HashiCorp Vault

5 Best Practices for Secrets Management

How do you take the best secrets management approach and go from a sprawl of hard-coded and plaintext credentials lying around, to a well-managed secrets management platform? Here are five best practices to use as milestones on your journey.

1. Central secrets control plane

Benefits: *Reduces errors, speeds up debugging and auditing, simplifies security management*

Get your secrets into one central tool or platform. Some might see it as counterintuitive to have one place where attackers know they can find your secrets, but it's actually much safer to harden one cluster or a series of redundant clusters instead of having them stored in several different places with varying styles of management.

A surprising number of organizations don't even have a central secrets management plan. Some teams might be using a secrets manager, or rolling their own, but others might have secrets sitting in version control systems like GitHub. They might have them in Excel spreadsheets or even a Post-it note under their keyboard. While most of us know those are not good places to keep secrets, there are plenty of junior developers or other engineers with low security knowledge that are leaving secrets like AWS keys in plaintext and publishing them on public or private repositories — and attackers can often find them in both.

This horrific state of affairs is called secret sprawl, and just getting some sort of purpose-built secrets management tool and org-wide process is the least you can do to begin remedying the situation.

But let's say you have some loose secrets management requirements and a few teams are managing secrets with one tool, and another team is using a different tool, and yet another team built their own tool. What's wrong with that?

This is an age-old trade-off: Flexibility vs standardization.

Flexibility is great sometimes. Let developers use the programming languages, frameworks, databases, and cloud services they want. Some of you reading this know you need a high level of freedom and autonomy to be motivated and creative.

But having **core stability** in some cases is also great. If teams end up going so far in different directions, costs can go out of control, security holes can emerge all over the organization, and no one can debug issues because there are five different places the logs could be. Operators and engineers understand this as well and gladly welcome standardization in many areas.

Secrets management is one of those areas.

There's no way to build solid governance, auditing, and security around organizational access secrets if you don't centralize your management through one control plane. There's no way to easily scale security practices either. You'll be constantly reinventing the security wheel in different corners of your company.

Secrets management is just one more reason why the concept of platform teams has gained so much momentum in the past few years. Centralized secrets management needs to be a top priority for every medium-to-large organization's platform strategy.

If you don't have a platform team or anything like one, find your senior-level engineers with the most security and automation expertise and see what tools they should look into. Have them meet with other teams and build an action plan for implementing a centralized secrets management platform. Then standardize secrets management workflows that even a first-day junior developer can't mess up.

2. Access control lists (ACLs)

Benefit: *Limits lateral movement through your systems*

For those who have already bought into best practice #1, you might start with a secrets manager, or you might try and centralize secrets in a simple database like a key-value store. While centralizing is an important first step, this isn't a very secure option if you don't have access control features around that central store.

This brings us to an overall best practice that runs parallel to these five practices: Use a purpose-built secrets manager. Don't try to roll your own.

Security experts have built these tools using years of experience and, depending on the age of the tool, have poured hundreds or thousands of hours into building them. If it's an open source secrets manager, even better — you're harnessing the expertise of security engineers working at dozens of industries and companies.

Unless your company's product is a secrets manager, you shouldn't spend time building one. That time could be better spent on building your core product. Your platform team is there to pick tools and build a workflow around them, not to build a ton of tools themselves (especially ones that demand a high level of domain expertise like secrets managers).

Bottom line: Use a secrets manager. And take advantage of its ACL capabilities. These are critical capabilities for secrets management because your developers and their applications need boundaries around what they can and cannot access. If you get lazy with ACLs, you're repeating the same mistakes from the secret sprawl stage. Giving blanket or large-scale access out of convenience will make your systems just as insecure as they were before secrets management.

Platform teams will need to build a workflow and configure access controls that allow speed but also adhere to the principle of least privilege.

Really good platform teams can build a secrets management workflow so subtle and streamlined that their developers barely even know they're using it.

