Section 7

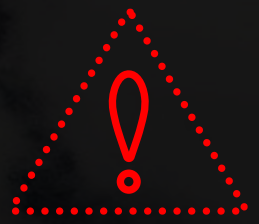
Vault
Security Hardening

Vault Security Hardening

- Vault is a security product, and it itself must be secured
- Equally as important, the underlying operating system must be secured along with the environment where Vault runs

- A mixture of both Vault and OS security hardening is essential for the overall security of Vault

- Suggestions are grouped into three topics:
  - General
  - Operating System
  - Vault-specific



Disclaimer

The suggestions in the following slides and lecture are merely suggestions on securing a Vault environment. It is possible to run Vault without these settings. Consult your own security team or experts to validate the suggestions included in this lecture as well as ask for additional settings required for your specific environment.

I am not a security expert. There are configurations that might not appear in this lecture. These are topics that should trigger your own ideas on how to secure your environment.

General Topics

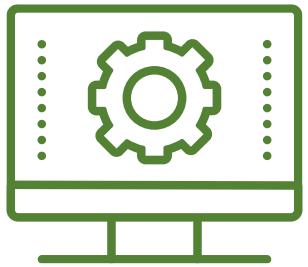
Deployment Model

- The fewer shared resources, the better
- Secure deployments: Hardware > VMs > Containers
- Ultimately comes down to protecting memory contents

Limit Access to Vault Nodes

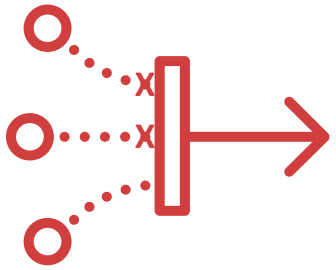
- Reduce or eliminate access to Vault nodes
- Includes SSH and through cloud-based access (i.e., AWS SSM)
- Use immutable upgrades where possible

Limit Services on Vault Nodes



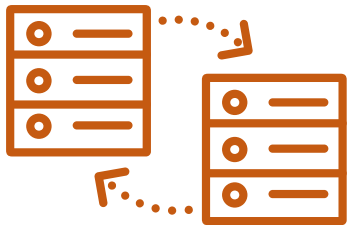
- Vault nodes should be dedicated to Vault services
- Encryption keys are stored in-memory
- More services = more firewall requirements

Firewalls Should Permit Only Required Ports



- Vault and Consul use dedicated ports for communication
- Permit only the required ports to reduce attack surface
- Many Vault deployments don’t even allow SSH or UI ports

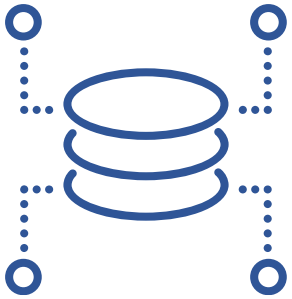
Immutable Upgrades



- Vault stores no data locally (beyond Audit Devices)
- Immutable upgrades guarantees a known state
- Easy to bring new online, destroy the old
- Consul can use AutoPilot to assist with upgrades (Ent)

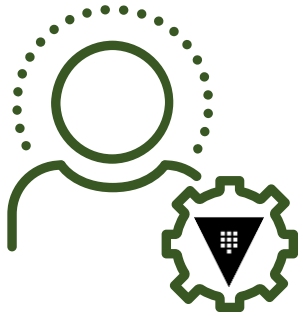
Operating System Topics

Disable Swap



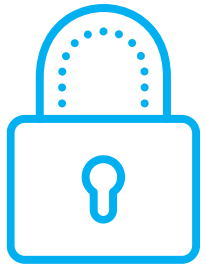
- Vault stores sensitive data in-memory, unencrypted
- That data shouldn't be ever be written to disk
- Disabling swap provides an extra layer of protection

Run Vault as an Unprivileged User



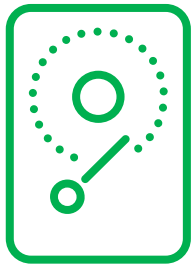
- Never run Vault as Root
- Running as root can expose Vault's sensitive data
- Limit access to configuration files and folders to the Vault user
- I normally use a user named "vault"

Secure Files and Directories



- Protect and audit critical Vault directories and files
- Ensure unauthorized changes can’t be made
- Includes binaries, config files, plugins files and directory, service configurations, audit device files and directory, etc.

Protect the Storage Backend

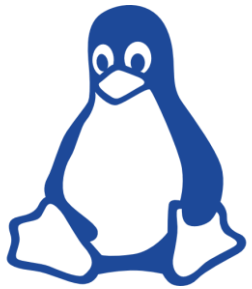


- Vault writes all configuration and data to the storage backend
- No storage backend = No Vault!
- Use Consul ACLs when running on Consul
- See Consul ACLs for Vault section for in depth coverage

Disable Shell History

- Disabling history prevents retrieval of commands
- Possible to discover credentials/tokens in history
- Can also disable all just vault command in history

