



Fresh Secrets From The Docks

Lessons Learnt From Analysing **15 Million** Public DockerHub Images



Guillaume VALADON - @guedou

GitGuardian

\$ whoami



Guillaume

Cybersecurity Researcher

Scapy maintainer

previously at Quarkslab, **ANSSI**...

editor-in-chief of the french **MISC magazine**

looking for secrets in unusual places



Network Packets Manipulation

<https://www.scapy.net/>

Secrets Security 🤝 NHI Governance

0% LEAKED SECRETS
Secrets Security

INTEGRATED SOURCES

Package Registries

Container Registries

Code Repositories

CI/CD Pipelines

Messaging Systems

Ticketing Systems

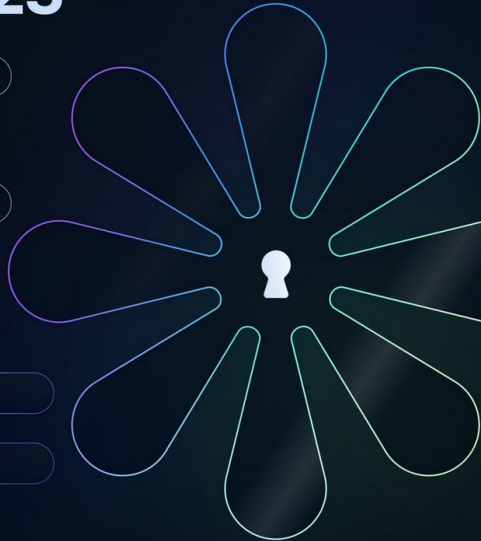
Logs

Knowledge Database



GitGuardian

THE STATE OF **Secrets Sprawl** 2025



Data analysis by **GitGuardian**

100% MANAGED IDENTITIES & SECRETS

NHI Governance

INTEGRATED SOURCES

IAM Cloud Providers

Secret Managers

Cloud Infrastructure

Deployment Tools

Container Registries

Third-Party Apps



NHI Governance

check it out!

Secrets Lifecycle

Discovery, Vaulting, Ownership



GitGuardian

Attackers Are Looking For Secrets

APPLICATION SECURITY

PyPI Packages Found to Expose Thousands of Secrets

GitGuardian discovered roughly 4,000 secrets in nearly 3,000 PyPI packages, including Azure, AWS, and GitHub keys.



By [Ionut Arghire](#)
November 14, 2023

[Home](#) » [Security](#)

Websites exposing over a million secrets, leaving visitors at risk

Last updated: 29 May 2024

[Home](#) > [News](#) > [Security](#) > Nearly 12,000 API keys and passwords found in AI training dataset

Nearly 12,000 API keys and passwords found in AI training dataset

By [Ionut Ilascu](#)

March 2, 2025 10:23 AM 0

MITRE ATT&CK Paths Examples



Secrets Categories Exploited by Attackers

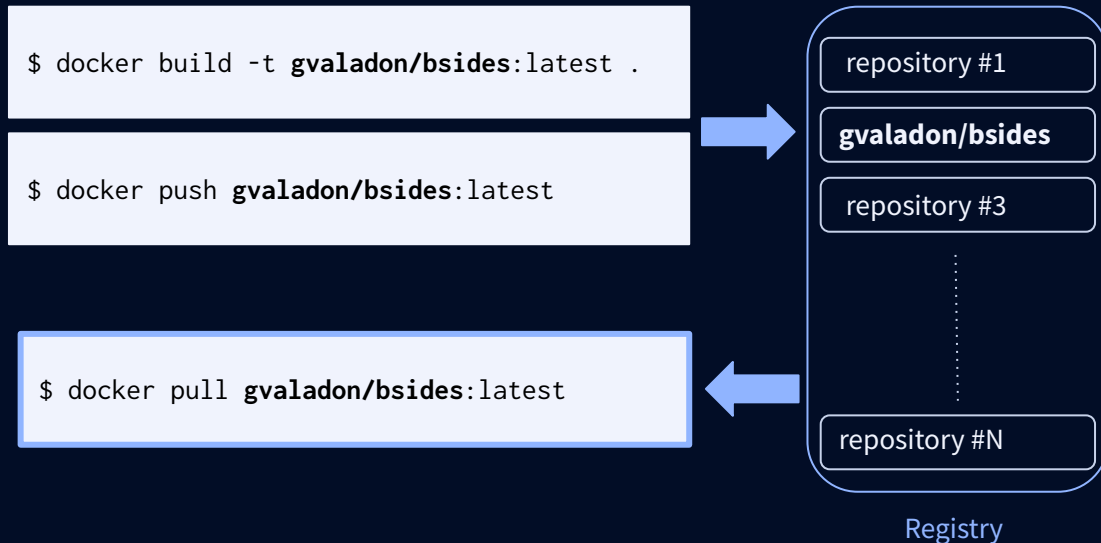


01

Scanning a Docker Image

Everything you need to know about the structure of an image to look for secrets.

Interacting With A Docker Registry



Steps?

-
0. build
 1. push
 2. **pull**

Pulling a Docker Image

```
$ docker pull gvaladon/bsides:latest
latest: Pulling from gvaladon/bsides
cd271e318: Downloading [=====>] 16.66MB/144.1MB
fcbb27488: Download complete
a2b76ca18: Downloading [=====>] 19.37MB/28.15MB
```

what are these **values**?
whats is inside a Docker image?

From a Dockerfile to a Docker Image

```
FROM ubuntu:latest
```

```
COPY . /src
```

```
RUN pip install -r requirements.txt
```

```
EXPOSE 8080
```

```
ENV NAME=KubeCon
```

```
CMD ["python", "app.py"]
```



Image X-ray

JSON manifest

- list of layers (aka tarballs)
- JSON config

everything stored in **blobs**

- SHA256 identify the content



Inspecting images

<https://github.com/containers/skopeo>



Scanning Docker images for secrets means
accessing **JSON files and tar archives** (aka blobs).

Manual Secrets Scanning

```
$ ggshield secret scan docker gvaladon/bsides:latest

>> Secret detected: AWS Keys
Validity: Invalid
Occurrences: 1
Known by GitGuardian dashboard: NO
Incident URL: N/A
Secret SHA: 4d3bb359d157287dbc19a5af9e1019e2547ee8aad651a88973dcf62d0776dd88

8 8 |
9 9 | def aws_upload(data: Dict):
10 | +     database = aws_lib.connect("AKIA*****WSF5", "hjshuk5*****89sjkja")
...
10 | -     database = aws_lib.connect("AKIA*****WSZ5", "hjshnk5*****89sjkja")
    |         |__client_id__|
10 | -     database = aws_lib.connect("AKIA*****WSZ5", "hjshnk5*****89sjkja")
    |         |_____client_secret_____|
11 11 |     database.push(data)
```



Scanning for secrets

<https://github.com/GitGuardian/ggshield>



Validity Checks

Test the secret with a known endpoint

and check the **HTTP status code**.

