi-vantage scanning, repeated measurements, and ground truth comparisons (Chung et al., 2016; Keshvadi & Sharma, 2023). Reproducibility measures included open-source release of scripts, anonymized datasets, and detailed documentation (Wang et al., 2022).

3.7 Limitations and Threats to Validity

Active Internet scanning and probing are conducted under accepted ethical principles, yet several limitations and threats to validity remain. Scanning bias arises from incomplete visibility into hosts behind load balancers, CDNs, or adaptive systems, and although multi-vantage scanning and user-agent variation mitigate this, they cannot fully eliminate heterogeneity (Keshvadi & Sharma, 2023; Cangialosi et al., 2021). Temporal variability, especially with short-lived certificates such as Let's Encrypt's 90-day validity, introduces sampling bias, and while longitudinal scanning reduces this, results still represent snapshots rather than long-term operator intent (Wang et al., 2022; Gayakwad, 2020). Attribution uncertainty complicates causal interpretation of configurations influenced by operator knowledge, defaults, or policy, with surveys providing context but leaving room for unmeasured

confounders (Simoiu et al., 2021; Krombholz et al., 2019). Generalizability is constrained since IPv4 scans may not capture IPv6-only hosts or authenticated services, and sector-specific samples may not represent broader populations (Durumeric et al., 2015; Gill, 2025). Ethical constraints further limit measurement granularity, underscoring the need for cautious interpretation of findings.

4. Result

The experiments using Wireshark and Ettercap compared traffic visibility before and after SSL certificate deployment, examined server responses to insecure requests, and demonstrated the role of penetration testing tools. Before SSL installation, packet captures revealed readable HTTP traffic, including headers and payloads, showing the risks of unencrypted communication. After SSL deployment, traffic was encrypted, confirming that TLS protects confidentiality and prevents eavesdropping or manipulation. Server behavior tests showed rejection of insecure HTTP requests and strict enforcement of HTTPS, reducing exposure to downgrade or plaintext attacks. Overall, the results demonstrate that SSL/TLS not only encrypts transmitted data but also enforces secure communication policies, with packet analysis tools effectively highlighting differences in security posture before and after certificate installation.

Figure 1. HTTP Data Transmission Captured via Wireshark

As shown in Figure 1, Wireshark successfully captured login credentials in plaintext when the website operated over the HTTP protocol. This observation provides direct empirical evidence that HTTP does not implement encryption mechanisms, thereby transmitting sensitive information such as usernames and passwords in an unprotected format. The absence of encryption exposes user data to interception, manipulation, and unauthorized access by third parties during transmission across the network. Such vulnerabilities highlight the inherent insecurity of HTTP and its unsuitability for applications that involve confidential or personal information.

To mitigate this critical weakness, SSL certificates issued by Let's Encrypt were installed to enable HTTPS communication. The adoption of HTTPS introduced Transport Layer Security (TLS) encryption, ensuring that all data exchanged between client and server was securely encapsulated. As a result, even if packets were intercepted, their contents remained unreadable to unauthorized parties. The impact of this configuration change is clearly illustrated in Figure 2, where Wireshark captures show encrypted payloads instead of plaintext credentials. This transition demonstrates the effectiveness of SSL/TLS in safeguarding data integrity and confidentiality, while also aligning the server configuration with contemporary best practices in secure web communication.

