Universal centralized secret data management for automated public cloud provisioning

Yevhenii Martseniuk[1,†], Andrii Partyka[1,†], Oleh Harasymchuk[1,†] and Svitlana Shevchenko[2,*,†]

[1] Lviv Polytechnic National University, 12 Stepana Bandery str., 79013 Lviv, Ukraine
[2] Borys Grinchenko Kyiv Metropolitan University, 18/2 Bulvarno-Kudryavska str., 04053 Kyiv, Ukraine

Abstract

In modern cloud environments, secret management plays a key role in ensuring the security of sensitive data, such as passwords, API keys, credentials, and other critical resources. This paper discusses the use of HashiCorp Vault as a universal platform for centralized secret management and automated provisioning of cloud resources. A comparison is also made with native secret management services, such as AWS KMS, Azure Key Vault, and Google Cloud KMS, to determine their capabilities and limitations in providing security. The comparison shows that Vault offers more flexible and universal secret management thanks to advanced cryptographic methods and integration with automation platforms. The research demonstrates that Vault provides secure storage, dynamic creation, and automatic revocation of credentials, allowing access management based on security policies. The integration of HashiCorp Vault with automation platforms like Rundeck and Ansible enables the automation of cloud resource provisioning while maintaining information confidentiality and reducing the risk of human error. The use of dynamic creation methods for temporary credentials enhances security and compliance with standards, adhering to the principle of least privilege. The results highlight the importance of using HashiCorp Vault as a central platform for managing secrets and credentials, which improves the overall level of security and efficiency in cloud environments.

Keywords

HashiCorp Vault, secrets, automation, data security, dynamic credentials, AWS, authentication, authorization, cloud infrastructure, centralized management

1. Introduction

In the modern world of cloud computing, infrastructure is becoming increasingly ephemeral (temporary) and elastic, adapting to changing loads and needs. The dynamic nature of IP addresses and the lack of a clear network perimeter introduce new challenges in ensuring cybersecurity, as traditional protection methods based on fixed network boundaries become less effective. Consequently, modern security systems are oriented towards the 'Zero Trust' principle, which assumes potential breaches within the network, regardless of its boundaries [1, 2].

The Zero Trust concept posits that no system or user can be trusted by default; every access request must be thoroughly verified, whether it originates from within the organization's internal network or externally [3]. This approach requires more integrated security methods, where access to systems and endpoints is controlled directly, rather than relying on being within a privileged network. This means that instead of using IP addresses as the sole unit of access, each application or service is given a unique identification, allowing it to work with the ephemeral and elastic nature of cloud infrastructure [4].

In the context of Zero Trust, effective secret management is critically important for securing access to applications, systems, and endpoints. Data must be properly protected and not stored in plaintext, as this poses a significant risk of unauthorized access [5]. This necessitates centralized secret management and the use of intermediary software for key management and data encryption [6].

A secret is considered any information that requires strict access restrictions, such as API encryption keys, passwords, certificates, or other credentials used for authentication and authorization of access to resources [1]. In modern conditions, most web applications and various services have already transitioned to a microservices architecture or are actively undergoing this transition. This is because microservices architecture enhances flexibility, scalability, and independence in the development and deployment of individual system components [3]. However, instead of a single monolithic configuration file, there are now numerous small configuration files (one or several per microservice), which require secure storage of their contents.

Additionally, environment variables are often used for configuring service parameters instead of traditional configuration files. As a result, each service has its unique

CPITS-II 2024: Workshop on Cybersecurity Providing in Information and Telecommunication Systems II, October 26, 2024, Kyiv, Ukraine
*Corresponding author.
†These authors contributed equally.
yevhenii.v.martseniuk@lpnu.ua (Y. Martseniuk);
andrijp14@gmail.com (A. Partyka);
garasymchuk@ukr.net (O. Harasymchuk);
s.shevchenko@kubg.edu.ua (S. Shevchenko)

0009-0009-2289-0968 (Y. Martseniuk);
0000-0003-3037-8373 (A. Partyka);
0000-0002-8742-8872 (O. Harasymchuk);
0000-0002-9736-8623 (S. Shevchenko)

settings for connecting to databases, external APIs, message queues, caches, and other systems [4]. Moreover, other parameters require secure storage, such as 'salt' (a modifier used for password hashing) or keys for generating JWT tokens. Thus, there is a significant number of entities that require secure storage to prevent unauthorized access [6].

This research aims to identify the optimal tool for centralized secret management in cloud environments that meets modern security requirements and ensures flexibility and efficiency. The study includes an analysis of existing native secret management services, such as AWS KMS, Azure Key Vault, and Google Cloud KMS, to evaluate their capabilities and limitations [1]. Additionally, the choice of HashiCorp Vault as the best solution for secret management is justified due to its ability to provide advanced cryptographic methods, dynamic credential creation, integration with various automation platforms, and support for a universal approach to secret management in heterogeneous cloud infrastructures [7].

2. Challenges and risks in public clouds and native services for their mitigation

The rapid adoption of cloud computing has fundamentally changed approaches to data storage and management. Due to the high convenience, scalability, and flexibility offered by public cloud services such as Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP), organizations are increasingly moving to cloud technologies for deploying their operations and storing data. However, these changes come with significant challenges related to ensuring the security and privacy of information [8]. Reliable secret storage solutions that include secure management of keys, passwords, and other sensitive data are critically important for protection against unauthorized access and potential data breaches.

