IBIMA Publishing
Communications of International Proceedings
https://ibimapublishing.com/p-articles/46SP/2025/4638925/
Vol. 2025 (37), Article ID 4638925

From Hardcoded Keys to Centralized Control: Modern Approaches to Secret Management*

Piotr KONTOWICZ

Poznan University of Technology, Faculty of Computing and Telecommunications,
Piotrowo 3, 60-965 Poznan, Poland

Correspondence should be addressed to: Piotr KONTOWICZ, piotr.kontowicz@put.poznan.pl

* Presented at the 46th IBIMA International Conference, 26-27 November 2025, Ronda, Spain

Abstract

The growing complexity of modern IT environments, supported by cloud computing, microservices, and DevSecOps practices, makes secret management an essential component of organizational security. Secrets, including API keys, credentials, and cryptographic keys, enable authentication between human and system entities. Their compromise leads to severe data breaches and operational disruption. Existing literature and standards, such as NIST and ISO 27001, provide guidance for key management, yet there remains a clear gap in integrating technical, procedural, and compliance perspectives into one consistent framework.

This study addresses this gap through an analytical review of modern secret management mechanisms, focusing on software-based secret managers, key management services, and hardware security modules. The analysis examines how each solution contributes to confidentiality, integrity, and availability in distributed systems and evaluates security challenges related to cloud and hybrid environments.

The findings show that effective secret management requires not only the implementation of technology but also the development of comprehensive policies, automation, and awareness programs. Centralization, lifecycle control, and adherence to the principle of least privilege are essential elements that strengthen protection against unauthorized access. Combining technology, processes, and education results in higher organizational resilience and positions secret management as a fundamental pillar of modern cybersecurity.

Keywords: Secret Management, Cryptographic Keys, DevSecOps, Cloud Security, CIA Triad, Key Management, Security Policy

Introduction

Secrets in IT systems are all authentication data that allows access to resources. Examples include certificates, credentials, API keys, passwords, and tokens. Modern systems often rely on cloud infrastructure and microservices, a highly dynamic environment in which credentials are frequently exchanged not by humans but by interacting systems. This data is a valuable target for attackers, as obtaining an API key belonging to an administrator immediately grants privileged access to the system, which is difficult to detect.

Historically, mistakes related to secret storage have also occurred in large organizations. These errors have included the exposure of access keys in code repositories, from which attackers obtained them, leading to data breaches [1].

Cite this Article as: Piotr KONTOWICZ, Vol. 2025 (37) "From Hardcoded Keys to Centralized Control: Modern Approaches to Secret Management " Communications of International Proceedings, Vol. 2025 (37), Article ID 4638925, https://doi.org/10.5171/2025.4638925

Managing Secrets in Modern IT Systems

A secret is any authentication data that permits access to resources. While early systems emphasized human-to-machine interactions, contemporary environments are characterized by system-to-system communication, where credentials are exchanged for mutual authentication.

Different categories of secrets can be distinguished:

1. Certificates: An example includes SSL/TLS certificates [2] used to secure communication. In this case, the secret is the private key associated with the certificate.
2. Credentials: This refers to traditional user passwords as well as other data, such as SSH keys used to securely connect to remote systems.
3. Tokens: These include API keys, OAuth/JWT tokens, and session identifiers.
4. Cryptographic keys: can be divided into two categories: symmetric and asymmetric keys. Symmetric keys are used for encryption and decryption, while in the case of asymmetric keys, the secret is the private key.

The evolution and proliferation of systems have resulted in a growing number of credentials used for mutual authentication between systems without human intervention [3].

Protecting Data in IT Systems

The primary objective is to safeguard the security of data processed within systems. Achieving data security involves the following elements:

1. Confidentiality: Unauthorized access to secrets by attackers leads to unauthorized access to systems and data. Protecting these elements is essential to maintaining the confidentiality of the resources they provide access to.
2. Integrity: Protection against unauthorized modification. For example, losing control of a private key used for software signing can allow the distribution of modified, malicious code.
3. Availability: Losing access to a critical key due to a failure or it being encrypted during a ransomware attack may lead to operational disruption.