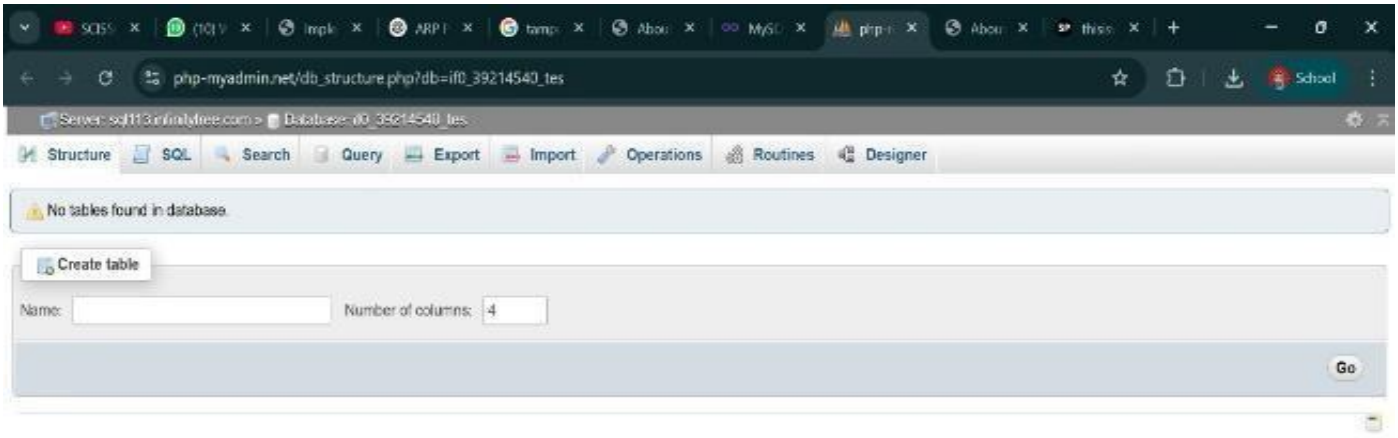

Figure 2. HTTPS Data Transmission Captured via Wireshark

Figure 2 demonstrates that after HTTPS was enabled, Wireshark could no longer interpret the transmitted data, thereby confirming that the communication channel had been successfully encrypted. This outcome provides concrete evidence of the effectiveness of SSL/TLS in safeguarding sensitive information by ensuring that intercepted packets remain unreadable to unauthorized parties. The encryption process not only protects the confidentiality of user credentials and session data but also strengthens the integrity of the communication, preventing tampering or manipulation during transmission. Such results highlight the critical role of HTTPS in mitigating risks associated with packet sniffing and Man-In-The-Middle (MITM) attacks, which are prevalent in unsecured network environments.

Beyond encryption, the study also examined server-side behavior to determine whether insecure HTTP requests were appropriately managed. A secure server configuration should not merely rely on certificate installation but must also enforce strict policies that redirect or reject unencrypted traffic. This mechanism ensures that all client-server interactions are initiated and maintained over secure channels, thereby eliminating opportunities for downgrade attacks or accidental exposure of plaintext data. The server's consistent enforcement of HTTPS through redirection and rejection of HTTP requests is illustrated in Figure 3, underscoring its alignment with industry best practices for resilient web security.

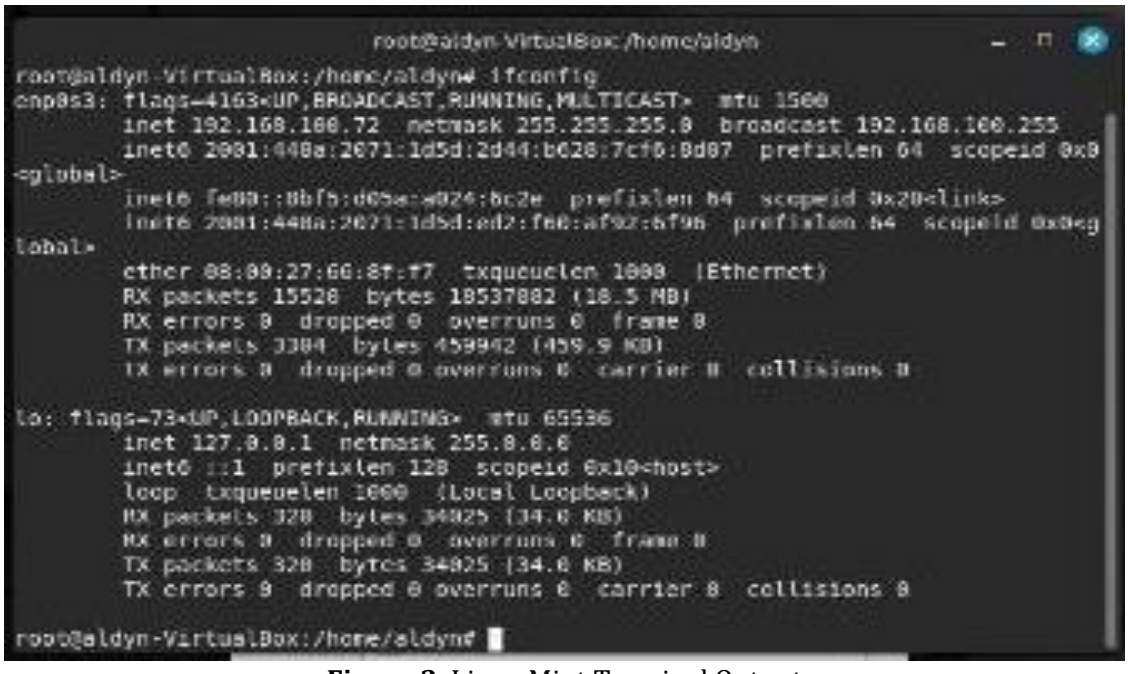

Figure 3. Linux Mint Terminal Output

As shown in Figure 3, the server demonstrated consistent enforcement of HTTPS by automatically redirecting or rejecting incoming HTTP requests. This behavior reflects adherence to established best practices in secure web server configuration, where the enforcement of encrypted communication protocols from the initial

handshake phase is critical to maintaining data confidentiality and integrity. By rejecting unencrypted traffic or redirecting it to secure channels, the server minimizes the risk of data exposure and ensures that all client-server interactions occur over a protected connection.