By moving their data and applications to public cloud infrastructures, organizations face some security challenges inherent to cloud environments. Cloud environments, by nature, involve storing data on remote servers managed by third parties. This setup creates potential vulnerabilities, as sensitive information, such as encryption keys, passwords, and certificates, must be securely managed and protected from unauthorized access [9]. The lack of effective secret management solutions significantly increases the risks of data leaks, unauthorized access, and insider threats [10]. Therefore, the implementation of advanced cryptographic methods for protecting secrets in the cloud and ensuring data security is especially important [11].

2.1. Main security challenges and risks in public clouds

1. **Vulnerability to Unauthorized Access:** The use of cloud services requires storing keys, passwords, and certificates in a secure environment. However, cloud environments themselves can be targets of attacks, making effective secret management critically important to prevent unauthorized access. Even if data is intercepted, advanced cryptographic methods, such as encryption using robust algorithms, ensure that without the appropriate decryption keys, the data remains inaccessible [12].
2. **Insider Threats:** Insider threats pose a significant risk to cloud environments. Employees or contractors with access to sensitive data may intentionally or unintentionally compromise security. Threshold cryptography methods and distributed key management help mitigate this risk by ensuring that no individual has complete access to the secret key, thus limiting the potential for malicious use of the data.
3. **Ensuring Data Integrity and Confidentiality:** Data integrity and confidentiality are the most important aspects of security in cloud environments. Advanced encryption methods, such as DNA-based encryption or hybrid cryptography, provide robust mechanisms to protect against tampering, and unauthorized access and ensure that only authorized users can decrypt the data.
4. **Scalability and Performance:** Scalability is a key advantage of cloud computing, but it also creates security challenges. Decentralized encryption and other scalable cryptographic solutions enable cloud environments to handle large volumes of data efficiently without compromising security. They provide effective key management, prevent data duplication, and enhance the overall performance of systems.
5. **Regulatory Compliance and Building Trust:** Implementing reliable secret storage solutions helps organizations not only protect their data but also build trust with their clients and ensure compliance with regulatory requirements, such as GDPR and HIPAA. This is achieved by demonstrating a commitment to data security and using advanced protection methods [13].

3. Native services for secure secret management

To address these issues, public cloud providers like AWS, Azure, and GCP offer their own built-in secret management services:

3.1. AWS key management service

AWS Key Management Service (KMS) provides centralized control over cryptographic keys used to protect data stored within AWS infrastructure. This service is closely integrated with other AWS cloud services, allowing automatic encryption of data within those services and managing access to the keys needed for decryption. Integration with AWS CloudTrail enables auditing of key operations, providing detailed information about who used specific keys, on which resources, and when.

AWS KMS also simplifies the process for developers to add encryption or digital signing capabilities to their software, either directly or through the use of AWS SDK, which supports AWS Encryption SDK as the key provider for encrypting and decrypting data locally in applications.

The service allows the effective management of the lifecycle and permissions of keys, including the ability to create new keys at any time and manage their permissions separately from the rights to use them [14]. Users can choose between keys generated in AWS KMS or other options, such as importing keys from their key management infrastructure, using keys stored in an AWS CloudHSM cluster, or keys from an external key manager outside of AWS. AWS KMS also supports the automatic rotation of root keys once a year without the need to re-encrypt previously encrypted data, ensuring their long-term security. The service retains old versions of keys, making them available for decrypting previously encrypted data. Key management is performed through the AWS console, AWS SDK, or AWS Command Line Interface (CLI).

Security and Compliance: AWS KMS is designed so that even AWS employees do not have access to your plaintext keys. This is achieved through the use of Hardware Security Modules (HSMs) that are validated against the Federal Information Processing Standards (FIPS) 140-2 of the U.S. National Institute of Standards and Technology (NIST). The FIPS 140-2 Cryptographic Module Validation Program ensures that HSMs provide robust protection for the confidentiality and integrity of keys.

The HSMs used in AWS KMS serve as the cryptographic root of trust and create a secure hardware environment for performing all cryptographic operations within KMS. All key material for KMS keys is generated in AWS KMS HSMs, and all operations requiring access to plaintext keys are strictly performed within the HSMs, in compliance with FIPS 140-2 Level 3 security requirements [15].

Updates to the HSM firmware in AWS KMS are controlled by multi-party access management and are reviewed by independent expert groups at Amazon. All firmware changes are sent to an accredited NIST laboratory for verification of compliance with FIPS 140-2 Level 3 security standards. Your plaintext keys are never written to disk and are only used in the volatile memory of the HSM for the duration required to perform the requested cryptographic operation. This applies to cases where keys are created on user request, imported into the service, or created in an AWS CloudHSM cluster using a dedicated key storage function.

Regulatory Compliance and Key Geographical Control: AWS KMS allows users to choose whether to create keys restricted to a single region or keys that can be used across multiple regions. This is crucial for meeting regulatory requirements regarding the storage and processing of data within specific geographical boundaries. Keys created for a single region are never transferred outside of that region, ensuring data control within the defined jurisdiction.

The benefits of AWS KMS include flexibility in choosing encryption methods, integration with other AWS services, a high level of security through the use of certified HSMs, and the ability to comply with various regulatory standards, such as GDPR or HIPAA. This makes AWS KMS a powerful tool for ensuring data security and privacy in cloud environments [16].

3.2. Azure key vault: A native service for secure secrets management

Azure Key Vault is a cloud service from Microsoft designed for the secure storage of secrets and access management. Secrets are considered data that require strict access control, such as API keys, passwords, and cryptographic keys. Azure Key Vault supports two types of containers: general vaults and managed Hardware Security Module (HSM) pools. Vaults are used for storing software keys, secrets, and certificates with support for Hardware Security Modules (HSMs), while HSM pools are exclusively designed for storing keys protected by HSM hardware [17].

