The state of Secrets Sprawl on GitHub

HOW LEAKY CAN IT GIT

Summary

GitHub is more than ever “*The Place to Be*” for developers when it comes to innovating, collaborating and networking.

This amazing “octoverse” gathers more than 50 million developers working on their personal and/or professional projects. So when 60 million repositories are created in a year and nearly 2 billion contributions* are added, **some mistakes can happen**, such as leaked secrets, Intellectual Property or PII.

Some companies may think: I don’t really care about public GitHub, we are not open sourcing our code, everything is stored on our private repositories. But what about the developers of these companies... they most likely have open source repositories and can leak secrets.

*State of the octoverse 2020

Secrets Sprawl

Let’s now focus on secrets. You would say that secrets stored in internal Version Control Systems is a very bad practice but in fact it is much more frequent than you would think. But why is that?

API keys, database connection strings, private keys, certificates, usernames and passwords... As organizations move to cloud architectures, SaaS platforms and microservices, developers handle increasing amounts of sensitive information, more than ever before.

To add to that, companies are pushing for shorter release cycles, developers have many technologies to master, and the complexity of enforcing good security practices increases with the size of the organization, the number of repositories, the number of developer teams and their geographical spread.

As a result, secrets are spreading across organizations, particularly within the source code. This pain is so huge that it even has a name: Let us introduce you to the concept of “secrets sprawl” and how this can lead to public exposure of some of your most sensitive assets.

At GitGuardian, we’ve been monitoring every single commit pushed to public GitHub since July 2017. Three and a half years later...

we’ve uncovered millions of secrets and sent nearly 1 million pro bono alerts to developers in 2020 alone.

SECRETS

A secret can be any sensitive data that we want to keep private. When discussing secrets in the context of software development, secrets generally refer to digital authentication credentials that grant access to services, systems and data. These are most commonly API keys, usernames and passwords, or security certificates. Secrets are what tie together different building blocks of a single application by creating a secure connection between each component. Secrets grant access to the most sensitive systems.

Learn more about secrets on our blog

SECRETS SPRAWL

Keeping secrets encrypted and tightly wrapped makes it harder for developers to both access and distribute them. This can lead developers to choose the path of least resistance when handling them which may include hardcoding them into source code, distributing them through email or messaging systems like Slack, saving them directly into config files and storing them inside internal wikis. Once secrets start to enter different systems:

- Attackers can move laterally through infrastructure
- You lose visibility over where secrets end up.

Commit

A commit is an incremental change that has been made to an individual or set of files.
When making a commit, the difference (or diff) between the current version of files and the previous version is saved, including data that was removed.

So here is a deep dive into what we find...

WHAT ARE WE LOOKING AT

2.5 M
public commits
scanned/day

almost **1 B**
public commits
scanned /year

AND THE VOLUME IS GROWING*

35 %
more repositories
created last year

25 %
more contributions
to open source projects

*State of the octoverse 2020

WHAT DO WE FIND

more than **5 K** secrets detected/day

over **2 M** secrets detected in 2020

A GROWING NUMBER...

+20 % compared to previous year

WHERE DO WE FIND THE SECRETS

Secrets present in all these repositories can be either personal or corporate and this is where the risk lies for organizations as some of their **corporate secrets** are exposed publicly through their current or former **developers' personal repositories**.

85 % of the leaks occur on **developers' personal repositories**.

15 % of leaks on GitHub occur within public repositories **owned by organizations**.

We launched this audit, and several leaked secrets were brought to our attention. What was very interesting and what we didn't anticipate was that **most of the alerts came from the personal code repositories of our developers.**

Anne Hardy, CISO
talend

TOP 10

Where leaks come from

01	India
02	Brazil
03	United States
04	Nigeria
05	France
06	Russia
07	UK
08	Canada
09	Bangladesh
10	Indonesia

Why

Usually these leaks are unintentional, not malevolent. They happen because:

- Developers typically have one GitHub account that they use both for personal and professional purposes, sometimes mixing the repositories.
- It is easy to misconfigure git and push wrong data.
- It is easy to forget that the entire git history is still publicly visible even if sensitive data has since been deleted from the actual version of source code.