GitHub Token

```
$ curl --write-out "%{http_code}" --silent --output /dev/null \  
  --header "Authorization: Bearer $VALID_TOKEN" \  
  https://api.github.com/user
```

200

```
$ curl --write-out "%{http_code}" --silent --output /dev/null \  
  --header "Authorization: Bearer $INVALID_TOKEN" \  
  https://api.github.com/user
```

401

02

Scanning a Docker Registry

Methodology and API calls to retrieve configuration files and layers content.

Methodology

Four simple steps

combined to **scan all Docker images** from a single registry.



Corresponding Docker Registry API

Pulling an image manually

leaving authentication aside, these API calls retrieve Docker images content.



03

Scanning Docker Hub

Techniques to retrieve blobs, and look for secrets.

Enumerating Docker Hub Repositories

`/v2/_catalog` is not available

Docker Hub `web search` returns a maximum of
10k results per keyword

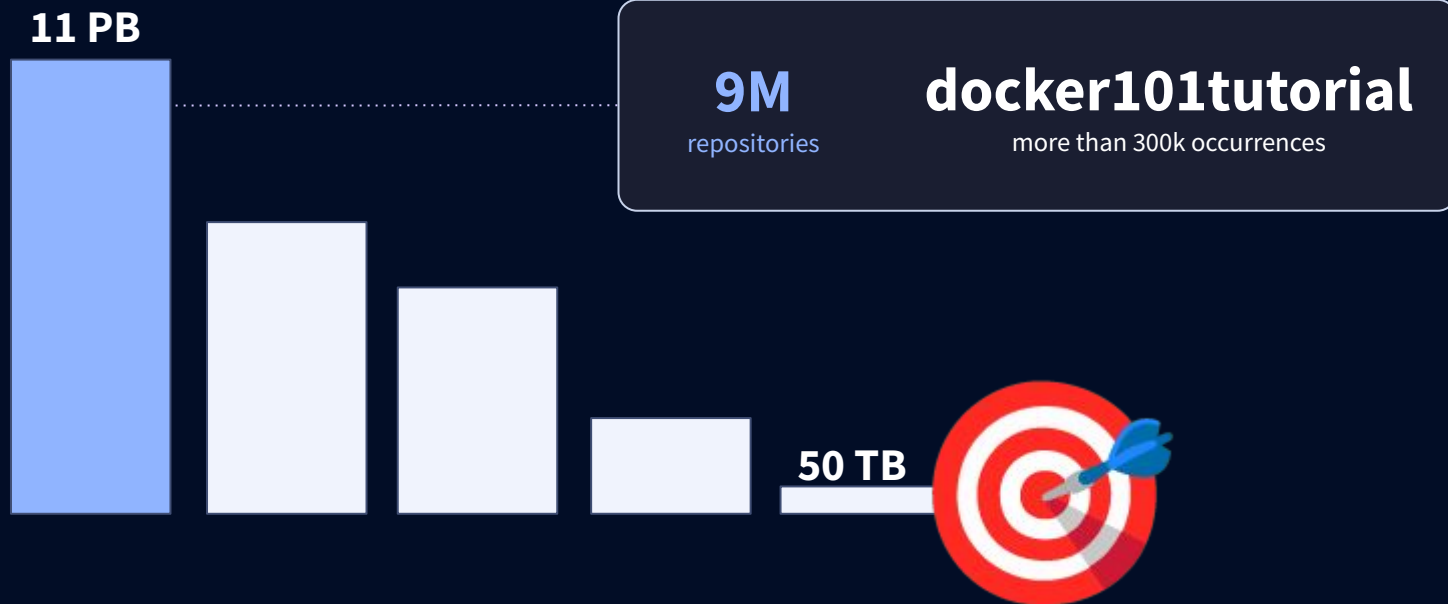


 ~1 day

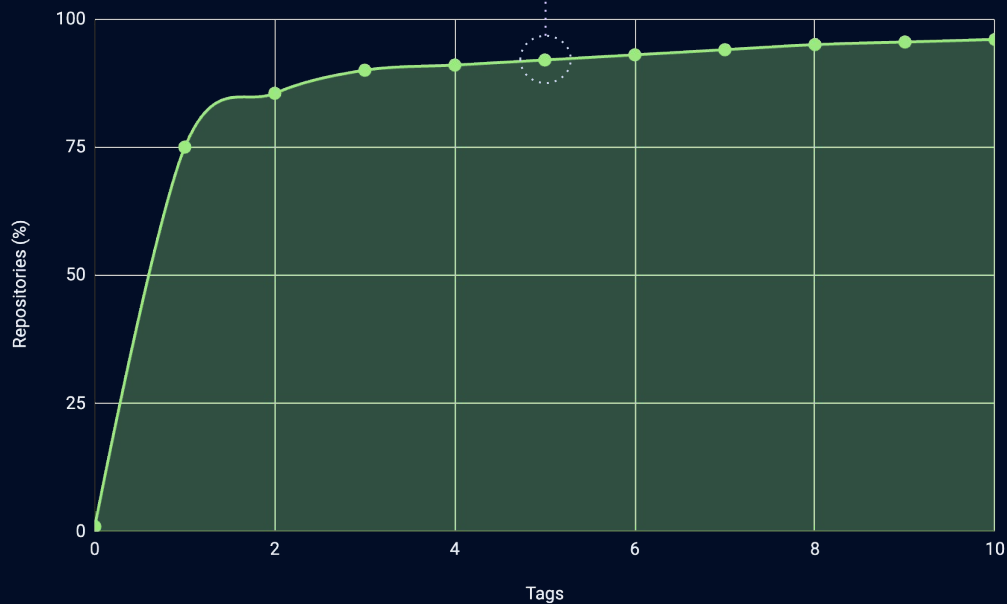
sko?

bruceejacobs/**sko**
rancher/mirrored-**sko**peo-skopeo
skorochkin/gradle-node-phantomjs2
opsani/**sko**pos
ivaldidk/**sko**leit_apcupsd
skola/kolawebserver
skottaramcs/shields-rail
skopciewski/dnsmasq

