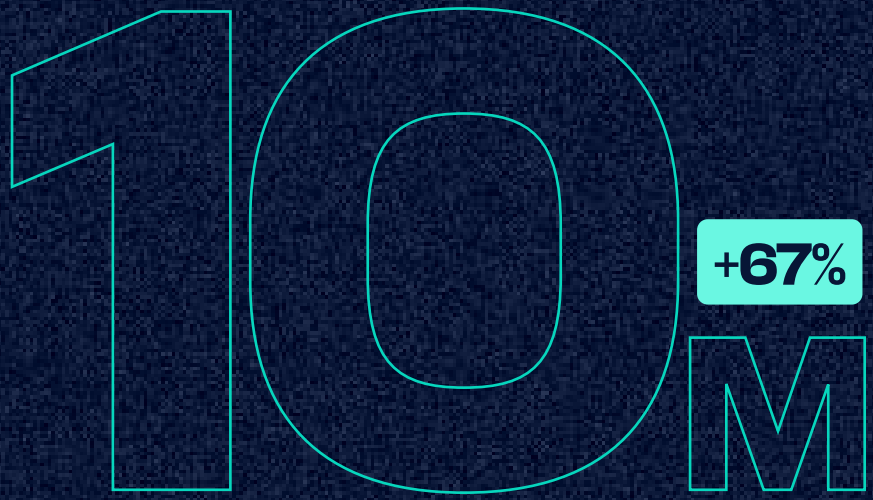
THE STATE OF SECRETS SPRAWL 2023



new secrets detected
in public GitHub commits in 2022

We have never detected as many secrets and secrets sprawl **has been accelerating** yearly since 2020.
Hard-coded secrets **increased by 67%** compared to 2021, whereas the volume of scanned commits rose by 20% (from 860M to 1.027B commits between 2021 and 2022).

Foreword

Hard-coded secrets have never been a more significant threat to the security of people, enterprises, and even countries worldwide. IT systems, open-source, and entire software supply chains are vulnerable to exploiting keys left by mistake in source code.

As the world digital footprint grows, millions of such keys accumulate every year, not only in public spaces such as code-sharing platforms but especially in closed spaces such as private repositories or corporate IT assets. In other words, secrets sprawl on GitHub is only the tip of the iceberg.

This wouldn't be so concerning if credentials theft weren't the most common cause of data breaches. The 2022 editions of Verizon's DBIR and the IBM Cost of a data breach reports highlighted that this attack vector had remains the top concern since 2021:

> **Use of stolen or compromised credentials remains the most common cause of a data breach. Stolen or compromised credentials were the primary attack vector in 19% of breaches in the 2022 study and also the top attack vector in the 2021 study, having caused 20% of breaches. Breaches caused by stolen or compromised credentials had an average cost of USD 4.50 million.[1]**

Secrets are not just any kind of credentials; they are the keys to obtaining privileged access to secure systems. Because of the leverage they provide, they are hackers' most sought-after information. However, many infosec incidents that occurred in 2022 pointed to how inadequate their protection is.

[1] From the IBM Cost of a data breach 2022

A look back at 2022 major incidents

Secrets are found in one way or another in most of the security incidents that happened in 2022. We can classify them into three categories:

Secrets exploited in an attack

Uber

An attacker breached Uber and used hard-coded admin credentials to log into Thycotic, the firm's Privileged Access Management platform. They pulled a full account takeover on several internal tools and productivity applications.

Sep. 15

circleci

An attacker leveraged malware deployed to a CircleCI engineer's laptop to steal a valid, 2FA-backed SSO session. They could then exfiltrate customer data, including customer environment variables, tokens, and keys.

Dec. 29

Secrets exposed publicly

android

Research reveals 18,000+ Android apps leak hard-coded secrets.

Sep. 1

TOYOTA

Toyota disclosed a contractor exposed a credential giving access to user data on GitHub for five years.

Oct. 7

Infosys

Tom Forbes disclosed Infosys leaked FullAdminAccess AWS keys on PyPi for over a year (and then 57 other AWS keys on PyPi).

Nov. 16

Stolen source code repositories

NVIDIA

NVIDIA source code is leaked by the "Lapsus$" group.

Feb. 25

SAMSUNG

200GB of Samsung source code is leaked, revealing 6,695 hard-coded secrets.

Mar. 7

Microsoft

250 Microsoft projects are leaked, revealing 376 hard-coded secrets.

Mar. 22

LastPass

LastPass source code is stolen, leaking credentials and keys used months later to access and decrypt storage volumes.

Aug.-Dec.

Dropbox

Dropbox disclosed that 130 stolen code repositories contained API keys.

Nov. 1

okta

Okta admitted a breach of its GitHub repositories resulting in source code theft.

Dec. 21

slack

Slack employee tokens are stolen and misused to download private code.

Dec. 7

1.027B commits scanned by GitGuardian

(+20% compared to 2021)

PUBLIC MONITORING

85.7M+ new repositories

(+20%)

About GitHub[1] in 2022

94M developers (+27%)

HCL (Hashicorp Configuration Language) is the fastest-growing language on GitHub.

[1]From the Octoverse 2022, see Methodology

How leaky was 2022?

10M
secrets occurrences detected in 2022
(3M unique secrets)

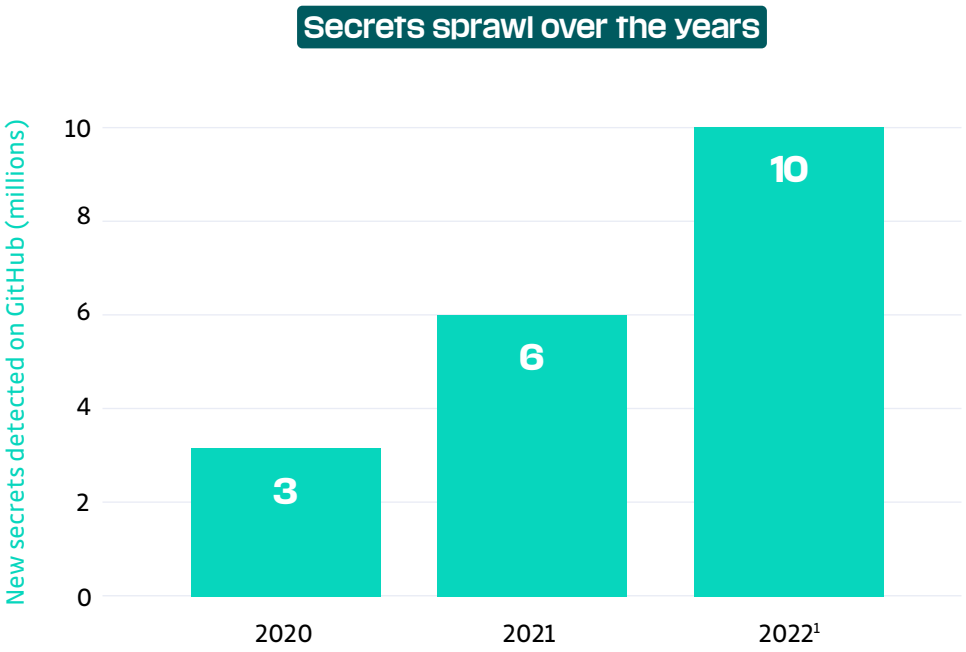
1 in 10
authors exposed a secret in 2022
To err is human. Of the 13.3M distinct authors who pushed code to GitHub in 2022, 1.35M accidentally exposed a secret.

