Secret Scanning

A Key to Your Cybersecurity Strategy

WRITTEN BY GITHUB WITH ♥

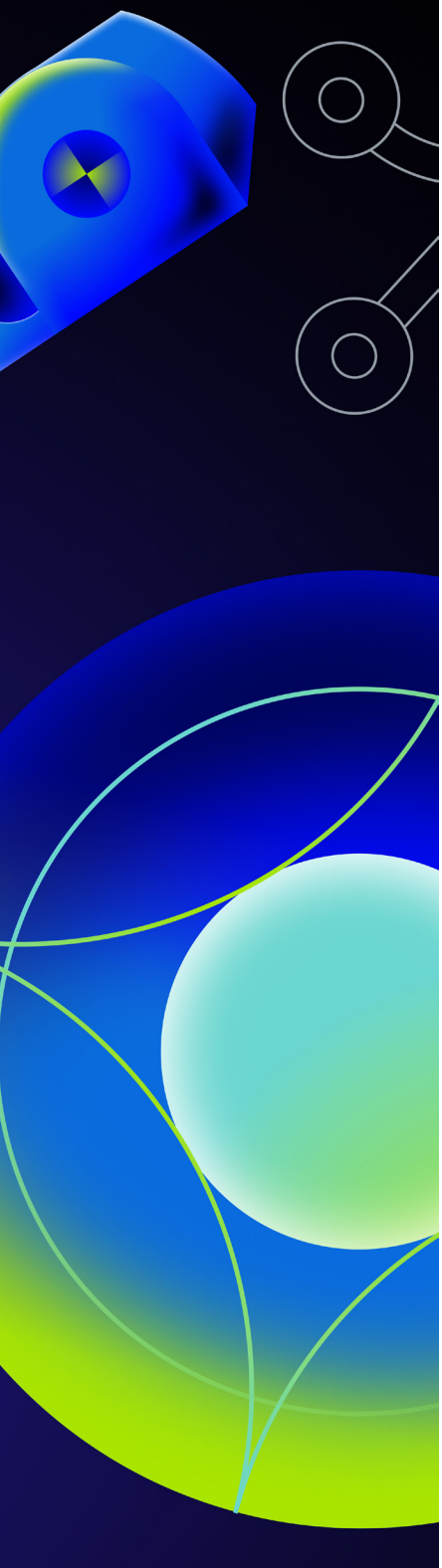


Table of Contents

Introduction

In today's digital era, securing access to systems and sensitive information is more important than ever. This is especially difficult given the scale and interconnected nature of software and the increasing digital footprint of organizations of various sizes and industries. As a result, malicious actors now have more ways to gain unauthorized access and launch attacks.

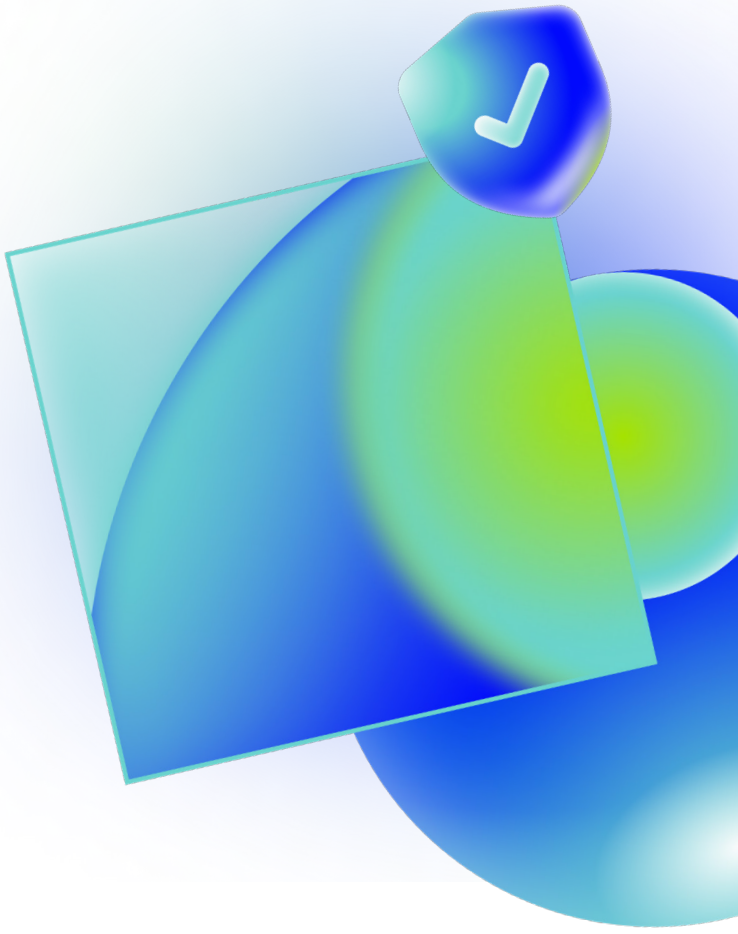
Secrets that give people access are popular targets for bad actors and the number of secret leaks is increasing. A contributing factor is the constant drive for developers to ship code faster and more efficiently. Another contributing factor is that the attack surface of secrets is growing as software continues to expand.

