Ensuring High-Availability and Security for Secret Management in EGI Cloud

Viet Tran[a,*]

[a] *Institute of Informatics, Slovak Academy of Sciences*
Dubravska cesta 9, Bratislava, Slovakia
E-mail: viet.tran@savba.sk

Secret management is an important security service within the EGI Cloud Federation, encompassing the management of various types of secrets, including tokens and certificates, along with their secure delivery to target cloud environments. Given its critical role, high availability and security are paramount. The service must remain operational continuously, even in the event of individual component failures. This is achieved through the configuration of three geographically distributed Vault servers connected in a cluster, complemented by Dynamic DNS to facilitate seamless switching of generic service endpoints during failures. Additionally, the service incorporates several security enhancements, including client-side encryption and a locker mechanism to ensure the secure delivery of secrets to virtual machines (VMs) in the cloud.

International Symposium on Grids and Clouds (ISGC2024)
24 -29 March, 2024
Academia Sinica Computing Centre (ASGC), Institute of Physics, Academia Sinica
Taipei, Taiwan

*Speaker

1. Introduction

Applications in EGI Infrastructure [1] [2] require various secrets (credentials, tokens, passwords) for deployments and operations. These secrets are often stored in plain text within configuration files or code repositories, creating security vulnerabilities. Additionally, statically storing secrets makes them difficult to rotate. The secret management service for EGI Infrastructure, developed by IISAS in collaboration with INFN and CSIC, addresses these issues.

The Secret Management Service offers:

- Usability: A dedicated FedCloud client module seamlessly integrates the service with simple syntax. Authentication leverages OIDC tokens from EGI Check-in [4], eliminating the need for additional registration or credentials.
- Advanced Features: The service supports secret values from files, allowing import/export in YAML/JSON formats. Additionally, on-the-fly encryption/decryption of secret values on the client-side enhances security and trust.
- Compatibility: The service is built on Hashicorp's Vault [6], a well-established industry solution with numerous client tools and libraries. Both service and client software are open-sourced with strong community support.
- High Availability: Service instances are distributed across different sites, eliminating single points of failure. A generic endpoint (*https://vault.services.fedcloud.eu:8200*) dynamically points to a healthy instance through a Dynamic DNS service [5].

This paper proposes a novel approach for delivering secrets to VMs in the cloud via a locker mechanism. Users can effortlessly create lockers, deposit secrets within them, and then provide the locker's token to their VMs. This eliminates the need to store user credentials on VMs for accessing the secret management service, significantly improving overall security.

2. Ensuring High Availability for the EGI Secret Management Service

High availability is essential for critical services like secret management, which must remain operational even when individual components fail. A common strategy for achieving high availability is through redundancy, which eliminates single points of failure by deploying multiple service instances. If the primary instance fails, a backup seamlessly takes over, ensuring uninterrupted service for users.

However, this failover process can be complex, often requiring a proxy server to act as a dispatcher or load balancer. While effective, this approach introduces its own challenge: the proxy itself must be highly available, often requiring costly hardware redundancy.

An alternative, more cost-effective solution for some services is Dynamic DNS, which simplifies high availability by managing service failover without the need for additional expensive hardware. In this section, we will detail how to leverage Dynamic DNS to achieve high availability for the EGI Secret management service.

2.1 Geographically distributed Vault servers

The EGI Secret Management service comprises a cluster of three interconnected Vault servers -one active and two on standby (Figure 1). If the active server fails, one of the standby servers automatically takes over. Data replication between the servers is managed by the Raft Consensus algorithm, using Vault's integrated storage system. Users can connect to any of the servers directly, and if they connect to a standby server, it will seamlessly forward their requests to the active server.

To minimize disaster risks, the servers are distributed across data centers in three different countries: IISAS in Slovakia, INFN in Italy, and IFCA in Spain.

