binarly

BINARLY.IO

CONTAINERDAYS 2026

Containers Don’t Keep Secrets: Scanning Docker Hub for Leaked Credentials and Private Keys

Fabio Pagani and Anton Ivanov


Travis CI
RubyGems
HashiCorp Terraform
Microsoft Teams
Generic API Key
Confluent
Notion
Bitbucket
Atlassian
Doppler
Telegram
ClickHouse
New Relic
1Password
Private Key
CircleCI
Shopify
NuGet
PlanetScale
Snyk
PyPI
Dropbox
Twitter
Slack
npm
JWT
Grafana
Sonar
Pulumi
Google Cloud
GitHub
AWS
Cohere
Okta
Plaid
Mailgun
Codecov
Harness
Stripe
GitLab
JFrog
Azure AD
Sentry
Discord
Facebook
Coinbase
HubSpot
OpenAI
Anthropic
HashiCorp Vault
Basic Auth
Netlify
OpenShift
Databricks
SendGrid
Datadog
Cloudflare
Twilio
Heroku
DigitalOcean
Hugging Face
Postman
URL Credentials
Perplexity
Sourcegraph

Novo Nordisk Breach Highlights Software Development Pipeline Risk

1

A leaked GitHub token underscores what most organizations get wrong: Treating secrets management as a tooling problem rather than an identity problem.

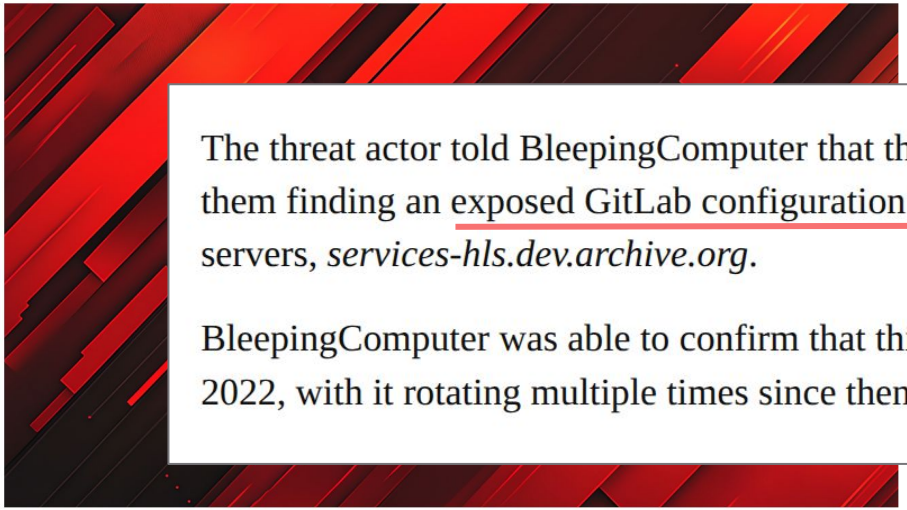
According to information provided to DataBreaches, FulcrumSec first gained access to Novo Nordisk's network in March, after finding an "exposed high-priv GitHub personal access token in client-side JS on an obscure subdomain. We cloned these repos and searched for additional credentials to move laterally. We found them, and kept finding them in the new data, and kept spidering through their systems in this fashion."

2

SOURCES:
[1] DARKREADING.COM/CYBER-RISK/NOVO-NORDISK-BREACH-EXPOSES-DEV-PIPELINE-RISK
[2] DATABREACHES.NET/2026/06/15/SCOOP-FULCRUMSEC-LEAKS-NOVO-NORDISK-DATA-AFTER-25M-DEMAND-GOES-UNPAID

Internet Archive breached again through stolen access tokens

By **Lawrence Abrams** October 20, 2024 10:46 AM 8



The threat actor told BleepingComputer that the initial breach of Internet Archive started with them finding an exposed GitLab configuration file on one of the organization's development servers, *services-hls.dev.archive.org*.

BleepingComputer was able to confirm that this token has been exposed since at least December 2022, with it rotating multiple times since then.

SOURCE: BLEEPINGCOMPUTER.COM/NEWS/SECURITY/INTERNET-ARCHIVE-BREACHED-AGAIN-THROUGH-STOLEN-ACCESS-TOKENS

Motivation

The Evolving Threat

- **Supply-Chain Attacks:** A single leaked credential can provide access to source code, build pipelines, package distribution, or production infrastructure.
- **AI and Secrets:** AI services introduce additional API credentials that can leak and new ways to expose existing secrets, while also making large-scale secret triage faster.

