LEGIT SECURITY

WHITE PAPER

Overcoming the Challenge of Protecting Secrets in the SDLC

Secrets in the Modern Development Landscape

In today's modern development ecosystems, every environment contains hundreds of secrets that hold access to the most sensitive parts of your organization. Secrets play a pivotal role in the functionality of modern applications: tokens and passwords guard your source code, cloud environment, and most important data, while API keys manage all your 3rd party integrations. An average application needs dozens or hundreds of secrets to be fully deployed and function properly.

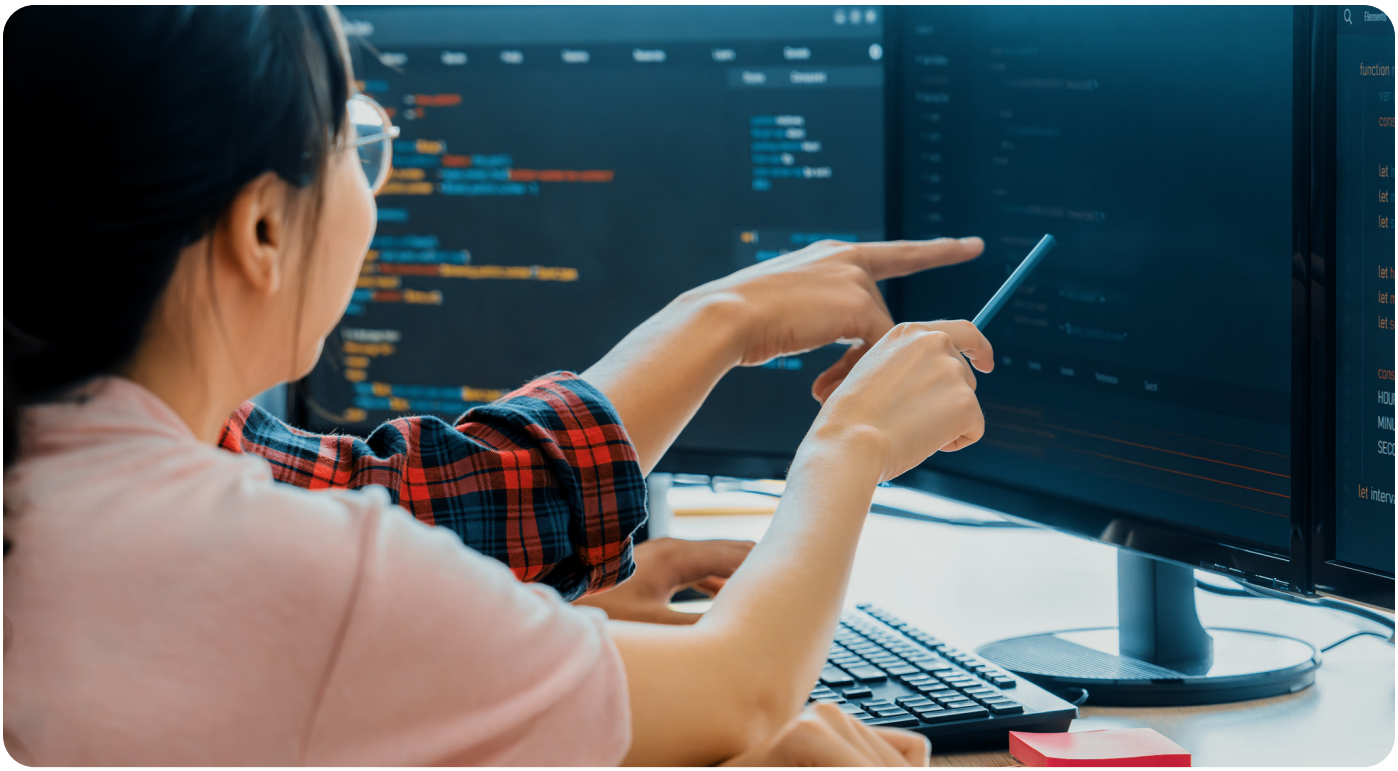
In addition to secrets related to access, PII, including social security numbers, credit cards, and more, is another form of secret that often ends up in the development environments when developers use it to test application functionality. Once they've made it into the development ecosystem, they are vulnerable to exploit and are a prime target for attackers. And in most instances, PII in the developer environment is a compliance violation putting the organization at risk of costly fines and other potentially serious consequences.

