The Invisible Storm: Risks of Neglected Cloud Security

Alexander Nin
CS3100
Utah Valley University
Orem, UT, U.S.A
10998493@uvu.edu

***Abstract*—This research involves navigating cloud environments in an attempt to understand what makes a poorly secured cloud environment particularly susceptible to cyber-attacks. We live in an era where web spaces are becoming essential to our well-being as electricity was in the early 20th century. The difference, however, attacks on power stations and grids were not nearly as prevalent as they are with cloud computing. As we can see, the ease of connectivity brought a lot of bad with the good, to put it simply. The objective of this project is to address major issues surrounding cloud computing infrastructure and services. It aims to understand and analyze current programming standards and security measures in the attempt to discover potential problems and find solutions.**

***Index Terms*—**

I. Introduction

The evolving threat landscape for online communication and increased demand for secure digital platforms raise the need for robust cryptographic suites that ensure optimal online information confidentiality. Different protocols and cipher suites have been developed to provide such functionalities. One such cipher suite is the Transport Layer Security (TLS), which provides a framework for establishing encrypted connections between networked applications. This cipher suite has undergone substantial revisions and upgrades to address emerging network security and performance requirements. The current cipher suite is TLS version 1.3, incorporating diverse features and functionalities. This essay examines the incorporation of Zero Round-Trip Time into TLS 1.3, assessing its advantages and disadvantages. Through this focus, the report offers recommendations to enhance the viability of this feature in addressing evolving network security needs.

II. Various Cipher Suites Used in TLS

Ciphers denote algorithms for executing cryptographic functions, such as hashing, encryption, decryption, and digital signatures. According to Messier (2017), some of the most common types of cipher suites include Transport Layer Security version 1.2 (TLS 1.2), Rivest–Shamir–Adleman (RSA), and Advanced Encryption Standard (AES). The primary purpose of the TLS protocol is to offer data integrity and privacy between communicating applications (Dierks & Rescorla, 2008). TLS comprises two layers: the TLS Handshake Protocol and the TLS Record Protocol. The TLS Handshake Protocol enables the server and client to authenticate each other and negotiate cryptographic keys and encryption algorithms before applications send or receive data. The record layer carries Content-layer messages as typed TLS records (Thomson & Turner, 2021). By contrast, the TLS Record Protocol facilitates the encapsulation of higher-level protocols. These processes allow the secure communication of applications to reduce the risk of attacks or communication interception.

RSA is another type of cipher suite used in modern cryptographic systems. The cryptosystem stands for Rivest, Shamir, and Adleman, using the exponentiation of prime numbers to achieve security (Heng, 2010). RSA has a simple operation mechanism that begins with the client using a public key a server sends to encrypt the pre-master secret and transmit it (Pitt, 2006). The server proceeds to use a private key to decrypt the pre-master secret, and both parties rely on a PRF, the server random, the client random, and the pre-master secret to derivate the master secret. After that, the parties use pseudo-random and the master secret to calculate the session key.

Advanced Encryption Standard (AES) is another type of NIST-approved cipher suite. It has a block size of 128 bits and uses symmetric keys that are 128, 192, or 256 bits long (Daemen & Rijmen, 2021). AES operates through four-by-four arrays known as ”states.” Under this cryptosystem, plaintext undergoes substitutions based on a lookup table where rows are changed cyclically, and the linear mixing of operations allows the combination of bytes in a column (Stapko, 2011). AES is a cipher often supported by TLS 1.2 and TLS 1.3, and it serves as a stream cipher that includes the capability to authenticate messages. Modern technologies include the AES cipher suite as a default tool, for example, Open Client 12.5.1 and Adaptive Server 12.5.3. Administrators and users no longer have to take any action to use this cipher suite. However, some clients can restrict the algorithm, necessitating action to remove the limitations. The AES cipher suite provides the most robust encryption in some technologies, such as Adaptive Server.

