

GOTO **COPENHAGEN** 2025

September 29 - October 3 • TAP1, Copenhagen • gotocph.com



Supply Chain Security and the Real World: Lessons from Incidents

Adrian Mouat
Technical Community Advocate
Chainguard

Heartbleed Bug

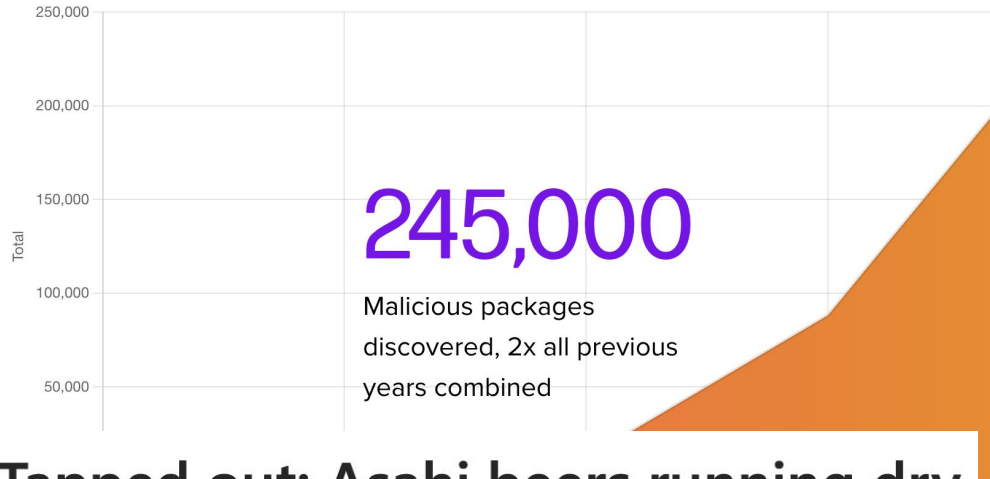
The Heartbleed Bug is a serious vulnerability in the popular OpenSSL software library. This weakness allows stealing the information protected, under normal conditions, by the SSL/TLS encryption used to secure the Internet. SSL/TLS provides communication privacy over the Internet for applications such as web, email, instant messaging (IM) and some virtual private networks (VPNs).

The Heartbleed bug allows anyone on the Internet to read the memory of vulnerable versions of the OpenSSL. This compromises the secret keys used to identify the service and to encrypt the traffic, the names and passwords of the actual content. This allows attackers to eavesdrop on communications, steal data directly from the services and users and to impersonate services and users.



SCAN			
	Vulnerability	Severity	Package
>	CVE-2018-5709	Negligible	krb5
>	CVE-2018-7738	Negligible	util-linux
	CVE-2016-10228	Negligible	glibc
	CVE-2019-7309	Negligible	glibc
	CVE-2017-7245	Negligible	pcrc3
	CVE-2017-7246	Negligible	pcrc3
	CVE-2018-0654	Negligible	libtasn1-6
		Medium	krb5
		Medium	glibc
>	CVE-2019-11461	Medium	libonig
>	CVE-2019-11462	Medium	gnupg2
	CVE-2019-11463	Medium	gnupg2

FIGURE 1.7. NEXT GENERATION SOFTWARE SUPPLY CHAIN ATTACKS (2019-2023)



Critical npm Supply Chain Attack Exposes Global Firms

By Sabica Tahira | 2 weeks ago | 📖 2 min read

News

Cybercrime soars by 31% in India, Karnataka tops with 21,889 cases

Tapped out: Asahi beers running dry in Japan as cyberattack shutdown lingers

The Jaguar Land Rover Hack Is So Bad Automaker Needs a \$2 Billion Loan















How can developers protect against supply chain attacks?

- To defend against supply-chain threats, it is important to know about the components of your software. It's a best practice to map out the source of open-source software and third-party application codes, Infrastructure as Code (IaC), and containers. Keeping track of the suppliers and code packages is the way to detect threats in software supply chains.
- We've observed that 2 in 10 open-source software contain security vulnerabilities that can prove to be blind spots. It's important to analyze the security of open-source components running through software supply chains.
- Another best practice to prevent supply chain attacks is to map out your build and release platforms. Is your infrastructure publicly accessible through DevOps pipelines?

