Proactive guide

The cost of secret sprawl and how to stop it

Understand the risk, quantify the impact and act with proven frameworks



HashiCorp
an IBM Company

Contents

00 →
Executive summary

01 →
Understanding secret sprawl

02 →
Discovering the hidden risks behind secret sprawl

03 →
Quantifying the cost of secret sprawl

04 →
Building the business case for a secrets detection product

05 →
Learning why traditional solutions fall short

06 →
Regaining control with a unified strategy

07 →
Building a secrets hygiene program

08 →
Evaluating secrets discovery tools

09 →
Conclusion: Securing developer workflows at scale

As engineering velocity increases, security practices often struggle to keep pace. API keys, tokens and credentials now move at a more dynamic speed found in modern hybrid and multicloud environments. The result is secret sprawl: the unchecked proliferation of sensitive data across application code, infrastructure configurations and collaboration tools. Left unaddressed, secret sprawl exposes organizations to credential leaks, costly incidents, audit failures and operational inefficiency.

This guide is designed to help security, platform and developer teams understand and quantify the problem and take action. You'll learn how to reduce exposure, prioritize remediation and embed secrets hygiene into everyday workflows, without slowing your teams down.



01

Understanding secret sprawl

Secret sprawl is a structural byproduct of how modern software is built. In the past, credentials were managed manually with unsophisticated, decentralized processes. Before the cloud era, IT operations teams could tightly control access to infrastructure and manage credentials themselves, but the ticketing process was slow and not scalable.

When organizations began building in the cloud, secrets ended up everywhere. They could be going the following places:

- Copied into test scripts
- Pushed into GitHub repositories (repos)
- Embedded in Continuous Integration and Continuous Delivery (CI/CD) pipelines
- Shared in Jira tickets
- Put in Slack threads
- Scattered across configuration (config) files
- Exposed in logs and countless other hiding places

Teams are shipping faster than ever, often without shared tooling or policies to manage credentials end to end. Developers, security engineers, IT and platform teams all touch secrets, but at many organizations, no one fully owns the lifecycle. And without consistent scanning or hygiene practices, exposures remain invisible until it’s too late.

Understanding secret sprawl means understanding that it’s not just about poor development or security practices. It’s about modern infrastructure evolving faster than previous practices can maintain.



02

Discovering the hidden risks behind secret sprawl



Every exposed credential is a potential entry point for attackers. And when a secret is compromised, it enables unauthorized access to your systems. Once inside, attackers can move laterally across services, escalating access, querying sensitive data or spinning up compute resources. What begins as a leaked token in GitHub can cascade into a full-scale breach that can go undetected for months or years.

The problem is compounded by where secrets now live. No longer are they limited to config files or source code. They’re found in CI/CD pipelines, support tickets and collaboration platforms such as Slack, Jira or Confluence. Without visibility into all of these systems, you’re operating with blind spots. Even low-privilege credentials can lead to problems if left unchecked.

USD 4.44M

Global average cost of a data breach in 2025[1]

246 days

Average time to identify and contain credential-based breaches[1]

USD 4.67M

Average cost of breaches involving stolen or compromised credentials[1]

03

Quantifying the cost of secret sprawl

Secret sprawl is expensive, but not always in the ways teams expect. There's the obvious cost of a breach, but the hidden costs include the following items:

- Developer time
- Management overhead
- Firefighting
- Brand reputational risk
- Security team time

To fully understand the impact, it helps to break the cost of secret sprawl into three dimensions.

Exposure risk
Secrets that go unmanaged often sit in sensitive locations, such as production systems, administrative tools and customer databases. Without early detection, they remain exploitable for months. On average, it takes 246 days to identify and contain a credential-based incident.

Financial risk
Credential-related breaches carry a premium. The average cost of a breach is USD 4.44 million, while the average cost involving credentials is USD 4.67 million. That's before factoring in regulatory penalties.

For example, violations of General Data Protection Regulation (GDPR) for the European Union can cost up to 4% of global annual revenue. Add any other regulations your organization has to follow—such as the Health Insurance Portability and Accountability Act (HIPAA) and the Federal Information Security Modernization Act (FISMA) for the United States and so on—and that number can grow fast.