> **Human error exists, but the key is to be alerted and be able to take appropriate action when a leak is found.**
>
> Anne Hardy, CISO
> talend

> **Human error is nothing you can avoid and prevent, especially if it is not an error but just laziness, or even provoked, implement a risk based approach and simply add many layers to prevent it in your whole lifecycle.**
>
> David Dos Neves - Munich Re

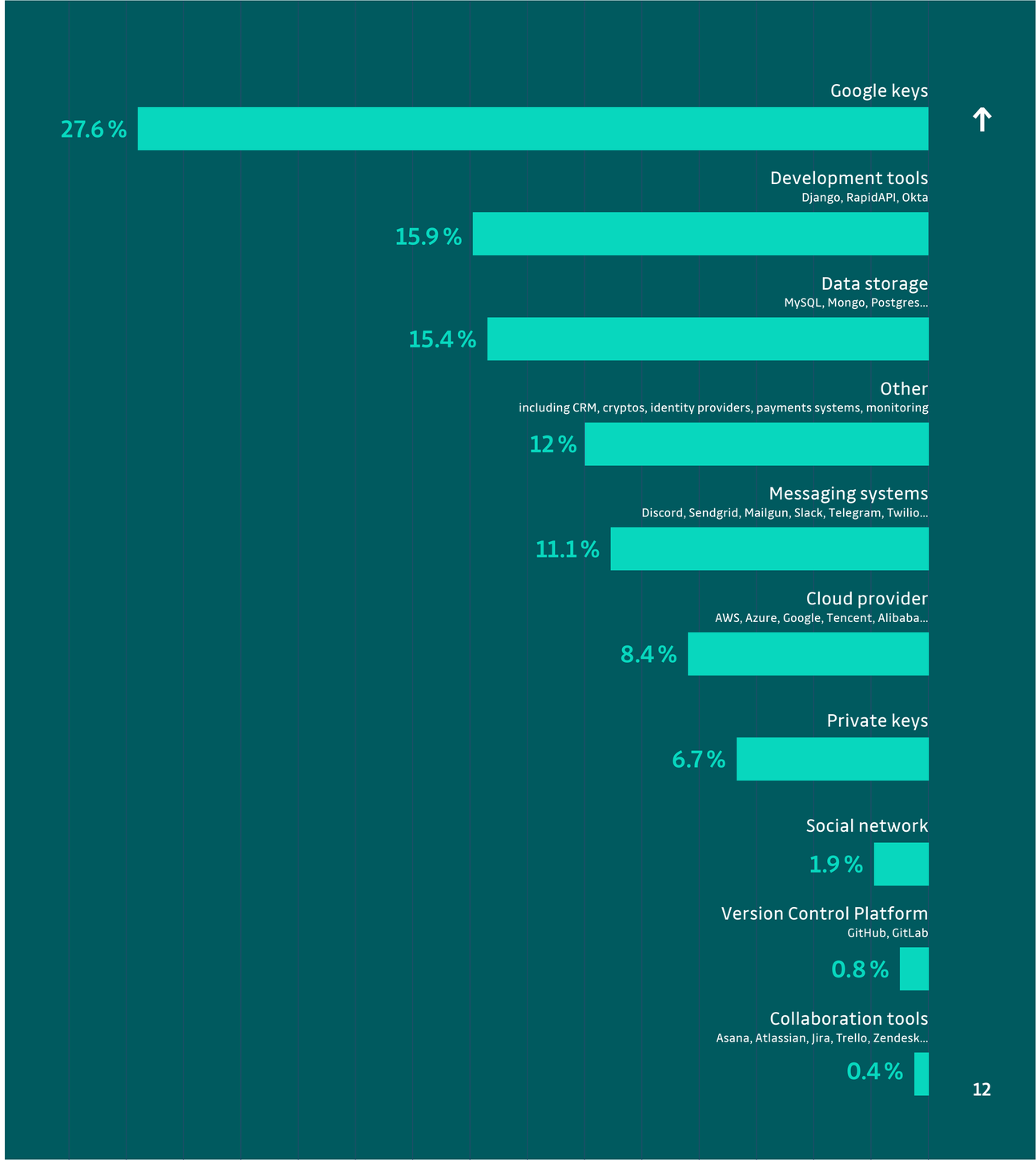
What type of secrets do we find

Secrets are digital authentication credentials that grant access to services, systems and data (API keys, usernames and passwords, or security certificates). The volume and diversity of these digital authentication credentials is growing fast as architectures move to the cloud but also rely on more and more components and apps.