A. Importance of Forward Secrecy

Forward secrecy is a critical feature of cryptographic suites, providing four key benefits. Firstly, it offers protection against key compromise by ensuring that even if an attacker gains access to long-term private keys, they cannot retroactively decrypt past communications (Li, 2019). This resilience mitigates the impact of key compromise and prevents unauthorized access to sensitive information exchanged in previous sessions. Secondly, forward secrecy provides resilience against future threats and cryptographic advances (Li, Su & Wang, 2020). This phenomenon ensures that past communications remain secure despite the evolving attack vectors. Thirdly, it enhances user privacy by thwarting adversaries' attempts to decrypt and eavesdrop on past communications (Li, 2019). This functionality is crucial in scenarios involving sensitive or personal information exchange, guaranteeing the data's confidentiality. Lastly, forward secrecy safeguards online presence against massive surveillance initiatives by big corporations and governments (Li, Su & Wang, 2020). This feature preserves privacy and civil liberties in an era of pervasive data interception and unwarranted intrusion into personal communications. Thus, adopting forward secrecy in TLS cipher suites is critical in guaranteeing security and privacy for secure online communication.

III. Best Practices in Programming Cloud Environments

When using server suites for data integrity and privacy, it is crucial to follow some best practices. First, users or administrators must avoid using mixed content to improve performance. They should ensure that JavaScript images, files, and CSS files are accessed using SSL/TLS to reduce the potential for browser security warnings and SEO issues (Gilchrist, 2015). Second, it is essential to include secure cookies to enhance data integrity and privacy. Setting secure cookies reinforces transmission via secure channels and can limit cross-site usage of cookies, which can threaten data security (McCafferty, 2022). Third, website owners and administrators must assess third-party code to ensure security and privacy. Third-party libraries and tools can introduce website vulnerabilities that affect cipher suites' performance (Grigorik, 2013). Vetting these tools and libraries is critical to guarantee trustworthiness and capability to maintain data integrity and privacy during communication.

IV. Transport Layer Security

Transport Layer Security (TLS) is a cryptographic protocol that ensures secure communication over a computer network, commonly the internet. It operates by providing encryption, authentication, and data integrity, safeguarding against eavesdropping, tampering, and forgery. TLS protocols establish a secure connection between a client and a server, facilitating the exchange of sensitive information such as passwords, credit card numbers, and personal data. It employs a combination of symmetric and asymmetric encryption techniques, typically using public-key cryptography for key exchange and symmetric encryption for data transmission. TLS is widely employed in web browsing, email, messaging applications, and other network services to protect users' privacy and data confidentiality. In this section we will navigate the framework for thrid party implementation as well as TLS's session resumption abilities.

A. Third Party TLS Implementation

Third-party implementation of TLS is possible through software libraries created by entities other than those maintaining TLS specifications. Some popular third-party tools for implementing TLS are OpenSSL and LibreSSL (Lee, 2021). OpenSSL is among the most commonly used implementations of TLS protocols because it provides strong cryptographic functions and enables different versions of TLS. For example, OpenSSL allows symmetric and asymmetric encryption. Symmetric encryption permits data exchange between applications during TLS sessions, while asymmetric encryption is helpful for digital signatures and key exchanges (Ristic, 2013). The features allow the implementation of TLS protocols by reducing the risk of attackers eavesdropping on communication and improving communication efficiency. Therefore, OpenSSL supports asymmetric encryption to ensure key security and symmetric encryption to encrypt data during transmission, allowing TLS implementation.

LibreSSL is another third-party implementation of TLS protocol that seeks to guarantee security while ensuring compatibility with current applications that rely on OpenSSL. LibreSSL is similar to OpenSSL because it supports symmetric and asymmetric encryption to enhance security and performance. Besides, LibreSSL incorporates certificate management capabilities comprising functions such as parsing, manipulating, and validating X.509 certificates. X.509 digital certificates are helpful in TLS because they enable peer authentication, supporting certificate revocation lists, validation processes, and chains (Opplinger, 2023). Another critical role of LibreSSL in TLS implementation is the improvement of performance. LibreSSL facilitates the resumption of previous TLS sessions while restricting the need for full handshakes, resulting in enhanced performance.

B. Session Resumption Through One Round Trip Time

The One Round-Trip Time (1-RTT) in TLS 1.3 is critical in establishing robust forward secrecy while maintaining efficient and secure communication between the client and server. This handshake involves a single round trip between the

client and server, during which a shared secret is established using the Diffie-Hellman key exchange algorithm (Kumari & Mohapatra, 2022). This shared secret serves as the foundation for deriving session keys that are unique to each session and are used for encrypting and authenticating subsequent communication. Using ephemeral key pairs and session keys ensures forward secrecy, as compromising long-term secrets does not jeopardize the confidentiality of past communications encrypted with session keys from previous sessions (Chen, 2021). Further, the 1-RTT handshake includes mechanisms for authentication and integrity verification. This mechanism involves the exchange of digital signatures or authentication certificates to verify the parties' identities and ensure data integrity (Dowling, 2021). Through these processes, the 1-RTT handshake establishes secure connections efficiently, providing robust protection against eavesdropping, tampering, and identity spoofing.