In 2024, developers across GitHub used secret scanning to detect more than 39 million secret leaks.

– GitHub Octoverse 2024[1]

1: GitHub, "Octoverse: AI leads Python to top language as the number of global developers surges", Oct 2024

Companies need a strategy for protecting their secrets to prevent unauthorized access and ensure the security of their code and software. If compromised, secret leaks can lead to more devastating breaches involving proprietary, financial, or customer data. In this ebook, we'll review secrets detection and management, and highlight why it should be a core part of your application security program.



What are secrets?

Secrets are the sensitive information used to authenticate, authorize, or encrypt data within an application or system. These digital keys or credentials are crucial for secure operations but can pose significant risks if exposed. Secrets include API keys, passwords, tokens, encryption keys, SSH keys, and other confidential strings granting access to various resources or services.

Secrets are essential for modern development workflows, enabling applications to securely communicate with databases, third-party APIs, cloud services, and other critical infrastructure. Here are some examples:

- A web application using an **API key** to authenticate requests to a payment gateway
- A **service provider access key** hard-coded in a cloud deployment script
- A database connection string with login **credentials** in an application's configuration file

While these secrets are vital for functionality, their sensitive nature means you should never expose them in public repositories, logs, or any other unsecured locations.

The challenge with secrets lies in their dual nature: they must be accessible to the systems that need them while remaining hidden from unauthorized users. This balancing act becomes increasingly complex in distributed development environments where code is shared, reviewed, and deployed across multiple platforms and teams. While there are technologies for managing and encrypting keys, deploying them across all development use cases is difficult. Furthermore, there is no universal standard to adhere to.

As organizations scale their development efforts and adopt more cloud-based services, the number of secrets in circulation grows, amplifying the potential impact of any leak or breach. According to the IBM Cost of Data Breach Report 2024[2], 40% of breaches involved data stored across multiple environments including public cloud, private cloud, and on-premises. This is where robust secret management practices, including secret detection and remediation, become crucial for maintaining security and compliance in the software development lifecycle.

2: IBM, Cost of a Data Breach Report 2024

The costs of secret leaks

Secrets are usually the first step for bad actors to launch broader attacks. Leaked credentials and API keys can be used to gain direct access to personal information, financial data, or intellectual property. For example in 2023, one of the largest mobile carriers in the United States, saw a single compromised API lead to the exposure of sensitive data of over 37 million customers.

In 2024, the average total cost of a data breach was $4.88M, up 10% from the year prior

– IBM Cost of Data Breach 2024[3]

Once attackers access one system, they can move to other areas of an organization's network, often undetected. The costs to organizations can be immense.

- **Financial penalties:** Breaches from leaked secrets can be subject to regulatory fines and legal fees. For example, in 2019 the FTC fined a large American credit bureau $700M for a data breach affecting 147 million people.
- **Remediation and incident response:** Fixing the problem is not straightforward and can be time-consuming. In larger incidents, a 3rd party is often called in to ensure the breach is contained and resolved.
- **Operational downtime:** Organizations may be forced to pause operations or incur downtime while addressing the issue. This is especially prominent in healthcare, where leaked secrets have led to ransomware attacks forcing hospitals to cease operations for multiple days or even weeks.
- **Reputational damage:** And then there's what could be the most devastating, but also the most difficult to measure: damage to an organization's reputation. If the customers lose trust, it can take years to win them back, if at all.

Organizations need a strategy for securing their growing secret footprint to lower their cybersecurity risk. That starts with understanding where and when secret leaks may occur, and what attacks may result.