The Research Gap

- **Counts vs. Impact:** Past research heavily focused on headline numbers (e.g., "we found XYZ thousand secrets!") instead of proving actual security impact.
- **Generic Credentials:** Many secrets have no recognizable format, making rule-based detection noisy. How can we reliably identify them?
- **Focus on Credentials:** Prior studies focused almost exclusively on common credentials. But what about exposed private keys?

01 · Scanning Docker Hub namespaces for secrets

Selecting high-impact namespaces

CREDENTIAL-FOCUSED SCOPE

Scope Definition: Prioritized popular namespace, Fortune 500 companies, and organizations with public Bug Bounty/VDP programs based on potential security impact.

Asset Mapping: Searched Docker Hub, GitHub, and GitLab using organization names, program handles, and website domains to identify associated accounts.

Developer Exposures: Included personal Docker Hub namespaces of public members and contributors, as corporate secrets can also end up in developers' personal images.

PRIVATE-KEY SCOPE [1]

Private Key Hypothesis: Used a separate approach to identify images likely associated with industrial and IoT environments.

Protocol-Based Search: Used IIoT protocols and technologies, such as MQTT, as Docker Hub search terms to identify relevant namespaces.

90K+ DOCKER HUB NAMESPACES SELECTED FOR SCANNING

[1] "SECRETS REVEALED IN CONTAINER IMAGES: AN INTERNET-WIDE STUDY ON OCCURRENCE AND IMPACT", DAHLMANNS ET AL., ASIA CCS 2023

Building a layer-level scanning pipeline

STAGE 01

Repository, tag & layer enumeration

Queries Docker Hub API for repos, tags, and layer digests without downloading content first.

Scans all tags and platform variants instead of relying solely on `latest` tag for linux/amd64.

STAGE 02

Layer-Level Deduplication

Analyzes individual layers to recover secrets deleted in later layers and absent from the final merged filesystem.

Deduplicates layers by digest, downloading and processing each unique layer only once.

STAGE 03

Normalization & Scanning

Normalizes content into Binarly Analysis Archive (BA2) format, recursively extracting compressed contents.

Includes Docker build instructions (RUN, ENV)

Uses format-specific detection rules, including regexes, known prefixes, and encodings to identify common API keys, access tokens, private keys, and other recognizable secret formats.

WITHOUT DEDUPLICATION

3.6 PB

REFERENCED LAYER VOLUME IF PROCESSED INDEPENDENTLY

DEDUPLICATION

−90%

IN DATA TO PROCESS THROUGH LAYER DEDUPLICATION

WITH DEDUPLICATION

350 TB

UNIQUE LAYER DATA SCANNED AFTER DIGEST DEDUPLICATION

Secret candidates

We identified 98,816 unique secret candidates.

SECRET TYPE	UNIQUE CANDIDATES	SHARE
JWT tokens	68,101	68.9%
Weak shadow password hashes	7,770	7.9%
Square access token candidates	6,439	6.5%
Credentials embedded in URLs	4,291	4.3%
GCP API key candidates	1,938	2.0%

Why raw scanner counts are misleading

A syntactic match does not establish validity, ownership, permissions, or impact.

01

Runtime data inflates counts

More than 50,000 JWTs came from just seven files containing runtime application data.

02

Regex false positives

Broad detection rules can also match unrelated high-entropy data (the EAAA prefix used for Square is one example)

03

Test and development artifacts

Valid-looking credentials often appear in fixtures, examples, or default configurations irrelevant to a production system.

04

Configuration-dependent risk

A GCP API key can be harmless or security-sensitive depending on enabled APIs and restrictions.

Credential validation reduces triage noise

For supported credential types, validators:

- Make minimal, idempotent API requests to confirm current validity
- Never modify remote state
- Typically cannot confirm ownership, scope, or full permissions

Across the implemented validators, we confirmed 199 unique valid credentials.

81 of 199 validated credentials were AI-related: OpenAI, Hugging Face, or Anthropic credentials.