To ensure secure data transmission between Azure Key Vault and clients, the service uses the Transport Layer Security (TLS) protocol. TLS ensures reliable authentication, message confidentiality, data integrity, and detection of unauthorized alterations, interceptions, or message forgeries. Perfect Forward Secrecy (PFS) further secures connections between clients and Microsoft's cloud services by using unique keys, including 2048-bit RSA encryption keys. This configuration significantly complicates the interception and access to data during its transmission.

Access to Azure Key Vault is managed through two interaction planes: the management plane and the data plane.

- **Management Plane:** This plane is used for administering Key Vault, including creating and deleting key vaults, retrieving their properties, and configuring access policies.
- **Data Plane:** This plane is intended for working with the data stored in the vaults. It allows adding, deleting, and modifying keys, secrets, and certificates.

Authentication in both planes is handled using the **Microsoft Entra** ID. The management plane employs Azure Role-Based Access Control (Azure RBAC), while the data plane uses Key Vault access policies alongside Azure RBAC to manage operations within the vault.

Access to either plane of Azure Key Vault requires proper authentication and authorization of all calling entities (users or applications). The authentication process identifies the requesting party, while authorization determines the permissible operations for that entity. Azure Key Vault uses Microsoft Entra ID to authenticate any security principal that needs access to Azure resources [18].

3.3. Security principles and authentication mechanisms

A security principal in Azure is an entity that represents a user, group, service, or application that requires access to resources. Each security principal in Azure is assigned a unique object identifier. Types of security principals include:

- **User Security Principal:** Represents an individual user with a profile in Microsoft Entra ID.
- **Group Security Principal:** Represents a group of users created in Microsoft Entra ID. Any roles or

permissions assigned to the group automatically apply to all its members.

- **Service Principal:** Used to identify an application or service (a piece of code), rather than a user or group. The object identifier for a service principal is called a client ID and serves a similar function to a username. The client's secret or certificate for a service principal acts as the password.

Many Azure services support the assignment of a Managed Identity, which provides automatic management of the client ID and certificate. This is the most secure and recommended method of authentication in Azure, as it reduces the risks associated with manual credential management [19].

Azure Key Vault is a key security component in the Microsoft Azure ecosystem, providing reliable management and storage of secrets. Its functionality enables organizations to control access to critical data using advanced authentication, authorization, and encryption technologies that meet the highest security standards.

3.4. Google cloud key management service

Google Cloud KMS (Key Management Service) is a cloud service for centralized encryption key management provided within the Google Cloud ecosystem. It allows businesses to create, use, and manage cryptographic keys, as well as securely perform cryptographic operations for other Google cloud services. The primary purpose of Google Cloud KMS is to enable organizations to control the process of data encryption and key management within the cloud infrastructure [20].

By default, all data stored in Google Cloud is automatically encrypted. Using Google Cloud KMS, users gain greater control over how their data is encrypted at rest and how encryption keys are managed. This service provides a high level of security and scalability, meeting the needs of various industries and types of software. Google Cloud KMS allows you to create and use keys for encrypting data in cloud services and applications, ensuring their protection both during storage and transmission.

The Google Cloud KMS platform offers centralized management of cryptographic keys for both direct use and integration with other cloud resources and applications. It supports various key sources for encryption:

- **Cloud KMS Software Keys:** Allow flexible data encryption using manageable symmetric or asymmetric keys.
- **Cloud Hardware Security Modules (HSMs):** Used for secure storage and processing keys in environments with high security requirements.
- **Customer-Managed Encryption Keys (CMEK):** Provide the option to select and use keys generated by Cloud KMS for other Google Cloud services.
- **Cloud External Key Manager (EKM):** Allows the use of external keys that are stored outside of Google Cloud.
- **Customer-Supplied Encryption Keys (CSEK):** Customers can use their encryption keys, maintaining full control over these keys [21].

3.5. Data protection in Google Cloud

Google Cloud automatically encrypts all data using an internal mechanism called **Keystore**. This is a key management service that automatically generates and manages keys without user intervention. Keystore supports a primary key for encrypting new data encryption keys, as well as a limited number of older key versions to maintain compatibility and allow decryption of existing data. Users do not have direct control over these keys or access to their usage logs.

Data from multiple clients may be encrypted with the same default key. This process relies on cryptographic modules validated for compliance with **FIPS 140-2 Level 1,** ensuring a high level of data protection [22].

Google Cloud KMS provides a comprehensive set of tools for managing encryption keys in the cloud environment, allowing organizations to protect data according to modern security requirements. The use of different types of keys, integration with other cloud services, and support for encryption standards enable clients to flexibly adapt the level of protection to their needs and to ensure compliance with regulatory requirements.

Considering the aforementioned risks, it can be concluded that secret management platforms such as **AWS Key Management Service (KMS), Azure Key Vault, and Google Cloud Key Management Service (KMS)** are key tools for overcoming modern security challenges in cloud environments. They provide integrated, scalable, and reliable solutions for managing cryptographic keys and other sensitive data, helping organizations minimize the risks associated with storing secrets in the cloud [23].

