International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-13 Issue-4, March 2024

Survey of Attacks against HTTPS: Analysis, Exploitation, and Mitigation Strategies

Adithyan Arun Kumar, Gowthamaraj Rajendran, Nitin Srinivasan, Praveen Kumar Sridhar, Kishore Kumar Perumalsamy

Abstract: ***This research paper provides a comprehensive overview of known attacks against HTTPS, with a focus on the SSL and TLS protocols. The paper begins by explaining the working of HTTPS, followed by detailed descriptions of SSL and TLS protocols. Subsequently, it explores common attacks against HTTPS, providing an in-depth analysis of each attack, along with proof-of-concept (PoC) demonstrations. Furthermore, the paper outlines mitigation strategies to address each attack, emphasizing the importance of proactive security measures. Finally, a conclusion is drawn, highlighting the evolving nature of HTTPS attacks and the continuous need for robust security practices.***

Keywords: HTTPS, TLS, SSL, Heartbleed, BEAST

I. INTRODUCTION

In today's digital age, secure web communication is of paramount importance for protecting sensitive data, ensuring privacy, and establishing trust between users and websites. Hypertext Transfer Protocol Secure (HTTPS) has emerged as the standard protocol for secure web communication. It combines the Hypertext Transfer Protocol (HTTP) with the Secure Sockets Layer (SSL) or Transport Layer Security (TLS) protocols to provide encryption, integrity, and authentication mechanisms. HTTPS serves as a crucial safeguard against various security threats that can compromise the confidentiality and integrity of data transmitted over the web. By encrypting the communication channel between the client and the server, HTTPS mitigates the risk of eavesdropping, data tampering, and unauthorized access. It ensures that sensitive information, such as login credentials, financial transactions, and personal data, remains confidential and protected from malicious actors.

Manuscript received on 28 February 2024 | Revised Manuscript received on 08 March 2024 | Manuscript Accepted on 15 March 2024 | Manuscript published on 30 March 2024.

*Correspondence Author(s)
Adithyan Arun Kumar*, Department of Information Security, Carnegie Mellon University, San Jose, United States. E-mail: adithyanhaxor@gmail.com, ORCID ID: 0000-0001-9790-2657
Gowthamaraj Rajendran, Department of Information Security, Carnegie Mellon University, San Jose, United States. E-mail: gowthamaraj360@gmail.com, ORCID ID: 0000-0001-9422-3907
Nitin Srinivasan, Department of Computer Science, University of Massachusetts Amherst, Sunnyvale, United States. E-mail: nitinsr1217@gmail.com, ORCID ID: 0009-0001-6472-4441
Praveen Kumar Sridhar, Department of Data Science, Northeastern University, San Jose, United States. E-mail: prasri.pk@gmail.com, ORCID ID: 0009-0001-6486-1614
Kishore Kumar Perumalsamy, Department of Computer Science, Carnegie Mellon University, San Jose, United States. E-mail: kishore7173@gmail.com, ORCID ID: 0009-0009-5661-7644

HTTPS also plays a pivotal role in building trust and confidence among users. The presence of a valid SSL/TLS certificate, indicated by a padlock symbol or a green address bar, assures users that they are interacting with a legitimate and trusted website. This not only protects users from phishing attacks and spoofed websites but also enhances the reputation and credibility of online businesses.

II. OVERVIEW OF HTTPS PROTOCOL

HTTPS is an extension of the standard HTTP protocol that adds an extra layer of security through the integration of SSL or TLS. The SSL/TLS protocols provide encryption, authentication, and integrity mechanisms to secure the communication channel. The process of establishing an HTTPS connection involves several steps:

