Addressing Security Concerns in Every Stage of the Software Supply Chain

Melissa McKay, Developer Advocate, JFrog

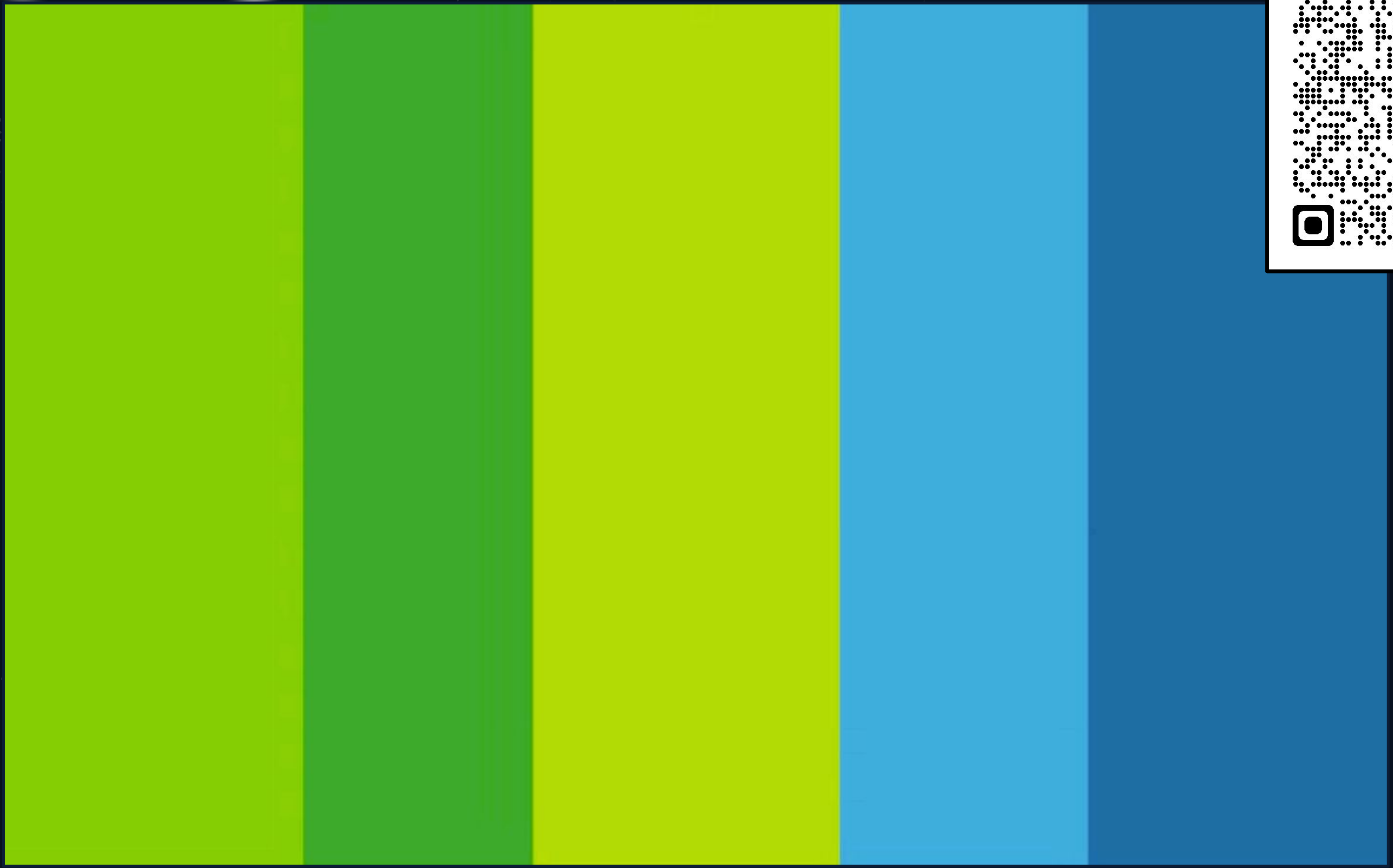
Background - Melissa McKay

- Developer!
- Speaker / Developer Advocate
- Author: Devops Tools for Java Developers
- Java Champion
- Docker Captain

@melissajmckay

linkedin.com/in/melissajmckay

The Application Journey...



JFROG & NGINX Series

Episode #1: The One Where We Planned

Episode #2: The One Where We Set Up

Episode #3: The One Where We Considered Security

Episode #4: The One Where We Deployed

Episode #5: The One Where We Updated

Episode #6: The One Where We Observed

JFROG & NGINX Discussion Series

Episode #1: The One Where We Planned

Episode #2: The One Where We Set Up

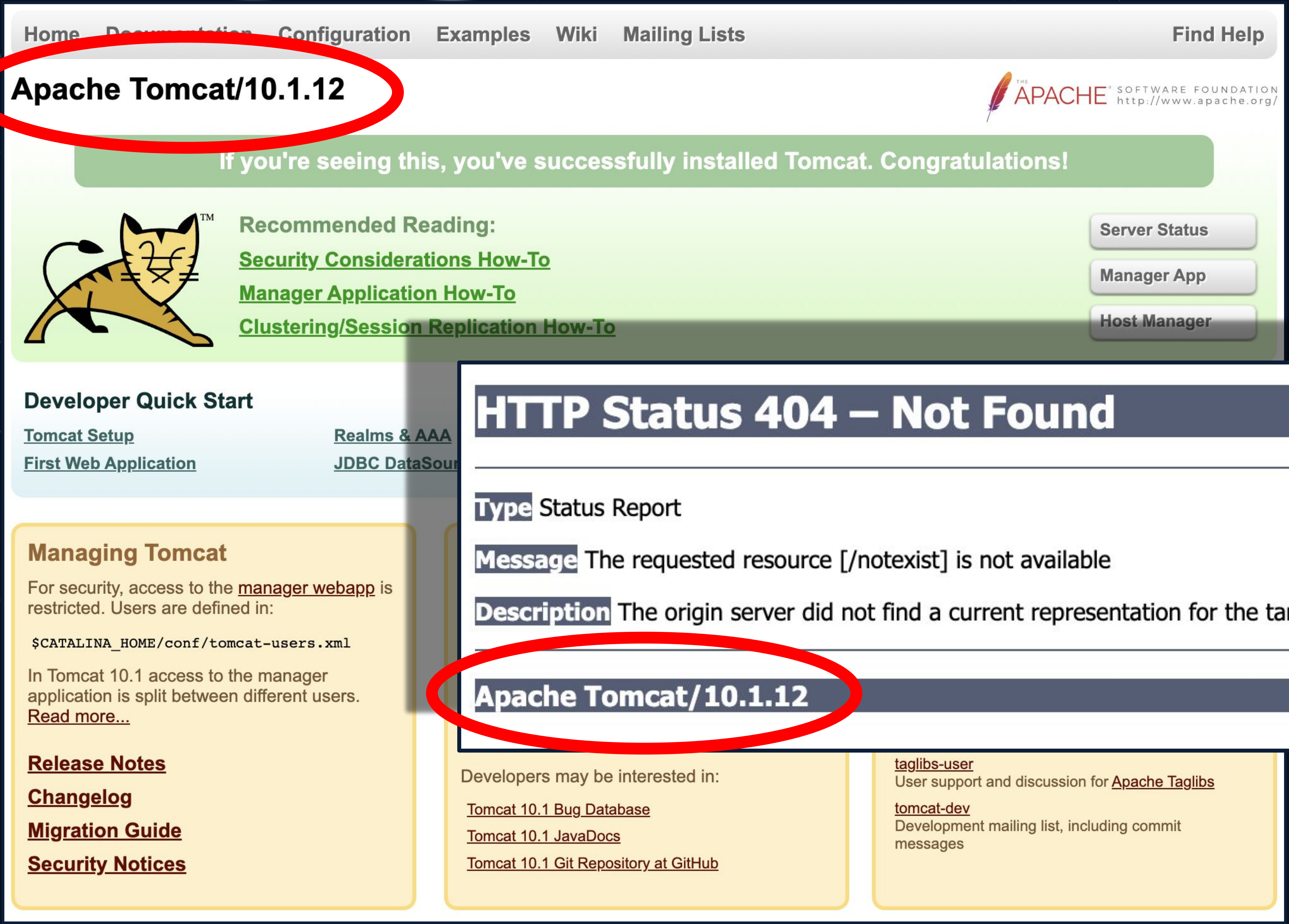
Episode #3: The One Where We Considered Security

Episode #4: The One Where We Deployed

Episode #5: The One Where We Updated

Episode #6: The One Where We Observed

Security through Obfuscation



Home Documentation Configuration Examples Wiki Mailing Lists Find Help

Apache Tomcat/10.1.12

If you're seeing this, you've successfully installed Tomcat. Congratulations!