However, even with these tools, organizations remain vulnerable to the risks mentioned above:

1. **Vulnerability to Unauthorized Access:** All three platforms store critical keys and secrets in the cloud, which potentially exposes them to unauthorized access, including data breaches or cyber-attacks. AWS KMS, Azure Key Vault, and Google Cloud KMS implement appropriate security measures, such as using Hardware Security Modules (HSMs) and advanced encryption methods, but the risk persists, especially if authentication and access management processes are not properly configured.
2. **Internal Threats:** Insider threats, such as unauthorized actions by employees or contractors, pose a significant risk to any cloud infrastructure. All three platforms use various methods to mitigate these risks: AWS KMS supports threshold cryptography to distribute access to keys, Azure Key Vault implements role-based access control (Azure RBAC), and Google Cloud KMS uses centralized management and auditing capabilities to prevent unauthorized access.
3. **Ensuring Data Integrity and Confidentiality:** Cloud data storage requires continuous assurance of data integrity and confidentiality. While AWS KMS, Azure Key Vault, and Google Cloud KMS use advanced encryption methods and access control

mechanisms, storing data on remote servers creates a risk of data tampering or unauthorized modification.

4. **Scalability and Performance:** The scalability of cloud environments also presents certain security challenges. The use of decentralized encryption solutions, as in the case of Google Cloud KMS, can efficiently handle large volumes of data but may require additional resources to maintain security. Azure Key Vault and AWS KMS offer scaling solutions, but the risk of reduced performance when processing large amounts of data should be considered.
5. **Compliance and Building Trust:** Implementing reliable secret storage solutions is also associated with the risk of non-compliance with regulatory requirements, such as GDPR or HIPAA, which can lead to fines and reputational damage. All three platforms offer tools to ensure compliance with standards, but their effective use requires organizations to carefully configure security policies and maintain constant monitoring [24].

Thus, **AWS KMS, Azure Key Vault, and Google Cloud KMS** offer effective tools for secret management and data protection in cloud environments but require proper configuration and continuous monitoring to minimize security risks. Implementing these solutions allows organizations to reduce the risk of data breaches, prevent unauthorized access, ensure data integrity and confidentiality, and comply with regulatory requirements. However, the success of these measures depends on the level of integration with existing cloud services and the specific needs of the business.

4. Universal secret management platform

In modern cloud environments, the need for effective and reliable secret management is becoming increasingly important. Advanced cryptographic solutions, such as state-of-the-art algorithms, threshold cryptography, hybrid automated security systems, server-based decentralized encryption, DNA-based encryption, and hybrid cryptography, provide robust mechanisms for protecting secrets in the cloud. However, to achieve effectiveness, it is essential to have a universal secret management platform capable of integrating these solutions and providing flexible, scalable, and centralized management of keys and other sensitive data [25].

HashiCorp Vault is an example of such a secret management and encryption system based on identity. Vault provides encryption services that ensure secure access to secrets using authentication and authorization methods to verify and restrict access. This tool allows the protection, storage, and management of secrets and other sensitive data through various interfaces, such as a UI, CLI, or HTTP API. A secret is any information that requires strict access control, such as tokens, API keys, passwords, encryption keys, or certificates. Vault provides a unified interface for managing any secrets, offering strict access control and detailed audit logging [26].

This is particularly important in cases where API keys for external services or credentials for interacting in service-oriented architectures are used extensively and across various environments, complicating the understanding of who has access to which secrets (Fig. 1).

1. **Authentication** — The process by which a client provides information that allows Vault to determine if the client is who it claims to be. After successful authentication, a token is generated that is associated with a specific policy.
2. **Verification** — Vault verifies the client using external trusted sources, such as GitHub, LDAP, AppRole, etc., which provides an additional layer of security.
3. **Authorization** — The process of mapping the client to a security policy defined in Vault. A policy consists of a set of rules that determine which API endpoints the client has access to with their token. Policies provide a declarative way to grant or restrict access to specific resources in the vault.
4. **Access** — After successful authentication and authorization, Vault grants the client access to secrets, keys, and encryption capabilities based on the policies associated with the client's identifier. The client can use the issued token to perform future operations [27].

Thus, **HashiCorp Vault** demonstrates an example of a universal secret management platform that integrates modern cryptographic methods and provides a flexible and scalable approach to data security. Such a platform allows the centralized management of secrets and ensures compatibility with various cloud environments and services, providing robust access control and auditing capabilities. This makes it a crucial tool for organizations looking to enhance their data security and meet modern information protection requirements.

4.1. Challenges and risks associated with secret management and the vault's role in addressing them

Modern enterprises face numerous challenges and risks related to secret management, such as credentials, API keys, passwords, and other confidential data.

Diversity and Lack of Centralized Control: Most organizations have credentials scattered across their environments: in plaintext within source code, configuration files, or even in emails. This creates significant challenges in tracking and controlling access, increasing the risk of unauthorized data use. The dispersion of secrets complicates understanding who has access to which resources and reduces organizations' ability to respond promptly to security incidents.