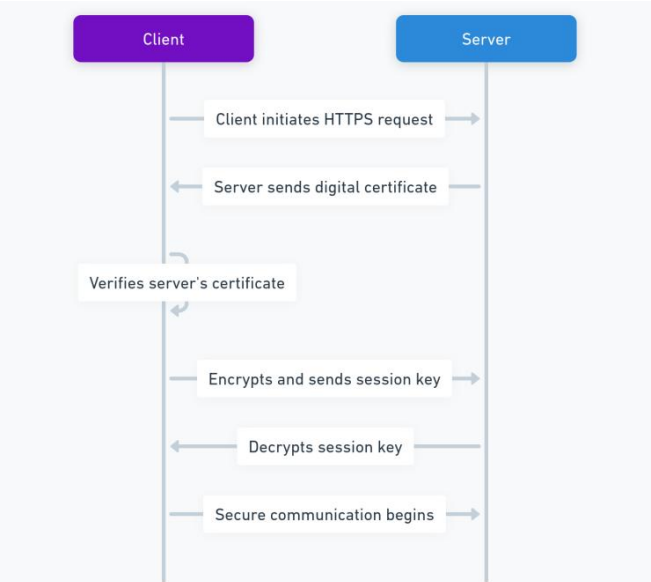


Fig. 1: Establishing an HTTPS Connection

1) Client Request: The client initiates a connection to the server by sending an HTTPS request. The request includes the URL of the website and specifies the use of HTTPS.
2) Server Certificate: The server responds by sending its digital certificate, which contains the server's public key, identifying information, and the digital signature from a trusted Certificate Authority (CA).
3) Certificate Verification: The client verifies the authenticity and validity of the server's certificate. It checks the certificate's digital signature, expiration date, and the CA's trustworthiness.

4) Key Exchange: Upon successful verification, the client generates a random session key and encrypts it using the server's public key. The encrypted session key is sent to the server.
5) Session Establishment: The server decrypts the session key using its private key, which it alone possesses. Both the client and server now share the same session key, which will be used to encrypt and decrypt data during the session.
6) Secure Communication: With the session key established, the client and server can now securely exchange data over an encrypted channel. This ensures the confidentiality and integrity of the transmitted information.

HTTPS employs various encryption algorithms, such as the Advanced Encryption Standard (AES), RSA, and Elliptic Curve Cryptography (ECC), to protect the data exchanged between the client and the server. The choice of encryption algorithm depends on the SSL/TLS version, the cypher suite negotiation, and the server configuration. By implementing HTTPS, websites can create a secure environment for users to browse, interact, and conduct transactions, thereby mitigating the risks of data breaches, identity theft, and unauthorised access. However, it is essential to be aware of the potential attacks against HTTPS to ensure robust security measures are in place.

III. SSL AND TLS

A. Key Principles of Secure Web Communication:

- Confidentiality: Secure web communication ensures that the information transmitted between the client and the server remains confidential and private. Encryption techniques, such as SSL and TLS, are used to encrypt the data, making it unreadable to unauthorized parties.
- Integrity: Secure web communication ensures the integrity of data. It guarantees that the data remains unaltered during transmission. This is achieved through the use of hash functions and digital signatures, which enable verification of data integrity at the receiving end.
- Authentication: Secure web communication involves verifying the identities of both the client and the server. It ensures that the client is communicating with the intended server and that the server is a trusted entity. Authentication prevents impersonation and man-in-the-middle attacks.
- Non-repudiation: Non-repudiation ensures that the sender of a message cannot deny sending it. It provides proof of the authenticity and integrity of the communication, which can be crucial for legal or audit purposes.

B. Role of SSL and TLS in Establishing a Secure Connection:

SSL (Secure Sockets Layer) and its successor, TLS (Transport Layer Security), are cryptographic protocols that play a critical role in establishing secure connections over the internet. They provide the necessary encryption, authentication, and integrity mechanisms to ensure safe web communication.
1) Encryption: SSL and TLS protocols utilise encryption algorithms to safeguard the confidentiality of data exchanged between the client and the server. They employ symmetric key encryption to encrypt the actual data and asymmetric (public-key) encryption for exchanging the symmetric session key securely.

2) Key Exchange: SSL and TLS protocols facilitate the secure exchange of encryption keys between the client and the server. This is achieved through a process called the handshake, where the client and server negotiate and agree upon the encryption algorithm, generate session keys, and establish a secure communication channel.

3) Authentication: SSL and TLS protocols verify the authenticity of the server's identity using digital certificates. These certificates are issued by trusted Certificate Authorities (CAs) and contain the server's public key, identifying information, and the digital signature from the CA. The client validates the certificate to ensure it has not been tampered with and belongs to the intended server.

4) Integrity: SSL and TLS ensure the integrity of data by using hash functions and digital signatures. Hash functions generate unique checksums for data, allowing the receiver to verify that the data has not been modified during transmission. Digital signatures provide an additional layer of integrity verification by enabling the receiver to verify the authenticity of the data and ensure it has not been tampered with.