VALIDATED CREDENTIAL	COUNT	SHARE
OpenAI API keys	54	27.1%
SendGrid API tokens	43	21.6%
Slack tokens	36	18.1%
GitHub tokens	29	14.6%
Hugging Face access tokens	23	11.6%
AWS access keys	5	2.5%
Anthropic API keys	4	2.0%
Stripe live tokens	4	2.0%
Terraform Cloud API token	1	0.5%

Apache images exposed long-lived GitHub PATs

BRLY-2026-011 & BRLY-2026-012 · **Apache Doris & Impala**

Two non-expiring GitHub PATs were embedded in public development images. They exposed private Apache infrastructure and could modify source code or CI/CD definitions where the accounts had write access, including apache/doris (16K★) and apache/impala (1.3K★).

```
$ docker run --platform linux/arm64 --entrypoint "" apache/gravitino-ci:doris-0.1.5 \
    cat /opt/apache-doris-1.2.7.1-bin-arm64/be/lib/hadoop_hdfs/common/hadoop-common-3.3.4.jar \
    | bsdtar -xf - --to-stdout common-version-info.properties \
    | grep ghp_
url=https://[redacted]:ghp_Ji99n...@github.com/[redacted]/doris.git
```

Another administrative GitHub PAT in a JAR

BRLY-2026-056 · **Open-Source Project**

A GitHub PAT had been distributed in release JARs since January 2025. It provided administrative access to 45 repositories, including admin, maintain, and push permissions on a popular repository (13K★).

```
$ docker run --entrypoint "" [redacted]... \
   cat /path/to/release.jar \
   | bsdtar -xf - --to-stdout io/quarkus/runtime/generated/RunTimeDefaultsConfigSource.class \
   | strings   | grep 'ghp_'
(ghp_...
```

GitHub Advisory Database / GitHub Reviewed / CVE-2024-2700

quarkus-core leaks local environment variables from Quarkus namespace during application's build

High severity | GitHub Reviewed | Published on Apr 4, 2024 to the GitHub Advisory Database • Updated on Dec 13, 2024

Vulnerability details | Dependabot alerts 0

Package	Affected versions	Patched versions
io.quarkus:quarkus-core (Maven)	>= 3.9.0.CR1, <= 3.9.1 >= 3.3.0.CR1, <= 3.8.3 < 3.2.12.Final	3.9.2 3.8.4 3.2.12.Final

Severity

High 7.0 / 10

CVSS v3 base metrics

Attack vector	Local
Attack complexity	High
Privileges required	Low

Private-repository credentials in enterprise images

Major semiconductor company

Two non-expiring GitHub credentials were found in separate images: a PAT in .git/config and an OAuth token in an LMDB database. Both were valid credentials associated with private internal repositories.

```
$ docker run --entrypoint "" [redacted] \
    cat /home/[redacted]/.git/config
...
[remote "origin"]
        url = https://[redacted]:ghp_...@github.com/[redacted]/[redacted].git
        fetch = +refs/heads/*:refs/remotes/origin/*
...
```

```
$ docker run -ti --entrypoint "" [redacted] \
    strings -e l [/long/redacted/path]/data.mdb  | grep GITHUB_TOKEN
GITHUB_TOKEN=gho_...
```

Terraform token with administrative access

BRLY-2026-036

Exposed Terraform API Token

Terraform API token inside a publicly available Docker image. The credential provided administrative access across 27 Terraform organizations, including production-related environments.

```
$ docker run [redacted]/tfh@sha256:... \
    cat scripts/.envrc
...
export WS_NAME="[redacted]"
export TFC_ORG="[redacted]"
export TF_TOKEN="5M2A..."
...
```

POTENTIAL IMPACT

Infrastructure: Modify or destroy Terraform-managed resources.

Authentication: Alter authentication settings and manage modules and providers.

Supply-Chain: Changes could affect both infrastructure and software supply-chain systems.

Additional secrets: Terraform state and configuration data could expose further credentials and sensitive information.

Other reported credential exposures

BUILD & PACKAGE INFRASTRUCTURE

BRLY-2026-015/016
JFrog Artifactory tokens provided access to a private PyPI registry, creating risk of package theft or package poisoning.

BRLY-2026-017
Two expired Azure AD client secrets and an RSA private key used to sign configuration JWTs.

AI INFRA / MAILING SERVICES

BRLY-2026-014
Valid OpenAI API Key in a container image.