Recommended Reading:

Security Considerations How-To

Manager Application How-To

Clustering/Session Replication How-To

Server Status | Manager App | Host Manager

Developer Quick Start

Tomcat Setup

First Web Application

Realms & AAA

JDBC DataSou

Managing Tomcat

For security, access to the manager webapp is restricted. Users are defined in:

```
$CATALINA_HOME/conf/tomcat-users.xml
```

In Tomcat 10.1 access to the manager application is split between different users. Read more...

Release Notes

Changelog

Migration Guide

Security Notices

Developers may be interested in:

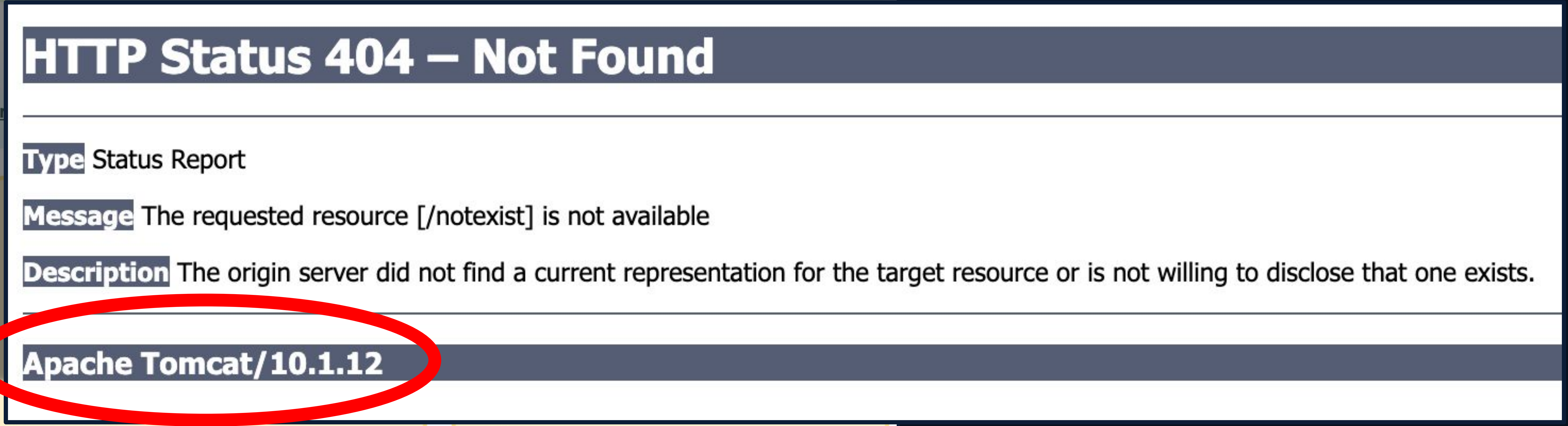
Tomcat 10.1 Bug Database

Tomcat 10.1 JavaDocs

Tomcat 10.1 Git Repository at GitHub

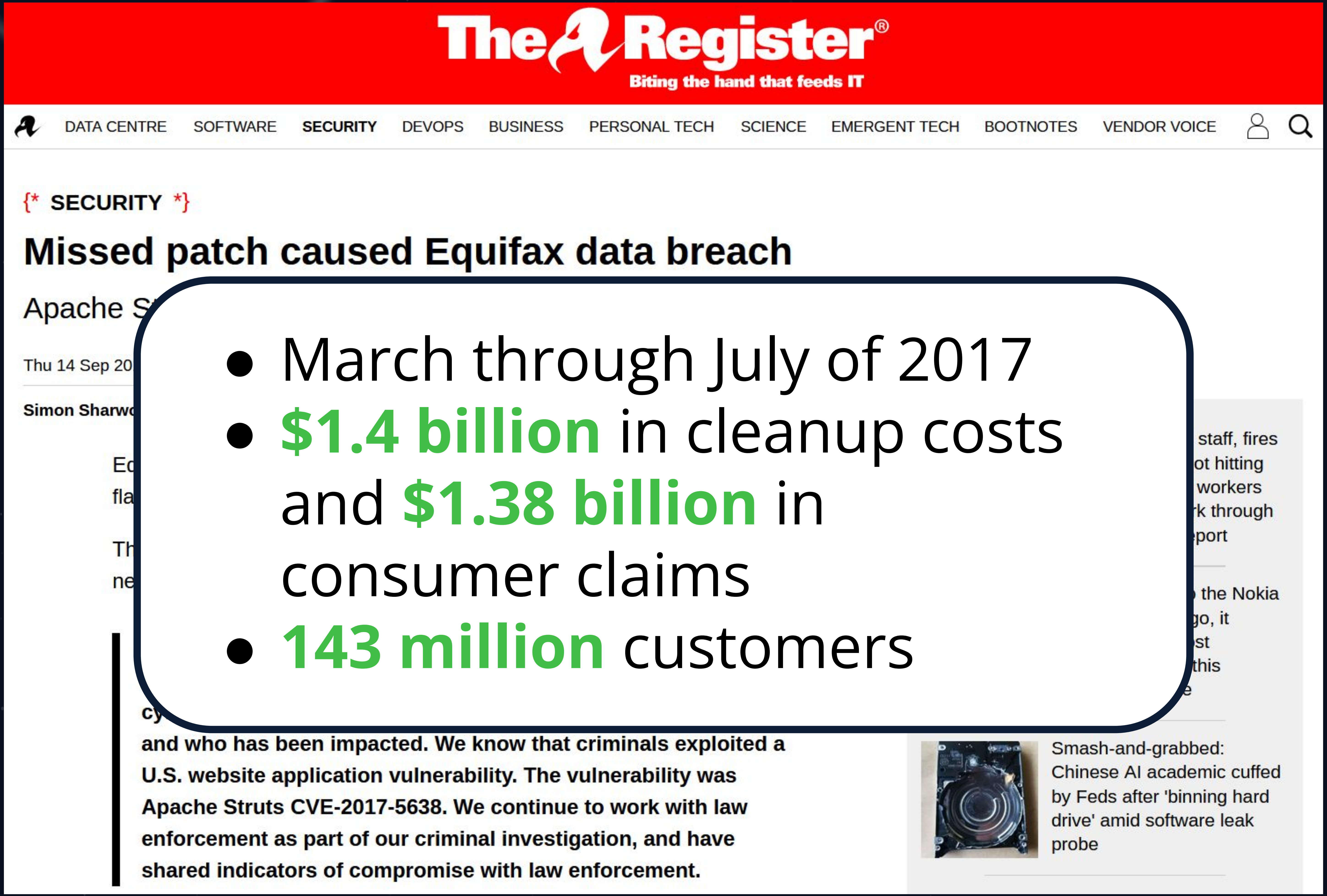
taglibs-user
User support and discussion for Apache Taglibs

tomcat-dev
Development mailing list, including commit messages



```
HTTP Status 404 – Not Found

Type Status Report

Message The requested resource [/notexist] is not available

Description The origin server did not find a current representation for the target resource or is not willing to disclose that one exists.

Apache Tomcat/10.1.12
```