By leveraging SSL and TLS, secure web communication is established through a safe and encrypted channel, ensuring the confidentiality, integrity, and authenticity of data exchanged between the client and the server. These protocols have become the industry standard for secure web communication, providing a foundation for secure e-commerce, online banking, and other sensitive online transactions.

C. Overview of SSL and TLS Versions:

SSL (Secure Sockets Layer) and TLS (Transport Layer Security) are cryptographic protocols used for secure communication over the internet [4]. They have evolved with multiple versions, each introducing improvements in security and functionality. The major versions of SSL and TLS are as follows:

- SSL 1.0: This was the first version of SSL, but it had significant security flaws and was quickly deprecated.
- SSL 2.0: Released in 1995, SSL 2.0 addressed some of the security vulnerabilities in SSL 1.0. However, it still has several weaknesses, and its usage is strongly discouraged due to security concerns.
- SSL 3.0: Introduced in 1996, SSL 3.0 improved upon its predecessors and became widely adopted. It provided stronger encryption algorithms and better security features. However, vulnerabilities such as POODLE and BEAST led to its deprecation.
- TLS 1.0: TLS 1.0, released in 1999, was an upgrade to SSL 3.0. It introduced stronger security mechanisms, including better integrity checks and protection against cypher block chaining attacks.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

- While still widely supported, it is considered outdated and insecure due to various vulnerabilities.
- TLS 1.1: Released in 2006, TLS 1.1 further enhanced security by addressing vulnerabilities in TLS 1.0. It introduced support for newer cryptographic algorithms and improved protection against cypher suite downgrade attacks.
- TLS 1.2: Released in 2008, TLS 1.2 brought significant security enhancements. It introduced stronger cypher suites, improved protocol negotiation, and enhanced protection against known security vulnerabilities. TLS 1.2 is widely supported and considered the minimum recommended version for secure web communication.
- TLS 1.3: Released in 2018, TLS 1.3 is the latest version of the Transport Layer Security (TLS) protocol. It provides substantial improvements in security, performance, and privacy. TLS 1.3 features a simplified handshake, enhanced encryption algorithms, and forward secrecy by default. It also reduces latency and mitigates various attacks, making it the most secure and efficient version of TLS.

D. Handshake Process: Key Exchange, Authentication, and Session Setup

The SSL/TLS handshake is a process that occurs at the beginning of an HTTPS connection. It establishes the parameters for secure communication, ensuring the confidentiality and integrity of subsequent data exchange. The handshake process involves the following steps:

- Client Hello: The client initiates the handshake by sending a Client Hello message to the server. This message includes the highest supported SSL/TLS version, a random number, and a list of supported cypher suites.
- Server Hello: The server responds with a Server Hello message, selecting the highest SSL/TLS version mutually supported by the client and server. The server also sends its digital certificate, which contains the server's public key.
- Certificate Validation: The client validates the server's certificate to ensure its authenticity. It checks the certificate's digital signature, expiration date, and verifies that it has been issued by a trusted Certificate Authority (CA).
- Key Exchange: The client generates a random session key and encrypts it using the server's public key, which is obtained from the server's certificate. The encrypted session key is sent to the server.
- Session Setup: The server decrypts the session key using its private key, which is unique to the server. Both the client and the server now share the same session key, which will be used for symmetric encryption and decryption of data during the session.
- Cypher Suite Negotiation: The client and server agree on a cypher suite from the list provided by the client during the ClientHello. The cypher suite determines the encryption algorithms, key exchange mechanism, and message authentication

IV. ATTACK 1: MAN-IN-THE-MIDDLE (MITM)

A Man-in-the-Middle (MitM) [10] attack is a security threat that targets the integrity and confidentiality of HTTPS communications. In this attack, an adversary intercepts the communication between a client and a server, effectively positioning themselves as a proxy between the two parties. The attacker can eavesdrop on the exchanged data, modify it, or inject malicious content without the knowledge of the client or the server.

