PeerSpot

Case Study

GitGuardian Internal Monitoring

reviewer1692456

DevSecOps Engineer at a computer software company with 1,001-5,000 employees

✓ Review by a Real User
✓ Verified by PeerSpot

What is our primary use case?

Mainly we use GitGuardian to keep secrets out of our source code. That is something that we wanted to get serious about getting our hands around. This was the main driver because I had tried other tools like TruffleHog. It was cumbersome to manage the unwieldy Git history and to figure out. When you run TruffleHog, you have no way of knowing what's in the current branch versus your Git history. Hence, it's tough to decipher what secrets are still possibly valid.

How has it helped my organization?

We didn't have a secret detection tool in place before GitGuardian, so we had no solution that could detect when secrets were committed and sourced. With GitGuardian, we get an instant notification every time a secret is committed, so we can immediately triage it.

GitGuardian has absolutely supported our shift-left strategy. We want all of our security tools to be at the source code level and preferably running immediately upon commit. GitGuardian supports that.We get a lot of information on every secret that gets committed, so we know the history of a secret. For example, if there are SMTP credentials that get used and reused, we can see where the secret may have traveled, so GitGuardian may give us a little more information about that secret because it can tie together the historical context and tell you where the secret has been used in the past. You can say, "Oh, this might be related to some proof-of-concept work. This could be a low-risk secret because I know it was using some POC work and may not be production secrets."

I don't know how to quantify how much time it has saved our security team because we didn't have anything similar in place before GitGuardian. I can say that tracking down a secret, getting it migrated out of source code, getting the secret rotated, and cleaning the Git history took much longer from commit until the full resolution before GitGuardian. We weren't notified until it was too late, but with GitGuardian, we know almost instantly.

We have standard operating procedures for every notification. We know how to rotate the secret. We know how to remove it from the source code. We have documented procedures for how to do that. We can rip it from the code, rotate it, and clean the Git history in a couple of hours. If something gets committed, it sits there for a while before we notice it.

Overall, GitGuardian has also helped us develop a security-minded culture. We're serious about shift-left and getting better about code security. I think a lot of people are getting more mindful about what a secret is. It's like back in the day before campaigns like Cofense PhishMe became a big thing. People were clicking phishing links all the time. Now you have these training programs where people see these things, and they're more aware of it.

It's a similar situation when you're writing code as well. I think people are getting more aware of secrets. What is a secret? Does this belong in the source code? Sometimes they even come out and ask, "Is this a safe thing to commit to the source?" before they even commit it. They don't want to be "yelled at" by the GitGuardian. I think that it has had a positive impact on the culture itself.

You're only as good as the software you write, and you're in for a world of hurt if you put the keys to the castle inside of that source code that could be somehow reverse-engineered. By separating the two, the source code and the keys, you're one step ahead of that. I think it's essential.

What is most valuable?

The most valuable thing about GitGuardian is the speed with which it works. If you accidentally commit a private key to a public repo, you need to know that instantly. GitGuardian has this thing called "Dev in the loop." The developer who committed the secret is notified, and they get a form to fill out so they can give us instant feedback, which is super helpful for us. Due to the nature of the software we write, sometimes we get false positives. When that happens, our developers can fill out a form and say, "Hey, this is a false positive. This is part of a test case. You can ignore this." What's more, the tool helps us with triage. As soon as the secret is committed, we receive Slack alerts and jump right on it.

GitGuardian's "Dev in the loop" feature has sped up our time to remediation quite a bit. Of course, not every developer is responding, but that's just the nature of the organization itself. It's not the fault of the product. It's just that some people are not as quick to act. So when developers do respond, I would say issues get resolved several times faster because we know from the jump if it's an issue or not.

It's hard to evaluate how accurate the tool is because of the type of software we write. We're a vulnerability company here, so we write a lot of test cases using test data that are looking for things like secrets, so we have false positives. Some of GitGuardian's detectors take that information into account. With things like a general high-entropy detector, we expect a potentially high false-positive rate. However, for something like an AWS key detector, GitGuardian's efficacy is near a hundred percent, if not 100%. I can't recall any instances off the top of my head where it inaccurately flagged an AWS key or an Azure key.

What needs improvement?