But not all financial damage comes from breaches. Erosion of brand trust has these long-term consequences:

- Delayed deals
- Longer sales cycles
- Customer churn
- Rising acquisition costs

Operational risk
Developers lose hours hunting down exposed secrets, rotating credentials across environments and rebuilding broken pipelines. CI/CD jobs fail due to invalid tokens. Security teams chase false positives or file tickets for secrets they can't trace. Over time, these events add up to missed deadlines, distracted engineers and lost velocity.

Canva shared how a single leaked secret triggered days of downtime across multiple teams, even without a breach. The cleanup effort consumed engineering resources that should've been focused on product work.

Read more →

04

Building the business case for a secrets detection product

There are two first steps you should take to avoid the costs of secret sprawl. First, take a full inventory of all your organization's secrets. Next, implement secrets detection gates that stop exposed secrets before they reach production. Instead of responding after incidents occur, secrets detection tools scan code, pipelines and systems in real time and catch exposed credentials early, before they lead to breaches or outages.

The challenge for this preventive measure is often budget and buy-in. This framework helps quantify the cost of inaction, estimate ROI and align stakeholders around the value of secrets detection. Often developers have to spend time chasing false positives or filing tickets for secrets they can't trace. Over time, this adds up to missed deadlines, distracted engineers and lost velocity.



Risk dimension	Metric	Example benchmarks and scenarios[2]	Cost impact
Exposure	Number of unmanaged secrets Time to detect (TTD) and time to remediate (TTR) % of secrets with production-level access.	Estimated to be 1,000s across large organizations[2] 246 average days to contain a breach 20%-40% in high-risk categories[2]	Increased attack surface and lateral movement risk Longer TTD = increased chance of exploitation High-priority secrets multiply breach cost impact
Financial	Average cost of a breach involving credentials. Regulatory fines for GDPR, HIPAA and so on.	USD 4.67M average cost of a data breach involving compromised credentials	Multimillion-dollar fines for noncompliance
Operational	Developer time lost to manual secret handling and exposure investigation. Security team hours triaging false positives or managing manual detection	Developer time spent and salary costs: 3 hours per week × 10 developers × USD 120 per hour = USD 3,600 per week = USD 172K per year[2] 80% of alerts are false positives, with average time to evaluate approximately 25 minutes[2]	High costs associated with wasted developer productivity annually

05

Learning why traditional solutions fall short

Most tools designed to manage secrets were built for an era where IT tightly controlled secrets. Today, secrets move fast across teams, tools and environments. Earlier solutions can't keep up with the changes.



Static code scanners fall short.
They miss secrets shared in Slack, Jira, Confluence or CI/CD systems. Even when they do find something, the results are noisy, with false positives cluttering dashboards.

The cost:
Wasted security team hours and buried real threats can later result inexpensive breaches.



Manual audits don't scale.
Reviewing code line by line or running one-off scripts isn't feasible in modern pipelines. Teams are pushing code multiple times a day across dozens of repos. Secrets get missed or flagged too late to be useful.

The cost:
What results is remediation after deployment and time-consuming incident investigations.



Developer experience is an afterthought.
Tools that generate alerts but don't integrate with existing workflows—such as GitHub pull requests or integrated development environment (IDE) alerts—are easily ignored.

The cost:
Security teams open Jira tickets that go unaddressed. Issues pile up and remediation lags, costing developer hours and increasing organizational risk.



None of this activity occurs in real time.
If a secret leaks in a public repo, you have only minutes to act, not hours. Without real-time detection and automated response, the window between exposure and exploitation stays dangerously wide.

The cost:
Every hour of delayed response could result in compounding resource theft costs, regulatory penalties and reputational damage.

06

Regaining control with a unified strategy

Secret sprawl is an outcome of disconnected systems, fragmented ownership and reactive processes. Regaining control means moving away from isolated fixes and toward a unified, proactive strategy for managing secrets across the organization.

At the foundation of this strategy is centralized secrets management. It creates a single source of truth for sensitive credentials while layering in dynamic secrets generation, automated rotation and policy-based access controls. These features turn a slow, static secrets management process into a governed, auditable one.