Business and financial risks should also be considered. The cost of security breaches can result in significant financial losses, including those related to operational downtime and potential regulatory penalties.

Managing Secrets in Environments

The complexity of the secret management process is increased by changes in software architecture, the growth of microservices architecture, and the adoption of the DevSecOps approach.

Traditionally, applications were built using a monolithic architecture, where secrets could be managed from a single, centralized location. Microservices architecture, while offering scalability, introduces several security challenges:

1. Complex access control: All services require their own permissions to communicate with other services. Managing a growing number of such relationships becomes increasingly complex as the number of services increases.
2. Inconsistency: Different development teams may choose different and often insecure methods of storing secrets. This leads to a lack of a unified security policy across the organization.
3. Attack surface: A larger number of services directly increases the attack surface. The more applications there are, the greater the chance of misconfiguration in one of them.

In distributed architectures, communication must be secured and supported by robust authentication and authorization mechanisms.

The DevSecOps paradigm, which integrates security processes into the software development lifecycle, requires the inclusion of secret management mechanisms in the Continuous Integration/Continuous Deployment (CI/CD) pipeline. Best practices include:

1. Centralized management: Using a dedicated system to manage all credentials.
2. Secure injection: Automatically retrieving secrets during the CI/CD process [4] and injecting them into the application at build or runtime. This eliminates the need to store credentials in the application's source code.
3. Segmentation: Practically implementing the principle of least privilege by separating secrets used in production, testing, and development environments.

Consequently, securing the CI/CD pipeline is a critical component of overall system security.

Compliance and Secrets

An appropriate level of security in secret management is also regulated by legal and compliance frameworks. Recommendations issued by the National Institute of Standards and Technology (NIST) are considered an industry standard. In a document focused on secret security [5] that covers their entire lifecycle, it is stated that symmetric keys and private keys must be protected from unauthorized access, and all keys must be safeguarded against unauthorized modification [6]. Secure storage must be ensured through the use of appropriate methods.

The ISO 27001 standard [7], a leading industry benchmark, requires the implementation of comprehensive control measures. It mandates the protection of authentication data throughout its lifecycle, and the deployment of a centralized secret management system is often considered essential for compliance. The guidelines address underlying causes of breaches, such as insufficient awareness or poor practices. Achieving ISO compliance extends beyond acquiring tools; it necessitates the establishment of effective processes and ongoing training to ensure secure and proper use of implemented solutions.

Common Mistakes in Secret Management

Based on the analysis of security incidents, it can be concluded that many cases involving secret leaks are not always the result of advanced attacks. Secret exposure also occurs due to fundamental errors in processes that arise from haste, lack of awareness, or the absence of proper tools and procedures.

Hardcoding refers to embedding secrets such as passwords or API keys directly into an application's source code. This practice is often used during application development. From a developer's perspective, it is quick and easy to implement. However, a secret placed this way becomes an integral part of the application, meaning that anyone with access to the source code also gains access to the embedded secret. Source code can be obtained, for example, through the decompilation of an application.

Repositories are used in application development to enable collaboration between multiple developers working on the same source code. If a secret is committed to a public repository, it becomes immediately accessible to everyone. Even if the repository is private, the problem is not fully resolved, as there is still a risk of unauthorized access, for example, through a compromised administrator account. Additionally, removing a secret from a repository is problematic. Simply deleting the secret from the code and committing a new version does not solve the issue, since the secret remains in the repository's history.

Secrets are sometimes treated as static elements created once and never rotated. This approach is flawed. Every secret should have a defined usage period. The practice of revoking and replacing secrets after a defined time is called rotation. Without this mechanism, the risk of accidental exposure increases over time. A security-conscious organization should have policies and procedures in place for periodic secret rotation. A defined revocation procedure is also essential.

According to the principle of least privilege, each entity should be granted only the minimum set of permissions necessary to perform its tasks. Assigning overly broad privileges to secrets is a mistake. This is sometimes done for convenience, as a broadly privileged secret is more likely to "just work," which saves time by avoiding additional configuration.