12
average number of secrets per 100 repositories.

204
days is the average time to identify a data breach.

$4.5M
is the average cost of a data breach.

$183
average cost per record containing PII.



What are Secrets in Code?

Keeping secrets away from code is already a known security practice, yet tokens, keys, passwords, and other secrets still make their way into code repositories. In fact, research conducted by Legit found that on average, 12 live secrets are submitted per 100 code repositories every week.

So why does it happen?

Handling secrets safely is not a simple task, and the pressure to develop and deploy software as fast as possible is high. Many developers write and release code quickly, leaving secret handling for later. A developer can choose to temporarily hard-code the secret, and when merging the final revision to the main branch, remove the secret and switch to a more secure option (for example, injecting the secrets in run-time from a secret management software).

Unfortunately, people make mistakes, and many times those secrets are forgotten in the code, overlooked in the code review, make their way into the main branch – and ultimately the final artifact. That secret will also be copied to every developer's endpoint once they clone the repository, so even if the secret is deleted at the source, it will live on in its copies.

But even if the developer doesn't forget to delete the secret in their final commit, that doesn't mean the code is free from secrets.

12

live secrets, on average, are submitted per 100 code repositories every week.

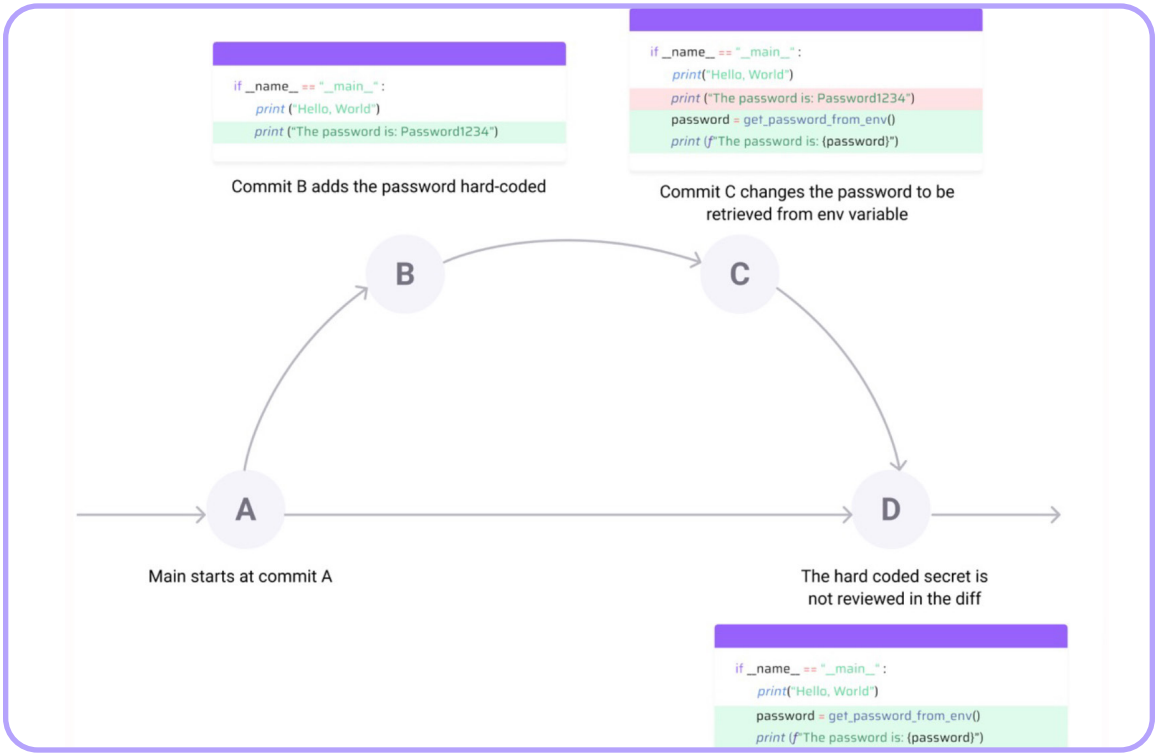
Your GIT History: Where Secrets Go to Hide

When a secret exists in one or more of your code commits, it stays in your git history forever, even if it was deleted in a later commit.