BRLY-2026-048
A Kubernetes manifest contained a valid SendGrid API key with broad administrative permissions.

INTERNAL COMMUNICATIONS

BRLY-2026-050
Slack bot and app-level tokens were embedded in a Docker Compose configuration file: slack-jira-compose.yml.

One token had visibility into 980 public and 3 private channels, with message-history, file, and posting permissions.

Short-lived CI tokens: recurring exposure windows

Point-in-time validation blind spot

A credential that is invalid when scanned may still have been exploitable when the image was first published.

BRLY-2026-035 · **GitLab CI Job Tokens**

Tokens copied into layers via `.git/config` or metadata. Because images publish before jobs complete, attackers monitoring releases can extract still-valid credentials.

GitHub Actions Workflow Tokens

Workflow tokens embedded in layers or compiled binaries. Granted '`packages:write`' permissions or access to private repositories during the run window.

RELATED WORK:
UNIT42.PALOALTONETWORKS.COM/GITHUB-REPO-ARTIFACTS-LEAK-TOKENS/

THE PIPELINE PROBLEM

Repeatable Leak Window: Each affected build can publish a fresh token, repeatedly opening brief attack windows even if old ones expired.

Retrospective Blind Spot: Retrospective scanning may find only already-expired tokens, completely missing active CI-time exploitation.

Pipeline Defenses: Typical mitigations include pipeline changes such as using `.dockerignore` or multi-stage builds to exclude build-time secrets.

02 · AI-assisted detection of generic secrets

Beyond format-specific detection

FORMAT-SPECIFIC CANDIDATES

```
ghp_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
AKIAXXXXXXXXXXXXXXXX
-----BEGIN RSA PRIVATE KEY-----
```

Known prefixes and encodings support high-precision format rules.

GENERIC CANDIDATES

```
f93dd3f4673e6f116511c6022e1f34c14c2d5fee
c9a71afa-411f-4690-b405-766edbb8074b
rpcr8dgmJ7EfHjXHb1KISLdQW9cibnW6gc6hMVo1HhBn
```

Passwords and application-specific tokens may have no distinctive format

695,376 UNIQUE GENERIC-SECRET CANDIDATES

Found a candidate: real secret or noise?

```
API_BASE_URL = "http://10.0.0.19:8000"
API_KEY = "c9a71afa-411f-4690-b405-766edbb8074b"
```

Same candidate, different context

TRUE POSITIVE

Used as an API credential

The value is inserted into an Authorization header for an API request, indicating that it is intended to function as a real credential rather than an example or placeholder.

download.py

```python
import requests

API_BASE_URL = "http://10.0.0.19:8000"
API_KEY = "c9a71afa-411f-4690-b405-766edbb8074b"


def download_report(report_id, dest_path):
    response = requests.get(
        f"{API_BASE_URL}/api/v1/reports/{report_id}",
        headers={"Authorization": f"Bearer {API_KEY}"},
        timeout=30,
    )
    response.raise_for_status()
```

FALSE POSITIVE

Example or placeholder value

The value appears only in documentation or comments, while runtime code loads the real credential from an environment variable.

download.py

```
"""Download reports from the internal API.

Configuration
-------------
``API_BASE_URL``
    Base URL of the internal API.
``API_KEY``
    Bearer token sent in the ``Authorization`` header of 

Copy ``.env.example`` to ``.env`` and fill in the values,

    API_BASE_URL=http://10.0.0.19:8000
    API_KEY=c9a71afa-411f-4690-b405-766edbb8074b

```

Giving the model the context it needs

The model receives the candidate, surrounding content, file path, and file type, then returns a binary label.

CLASSIFICATION INPUT

Prompt inputs

- Candidate value
- Surrounding file content
- File path
- File type

Expected output

```
{"verdict": "true_positive"}
{"verdict": "false_positive"}
```

PRACTICAL CONSTRAINTS

01 · Inference cost

A full pass over 695,376 candidates was estimated at about $5,000; prompt changes would require relabelling.

02 · Data boundary

Third-party inference could send live credentials outside the research environment, preventing use on non-public images.

03 · Execution boundary

A network-enabled model could act on supplied credentials.

Fine-tuning a local open-weight LLM

A frontier model labels a reduced dataset; an open-weight student is fine-tuned on those labels and run locally.