An Overview of Secret Management Tools

In response to the identified problems, specialized tools have been developed. There are various categories of solutions available on the market [8] [9]:

1. Secrets Managers: The goal is to centralize the entire lifecycle of secrets. These tools automate rotation, versioning, and revocation of secrets while ensuring confidentiality through the use of strong cryptography.
2. Key Management Service (KMS): These operate under a different model. Their greatest advantage is that the application delegates cryptographic operations to the service's API and never has direct access to the encryption key, which always remains within the secure service.
3. Hardware Security Module (HSM) / Trusted Platform Module (TPM): These are hardware-based solutions offering the highest level of security. Keys are generated and stored inside a secure environment. Cryptographic operations are also performed within that environment.

Understanding the distinction between secrets managers and KMS is essential. Secrets managers supply credentials, whereas KMS performs critical cryptographic operations on behalf of applications. While the KMS model offers enhanced security, it may not always be feasible to implement.
Cloud service providers (AWS [10], Google [11], Azure [12]) offer native secret managers deeply integrated within their ecosystems. Choosing a management platform is an important decision. Solutions delivered by cloud providers are easy to implement, especially when the organization is already operating within a given provider's ecosystem. An alternative is HashiCorp Vault [13], which is designed to work across multiple environments. Native services offered by AWS, Google, and Azure are intended to be used within the provider's ecosystem. HashiCorp Vault, thanks to its API interface, can be used in private clouds and with other providers.

A risk that must be considered when selecting a secret management mechanism is vendor lock-in. Native cloud services are often the best fit for organizations looking to implement a solution quickly. However, such solutions come with a dependency on the provider and limited portability. When an organization's strategy evolves toward a multicloud or hybrid model, relying on a secrets manager tightly integrated with a specific provider becomes an architectural burden. For this reason, from the perspective of product growth and evolution, it is often better to choose HashiCorp Vault because it supports all environments.

Hardware mechanisms are considered to provide the highest level of protection possible. Examples of hardware-based solutions include:

1. Hardware Security Module (HSM): A tamper-resistant device designed to protect against physical manipulation. Keys are generated inside the device and never leave it. All cryptographic operations are performed internally.
2. Trusted Platform Module (TPM): A specialized microprocessor embedded in the motherboard that allows secure storage of keys and credentials.

The current threat landscape shows that attackers do not break hardware security directly. Instead, they bypass it by targeting the surrounding cloud ecosystem. Analysis indicates that the main attack vectors include the APIs of HSM modules and the virtualization layer of TPM. For cloud-based HSM, the biggest threat is misuse of APIs. Attackers who gain access to valid credentials can impersonate legitimate services and interact with the HSM. Rather than stealing the key directly, they remotely use it for malicious operations. Another attack vector involves misconfigured roles and permissions in the cloud, which allow attackers to gain administrative access to modules and bypass protections.

Trust in vTPM is rooted in the hypervisor, which means the threat model is different. Attacks on the hypervisor or its control mechanisms can lead to a breakdown in trust in the vTPM. There is also a risk that multi-tenant instances (used by multiple clients) could expose keys to another client operating on the same physical infrastructure.

In practice, the level of security offered by hardware solutions depends on the protection of APIs and the effectiveness of identity management mechanisms.

Building a Comprehensive Secret Management Policy

The use of modern technologies for secret management is an important step toward securing them. However, it is not sufficient on its own. Technology must be supported by proper processes and a culture of security awareness. The foundation of secret protection should be a formal secret management policy that covers the entire lifecycle of a secret, from creation to destruction. The secret management policy should also include a response procedure in the event that a secret is compromised.

In large, particularly distributed environments, manual secret management within applications is impractical and must be supplemented by appropriate technologies. However, technology and procedures alone are insufficient. Organizations should also provide regular training for developers, administrators, and security teams to prevent common errors such as hardcoding secrets or storing them in code repositories.

Acknowledgements

This research was funded by the Polish Ministry of Science and Higher Education under Grant 0313/SBAD/1311.