3: IBM, Cost of a Data Breach Report 2024

How secrets can be exploited

When a secret is exposed, it often serves as a critical entry point for attackers to initiate a more extensive and damaging breach. Here are some of the tactics a malicious actor can leverage once they possess a leaked secret:

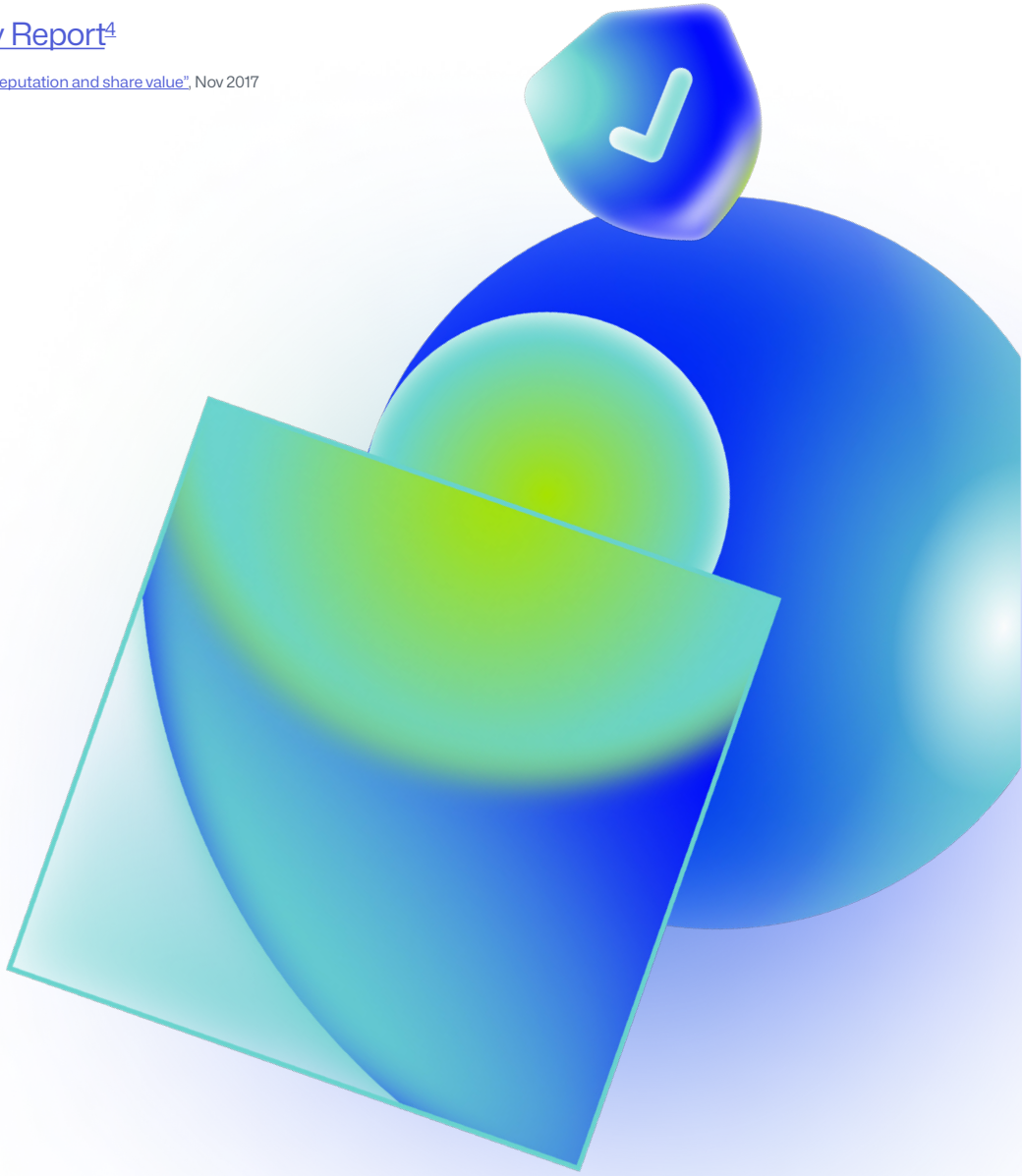
- **Initial Access:** Attackers leverage exposed secrets to gain legitimate credentials to access a system or service. This initial foothold bypasses traditional security measures since the attacker looks like an authorized user.
- **Reconnaissance:** Once inside, attackers can use the compromised secret to gather information about the target environment. They may access logs, configuration files, and databases, map the network architecture, and identify additional targets.
- **Privilege Escalation:** With knowledge of the system, attackers often seek to elevate their privileges. They might exploit vulnerabilities or use the exposed secret to access more sensitive areas of the network, potentially gaining administrative rights.
- **Lateral Movement:** Using the elevated privileges, attackers may move laterally across the network, compromising additional systems and services. This expansion increases their reach and the potential impact of the breach.
- **Data Theft:** As the attack progresses, sensitive data becomes a prime target. Attackers may use the compromised systems to locate and extract valuable information, such as customer data, financial records, or intellectual property.
- **Persistence:** To maintain long-term access, attackers might create backdoors or compromise additional accounts. This enables them to return even if the initial exposed secret is discovered and revoked.
- **Supply Chain Attack:** In some cases, attackers may use the compromised environment to target connected third-party vendors or customers, turning a single breach into a far-reaching supply chain attack.
- **Infrastructure-as-Code (IaC) Exploitation:** If the exposed secret grants access to cloud resources or CI/CD pipelines, attackers can manipulate IAC configurations, potentially compromising entire cloud environments or development processes.
- **Ransomware:** With broad access to the network, attackers may deploy ransomware, encrypting critical data and systems across the organization, leading to operational disruption and potential financial losses.

