ESP-JETA
ESP Journal of Engineering & Technology Advancements
ISSN: 2583-2646 / Volume 2 Issue 2, June 2022 / Page No: 168-173
Paper Id: JETA-V2I2P118 / Doi: 10.56472/25832646/JETA-V2I2P118

Original Article

A Deep Dive Into SSL Certificate Storage Options: Google Cloud Secret Manager Vs. In-App for Enhanced Security and Scalability

Srinivas Adilapuram
Senior Application Developer, ADP Inc, USA.

Abstract: *The paper examines the security, scalability, and operational implications of storing SSL certificates within Spring Boot applications compared to storing them in Google Cloud Secret Manager. It explores their implementation processes, strengths, and vulnerabilities. By analyzing performance, security, and cost factors, the research provides insights into selecting the best approach for securing sensitive data, offering recommendations tailored to modern cloud application development.*

Keywords: *SSL certificate, Google Cloud Secret Manager, storing certificates within the application, Spring Boot Applications.*

I. INTRODUCTION

Communication and data exchange over the Internet has evolved over decades. With billions of users and bots generating and exchanging terabytes of data every day, the number of attack vectors and the magnitude of attack surfaces has grown exponentially. Thanks to the several technologies and measures ensuring safe online data transmission, the number of breaches is within a reasonable range. SSL certificates are a critical part of safe communication and data transmission over the Internet. They ensure that frequent, real-time exchanges of all forms of data, including sensitive data (e.g., user credentials, personal information) between the client and server (in web applications) are encrypted and safe during transit.

The layer of security SSL certificates offer extends to the Application Programming Interface (API), which facilitates communication and safe exchange of data between Web app components and among web apps. They offer this security through encryption and preventing man-in-the-middle attacks. SSL certificates also complement API's other authentication mechanisms, such as OAuth 2.0 or JWT, by ensuring that authentication tokens transmitted between clients and servers are secure. Leveraging their full potential requires proper handling (including storage), and for a specific range of applications, i.e., built on Spring Boot framework, it's possible to store these certificates locally (server) and on Google Cloud (handled by Secret Manager). Understanding which option is better can help you improve the security of your Spring Boot applications.

II. LITERATURE REVIEW

There is ample literature available on SSL certificates, and a significant segment of it is from the last decade. This time was especially significant for the development and widespread adoption of SSL certificates. Many papers discuss its storage, but it's typically not a focal point in these two dimensions (cloud and storing within the application). However, there is a paper on storing similar certificates in Blockchain [1]. However, if we expand our search paradigm to SSL certificate management, the amount of available literature expands significantly. But they don't discuss its storage as extensively either. One mention we found was for server-based storage [2]. An overlapping area is Certificate Transparency (CT), where SSL and TSL certificate storage is discussed, usually in the context of validation and authentication [3]. As the root of SSL certificate chain reliability, root stores have also been researched in detail [4].

The literature on securing sensitive data (and SSL's place in it) is more abundant in contrast. This includes their use and relevance in validating/securing free content websites [5]. There are also papers broadly discussing it in the context of data security [6]. There is also plenty of literature on the digital safety of on-premises servers and the safety of web applications hosted on local servers [7]. Security is also an important domain in papers focused on web application deployment, including on Google Cloud [8].

III. PROBLEM STATEMENT

Securing sensitive data such as SSL certificates is crucial for maintaining the integrity and security of Spring Boot applications, especially when deploying on cloud platforms like GCP. Storing SSL certificates directly within the application might seem straightforward, but it exposes the application to security risks, making the certificates vulnerable to unauthorized

access. On the other hand, using cloud-native solutions like Google Cloud Secret Manager provides a more secure, scalable, and centralized way to manage sensitive data, but it adds complexity and overhead to the deployment process.

IV. OVERVIEW OF SSL CERTIFICATES AND THEIR IMPORTANCE

SSL (Secure Sockets Layer) certificates are essential for establishing a secure, encrypted connection between clients and servers, ensuring that sensitive data such as passwords, API keys, and personal information is transmitted securely. In modern web applications, particularly those built with Spring Boot in a Java environment, SSL certificates play a pivotal role in enabling HTTPS, which is a standard for secure communication.

Spring Boot developers often work with APIs that handle sensitive data or require authentication. In such cases, SSL certificates prevent man-in-the-middle attacks, data tampering, and eavesdropping. By encrypting the data in transit, they ensure that only authorized clients and servers can access the information.

Spring Boot provides robust support for configuring SSL, making it straightforward to implement. By adding a keystore containing the SSL certificate and private key, developers can enable HTTPS with minimal configuration changes in the application properties file. Additionally, SSL plays a crucial role in securing REST APIs, ensuring that third-party consumers of the API can communicate safely.