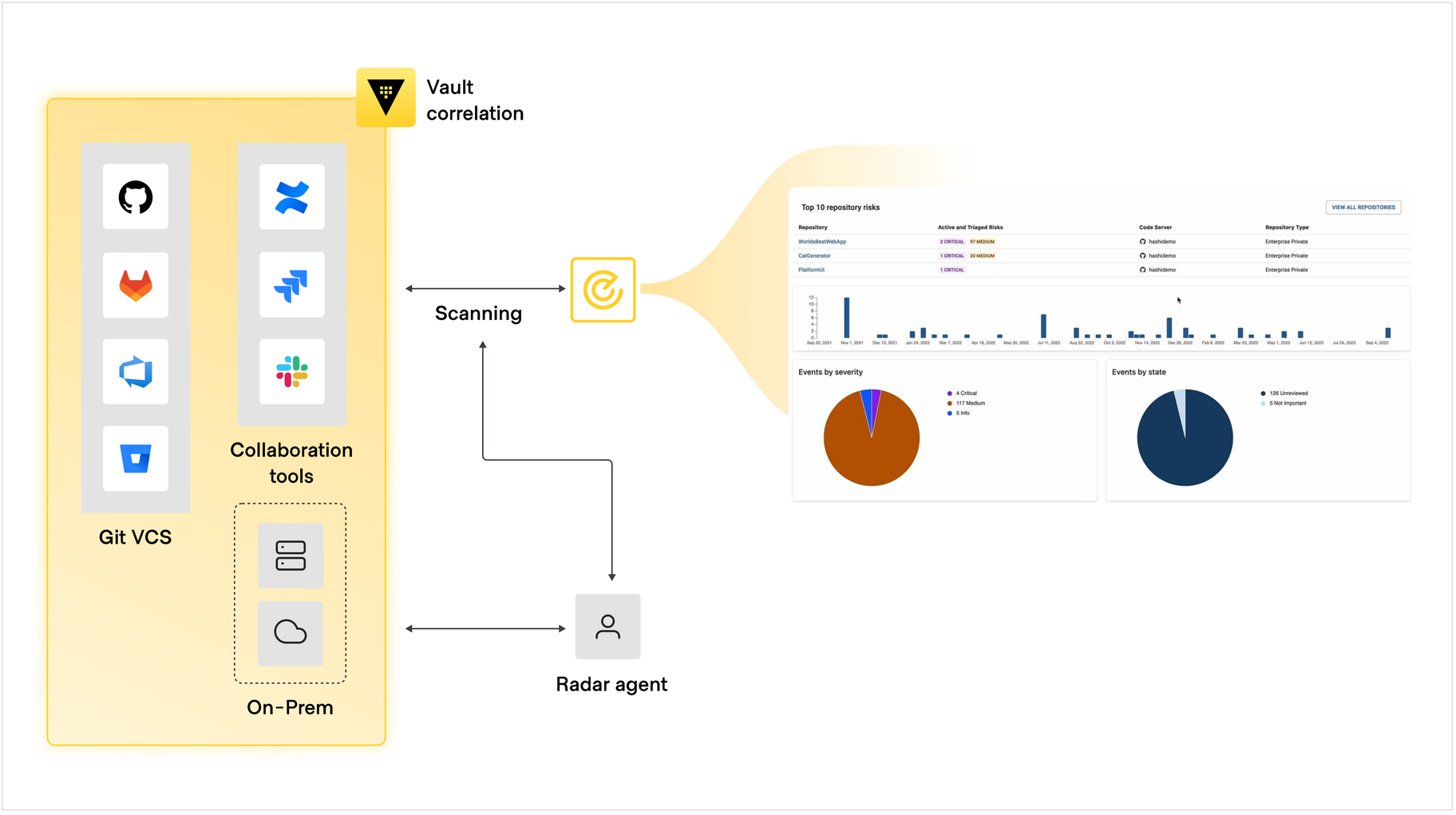
Teams should also aim to regain control over secrets at preproduction stages. Unmanaged secrets detection needs to happen in the tools developers use every day when creating and sharing secrets. Embedding detection into CI/CD pipelines and IDE plug-ins enables real-time detection, while scanning collaboration tools such as Jira, Slack and Confluence reveals hidden risks that often slip through the cracks.