> **Our larger customers, with 2,000 or more employees, deploy an average of 175 apps per customer, while our smaller customers, with 1,999 or fewer employees, deploy an average of 73 apps per customer.***

*Okta

All these categories of secrets expose companies to easy and direct attacks. Cloud provider and data storage secrets by data loss but also by allowing infrastructure suppression. Identity provider and messaging system by allowing legitimate identity usage.



TOP 10

File extensions that cause data breaches on GitHub

As you might expect, with the many programming languages, frameworks and coding practices adopted throughout the world, there is a very long list of extensions that can contain secrets, here is the view of the top 10.

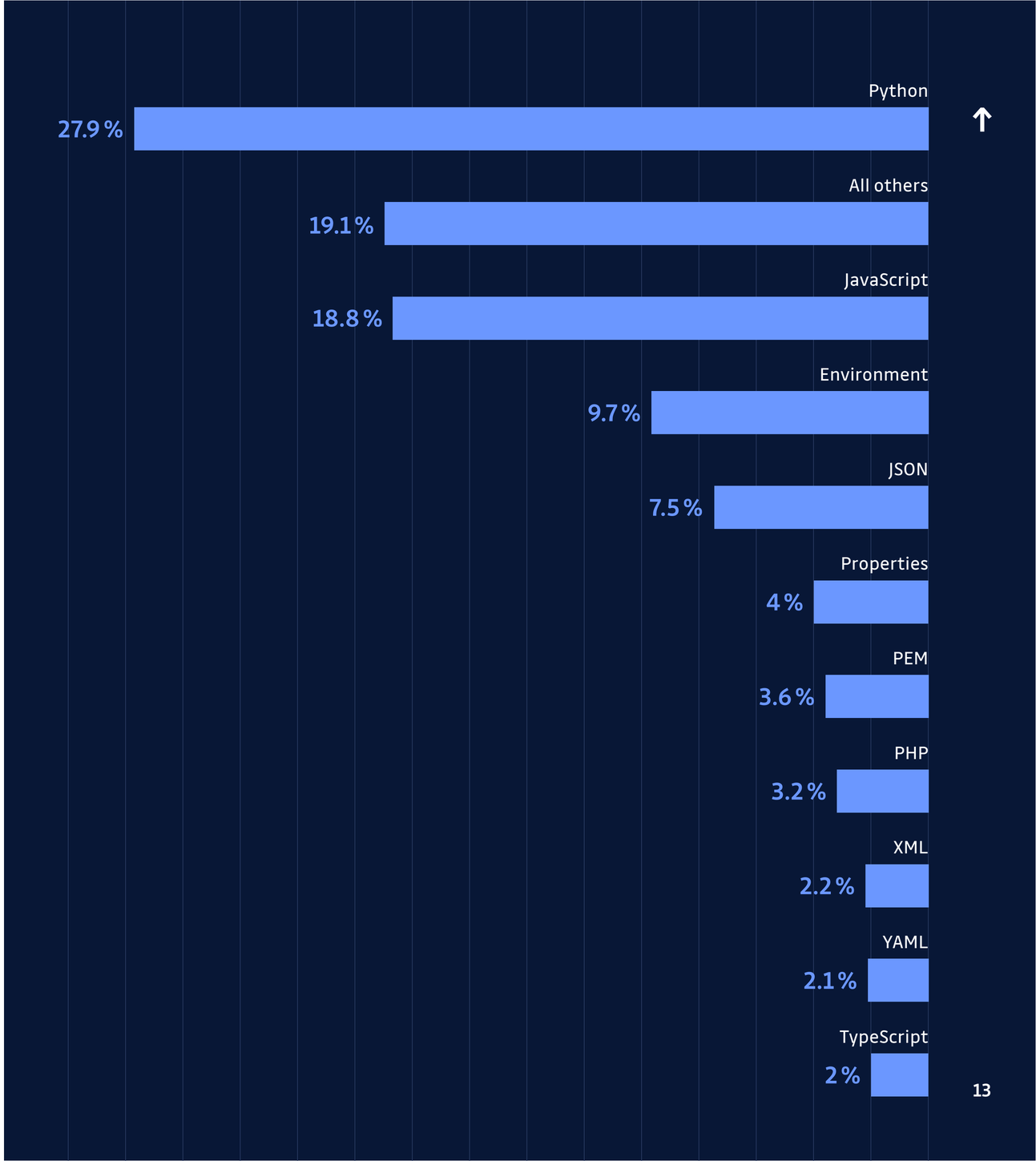
- Top 10 file extensions account for 81% of all the results
- The top 3 accounting for over 56% of the results