- Theft of Private Customer and/or Company Data
- Loss of Money
- Loss of Credibility



The Register®
Biting the hand that feeds IT

DATA CENTRE SOFTWARE SECURITY DEVOPS BUSINESS PERSONAL TECH SCIENCE EMERGENT TECH BOOTNOTES VENDOR VOICE

{* SECURITY *}

Missed patch caused Equifax data breach

Apache S

Thu 14 Sep 20

Simon Sharw

Ed
fla

Th
ne

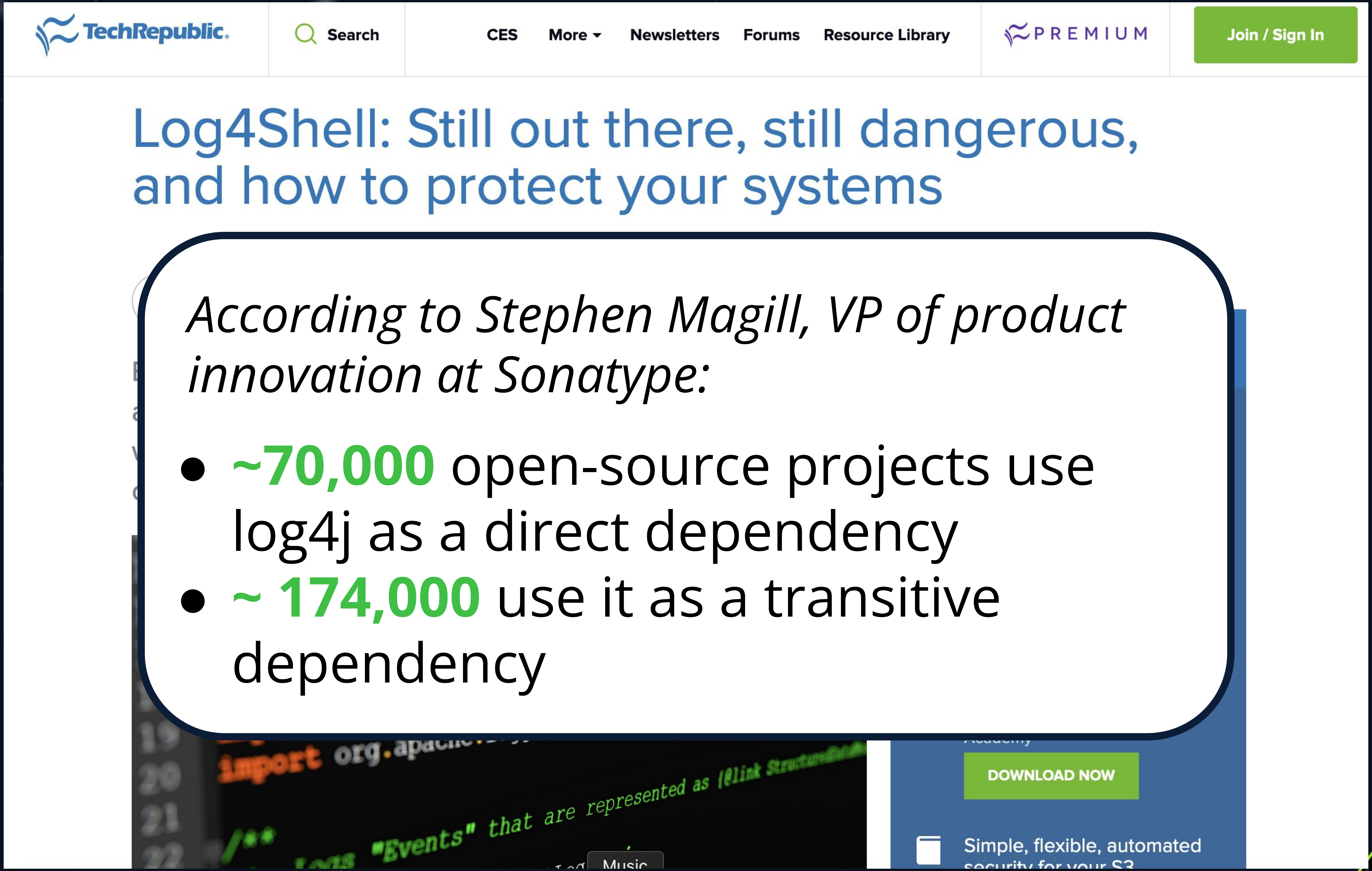
cy
and who has been impacted. We know that criminals exploited a U.S. website application vulnerability. The vulnerability was Apache Struts CVE-2017-5638. We continue to work with law enforcement as part of our criminal investigation, and have shared indicators of compromise with law enforcement.

staff, fires
ot hitting
workers
k through
port

the Nokia
go, it
st
this
e

Smash-and-grabbed: Chinese AI academic cuffed by Feds after 'binning hard drive' amid software leak probe

- March through July of 2017
- **$1.4 billion** in cleanup costs and **$1.38 billion** in consumer claims
- **143 million** customers



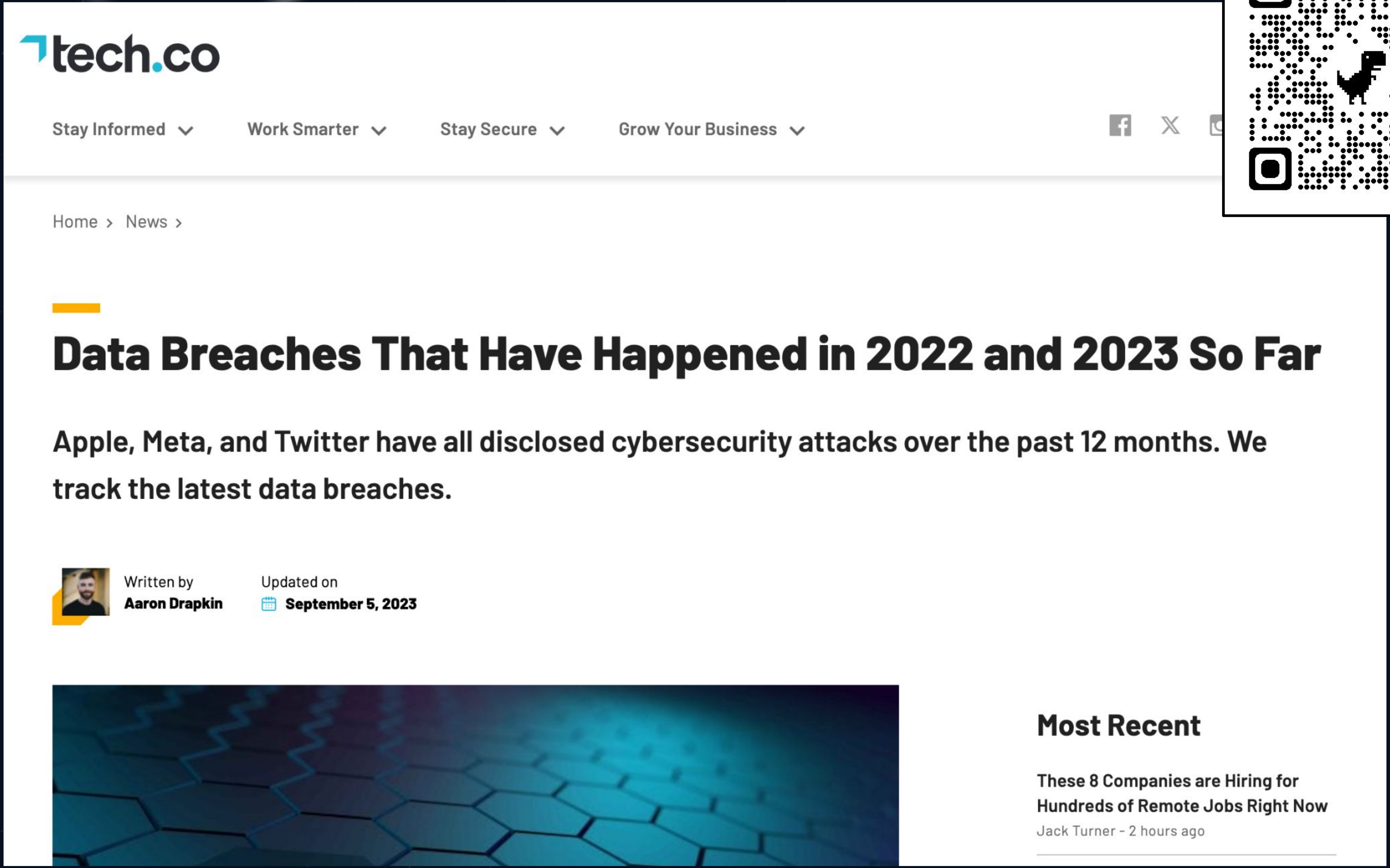
Log4Shell: Still out there, still dangerous, and how to protect your systems

```
import org.apache.
/**
 * Logs "Events" that are represented as {@link Structured
```