5.5
commits out of 1,000 exposed at least one secret (+50%)

3.7% of repositories active during 2022 leaked a secret
- 61.2M repositories were active in 2022
- 2.27M of those repositories leaked a secret

GitHub's organic growth and the improvements of our detection engine (including +35 new detectors in 2022) partly explain the growth in the number of detected secrets. But all things equal, there is no doubt:

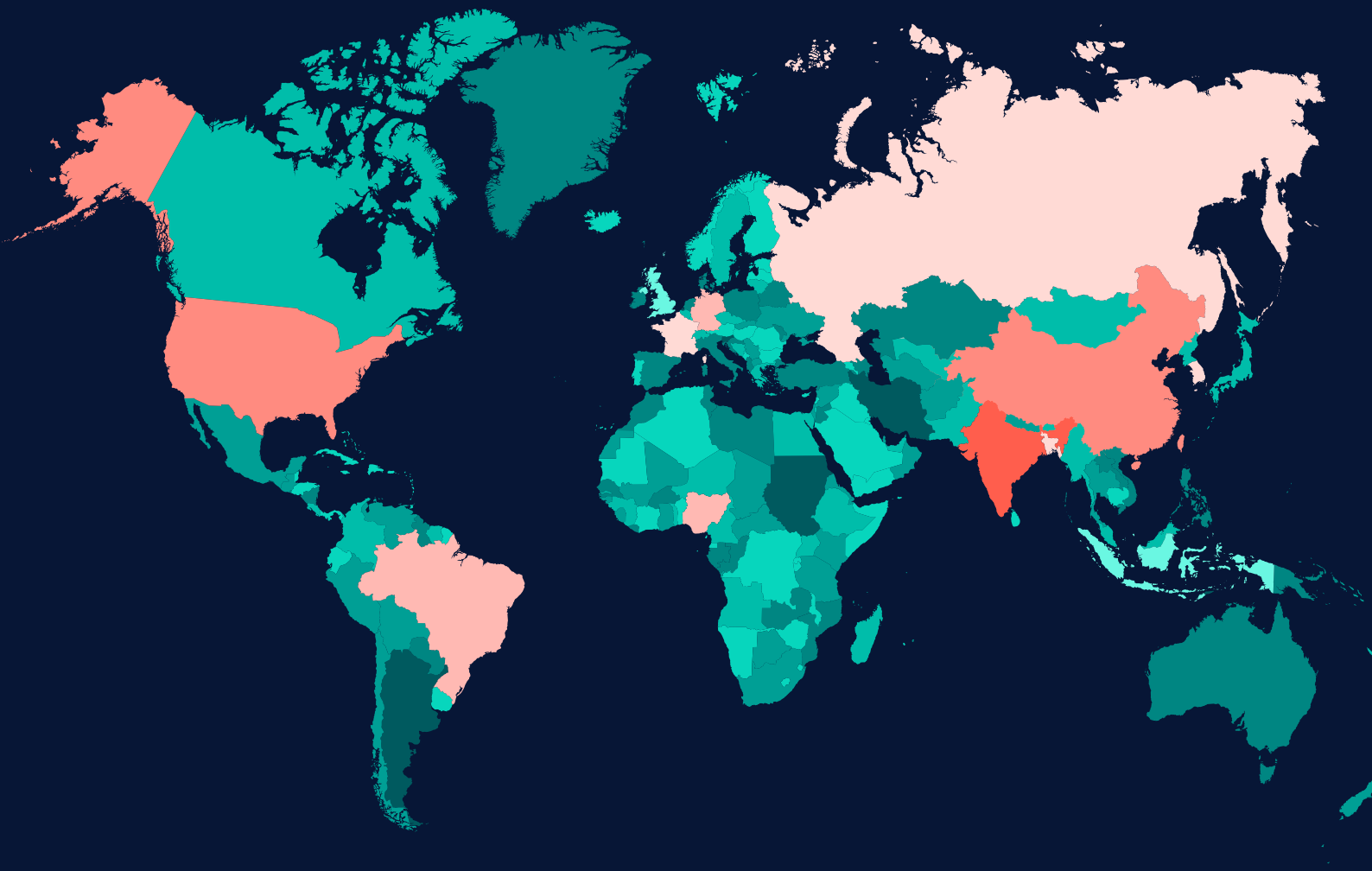
Secrets sprawl continues to expand worldwide.



[1] GitGuardian State of Secrets Sprawl report

Map of secrets leaks

From GitHub profiles mentioning location



01	India
02	China
03	USA
04	Brazil
05	Germany
06	Nigeria
07	South Korea
08	Bangladesh
09	France
10	Russia

Debunking a myth: hard-coding secrets is a junior developer mistake.

It is a common myth that hard-coded secrets are committed mainly by junior developers. **The reality is that this can happen to any level of developer, regardless of experience or seniority.** Hard-coding secrets is often a result of convenience rather than a lack of knowledge or skill. Senior developers, who might be simply testing a database connection or an endpoint, are under tremendous pressure to deliver quickly to meet business demands. They are responsible for many hard-coded secrets too! Therefore, it is essential to recognize that secrets sprawl is a **systemic issue**, not just a problem for junior developers.

shittysecrets.dev stories:

I was working on building a docker image in a new repo & needed it to be pushed to Dockerhub.

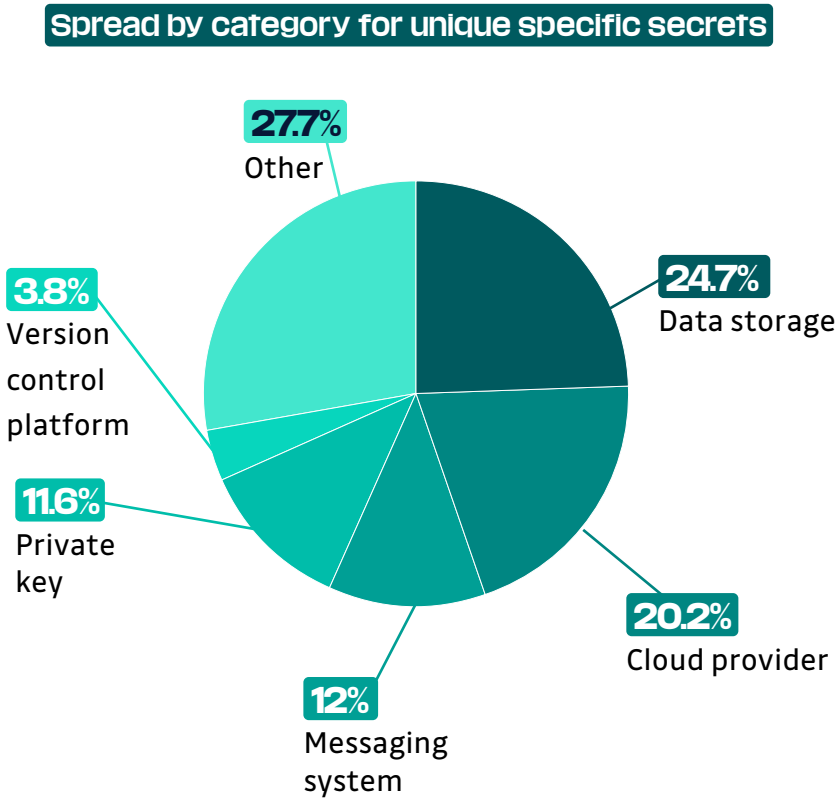
When I was happy with my Dockerfile, I ran git add. & pushed everything to my public github repo.

Unfortunately, I hadn't added a gitignore file to the repo, so my env was committed to Github, along with my Dockerhub credentials..

A coworker of mine, after some restructuring of env files and how they are loaded, commited an env file that was supposed to be ignored, with all the credentials for multiple integrations - postmark, b passwords, algolia keys, you name it

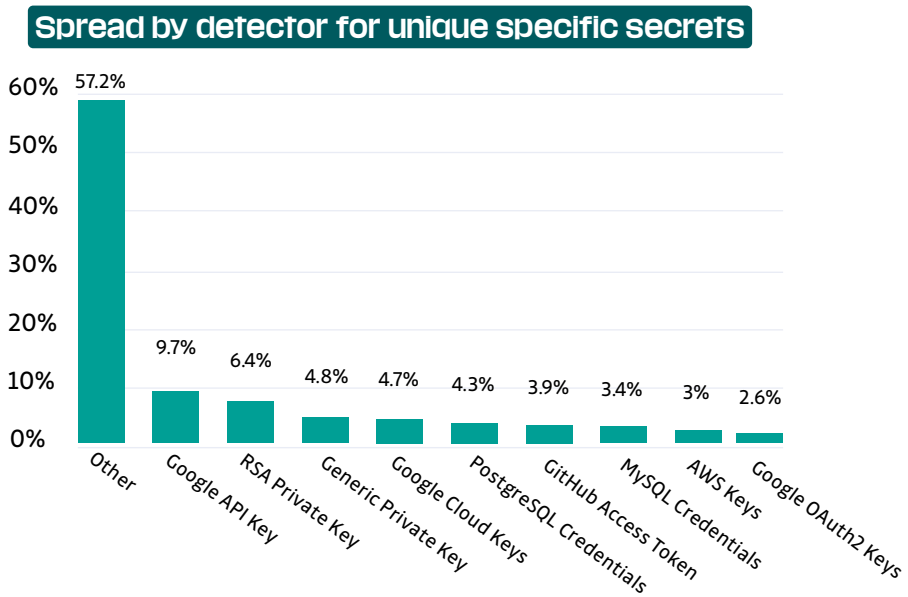
Well, a typical story of a temporary hardcoded solution, which got pushed to git and exposed everything for hell a lot longer time than it was originally expected.