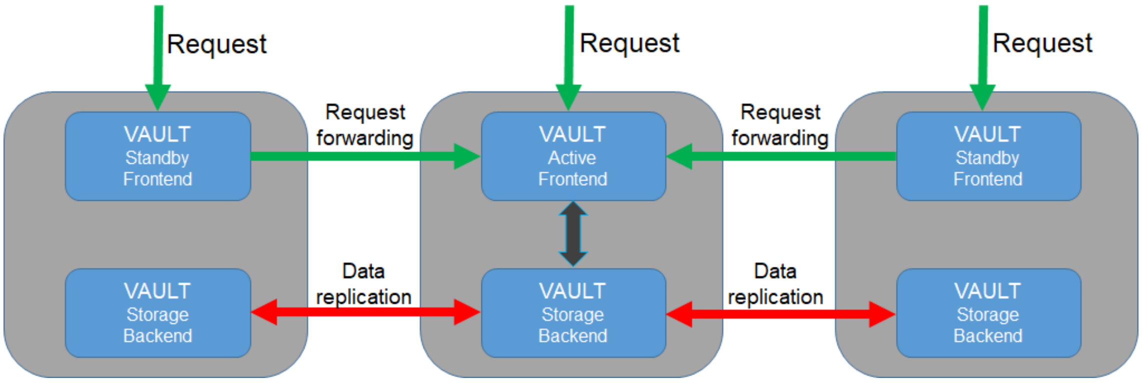

Figure 1: Service architecture

2.2 Dynamic DNS service

Nowadays, more and more services are dynamically deployed in Cloud environments. Usually, the services hosted on virtual machines in Cloud are accessible only via IP addresses or pre-configured hostnames given by the target Cloud providers, making it difficult to provide them with meaningful domain names [3]. The Dynamic DNS service was developed by Institute of Informatics, Slovak Academy of Sciences (IISAS) to alleviate this problem.

The Dynamic DNS service provides a unified Dynamic DNS support for virtual machines across the EGI Cloud infrastructure. Users can register their chosen hostnames in predefined domains (e.g., my-server.vo.fedcloud.eu) and assign them to the public IPs of their servers.

The Dynamic DNS service significantly simplify the deployment of services that are dynamically deployed in Cloud infrastructures. It removes the obstacles of changing IP addresses of services in Cloud at every deployment and enables obtaining SSL certificates for the hostnames. Service providers can migrate services from local servers to Cloud or from a Cloud site to another without noticing users from the change.

The architecture of the Dynamic DNS service is illustrated in Fig. 2. The core component of the service is an NS-update server, which consists of a GUI front-end, a REST API, and a back-end engine. Users log into the GUI front-end using their EGI Check-in account [6], where they can register hostnames within supported domains (e.g., by default, *vm.fedcloud.eu*). Once registered, users can assign or update the IP addresses of VMs or servers to these hostnames, allowing easy access to services through the registered names.

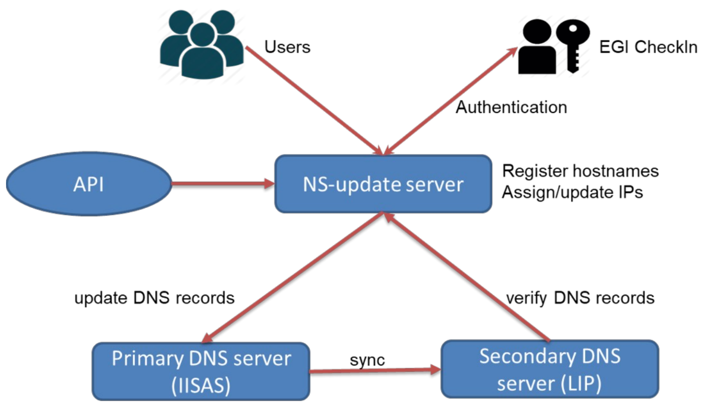

Figure 2: Architecture of Dynamic DNS service

The communication between the service's components relies on standardized protocols, ensuring long-term sustainability and allowing for seamless replacement of components with compatible software and services. The primary and secondary DNS servers are powered by BIND9 [9], the industry-standard DNS software developed at the University of California, Berkeley, and maintained by the Internet Systems Consortium. For high availability, the primary DNS server is hosted at IISAS in Slovakia, while the secondary servers are distributed across IFCA in Spain and INCD in Portugal. Synchronization between these servers is performed using the zone transfer protocol (RFC 5936), following IETF DNS standards.