File extensions can be grouped into 3 categories

- **Programming languages:** Python, JavaScript, PHP, TypeScript
- **Data serialization files:** JSON, XML, YAML, .properties
- **Forbidden or sensitive files:** .env, .pem

Learn more about how secrets leak through file extensions on our blog*.

*Read the article



EXAMPLES OF SECRETS LEAKS

Publicly disclosed examples of recent data breaches through leaked credentials.

Uber Data Breach*

May 2014

Hackers discovered credentials in a personal public repository on GitHub that granted access to a database containing private information of thousands of Uber drivers.

*Read the article

Starbucks Data Breach*

January 2020

JumpCloud API key found in GitHub repository.

*Read the article

Equifax Data Breach*

April 2020

Leaked secrets in personal GitHub account granted access to sensitive data for Equifax customers.

*Read the article

UN Data Breach*

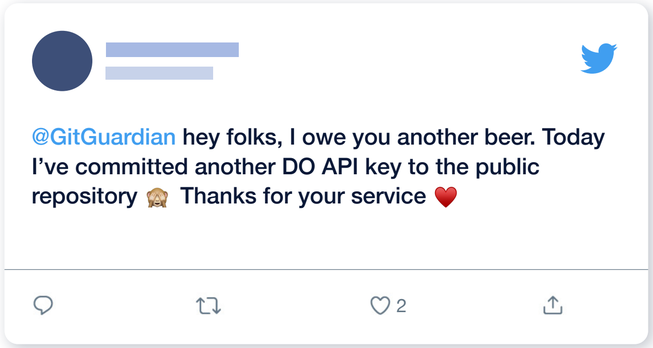
January 2021

.gitcredentials in a public repository giving hackers access to private repositories with sensitive information.

*Read the article

WHAT USUALLY HAPPENS

A user that first writes his code with credentials in the code so that it is easier to write/debug, he then forgets to remove it from all his files after his work is done.
He then commits and pushes his changes.
When he understands that he made a mistake, either he does a deletion commit or a push force so that the secrets do not appear in his current version.
Most of the time, he forgets that git and the internet are not forgiving: Secrets can be accessed in the git history even if they aren’t in the current version of code anymore, and public data hosted on GitHub can be duplicated and cloned into multiple different locations.



WHEN IT REALLY GOES WRONG

This is when a user pushes professional work on a personal repository while not really understanding git/GitHub. In his repository, we find shell commands history, environment files as well as copyrighted content.
When the developer understands that he made a mistake, he only adds a deletion commit (or multiples if he doesn’t find all leaks at the same time). This commit has a message such as “remove secrets from repo”. The credentials that he leaked will not be revoked and will remain public in his git history.

Pro bono alerting

Such knowledge of leaked credentials comes with a great responsibility. We alert developers in a pro bono manner. Here is an idea of the volume of alerts we sent in 2020.

937,539

ALERTS WERE SENT PRO BONO

558,085

DEVELOPERS WERE ALERTED PRO BONO

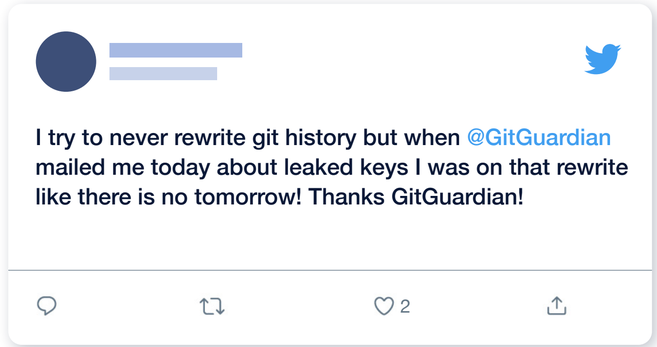
IT REPRESENTED

700,000

UNIQUE REPOSITORIES

860,000

UNIQUE COMMITS



What happens after a leak

GitGuardian’s algorithm reaction to a leak is **4 seconds** (Mean Time To Detect). The alert is sent right away.