Refining The Scope of the Scan



Listing Tags



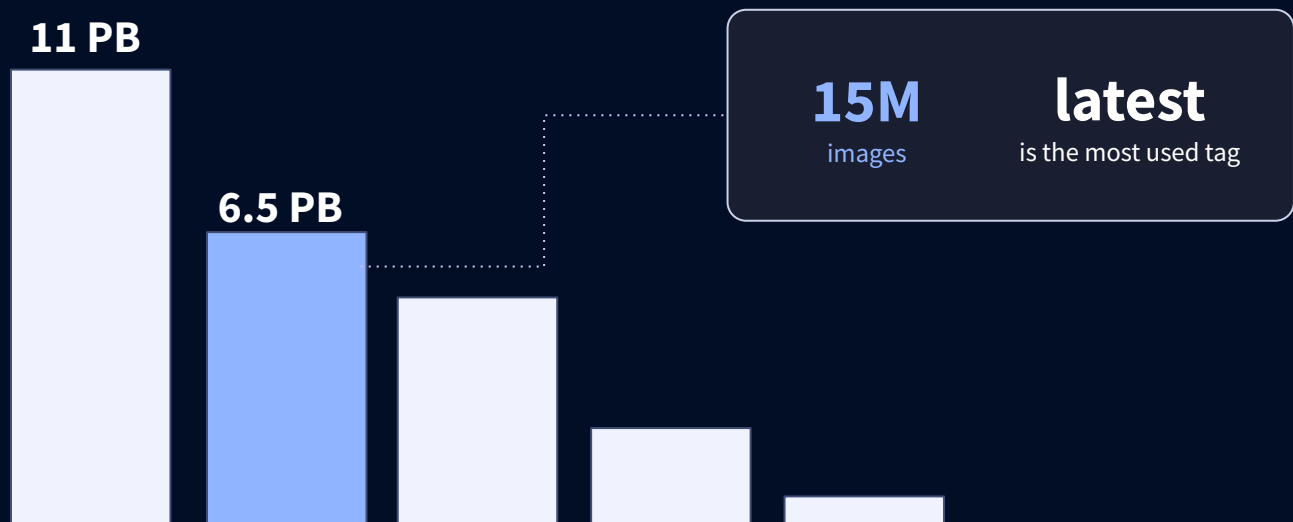
5
tags

93%
of repositories

 ~3 days

Refining The Scope of the Scan

At Most 5 Tags Per Repository



[Explore](#) / [liuzyg/trajdeleter](#) / latest


liuzyg/trajdeleter:latest

MANIFEST DIGEST sha256:3b3170498d184cf1f07711f8c0ca4fe10baebbf2617c196ee0561e338f66c5d 

OS/ARCH
linux/amd64

COMPRESSED SIZE ⓘ
170.08 GB

LAST PUSHED
8 months by [liuzyg](#)

TYPE
Image

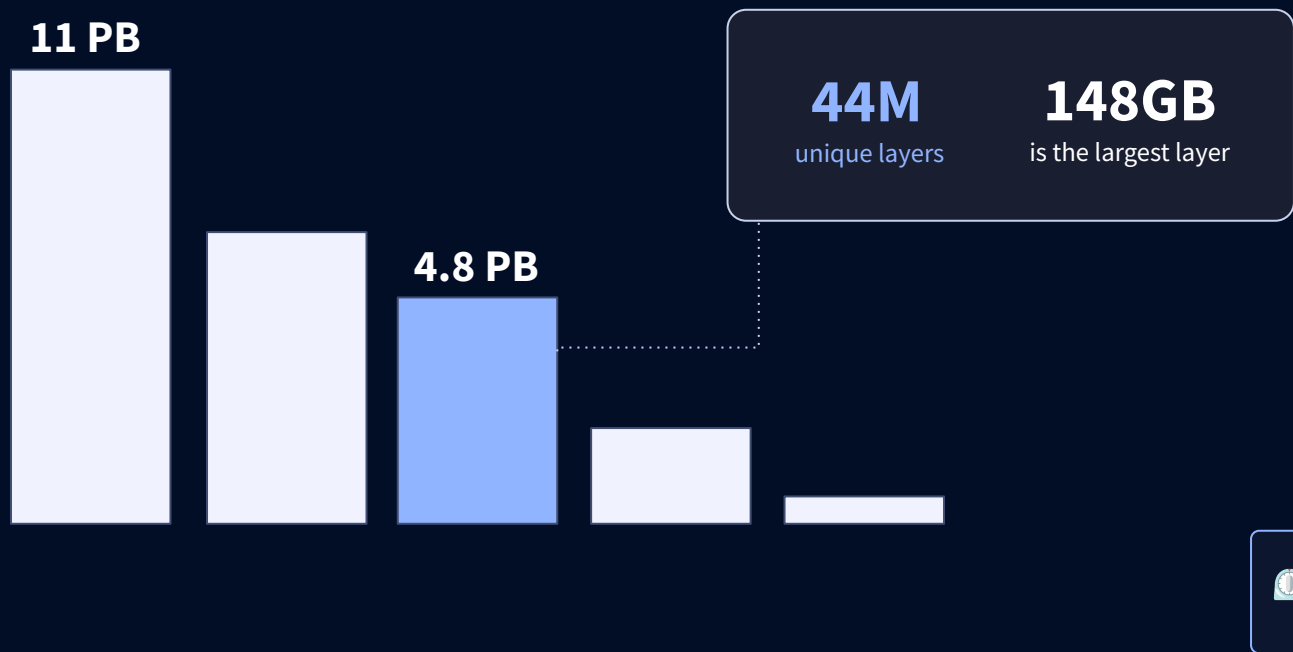
MANIFEST DIGEST
sha256:3b317049... 