3. Dynamic secrets

Benefit: *Reduces time of breach*

Typically the first type of secrets used in an organization that's new to secrets management is static secrets. These are secrets that aren't rotated unless someone manually goes into the secrets manager and rotates it. There are plenty of use cases where it's safe and acceptable to use static, long-lived secrets. But in many instances, teams will be content and won't transition to a mainly automated secret generation and rotation strategy.

That's bad, because most of our secrets need to be dynamic. Dynamic secrets are short-lived, temporary credentials that are generated by the secrets manager (if they have an integration with the target system) and given to individuals or entities with an expiration usually set by the platform team.

With dynamic secrets, the workflow is: I am a human or a machine, and I need access to a database, or to AWS, or to Microsoft Azure. I ask the secrets manager to generate a secret and the secrets manager uses a secrets engine or integration with AWS or Azure to create one. The machines can initiate this workflow thousands of times a day automatically as part of their daily operation.

A secrets manager should also be able to revoke secrets at any time, whether there's been a breach or a machine has been decommissioned.

4. Encryption as a service

Benefit: *Prevents breaches*

Encryption functions across IT are not inherently a part of secrets management, however as you begin to adopt best practices 1-3, you might notice there's an obvious tie-in: Encryption key management.

Imagine a malicious actor gets onto your network and they grab data being passed around between applications, or they get the keys to a database. Now imagine your encryption keys are, like your credentials and other secrets, centralized in your secrets manager so you can provide proper access control around who's gaining access to those keys. And your secrets manager also has the ability to encrypt data during transit, as a service.

In this scenario, you can revoke those database keys the attacker got. And that data they grabbed being passed between applications? It's encrypted and unreadable to them. Now you're managing encryption and secrets in one central platform and you can start to manage and audit all of your encryption actions in one place, achieving a well-known security best practice of encrypting everything at rest and in-transit.

There's a saying in IT: Don't roll your own crypto. Having a workflow built into your secrets manager encryption service that easily and automatically stores encryption keys and can also act as aa pass-through for users that want to encrypt application data but not necessarily store them is a huge win. This is particularly useful for web applications that don't need to store the data over time, such as single-page web apps or applications that use different data stores.

With encryption as a service, developers can offload cryptographic operations rather than having to solve them on their own. And a secrets manager is likely built on the best available encryption methods and tools.

Building your own certificate authority (CA) by automating public key infrastructure (PKI) is also something many teams are doing with secrets managers. It's definitely something you should look into.

5. Auditing

Benefit: *Better understanding of your security posture, breach detection*

Nothing's ever truly managed until you have full visibility into what's happening. You need to know who has access to what through your ACLs. You need to be able to trace every secret from its generation to its revocation. You need to have logs so you can hook your real-time secrets data into other monitoring tools that can analyze patterns and send alerts. In essence, your secrets management platform needs to have comprehensive observability features and insights.

Secrets monitoring is much easier to achieve once you've implemented dynamic secrets automation. A good secrets manager will have detailed audit logs every time a user is authenticated to retrieve a secret. You should also be able to see when the secret was actually used in the underlying subsystem. For encryption as a service, you want to be able to see who is accessing an encrypt or decrypt operation. This gives teams complete visibility into everything that's going on with their secrets.

Having good secrets auditing is critical for two main reasons. One, your security teams and your entire organization now have metrics they can use to know the state of security in their systems at any given time. They can also use data from your secrets manager to prove compliance with certain cybersecurity regulations.

Two, you have the visibility you need in order to know when to take action. Whether you need to manually rotate some compromised static secrets or automatically and instantly revoke a large group of dynamic secrets, auditing will help you determine when that needs to happen. It can help you detect a breach before it becomes harmful.

A secrets manager with the broadest adoption

If you don't already have a secrets manager being used by your teams that has these best practices built in, you'll obviously want to do your own research on them. Just please don't try to build it all on your own.

It's also recommended that you don't use Kubernetes Secrets in non-trivial security use cases. Kubernetes Secret data is encoded in the base64 format and stored as plaintext in its etcd key-value store. This default method of storage and format make Kubernetes Secrets easy prey for attackers.

If you're just looking for the secrets manager with the broadest adoption across a variety of industries, along with one of the largest, most active open source communities, check out HashiCorp Vault.