Finally, a strong remediation framework will bring your strategy together. Remediation should start with context and ownership. When ownership is clear and context is provided, remediation can happen within developer workflows organically.

Starbucks uses Vault to manage short-lived, dynamic secrets across 100,000-plus edge devices, with detection, rotation and authentication built into its infrastructure. This implementation simplified automation and removed unnecessary operational overhead and risk.

Read more →

Solutions such as Vault Radar can correlate leaked secrets findings with a central secrets manager, such as Vault. This connection can provide greater context and enable direct remediation into Vault for centralized management.



07

Building a secrets hygiene program

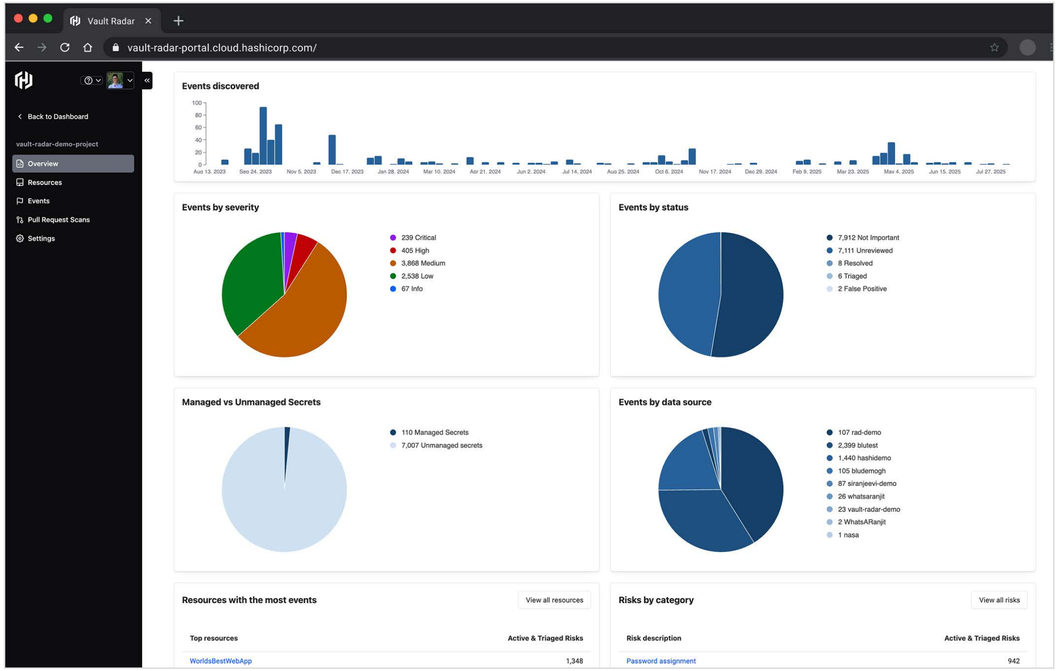
When you've regained control through centralized management and integrated detection, the next step is ensuring that control lasts. A successful secrets hygiene program embeds best practices directly into day-to-day development and security workflows. It's proactive, continuous and measurable.

Start by establishing the following baselines:

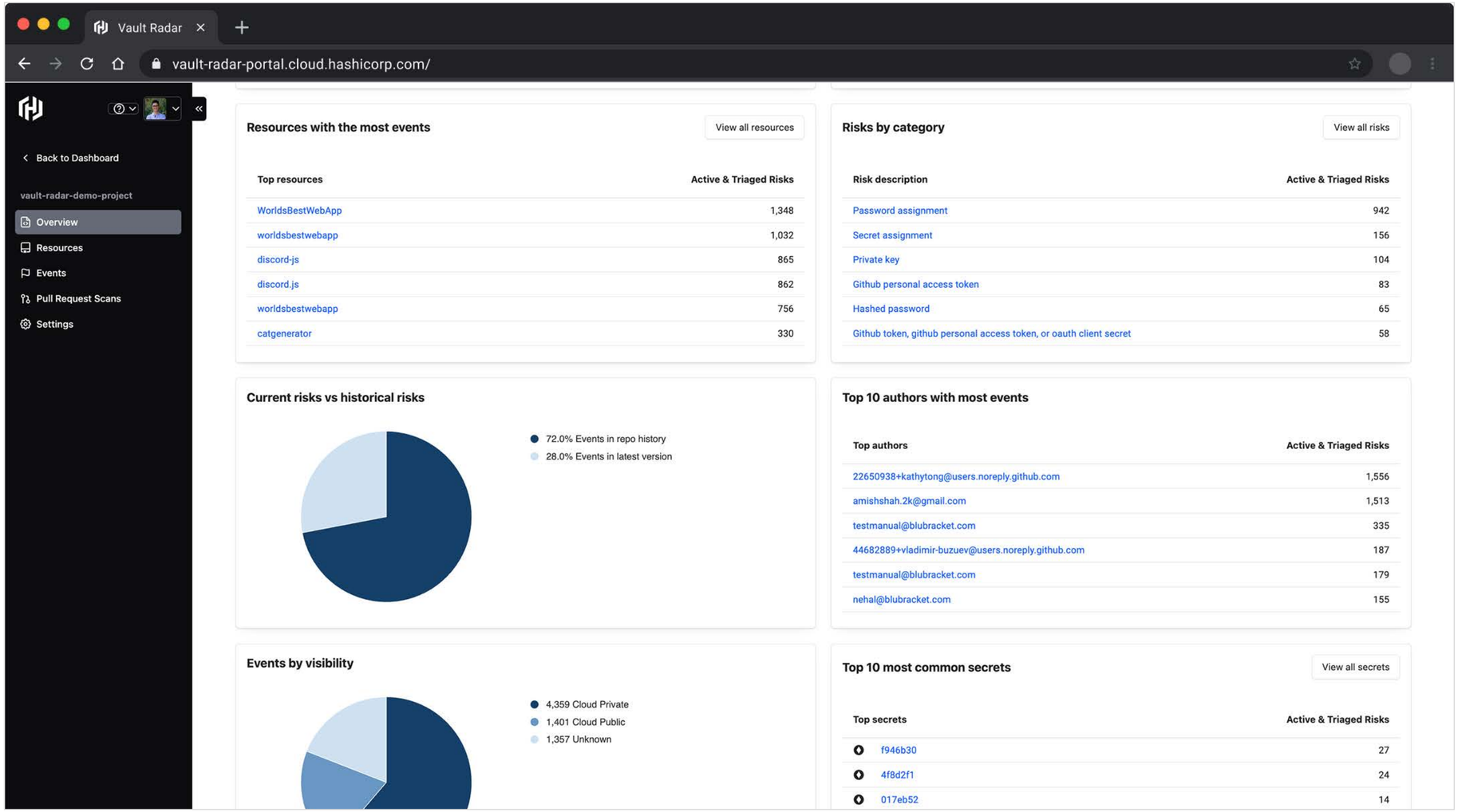
- How many secrets are unmanaged?
- How often are exposures occurring?
- How long does it take to remediate issues?
- What level of risk do existing unmanaged secrets pose?

These insights will help you understand the current level of risk and expose systemic gaps that one-time remediations can't solve. These systemic gaps often lead to recurring security incidents, compliance fees and unnecessary operational spend.

Remediation within the program should be structured and prioritized by the secret's risk level. For example, high-impact exposures, such as production credentials with broad access, require immediate attention. By assigning severity scores to unmanaged secrets and setting remediation SLAs based on impact, teams can focus engineering time where it matters most.



Platform, security and engineering teams should have clearly defined responsibilities, supported by shared dashboards, reporting loops and outcome-orientated metrics. Implement these changes not only to improve security posture but also to reduce incident response costs, rework hours and compliance risk.



08

Evaluating secrets discovery tools

As organizations mature their approach to secret hygiene, choosing the right tools is critical. A secrets detection product should be built for scale, automation and governance. Look for the following features.



Data source coverage
Scan across code repositories, cloud environments and collaboration platforms such as Jira, Slack and GitHub.



Activeness checks
Verify whether a secret is still active by integrating with systems such as Amazon Web Services, Microsoft Azure, Google Cloud, GitHub or JSON Web Tokens (JWT) lifecycles.



Real-time detection
Catch exposed secrets before code is merged into version control and production with features such as IDE plug-ins or pre-commit webhooks.