Secrets categories



Secrets detectors

GitGuardian uses two classes of detectors: specific and generic.
Specific detectors match recognizable secrets, like an AWS access key or MongoDB database credentials. In 2022, our specific detectors accounted for **33%** of the secrets detected[1]. Here are some of the top specific secrets caught in 2022:

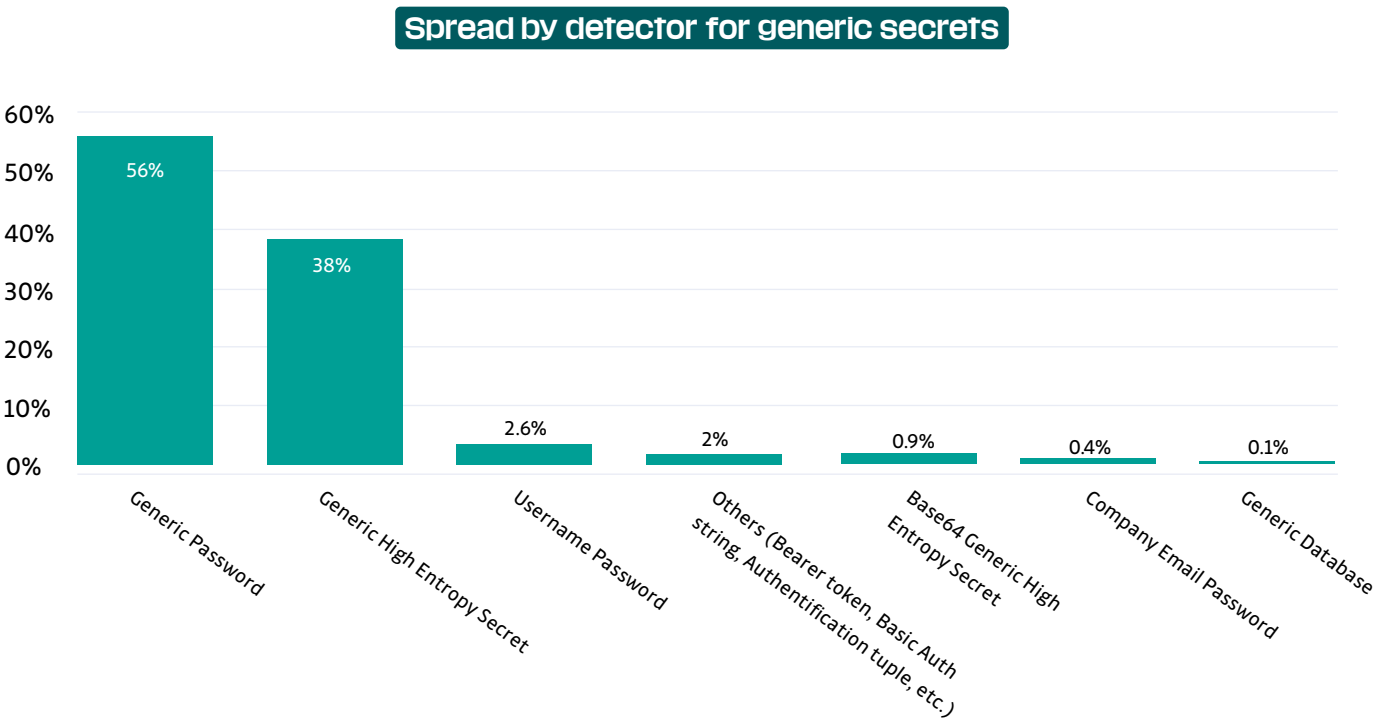


[1] See the Methodology section.

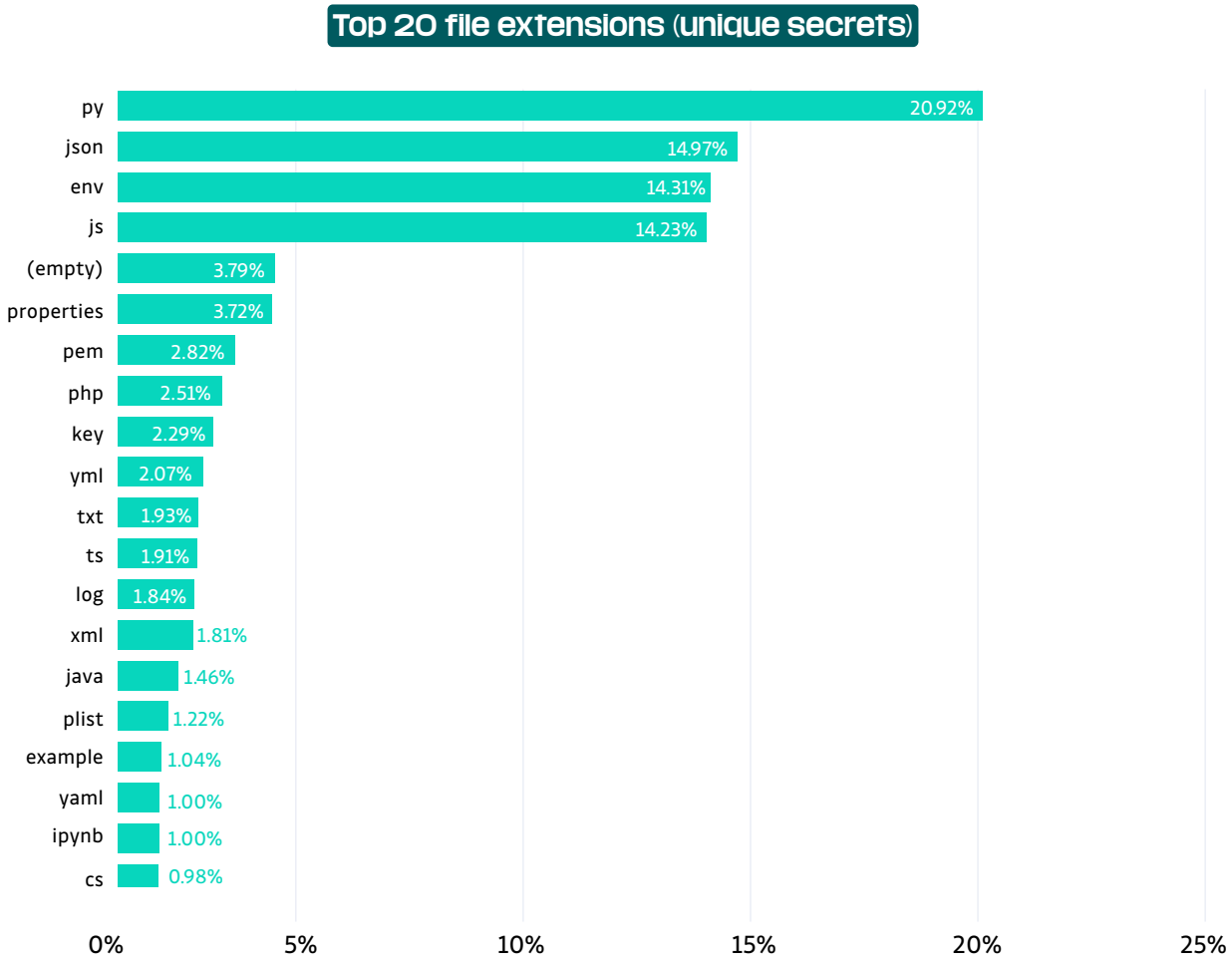
On the other hand, **generic detectors** match a broad range of secrets, for example, a company email and a password that would end up hard-coded in a file.

In a detection strategy, generic detectors are essential to ensure that no valid secrets fall through the cracks of specific detectors. To maximize precision and avoid false positives, each uses a carefully crafted set of conditions (regarding the filename, the file path, the surrounding context, etc.)