A. Working and Proof-of-Concept (PoC) Demonstration

The following steps illustrate the working of a MitM attack against HTTPS [10]:

- Intercepting the Communication: The attacker positions themselves between the client and the server by either compromising a network device, exploiting vulnerabilities in network protocols, or conducting attacks like ARP spoofing [2]. This allows them to intercept the HTTPS communication.
- Impersonating the Server: The attacker generates a fake SSL/TLS certificate that resembles the legitimate server's certificate. A self-signed or compromised certificate authority signs this certificate.
- Initial Handshake: The client initiates the SSL/TLS handshake by sending a Client Hello message, intending to establish a secure connection with the server.
- Attacker's Response: The attacker intercepts the Client Hello message and responds to the client, pretending to be the server. The attacker's response includes their fake certificate.
- Certificate Validation: The client receives the attacker's fake certificate and performs validation on it. If the client does not properly validate the certificate or if the attacker has employed techniques such as Certificate Authority (CA) compromise, the client may consider the certificate as legitimate.
- Session Setup: Assuming the attacker's certificate is genuine, the client proceeds with the SSL/TLS handshake and establishes a secure connection with the attacker, mistakenly believing it to be the actual server.
- Data Interception or Modification: With the secure connection established, the attacker can now intercept, monitor, modify, or inject malicious content into the HTTPS communication between the client and the server. This can include stealing sensitive information, such as login credentials or financial data, or injecting malware or phishing content into systems.

B. Mitigation Strategies: Certificate Validation, Public Key Pinning

To mitigate MitM [10] attacks against HTTPS, the following strategies can be employed:

- Certificate Validation: Clients must validate the authenticity of the server's SSL/TLS certificate. This involves verifying the digital signature, expiration date, and the certificate chain leading to a trusted Certificate Authority (CA). Implementing certificate pinning, where the client stores and trusts specific certificate details, can add an extra layer of protection against forged or compromised certificates.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

- Public Key Pinning: Public key pinning is a technique where clients associate specific public keys with the server's domain. This ensures that only certificates signed by the authorized key(s) are accepted, preventing attackers from using fake or compromised certificates. Implementing HTTP Strict Transport Security (HSTS) headers and preloading can enforce the use of HTTPS and pinning mechanisms.
- Mutual Authentication: Employing mutual authentication can enhance security. In addition to the client validating the server's certificate, the server can also request that the client present a valid certificate, thereby ensuring the client's identity. This prevents attacks where the client unknowingly connects to a malicious server.
- Certificate Transparency: Implementing Certificate Transparency mechanisms can increase the visibility of certificate issuance and help detect any unauthorised or fraudulent certificates. Certificate Transparency logs allow the monitoring of certificate issuance and help identify potentially malicious certificates.

V. ATTACK 2: SSL/TLS STRIPPING

SSL/TLS Stripping is a type of attack that targets the downgrade of secure HTTPS connections to insecure HTTP connections [1][3]. This attack exploits the fact that many websites still support both HTTP and HTTPS protocols. The attacker intercepts the initial HTTPS request made by the client and downgrades it to an unencrypted HTTP request, removing the security provided by SSL/TLS. This allows the attacker to intercept and manipulate the communication between the client and the server, potentially compromising the confidentiality and integrity of the data.

A. Attack Methodology & Proof of Concept

The SSL/TLS Stripping attack typically involves the following steps:

- Intercepting the HTTPS Request: The attacker positions themselves between the client and the server, either through network-level attacks like ARP spoofing or by compromising an intermediary device [2]. When the client attempts to establish an HTTPS connection, the attacker intercepts the initial request.
- Modification of Response: The attacker modifies the server's response to the client, replacing HTTPS links with plain HTTP links. This manipulation causes the client's browser to establish an unencrypted HTTP connection instead of the intended HTTPS connection.
- Downgrade to HTTP: The modified response is sent back to the client, which, due to the attacker's modifications, contains HTTP links instead of HTTPS links. The client's browser, unaware of the manipulation, proceeds to connect to the server using an insecure HTTP connection.
- Communication Interception: With the communication downgraded to HTTP, the attacker can intercept and read the exchanged data between the client and the server. This includes any sensitive information transmitted over the connection, such as login credentials or personal data.

A PoC for SSL/TLS Stripping would involve setting up a controlled environment where the attacker can intercept the communication between a client and a server [1][4]. By using tools such as a proxy server or network sniffing software, the attacker can demonstrate the ability to modify HTTPS responses, replace HTTPS links with HTTP links, and intercept the data transmitted over the downgraded HTTP connection.