The cascading nature of these attacks underscores the critical importance of protecting secrets. A single exposed credential can lead to a comprehensive breach, potentially costing millions in addition to untold damage to an organization's reputation.

65% of data breach victims lost trust in an organization as a result of a breach.

– Ponemon Sullivan Privacy Report[4]

4: Ponenmon Sullivan, "How data breaches affect reputation and share value", Nov 2017



Secrets management approaches

When considering a strategy for secrets management, you need to meet developers where they are. Specifically developer-native tooling with security built in. If developers experience friction and context switching, they can lose trust in the tooling.

Next, understand your development workflows to determine where code is created. That should reveal potential areas where secret leaks may occur. There are a few different approaches you can take when implementing your secret management program:

- **Reactive Approach:** This approach focuses on responding to incidents after they occur. Reactive strategies include incident response processes that quickly address compromised secrets, including rotating and revoking secrets in the event of a breach. Post-incident analysis helps the organization learn from vulnerabilities to prevent future issues. While easier to implement initially, this approach leaves room for exposure between breach detection and resolution.
- **Proactive approach:** A proactive approach to secrets management is centered on prevention, aiming to protect sensitive information before issues arise. This approach includes automated rotation of secrets, ensuring they're regularly updated to reduce potential exposure risks. Encryption by default keeps secrets safe at rest and in transit, while role-based access control (RBAC) limits access strictly to necessary personnel. Additionally, proactive monitoring helps detect abnormal access patterns early, enhancing compliance and security. A proactive strategy offers a highly controlled environment, significantly reducing the risk of a breach.
- **Hybrid approach:** A hybrid approach combines the strengths of both proactive and reactive methods, offering a balanced strategy for secrets management. This approach layers security by implementing regular secret rotation and access control while maintaining reactive incident response protocols for added agility. Continuous monitoring and auditing are also key components, allowing the organization to detect issues in real-time and respond rapidly when needed. The hybrid approach provides a flexible, robust security posture, enabling organizations to proactively prevent issues while remaining prepared to react to unexpected events.

Key principles of secrets management

Preventing secret leaks is no easy task and requires trust and teamwork between security teams and developers. Here are some recommendations when you begin building out your secret management strategy.

- **Education:** Developers, engineers, and all other roles involved with DevSecOps should understand the importance of secret management. Training and awareness should be paramount as many secrets are exposed due to human error. Read more about best practices for securing secrets at Microsoft Learn[5].
- **Utilize secret management tools:** Dedicated secret management services like Azure Key Vault, HashiCorp Vault, and Google Secrets Manager[6] can help securely store and manage your most sensitive secrets. They also provide an additional layer of security.
- **Conduct regular, automated scanning:** Regular secret scanning of where your code lives can identify where to focus your efforts regarding education and remediation. Your organization will also be able to identify and quantify areas of secret exposure.
- Regular secret rotation: Implement a policy to rotate secrets regularly, reducing the risk of exposure over time.
- **Apply the principle of least privilege:** Limit access to secrets only to those who need it and apply "least privilege" concepts to restrict the access that each secret grants.
- **Monitor and audit access:** Regularly monitor access to secrets and audit logs to detect attempts at unauthorized access.
- **Scan beyond code:** Extend your scanning to include logs, binaries, and other data in the codebase that might contain secrets.
- **Leverage built-in solutions:** Integrated secret detection capabilities can lead to higher adoption rates while preventing tool fatigue and productivity losses.