Adrian Mouat

- DevRel Engineer at Chainguard
- Wrote "Using Docker" for O'Reilly
- adrian@chainguard.dev
- <https://bsky.app/profile/adrianmouat.com>
- <https://www.linkedin.com/in/adrianmouat/>
- <https://x.com/adrianmouat>

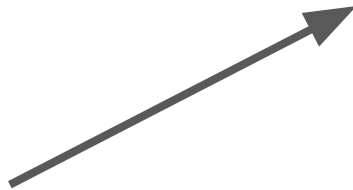
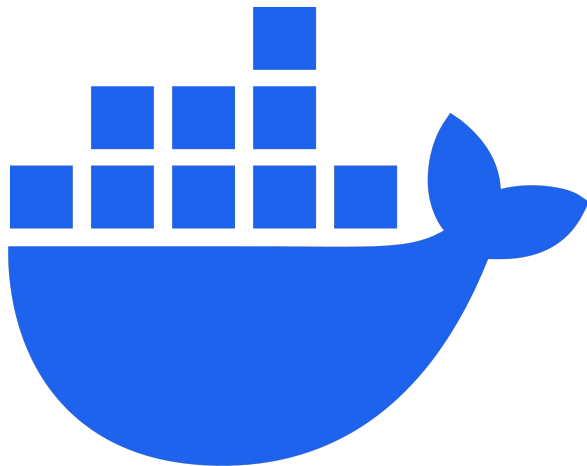


Incident 1

CodeCov

June 2021





BASH
THE BOURNE-AGAIN SHELL



BASH
THE BOURNE-AGAIN SHELL



CONFLUENT

RAPID7



HashiCorp

NEVER do this!

```
FROM debian
```

```
COPY mysecret ./
```

```
RUN curl -H 'Authorization: Bearer \  
    $(cat mysecret)' https://api.example.com
```

```
RUN rm mysecret
```

Instead

```
FROM debian
```

```
RUN --mount=type=secret,id=mysec \
```

```
curl -H 'Authorization: Bearer \
```

```
$(cat /run/secrets/mysec)' \
```

```
https://api.example.com
```

```
$ docker build --secret id=mysec,src=./mysecret .
```

name: Test and coverage

on: [push, pull_request]

jobs:

build:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v2

with:

fetch-depth: 2

- uses: actions/setup-go@v2

with:

go-version: '1.14'

- name: Run coverage

run: go test -race -coverprofile=coverage.out -covermode=atomic

- name: Upload coverage to Codecov

run: bash <(curl -s https://codecov.io/bash)

```
bash <(curl -s  
https://codecov.io/bash)
```



Not totally awful...

- Uses https
 - pretty sure it is from codecov
 - unlikely mitm
- No sudo
- But really want checksum and/or GPG check
- What if the transfer is interrupted?

What to do with these

- Split download and exec steps
- Take a look at code
- Add in checksum for automation
 - (or GPG if available)

Using Checksums

```
ENV REDIS_DOWNLOAD_URL http://download.redis-7.4.2.tar.gz
```

```
ENV REDIS_DOWNLOAD_SHA \  
4ddebb...9767dd89dbe712d2b68e808af6a1f
```

```
RUN wget -O redis.tar.gz "$REDIS_DOWNLOAD_URL"
```

```
RUN echo "$REDIS_DOWNLOAD_SHA *redis.tar.gz" \  
| sha256sum -c -
```

Using GPG

[Redis Dockerfile](#) good example

```
wget -O /usr/local/bin/gosu.asc "$url.asc"
```

```
wget -O /usr/local/bin/gosu "$url"
```

```
gpg --batch --keyserver hkps://keys.openpgp.org \  
    --recv-keys B42F6819007F00F88E364FD4036A9C25BF357DD4
```

```
gpg --batch --verify /usr/local/bin/gosu.asc \  
/usr/local/bin/gosu
```

For Distributing Software

- Put it on a Package Manager
 - e.g. homebrew tap
- Provide checksums and signatures
 - typically on releases page

THE TWELVE-FACTOR APP

III. Config

Store config in the environment

An app's *config* is everything that is likely to vary between deploys (staging, production, developer environments, etc). This includes:

- Resource handles to the database, Memcached, and other backing services
- Credentials to external services such as Amazon S3 or Twitter
- Per-deploy values such as the canonical hostname for the deploy

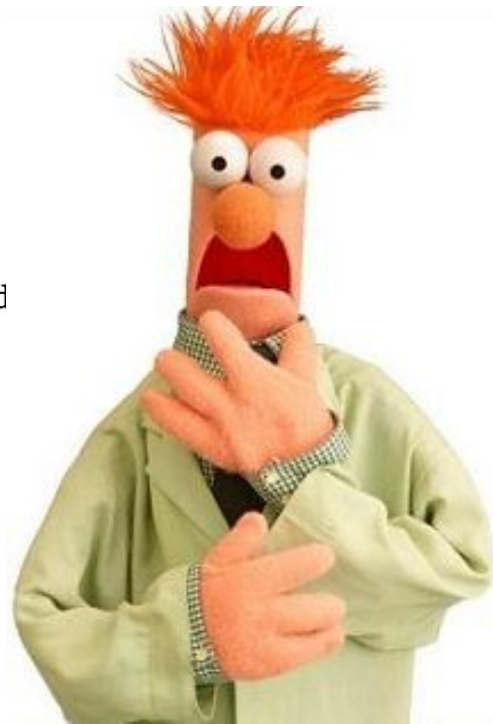
THE TWELVE-FACTOR APP

III. Config

Store config in the environment

An app's *config* is everything that is likely to vary between deploys (staging, production, d includes:

- Resource handles to the database, Memcached, and other backing services
- Credentials to external services such as Amazon S3 or Twitter
- Per-deploy values such as the canonical hostname for the deploy



Secrets and Environment Variables

- Generally not a great idea
- Too easily shared
- Accidentally printed in logs

Secrets and Environment Variables

- Prefer
 - Identity Federation
 - Using a Secret Manager
 - Temporary files
 - secret volumes in k8s

Incident 2

ChangedFiles

Mar 2025



tj-actions/changed-files

- GitHub Action modified to leak secrets in logs
- All users of latest version affected
- How were they compromised?

**Like all the best
stories, it has a
(review)dog**

Photo by [charlesdeluvio](#) on [Unsplash](#)



reviewdog

- Had automated invites for contributors!
- A PAT for an account was compromised
- Code committed that dumped secrets
- Tags modified to point to malicious commit



YES, I UNDERSTAND..

...BUT WHY?



Verify Contributors & Contributions

- Hard to vet people
- But automated workflows set you up as a target
 - probably A Bad Idea™
- Enforce commit signing
- Run checks for malware
 - malcontent

Restrict Tag Updates

- Use rulesets to stop tags being updated
- Also check your GitHub security configuration

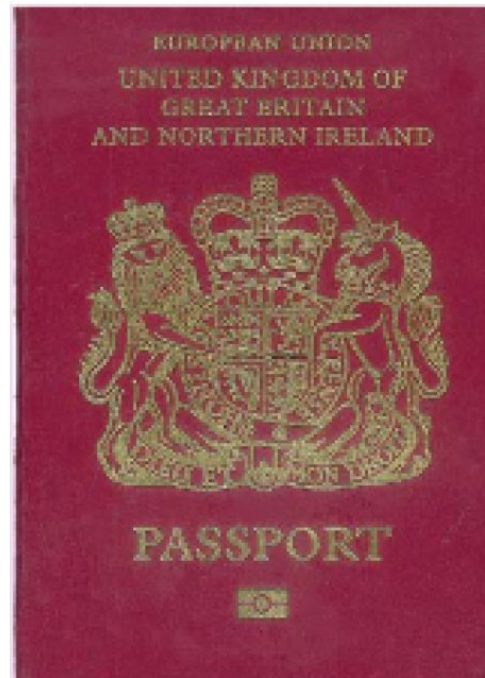
Pin Actions to Digests

- `uses: chainguard-dev/digestabot@v1.2.2`
- `uses: chainguard-dev/digestabot@432222...1c95ce2cedac # v1.2.2`
- Won't immediately use new version
- But still not perfect
 - probably use dependabot to automatically update
 - will you really spot the change?
- Also do this for container images



Avoid Long-Lived Credentials

- Prefer identity federation
- And short-lived tokens
- And/or revoke credentials on completion
- OIDC
- Chainguard created OctoSTS to replace PATs with short-lived tokens
 - <https://github.com/apps/octo-sts>



Incident 3

Shai-Hulud

Sep 2025



Shai-Hulud

- 500+ NPM Packages compromised
 - and GitHub repos
- "Worm" behaviour
 - each compromise published further hacked packages
- Forced GitHub to take action
 - stopped packages being pushed with IoC

Exfiltration

- Made public copies of private repos
- Used trufflehog to identify secrets
 - API keys and tokens
- Sent secrets to webhook SaaS
 - on free tier so got rate limited!
 - still exposed in workflow log





Q Shai-Hulud Migration

Filter by

<> Code 72

Repositories 697

Issues 0

Pull requests 0

Discussions 0

Users 0

More

Languages

Java

Shell

Python

JavaScript

Go

Jupyter Notebook

Scala

TypeScript

697 results (223 ms)

Sort by: Recently updated ▾

Save



Tyrell04/Shai-Hulud

Star

Shai-Hulud Migration

☆ 0 · Updated 19 minutes ago



avilum/Stop-Shai-Hulud

Star

Shai-Hulud Repository. Shai-Hulud Migration

☆ 0 · Updated 1 hour ago



nagliwiz/Shai-Hulud-Hulud-Shai

Star

Please stay safe, search over Github if any of your employees had secrets leaked through a repository named **Shai-Hulud** or **Shai-Hulud-Migr...**