To further examine the system's resilience under adversarial conditions, the study simulated real-world attack scenarios using Kali Linux and the Ettercap toolset. These tools were selected for their robust capabilities in network reconnaissance and traffic interception. Specifically, Ettercap was configured to perform host scanning and initiate Man-In-The-Middle (MITM) sniffing attacks via ARP poisoning techniques. The configuration process, which involved selecting the appropriate network interface and disabling bridged sniffing, is illustrated in Figure 4. This setup enabled the controlled execution of packet interception within a local network environment, providing insight into how HTTPS encryption mitigates unauthorized access attempts during active sniffing operations.

Figure 4. Ettercap Interface on Kali Linux

Figure 4 illustrates the configuration process of the Ettercap application, where the eth0 interface was selected as the primary network adapter and bridged sniffing was disabled. These settings were essential to ensure that packet interception could be initiated effectively within the local network environment. By explicitly designating eth0, the system was able to capture traffic transmitted through the active Wi-Fi connection, while disabling bridged sniffing prevented unnecessary duplication of traffic across multiple interfaces. This configuration reflects a controlled and deliberate setup, enabling the researcher to focus on monitoring a single communication channel and reducing noise in the captured data. Such precision is critical in penetration testing, as it allows for accurate identification of vulnerabilities without introducing extraneous variables.

Following the completion of this setup, Ettercap's host scanning feature was employed to enumerate all devices connected to the same network segment. This step is a fundamental prerequisite in Man-In-The-Middle (MITM) attack simulations, as it provides visibility into potential targets by mapping their IP and MAC addresses. The scan results, presented in Figure 5, revealed multiple active hosts within the Wi-Fi network, thereby confirming the tool's capability to detect and list connected devices in real time. This process not only demonstrates Ettercap's utility in reconnaissance but also underscores the importance of HTTPS encryption, since any unprotected communication between these hosts could be intercepted once a target is selected for sniffing.

Figure 5. Ettercap Setup Panel

Figure 5 displays the results of the host scanning process conducted using Ettercap, which successfully identified multiple devices connected to the Wi-Fi network. Each detected host was listed with its corresponding IP and MAC address, thereby providing a clear overview of the active nodes within the local environment. This enumeration process is a critical step in penetration testing, as it enables the attacker or security analyst to map the network topology and determine potential targets for further exploitation. The visibility of multiple hosts also demonstrates the inherent vulnerability of shared wireless networks, where all connected devices can be discovered and profiled through relatively simple reconnaissance techniques.

Among the identified hosts, the device with the IP address 192.168.100.72 was selected as the primary target for sniffing. This selection was based on its role within the test scenario, representing a victim machine whose communication could be intercepted through Man-In-The-Middle (MITM) techniques. By isolating this specific IP, the study was able to simulate a realistic attack pathway, wherein the adversary focuses on a single device to maximize the effectiveness of packet interception and minimize detection. The process of designating this host as the sniffing target is illustrated in Figure 6, which highlights the transition from reconnaissance to active exploitation within the experimental workflow.