Simple, flexible, automated security for your S3

According to Stephen Magill, VP of product innovation at Sonatype:

- **~70,000** open-source projects use log4j as a direct dependency
- **~ 174,000** use it as a transitive dependency



tech.co

Stay Informed · Work Smarter · Stay Secure · Grow Your Business

Home › News ›

Data Breaches That Have Happened in 2022 and 2023 So Far

Apple, Meta, and Twitter have all disclosed cybersecurity attacks over the past 12 months. We track the latest data breaches.

Written by
Aaron Drapkin

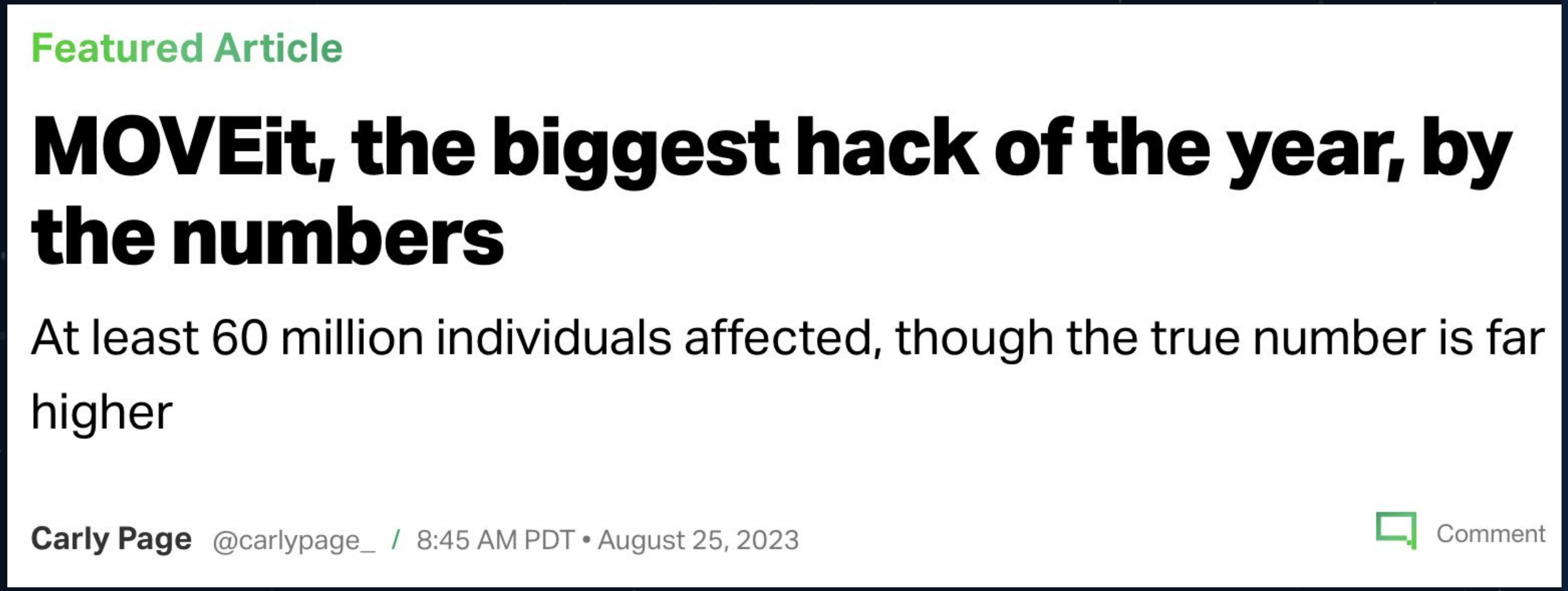
Updated on
September 5, 2023

Most Recent

These 8 Companies are Hiring for Hundreds of Remote Jobs Right Now
Jack Turner - 2 hours ago

MOVEit Transfer Vulnerability (Progress)

- June 1st - MOVEit hack, affecting Zellis, British Airways, BBC and others
- July 20 - PokerStars Data Breach (online poker - 110,000 users exposed)
- August 11 - IBM MOVEit Data Breach (4.1 million patients in Colorado)



Featured Article

MOVEit, the biggest hack of the year, by the numbers

At least 60 million individuals affected, though the true number is far higher

Carly Page @carlypage_ / 8:45 AM PDT • August 25, 2023

Comment

The global average cost of a data breach in 2023 was ***USD 4.45 million****, a 15% increase over 3 years.*