Let's look at the use case portrayed in the image below:

A developer created a new branch, committed a secret in B, and deleted it in C.

- During the CR process, the reviewer will only look at the difference between the last commit (C) and the main branch.
- As there are no secrets present in the C commit, the code would be approved and merged, and no special action will be taken.
- The secret will live on in the git history of the branch forever.



Beyond the Source Code: Secrets Everywhere Else

Beyond their common existence in source code, secrets may inadvertently appear in artifacts, containers, and build logs, for different reasons such as a misconfigured CI system or a mishandled secret in a docker image. Code is often kept private and guarded by other security measures, but containers and artifacts are sometimes inadvertently made public, posing a bigger risk to the secrets in these resources being exploited. Recognizing the extensive presence of secrets in your environment and the impact of even one leaked secret calls for a more comprehensive approach to secret scanning and management.

The Damage Just One Secret Can Cause

Recent incidents serve as stark reminders of the dangers that secrets can pose, and how code exposure can result in even more devastating outcomes if secrets are present. A number of high profile security incidents and breaches in the past couple of years highlight the extreme risk that secrets in code can pose.

Toyota

The Toyota data breach serves as a stark reminder that secrets aren't safe in private repositories. When a subcontractor working for Toyota accidently uploaded source code to his own public repo, the credentials for a database containing customer information that were present in the code were made publicly accessible. This stayed online for 5 years, allowing anyone access to this sensitive information.

Codecov

The Codecov software supply chain attack affected thousands of users who downloaded a malicious version of Codecov's Bash Uploader script that extracted users' sensitive information. But how was the attacker able to alter and upload the malicious script? It seems that they managed to extract Codecov's Google Cloud storage key from a public docker image.

Uber

The infamous 2016 Uber data breach started with an 18 year old attacker gaining access to a private Uber GitHub repository, but all the damage that followed was made possible due to the fact the repository contained hard coded credentials. The attackers discovered that the developers had committed Amazon Web Services (AWS) credentials. With these credentials, the hackers were able to access a treasure trove of sensitive data, including the personal information of millions of users and drivers.

Unlike many vulnerabilities, exposed secrets pose an immediate data security risk, whether it's your own internal data or your customers'. The damage is immediate and in many cases doesn't require much more than the secret itself. The scale of this issue far exceeds the awareness of many organizations that put 100% of the focus on application security, underscoring the critical need to address the risk that secrets in code and other developer assets can have on their data.

300K

the number of customers whose PII was exposed in the Toyota breach

4

months is the time attackers went undetected in Codecov customer environments

$148M

in fines were paid by Uber in response to the 2016 data breach

3

times Uber was breached in a recent 6-month period

Addressing the Problem of Secrets – How to Get it Right

Robust secret scanning capabilities are paramount to an effective secrets management strategy. Tools and technologies that can meticulously analyze code repositories, build logs, artifacts, and even container images are essential to understanding where secrets exist in your developer environment and how they got there. These scanners offer many different features and possibilities, so let's talk about what you should look for in a secret scanning solution.

These scanners offer many different features and possibilities, so let's talk about what you should look for in a secret scanning solution

1. Scanning the entire SDLC

Choosing the right secret scanner that scans all the moving parts of your SDLC is critical.

In addition to choosing a solution that works with your SCM (whether it's GitHub, GitLab, Bitbucket, etc.), look for a scanner that scans beyond your main branch, like your entire git history and subbranches, where secrets are typically overlooked.