TEACHER

Teacher model

Use a frontier model to label a reduced set of generic-secret candidates.

TRAINING DATA

Teacher-labelled examples

Selected candidates are labelled by the teacher model and used as training data for the student.

STUDENT

Student model

Fine-tuned on teacher-labelled examples and later evaluated against both human and teacher labels.

LOCAL INFERENCE

Run repeated passes on controlled infrastructure without per-request API billing.

DATA BOUNDARY

Candidates remain inside the research environment, including future non-public images.

EXECUTION BOUNDARY

Inference runs without network access, so the model cannot test supplied credentials.

PRIOR WORK: WIZ.IO/BLOG/SMALL-LANGUAGE-MODEL-FOR-SECRETS-DETECTION-IN-CODE

Selecting the teacher model

Balanced benchmark: 50 hand-reviewed positives cases and 50 negatives cases.

Precision was similar across models (95.2–98.0%), but recall varied substantially (68–96%).

For triage, we prioritized recall: missing a credential could represent a security breach, while a false positive costs only a few minutes of review.

MODEL	PRECISION	RECALL	F1	ACC.
claude-opus-5	98.0%	96.0%	97.0%	97.0%
gpt-5.6-sol	97.7%	84.0%	90.3%	91.0%
deepseek-v4-pro	95.2%	80.0%	87.0%	88.0%
kimi-k3	97.1%	68.0%	80.0%	83.0%

SELECTED AS TEACHER

claude-opus-5

Selected for 96% recall, 12 percentage points above the next-best model on this benchmark.

Constructing the teacher-labelled dataset

STEP 01

695,376

UNIQUE GENERIC-SECRET CANDIDATES

All unique candidates produced by the generic rules.

STEP 02

224,896

GROUPS OF NEAR-DUPLICATES

MinHash over word shingles grouped contexts with Jaccard similarity ≥ 0.85; one representative per group was retained.

STEP 03

10,983

SELECTED REPRESENTATIVES

Selected using candidate shape, prefix, and file type; 10,983 were sent to the teacher.

TRUE POSITIVES

1,591

FALSE POSITIVES

9,392

LABEL DISTRIBUTION

On the selected representatives, the teacher labelled 14.5% positive and 85.5% negative.

Student model and fine-tuning configuration

MODEL

gemma-4-E4B-it

OPEN-WEIGHT MODEL WITH 4B EFFECTIVE PARAMETERS.

- Small enough to fine-tune on one NVIDIA L4 and run in a controlled environment.
- Apache 2.0 licence.
- Published fine-tuning guidance.

TRAINING

Supervised examples from teacher labels

Each example uses the original prompt and teacher verdict. Loss is computed only on the answer tokens.

HELD OUT OF TRAINING

Excluded from training: 100 hand-reviewed benchmark cases, 544 teacher-labelled holdout cases, and groups sharing the same candidate value.

TRAINING CONFIGURATION · 1 × NVIDIA L4 (24 GB) · Unsloth · QDoRA rank 32 · 3 epochs · learning rate 2e-4

Base and fine-tuned Gemma on two evaluation sets

MODEL	EVALUATION SET	PRECISION	RECALL	F1	ACC.
gemma-4-E4B-it (base)	hand-reviewed (100)	80.0%	90.0%	84.9%	84.0%
gemma-4-E4B-it (base)	teacher holdout (544)	40.6%	87.3%	55.4%	79.6%
gemma-4-E4B-it (fine-tuned)	hand-reviewed (100)	97.9%	94.0%	95.9%	96.0%
gemma-4-E4B-it (fine-tuned)	teacher holdout (544)	91.3%	92.4%	91.8%	97.6%

HOLDOUT PRECISION VS TEACHER

40.6% → 91.3%

Fine-tuning sharply reduced false positives.

HOLDOUT RECALL VS TEACHER

87.3% → 92.4%

Catches more real secrets while sending far fewer candidates to analysts.

CANDIDATES FLAGGED FOR REVIEW

34.4% → 14.7%

The fine-tuned model more than halves the analyst review queue.