☆ 1 · Updated 2 hours ago



amadan21/walkerdigitaltablesystems-PlaywrightAutomation-migration

Star

Shai-Hulud Migration

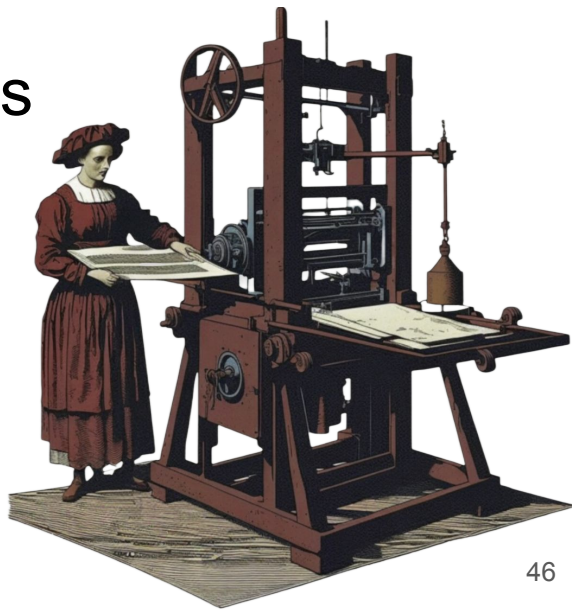
Python · ☆ 0 · Updated 7 hours ago

Friends don't let friends
publish from a laptop



Trusted Publishing

- Don't have NPM tokens!
 - Or cargo, PyPI
- OIDC based
 - short-lived and tightly scoped tokens
- Works with GitHub, GitLab etc



Trusted Publishing

- Configure via NPM/PyPI/Cargo etc
- Example from PyPI

Add a new publisher

GitHub

GitLab

Google

ActiveState

Read more about GitHub Actions' OpenID Connect support [here](#).

Owner (required)

octo-org

The GitHub organization name or GitHub username that owns the repository

Repository name (required)

sampleproject

The name of the GitHub repository that contains the publishing workflow

Workflow name (required)

release.yml

The filename of the publishing workflow. This file should exist in the `.github/workflows/` directory in the repository configured above.

Environment name (optional)

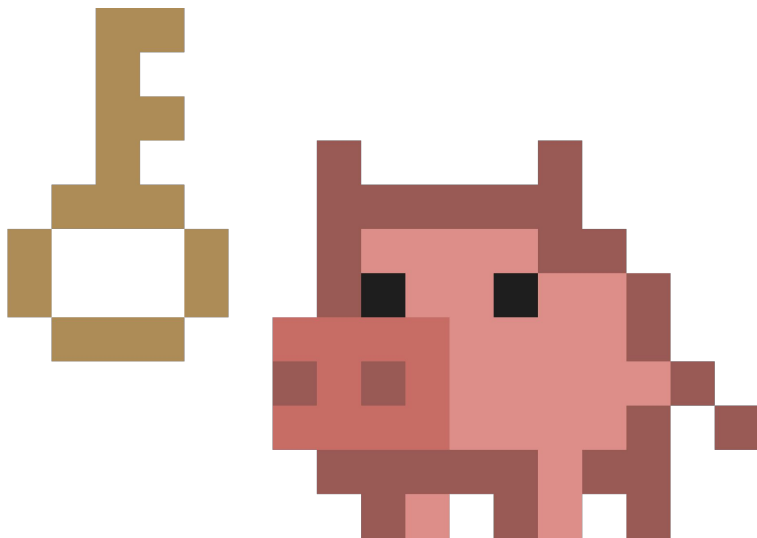
pypi

The name of the [GitHub Actions environment](#) that the above workflow uses for publishing. This should be configured under the repository's settings. While not required, a dedicated publishing environment is **strongly** encouraged, **especially** if your repository has maintainers with commit access who shouldn't have PyPI publishing access.

Add

Trufflehog

- Legitimate tool
- Scans for secrets
- Use it!
 - Or the attackers will



Multi Factor Authentication (MFA)

- Do I really need to explain
- Enforce on GitHub & NPM etc
- Buy keys for your devs



Wrap Up



Recap

- Be careful with secrets in container builds
- Verify downloads
- Avoid secrets in environment variables
- Verify contributors and contributions
- Pin actions (and images) to digests
- **Avoid long-lived credentials**
- Check GitHub Security Config
- Enforce MFA

Tooling

- [Octo STS](#) (Get rid of GitHub PATs)
- [trufflehog](#) (Don't check secrets into Git)
- [dependabot/renovate](#) (Keep everything up to date)
- [digestabot/frizbee](#) (Pin images and actions to digests)
- [yubikey](#) and [signed commits](#) (Verify commit provenance)
- [malcontent](#) (Identify malicious code or updates)
- [grype](#) (Identify known vulnerabilities)
- [sigstore](#) (Sign artifacts)

"If all **ransomware groups** suddenly became **cybersecurity specialists**, and those who are now **cybersecurity specialists** began **pentesting networks**, the **ransomware would end.**"

wazawaka