Image Layers ⓘ

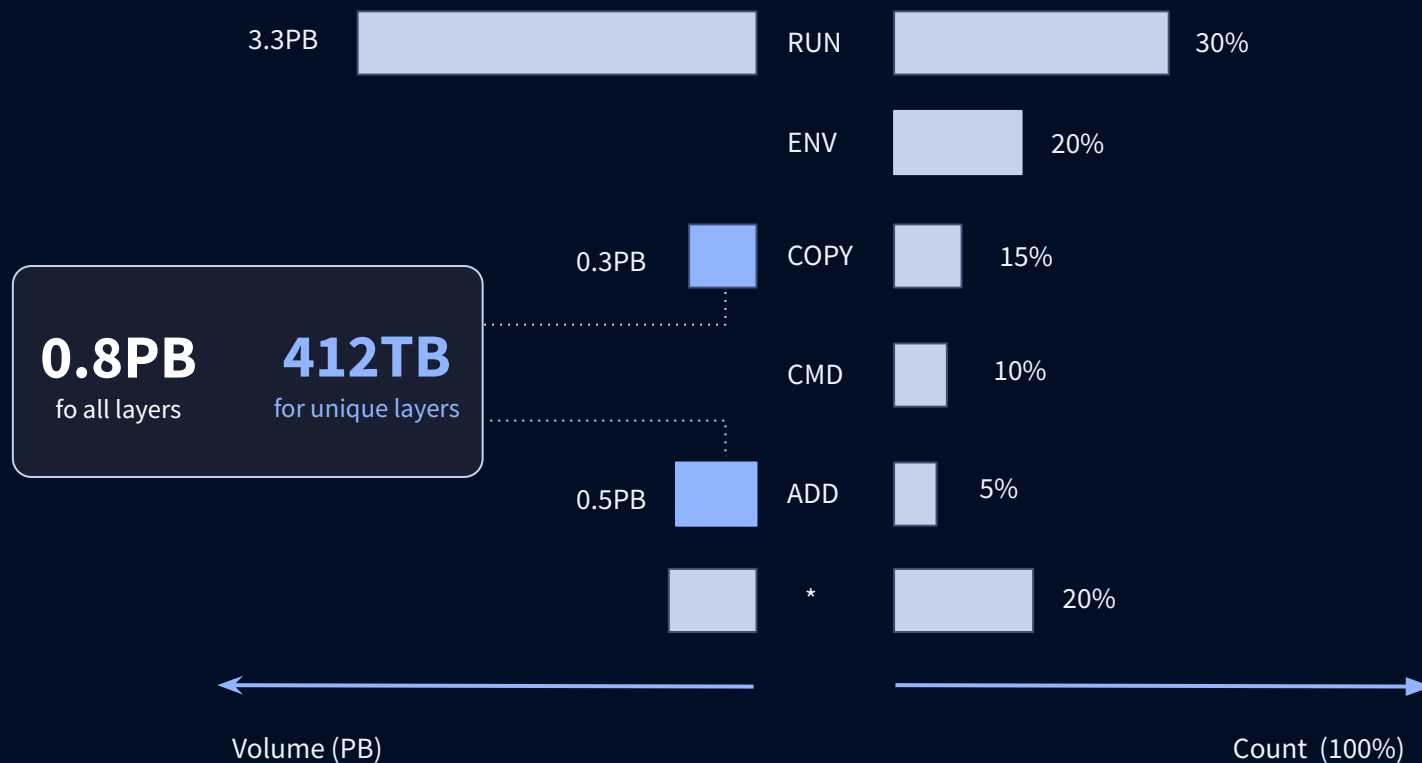
1	128.25 GB
2 /bin/bash	41.72 GB
3 /bin/bash	25.26 KB
4 /bin/bash	47.98 KB