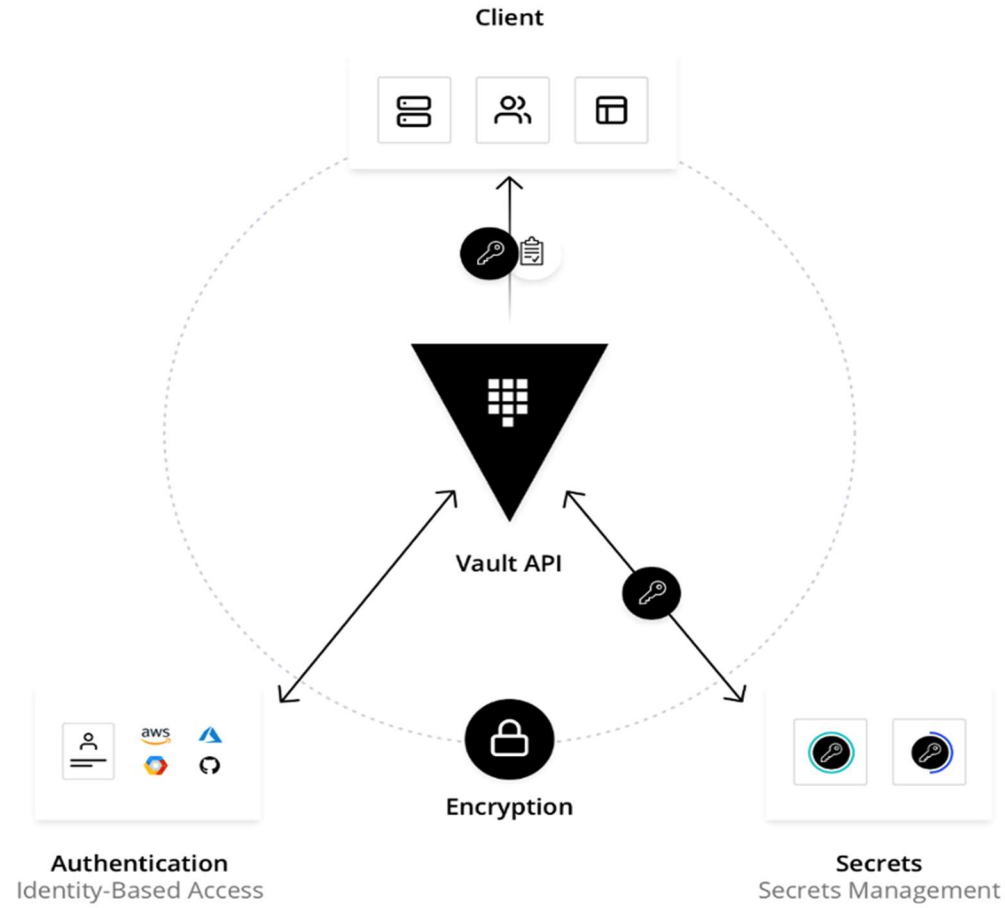

Figure 1: Universal Secret Management Platform

Increased Risk of Malicious Attacks: Storing credentials in plaintext or vulnerable formats increases the likelihood of their compromise by both internal and external attackers. In the event of a data breach or misuse of credentials, the consequences can be catastrophic, including data loss, leakage of confidential information, and significant financial losses.

Inability to Manage the Secret Lifecycle: Without a specialized solution, organizations often struggle to effectively manage the lifecycle of secrets, including generation, storage, updating, and revocation of credentials. This leads to the accumulation of outdated or compromised data, increasing the organization's vulnerability to attacks [28].

HashiCorp Vault was designed to address these issues by providing a universal secret management platform that centralizes credentials, ensures their protection, and offers effective management tools. The core features of Vault aim to minimize risks associated with secret management as follows:

- **Secure Secret Storage:** Vault allows the storage of any secret keys and values by encrypting them before writing them to persistent storage. This ensures that even if unauthorized access to the raw storage occurs, the secrets remain protected as they cannot be read without appropriate access rights.
- **Dynamic Secrets:** Vault can generate secrets on demand for specific systems, such as databases or cloud services. For instance, if an application needs access to an S3 bucket, Vault creates an AWS key pair with valid permissions and automatically revokes them after the lease period expires, significantly reducing the risk of compromise.
- **Data Encryption:** Vault provides the capability to encrypt and decrypt data without storing it, allowing security teams to control encryption parameters while developers can store encrypted data in secure locations, such as SQL databases, without the need to create their encryption methods [29].
- **Lease and Renewal of Secrets:** All secrets in Vault are associated with leases. After the lease period expires, the secret is automatically revoked, helping to avoid the storage of outdated or insecure credentials. Clients have the option to extend the lease using built-in APIs for renewal.
- **Revocation of Secrets:** Vault has built-in support for revoking secrets, allowing the immediate invalidation of access to specific credentials or all secrets associated with a particular user or data type. This feature helps to quickly respond to security incidents and prevents the further use of compromised data [30].

Thus, **HashiCorp Vault** effectively addresses the challenges of dispersion, reduced management transparency, and increased attack risk by providing centralized secret management, robust encryption, dynamic creation and revocation of credentials, and detailed audit logs. This makes it an essential tool for enterprises looking to enhance the security of their confidential data and protect it from modern cyber threats.

4.2. Using HashiCorp Vault for automating cloud environments

In the context of automating cloud resource provisioning, HashiCorp Vault serves as a centralized platform for secret and credential management, ensuring a secure and efficient environment setup. Vault acts as a secret management cluster, critical for automation tasks carried out with Rundeck — a platform that integrates secrets into its tasks, primarily through Ansible instructions [31].

Access to secrets in Vault is achieved through the **AppRole** authentication method, which supports machine-to-machine authentication, providing secure access to secrets. This method allows the creation of roles with specific policies that regulate access to secrets. Each role is associated with a **RoleID** and **SecretID**, used for authentication, ensuring that only authorized services, such as Rundeck, have access to the needed secrets (Fig. 2).