Conclusions

In the modern world, secret management is a key element of system security. The increasing complexity of systems, the shift to microservices architecture, and the integration of DevSecOps processes result in a growing number of credentials used within organizations. Ensuring security requires not only the implementation of appropriate technology but also the development of processes and employee awareness.

Effective secret management requires centralization, automation, and adherence to the principle of least privilege. A crucial part of secret security is the use of dedicated solutions such as secret managers, key management systems, and hardware modules that help eliminate the risk of unauthorized access.

A clearly defined secret management policy covering the full lifecycle of secrets is a vital component of a comprehensive protection system. Given that many incidents result from human error, regular training for development and administration teams is equally critical. Organizations that implement these measures achieve greater resilience to attacks. Security should be regarded as an ongoing process rather than a one-time action.

Bibliography

- S. K. Basak, L. Neil, B. Reaves, and L. Williams, “What Challenges Do Developers Face About Checked-in Secrets in Software Artifacts?,” Jan. 29, 2023, *arXiv*: arXiv:2301.12377. doi: 10.48550/arXiv.2301.12377.
- Nawazpasha Shaik, “Automated TLS certificate lifecycle management: A policy-driven framework for kubernetes security hardening,” *Global Journal of Engineering and Technology Advances*, vol. 23, no. 1, pp. 250–257, Apr. 2025, doi: 10.30574/gjeta.2025.23.1.0110.
- Rameshreddy Katkuri, “Security in cloud-native microservices: The critical foundation,” *World Journal of Advanced Engineering Technology and Sciences*, vol. 15, no. 1, pp. 1255–1271, Apr. 2025, doi: 10.30574/wjaets.2025.15.1.0321.
- Sarathe Krisshnan Jutoo Vijayaraghavan, “Security as code: Transforming DevSecOps through CI/CD Integration,” *World J. Adv. Eng. Technol. Sci.*, vol. 15, no. 1, pp. 2219–2225, Apr. 2025, doi: 10.30574/wjaets.2025.15.1.0446.
- Barker, Elaine, and William Barker. *Recommendation for key management, part 2: best practices for key management organization*. No. NIST Special Publication (SP) 800-57 Part 2 Rev. 1. National Institute of Standards and Technology, 2018.
- Barker, Elaine, and William Barker. *Recommendation for key management, part 2: best practices for key management organization*. No. NIST Special Publication (SP) 800-57 Part 2 Rev. 1 Page 49. National Institute of Standards and Technology, 2018.
- ISO/IEC 27001:2022, Information security, cybersecurity and privacy protection — Information security management systems — Requirements, Edition 3” ISO Accessed: Nov. 01, 2025. [Online]. Available: https://www.iso.org/standard/27001

- “Secrets Management - OWASP Cheat Sheet Series.” Accessed: Nov. 01, 2025. [Online]. Available: https://cheatsheetseries.owasp.org/cheatsheets/Secrets_Management_Cheat_Sheet.html
- Kuzminykh, I., Ghita, B., Shiaeles, S. (2020). Comparative Analysis of Cryptographic Key Management Systems. In: Galinina, O., Andreev, S., Balandin, S., Koucheryavy, Y. (eds) Internet of Things, Smart Spaces, and Next Generation Networks and Systems. NEW2AN ruSMART 2020 2020. Lecture Notes in Computer Science(), vol 12526. Springer, Cham. https://doi.org/10.1007/978-3-030-65729-1_8
- “AWS Key Management Service - AWS Key Management Service.” Accessed: Nov. 01, 2025. [Online]. Available: https://docs.aws.amazon.com/kms/latest/developerguide/overview.html
- “Cloud Key Management,” Google Cloud. Accessed: Nov. 01, 2025. [Online]. Available: https://cloud.google.com/security/products/security-key-management
- “Key Vault | Microsoft Azure.” Accessed: Nov. 01, 2025. [Online]. Available: https://azure.microsoft.com/en-us/products/key-vault
- HashiCorp, “HashiCorp Vault | Identity-based secrets management,” HashiCorp | An IBM Company. Accessed: Nov. 01, 2025. [Online]. Available: https://www.hashicorp.com/en/products/vault