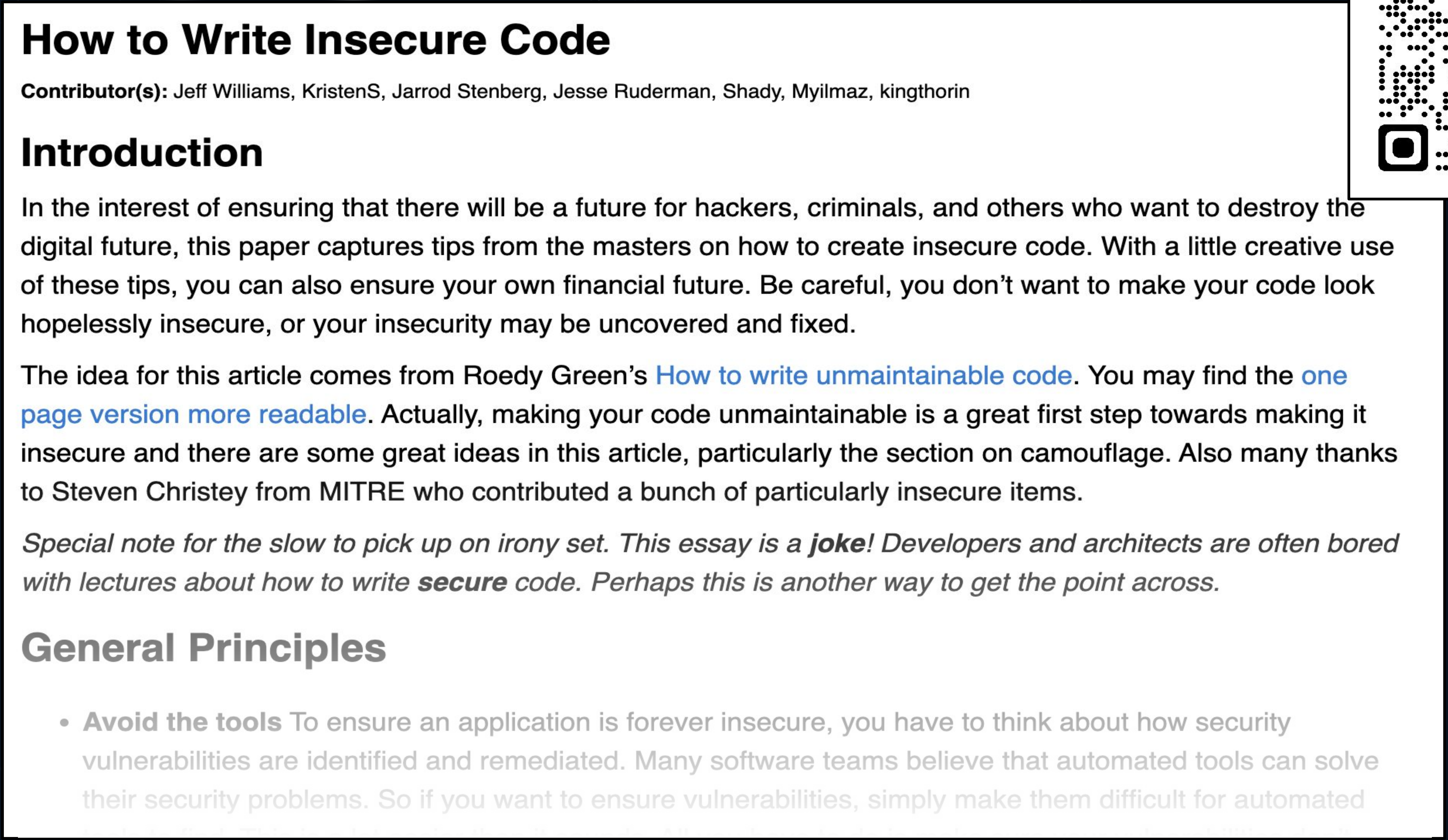
Cost of a Data Breach Report 2023, IBM

AS A DEVELOPER,

IT IS MY RESPONSIBILITY

TO WRITE CODE THAT IS SECURE.

OWASP (Open Web Application Security Project) Joke Essay



How to Write Insecure Code

Contributor(s): Jeff Williams, KristenS, Jarrod Stenberg, Jesse Ruderman, Shady, Myilmaz, kingthorin

Introduction

In the interest of ensuring that there will be a future for hackers, criminals, and others who want to destroy the digital future, this paper captures tips from the masters on how to create insecure code. With a little creative use of these tips, you can also ensure your own financial future. Be careful, you don't want to make your code look hopelessly insecure, or your insecurity may be uncovered and fixed.

The idea for this article comes from Roedy Green's How to write unmaintainable code. You may find the one page version more readable. Actually, making your code unmaintainable is a great first step towards making it insecure and there are some great ideas in this article, particularly the section on camouflage. Also many thanks to Steven Christey from MITRE who contributed a bunch of particularly insecure items.

*Special note for the slow to pick up on irony set. This essay is a **joke**! Developers and architects are often bored with lectures about how to write **secure** code. Perhaps this is another way to get the point across.*

General Principles

- **Avoid the tools** To ensure an application is forever insecure, you have to think about how security vulnerabilities are identified and remediated. Many software teams believe that automated tools can solve their security problems. So if you want to ensure vulnerabilities, simply make them difficult for automated

Essay: https://owasp.org/www-community/How_to_write_insecure_code

- **Always use default deny** Apply the principle of “Default Deny” when building your application. Deny that your code can ever be broken, deny vulnerabilities until there’s a proven exploit, deny to your customers that there was ever anything wrong, and above all - deny responsibility for flaws. Blame the dirty cache buffers.
- **Secure languages** Pick only programming languages that are completely safe and don’t require any security knowledge or special programming to secure.
- **Mix languages** Different languages have different security rules, so the more languages you include the more difficult it will be to learn them all. It’s hard enough for development teams to even understand the security ramifications of one language, much less three or four. You can use the transitions between languages to hide vulnerabilities too.

- **Rely on security checks done elsewhere** It’s redundant to do security checks twice, so if someone else says that they’ve done a check, there’s no point in doing it again. When possible, it’s probably best to just assume that others are doing security right, and not waste time doing it yourself. Web services and other service interfaces are generally pretty secure, so don’t bother checking what you send or what they return.
- **Trust insiders** Malicious input only comes from the Internet, and you can trust that all the data in your databases is perfectly validated, encoded, and sanitized for your purposes.
- **Code wants to be free!** Drop your source code into repositories that are accessible by all within the company. This also prevents having to email those hard-coded shared secrets around.