Beyond encryption, SSL certificates also authenticate the server's identity, building trust with clients. This is particularly important in production environments where ensuring the legitimacy of the server is critical for compliance and user confidence.

V. STORING SSL CERTIFICATES WITHIN THE APPLICATION

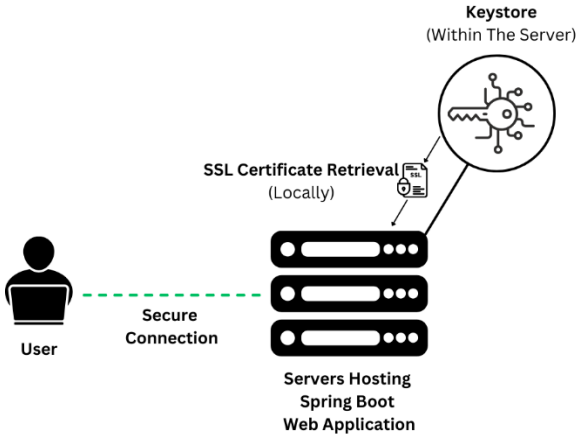

Figure-1: Retrieving Locally Stored SSL Certificates

In Spring Boot applications, SSL certificates are commonly stored locally within a keystore (essentially within the application), a secure file format used in Java environments to store cryptographic keys and certificates. Configuring SSL in Spring Boot involves referencing this keystore in the application properties file and specifying parameters such as *server.ssl.key-store* (path to the keystore), *server.ssl.key-store-password* (password protecting the keystore), and optionally *server.ssl.key-alias* (specific key to use).

Storing SSL certificates within the application offers several advantages. It simplifies the development and deployment process, as the certificates are readily accessible to the application. This approach eliminates the need for external dependencies or sophisticated certificate management systems, making it especially suitable for small-scale or internal applications. Storing SSL certificates within the application also streamlines testing and debugging, as developers can validate SSL configurations directly within the application environment.

However, this method is not without risks. Keystore files stored on the local filesystem can become a single point of vulnerability if not adequately protected. Unauthorized access to the file could compromise private keys, allowing attackers to impersonate the server. Additionally, hardcoding the keystore password in configuration files poses a security risk, especially if the files are included in version control systems.

To mitigate these risks, developers should follow best practices such as using environment variables to inject sensitive configurations, encrypting keystore files, and applying strict file permissions. For production environments, integrating with a

certificate management solution or leveraging tools like Spring Cloud Config for centralized, secure configuration can further enhance security.

Storing SSL certificates within a Spring Boot application, especially in local configurations such as the file system or a keystore, can limit scalability as the application grows or moves to a distributed environment. In scenarios where multiple instances of the application are deployed, each instance requires its own copy of the SSL certificate, necessitating manual synchronization and increasing the complexity of deployment pipelines. This approach also complicates tasks like certificate rotation or updates, as changes must be propagated consistently across all instances, leading to potential downtime or configuration drift. Additionally, storing certificates within the application sometimes ties the application to specific environments, making it less flexible in cloud-native or containerized architectures where scalability depends on stateless and dynamic configurations. For large-scale systems, centralized solutions like cloud-based secret managers are often preferred, as they enable seamless access to SSL certificates across all instances while supporting automated updates and secure management.

While storing SSL certificates within the application is convenient and often sufficient for many applications, understanding its limitations and implementing appropriate safeguards is crucial for maintaining a robust security posture.

VI. USING GOOGLE CLOUD SECRET MANAGER FOR SSL CERTIFICATE MANAGEMENT

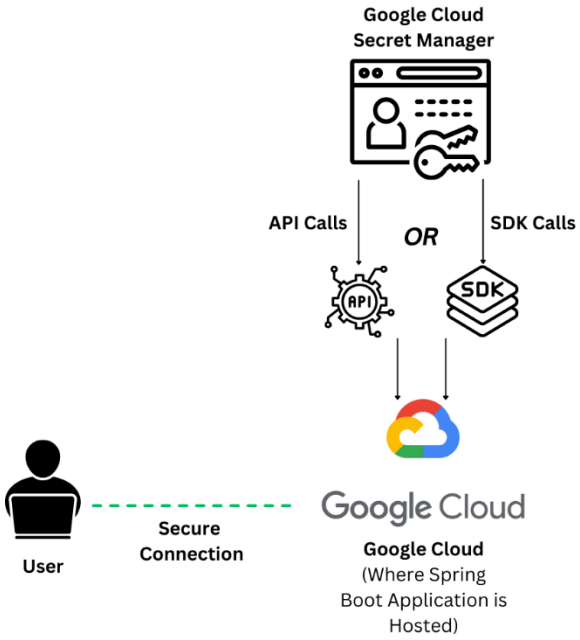

Figure-2: Retrieving SSL Certificates Stored on Google Cloud Secret Manager

Google Cloud Secret Manager is a managed service designed to securely store and access sensitive information, such as API keys, passwords, and SSL certificates. It provides features like automatic encryption at rest using Google-managed keys, fine-grained access control via Identity and Access Management (IAM), and seamless integration with other Google Cloud services.