Refining The Scope of the Scan

Removing Duplicated Layers Per Repository

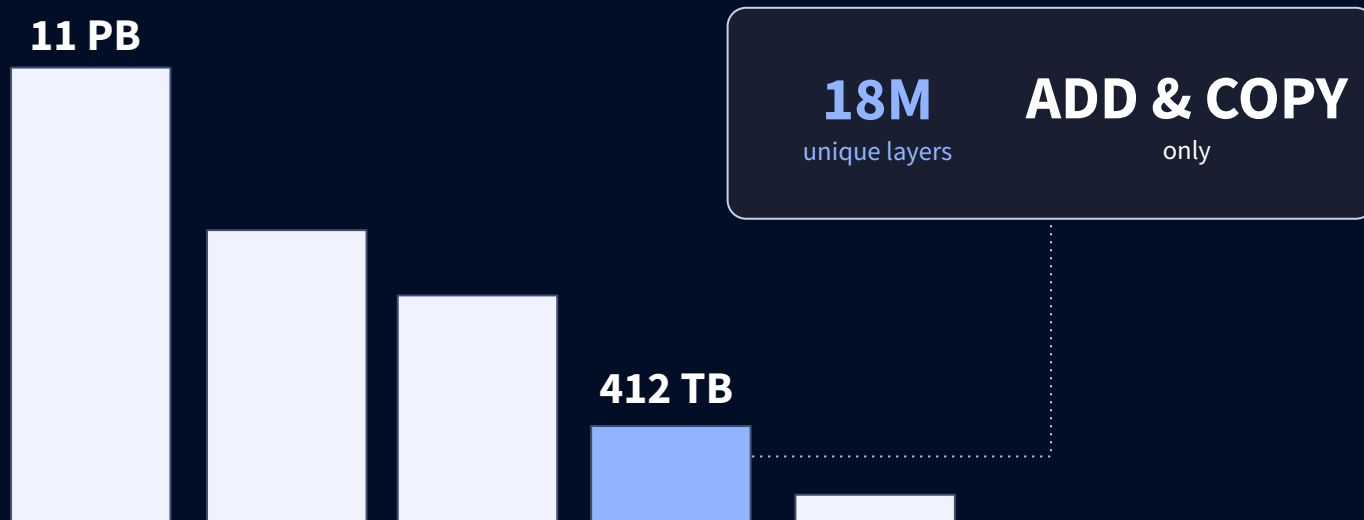


Layers Sizes & Instructions

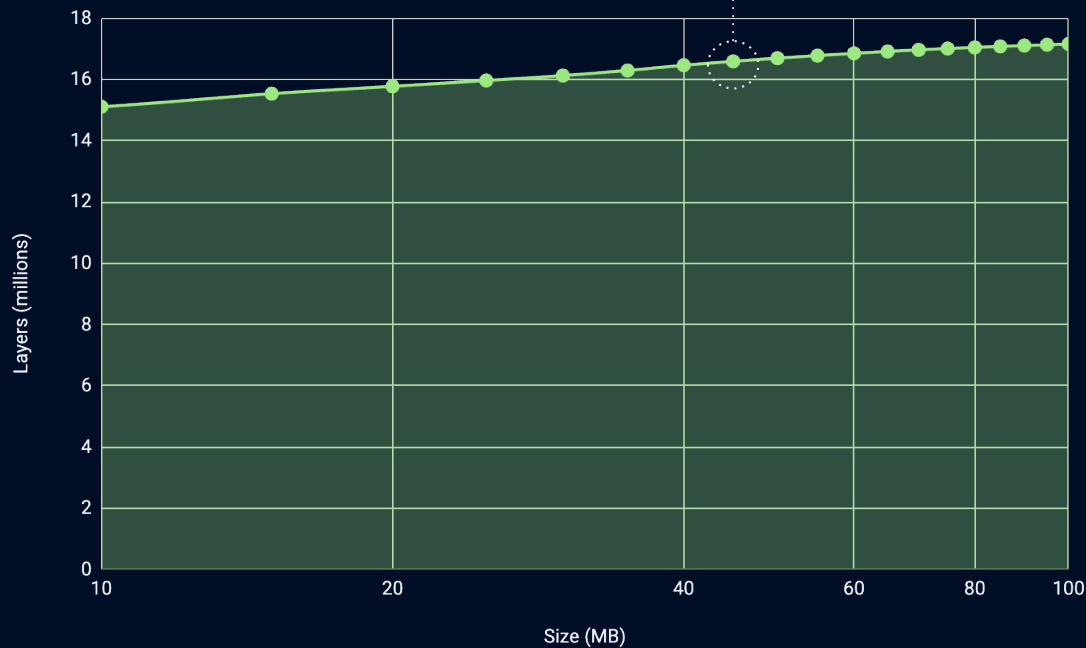


Refining The Scope of the Scan

Filtering Dockerfile Instructions



ADD & COPY Layers Sizes



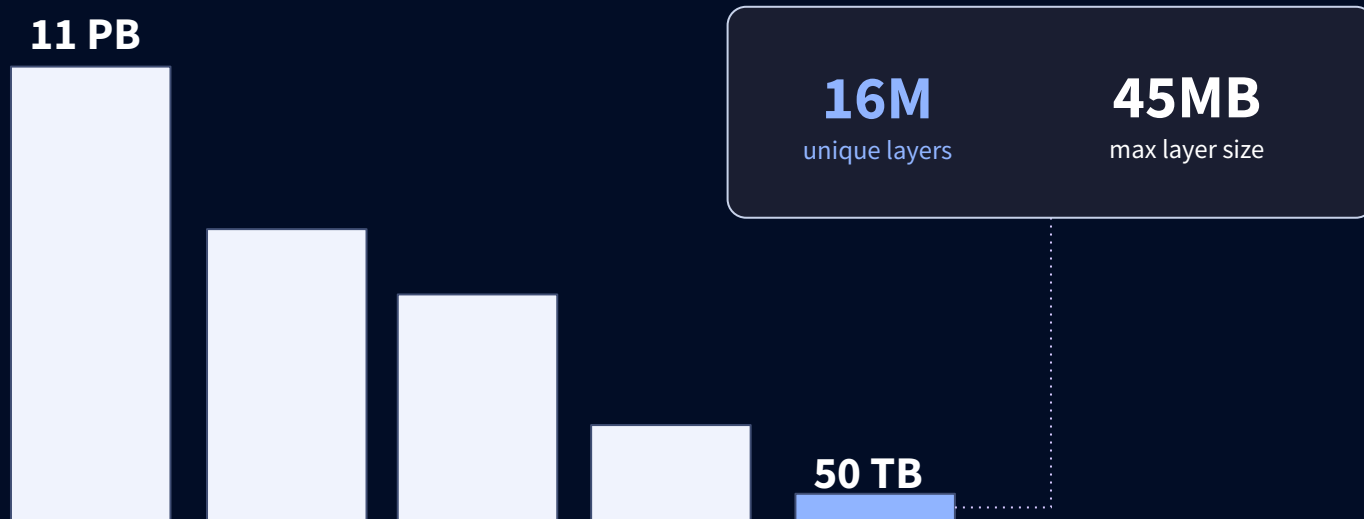
99%

below 200MB

90%

below 45MB

Refining The Scope of the Scan



Scanning Docker Hub



04

Secrets Analysis

Charts & Numbers.

Two Types of Secrets Detectors

Specific

well-known patterns & formats
mapped to services and providers
may be **automatically validated**

500k unique secrets

Generic

random looking
unknown service or provider
lack context

700k unique secrets



13%

categorized

Generic-ML Certified

Classify Generic Secrets

to find **their corresponding types** using the file content.

Version Control

```
RUN /bin/sh -c git clone https://user:  
■■■@gitlab.com/user/repo.git
```

Data Storage

```
ENV SA_PASSWORD=■■■
```

```
ENV SNOWFLAKE_USER=user SNOWFLAKE_PASSWORD=■■■  
SNOWFLAKE_ACCOUNT=account SNOWFLAKE_WAREHOUSE=COMPUTE_WH  
SNOWFLAKE_DATABASE=NAME_1 SNOWFLAKE_SCHEMA=PUBLIC
```

Generic-ML False Positives

Generic Secrets

are a **minefield of false positives**.

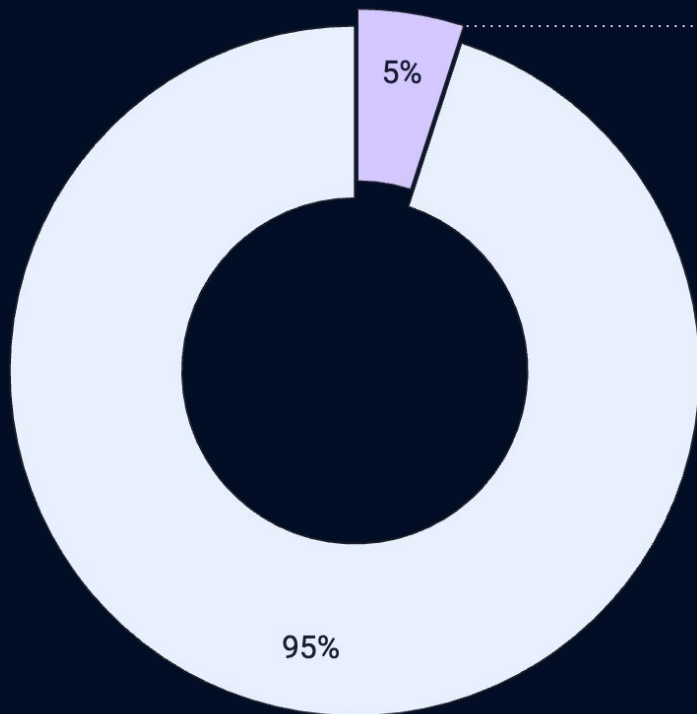
Path to secrets registry

```
variables:  
  RESOURCE_TYPE: "secrets"  
  VAULT_ADDR: https://vault.prod.company_name.com  
  SECRET_PATH: "name/XYZ12345/AB98765/prod/secrets"
```

Variables Names

```
credentialsBinding {  
  file('AZURE_SECRET_FILE',  
    Credentials.azuresecret_credentials_id)  
  usernamePassword('AWS_ACCESS_KEY_ID',  
    'AWS_SECRET_ACCESS_KEY', Credentials.jenkins_aws_access_id)  
  usernamePassword('sf_user', 'sf_password',  
    Credentials.python_test_user_qa1account2_credentials_id)  
}  
}
```