Configure SELinux/AppArmor

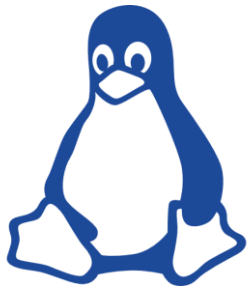


- Don’t disable to make install/management easier
- Provides additional layers of protection for the OS
- Adhere to CIS or DISA to improve posture of the host OS

Turn Off Core Dumps

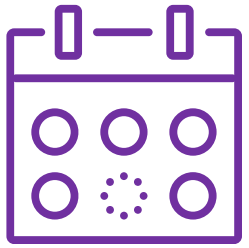
- Core dumps could reveal encryption keys
- Different process depending on the OS

Protect and Audit the vault.service File



- Make sure you know if this file is modified or replaced
- An attacker could point to compromised binaries to leak data

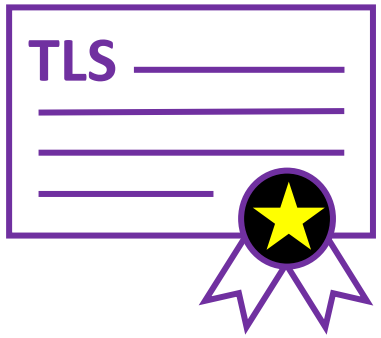
Patch the Operating System



- Make sure to patch the OS frequently
- Follow your standards or be more stringent for Vault
- Options include Satellite, SpaceWalk, or other solution

Vault-Specific Topics

Secure Vault with TLS



- Vault contains sensitive data
- Communications should never occur without TLS in place
- Load Balancers used with Vault should *not* terminate TLS and instead use pass through to the Vault nodes.
- Verify “tls_disable” configuration does not equal “true” or “1”

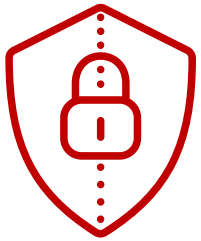
Secure Consul with TLS

- Consul contains sensitive data
- Communications should never occur without TLS in place
- Issue a certificate from a trusted CA

Enabling Auditing

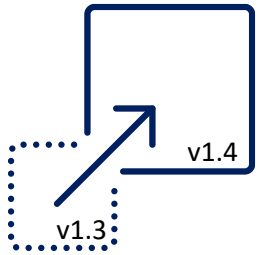
- Use multiple Audit Devices to log all interactions
- Send that data to a collection server
- Archive log data based upon security policies
- Create alerts based on certain actions

Clear Text Credentials



- Don’t put credentials in configuration files
- Use Environment Variables, where supported
- Use cloud-integrated services, such as AWS IAM Roles or Azure Managed Service Identities

Upgrade Frequently

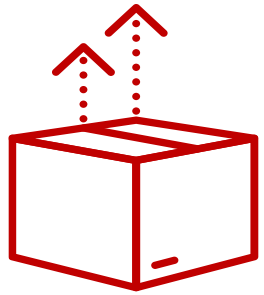


- New updates frequently include security fixes
- New cipher suites can be enabled or supported
- New functionality enabled

Stop Using Root Tokens

- Root tokens have unrestricted access to Vault
- Not bound by ACL policies, ERPs, or RGPs
- Get rid of the initial root token after initial setup
- vault token revoke s.xxxxxxxxx

Verify the Integrity of Vault Binary

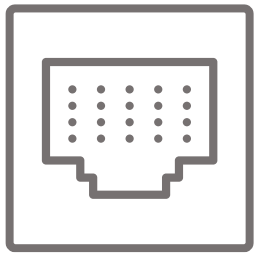


- Always get Vault binaries directly from HashiCorp
- Use the HashiCorp checksum to validate
- Modified version of Vault binary could leak data

Disable the Vault UI - if Not in Use

- Vault UI is disabled by default
- Configured in the Vault configuration file
- Do the same for Consul UI, as well

Encrypt the Consul Gossip Protocol

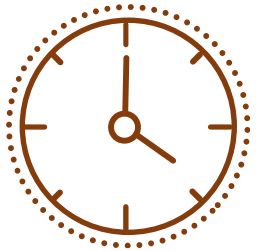


- TLS only secures the interfaces, not Consul gossip traffic
- Use the –encrypt flag in the Consul configuration file
- Uses a 32-byte key – can use consul keygen to generate

Secure Unseal Keys

- Initialize Vault using PGP keys, such as keybase.io
- Distribute to multiple team members, no single person should have all the keys
- Don’t store the keys in Vault itself

Minimize TTLs



- Use the smallest TTL possible for tokens
- Define Max TTLs to prevent renewals beyond reasonable timeframe

Adhere to Principle of Least Privilege

- Only give tokens access to paths required for business function
- Separate policies for applications and users
- Limit use of * and + in policies, where possible
- Templated policies can help with policy creation and maintenance

Perform Regular Backups

- Backup configuration files and directories
- Automate Vault backup using Consul snapshots or equivalent depending on the storage backend
- Regularly test backups to ensure functionality

Integrate to Existing Identity Providers

- Use your existing IdP to provide access to users
- If/when users leave, they immediately lose access to Vault
- The fewer places a user has credentials, the better
- Using locally defined credentials is an administrative burden

End of
Security Hardening Section