The NS-update server manages the addition, update, and deletion of hostnames in DNS servers via the nsupdate protocol (RFC 2136), which is part of the IETF standards for DNS. Authentication and authorization between components are handled via pre-shared secrets, following the TSIG standard (RFC 2845). Every DNS modification is carried out in two steps: first, the NS-update server applies changes (add/update/delete) to the primary DNS server, and then verifies the results on a secondary server to ensure the updates are properly applied and active.

User authentication and authorization on the NS-update server are handled using the OpenID Connect protocol (IETF RFC 6749 and 6750) integrated with EGI Check-in. This removes the need for the NS-update server to manage user registration, enabling easy integration with the broader EGI Cloud infrastructure. Since OpenID Connect is supported by a wide range of identity providers—academic communities like ORCID [10] and eduTEAMS [11], as well as commercial platforms like Google and Microsoft—it is straightforward to add support for additional identity providers if needed.

The client can update the IP address of a registered hostname using the DynDNS2 protocol [12], which is compatible with popular Dynamic DNS services like *dyn.com* and *noip.com*. The key distinction between EGI Dynamic DNS and commercial services is that EGI generates a unique password for each registered hostname. When a user updates the IP address via DynDNS2 (e.g., using the *ddclient* tool), the hostname serves as the username, while the corresponding password. This allows users to update IP addresses without needing their personal credentials, simplifying update scripts and enhancing security.

2.3 Ensuring High Availability for the EGI Secret Management via Dynamic DNS

As mentioned above, the EGI Secret Management service comprises a cluster of three interconnected Vault servers. Users can use any of the endpoints directly:

- *https://vault-iisas.services.fedcloud.eu:8200*,
- *https://vault-infn.services.fedcloud.eu:8200*,
- *https://vault-ifca.services.fedcloud.eu:8200*,

However, direct access to these endpoints is not recommended, as they may become unavailable if a server goes down. For convenience and high availability, two generic endpoints *https://vault.services.fedcloud.eu:8200* and *https://secrets.egi.eu* is created for accessing the service. These endpoints dynamically attach to one of the active servers, either at INFN or IFCA, via the Dynamic DNS service hosted at IISAS.

A simple *cron* script runs on the INFN and IFCA servers to periodically check their health status. It automatically assigns the generic hostname (*vault.services.fedcloud.eu*) to the active and healthy server. In the case of unscheduled downtime, the recovery time for the generic endpoint is *T+1* minutes, where *T* is the *cron* check interval (set to 1 minute).

For scheduled maintenance, administrators can easily reassign the generic endpoint to another server before shutting down the service, ensuring uninterrupted availability.

This configuration guarantees that the EGI Secret Management service remains accessible through the generic endpoint, even in the event of a complete data center failure caused by disasters or power outages.

3. Security enhancements for EGI Secret management service

3.1. FedCloud client for EGI Secret management service

The EGI Secret Management service is built on HashiCorp's Vault, which offers a web-based GUI and is compatible with various client tools and libraries. To further enhance usability and security, a dedicated command-line client has been developed as a module of the FedCloud client [7], the generic client for the EGI Cloud Federation.