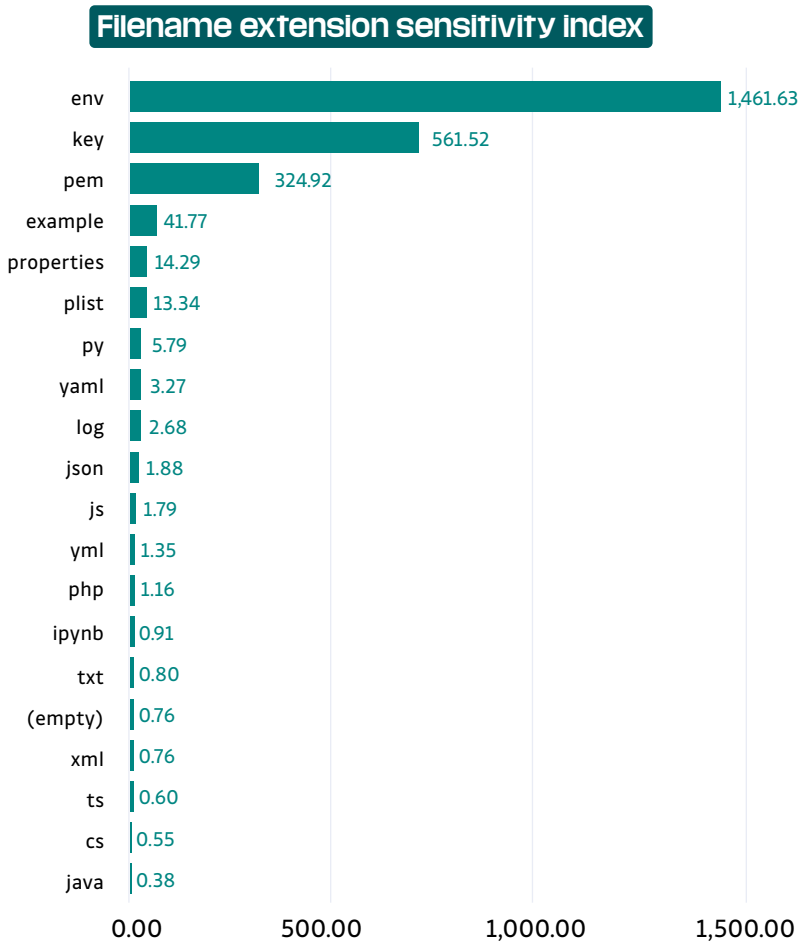
In 2022, they accounted for **67%**[1] of the secrets detected, which shows their importance.



[1] See the Methodology section.



Related to the filename extension frequency on GitHub, we can build a sensitivity index (the higher the index, the more secrets uncovered per file of this type):



> I would say, "Good luck," to someone who says secrets detection isn't a priority. Their priorities are probably wrong. One of the easiest ways for intrusion, as well as losing a lot of money in your company, is getting your secrets leaked somehow.

Andrei Predoiu
DevOps Engineer at a wholesaler/distributor with 10,001+ employees

How does secrets sprawl threaten software supply chain security?

When weighing the risk posed by secrets sprawl, it's essential to consider the ensemble of hard-coded plaintext secrets rather than individual secrets taken separately: the more secrets there are, the more potential attack vectors there are for a malicious actor.

An excellent example of this principle was demonstrated in research by Ronen Shustin and Shir Tamari from Wiz, a cloud security vendor.

They use the image of a **keychain** to better illustrate the concept:

> The keychain [...] symbolizes the collection of one or more scattered secrets the attacker finds throughout the target environment. Although both components (the forbidden link and the keychain) are individually unhygienic, they form a fatal compound when combined.

Harvesting plaintext credentials along their virtual tour of IBM Cloud Databases for PostgreSQL, they discovered a vulnerability, dubbed "Hell's Keychain," that combined three exposed secrets and a network misconfiguration.

This vulnerability would have allowed them to compromise IBM Cloud's internal image-building process to finally read and modify the data stored in every instance of IBM Cloud Databases for PostgreSQL databases.

In other words, they would have been able to expose IBM Cloud's customers to a supply-chain attack.

> Heil's Keychain illustrates how scattered plaintext credentials across your environment can impose a huge risk on your organization by impairing its integrity and tenant isolation. Moreover, the vulnerability emphasizes the need for strict network controls and demonstrates how pod access to the Kubernetes API is a common misconfiguration that can result in unrestricted container registry exposure and scraping.

The third plaintext credential in the keychain was a hard-coded secret in a container images' metadata, allowing them to infiltrate IBM Cloud build servers.

> **Finally, we were reminded of the value of secret scanning.** Although in previous cases, our team managed to violate tenant isolation by exploiting vulnerabilities in neighbor-tenant instances or the control plane, in the case of IBM Cloud Databases for PostgreSQL, the Achilles heel was improper secrets management. Regardless of how strong your organization's security measures are, it faces a huge risk if plaintext credentials are scattered across its environment.

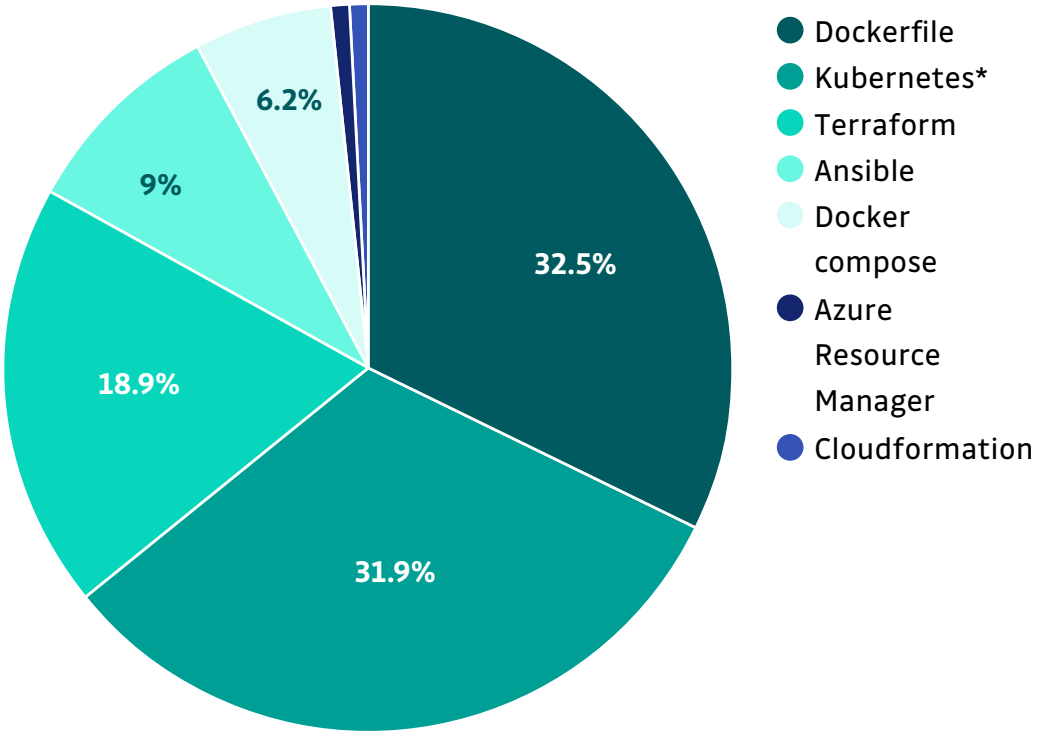
From code to cloud: Infrastructure as code

Infrastructure as code (IaC) is the abstraction layer used to declare the final/desired state of IT infrastructure with code: servers, storage, databases, networks, and all the basic configurations (DNS entries, firewalls, etc..).

Empowering developers, SREs, and platform engineers to collaborate faster and more effectively than ever before, IaC has become a staple in cloud-native deployments.