C. Session Resumption Through Zero Round Trip Time

The Zero Round-Trip Time (0-RTT) feature introduced in Transport Layer Security (TLS) 1.3 enhances the performance and efficiency of secure communication over the Internet. The feature allows the client to send encrypted data to the server in the first message without waiting for a response (Aviram, Gellert & Jager, 2021). This phenomenon reduces latency and speeds up the connection establishment process, leading to faster page load times and improved user experience. In 0-RTT, the client utilizes session resumption mechanisms, such as session tickets or session identifiers, to resume previous connections without a full handshake (Goncharskyi., 2022). This feature allows returning users to bypass the initial handshake steps and immediately send encrypted data to the server, resulting in quicker access to content and reduced network overhead. Thus, 0-RTT offers a valuable feature for optimizing web performance and enhancing the efficiency of secure communication protocols like TLS.

V. 0-RTT, An Analysis

A. The Adoption of Zero Round Trip Time

There is negligible implementation of Zero Round-Trip Time (0-RTT) despite the promises the feature offers for web services. A review of websites implementing TLS 1.3 0-RTT using Wireshark and manual approaches indicated that no websites leveraged 0-RTT. This finding underscores a significant gap between the theoretical potential of 0-RTT and its practical adoption in real-world web environments. While 0-RTT holds the potential to significantly improve performance and efficiency by reducing latency and speeding up connection setup, its limited implementation suggests barriers to adoption or awareness among website operators and developers. Understanding the reasons behind this discrepancy is crucial for unlocking the benefits of 0-RTT and maximizing its impact on web services. Such an approach relies on awareness and industry collaboration to enhance the adoption of 0-RTT. Such a venture's success depends on an adequate understanding of the limitations of 0-RTT, potentially contributing to its negligible adoption.

B. Pros and Cons of Session Resumption with 0-RTT

The 0-RTT has five main benefits in web services. Firstly, by allowing the client to send encrypted data in the initial message without waiting for round trips, 0-RTT significantly reduces latency, resulting in faster page load times and smoother user interactions (Dallmeier, 2020). This streamlined connection setup process is particularly beneficial for content-heavy websites or applications, enabling quicker access to content and improving engagement and satisfaction. Secondly, 0-RTT leverages session resumption mechanisms to enable faster load times for repeat visitors, eliminating the need to renegotiate connection parameters and further enhancing the browsing experience (Cao, Zhao & Zhang, 2019). Thirdly, 0-RTT improves the efficiency of servers, especially in stateless architectures, by reducing computational overhead and enabling more efficient processing of incoming requests. This feature enhances server scalability and resource utilization (Abdallah, Kuang & Huang, 2022). Further, 0-RTT enhances mobile performance by reducing the number of round trips required to establish a connection, resulting in faster loading times and smoother interactions for users on mobile devices, particularly in scenarios with unstable or congested network conditions (Dallmeier, 2020). Lastly, 0-RTT supports real-time applications by enabling quick connection establishment and timely updates to users, maintaining the continuity and quality of real-time communication, and ensuring a smooth and immersive experience (Göth, 2023). Thus, 0-RTT provides robust functionalities that improve the overall web experience.

Despite these benefits, there are limitations attributed to 0-RTT. Firstly, compatibility issues with certain TLS implementations or environments can hinder its widespread adoption, creating challenges in maintaining consistent and secure communication across diverse platforms and configurations (Abdallah, Kuang & Huang, 2022). This phenomenon undermines the effective orchestration of services on the web platform, potentially leading to fragmented user experiences and interoperability issues. Secondly, the reduced latency and faster connection setup facilitated by 0-RTT make it susceptible to exploitation by attackers for launching Denial-of-Service (DoS) attacks (Joarder & Fung, 2022). These attacks inundate server resources, disrupting legitimate traffic and causing service downtime. As a result, the overall Quality of Service for hosted applications on the platform is compromised, impacting user satisfaction and trust.