25 minutes Median Time To React. The developer is on the front line of the issue, which allows to nullify most of the potential damage very quickly, if the developer takes immediate action after the alert.

When a secrets detection solution is in place, security teams also receive dual alerts to make sure they can follow up, remediate and report easily on security incidents.

> If you leave your keys to your house in the lock and you notice they are gone then you change the locks.

Allan Alford

Gitignore is not a Vault!

REMINDER

Gitignore allows you to tell what file you don’t want to commit. Your files containing your secrets should be listed in your gitignore file but your secrets should not be described in plain text in your gitignore file… **Hundreds of developers committed this mistake in 2020.**

Don’t bury the secret

REMINDER

If you search GitHub for “removed AWS key” you will see thousands of results. Removing a hardcoded secret and pushing a new commit only buries the secret in the history, making it harder for you to find but still accessible to attackers.

Recommendations

Companies can't avoid the risk of secrets exposure even if they put in place centralized secrets management systems. These systems are typically not deployed on the whole perimeter and are not coercitive as they do not prevent developers from hardcoding credentials stored in the vault. Solutions are available for them to automate secrets detection and put in place the proper remediation, but the market is far from mature on this subject. Companies need to scan not only public repositories but also private repositories to prevent lateral movements of malicious actors.

Some best practices can be followed to limit the risk of secrets exposure or the impact of a leaked credential:

- Never store unencrypted secrets in .git repositories
- Don't share your secrets unencrypted in messaging systems like Slack
- Store secrets safely
- Restrict API access and permissions.

Developers training programs should be put in place although these do not eradicate the risk of leaked credentials.

Following best practices is not sufficient and companies need to secure the SDLC with automated secrets detection. Choosing a secrets detection solution they need to take into account:

- Monitoring developers' personal repositories capacities
- Secrets detection performance* – Accuracy, precision & recall
- Real-time alerting
- Integration with remediation workflows
- Easy collaboration between Developers, Threat Response and Ops teams.

*Learn more about detection performance

To conclude

There are millions of commits per day on public GitHub, how can organizations look through the noise and focus exclusively on the information that is of direct interest to them? How can they make sure their secrets are not ending on their developers' personal repositories on GitHub? They can't avoid that developers have personal repositories, they need automated detection and efficient remediation tools.

In this state of secrets sprawl on GitHub analysis we focused on secrets although this is not the only sensitive information that can end up being publicly exposed: Intellectual Property, personal and medical data are also at risk. But this is for another State of Report!

ABOUT GG DETECTION ENGINE, DATA GATHERING & METHODOLOGY

GitGuardian's secrets detection engine has been running in production since 2017, analyzing billions of commits coming from GitHub. Since day one we began to train and benchmark our algorithms against the open source code. It allowed GitGuardian to build a language agnostic secrets detection engine, integrating new secrets or new way of declaring secrets really fast while keeping a really low number of false positives. We have developed the vastest library of specific detectors being able to detect more than 200 different types of secrets*.

*You can find the exhaustive list here

We are also collecting feedback from the alerts we are sending including the pro bono alerts:

- Explicit feedback when a developer or security team marks an alert as a false alert.
- Implicit feedback when a developer takes down a public repository or deletes a public commit a few minutes after we sent an alert.

Our secrets detection engine is

- **High precision:** We want to keep a low number of false positives to avoid alert fatigue.
- **High recall:** We want to keep a low number of secrets missed to keep our customers safe.
- **Fast:** While speed is less important than recall and precision our secrets detection engine is designed to be fast and scan a common git repository history under a minute.
- **Community and customer driven:** Our engine is constantly trained and improved by the feedback of the hundreds of thousands developers using our applications and by the feedback of our customers.

GitGuardian is solving the issue of secrets sprawling through source code, a widespread problem that leads to some credentials ending up in compromised places or even in the public space. The company solves this issue by automating secrets detection for Application Security and Data Loss Prevention purposes. GitGuardian helps developers, ops, security and compliance professionals secure software development, define and enforce policies consistently and globally across all their systems.
GitGuardian solutions monitor public and private repositories in real time, detect secrets and alert to allow investigation and quick remediation.