As established earlier, secrets no longer exist only in source code, but can also be found in Docker containers, build logs, artifacts, and documentation services like Confluence. This poses an even bigger risk because while secret scanners are becoming standard in most organizations, not all of them scan more than just code. Choosing a solution that scans resources beyond source code is key to overcome this risk.

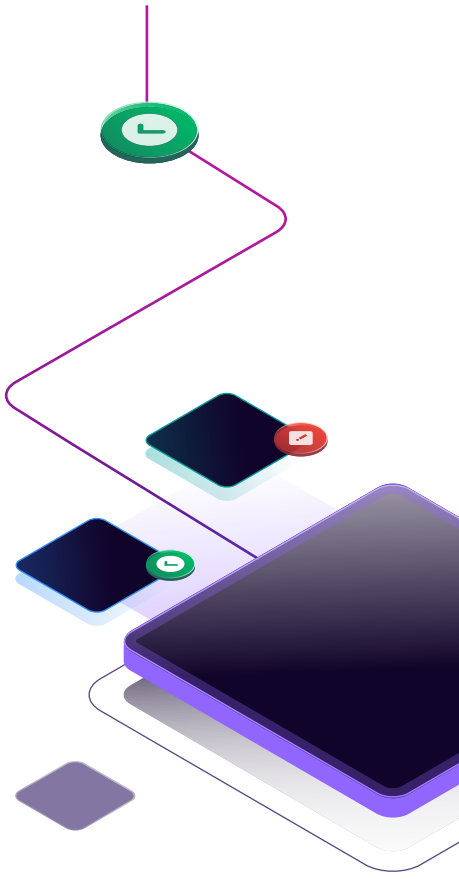
2. Automation & prevention

Automation and prevention lie at the heart of an effective secrets management strategy, and you should look for them in your solution. A scanner with those capabilities that integrates into your CI/CD pipeline can help ensure that all code will undergo scanning before being merged to your main branch and deployed, to minimize the likelihood of missed secrets.

It's important to evaluate a solution's prevention capabilities to stop the introduction of new secrets into your SDLC. Look for a scanner that can stop secrets from being pushed to the remote git repository using pre-receive hooks. Then assess the solution's automated tools to streamline the process of scanning for secrets and reduce the reliance on manual checks, which can be time-consuming and prone to human error.

3. Performance at scale

An important yet sometimes overlooked quality in a scanner is its performance at scale—particularly its ability to scan thousands of repos within minutes to ensure that secrets are identified and addressed immediately. When faced with the monumental task of scanning all of an



organization's repositories, some scanners may turn out to be notable bottlenecks.

Most open source-based scanners, while valuable in many contexts, may fall short when it comes to meeting the demands of large-scale organizations. In today's security landscape, where an exposed key can take only minutes to be found and exploited, a fast and efficient tool can make or break your security posture.

4. Avoiding false positives and negatives

A proficient secret scanner will offer a balance by identifying all the secrets present in your code on the one hand and minimizing false positives on the other. Unfortunately, when dealing with secrets, a certain percentage of false negatives is unavoidable without missing critical events. This is mainly in cases where additional context and user input are keys to accurately labeling code.

> A trustworthy scanner should lean towards maintaining a zero-risk policy of missing genuine secrets, but with the ability to provide solutions for quickly analyzing and triaging to optimize efficiency.

A centralized managed secret scanner offering easy customization for assessing secrets is vital in dealing with cases like these, allowing users to quickly identify and ignore specific patterns or files after they are deemed to be non-critical.

5. Contextualized secrets

In the intricate landscape of AppSec, you can easily get lost in the plethora of tools and the issues they detect and alert on. A secrets scanner that connects to your AppSec ecosystem and helps you navigate those issues easily is crucial. Offering context around detected secrets can be done in several ways:

- Leveraging APIs to send results to an AppSec dashboard or vulnerability management platform.
- Integrating with a ticketing system for swift identification and resolution.
- Providing insight into the severity of the issue, based on the environment surrounding the secret (i.e. a secret in a public repository should have a higher severity than in a private one).