Coding Safely: Developer Education

SQL Injection

Cross Site Scripting

Cross Site Request Forgery

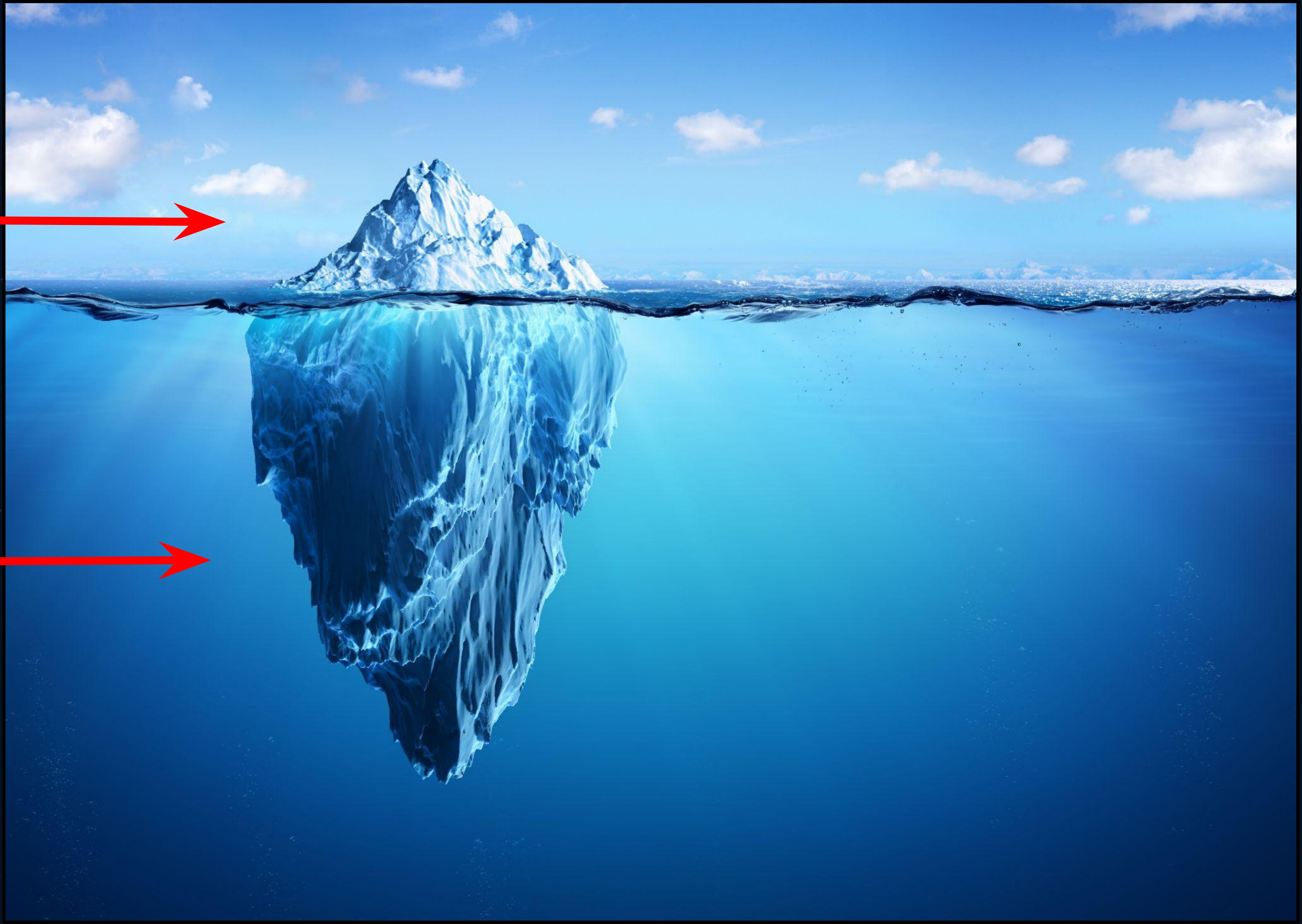
Insecure Cryptographic Storage

LDAP Injection

Software Dependencies

Code I wrote

Other stuff pulled in during the build



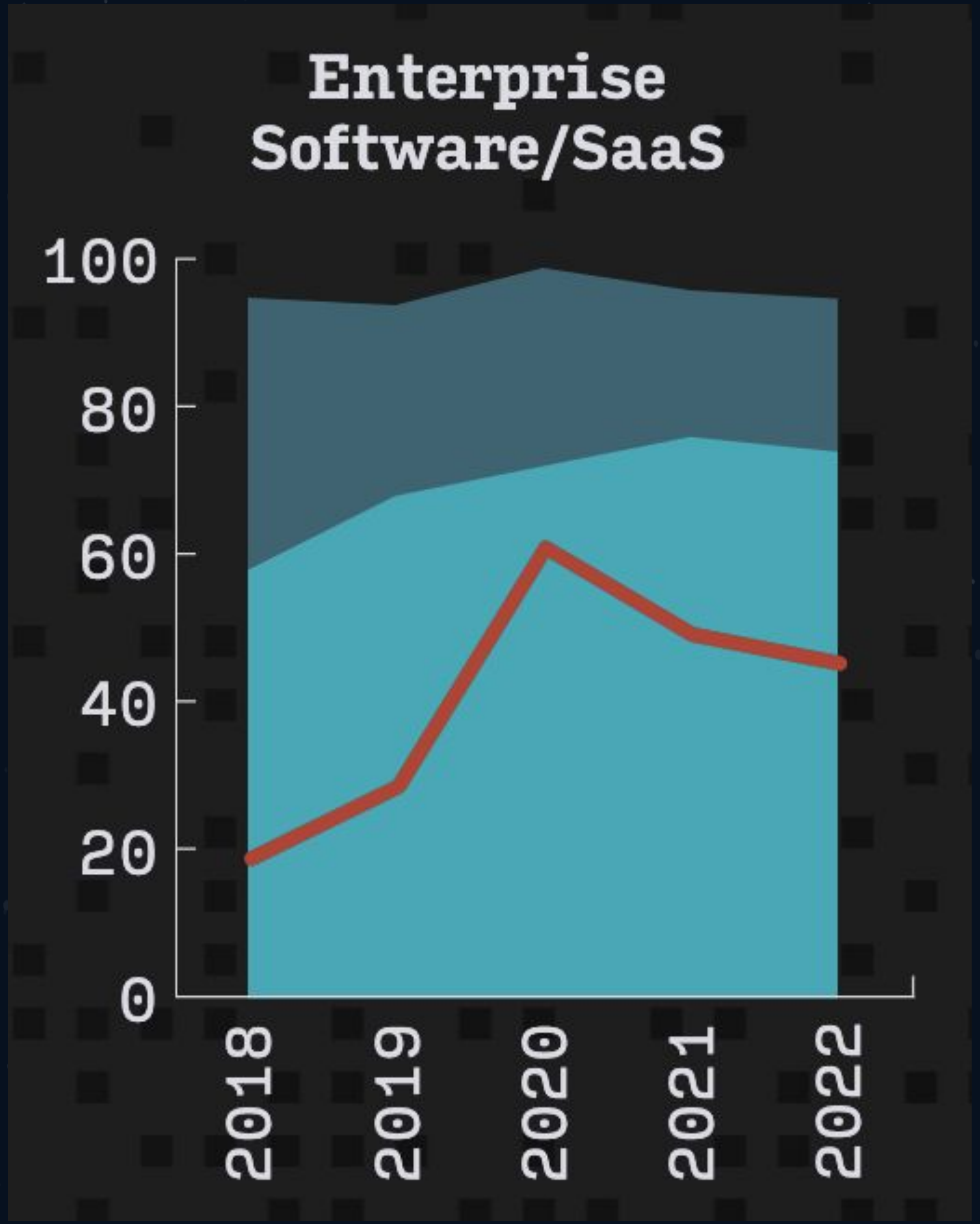
Dependencies

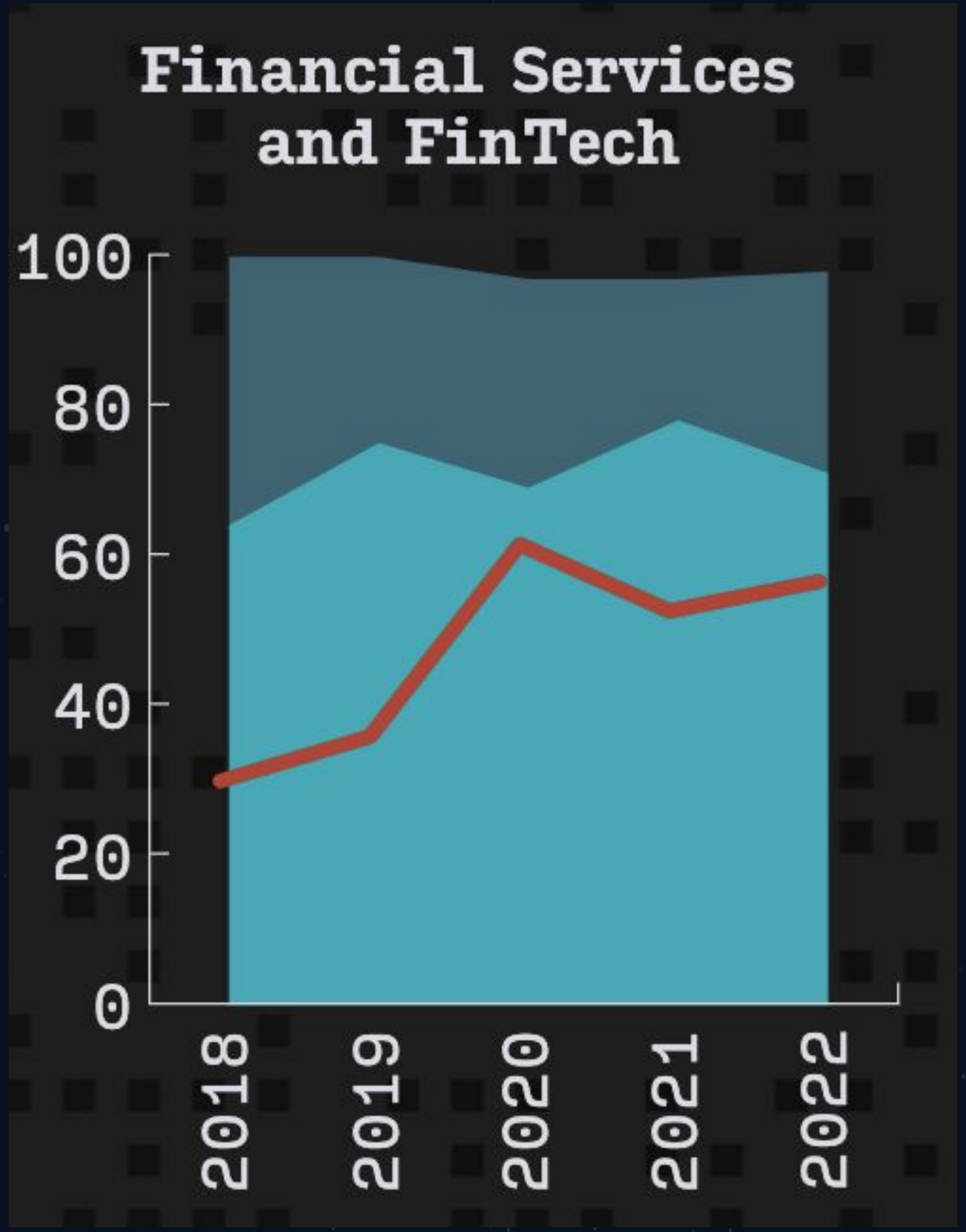
```
...
[INFO] +- de.codecentric:spring-boot-admin-starter-server:jar:2.5.5:compile
[INFO] |  +- de.codecentric:spring-boot-admin-server:jar:2.5.5:compile
[INFO] |  |  +- org.springframework.boot:spring-boot-starter-webflux:jar:2.
[INFO] |  |  |  +- org.springframework.boot:spring-boot-starter-json:jar:2.
[INFO] |  |  |  |  +- com.fasterxml.jackson.datatype:jackson-datatype-jdk8:
[INFO] |  |  |  |  +- com.fasterxml.jackson.datatype:jackson-datatype-jsr31
[INFO] |  |  |  |  \- com.fasterxml.jackson.module:jackson-module-parameter
[INFO] |  |  |  +- org.springframework.boot:spring-boot-starter-reactor-net
[INFO] |  |  |  |  \- io.projectreactor.netty:reactor-netty-http:jar:1.0.15
[INFO] |  |  |  |     +- io.netty:netty-codec-http:jar:4.1.73.Final:compile
[INFO] |  |  |  |     |  +- io.netty:netty-common:jar:4.1.73.Final:compile
[INFO] |  |  |  |     |  +- io.netty:netty-buffer:jar:4.1.73.Final:compile
[INFO] |  |  |  |     |  +- io.netty:netty-transport:jar:4.1.73.Final:compi