One improvement that I'd like to see is a cleaner for Splunk logs. It would be nice to have a middle man for anything we send or receive from Splunk forwarders. I'd love to see it get cleaned by GitGuardian or caught to make sure we don't have any secrets getting committed to Splunk logs. That was an issue that I brought up a while ago. However, my workload just hasn't allowed me to sit down and figure out how to solve that. That is one thing that I wanted to see if I can use in that regard because secrets are a thing that ends up in logs, and that's not something we want.

For how long have I used the solution?

The first time I looked at GitGuardian was about a year ago now. We have open-source information on public GitHub, but all of our proprietary code is on an internal GitHub Enterprise Server. When we set up our internal GitHub Enterprise Server and deployed GitGuardian, it had no network path out to the public GitHub. I worked with GitGuardian, and they set me up with public monitoring. I would monitor all of my public open-source information with the public offering. Then I would also have my internal monitoring setup for everything on our GitHub Enterprise Server.

What do I think about the stability of the solution?

GitGuardian has been pretty stable probably 99% of the time. There was one time where I had a slight hiccup, so I restarted the cluster, and it was good to go.

What do I think about the scalability of the solution?

I think GitGuardian scales well. It's adequately scaled for what we are using it for right now. I don't see that growing. Right now, we just have it hooked up to our source, and it can handle that. Now, if we were to expand into possibly doing the Splunk use case, that might bring in an API. In that case, I'm not sure what the performance impact would be, but I don't think it would be that bad. You throw a couple of extra nodes out there, and it should be fine. It's currently being used by all of our developers. Everyone who commits code is using it. It scans all of our code.

How are customer service and support?

GitGuardian's support is fantastic. I don't think I could rate them anything less than a 10 out of 10. We had a few questions about how to stand up our deployment. The SRE assigned to our project was readily available and very knowledgeable. He jumped on a call and spent crazy hours helping us out. I thought they were very flexible and easy to work with. I've never had an issue with their support. They've given us everything I've needed when I needed it.

How would you rate customer service and support?

Positive

How was the initial setup?

We installed the software and connected it to our GitHub. Literally within minutes, it was scanning and finding secrets in our GitHub. It doesn't take long to get it up and running and we didn't have to make any significant architectural changes before deploying GitGuardian. We only had to stand up a VM and then set up the network pathways to talk to our GitHub. That was a very minimal amount of work from our CIS ops team to put that out. After installation, it doesn't require much maintenance. When they tell me a new release is out, I log into the console, click the upgrade button, and it does its thing.

What was our ROI?

We've absolutely seen ROI. For example, if somebody accidentally commits an AWS key to your public GitHub, somebody can take that key and spin up EC2 instances, which can cost us thousands of dollars. The fact that we can catch it is almost invaluable, but it's worth the investment to have the tool. Everything is cheaper if we can find an issue and resolve it sooner. It's much more affordable to remove a secret well before it gets merged into a master branch than it is to try to rip out the historical commit. It affects the bottom line in that regard.

What's my experience with pricing, setup cost, and licensing?

I think GitGuardian's price isn't too expensive. I'm not sure about any add-ons or additional costs because I wasn't involved in purchasing GitGuardian. I know the ballpark price, but I did not handle the pricing. Other people in our organization negotiated the pricing, but I'm not aware of any hidden costs or anything like that.

Which other solutions did I evaluate?

We looked at some open-source solutions like TruffleHog, and we also looked at the GitHub secrets detection, but the issue was that it was bundled with their advanced security, which we were not planning to purchase. GitGuardian just made perfect sense for us.

GitGuardian has the GUI that TruffleHog doesn't have. TruffleHog can scan your GitHub and tell you where secrets live. But it does not do a perfect job of telling you where those secrets live within your timeline. GitGuardian does an excellent job of telling you the branch where those secrets live and where they are on the timeline. The Github tool does pretty much the same thing, but it was off the table for us because we were not planning on purchasing their advanced security toolkit.

What other advice do I have?

I rate GitGuardian 10 out of 10. It does everything that I need it to do, and I'm excited about the new features that are coming along at this point. It has really helped us change our culture, and it's impressive to see that. People are now more mindful of what gets committed to source code. I would recommend GitGuardian.

Which deployment model are you using for this solution?

On-premises

PeerSpot

Validated User Review

Read 7 reviews of GitGuardian Internal Monitoring

See All Reviews