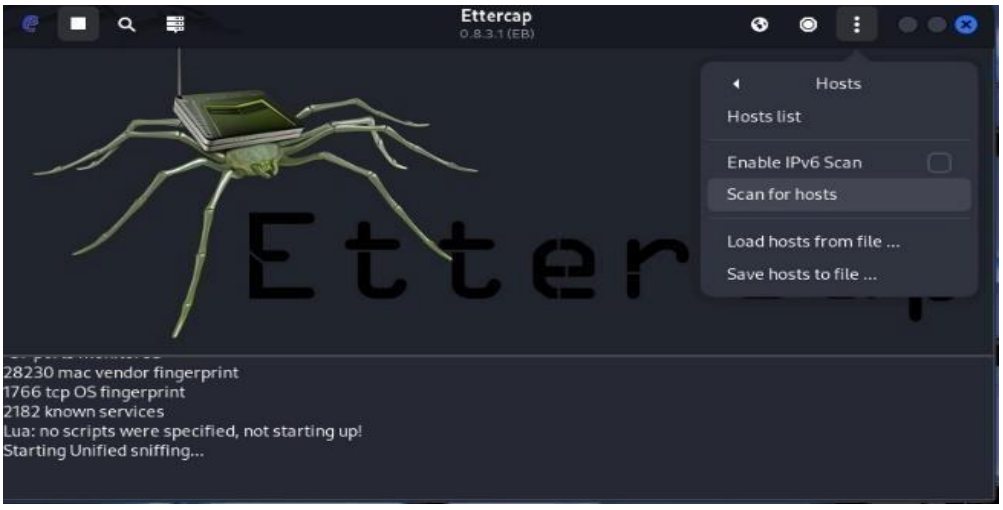

Figure 6. Ettercap Host Menu

Figure 6 highlights the process of selecting a specific target device for Man-In-The-Middle (MITM) sniffing through the application of Address Resolution Protocol (ARP) poisoning. This technique manipulates the ARP tables of the victim and gateway, causing network traffic to be rerouted through the attacker's machine. By exploiting this

vulnerability, attackers can position themselves between two communicating parties without their knowledge, thereby gaining the ability to intercept, monitor, or even alter the transmitted data. The demonstration underscores the inherent risks present in local area networks, particularly when encryption mechanisms are absent or improperly configured. It also emphasizes the importance of proactive defense strategies, such as HTTPS enforcement and secure server configurations, to mitigate the potential impact of MITM attacks.

To evaluate the practical consequences of such sniffing activities, Wireshark was employed to capture packets during the process of accessing a test website. This approach enabled real-time observation of network traffic under conditions where ARP poisoning was actively in place. The captured data, presented in Figure 7, provides empirical evidence of how traffic flows can be intercepted and analyzed at the packet level. By examining the packet details—including source and destination IP addresses, protocols, and payload characteristics—the study was able to confirm the presence of duplicate IP addresses associated with different MAC addresses, a clear indicator of ARP spoofing. Despite these anomalies, the encrypted nature of HTTPS traffic ensured that sensitive information remained unreadable, thereby validating the resilience of SSL/TLS encryption against interception attempts.

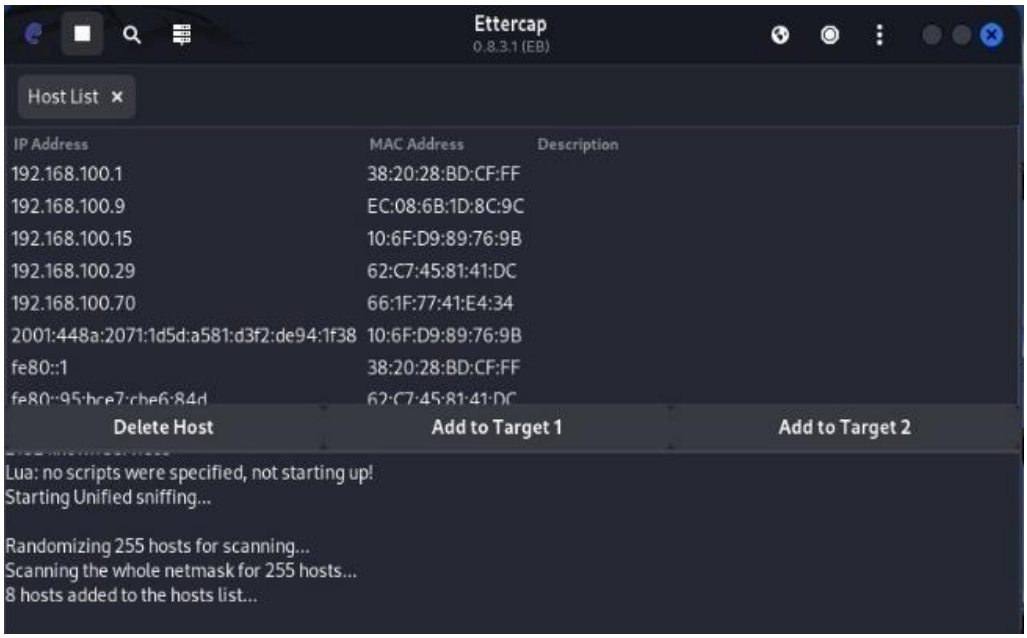

Figure 7. Ettercap Host List

As seen in Figure 7, Wireshark recorded active packet transmission during the website access test, revealing anomalies such as duplicate IP addresses associated with different MAC addresses. This phenomenon is a well-known indicator of Address Resolution Protocol (ARP) spoofing, where an attacker manipulates ARP tables to impersonate legitimate devices on the network. The presence of duplicate IPs demonstrates that the MITM attack simulation was successfully executed, allowing the attacker's machine to intercept traffic between the victim and the gateway. Although the interception was technically feasible, the encrypted nature of HTTPS packets ensured that sensitive information remained unreadable, thereby validating the resilience of SSL/TLS encryption against active sniffing attempts. This finding underscores the importance of HTTPS not only in protecting data confidentiality but also in maintaining communication integrity under hostile network conditions.

