PeerSpot

Case Study

GitGuardian Internal Monitoring

Melvin Mohadeb

Security Engineer at PayFit

Review by a Real User

Verified by PeerSpot

What is our primary use case?

The main goal is to be alerted and to react when a secret has been leaked in our code base.

We have GitGuardian linked to our code-based storage on GitHub. GitGuardian also has a notification integration with Slack which is what we use internally for communication. We are alerted on Slack, "There's an incident here on GitGuardian for a secret leak on GitHub." From there, we can go into incidents and start managing the incident.

How has it helped my organization?

Before this solution, we didn't have anything for secret detection. We went from zero to having something. We really needed it. It was really a big risk for us without it. The more the company grows, and the more we have employees coming and leaving, the risk of secrets leaks in our asset base is really big. Thanks to the tool, we have decreased the risk.

Before, what we did was check the code manually to detect secrets. Now, it's automated, and that's a big change for us. Security team productivity has also increased because it helps us manage incidents. Everything that GitGuardian does is something we don't have to do manually. That is definitely increasing our productivity.

It also supports a shift-left strategy.Dev in the loop is pretty good when it comes to collaboration between developers and security teams. The fact that GitGuardian is very fast in detecting and alerting makes remediation easier. When a secret leaks, we get the alert

within 30 seconds, or a maximum of one minute, which is very fast. Once we get the alert, we can warn the developer and it will not require a big change because they would have just committed the secret. It won't be a secret that was committed multiple days before. The few times we used it, it definitely made remediation faster.

What is most valuable?

The detection feature works really well. It's pretty fast and we are alerted very well.

Also, the breadth of the solution detection capabilities is pretty good. They have good categories and a lot of different types of secrets. There is one generic type when they don't know specifically what it is, but it gives us a great range when it comes to types of secrets, and that's good for us.

The detection accuracy is also good. We haven't had a lot of false positives, which is nice. We are not aware of any false negatives, such as not being alerted when a real secret has leaked.

The web interface helps to quickly prioritize remediation as you can manage incidents. You have to indicate the severity of an incident after seeing the secret, knowing where it is used. We definitely use this feature.

What needs improvement?

The good thing about GitGuardian is that we don't get many false positives. The issue with this kind of tool is that it detects secrets but it can also detect some things that are not secrets, and you have to manage an incident for something that is not an incident. But we tested multiple secret detection tools and GitGuardian was pretty good, not having many false positives.

There is also something we shared with them already about user management with teams. They have an integration with Okta to manage our employees' access to the tools. It would be best to have different teams. In our engineering department we have a lot of different teams, and the more we grow the more teams we will have. But currently, you can only assign one person to an incident. We would like to have the ability to assign it to a team because code, in our company, is owned by a team and not one person. That's one feature that's really lacking in GitGuardian.

For how long have I used the solution?

I have been using GitGuardian Internal Monitoring for about 10 months.

What do I think about the stability of the solution?

We haven't had any issue with its reliability. It has always worked and we have never had downtime with GitGuardian. It's very good.

What do I think about the scalability of the solution?

The scalability is definitely not bad, but it's not the biggest strength, for sure. But it's not a "no-go, definitely do not use this tool."

There are some features that are lacking in GitGuardian. The more we grow and the more engineers we have, the more difficult it will become to assign an incident because the assignment is not automatic. I know they are working on that and we are waiting for it.

We currently have 52 members using it. It checks our entire developer worker base. We're satisfied with the current usage, but we'll increase the number of members as we grow.

How are customer service and support?

There have only been rare cases where they didn't answer all my questions. Some things were not possible, but they are very responsive and try to do their best to answer my concerns.

How would you rate customer service and support?

Positive

Which solution did I use previously and why did I switch?

We didn't have a previous solution.

How was the initial setup?

I don't remember that there was a lot of preparation involved. It was really just a matter of doing the integration between GitGuardian, GitHub, and Slack. That's all. The implementation of GitGuardian is really easy. You just have to set up the integration, which takes, maybe, five minutes, maximum.

There is no maintenance. We have to manage incidents, but that's the point of the tool. But we don't have to maintain the tool itself. It's SaaS and it works on its own.

Which other solutions did I evaluate?

We checked Gitleaks, which is a free tool for detecting secrets. Detections were pretty much the same in both GitGuardian and Gitleaks. The main difference was that with Gitleaks, you don't have the interface for incident management. It's really just detection. GitGuardian was the whole environment that we really needed to work at scale.

What other advice do I have?

The tool itself mainly helps us with detection.

The whole remediation is done outside of the tool. Once GitGuardian has detected a secret leak, we are alerted and an incident is created in the tool itself. After that, the revocation or rotation of the secret will be done outside of the tool. We use GitGuardian to track the incident and the comments on it, but we don't really manage the secrets directly in it.

We had some issues with the Dev in the loop feature, so we don't use it that much. Dev in the loop is used to share an incident with the developer who committed the secret. But to manage our database in our GitHub organization, we let our developers use their personal emails. Because an email is sent to that address about a secret leak, we are not very fond of it. It works well and is helpful because we don't have to manually send a message to the developer for an incident. We can let the developer manage the whole thing on their own, which is good. We just have this email issue, but other than that, the feature in itself works well.

If a security colleague at another company were to say to me that secrets detection is not a priority, I would disagree. The risk is pretty big when you think about what a secrets leak could do. You don't need to start with a solution like this when your company has, say, five people. But at a certain point, you definitely have to have a secrets detection tool.

Read 9 reviews of GitGuardian Internal Monitoring

See All Reviews