By implementing these suggestions, organizations can significantly reduce the risk of secret leaks and improve their security posture. Eventually, your goal should be to have a detailed plan of how your organization will approach secrets management and get alignment from DevOps stakeholders on your plan. The Open Worldwide Application Security Project (OWASP) Secrets Management Cheat Sheet[7] is a resource to reference as you build out your best practices.

5: Microsoft, "Best practices for protecting secrets", Aug 2024

6: Google "Store API keys, passwords, certificates, and sensitive data"

7: OWASP Cheat Sheet Series "Secrets Management Cheat Sheet"

Getting started on secrets management

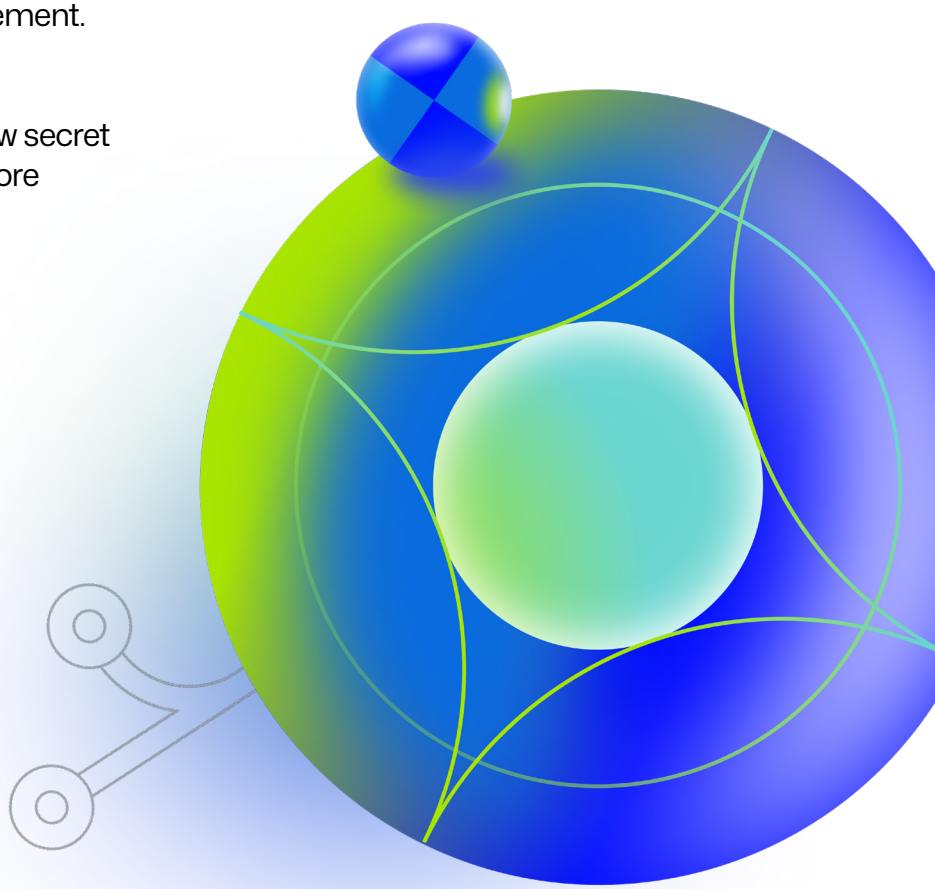
Hopefully, by now it's clear that a secrets management strategy is essential to reducing cyber risk in your organization and scaling your DevSecOps practice. But where do you start?

1. First, identify the areas of highest risk for secret leaks. This can include third-party penetration testing, creating an asset inventory, and interviewing various stakeholders, especially those with access to secrets.

2. Next, educate and increase security awareness around secrets throughout the organization. Remember secret leaks often include a human element. Formal training and awareness campaigns can reduce risk by educating employees about how secret exposures can lead to larger, more devastating attacks.

3. Finally, when choosing a secrets management solution, ensure it works with your current development process. Native, embedded tooling is the clearest and most effective path to setting you up for success with your DevSecOps vision.