The key features of the FedCloud client for the Secret Management service include:

- Out-of-the-box functionality: When using the native Vault CLI, users need to manually configure several parameters, including the service URL, personal folder path, and authentication details. The FedCloud client has these parameters pre-configured, allowing it to work immediately without additional setup.
- Single-step secret access: The native Vault client requires two steps to access a secret: first, the user must log in to the service using an access token, and then they can retrieve the secret using the Vault token. The FedCloud client automates these steps, enabling users to access secrets directly with a single command.
- Integration with the EGI Cloud ecosystem: The FedCloud client integrates seamlessly with various tools and services in the EGI Cloud Federation, including *oidc-agent* and *mytoken* for managing access tokens, EGI Check-in for authentication, and VO-based authorization.

- Enhanced security: The FedCloud client offers additional security features such as client-side encryption for secrets and "lockers," which are described in detail in the following sections.

3.2 Protecting Secrets with Client-Side Encryption

For highly sensitive secrets, FedCloud client provide users the option to encrypt them prior to storing them in the service. This ensures that no individual, including service administrators, can access the secrets in plain text. Encryption is performed automatically on-the-fly if users provide an encryption key (passphrase) using the *--encrypt-key* option when putting secrets to the service, for example:

fedcloud secret put certificate cert=@hostcert.pem --encrypt-key passphrase

Decryption is carried out automatically in a similar manner. Users provide the decryption key using the *--decrypt-key* option when retrieving the secret. The value will be automatically decrypted if the key is valid:

fedcloud secret get certificate cert --decrypt-key passphrase

Passphrases are assigned individually to each secret, allowing for different passphrases for different secrets. This practice minimizes potential damage in the event that a passphrase is disclosed.

The FedCloud client employs Fernet symmetric encryption from the standard Python Cryptography library [8] for secure client-side encryption. Specifically, it utilizes:

- Advanced Encryption Standard (AES) in Cipher Block Chaining (CBC) mode with a 128-bit key for encryption, incorporating PKCS7 padding.
- Hashed Message Authentication Code (HMAC) with SHA-256 for message integrity verification.
- Initialization vectors generated using *os.urandom()*.

3.3 Secure secret delivery via locker mechanism

By default, accessing secrets stored in the EGI Secret management service from virtual machines (VMs) has relied on OIDC access tokens, a method that harbors potential security vulnerabilities. In the event of VM compromise, these access tokens can be pilfered, enabling attackers to gain access to all user secrets.

The locker mechanism in the EGI Secret management service introduces an innovative and robust approach to securely deliver secrets to VMs. Instead of permanently storing secrets in the Vault's key-value secret engine, users can create a locker – a temporary, isolated space within the Vault's secret engine *cubbyhole*. They can store the necessary secrets in the locker and provide the locker token to the VMs. Scripts running on the VMs will use the locker token to access the secrets without requiring users' personal credentials.

For enabling the locker mechanism, a special workflow has been developed in FedCloud client automates all security settings for users. That includes:

- Generate the locker tokens in EGI Secret management service
- Set limits for lifetimes and number of uses for the locker tokens
- Revoking all privileges associated with the locker tokens, except for accessing the lockers.

This automation ensures that all lockers are securely configured without user intervention, minimizing the potential for errors in security settings and simplifying access.

The key security attributes of the locker system include:

- Temporary and autoclean: Lockers have a limited lifespan and quantity. Upon expiration, lockers are automatically purged, along with all the secrets contained within them.
- Isolation: Access to the secrets within a locker is exclusively through its associated token, which can solely be used for accessing the locker's secrets—nothing more. This isolation allows users to store tokens in Continuous Integration/Continuous Deployment (CI/CD) pipelines and similar tools, mitigating the risk of exposing personal secrets.
- Malfeasance detection: The locker mechanism possesses the capability to detect if a token has been compromised and is being misused. More details are provided in the next section.

3.4 Single-use locker

In some scenarios, users may need to transmit secrets through untrusted communication channels. This poses a significant security risk, as stolen secrets (passwords, tokens, etc.) can be exploited by attackers. However, the most concerning threat is the potential for attackers to silently misuse these secrets for extended periods, causing significant damage before detection.