Repositories



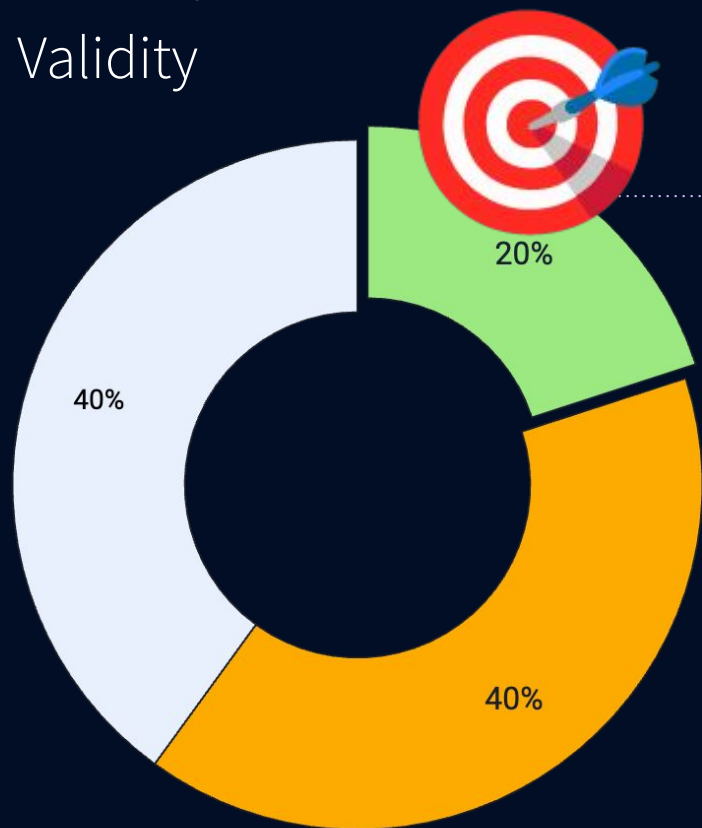
9M

repositories

450k

repositories with a secret

Validity



100k

valid secrets

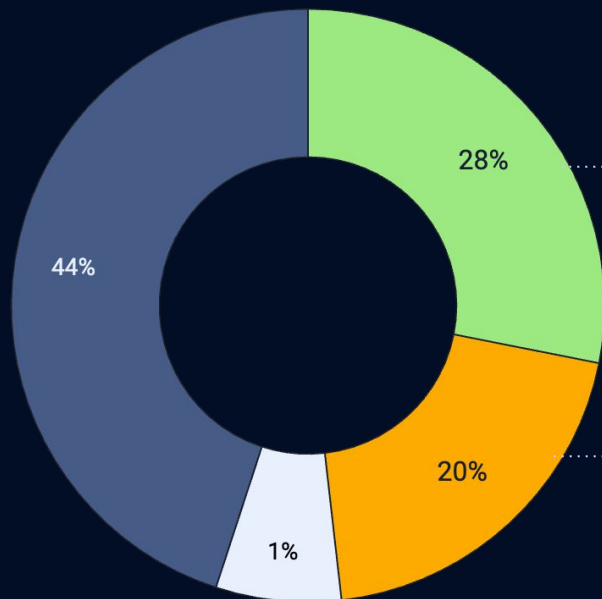
230

detector types

● Valid ● Cannot Check ● Not Valid



Types



13% are valid

out of 140k

50% are valid

out of 100k

● Data Storage ● Cloud Provider ● Version Control System ● Other

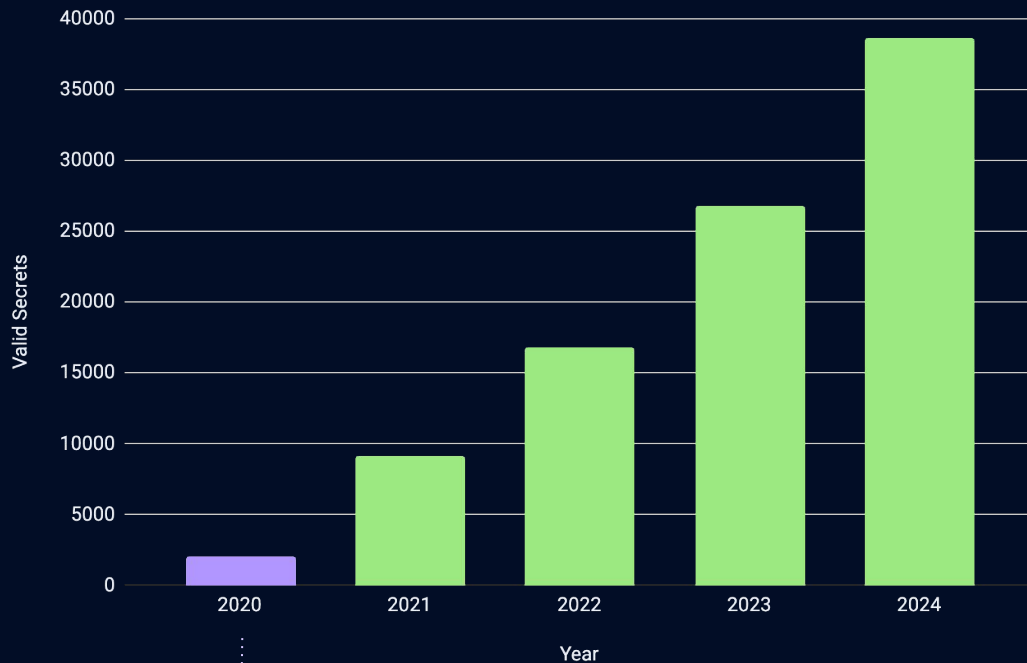
Longevity

60%

of valid secrets
before 2024

2053

valid secrets
from 2020



RUN Layers Deep Dive

Commands manually selected

such as **git**, curl, unzip, env, gcloud, **aws**...

~1500
layers handpicked

20%
contains a secret

Already in configs

```
RUN |1 password=[REDACTED] /bin/sh -c git clone  
https://call1969:${password}@bitbucket.org/call1969/frontend
```

Never seen

```
RUN |2 AWS_ACCESS_KEY_ID=[REDACTED]  
AWS_SECRET_ACCESS_KEY=[REDACTED] /bin/sh -c aws s3 cp  
s3://eventstry-config/docker-config/vendor-dashboard/.env .
```

```
RUN /bin/sh -c printenv > /env.txt
```

05

Responsible Disclosure Experiences

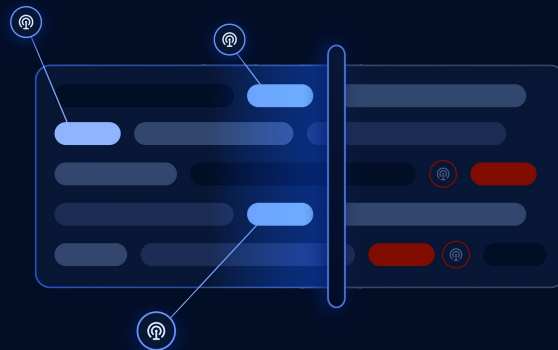
The good, the bad, and the ugly.

Statistics

6 months

28 disclosures

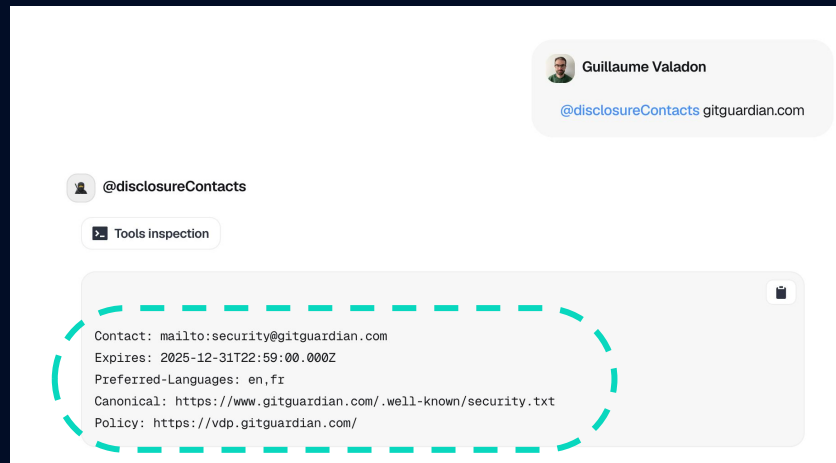
15 resolved



Agentic Assisted - Security Contacts

finding security contacts is **timing consuming**
browse websites, VDP, security.txt ...