With these principles in mind, you can effectively build trust and communicate the importance of securing the secrets in your organization and drive meaningful change.



Securing secrets with GitHub

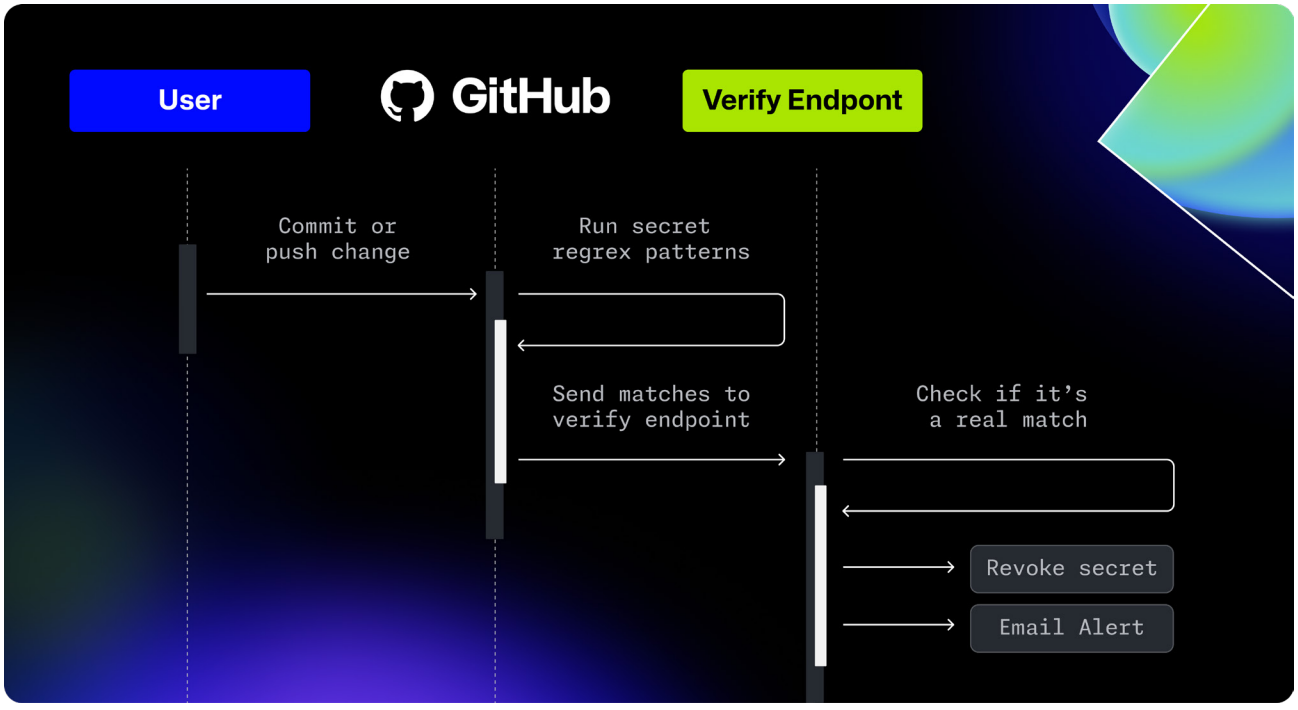
As the world's leading AI-powered developer platform and home of open source, GitHub wants to enable customers and OSS contributors to stay secure as they build software. That's why secret scanning is a key part of GitHub's security suite and free for all open-source and public repositories[8].

8: GitHub, "Secret scanning alerts are now available (and free) for all public repositories", Feb 2023

In 2024 alone, 3.3M secret leaks were prevented on GitHub via push protection

GitHub continuously scans your Git history, including branches, issues, commits, and pull requests, with a developer-native experience directly in the GitHub UI or command line. In addition, GitHub partners with over a hundred service providers[9] to protect mutual users with regularly updated token information and remediation instructions.

9: GitHub, "Secret scanning partner program"



The next evolution is preventing secret leaks before they occur. With secret scanning push protection, you can scan code before it's pushed to see if it contains a secret. If it does, the push is blocked, and the user is notified that it contains a secret and is provided guidance for a fix. Administrators can also set up rules around whether a developer can bypass push protection. For example, they may want stronger rules on a production repository versus a testing environment.

If your organization is leveraging GitHub, it's easy to activate secret scanning today. To learn more, please visit GitHub Advanced Security.