[INFO] |  |  |  |     |  +- io.netty:netty-codec:jar:4.1.73.Final:compile
[INFO] |  |  |  |     |  \- io.netty:netty-handler:jar:4.1.73.Final:compile
[INFO] |  |  |  |     |     \- io.netty:netty-tcnative-classes:jar:2.0.46.F
[INFO] |  |  |  |     +- io.netty:netty-codec-http2:jar:4.1.73.Final:compil
[INFO] |  |  |  |     +- io.netty:netty-resolver-dns:jar:4.1.73.Final:compi
[INFO] |  |  |  |     |  +- io.netty:netty-resolver:jar:4.1.73.Final:compil
[INFO] |  |  |  |     |  \- io.netty:netty-codec-dns:jar:4.1.73.Final:compi
[INFO] |  |  |  |     +- io.netty:netty-resolver-dns-native-macos:jar:osx-x86_64:4.1.73.Final:compile
[INFO] |  |  |  |     |  \- io.netty:netty-resolver-dns-classes-macos:jar:4.1.73.Final:compile
[INFO] |  |  |  |     +- io.netty:netty-transport-native-epoll:jar:linux-x86_64:4.1.73.Final:compile
[INFO] |  |  |  |     |  +- io.netty:netty-transport-native-unix-common:jar:4.1.73.Final:compile
[INFO] |  |  |  |     |  \- io.netty:netty-transport-classes-epoll:jar:4.1.73.Final:compile
[INFO] |  |  |  |     \- io.projectreactor.netty:reactor-netty-core:jar:1.0.15:compile
[INFO] |  |  |  |        \- io.netty:netty-handler-proxy:jar:4.1.73.Final:compile
[INFO] |  |  |  |           \- io.netty:netty-codec-socks:jar:4.1.73.Final:compile
...
```

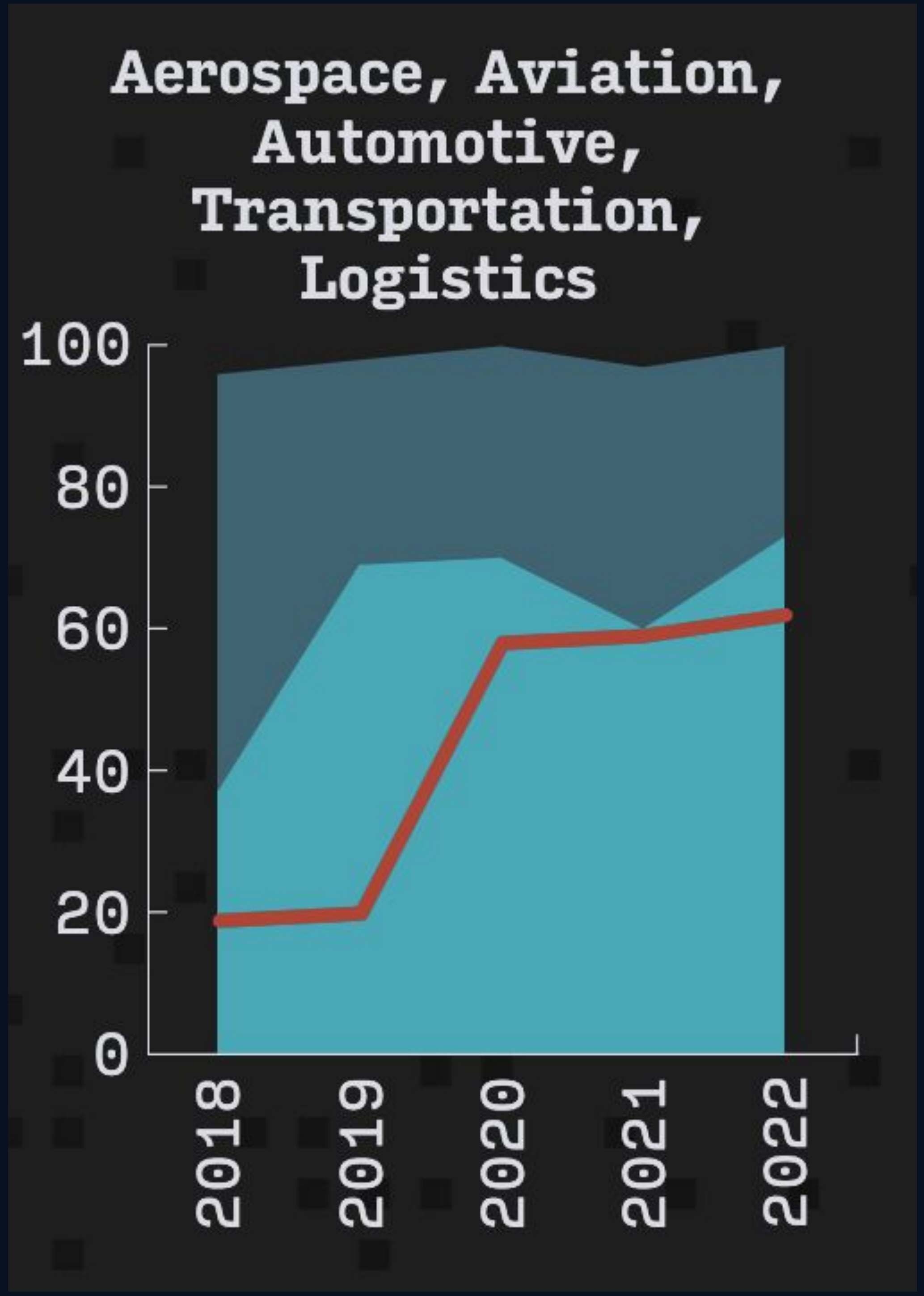
114
Direct and indirect dependencies!