B. Mitigation Strategies

To mitigate SSL/TLS Stripping attacks, the following strategies can be implemented:

- HSTS (HTTP Strict Transport Security): Websites can implement HSTS. This security feature instructs the client's browser to only communicate with the server over HTTPS, even if the user enters an HTTP URL or follows a link with an HTTP protocol. HSTS ensures that the connection is always encrypted and prevents attackers from downgrading to insecure HTTP connections.
- HTTPS Enforcement: Websites should enforce the use of HTTPS by redirecting all HTTP requests to their HTTPS counterparts. This can be achieved through server-side configurations or the use of web application firewalls (WAFs) that enforce HTTPS connections. By redirecting all traffic to HTTPS, the risk of SSL/TLS Stripping attacks is significantly reduced [1].
- Network Monitoring and Intrusion Detection: Deploying network monitoring and intrusion detection systems can help detect and identify SSL/TLS Stripping attacks. These systems can detect anomalies, such as sudden downgrades from HTTPS to HTTP, and raise alerts for further investigation and mitigation.

VI. ATTACK 3: BEAST

The BEAST (Browser Exploit Against SSL/TLS) attack is a cryptographic attack that targets the SSL/TLS protocols used in HTTPS [4]. It exploits a vulnerability in the implementation of the Cypher Block Chaining (CBC) mode of encryption. The attack allows an attacker to decrypt portions of an encrypted HTTPS session, potentially compromising the confidentiality of sensitive information.

The impact of the BEAST attack on HTTPS is significant as it poses a threat to the security of data transmitted over encrypted connections. By decrypting parts of the session, an attacker could potentially extract sensitive information, such as login credentials or session cookies, thereby enabling unauthorised access or session hijacking.

A. Attack Methodology and Proof-of-Concept

The BEAST attack follows the following steps:

- Initialization Vector (IV) Discovery: The attacker initiates a large number of HTTPS requests, each containing a known plaintext block [5]. By observing the resulting encrypted ciphertext, the attacker can deduce information about the initialisation vector (IV) used in the CBC encryption.
- Byte-by-Byte Decryption: The attacker systematically decrypts one byte of the targeted encrypted block by sending a crafted HTTPS request. The attacker manipulates the plaintext of the request in such a way that the corresponding ciphertext, when decrypted, provides information about the targeted byte.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

- Exploiting Cookie Vulnerabilities: The attacker focuses on capturing and decrypting the session cookie, which is typically sent with each HTTPS request. By extracting the session cookie, the attacker can gain unauthorized access to the user's session.

Proof-of-concept (PoC) demonstrations of the BEAST attack have been performed in controlled environments, demonstrating the ability to decrypt parts of encrypted communication. These demonstrations highlight the vulnerability of the CBC mode of encryption in SSL/TLS and the need for effective mitigation strategies.

B. Mitigation Strategies

To mitigate the BEAST attack and enhance the security of HTTPS, the following strategies can be employed:

- Upgrade to TLS 1.1 or TLS 1.2: The BEAST attack primarily targets SSL 3.0 and TLS 1.0, which are vulnerable to the attack. Upgrading to TLS 1.1 or TLS 1.2, which have improved security mechanisms, helps protect against the BEAST attack [6]. TLS 1.1 and TLS 1.2 use countermeasures, such as per-record Initialization Vectors (IVs), to mitigate the vulnerability exploited by the BEAST attack.
- Cypher Suite Selection: Selecting a secure cypher suite is crucial in mitigating the BEAST attack. Preferably, cypher suites that use authenticated encryption, such as those based on the Galois/Counter Mode (GCM), should be used. These cypher suites offer enhanced security and are not susceptible to the BEAST attack. It is essential to prioritize cipher suites that support TLS 1.1 and TLS 1.2 and have strong encryption algorithms, key exchange mechanisms, and message authentication codes [9].
- Implementing Forward Secrecy: Forward secrecy, also known as Perfect Forward Secrecy (PFS), ensures that even if the server's private key is compromised in the future, past encrypted communications remain secure. By supporting cypher suites that provide forward secrecy, such as Diffie-Hellman-based key exchange algorithms (e.g., DHE and ECDHE), the impact of a successful BEAST attack can be significantly mitigated.
- TLS Renegotiation: Disable or strictly control the usage of TLS renegotiation, as it can be exploited to facilitate the BEAST attack. Renegotiation introduces additional complexities in the SSL/TLS protocol, which can potentially be leveraged by attackers.