The process of using Secret Manager to manage SSL certificates in a Spring Boot application begins with storing the SSL certificate (or the keystore file containing it) as a secret. The file can be uploaded to Secret Manager either through the Google Cloud Console or the command-line interface (CLI). Each secret version is automatically encrypted and can be updated without altering the secret name, facilitating smooth updates.

In Spring Boot, accessing the stored certificate involves using the Google Cloud SDK or the Secret Manager API. Developers can retrieve the secret at runtime, typically by integrating the Secret Manager client library with Spring Boot. This eliminates the need to hardcode sensitive data in configuration files. Once retrieved, the certificate or keystore can be loaded into the application context programmatically, allowing SSL configuration to proceed as usual.

Using Secret Manager offers significant security benefits over storing certificates within the application. Secrets are encrypted at rest and in transit, reducing the risk of exposure. Role-based access control ensures that only authorized entities can

retrieve secrets, and all access is logged for audit purposes. Additionally, the centralized nature of Secret Manager simplifies management, especially in environments with multiple applications or instances.

By integrating Google Cloud Secret Manager, Spring Boot applications gain a robust mechanism for handling SSL certificates, enhancing security while reducing operational complexity. This approach is particularly valuable in production environments where compliance and scalability are critical.

VII. COMPARISON OF STORING CERTIFICATES WITHIN THE APPLICATION VS. GOOGLE CLOUD SECRET MANAGER

When deciding between storing certificates within the application or using Google Cloud Secret Manager to handle SSL certificates in a Spring Boot application, developers must weigh the ease of setup, performance, scalability, security, cost, and operational overhead.

A. Ease of Setup and Configuration

Certificates within the application are straightforward to configure, requiring only a keystore file and a few entries in the *application.properties* file. In contrast, Google Cloud Secret Manager involves additional setup, including creating a Google Cloud project, enabling the Secret Manager API, configuring IAM roles, and integrating the Secret Manager client with Spring Boot. While more complex initially, Secret Manager simplifies secret updates and management in the long run.

B. Performance and Scalability

Storing within the application offers the fastest access to SSL certificates since the keystore is stored directly on the application's filesystem. However, it can become cumbersome to manage as the application scales across multiple instances or environments. Secret Manager, though slightly slower due to network calls, excels in scalability by providing a centralized solution accessible to all instances, ensuring consistency and reducing duplication.

C. Security Features

Google Cloud Secret Manager provides superior security features, such as automatic encryption, IAM-based access control, and comprehensive access logging. While storing SSL certificates within the application is simpler, it relies heavily on filesystem permissions and manual security measures, making it more susceptible to breaches if misconfigured.

D. Infrastructure Costs and Operational Overhead

Storing SSL certificates within the application incurs no direct cost but can lead to operational challenges, especially in dynamic or distributed environments. Secret Manager, while incurring a nominal cost for storage and API usage, significantly reduces overhead by centralizing secret management and providing automatic versioning and lifecycle management.

E. Deployment

For Spring Boot applications deployed in the Google Cloud environment, Secret Manager is the ideal choice. Its native integration with other Google Cloud services ensures seamless setup and operation. Applications running in Google Kubernetes Engine (GKE), App Engine, or Compute Engine can securely access secrets without exposing credentials, thanks to built-in identity and access management (IAM) features. For applications hosted outside the Google Cloud environment—on-premises, other cloud providers, or local development environments—Storing certificates locally remains a practical solution. It avoids the complexity of setting up and maintaining connectivity with the Secret Manager.

F. Other Considerations

Storing certificates within the application is better suited for small, single-instance applications or internal tools where simplicity is key. Conversely, Secret Manager is ideal for production-grade, distributed systems where security, compliance, and scalability are priorities.

Ultimately, storing SSL certificates within the application is quick and practical for small-scale applications, whereas Google Cloud Secret Manager is a robust, enterprise-grade solution for secure and scalable certificate management.

VIII. RECOMMENDATION FOR SECURING SSL CERTIFICATES IN THE SPRING BOOT APPLICATION

Based on the analysis, Google Cloud Secret Manager is the most suitable method for securing SSL certificates in Spring Boot applications, especially for cloud-based or production environments. It offers superior security with automatic encryption, fine-grained access control, and centralized management, making it ideal for large-scale, distributed systems that require scalability and ease of maintenance.