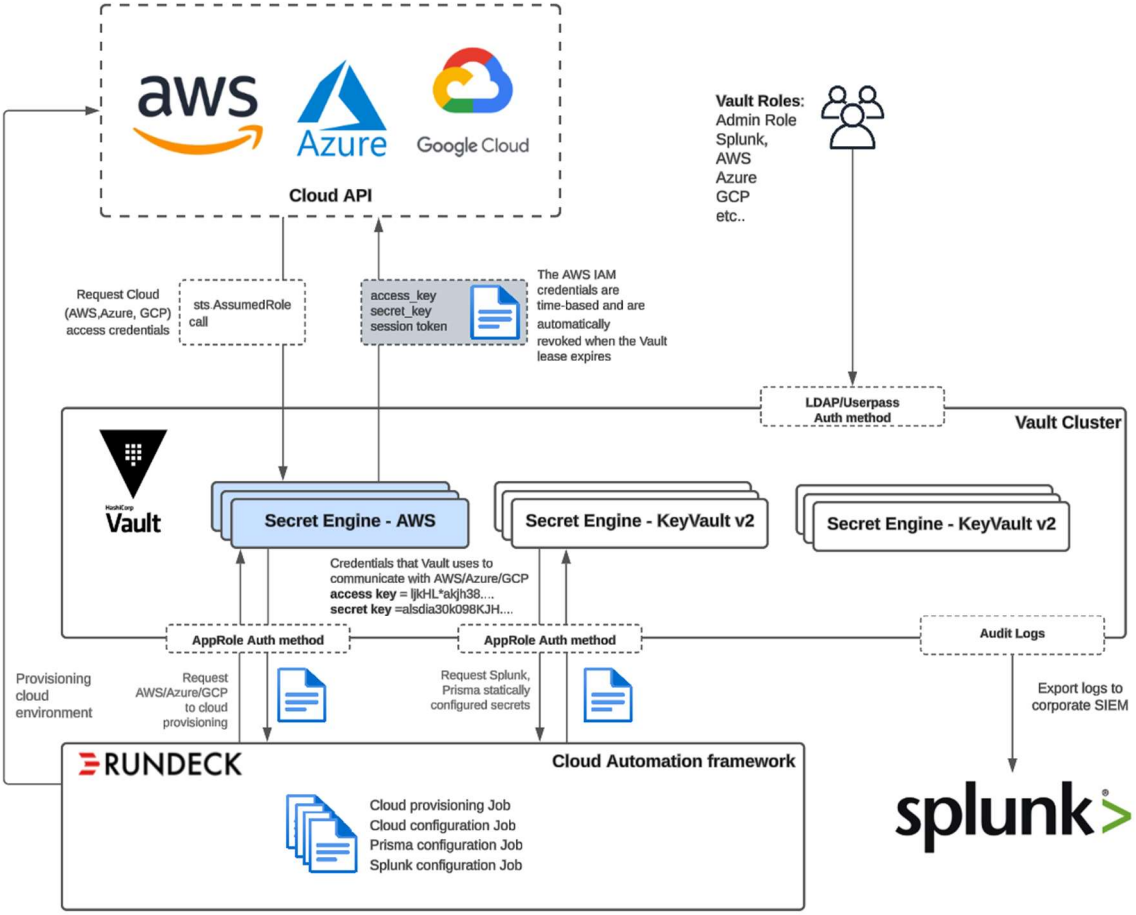

Figure 2: Cloud Environments Management Automation

Most secret functions in this architecture are based on the **HashiCorp Vault Key/Value (KV) v2** mechanism, which provides advanced capabilities for storing and managing data. KV v2 supports secret version history, allowing the secure storage of multiple versions of a single secret and the ability to revert to previous versions in case of accidental deletion or unwanted changes. Additionally, this mechanism includes metadata for each secret, aiding in better management and lifecycle control of the stored data, ensuring detailed access control, and audit logging necessary for maintaining data integrity and confidentiality [32].

In practice, when a task is launched on **Rundeck**, it uses the provided **RoleID** and **SecretID** to authenticate with Vault and obtain the necessary secrets. These secrets are then used in **Ansible** playbooks, which are part of the task, ensuring secure automation of various operations without exposing confidential information in task scripts. This setup not only provides centralized and secure secret management but also automates credential handling, reducing the risk of human error and enhancing the efficiency of automation workflows [33].

In addition to static secrets, this system utilizes **AWS dynamic secret management in HashiCorp Vault,** allowing the automatic creation of temporary AWS credentials (Fig. 3). This is particularly useful for tasks that require modifications to the AWS cloud environment, enhancing security and compliance. The AWS mechanism

in Vault is configured to generate credentials with a limited lifespan based on predefined roles, which determines the access level and permissions required to perform specific automation tasks. This ensures that credentials can only perform specific actions in AWS, minimizing the risk of excessive permissions and improving security by adhering to the principle of least privilege [34].