Entropy-based accuracy
Use advanced entropy checks to detect high-risk patterns while minimizing noise.



Contextual remediation workflows
Surface issues with relevant context and guide the right owner to resolve them quickly.



Workflow integrations
Automatically create tickets in tools such as Jira and trigger secrets rotation in platforms such as Vault, closing the loop from detection to remediation.

The right tools help teams act on, prioritize and prevent secrets from reoccurring.

09

Conclusion: Securing developer workflows at scale

When secret sprawl goes unchecked, it silently erodes security, developer velocity and even customer trust. But with a unified strategy, the right tools and a disciplined hygiene program, organizations can fundamentally transform their security posture.

If you're ready to reduce risk, accelerate development and avoid the high cost of secrets exposure, try Vault Radar and see how it fits into your workflow.

Many enterprises already use Vault to help teams manage secrets at scale by serving as a centralized, policy-driven system of record. It enables secure secret creation, access controls, dynamic rotation and auditable workflows.

By pairing Vault Radar and Vault together in your security platform, you can extend secrets control with proactive secrets detection across code, pipelines and collaboration tools. Together, Vault and Vault Radar create an end-to-end secrets management lifecycle, detecting exposed credentials early and seamlessly bringing them under centralized management.

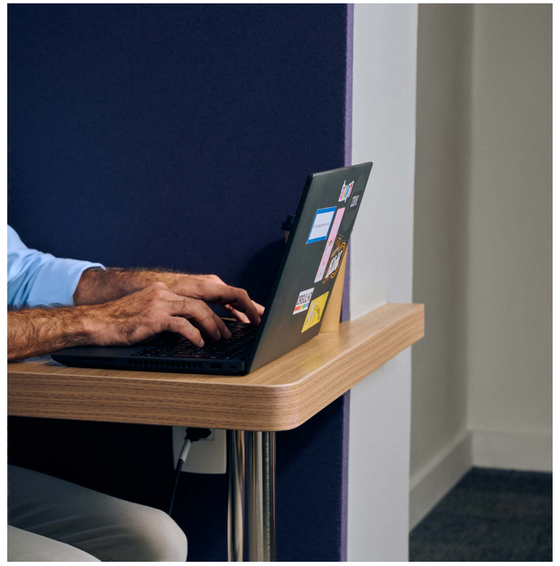
Sign up for our webinar series to learn how HashiCorp® and IBM® can help you reduce risk, enforce continuous compliance and implement zero trust security.

About IBM
IBM is a leading global hybrid cloud, AI and business services provider, helping clients in more than 175 countries capitalize on insights from their data, streamline business processes, reduce costs and gain the competitive edge in their industries.

All of it is backed by IBM's legendary commitment to trust, transparency, responsibility, inclusivity and service. For more information, visit ibm.com.

Learn more about advancing your security posture: visit ibm.com/security.

Join the conversation in the IBM Security Community.



Notes

1. Cost of a Data Breach Report, IBM, 2025.

2. Based on customer data from HashiCorp, an IBM company. Benchmarks are illustrative examples based on industry research and may vary by organization.

© Copyright IBM Corporation 2026

IBM and the IBM logo are trademarks or registered trademarks of International Business Machines Corporation, in the United States and/or other countries. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on ibm.com/legal/copytrade.

HashiCorp is a trademarks or registered trademark of HashiCorp, an IBM Company.

Microsoft is a trademark of Microsoft Corporation in the United States, other countries, or both.

This document is current as of the initial date of publication and may be changed by IBM at any time.

Not all offerings are available in every country in which IBM operates.

It is the user's responsibility to verify the operation of any non-IBM products or programs with IBM products and programs. IBM is not responsible for non-IBM products and programs.

THE INFORMATION IN THIS DOCUMENT IS PROVIDED "AS IS" WITHOUT ANY WARRANTY, EXPRESS OR IMPLIED, INCLUDING WITHOUT ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND ANY WARRANTY OR CONDITION OF NON-INFRINGEMENT. IBM products are warranted according to the terms and conditions of the agreements under which they are provided.

The client is responsible for ensuring compliance with all applicable laws and regulations. IBM does not provide legal advice nor represent or warrant that its services or products will ensure that the client is compliant with any law or regulation.

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.