Examples surfaced by contextual filtering

AMBIGUOUS FORMAT

Base64-encoded legacy GitHub token
A 40-hex-character rule collides with SHA-1 values, so format-only detection is noisy.

```
docker-compose.yml
GIT_URL: github.redacted.com/redacted/redacted.git
GIT_BRANCH: master
TOKEN: YTk0YThmZTVjY2IxOWJhNjFjNGMwODczZDM5MWU5ODc5ODJmYmJkMw==
base64 -d  →  a94a8fe5ccb19ba61c4c0873d391e987982fbbd3
              ^ 40 hex chars — exactly like SHA-1 hash
```

TOKEN FORMAT NOT YET COVERED BY RULES

IBM Cloud API credentials
IBM Cloud credentials found before a dedicated rule existed.

```
.env
API_KEY=Rpcr8dgmJ7EfHjXHb1KISLdQW9cibn...
WATSON_URL=https://us-south.ml.cloud.ibm.com
PROJECT_ID=16964749-800b-410a-b226-90a62ffa0fcc
```

UNEXPECTED DATASET

More than 5,000 Google account credentials
3 images across 2 repositories automated phishing comments on crypto-related YouTube videos. Reported to Docker Hub [1].

```
redacted.txt
mail: redacted@gmail.com
password: redacted
submail: redacted@hotmail.com
...  more than 5,000 similar entries across
multiple files in the image
```

SOURCE: [1] KASPERSKY.CO.UK/BLOG/CRYPTOWALLET-FREE-SEED-PHRASE-SCAM/28537/

03 · Correlating exposed private keys with Internet-facing services

Private keys: are exposed keys actually in use?

No service-specific validity check: Unlike API tokens, a private key cannot be checked against a vendor API to determine whether it is deployed.

Syntactic validity is insufficient: Container images often contain test certificates, example keys, and other cryptographic material that may not protect a live system.

Security impact: Possession of the private key can enable service impersonation and, depending on the protocol and client trust model, man-in-the-middle attacks.

> THE CORE QUESTION
>
> Do any private keys exposed in public Docker Hub images correspond to keys used by services reachable on the Internet?

```
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAA
Ebm9uZQAAAAAAAAABAAAAMwAAAAtzc2gtcn
NhAAAAAwEAAQAAAYEA0bC3H/TzHA58y8N7x
JgV+xT3g/vJ8A8jH5C9P0B6Y/fU9q9g8c3V
7J9bM7Kq2l/O0Xf6fJdM4r3K9Wp0fA5K3Jd
Lm3v/6K9X9J3pLm5v8v6x9Z6p8dK2jdnv7d
aGg5c3Y3c2g4ZDJmYTM0NmY3Z2g4ajlrMGw
x5AbTNvNHA1cTZyN3M4dDl1MHYxdzJ4M3k0
ZTVhNnM3ZDhmOWcwYjFjMmQzZTRmNWc2aDd
qA5OGs5bDBtMW4yb3AzcTRxNXInN3M4dDl1
-----END OPENSSH PRIVATE KEY-----
```

Internet-wide scanning with ZMap and ZGrab2

Correlating exposed keys with Internet-facing services

1. Internet Scanning: Scan the Internet using custom tooling built on ZMap and ZGrab2 to collect TLS certificates and SSH host keys.

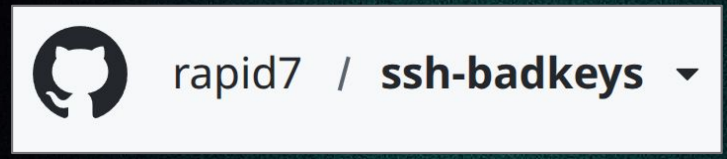
2. Derive fingerprints: parse private keys, derive public keys, and compute SPKI SHA-256 fingerprints for TLS and OpenSSH SHA-256 fingerprints for SSH host keys.

3. Match: compare observed public-key fingerprints with those derived from the container dataset.

PROTOCOL	RESPONSIVE HOSTS	HOSTS USING A DATASET KEY	UNIQUE DATASET KEYS
HTTPS	42,513,186	79,307	442
SSH	19,635,339	99,853	99
IMAPS	3,062,918	1,944	32
FTPS	378,002	7	3
MQTT-TLS	305,252	19,151	32
Total	65,894,697	200,262	583

SSH: many hosts, few distinct keys

- 99,853 SSH host observations matched fingerprints derived from 99 dataset keys.
- Many matches came from the `ssh-badkeys` reference corpus.
- The distribution was highly skewed: two known device keys (Huawei EchoLife BM626 and Ubee EVW3226) found on roughly 80,000 hosts.