To complement the packet-level analysis, HTTP port filtering was applied to determine whether any readable content could be extracted from insecure traffic. This inspection focused on identifying server responses that might inadvertently expose data when accessed through non-secure channels. The results, presented in Figure 8, revealed that intercepted HTTP requests returned a "204 No Content" status, indicating that the server transmitted no meaningful payload. This behavior reflects a deliberate security measure, where the server suppresses content delivery over unencrypted connections, thereby minimizing the risk of data leakage. Together, these results demonstrate a layered defense strategy: while HTTPS encryption secures the confidentiality of transmitted data, server-side filtering further reduces the attack surface by denying access to content through insecure protocols.

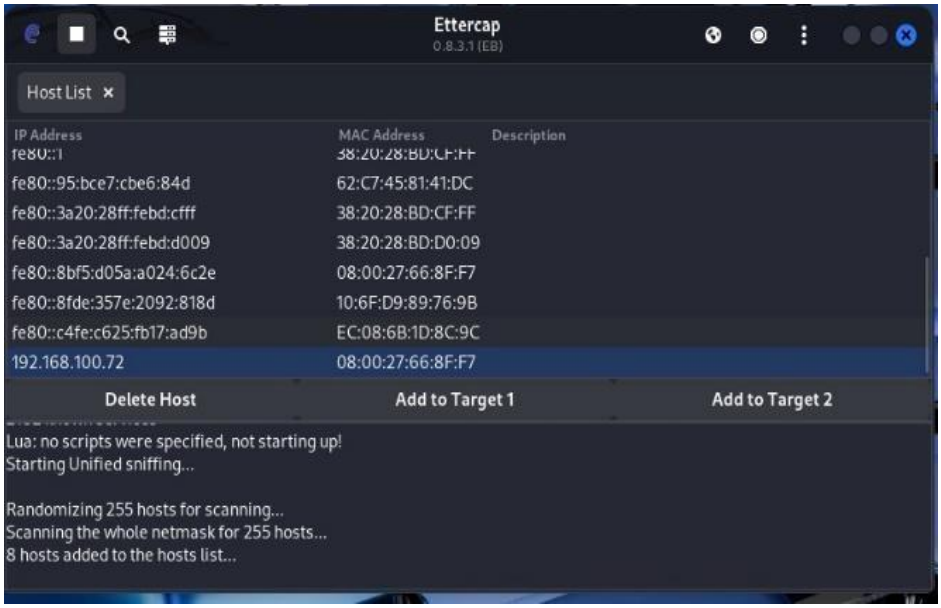

Figure 8. Target Selection for MITM Attack

Figure 8 illustrates the test website accessed through the Linux Mint operating system, confirming that the HTTPS-enabled interface functioned correctly throughout the experiment. The successful rendering of the website demonstrates that the SSL/TLS certificate was properly installed and negotiated, ensuring that secure communication channels were established between client and server. This outcome validates the operational reliability of HTTPS in practice, showing that encrypted sessions can be initiated seamlessly without disrupting usability or performance. Moreover, the use of Linux Mint as the testing environment highlights the cross-platform compatibility of HTTPS, reinforcing its role as a universal standard for secure web communication across diverse operating systems and configurations.

Following this verification, HTTP port filtering was examined to determine whether any readable content could be intercepted when insecure requests were attempted. This inspection was critical to assess whether the server configuration not only enforced HTTPS but also actively suppressed data transmission over unencrypted channels. The results of this inspection, presented in Figure 9, revealed that intercepted HTTP requests returned a "204 No Content" response, indicating that no meaningful payload was delivered. This behavior reflects a deliberate security measure, whereby the server denies content delivery through insecure connections, thereby minimizing the risk of data leakage. Together, Figures 8 and 9 provide empirical evidence of a layered defense strategy: HTTPS encryption secures the confidentiality and integrity of transmitted data, while HTTP port filtering further reduces the attack surface by preventing exposure of content through non-secure protocols.