Thirdly, 0-RTT is vulnerable to replay attacks due to the absence of forward secrecy. Zero Round-Trip Time allows clients to transmit encrypted data to the server in the initial message without awaiting a response. This feature enables hackers to intercept and maliciously replay these messages through man-in-the-middle attacks (Abdelhafez, Ramadass & Gismallab, 2023). Such attacks allow the attackers to potentially gain unauthorized access to sensitive information or execute data manipulation and injection attacks, compromising confidentiality and, at times, the integrity of the transmitted messages. Lastly, the lack of forward secrecy in 0-RTT compromises the confidentiality of past communications (Wei, 2023). Since the initial data exchange relies on session keys derived from the server's long-term secret, a compromised key could expose previously encrypted data. This vulnerability increases the risk of privacy violations, data breaches, and the exposure of sensitive information in the event of a key compromise (Wei, 2023). Hence, addressing these security concerns is essential to ensuring the integrity and confidentiality of communications in 0-RTT deployments within the 1.3 architecture.

C. Adjustments To 0-RTT

There are three crucial adjustments to 0-RTT to accommodate forward secrecy without significant tradeoffs. Firstly, there is a need to integrate ephemeral key exchange mechanisms into the 0-RTT handshake process. This consideration helps generate session keys unique to each connection, fostering forward secrecy (Aviram, Gellert & Jager, 2021). Secondly, there is a need to leverage robust cryptographic techniques such as key encapsulation to strengthen encryption frameworks. Techniques like the Elliptic Curve Diffie-Hellman key exchange or pre-shared keys derive session keys and offer robust encryption features that effectively safeguard the secrecy of the shared keys (Cebe, M., & Akkaya, 2019). These approaches maintain confidentiality and integrity while minimizing computational overhead. Lastly, there is a need to optimize the cryptographic and network protocols used in 0-RTT (Murthy, Asghar & Tu, 2022). This approach provides a robust and efficient 0-RTT implementation that offers forward secrecy without sacrificing performance or latency. Thus, these adjustments provide a strong balance between security and efficiency in 0-RTT, enhancing the overall resilience of TLS 1.3 protocols.

VI. Conclusion

Forward secrecy is a critical component within cryptographic suites, offering multifaceted benefits for secure online communication. Its role in protecting against key compromise, resilience against future threats, enhancement of user privacy, and defense against mass surveillance underscores its significance in maintaining security and privacy in digital interactions. However, integrating forward secrecy into Zero Round-Trip Time (0-RTT) in TLS 1.3 necessitates careful adjustments to balance security and efficiency effectively. By incorporating ephemeral key exchange mechanisms, leveraging robust cryptographic techniques, and optimizing cryptographic and network protocols, 0-RTT can accommodate forward secrecy without significant tradeoffs. These adjustments ensure that 0-RTT maintains its low-latency capabilities while offering robust protection against eavesdropping, tampering, and identity spoofing. By embracing forward secrecy and implementing tailored adjustments to 0-RTT, TLS 1.3 protocols enhance online communication channels' resilience and security, safeguarding user privacy and confidentiality.

VII. References

Abdallah, E. G., Kuang, R., & Huang, C. (2022). Generating just-in-time shared keys (JIT-SK) for TLS 1.3 zero Round-Trip Time (0-RTT). International Journal of Machine Learning and Computing, 12(3), 96-101.

Abdelhafez, M. E., Ramadass, S., & Gismallab, M. S. (2023, August). Replay Attack in TLS 1.3 0-RTT Handshake: Countermeasure Techniques. In 2023 IEEE 6th International Conference on Electrical, Electronics and System Engineering (ICEESE) (pp. 44-49). IEEE.

Aviram, N., Gellert, K., & Jager, T. (2021). Session resumption protocols and efficient forward security for TLS 1.3 0-RTT. Journal of Cryptology, 34(3), 1-57. https://doi.org/10.1007/s00145-021-09385-0

Cao, X., Zhao, S., & Zhang, Y. (2019, December). 0-RTT attack and defense of QUIC protocol. In 2019 IEEE Globecom Workshops (pp. 1-6). IEEE.

Cebe, M., & Akkaya, K. (2019, December). A replay attack-resistant 0-RTT key management scheme for low-bandwidth smart grid communications. In 2019 IEEE Global Communications Conference (GLOBECOM) (pp. 1-6). IEEE.