With a **24% CAGR** projected from 2022 to 2030[1], infrastructure as code adoption is in full swing. This is also evident on GitHub, where IaC-related contributions[2] increased by **28% in 2022.**

Distribution of IaC filetypes on GitHub in 2022



*Kubernetes-related .yaml or config files

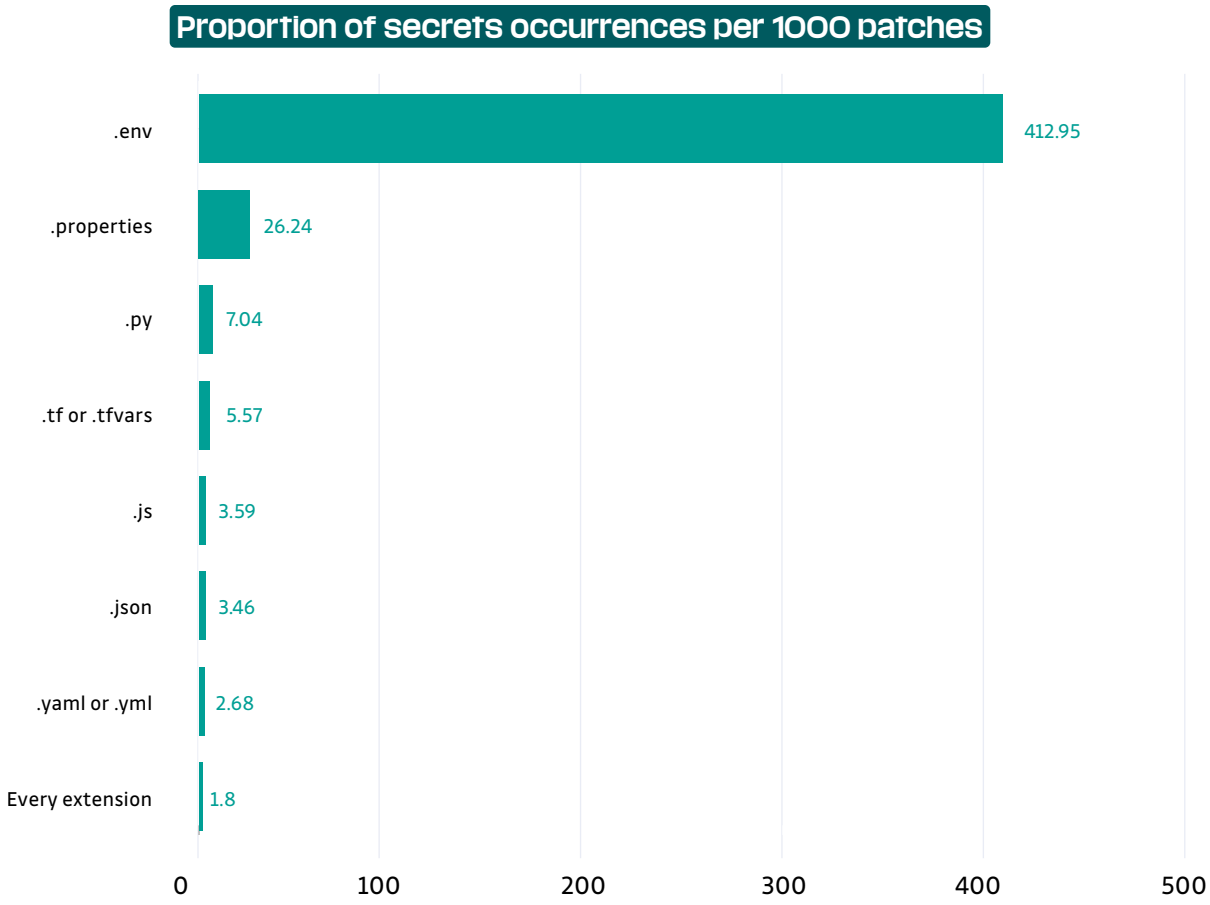
IaC has made infrastructure workflows shift left. Cloud security is shifting left with it.

[1] From Firefly's State of IaC 2023

[2] Patches to IaC files

Shifting left: Infrastructure as code security

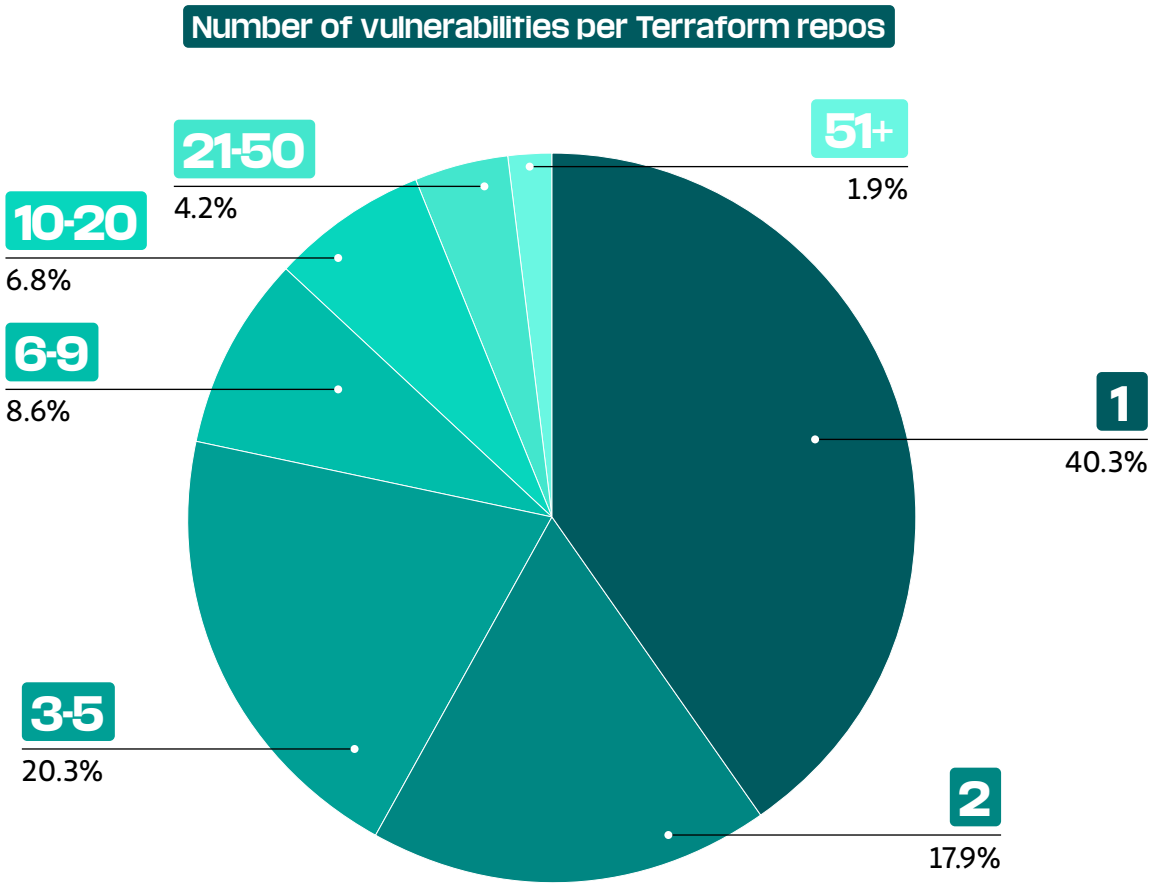
Code managing infrastructure can lead to uncaught mistakes and security vulnerabilities.
We found that Terraform files had an average of 5.57 occurrences of secrets (2.11 unique secrets) per 1,000 patches, which is **3x** the average for all file extensions.
The graph below shows that all file extensions can hide secrets. Therefore, it is essential to scan them all.



Moreover, secrets are not the only risk present in IaC files.

A single misconfiguration in an IaC manifest can break a security policy and make the deployed infrastructure vulnerable to attacks.

Infrastructure as code is an entirely new attack surface to protect. We estimate that **21.52% of all Terraform[1] repositories have one or more security policy vulnerabilities.**



The most common IaC vulnerabilities are:

- **Networking misconfigurations:** unrestricted egress or ingress traffic can expose assets to attacks such as remote code execution. The use of HTTP instead of HTTPS is also frequent.
- **Data exposure misconfigurations:** S3 buckets without encryption can lead to data leakage.
- **Secrets:** exposing a sensitive environment variable in the configuration can lead to a plaintext credentials leak.
- **Permission misconfigurations:** using the default service account on a compute instance allows an attacker to spread through the network.

[1] Based on a private repositories dataset where 7.5% of repos contained Terraform files, see Appendix

> [Secrets sprawl] will continue to grow as a problem since a lot of developers forget that IT security aspect. They just copy and paste stuff, then leave it in the code and forget about it. That is how attacks happen: somebody slipped, making a mistake or misconfiguration.

Edvinas Urbasius
IT Security Specialist – SOC analyst at a wholesaler/distributor with 10,001+ employees

Secrets in Docker images

Docker images are one of the largest unmonitored attack surfaces. Images are complex, layered structures that present more risk than meets the eye.