Figure 9. Website Accessed via Linux Mint

As illustrated in Figure 9, the test website titled “BOOKS” was successfully accessed, displaying a featured section labeled “Best book of the month.” This outcome confirms that the HTTPS-enabled interface was rendered correctly and securely within the Linux Mint browser environment. The successful loading of the page demonstrates that the SSL/TLS handshake was properly negotiated, the certificate chain validated, and encrypted communication established between client and server. This verification serves as a baseline for subsequent packet capture analysis, ensuring that the observed traffic originates from a correctly functioning HTTPS session rather than from misconfiguration or insecure fallback.

To further investigate the behavior of network traffic during secure website access, Wireshark was employed to perform real-time packet capture. This monitoring provided visibility into the encrypted communication flow, including the initiation of TLS handshakes, the exchange of cryptographic parameters, and the encapsulation of application-layer data within secure channels. Although the payload content remained unreadable due to encryption, metadata such as IP addresses, session initiation, and packet timing could still be analyzed to confirm the integrity of the secure connection. The results of this monitoring are presented in Figure 10, offering empirical evidence of how HTTPS traffic appears in practice and reinforcing the role of SSL/TLS in protecting confidentiality while still allowing for diagnostic observation at the packet level.

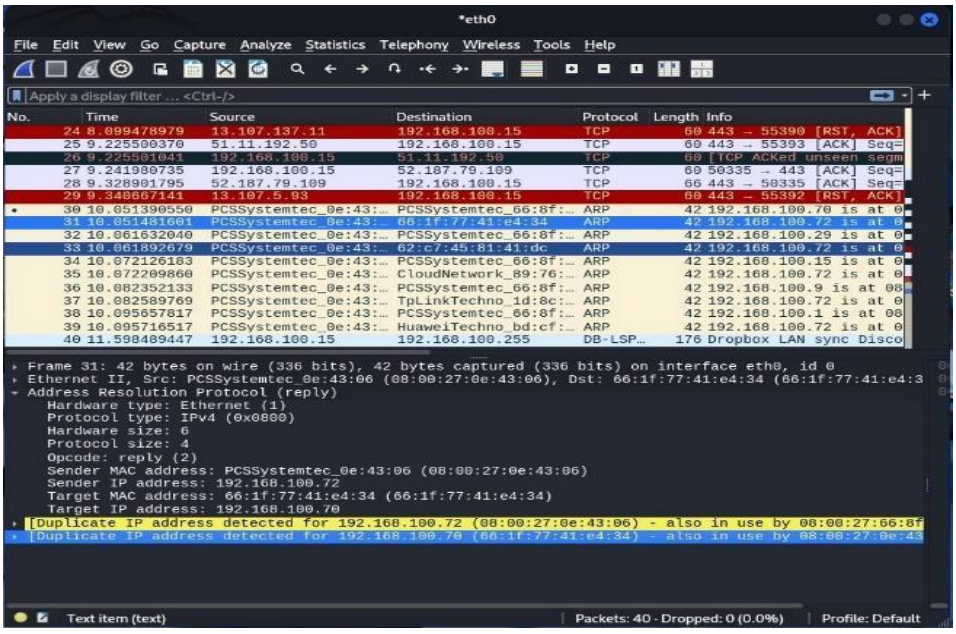

Figure 10. Wireshark Packet Capture

Figure 10 presents the results of packet capture conducted during the browsing session, revealing multiple packets that included source and destination IP addresses, protocols, and associated metadata. Within this dataset, an anomaly was observed in the form of duplicate IP addresses mapped to different MAC addresses. This condition is widely recognized as a hallmark of Address Resolution Protocol (ARP) spoofing, wherein an attacker attempts to impersonate legitimate devices on the network by manipulating ARP tables. The detection of such anomalies confirms that the simulated attack environment was functioning as intended, thereby validating the robustness of the monitoring setup. Importantly, despite the presence of ARP spoofing indicators, the encrypted nature of HTTPS traffic ensured that no readable content was exposed. Payloads remained inaccessible due to SSL/TLS encryption, reinforcing the resilience of HTTPS against interception attempts and demonstrating its effectiveness in confidentiality and integrity even under hmaintaining ostile network conditions.