Chen, S., Jero, S., Jagielski, M., Boldyreva, A., & Nita-Rotaru, C. (2021). Secure communication channel establishment: TLS 1.3 (over TCP fast open) versus QUIC. Journal of Cryptology, 34(3), 1-41. https://doi.org/10.1007/s00145-021-09389-w

Daemen, J. & Rijmen, V. (2021). The design of Rijndael: The advanced encryption standard (AES). Springer.

Dallmeier, F., Drees, J. P., Gellert, K., Handirk, T., Jager, T., Klauke, J., ... & Wolf, R. (2020). Forward-secure 0-RTT goes live: implementation and performance analysis in QUIC. In Proceedings of the 19th International Conference on Cryptology and Network Security (pp. 211-231). Springer.

Dierks, T. & Rescorla, E. (2008). The transport layer security (TLS) protocol. https://www.rfc-editor.org/rfc/rfc5246#section-1.2

Dowling, B., Fischlin, M., Günther, F., & Stebila, D. (2021). A cryptographic analysis of the TLS 1.3 handshake protocol. Journal of Cryptology, 34(4), 1-69. https://doi.org/10.1007/s00145-021-09384-1

Gilchrist, A. (2015). The concise guide to SSL/TLS for DevOps. RG Consulting.

Goncharskyi, D., Kim, S. Y., Serhrouchni, A., Gu, P., Khatoun, R., & Hachem, J. (2022, May). Delay Measurement

of 0-RTT Transport Layer Security (TLS) Handshake Protocol. In 2022 8th International Conference on Control, Decision and Information Technologies (CoDIT) (Vol. 1, pp. 1450-1454). IEEE.

Göth, C., Ramacher, S., Slamanig, D., Striecks, C., Tairi, E., & Zikulnig, A. (2023, November). Optimizing 0-RTT Key Exchange with Full Forward Security. In Proceedings of the 2023 on Cloud Computing Security Workshop (pp. 55-68). ACM.

Grigorik, I. (2013). High-performance browser networking: What every web developer should know about networking and web performance. O'Reilly Media.

Heng, S., Wright, R. N. & Goi, B. (2010). Cryptology and network security: 9th International Conference, CANS 2010, Kuala Lumpur, Malaysia, December 12-14, 2010, Proceedings. Springer.

Joarder, Y. A., & Fung, C. (2022, October). A Survey on the Security Issues of QUIC. In 2022 6th Cyber Security in Networking Conference (pp. 1-8). IEEE.

Kumari, N., & Mohapatra, A. K. (2022). A comprehensive and critical analysis of TLS 1.3. Journal of Information and Optimization Sciences, 43(4), 689-703.

Lee, H., Kim, D. & Kwon, Y. (2021). TLS 1.3 in practice: How TLS 1.3 contributes to the Internet. https://dl.acm.org/doi/fullHtml/10.1145/3442381.3450057

Li, P., Su, J., & Wang, X. (2020). iTLS: Lightweight transport-layer security protocol for IoT with minimal latency and perfect forward secrecy. IEEE Internet of Things Journal, 7(8), 6828-6841.

Li, X., Peng, J., Obaidat, M. S., Wu, F., Khan, M. K., & Chen, C. (2019). A secure three-factor user authentication protocol with forward secrecy for wireless medical sensor network systems. IEEE Systems Journal, 14(1), 39-50.

McCafferty, S. (2022). Energy IoT architecture: From theory to practice. Artech House. Messier, R. (2017). Network forensics. Wiley.

Murthy, A., Asghar, M. R., & Tu, W. (2022). Towards a data-driven framework for optimizing security-efficiency tradeoff in QUIC. Security and Privacy, 5(1), 1-12. https://doi.org/10.1002/spy2.184

Opplinger, R. (2023). SSL and TLS: Theory and practice. Artech House. Pitt, E. (2006). Fundamental networking in JAVA. Springer.

Ristic, I. (2013). OpenSSL cookbook: A guide to the most frequently used OpenSSL and commands. Feisty Duck.

Stapko, T. (2011). Practical embedded security: Building secure resource-constrained systems. Elsevier Science.

Thomson, M. & Turner, S. (2021). RFC 9001: Using TLS to secure QUIC. https://www.rfc-editor.org/rfc/rfc9001.html

Wei, J., Chen, X., Wang, J., Susilo, W., & You, I. (2023). Towards secure asynchronous messaging with forward secrecy and mutual authentication. Information Sciences, 626, 114-132.