AI agents to the rescue!
automatically do the discovery, and suggestions



Typical Negative Interactions

Some are similar to generic disclosures

don't care

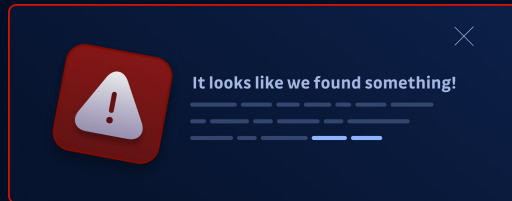
refusing to make findings public

already known

remember that we disclose **valid secrets**

no acknowledgement / no answer

escalating to National Security Agencies hardly helps



Halls of Fame

The Oracle logo, featuring the word "ORACLE" in a bold, red, sans-serif font.

most disclosures not public
despite existing bounty programs (less than \$1000 so far)

Zoo of Unexpected Interactions

vendor cannot reproduce

<https://github.com/GitGuardian/ggshield> is open-source

secret leaks not in Bug Bounty scope

reports are stopped during triaging

secret leaks not a product issue

no answer but internal ticket communicated to a GitGuardian sales representative

06

Docker Pitfalls & Takeaways

Docker Remembers

```
FROM ubuntu:latest  
  
COPY . /app  
  
RUN /app/do_things.sh  
  
RUN npm ci --production && rm .npmrc  
  
ENTRYPOINT ["docker-entrypoint.sh"]
```

.npmrc not removed

still in the COPY layer

Leaks in Dockerfiles?

```
$ skopeo inspect docker://[REDACTED]/[REDACTED]:latest --config | jq -rc '.history[].created_by'
[...]  
/bin/sh -c echo [REDACTED]:[REDACTED] > ~/.passwd-s3fs  
/bin/sh -c chmod 600 ~/.passwd-s3fs
```

in practice, there are **few hard-coded secrets in Dockerfiles**

Leaks in Configs?

```
$ skopeo inspect docker://[REDACTED]/[REDACTED]:latest --config |jq -rc '.history[].created_by'
[...]
```

/bin/sh -c #(nop)	ARG BUILD_ARG_GITHUB_ACCESS_TOKEN
/bin/sh -c #(nop)	ARG BUILD_ARG_GITHUB_USERNAME
/bin/sh -c #(nop)	ARG BUILD_ARG_CONCOURSE_URL
/bin/sh -c #(nop)	ARG BUILD_ARG_CONCOURSE_USER
/bin/sh -c #(nop)	ARG BUILD_ARG_CONCOURSE_PASSWORD
/bin/sh -c #(nop)	ENV GITHUB_ACCESS_TOKEN=ghp_[REDACTED]
/bin/sh -c #(nop)	ENV GITHUB_USERNAME=[REDACTED]
/bin/sh -c #(nop)	ENV CONCOURSE_URL=http://[REDACTED]:8080
/bin/sh -c #(nop)	ENV CONCOURSE_USER=concourse
/bin/sh -c #(nop)	ENV CONCOURSE_PASSWORD=con@123

leaks appear in configs, as a result of **docker build**

Leaking Build Arguments

docker build leaks
secrets into the config

```
FROM ubuntu:latest
```

```
ARG SECRET
```

```
ENV TEST=$SECRET
```

```
RUN /bin/bash -c "/bin/echo ${TEST}"
```

\$ docker build -t leak:latest --build-arg SECRET=BLuX25@ .

```
ARG SECRET=BLuX25@
```

```
ENV TEST=BLuX25@
```

```
RUN |1 SECRET=BLuX25@ /bin/sh -c /bin/bash -c "/bin/echo ${TEST}"
```

Build Arguments Leaks Are Documented

[Home](#) / [Manuals](#) / [Docker Build](#) / [Building](#) / Variables

Build variables

In Docker Build, build arguments (`ARG`) and environment variables (`ENV`) both serve as a means to pass information into the build process. You can use them to parameterize the build, allowing for more flexible and configurable builds.

Warning

Build arguments and environment variables are inappropriate for passing secrets to your build, because they're exposed in the final image. Instead, use secret mounts or SSH mounts, which expose secrets to your builds securely.

See [Build secrets](#) for more information.

Similarities and differences

Build arguments and environment variables are similar. They're both declared in the Dockerfile and can be set using flags for the `docker build` command. Both can be used to parameterize the build. But they each serve a distinct purpose.

[Edit this page](#)

[Request changes](#)

Table of contents

- Similarities and differences
 - Build arguments
 - Environment variables
 - ARG usage example
 - ENV usage example
 - Scoping
 - Pre-defined build arguments
 - Multi-platform build arguments
 - Proxy arguments
 - Build tool configuration variables
 - BUILDKIT_COLORS
 - BUILDKIT_HOST
 - BUILDKIT_PROGRESS
 - BUILDKIT_TTY_LOG_LINES

Best Practice: Secrets Mounts

secrets are only
accessible at build time

```
FROM ubuntu:latest
```

```
RUN --mount=type=secret,id=SECRET,env=SECRET /bin/bash -c "/bin/echo ${SECRET}"
```



```
$ docker build -t leak:latest --secret id=SECRET,env=RANDOM .
```



```
RUN /bin/sh -c /bin/bash -c "/bin/echo ${SECRET}" # buildkit  
ENV SECRET=
```

Could Secrets Leak in RUN Layers?

```
FROM ubuntu:latest

COPY . /src

RUN pip install -r requirements.txt

RUN env > .env

EXPOSE 8080

ENV NAME=KubeCon

CMD ["python", "app.py"]
```

similar files

```
config.js
settings.py
credentials.json
appsettings.json
```

Let's Grep This!

```
/bin/sh -c printenv > .env
```

```
RUN /bin/bash -o pipefail -c env > .env # buildkit
```

```
RUN /bin/sh -c cat /run/secrets/dot_env > .env # buildkit
```

```
RUN /bin/sh -c cat envs/standalone.k3s.env > .env # buildkit
```

```
RUN /bin/sh -c cat main-env > .env # buildkit
```

```
RUN /bin/sh -c cat main.env > .env # buildkit
```

```
RUN /bin/sh -c env > .env # buildkit
```