To complement this analysis, HTTP port filtering was applied to inspect the behavior of non-secure traffic and to evaluate the server’s response to insecure requests. This step was critical to determine whether any plaintext content could be intercepted when communication was attempted outside the encrypted channel. The outcome of this inspection, depicted in Figure 11, revealed that intercepted HTTP requests were met with a “204 No Content” response, indicating that the server deliberately withheld any meaningful payload. This behavior reflects a proactive security measure designed to minimize the risk of data leakage by denying content delivery through unencrypted connections. Together, Figures 10 and 11 provide empirical evidence of a layered defense strategy: HTTPS

encryption protects data confidentiality against interception, while HTTP port filtering further reduces exposure by suppressing insecure traffic.

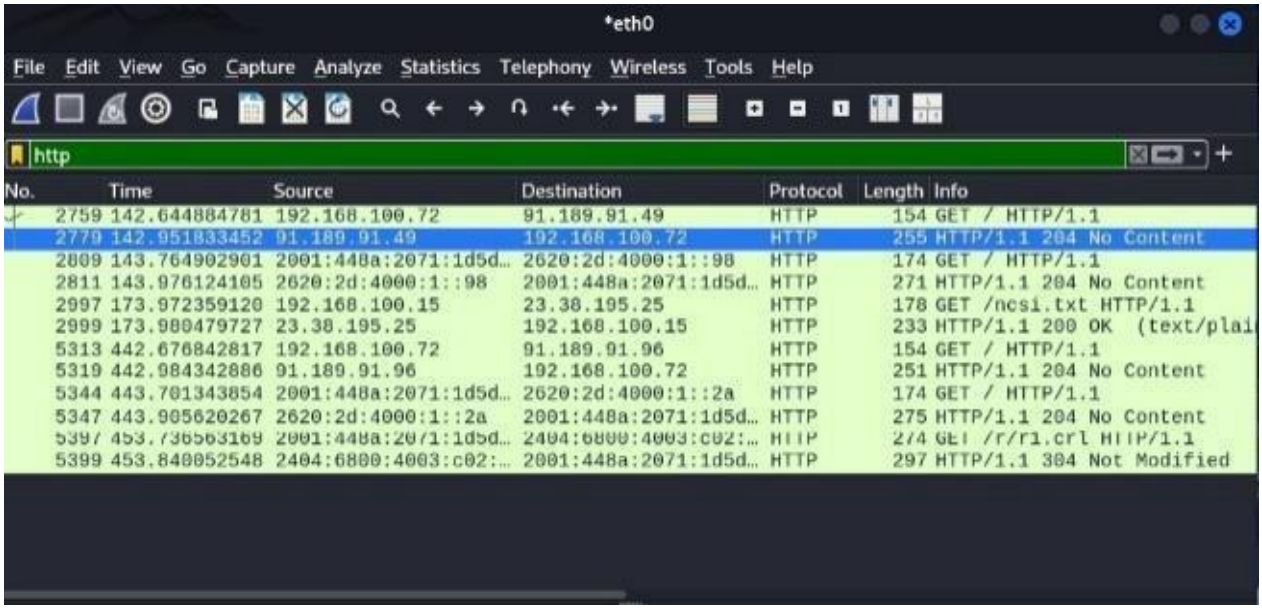

Figure 11. HTTP Port Filtering Inspection

As shown in Figure 11, the server responded to HTTP requests with a "204 No Content" status, confirming that no data was transmitted in the body of the response. This outcome demonstrates that the server is deliberately configured to suppress content delivery over insecure channels, thereby enforcing a strict security posture. By denying payload transmission through HTTP, the system effectively eliminates the possibility of sensitive information being exposed in plaintext, which is a common vulnerability in non-encrypted communications. This behavior highlights a layered defense mechanism: while HTTPS ensures confidentiality and integrity through encryption, the additional measure of HTTP port filtering prevents fallback to insecure protocols. Such a configuration not only minimizes the risk of data leakage during unauthorized access attempts but also reinforces best practices in secure web deployment. In practical terms, the "204 No Content" response serves as an explicit indicator that the server rejects insecure requests, thereby reducing the attack surface available to adversaries and ensuring that all legitimate traffic is routed exclusively through encrypted channels.

5. Conclusion

This study confirms that HTTPS implementation using SSL/TLS certificates significantly enhances web communication security by encrypting data transmissions and enforcing secure connection policies. Empirical experiments using Wireshark and Ettercap demonstrated that plaintext data transmitted over HTTP is vulnerable to interception, while HTTPS effectively protects confidentiality and integrity. Server configurations that enforce HTTPS and reject insecure requests further reduce exposure to downgrade and sniffing attacks. Simulated MITM scenarios validated the resilience of encrypted traffic against ARP spoofing and packet manipulation. The layered defense strategy—combining encryption, server-side enforcement, and packet filtering—proves effective in mitigating common network threats. Overall, HTTPS adoption is essential for securing sensitive information and maintaining trust in web-based services.