As highlighted earlier, even commercially available images distributed by reputable cloud vendors can leak secrets through their build process (see p.15).

A study we conducted last year on Docker Hub showed that more than 4,000 secrets (1,200+ unique) were hard-coded in a 10,000 image sample.

Also, **4.62% of the Docker Hub images sample exposed one or more secrets.**

It is clear today that scanning for secrets in images (whether home-built or third-party-provided) is essential to guard against supply chain attacks.

In December 2022,

was called 600K times to scan 17,300 Docker images for secrets.

Measuring time-to-hacked: our experiment with honeytokens

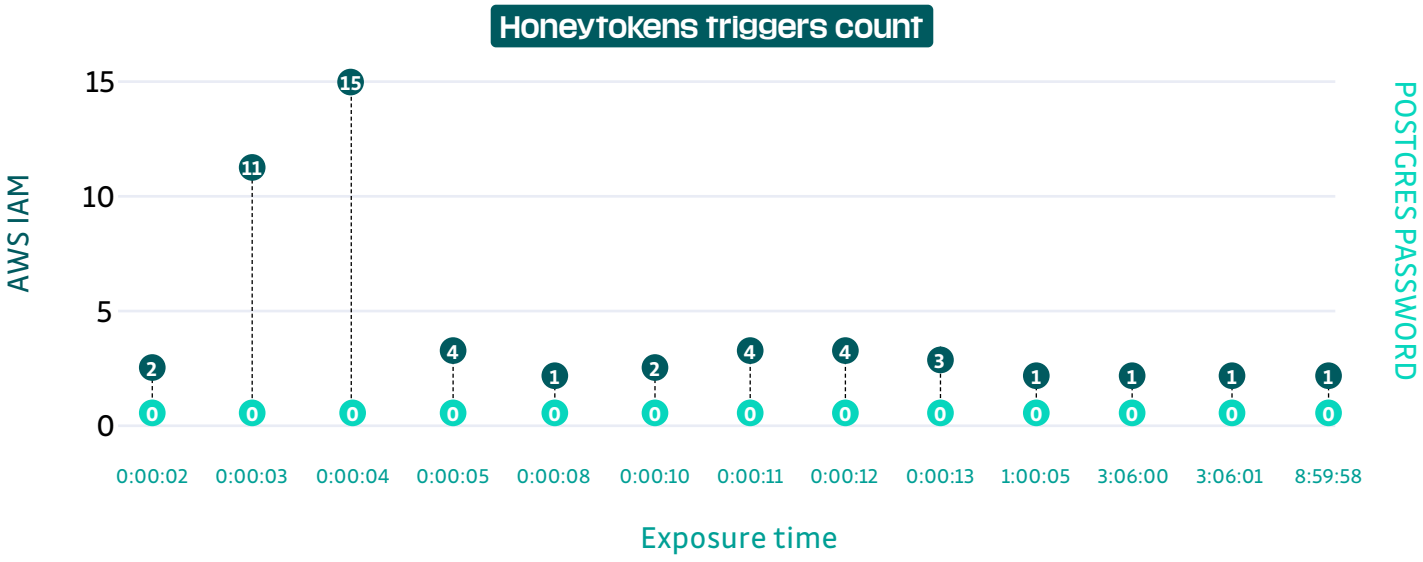
GitGuardian is not the only service monitoring GitHub in real-time. Malicious bots are also looking to exploit sensitive information as soon as it is exposed.

When a honeytoken (also called a canary token) is deliberately exposed as an experiment, it's easy to realize why **GitHub-exposed data should be considered compromised.**

What is a honeytoken?

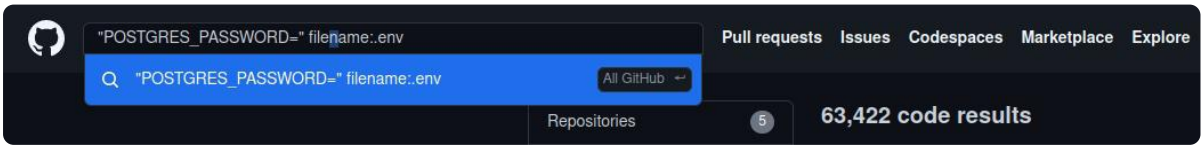
A honeytoken is a resource that is monitored for access or tampering. Usually, honeytokens come in the form of a URL, file, API key, email, etc., and trigger alerts whenever someone (presumably an attacker) trips over them, producing an Indicator of Compromise (IoC) for security teams.

Not all credentials are born equal: for this experiment, we leaked two types of honeytokens on GitHub: a set of AWS IAM credentials and PostgreSQL credentials. We then plotted the number of times they were used by third-parties over multiple hours:



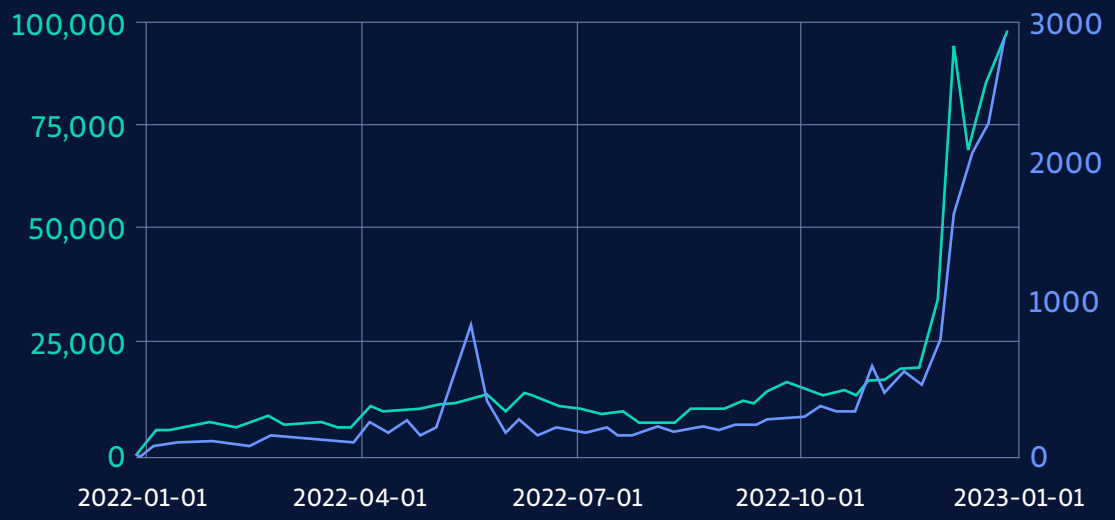
The first attempt occurred 2 seconds after publicly exposing the credentials. In total, 24 IPs (22 if we remove AWS and GitGuardian) requested AWS IAM information using the credentials in the first 9 hours after the leak.

Although no IP used the PostgreSQL credentials in the elapsed time, **that doesn't mean that they are any less problematic:** on the contrary, bounty hunters are actively looking for "dorks" on GitHub, which are easily recognized patterns to identify low-hanging fruits:



Fun facts

ChatGPT and OpenAI are (without surprise) the buzzwords of 2022:



Commit some ❤️

Remove leaked secret
Thanks gitguardian.com for the notice and thanks slack for the instant deactivation of the leaked token!
master

sanitizing API keys in example app manifests
not rewriting history to remove them because i've revoked and switched phones since then so the API keys aren't actually usable any more.

thanks **@GitGuardian**!
master

Removed API Key (oops, thanks GitGuardian!) and moved location to be ...
...a requested field by the console app for user testing
master

remove clientid api key
gitguardian pls be happy
main (#23)
v.3.0.0

... okay, but first, revoke that key!

> Secrets being used to access resources is probably one of the most common ways to be involved in a high-profile breach these days. If you are not detecting secrets in code, then every developer's machine is a security breach waiting to happen.

Jon-Erik Schneiderhan
Senior Site Reliability Engineer at a computer software company with 501-1,000 employees

Insight from DARKOWL™ The hidden economy of credentials on the darknet

When it comes to secrecy, there is one place that cannot be ignored: the darknet.

Darknet 101

The darknet, also referred to as the dark web, is a layer of the internet designed specifically for anonymity. It is more difficult to access than the surface web and is accessible only via using specialized software or network proxies – specifically browsers supporting special protocols. Users cannot access the darknet by simply typing a dark web address into a web browser. Adjacent to the darknet are other networks, such as instant messaging platforms like Telegram and the deep web (non-public web).