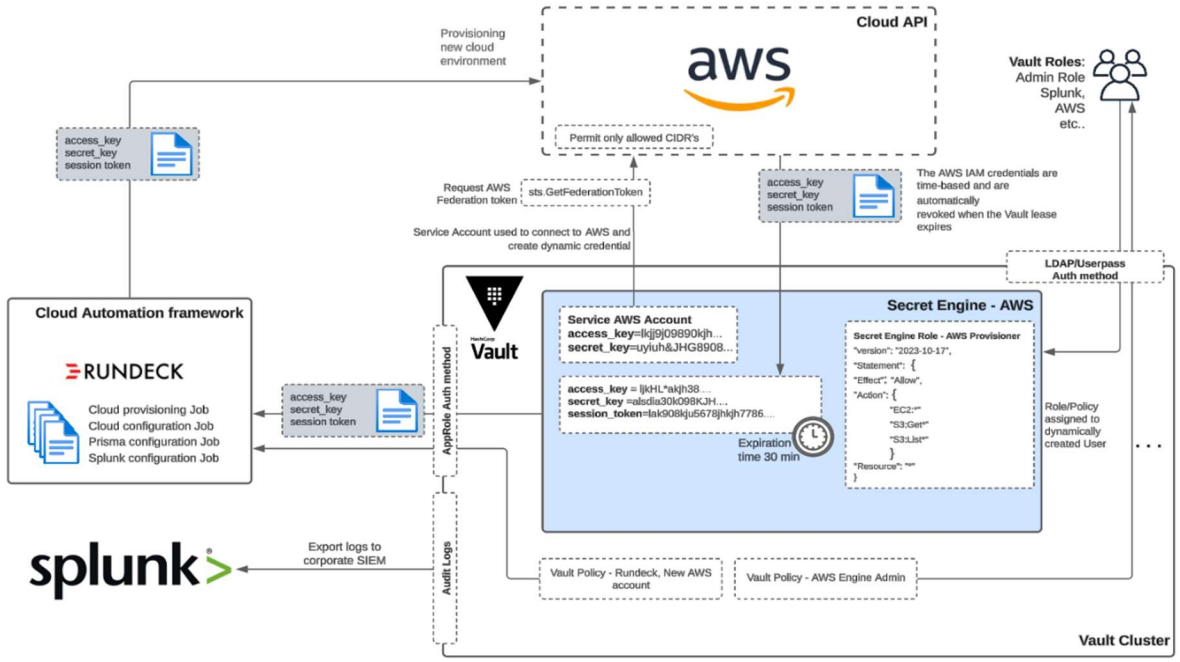

Figure 3: Dynamic Secret Management Mechanism: AWS Example

When a task is initiated in **Rundeck**, it requests temporary credentials from Vault. After authenticating the request using AppRole, Vault issues these credentials, which are then used by Ansible playbooks for secure interaction with AWS services. Once the task is completed or the set time expires, these credentials are automatically revoked, significantly reducing the risk of unauthorized access and misuse of resources. This dynamic credential management approach eliminates the need for long-term AWS keys, simplifies the credential rotation and auditing processes, and greatly enhances security and efficiency in cloud environments [35].

Thus, using **HashiCorp Vault** for automating cloud resource management not only centralizes and secures credentials but also significantly improves security by automating authentication and access management processes. This reduces the risk of human error and enhances the effectiveness of automation tasks in cloud environments [35].

5. Conclusions

In today's era of increasing cloud technology usage, managing secrets and ensuring data security have become critical tasks for organizations. Given the numerous challenges, such as the fragmentation of credentials, rising internal and external threats, the complexity of managing the lifecycle of secrets, and the need to comply with regulatory requirements, employing a universal secret management platform is essential.

The research indicates that **HashiCorp Vault** is an effective solution for centralized secret management in cloud environments. Vault provides secure storage, encryption, dynamic creation, and automatic revocation of credentials, offering policy-based access management. Integration of Vault with automation platforms like **Rundeck** and **Ansible** allows the automation of cloud resource provisioning while maintaining information confidentiality and reducing the risk of human error. The use of dynamic creation of temporary credentials enhances security and compliance, adhering to the principle of least privilege.

Comparing different secret management platforms, such as **AWS KMS, Azure Key Vault**, and **Google Cloud KMS** highlights their importance in securing cloud environments but also underscores the need for specialized solutions that allow effective integration with existing cloud infrastructures. Vault offers a universal approach, combining advanced cryptographic methods, centralized management, and dynamic credential management, making it a crucial tool for protecting confidential information.

Thus, using **HashiCorp Vault** as a universal secret management platform is an optimal solution for organizations seeking to enhance the security of their cloud environments, ensure regulatory compliance, and automate resource management. It not only minimizes risks associated with unauthorized access and data misuse but also significantly improves the efficiency of processes in modern dynamic infrastructures.

References

[1] R. Salecha, Security and Secrets Management. (2022). 10.1007/978-1-4842-8673-9_9.

[2] P. Skladannyi, et al., Improving the Security Policy of the Distance Learning System based on the Zero Trust Concept, in: Cybersecurity Providing in Information

and Telecommunication Systems, vol. 3421 (2023) 97–106.

[3] P. Somasundaram, Unified Secret Management Across Cloud Platforms: A Strategy for Secure Credential Storage and Access, Int. J. Comput. Eng. Technol. 15 (2024) 5–12.

[4] O. Vakhula, I. Opirskyy, O. Mykhaylova, Research on Security Challenges in Cloud Environments and Solutions based on the "Security-as-Code" Approach, Cybersecurity Providing in Information and Telecommunication Systems II, vol. 3550 (2023) 55–69.

[5] S. Shevchenko, et al., Information Security Risk Management using Cognitive Modeling, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems II, CPITS-II, vol. 3550 (2023) 297–305.

[6] V. Bysani, Automation in Cloud Infrastructure Management: Enhancing Efficiency and Reliability, Int. J. Sci. Res. Eng. Manag. 08 (2024). doi: 10.55041/IJSREM35750.

[7] P. Raj, Continuous Integration for New Service Deployment and Service Validation Script for Vault, Int. J. Sci. Res. Eng. Manag. (2024). doi: 10.55041/IJSREM35565.

[8] R. Banakh, A. Piskozub, Y. Stefinko, External Elements of Honeypot for Wireless Network, in: 13th International Conference on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET) (2016) 480–482. doi: 10.1109/TCSET.2016.7452093.

[9] R. Kyrychok, et al., Development of a Method for Checking Vulnerabilities of a Corporate Network using Bernstein Transformations, Eastern-European Journal of Enterprise Technologies, vol. 1, no. 9(115) (2022) 93–101. doi: 10.15587/1729-4061.2022.253530.