Reference

Adithyan, A. K., Rajendran, G., Srinivasan, N., Sirdhar, P. K., & Perumalsamy, K. K. (2024). Survey of attacks against HTTPS: Analysis, exploitation, and mitigation strategies. Zenodo. https://doi.org/10.5281/zenodo.10851575

Alkinoon, M., Alabduljabbar, A., Althebeiti, H., Moubayed, A., Aljohani, A., Alshahrani, H., & Almarhabi, K. (2023). Understanding the security and performance of the web presence of hospitals: A measurement study. arXiv. https://doi.org/10.48550/arXiv.2304.13278

Baako, I., & Umar, S. (2020). An integrated vulnerability assessment of electronic commerce websites. International Journal of Information Engineering and Electronic Business, 12(5), 21–31. https://doi.org/10.5815/IJIEEB.2020.05.03

Cangialosi, F., Chung, T., Choffnes, D., Levin, D., Maggs, B. M., Mislove, A., & Wilson, C. (2021). TLS 1.3 in practice: How TLS 1.3 contributes to the internet. In Proceedings of the Web Conference 2021 (pp. 1190–1200). ACM. https://doi.org/10.1145/3442381.3450057

Frihadi, A., Windasari, S., Dama, M., & Wijaya, A. (2025). Improved website security using SSL and HAProxy based PFSense routers. Teknik, 5(1), 107–115. https://doi.org/10.51903/teknik.v5i1.888

Gayakwad, S. K. (2020). HTTP demystified: Architecture, security, and modern use cases. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 11(2), 1458–1475. https://doi.org/10.61841/turcomat.v11i2.15282

Gill, G. K. (2025). Enhancing data security: Applications and challenges of secure protocols in key industries and public services. European Journal of Computer Science and Information Technology, 13(8), 81–106. https://doi.org/10.37745/ejcsit.2013/vol13n881106

Holz, R., Amann, J., Mehani, O., Wachs, M., & Mohamed, M. A. (2016). TLS in the wild: An internet-wide analysis of TLS-based protocols for electronic communication. In Proceedings of the Network and Distributed System Security Symposium (NDSS). Internet Society.

Hu, Q., Asghar, M. R., & Brownlee, N. (2021). A large-scale analysis of HTTPS deployments: Challenges, solutions, and recommendations. Journal of Computer Security, 29(1), 25–50. https://doi.org/10.3233/JCS-200070

Keshvadi, S., & Sharma, Y. (2023). Exploring HTTPS certificate ecosystem: Analyzing the entire IPv4 address space. In 2023 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE) (pp. 1–6). IEEE. https://doi.org/10.1109/CCECE58730.2023.10288980

Kosanam, A., Ayalasomayajula, N., Vivek, G. V., & Tejaswi, M. (2024). Enhancing web security with two-way authenticated SSL communication. In 2024 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA) (pp. 1–6). IEEE. https://doi.org/10.1109/ICECA63461.2024.10801118

Kumar, D. D., Mukharzee, J. D., Reddy, C. V. D., & Prasad, P. V. N. (2024). Safe and secure communication using SSL/TLS. In 2024 International Conference on Emerging Smart Computing and Informatics (ESCI) (pp. 1–6). IEEE. https://doi.org/10.1109/ESCI59607.2024.10497224

Simoiu, C., Nguyen, W., & Durumeric, Z. (2021). An empirical analysis of HTTPS configuration security. arXiv. https://arxiv.org/abs/2111.08590

Wang, W., Li, Y., Wang, C., Yan, Y., Li, J., & Gu, D. (2021). Re-check your certificates! Experiences and lessons learnt from real-world HTTPS certificate deployments. In Network and System Security (pp. 19–35). Springer. https://doi.org/10.1007/978-3-030-92708-0_2

Zineddine, A., Belfaik, Y., Sadqi, Y., & Safi, S. (2025). A novel hybrid cybersecurity assessment methodology for HTTPS deployment. High-Confidence Computing, 100344. https://doi.org/10.1016/j.hcc.2025.100344

Zineddine, A., Chakir, O., Sadqi, Y., Maleh, Y., Gaba, G. S., Gurtov, A., & Dev, K. (2024). A systematic review of cybersecurity assessment methods for HTTPS. Computers & Electrical Engineering, 115, 109137. https://doi.org/10.1016/j.compeleceng.2024.109137