**ideal secrets management
meets hard-coded secrets**

Takeaways

Include Secrets Leaks as a Key Security Threat

you are **probably** leaking secrets
without realizing it

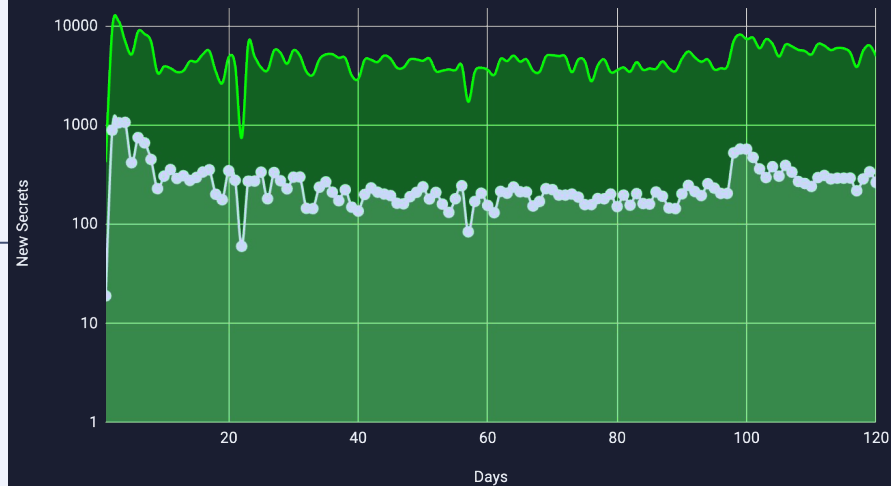
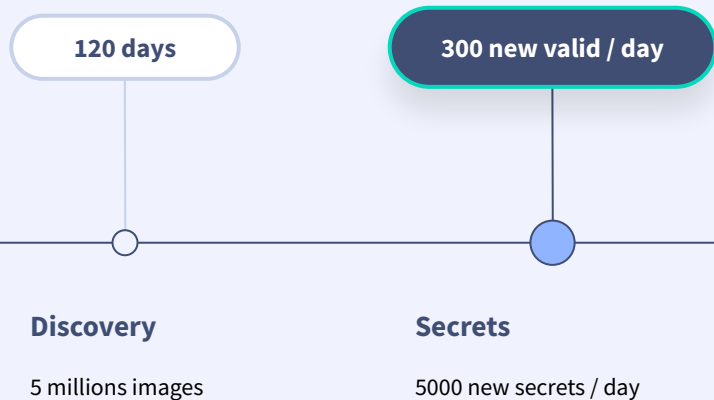
exposures come from a **wide** range of Docker pitfalls

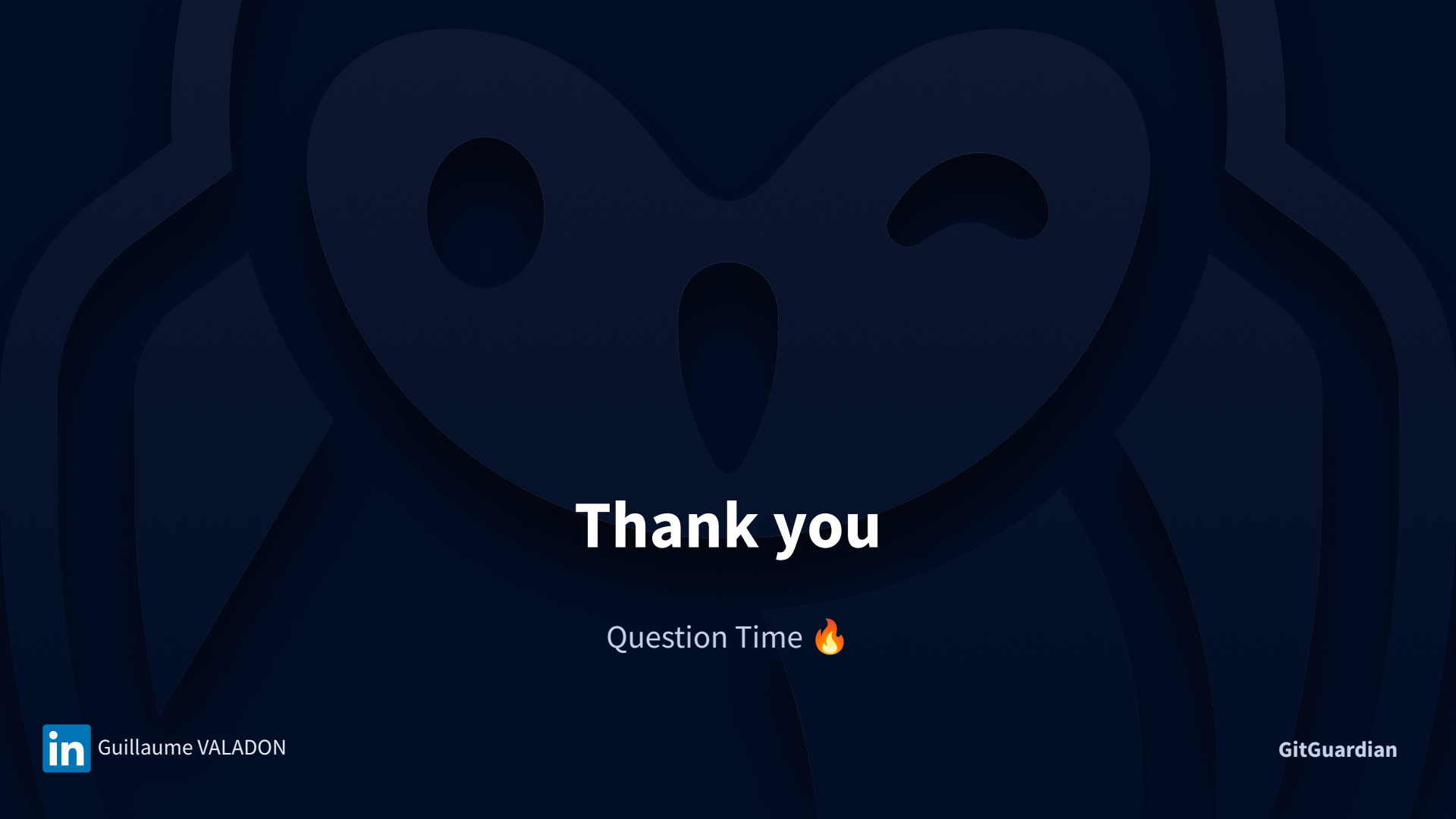
audit your images for **hard-coded** secrets



Prevention is more cost-effective than dealing with a breach!

Live Scanning Docker Hub



A large, dark blue, stylized owl logo serves as the background for the slide. The owl has large, circular eyes and a small, pointed beak.

Thank you

Question Time 🔥



Guillaume VALADON

GitGuardian



GitGuardian

Contacts

`contact@gitguardian.com`

`sales@gitguardian.com`
www.gitguardian.com