[10] V. Astapenya, et al., Conflict Model of Radio Engineering Systems under the Threat of Electronic Warfare, in: Workshop on Cybersecurity Providing in Information and Telecommunication Systems, CPITS, vol. 3654 (2024) 290–300.

[11] J. Chandra, Authentication and Authorization Mechanism for Cloud Security, Int. J. Eng. Adv. Technol. 8 (2019). 10.35940/ijeat.F8473.088619.

[12] A. S. George, S. Fernando, Securing Cloud Application Infrastructure: Understanding the Penetration Testing Challenges of IaaS, PaaS, and SaaS Environments. Partners Universal Int. Res. J. 02 (2023) 24–34. doi: 10.5281/zenodo.7723187.

[13] P. Kankwende, Cybersecurity_in_the Modern Age Protecting Digital Assets, J. Sci. Eng. Res. 15 (2024).

[14] S. T. Makani, Deep Dive into Terraform for Efficient Management of AWS Cloud Infrastructure and Serverless Deployment, Int. J. Comput Tech. 8(6) (2021).

[15] J. Almeida, et al., A Machine-Checked Proof of Security for AWS Key Management Service, CCS '19: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security (2019) 63–78. doi: 10.1145/3319535.3354228.

[16] T. Moore, et al., Encryption Methods and Key Management Services for Secure Cloud Computing: A Review (2023).

[17] P. Raj, Continuous Integration for New Service Deployment and Service Validation Script for Vault, Int. J. Sci. Res. Eng. Manag. 08 (2024). doi: 10.55041/IJSREM35565.

[18] A. Satapathi, A. Mishra, Storing Function Secrets in Azure Key Vault, Hands-on Azure Functions with C#. Apress (2021) 263–287. doi: 10.1007/978-1-4842-7122-3_11.

[19] P. Borra, Impact and Innovations of Azure IoT: Current Applications, Services, and Future Directions, Int. J. Recent Technol. Eng. 13 (2024) 21–26. doi: 10.35940/ijrte.B8111.13020724.

[20] R. Sajid, et al., Interpretation on the Google Cloud Platform and Its Wide Cloud Services. International Journal of Security and Privacy in Pervasive Computing, 14 (2022) 1–7. doi: 10.4018/IJSPPC.313586.

[21] P. Munyao, R. Chikoore, Impact of Cloud Solutions in Research Data Management, Scalable Computing and Collaborative Projects (2024). doi: 10.13140/RG.2.2.22876.40322.

[22] A. Kamaraju, A. Ali, R. Deepak, Best Practices for Cloud Data Protection and Key Management, Future Technologies Conference (FTC) 3 (2022). doi: 10.1007/978-3-030-89912-7_10.

[23] N. Golovacheva, M. Romanov, Study of Security Mechanisms Cloud Services, NBI Technologies (2023) 25–31. doi: 10.15688/NBIT.jvolsu.2023.3.3.

[24] S. Sarkar, S. Roychowdhury, Authentication Authorization and Security Issues in Cloud Computing, Int. J. Res. Appl. Sci. Eng. Technol. 11 (2023) 1275–1283. doi: 10.22214/ijraset.2023.56670.

[25] P. Somasundaram, Unified Secret Management Across Cloud Platforms: A Strategy for Secure Credential Storage and Access, Int. J. Comput. Eng. Technol. 15 (2024) 5–12.

[26] S. Ibn El Ahrache, H. Badir, Enhancing Cloud Data Security Through Long-Term Secret Sharing Schemes (2024). 10.21203/rs.3.rs-4770590/v1.

[27] J. Moazzam, Cloud Computing Security: AWS Data Security Credentials, Studies in Indian Place Names (UGC Care Journal) 40 (2020).

[28] P. Somasundaram, Unified Secret Management Across Cloud Platforms: A Strategy for Secure Credential Storage and Access, Int. J. Comput. Eng. Technol. 15 (2024) 5–12.

[29] A. Horpenyuk, I. Opirskyy, P. Vorobets, Analysis of Problems and Prospects of Implementation of Post-Quantum Cryptographic Algorithms, in: Classic, Quantum, and Post-Quantum Cryptography, vol. 3504 39–49.

[30] S. Nashkova, Addressing Criminal Liability for Misuse of Trade Secrets Under Australian Law: Is the Current Legal Framework Adequate to Protect the Interests of Owners of Trade Secrets?, IIC – International Review of Intellectual Property and Competition Law, 55 (2024) 1281–1315. doi: 10.1007/s40319-024-01490-4.

[31] Y. Martseniuk, et al., Automated Conformity Verification Concept for Cloud Security, in:

Cybersecurity Providing in Information and Telecommunication, vol. 3654 (2024) 25–37.

[32] J. Chandra, S. K. Pasupuleti, C. Narasimham, Authentication, Authorization and Auditing in Need of Internet of Things and Industry 4.0 for the Cloud Data Security, Authentication and Access Control for Secure Communication in Mobile Cloud Computing (2020) 70–81.

[33] V. Susukailo, I. Opirsky, O. Yaremko, Methodology of ISMS Establishment Against Modern Cybersecurity Threats, Future Intent-Based Networking, LNEE 831 (2022). doi: 10.1007/978-3-030-92435-5_15.

[34] R. Sajjan, V. Ghorpade, M. Dhange, Multi-factor Authentication as a Service for Cloud Data Security, Int. J. Comput. Sci. Eng. 4(4) (2016).

[35] K. R. Muppa, Advancing Cloud Security with AI-Enhanced AWS Identity and Access Management, Int. Res. J. Eng. Appl. Sci. (2022) 25–28. doi: 10.55083/irjeas.2022.v10i01005.