Due to its inherently anonymous and privacy-centric nature, it facilitates a complex ecosystem of cybercrime and illicit goods and services trade. The dark web is a thriving ecosystem within the global internet infrastructure that many organizations struggle to incorporate into security posture. Still, it is an increasingly vital component for organizations with forward-thinking strategies.

“Secret” data, including tokens and keys found on open repositories such as GitHub, are easily re-sold (or sometimes shared for free) on the darknet and deep web.

How does it work?

In some cases, such as that of the deep web site BreachForums, leaked data is offered for download via vendor-specific currency in the form of generally inexpensive credits. Another way to accrue credits is to share other breached data for users to download. Users can also gain credits to purchase these stolen data packets by commenting on and engaging with other user posts. Both of these aspects of the darknet breach economy encourage discovering and re-sharing of sensitive user data and creativity in exploiting previously-exposed information.

Consequently, **an extensive amount of sensitive information is available for download on the darknet and deep web, ranging in prices from free to several thousands of dollars.** While such free exchanges may challenge the use of the word “economy’” – it is crucial to remember how this stolen information is used. **The vast majority of cases result in hackers gaining illicit access to user accounts and either exploiting them for financial gain or using them to pivot into corporate network access to carry out more large-scale attacks.**

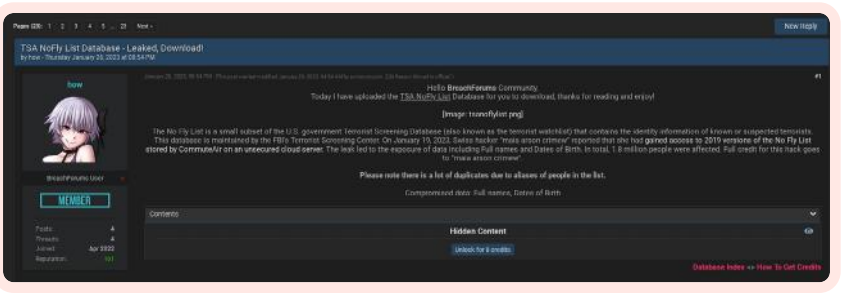
Examples

Verizon

The below screenshots demonstrate a typical database leak offering. This breached information has been offered entirely free (no digital currency or credits are required to download).

TSA No-Fly List

Hackers offered the recently leaked US TSA No-Fly list in exchange of credit tokens on a deep web forum.



DoorDash User Accounts

While the token or credit-based nature of the darknet economy does support “free” or more covert methods of exchanging Secret data (such as credits), this is not always the case. For example, as demonstrated by this Doordash database, containing username and password pairs for over 650,000 individuals, offered at a starting bid of US$ 10,000 in August 2022.

Are hackers exchanging secrets on the darknet?

The shift towards everything-as-an-API in the commercial landscape echoes what DarkOwl analysts see in the darknet.

Discussions around stealing and selling API keys are a relatively new phenomenon in the darknet over the last couple of years that we expect to continue to grow. Threat actors who are looking to facilitate the wider distribution of malware through supply chain compromises have also discussed credentials and pivot points sourced from open repositories.

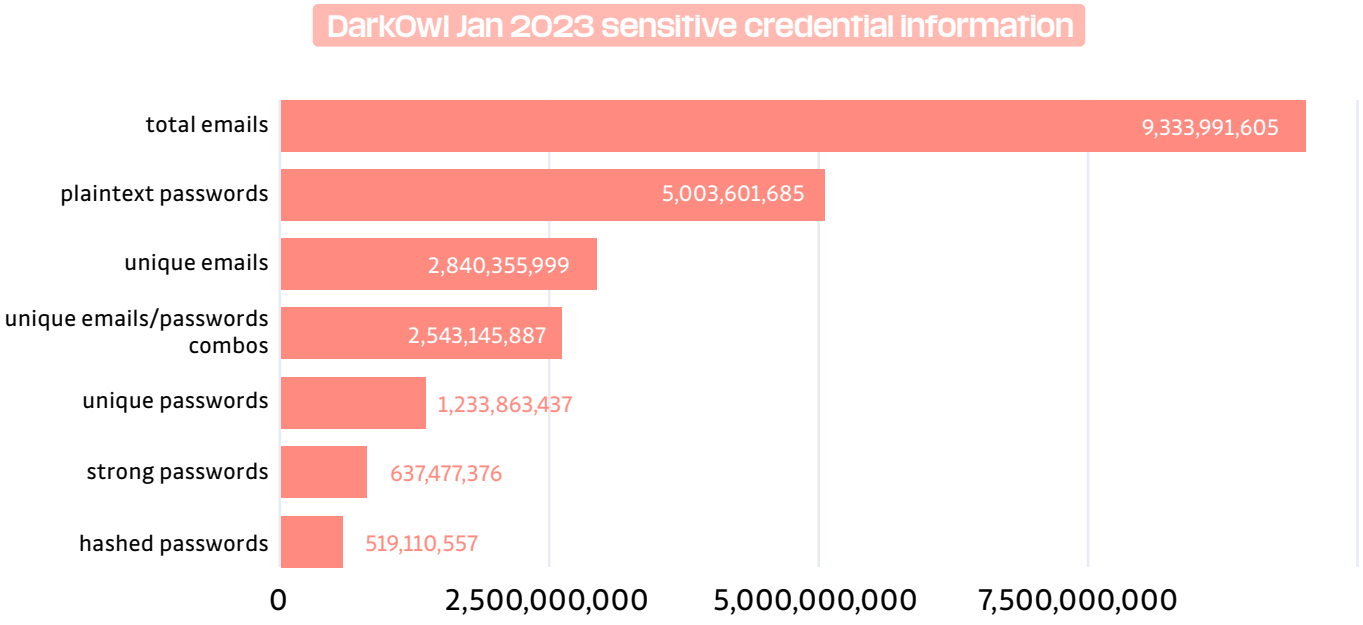
Developers and security researchers worldwide have been equally appalled and conflicted by the intentional sabotage of open-source software packages. Many are concerned about the reputational damage these incidents cause to the open-source software development movement.

How many credentials are for sale on the Darknet?

From DarkOwl's data (Jan 2023):

> While it is unclear exactly how many total API keys have been leaked on the darknet, DarkOwl has identified 11,246 unique exposed API keys for Amazon Web Services (AWS) alone.

While it is impossible to grasp the total size of the underground digital economy, DarkOwl does have insight into certain entities that indicate the potential for exploitation, including sensitive credential information. DarkOwl's AI and analyst-augmented database is updated in near real-time and collects from hard-to-reach places of the darknet, including authenticated forums, ransomware sites, chat platforms, open server databases, and breach/leak exchanges. As of January 2023, their records detected:



> If a colleague in security said to me that secrets detection is not a priority, I would say that's a mistake. Most of the big security problems come from either social engineering attacks or credential stuffing. So it's really important to know that your engineers and your employees are going to leak secrets. That's life. Most of the time, it's due to mistakes. But if it happens, we need to act on it. The more engineers there are, the more there is potential for leaks to happen.

Théo Cusnir

Application Security Engineer at Payfit

Taming Secrets Sprawl in the SDLC

Secrets can get exposed in many ways. **Source code is an asset that can quickly be lost to subcontractors and, of course, source code theft.** Developers must often be reminded of the security risks of storing sensitive information in source code and other software artifacts. Otherwise, they may be tempted to think their private accounts and assets will stay out of the reach of attackers. They may also be unaware of the need to use secure methods for storing secrets, such as environment variables or vaults – or simply not appropriately trained in the workplace.

Like many other security challenges, poor secrets hygiene involves the usual trifecta of people, processes, and tools. Organizations serious about taming secrets sprawl must work on all these fronts simultaneously.

People

The first part of addressing poor secrets hygiene in software development is to ensure that Dev, Sec, and Ops are involved in the process and trained on secure coding best practices, including the proper use of secrets. Additionally, the stakeholders should be held accountable for adhering to these best practices and have clear avenues to report potential security violations and contribute to their fixing.

Processes

Establishing a secure software development lifecycle (S-SDLC) is critical in preventing the misuse of secrets in software development. It should include secure coding guidelines, processes for provisioning and managing secrets, and processes for securely distributing them across various environments.