A collection of static SSH keys (public and private) that have made their way into software and hardware products.

HTTPS: host count and impact diverge

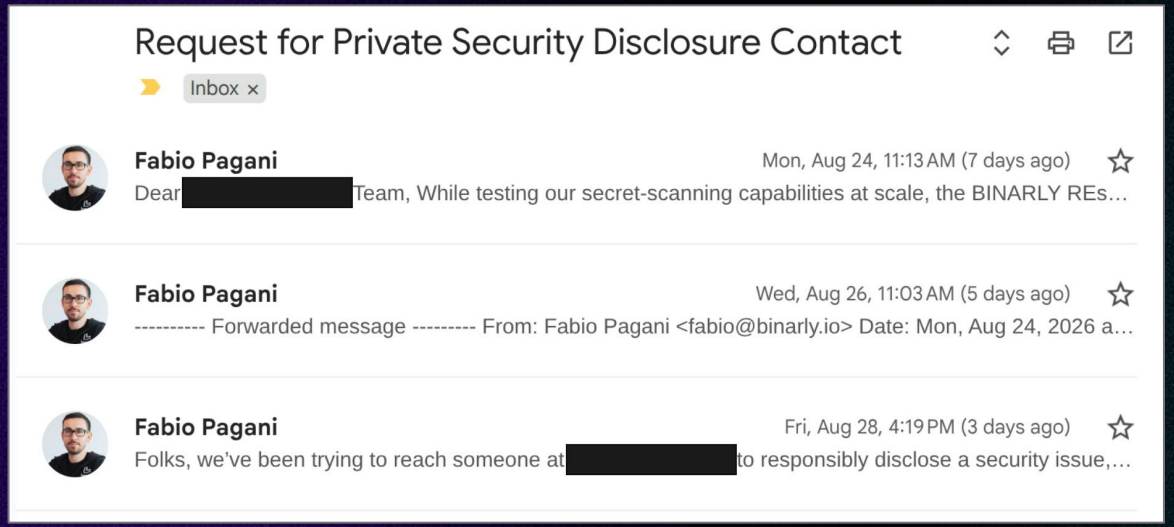
- 79,307 HTTPS endpoints presented certificates corresponding to 442 private keys recovered from our Docker Hub dataset.
- Many high-volume matches used expired certificates, lowering immediate security impact.
- Realtek RTL9607C was the notable exception: 8,145 endpoints presented certificates that were valid at scan time.
- Private keys corresponding to current, publicly trusted wildcard TLS certificates were still deployed on government and municipal in Asia: reported to `national CERTs`.

Default keys that made it to production

- Default TLS private-key material is meant to be replaced during deployment.
- One default TLS private key shipped by an MQTT provider was still observed on ~17,500 Internet-facing hosts.
- A mail-server platform reused one TLS server private key across HTTPS, IMAPS, and SMTP: the corresponding public key appeared on 1,725 endpoints: 1,575 IMAPS, 149 HTTPS, and 1 MQTT-TLS. The image also contained the private key of the CA that signed the bundled server certificate, compromising the certificate chain of trust.

04 · Conclusions

Don’t be like these folks!



Conclusions

01. **Scan the whole image, not just the final filesystem**. Secrets can persist in layers, binaries, databases, archives, build instructions, and other embedded artifacts.
02. **Validate before assessing impact.** Confirm API credential validity and correlate exposed private keys with Internet-facing services.
03. **Treat default keys as deployment risks.** Rotate TLS key material before exposing services to the Internet.
04. **Use context to triage generic secrets.** Contextual filtering helps distinguish real credentials from noise when format-based rules fall short.

Thanks to the Apache Security Team, Okteto, GitLab, Cisco Security, SAP, and the Docker Hub Security Team for working with us to investigate and quickly remediate these findings.

binarly

If any of this rings a bell, come talk to us — we can help

Binarly helps product security and engineering teams uncover and remediate vulnerabilities in third-party software, firmware, and binaries before and after release.

Book a Demo →

READ FULL DETAILS: HTTPS://WWW.BINARLY.IO/BLOG/DOCKER-HUB-SECRETS

FABIO@BINARLY.IO · BINARLY.IO