

Secrets Management Enterprise Technology Guideline

1. Scope

This Enterprise Technology Guideline (ETG) represents the official technology architecture standard for secrets management in VA IT systems. For more details, please visit the Technology Architecture Guidelines Confluence site (log-in required):

[Enterprise Technology Guidelines](#)

[Secrets Management Utility on CODE VA](#)

2. Background

A “secret” refers to any privileged credential that acts as a key to unlock protected resources or sensitive information in tools, applications, containers, and computing environments. Secrets can include:

- User or auto-generated passwords, including one-time passwords (OTP)
- Application Programming Interface (API) and other application keys/credentials (including within containers)
- Secure Shell (SSH) keys
- Database and other system-to-system usernames and passwords
- Private certificates for secure communication, transmitting and receiving of data (e.g., TLS)
- Private encryption keys for systems

Secrets management refers to the framework, tools, and methods for managing digital authentication credentials (secrets), including passwords, keys, APIs, and tokens for use in applications, services, privileged accounts, and other sensitive parts of the IT ecosystem.

3. Problem

Managing secrets can be complicated in an environment like VA, where we have multiple hosting environments, application types, and deployment processes on the network.

Despite these challenges, assuring the confidentiality, integrity and availability of VA systems and data depends on OIT effectively managing secrets.

The following problems pertain to secrets management:

- Problem 1: Not storing privileged secrets in a secure location presents challenges to protecting access to systems containing PII or PHI data.
- Problem 2: Not having a standard set of tools for secret management, resulting in secrets sprawl across different applications and hosting environments.
- Problem 3: "Secret Zero" occurs when all secrets are protected under another secret, a single master key that is a potential vulnerability that grants access to everything.
- Problem 4: Secrets are hard coded into applications.

4. Secret Management Solutions

This section describes the current secret management approaches in use at VA, and how to choose which of these solutions is right for your project. Please note that as of FY2025, there is no "single-source" solution for secrets management, but rather approved solutions that are available for specific use cases such as hosting an application on Microsoft Azure.

Privileged Access Security (PAS) Solution: Infrastructure Operations (IO) User Network Monitoring (UNM) group owns the Conjure Secret Manager and Core [Privileged Access Security \(PAS\)](#) solution. These tools represent VA's official Enterprise Solution in VA-managed Information Technology Centers (ITCs).

The PAS solution manages privileged human and non-human accounts, including but not limited to the local, domain, shared, and applications accounts and certificates, in a secure vault. Having privileged accounts in the secure vault provides who used the account, when the account was used, a recording of what was being accomplished, and the management of the passwords changed per VA's policy (VA Handbook 6500). The PAS solution enables the following capabilities:

- Dynamic insertion of secrets at runtime
- Management of privileged credentials
- Isolation and monitoring of privileged session
- Threat detection and response

- Management of nomadic devices
- Remote access to the PAS solution
- Adaptive Multifactor Authentication (MFA) and Single Sign-on (SSO)

How to access PAS: [Self-Service: Helpful Link, PAS Solution Training, and Service Requests](#).

For more information, contact the PAS team at OITTOPSSOIOSecurityMgmtCyberArk@va.gov. The PAS solution requires elevated privileges through the Enterprise Permission Access System (ePAS): [MyVA Elevated Privileges](#).

In addition to PAS, IO also owns [Conjur Secrets Management](#). It can be used along with PAS to manage non-human application accounts and hard-coded passwords in applications along with privileged credentials used in DevOps tools. Conjur Secrets Management provides a secure connection from the application to the password vault preventing an insider threat/hacker from a man-in-the-middle attack from gaining the password, protecting access to other systems or, more importantly, PII or PHI data. The same activity of who used the account and what was accomplished with the account is also tracked.

HashiCorp Vault: This [TRM-approved product](#) is an alternative to the PAS solution that is available to project teams working with on-premises solutions, or solutions hosted in the VAEC. It is currently not a Software Factory Utility but a product that can be individually purchased and installed. Vault is a technology for secrets management, encryption as a server, and privileged access management. This technology allows users to view who is accessing what data such as database credentials, Application Programming Interface (API) keys for external services or credential for service-oriented architecture communication.

Use Vault for on-premises solutions and for cloud-hosted solutions hosted in the VAEC. Use when the solution includes other HashiCorp products like [Terraform](#) and [Consul](#). Vault is not available as an Enterprise Solution as of FY25, but it is listed as part of the Software Factory Utilities on the Catalog of Developer Essentials (CODE) VA site. Vault is individually purchased, installed, and used successfully by several programs, including Loan Guaranty ([VASI ID 1489](#)), and it is also used by VA Profile ([VASI ID 2203](#)). Vault is also made available through [VA Platform One \(VAPO\)](#). Each Vault instance is managed per guidelines in the TRM.

How to access Vault: Regarding purchasing, installing and/or using Vault, please contact your local acquisition/procurement office, your local ISSO staff, and/or the Enterprise Service Desk at (855) 673-4357 or <https://yourit.va.gov/va> for assistance.

How-to articles:

- [Getting Started with Vault](#)

AWS Key Management Service (KMS): Use Amazon Web Services (AWS) KMS when developing an application in the VA Enterprise Cloud (VAEC) AWS environment using only AWS managed services. AWS KMS is a managed service that makes it easy to create and control the cryptographic keys that are used to protect data. AWS KMS uses hardware security modules (HSM) to protect and validate the AWS KMS keys under the [FIPS 140-2 Cryptographic Module Validation Program](#).