- Google Cloud Secret Manager is the best choice for applications deployed on Google Cloud Platform (GCP), such as GKE or App Engine, where integration with other Google services and centralized secret management is a priority.
- Storage within the application is more appropriate for smaller applications, internal tools, or environments with limited scalability, where simplicity is preferred and the application is not hosted in a cloud infrastructure.

Making an Informed Decision: Organizations should assess their infrastructure needs and security requirements. For cloud-native applications, especially those with compliance and security concerns, Google Cloud Secret Manager offers robust benefits. For legacy, smaller, or non-cloud environments, storing certificates within the application may provide a simpler, cost-effective solution. Ultimately, the choice depends on factors like scale, security needs, deployment strategy, and resource management preferences.

IX. SOLUTION: A HYBRID IMPLEMENTATION APPROACH FOR SSL CERTIFICATE STORAGE

For organizations seeking to balance accessibility and security, a hybrid approach to storing SSL certificates combines the benefits of local storage and Google Cloud Secret Manager. This method is particularly suited for applications requiring both low-latency certificate access and secure, centralized management.

Local Storage for Immediate Use: Store frequently accessed SSL certificates locally within the application in encrypted form. Use tools like JKS (Java KeyStore) to ensure strong encryption, and implement strict access controls to minimize the risk of unauthorized access.

Google Cloud Secret Manager for Backup and Rotation: Simultaneously store SSL certificates in Google Cloud Secret Manager for secure, centralized storage. This setup allows for automated rotation and secure retrieval during deployment or when certificate updates are required.

Integrate via Automation: Use CI/CD pipelines to fetch certificates from Secret Manager during build or deployment processes. Employ APIs or SDKs to seamlessly retrieve and update local copies from the cloud when necessary.

By leveraging the strengths of both methods, this hybrid approach ensures robust security without compromising performance, making it ideal for scalable and sensitive Spring Boot applications.

X. CONCLUSION

This article has explored the methods of securing SSL certificates in Spring Boot applications, comparing the storage of certificates within the application and handling them with Google Cloud Secret Manager. While the former offers simplicity, Google Cloud Secret Manager provides superior security, scalability, and centralized management, making it the preferred choice for cloud-based or production environments. Choosing the right method for SSL certificate management is crucial for ensuring secure communications and protecting sensitive data. Organizations must evaluate their infrastructure, security requirements, and scalability needs. Future research should explore enhanced integrations with third-party services and automation tools to streamline SSL certificate management across diverse environments.

XI. REFERENCES

[1] S. Yao, J. Chen, K. He, R. Du, T. Zhu, and X. Chen, "PBCert: Privacy-Preserving Blockchain-Based Certificate Status Validation Toward Mass Storage Management," *IEEE Access*, vol. 7, pp. 6117–6128, 2019, doi: 10.1109/ACCESS.2018.2889898.

[2] J. Vargas, F. Mayorga, D. Guevara, and H. D. Martinez, "Management of SSL Certificates: Through Dynamic Link Libraries," in *Technology Trends*, Springer, Cham, 2019, pp. 29–40. doi: 10.1007/978-3-030-05532-5_3.

[3] B. Li *et al.*, "Certificate Transparency in the Wild: Exploring the Reliability of Monitors," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, in CCS '19. New York, NY, USA: Association for Computing Machinery, Nov. 2019, pp. 2505–2520. doi: 10.1145/3319535.3345653.

[4] J. A. Berkowsky and T. Hayajneh, "Security issues with certificate authorities," in *2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, Oct. 2017, pp. 449–455. doi: 10.1109/UEMCON.2017.8249081.

[5] A. Alabduljabbar, R. Ma, S. Choi, R. Jang, S. Chen, and D. Mohaisen, "Understanding the Security of Free Content Websites by Analyzing their SSL Certificates: A Comparative Study," in *Proceedings of the 1st Workshop on Cybersecurity and Social Sciences*, in CySSS '22. New York, NY, USA: Association for Computing Machinery, May 2022, pp. 19–25. doi: 10.1145/3494108.3522769.

[6] R. Dastres and M. Soori, "Secure Socket Layer (SSL) in the Network and Web Security," *Int. J. Comput. Inf. Eng.*, vol. 14, no. 10, pp. 330–333, Nov. 2020.

[7] J. S. Resende, L. Magalhães, A. Brandão, R. Martins, and L. Antunes, "Towards a Modular On-Premise Approach for Data Sharing," *Sensors*, vol. 21, no. 17, Art. no. 17, Jan. 2021, doi: 10.3390/s21175805.

[8] A. Gupta, P. Goswami, N. Chaudhary, and R. Bansal, "Deploying an Application using Google Cloud Platform," in *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, Mar. 2020, pp. 236–239. doi: 10.1109/ICIMIA48430.2020.9074911.