Regardless of the situation, it's important to choose a secret scanner that fits seamlessly into your existing application security workflow.

6. Validity checks

It's important that your secret scanner offers effective validity checks that help determine if a detected secret is real and usable. This ensures your ability to identify real threats while minimizing mistakes. These checks help the scanner stick to a no-nonsense policy for finding actual issues. In simpler terms, it's like having a smart security guard that knows when to raise the alarm for a real threat and when to stay quiet for a false alarm. This option is especially important for production-related SaaS services, like AWS access key, and helps you determine the triage and criticality of the problem.

The Benefits

A well-implemented secrets solution holds an array of benefits, empowering organizations to protect sensitive internal and customer data to secure their SDLCs and uphold regulatory requirements. When properly executed it promotes the swift and secure release of software, enabling developers to focus on innovation without compromising security.

Don't waste developer time on secrets management

A good secret scanner can help minimize the time your developers are spending tracking down and removing secrets, helping them release secure software quickly. An automated scan replaces a reviewer's need to meticulously search for secrets in code and other parts of the developer environment, while pre-receive hooks prevent the secrets from ever even reaching the pull request.

It's also important to understand that addressing a secret that has already been deployed doesn't end in simply revoking it. Frequently a secret exists in a Docker image because the running container relies on it. Revoking the secret is a potentially destructive step that can eliminate an application's access to critical data, leading to the "breaking" of that container and any resource that relies on it. The result is a lot of time wasted by revoking the secret, finding a different way to pass it, building a new container, and deploying it. A secrets solution that scans all types of resources will allow you to be proactive about addressing secrets before they make it into the development pipeline and start causing issues.

Level up your breach prevention game

Exposed secrets are a huge and actively exploited attack surface that played a highly valuable part in all SSC incidents that took place in the recent years. Apart from the danger to customers and the pr problem, incidents like these cost organizations millions. According to IBM's 2023 Data Breach Report, compromised credentials is one of the two most common initial attack vectors, with an average cost of USD 4.62 million. Additionally, these breaches took the longest to resolve. It took an average of 328 days (nearly 11 months) to identify and contain data breaches resulting from stolen or compromised credentials. Implementing a secret scanner as part of your SDLC is a crucial step in preventing your next breach or stopping one in its tracks.

Choosing the right scanner when one size doesn't fit all

Most open source-based scanners, while valuable in many contexts, fall short when it comes to meeting the demands of large scale organizations. When faced with the monumental task of scanning hundreds or even thousands of repositories inherent to enterprise-scale operations, you need a scanner that won't be a bottleneck. When an exposed key can take only minutes to be found and exploited, the speed and efficiency of your tools can make or break your security posture.

According to IBM's 2023 Data Breach Report, compromised credentials is one of the two most common initial attack vectors, with an average cost of USD 4.62 million. Additionally, these breaches took the longest to resolve. It took an average of 328 days (nearly 11 months) to identify and contain data breaches resulting from stolen or compromised credentials.

Conclusion

Secrets have come to be an integral part of our development ecosystem. With the ever-growing number of services, integrations, and tools in the SDLC, the number of secrets is constantly growing and managing them correctly becomes quite the challenge. Despite this, keeping your code clean and your secrets secure is imperative.

The potential risk entailed is significant to organizations of all sizes, and addressing it requires a holistic approach that includes:

- Scanning all parts of your code, including git history
- Scanning SDLC resources beyond the code
- Implementing automation and prevention capabilities
- The ability to quickly identify and ignore false positives
- Contextualization of secrets for faster remediation
- Enterprise grade scalability and performance

By adopting these strategic measures, organizations can fortify their defenses against potential breaches and ensure the integrity of their sensitive information.

Legit Security: Solving the Secrets Conundrum

It's imperative for organizations to recognize the problem that secrets can pose and take proactive steps to fortify their defenses. Legit Security offers a comprehensive secrets management solution with the tools and capabilities needed to tackle this critical issue head-on. With the capability to scan a wide array of secret types across every facet of your CI/CD resources, it ensures a comprehensive and continuous approach to secrets management.