AWS KMS is provided in the VAEC Service Catalog, following the unified intake process for VAEC service requests: [VAEC Service Request - yourIT Service Portal](#)

How-to articles:

- [AWS KMS Developer Guide](#)

Azure Key Vault: Use Azure Key Vault when developing an application in the VA Enterprise Cloud (VAEC) Azure environment using only Azure managed services. Azure Key Vault encrypts keys and small secrets like passwords that use keys stored in hardware security modules (HSMs). For more assurance, Key Vault imports or generates keys in HSMs, and Microsoft processes keys in FIPS validated HSMs (hardware and firmware) - FIPS 140-2 Level 2 for vaults and FIPS 140-2 Level 3 for HSM pools. With Key Vault, Microsoft does not see or extract the keys. Additionally, users can monitor and audit key use with Azure logging, which can integrate with a security information and event management (SIEM) solution (e.g., Splunk) for more analysis and threat detection.

Azure Key Vault is provided in the VAEC Service Catalog, following the unified intake process for VAEC service requests: [VAEC Service Request - yourIT Service Portal](#).

How-to articles:

- [Azure Key Vault Developer Guide](#)

Microsoft recommends using separate key vaults when integrating with Key Vault. Use a vault per application per environment (development, pre-production, and production),

per region. This helps you not share secrets across environments and regions. It will also reduce the threat in case of a breach.

5. GitHub Secret Scanning Guidelines

- Use GitHub Advanced Security (GHAS) Secret Scanning to find vulnerabilities and errors in the code if you are using a VA GitHub.com repository:
<https://docs.github.com/en/code-security/code-scanning/automatically-scanning-your-code-for-vulnerabilities-and-errors/about-code-scanning>
 - GitHub Advanced Security scans the history of all commits in the repository (repo) for known patterns. From that point on GitHub Secret Scanning will scan all new commits that are pushed.
 - Check out which secrets GitHub scans for within the repository in real-time: <https://docs.github.com/en/enterprise-cloud@latest/code-security/secret-scanning/secret-scanning-patterns#supported-secrets-for-advanced-security>.
- Enforce a pull request process to ensure that the code is reviewed for secret disclosures.
 - While this does not stop secrets from landing in feature branches, it does catch leaked secrets as part of the merge to main/master flow and remediating in a feature branch is much less impactful than remediating in production branches.
- When a secret has been detected: Project team is notified of the leak and performs the following activities immediately upon detection (within a few hours):
 - Rotate the secret immediately, by ensuring that the secret can no longer be used (i.e., by resetting a password, generating a new token, revoking a credential, etc.)
 - Remove the secret from source code. Note that Git stores the history of the commits, so it is not enough to delete the secret from the latest commit in the branch. Go back and delete it starting at the commit it was introduced in.
 - GitHub recommends using git-filter-repo: <https://github.com/newren/git-filter-repo>.
 - Check relevant system logs to determine if any unauthorized access to resources protected by the secret were detected from the time the secret was leaked.
 - Write up the leakage as a security incident for internal documentation; if the severity of the leak meets VA incident guidelines, then notify ISSO and VA project manager.

- Write up a root cause analysis and investigate ways to prevent whatever happened from happening again.
- If the alert remains open for 24 hours, the person who introduced the secret and the application owner are notified, and the project team has 3 days to remediate (rotate and remove from code) and respond. After this time GitHub's Security team is notified and then you start getting escalations from security.

6. Secret Management Dos and Don'ts

DON'T:

- Put secrets into source code checked in to GitHub, regardless of whether the source code repository is public, internal-only, or private.
- Store secrets in a shared document and collaboration tools like SharePoint or encrypted file.
- Enter passwords or PII/PHI in clear text when using a command line interface (CLI).
- Re-use the same secret for multiple purposes.
- Store secrets in clear text.
- Transmit secrets in emails or in collaboration tools (Teams, Slack) in clear text.
- Put secrets into application codes on Microsoft Internet Information Services (IIS) servers.

DO:

- Store secrets in a dedicated secret management tool.
- Store secrets outside of container images.
- Provide secrets dynamically at runtime.
- Encrypt secrets at rest utilizing FIPS 140-2 cryptography.
- Record secret lifecycle and access events in an audit log.
- Rotate secrets per VA Handbook 6500 security policy.

For more information: [GitHub Security Policies: Secrets Management](#)

7. Secret Management Checklist

Project teams should reference the checklist below to ensure they are managing secrets correctly:

- Use one of the solutions described in this document for secrets management or VAEC-approved services (AWS KMS or Azure Key Vault).
- Enforce a pull request process with GitHub Advanced Security (GHAS) to ensure that the code is reviewed for secret disclosures.
- Rotate secrets immediately upon detection of a secret disclosure.
- Conduct notification and assessment of severity immediately upon a secret disclosure detection (within a few hours).
- Remove expired certificates per VA security policies to maintain Authority to Operate (ATO).
- Create access policies for secrets vaults, including role-based access control (RBAC).

Document Version Control

Date	Version	Change Details	By
11/15/2022	1.0	Final edits after review by OIS and approval by the CTO.	OCTO/AES
12/14/2023	2.0	Annual refresh and update of document to reflect the latest guidance for FY24.	OCTO/AES
11/22/2024	3.0	Annual refresh and update of document to reflect the latest guidance for FY25.	OCTO/AES
11/18/205	4.0	Annual refresh and validation for currency.	DSD/TAS

Disclaimer: This document serves both internal and external customers. Links displayed throughout this document may not be viewable to all users outside the VA domain. This document may also include links to websites outside VA control and jurisdiction. VA is not responsible for the privacy practices or the content of non-VA websites. We encourage you to review the privacy policy or terms and conditions of those sites to fully understand what information is collected and how it is used.

Statement of Endorsement: Reference herein to any specific commercial products, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, and shall not be used for advertising or product endorsement purposes.