VII. ATTACK 4: CRIME

The CRIME (Compression Ratio Info-leak Made Easy) attack is a security vulnerability that affects the security of HTTPS communication. It leverages the use of data compression within the SSL/TLS protocol to decrypt sensitive information, such as session cookies, transmitted over an encrypted connection.

A. Attack Scenario and Proof of Concept

The CRIME attack follows the following steps:

- Compression Negotiation: The attacker initiates a connection with the target server and engages in the SSL/TLS handshake process. During this process, the compression method used in the communication is negotiated between the client and the server.
- Injecting Malicious Code: The attacker injects malicious JavaScript or other code into a web page that the victim visits. This code performs repetitive requests to the target server while modifying a single character of the data being transmitted in each request.
- Data Analysis: By carefully controlling the injected data and monitoring the size of the encrypted response, the attacker can deduce information about the original plaintext. Through a process of trial and error, the attacker can decrypt sensitive information, such as session cookies, one byte at a time.

Proof-of-concept (PoC) demonstrations of the CRIME attack have been successfully performed, showcasing the ability to decrypt session cookies and gain unauthorised access to user accounts. These demonstrations highlight the potential threat that CRIME poses to the confidentiality of data transmitted over HTTPS connections.

B. Mitigation Strategies

To mitigate the CRIME attack and enhance the security of HTTPS, the following strategies can be employed:

- Disabling TLS/SSL Compression: The CRIME attack relies on the use of compression within the SSL/TLS protocol. Disabling compression mitigates the vulnerability exploited by the attack. Compression should be disabled on both the server and client sides. This can be achieved by modifying the server's SSL/TLS configuration or utilizing appropriate security frameworks and libraries.
- Protocol Version Upgrades: Upgrading to newer versions of the SSL/TLS protocol, such as TLS 1.2 or TLS 1.3, can protect against the CRIME attack. These versions incorporate countermeasures to mitigate the vulnerability exploited by the attack. By ensuring that both the server and client support the latest secure protocol versions, the risk of a CRIME attack can be significantly reduced.
- Implementing Context-Specific Encryption: Applying encryption selectively, based on the type of data being transmitted, can provide an additional layer of protection [8]. For example, encrypting sensitive data, such as session cookies or authentication tokens, using application-level encryption mechanisms can prevent their exposure even if a vulnerability like CRIME exists.
- Web Application Hardening: Implementing secure coding practices and performing regular security audits of web applications can help identify and mitigate vulnerabilities that can be exploited in attacks, such as the CRIME attack. Proper input validation, output encoding, and secure session management are crucial to prevent attacks targeting sensitive data.

By implementing these mitigation strategies, organizations can significantly reduce the risk of the CRIME attack and protect the confidentiality of data transmitted over HTTPS connections. Continuous monitoring and proactive security measures are crucial for staying ahead of evolving threats.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

VIII. CONCLUSION

In conclusion, this research paper explored various attacks against HTTPS, focusing on both SSL and TLS protocols [7]. The importance of HTTPS in securing web communications was emphasized, highlighting its role in protecting sensitive data and ensuring the privacy and integrity of online transactions. The paper provided an overview of the HTTPS protocol, discussing its key principles and the fundamental role of SSL and TLS in establishing secure connections.

DECLARATION STATEMENT

Funding	No, I did not receive.
Conflicts of Interest	No conflicts of interest to the best of our knowledge.
Ethical Approval and Consent to Participate	No, the article does not require ethical approval or consent to participate, as it presents evidence that is not subject to interpretation.
Availability of Data and Material	Not relevant.
Authors Contributions	All authors have equal participation in this article.