Single-use lockers offer a simple solution to this problem. These lockers have a mechanism for self-destruction after successful delivery. This means that even if the secrets are intercepted during transmission, their misuse can be immediately detected because the locker becomes invalidated upon opening. This significantly reduces the risk of undetected compromise and its potential consequences.

The process of delivering secrets with single-use lockers involves the following steps:

- Create a locker: The sender creates a locker with the number of uses set to 2.
- Store secrets: The sender deposits the secrets (passwords, tokens, etc.) into the locker. That will reduces the number of uses to 1. The locker now is truly single-use.
- Send locker token: The sender transmits the unique locker token to the recipient, potentially through an untrusted communication channel.
- Access and verification:
  - Successful access: If the recipient can access the secrets using the token, it signifies successful delivery. This also guarantees that no one else has accessed the secrets before (as the locker becomes unusable after the first access).
  - Failed access: If the recipient cannot access the secrets, it suggests a potential compromise. This triggers actions like notifying administrators, changing passwords, revoking tokens, initiating investigations, and taking other necessary security measures.

4. Conclusion

In this paper, we have presented a highly available and security-enhanced secret management service for the EGI Cloud Federation. The service's high availability is achieved through the deployment of geographically distributed servers connected in a cluster, which ensures continuous operation even in the event of individual component failures. The integration of Dynamic DNS facilitates seamless switching of service endpoints, further enhancing resilience during outages.

In addition to its robust availability features, we have implemented several significant security enhancements. Client-side encryption is employed to ensure that secrets remain confidential and inaccessible to unauthorized users, including service administrators. This approach not only protects sensitive information but also instills trust in users relying on the service.

Furthermore, the locker mechanism provides a secure method for delivering secrets to VMs without exposing user credentials. This mechanism allows for the temporary storage of secrets in an isolated environment, thereby reducing the risk of credential leakage while facilitating efficient access.

Acknowledgment

This work is supported by APVV grant No. APVV-20-0571 and VEGA grant No. 2/0131/23.

References

[1] Enol Fernandez-del-Castillo et al, “The EGI Federated Cloud e-Infrastructure”, Procedia Computer Science, vol. 68, pp. 196-205, 2015. https://doi.org/10.1016/j.procs.2015.09.235.

[2] Spiga, Daniele, et al. "The DODAS Experience on the EGI Federated Cloud." EPJ Web of Conferences. Vol. 245. EDP Sciences, 2020. https://doi.org/10.1051/epjconf/202024507033

[3] Marcus Hardt and Viet Tran, “EOSC SYNERGY HANDBOOK: A Handbook targeted at Computer Centre Management, Administrators, and Users of the Infrastructure”, 2022, http://hdl.handle.net/10261/280037.

[4] EGI Check-in home. https://aai.egi.eu/. Accessed: 16 September 2024.

[5] Dynamic DNS. https://docs.egi.eu/users/compute/cloud-compute/dynamic-dns/. Accessed: 16 September 2024.

[6] Vault by HashiCorp. https://www.vaultproject.io/. Accessed: 16 September 2024.

[7] FedCloud client. https://fedcloudclient.fedcloud.eu/. Accessed: 16 September 2024.

[8] Fernet (symmetric encryption) cryptography. https://cryptography.io/en/latest/fernet/#implementation. Accessed: 16 September 2024.

[9] Nathan, Shelena Soosay, et al. "BERKELEY INTERNET NAME DOMAIN (BIND)." International Journal on Cybernetics & Informatics (IJCI) Vol.1, No.1, Feburay 2012

[10] ORCID, OpenID Connect, and Implicit Authentication, https://info.orcid.org/orcid-openid-connect-and-implicit-authentication/. Accessed: 16 September 2024.

[11] eduTEAMS, https://eduteams.org/. Accessed: 16 September 2024.

[12] ddclient protocols DynDNS2. https://sourceforge.net/p/ddclient/wiki/protocols/#dyndns2/. Accessed: 16 September 2024.