Secrets Management Enterprise Design Pattern

1. Scope

This Enterprise Design Pattern (EDP) is developed by the Architecture and Engineering Service (AES) within the Office of the Chief Technology Officer (OCTO), to provide capability guidance, technical architecture standards, and industry best practices related to secrets management.

2. Context

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)
- API and other application keys/credentials (including within containers)
- 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. Secrets Management Dos and Don'ts

Keep in mind the following DOs and DON'Ts related to secret management.

DON'T:

- Put secrets into source code checked in to source code version control systems such as 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 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 policy

5. How to Manage Secrets

This section describes the secret management approaches in use at VA, and how to choose which of these solutions is right for your project.

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 the PAS solution: Self-Service: Helpful Link, PAS Solution Training, and Service Requests. The PAS solution is currently managed as a product (VASI ID 2448) in the Cybersecurity Product Line led by Greg Watson (Gregory.watson@va.gov).

For more information, contact the PAS team at OITITOPSSOIOSecurityMgmtCyberArk@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. 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. However, it is currently purchased, installed, and used successfully by several individual programs, including Loan Guaranty (VASI ID 1489), and it is also used by VA Profile (VASI ID 2203). 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: WFM Customer Portal - Service project (va.gov). Please contact Christopher Cardella at Christopher.Cardella@va.gov for more information on how to integrate with VAEC-provided services that make up the Enterprise Cloud Solutions Product Line.

How-to articles:

- AWS Key Management Service (KMS) - VAEC Knowledge - VAEC Knowledge
- 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: WFM Customer Portal - Service project (va.gov). Please contact Christopher Cardella at Christopher.Cardella@va.gov for more information on how to integrate with VAEC-provided services that make up the Enterprise Cloud Solutions Product Line.

How-to articles:

- Azure Key Vault Overview - VAEC Knowledge - VAEC Knowledge
- 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.

6. How to Protect Against and Remediate Secret Disclosures

- Use GitHub Advanced Security to find vulnerabilities and errors in the code for the project: 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 then 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.

7. Secrets Management Checklist

Project teams should reference the below checklist 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 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).

Appendix A: VA References

1. Applications Security Container Pattern, Office of Information Security (OIS), Version 2.4, Pages 14, 37. ESA DevSecOps (sharepoint.com)
2. VA Handbook 6500: https://www.va.gov/vapubs/Search_action.cfm?FormNo=6500
3. VA Directive 6551: https://www.va.gov/vapubs/viewPublication.asp?Pub_ID=829&FType=2
4. VA GitHub Handbook: https://department-of-veterans-affairs.github.io/github-handbook/
5. VA TRM categories covering secrets/key management Categorization Help File (va.gov)
   a. Application Security
   b. Encryption
   c. Security Administration Software
   d. Network Auditing

Appendix B: Summary of Industry Best Practices

1. Code & secret management best practices - GitGuardian Blog
2. Secrets management guide — approaches, open-source tools, commercial products, challenges and questions | by Mike Burshteyn | Medium).
3. Gartner: Managing Machine Identities, Secrets, Keys and Certificates, Eric Wahlstrom, 16 March 2022.

4. What is Secrets Management? - Definition | CyberArk

Document Version Control

Date	Version	Change Details	By
09/19/2022	0.1	Original Draft	Mike Dance
09/26/2022	0.2	Incorporated CTO's suggestions. Added Benefits section.	Mike Dance
09/27/2022	0.3	Listed problems as a separate section.	Madhavi Nookala
09/27/2022	0.4	Included detail on CyberArk PAS access	Madhavi Nookala
09/28/2022	0.5	Included detail on Azure Key Vault and AWS Key Management System detail	Grace-Marie Kolb
09/28/2022	0.6	Improved document flow by rearranging AWS, Azure, CyberArk, and Hashicorp Vault to highlight use cases	Mike Dance
10/12/2022	0.7	Added knowledge articles and other references, addressed CTO feedback, and gathered stakeholder inputs (Andrea Birkland, GitHub Team).	Mike Dance
11/2/2022	0.9	Incorporated inputs from the GitHub team. Additional edits to address stakeholder feedback before preparing final version.	Mike Dance
11/15/2022	1.0	Final edits after review by OIS and approval by the CTO.	Mike Dance