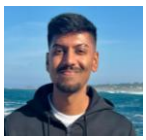
REFERENCES

1. S. Puangpronpitag and N. Sriwiboon, "Simple and Lightweight HTTPS Enforcement to Protect against SSL Striping Attack," 2012 Fourth International Conference on Computational Intelligence, Communication Systems and Networks, Phuket, Thailand, 2012, pp. 229-234, doi: 10.1109/CICSyN.2012.50.
2. Nagendran, K., et al. "Sniffing HTTPS Traffic in LAN by Address Resolution Protocol Poisoning." International Journal of Pure and Applied Mathematics 119.12 (2018): 1187-1195.
3. A. Adithyan, K. Nagendran, R. Chethana, G. Pandy D. and G. Prashanth K., "Reverse Engineering and Backdooring Router Firmwares," 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2020, pp. 189-193, doi: 10.1109/ICACCS48705.2020.9074317.
4. P. Sirohi, A. Agarwal and S. Tyagi, "A comprehensive study on security attacks on SSL/TLS protocol," 2016 2nd International Conference on Next Generation Computing Technologies (NGCT), Dehradun, India, 2016, pp. 893-898, doi: 10.1109/NGCT.2016.7877537.
5. V. Platenka, A. Mazalek and Z. Vranova, "Attacks on devices using SSL/TLS," 2021 International Conference on Military Technologies (ICMT), Brno, Czech Republic, 2021, pp. 1-6, doi: 10.1109/ICMT52455.2021.9502818.
6. F. Qi, Z. Tang and G. Wang, "Attacks vs. Countermeasures of SSL Protected Trust Model," 2008 The 9th International Conference for Young Computer Scientists, Hunan, China, 2008, pp. 1986-1991, doi: 10.1109/ICYCS.2008.433.
7. G. Rajendran, H. V. Sathyabalu, M. Sachi and V. Devarajan, "Cyber Security in Smart Grid: Challenges and Solutions," 2019 2nd International Conference on Power and Embedded Drive Control (ICPEDC), Chennai, India, 2019, pp. 546-551, doi: 10.1109/ICPEDC47771.2019.9036484
8. P. P. Parthy and G. Rajendran, "Identification and prevention of social engineering attacks on an enterprise," 2019 International Carnahan Conference on Security Technology (ICCST), Chennai, India, 2019, pp. 1-5, doi: 10.1109/CCST.2019.8888441
9. R. Oppliger, R. Hauser and D. Basin, "SSL/TLS Session-Aware User Authentication," in Computer, vol. 41, no. 3, pp. 59-65, March 2008, doi: 10.1109/MC.2008.98.
10. S. Stricot-Tarboton, S. Chaisiri and R. K. L. Ko, "Taxonomy of Man-in-the-Middle Attacks on HTTPS," 2016 IEEE Trustcom/Big Data SE/ISPA, Tianjin, China, 2016, pp. 527-534, doi: 10.1109/TrustCom.2016.0106.

AUTHORS PROFILE

Adithyan Arun Kumar is a distinguished Product Security Engineer at Salesforce, California, with a rich background in Information Security from Carnegie Mellon University. Specialising in risk evaluation, threat modelling, penetration testing, and security automation, Adithyan has a notable history of automating threat models during his tenure at Salesforce and the TEEL Lab at Carnegie Mellon. His contributions to the field are recognised by major corporations such as Apple and Microsoft, where he is inducted into their Hall of Fame.

Gowthamaraj Rajendran is a Security Researcher at Splunk, California, with a rich background in Threat detection and Red Teaming. He also holds certificates such as OSWE, OSWA, OSCP, and CRTP. He has authored around 10+ CVEs to date.



Nitin Srinivasan is a Software Engineer III at Google in California, where he works on the ML Developer Infrastructure and Security team. He has made significant contributions to prominent ML Frameworks such as TensorFlow, enhancing build efficiency and expanding its accessibility and usability across various platforms. He holds a Master's degree in Computer Science from the University of Massachusetts Amherst, where his focus was on deep learning and natural language processing.

Praveen Kumar Sridhar is a highly skilled Machine Learning Engineer and Research Assistant with an extensive background in data science, natural language processing (NLP), and machine learning. Holding a Master's degree in Data Science from Northeastern University, he has contributed significantly to the fields of machine learning and NLP. His research and development efforts include the creation of an advanced moderation system for social media platforms, the development of NLP pipelines utilizing architectures such as Llama and GPT, and the creation of churn frameworks. These contributions demonstrate his capacity to innovate and push the boundaries of AI and machine learning technologies.

Kishore Kumar Perumalsamy works as a Software Engineer at Adobe Headquarters in San Jose, California, with a fervour for building secure, intelligent, and scalable software systems. He earned his Master's in Mobile and IoT Engineering from Carnegie Mellon University, serving as a Teaching Assistant for Software Architecture and a Research Assistant in the renowned TEEL Labs of the CS department.

Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)