Tools

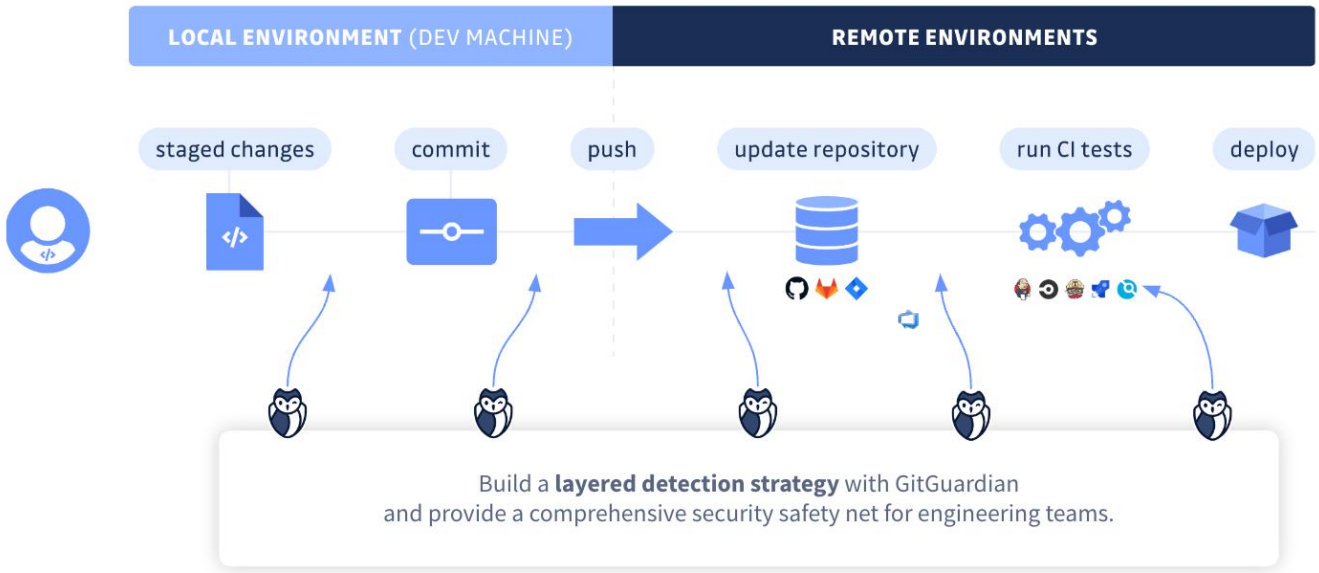
Using technologies such as detection, encryption, tokenization, and dynamic key rotation can help secure secrets in software development. Additionally, systems should be designed focusing on built-in, not bolt-on, security.

What's a good strategy for mitigating hard-coded secrets?

Hard-coded secrets detection and mitigation can be shifted left at various levels to build defense-in-depth across the development cycle. To gradually move to a "zero secrets-in-code" policy, here is a step-by-step:

- Start by monitoring commits and merge/pull requests in **real-time** for all your repositories with native VCS or CI integration (shift to the team level).
- Enable **pre-receive checks** to harden central repositories against leaks, and "stop the bleeding."
- Educate about using **pre-commit scanning** as a seatbelt (shift to the individual level).
- Plan for the longer term: develop your **strategy** for dealing with incidents discovered through the historical analysis.
- Implement a secrets' **security champion** program.

Continuously scan incremental code changes at every stage of the SDLC



Conclusion

Secrets sprawl continues accelerating and there will never be a better time to act. With 10 million secrets discovered in public GitHub commits and countless more silently accumulating behind closed doors, this is one of the biggest threats to the security of the digital world.

With attackers recognizing that **compromising machine or human identities yields a higher return on investment**, especially when targeting software supply chain components, this trend will unfortunately not dissipate soon.

Companies now understand that source code is one of their most valuable assets and must be protected. **To scale these efforts will require bringing security and development closer** and advancing towards an application shared responsibility. It is valid for the detection and remediation of incidents as well.

The very first step is to get a **clear audit of the organization's security posture regarding secrets:** where and how are they used? Where do they leak?
How to prepare for the worst? You can start right away by taking the (anonymous) secrets management maturity questionnaire and learn where to go from there with this white paper.

Or you can contact us to get a complimentary audit of your company's exposed secrets on GitHub.

Organizations must be prepared for secrets sprawl and have the right tools and resources to promptly detect and remediate any issues. It's time to take action!

About GitGuardian

GitGuardian is a code security platform that provides solutions for DevOps generation. A leader in the market of secrets detection and remediation, its solutions are already used by hundreds of thousands of developers.

GitGuardian helps developers, cloud operation, security, and compliance professionals secure software development and define and enforce policies consistently and globally across all systems.

GitGuardian solutions monitor public and private repositories in real-time, detect secrets, sensitive files, IaC misconfigurations, and alert to allow investigation and quick remediation. Additionally, GitGuardian's Honeytoken module exposes decoy resources like AWS credentials, increasing the odds of catching intrusion in the software delivery pipeline.

Gitguardian is by far

THE N°1 SECURITY APPLICATION ON THE GITHUB MARKETPLACE

and is trusted by leading companies, including Instacart, Snowflake, Orange, Bouygues Telecom, Iress, Maven Wave, NOW: Pensions, DataDog, and PayFit.

Learn more about GitGuardian:

Website | Public Monitoring | Internal Monitoring

Appendix

Definitions

Secret

A secret can be any sensitive data we want to keep private. When discussing secrets in the context of software development, we refer to digital authentication credentials that grant access to services, systems, and data. These are most commonly API keys, username and password combos, or security certificates. In this report, secrets refer to credentials hard-coded in cleartext (not encrypted).

Occurrence

When our detection engine detects a hard-coded secret, it becomes an occurrence. A single incident generally encompasses multiple occurrences, which are the various locations across files or repositories where the secret was identified. Occurrences map to the magnitude of the sprawl and correlate to the amount of work needed to redistribute the secret after it has been rotated. Occurrences can be assimilated to a technical debt.

Secret incident

A secret incident is a uniquely identified security event that has been determined to impact the organization and necessitates remediation. An incident often has multiple occurrences across files or repositories.

Supply chain security

A software supply chain is a logistical pathway that covers anything required to build a software artifact. It is the set of assembled components, building tools, and processes from source code to production deployment. Supply chain security is about securing each link in the chain by ensuring that components supplied by third parties have not been compromised and comply with security requirements.

Infrastructure as code security

Infrastructure as code has become the favorite way for the DevOps generation to manage and provision infrastructure, especially in the cloud. It's also a new responsibility since uncaught mistakes can result in misconfigurations and, in the end, important security failures. Securing infrastructure as code requires bridging the gap between Security, Operations, and Development and leveraging automation to build efficient guardrails.

Methodology

Octoverse 2022

The data from the GitHub Octoverse is for the period 0/01/2021 to 09/30/2022. GitGuardian data is for the period 01/01/2021 to 12/31/2022.

Specific and generic detectors

In 2022, GitGuardian's 350+ specific detectors collected, in total, 3.016M unique secrets on public GitHub. Secrets detection is always a tradeoff between precision and recall. To improve the accuracy of our study, we manually removed 626k (specific) secrets from this count: 399K django_secret_key (they often correspond to default or local settings) and 227K googlecloud_keys, which could be considered outliers.

The proportions do not represent the mix of specific vs. generic secrets found on GitHub. Our engine detects approximately the same volume of generic and specific secrets.

IaC Vulnerabilities

Our study data set comprised 6,311 git repositories hosting Terraform files, of which 1,358 were affected by one or more vulnerabilities (21.52%). In total, 12,046 Terraform vulnerabilities were detected.

About DarkOwl

DarkOwl is a Denver-based information security company specializing in darknet OSINT tools. Using machine learning and human analysts, DarkOwl continuously collects and indexes data from the darknet, deep web, and high-risk surface sites. This includes Tor, I2P, IRC, Telegram, ZeroNet, as well as transient paste sites, deep web criminal forums, and other high-value sites. Their data is collected and stored in near real-time, allowing it to be queried in a safe and secure manner without having to access the darknet itself.
DarkOwl's UI and API products were designed to make previously hard-to-access content actionable for cybersecurity companies, organizations and governments seeking to mitigate risk from potential threats such as ransomware or malicious actor activity.

For more information, visit www.darkowl.com