7
Layers deep!

Synopsis 2023 OSSRA Report (CyRC findings from 2022)







■ Percentage of codebases containing open source

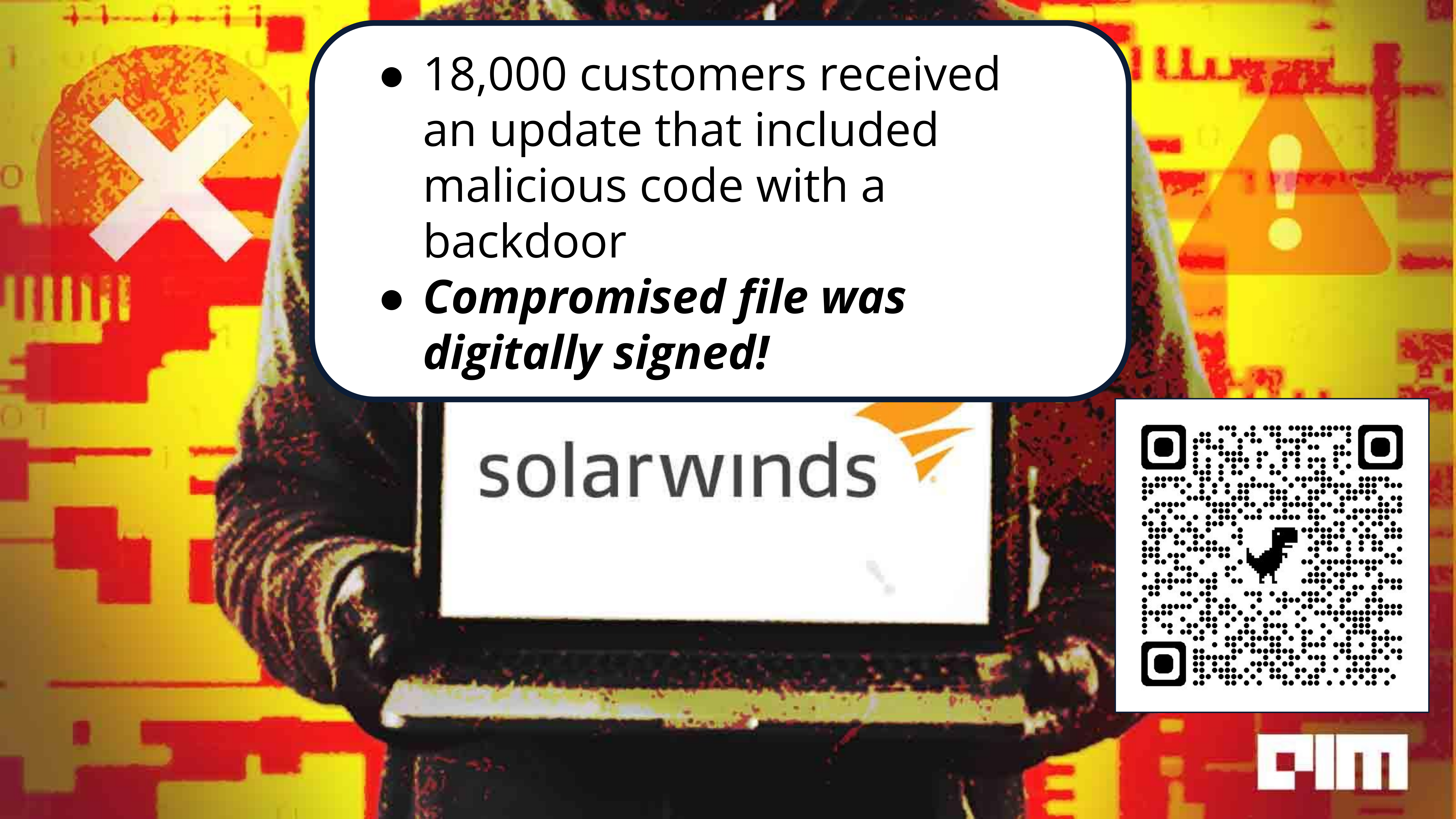
■ Percentage of code in codebases that was open source

🟥 Percentage of codebases containing high-risk vulnerabilities

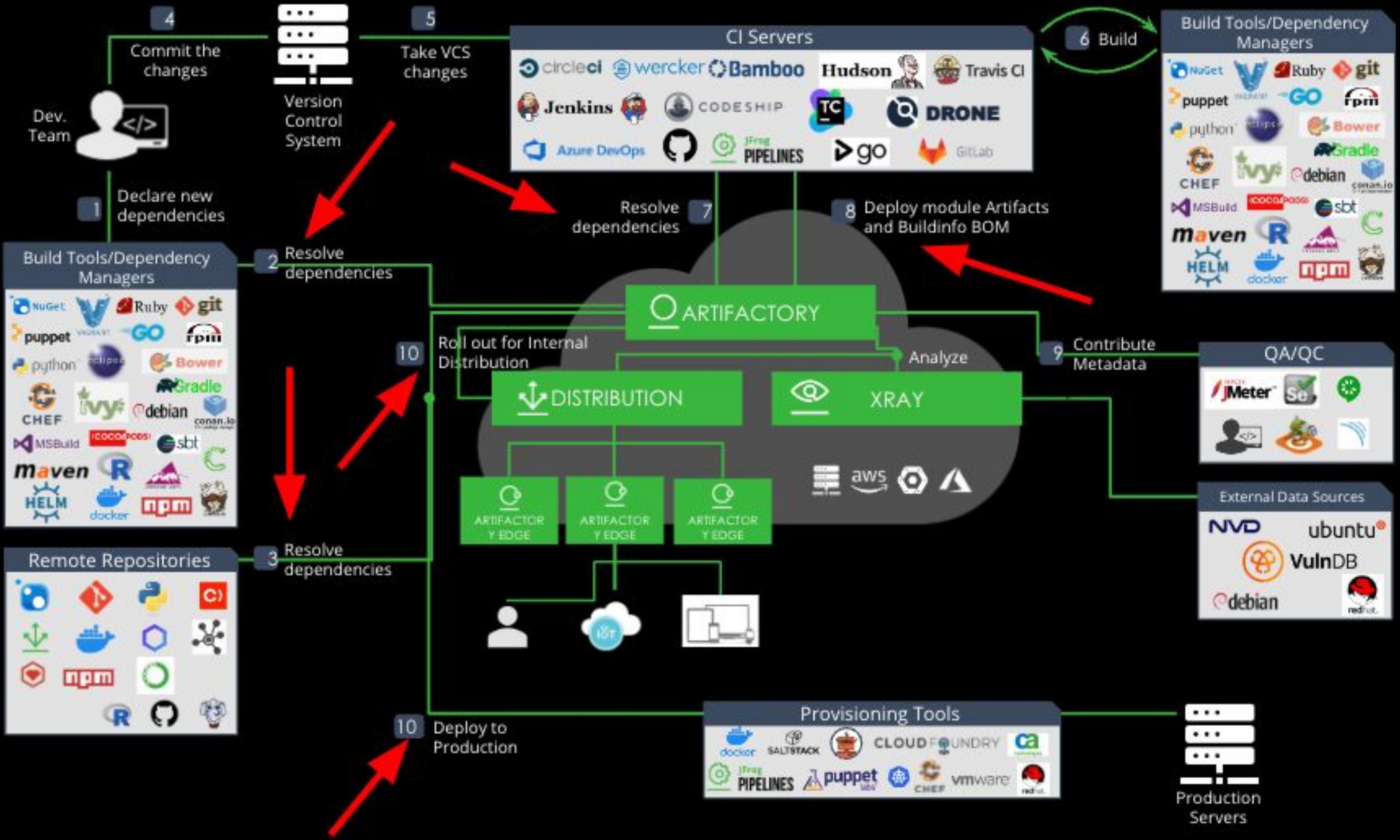
If developers didn't write insecure code... then we wouldn't have any of these problems!

IS IT ALL UP TO DEVELOPERS???

Well yes, but actually no
imgflip.com

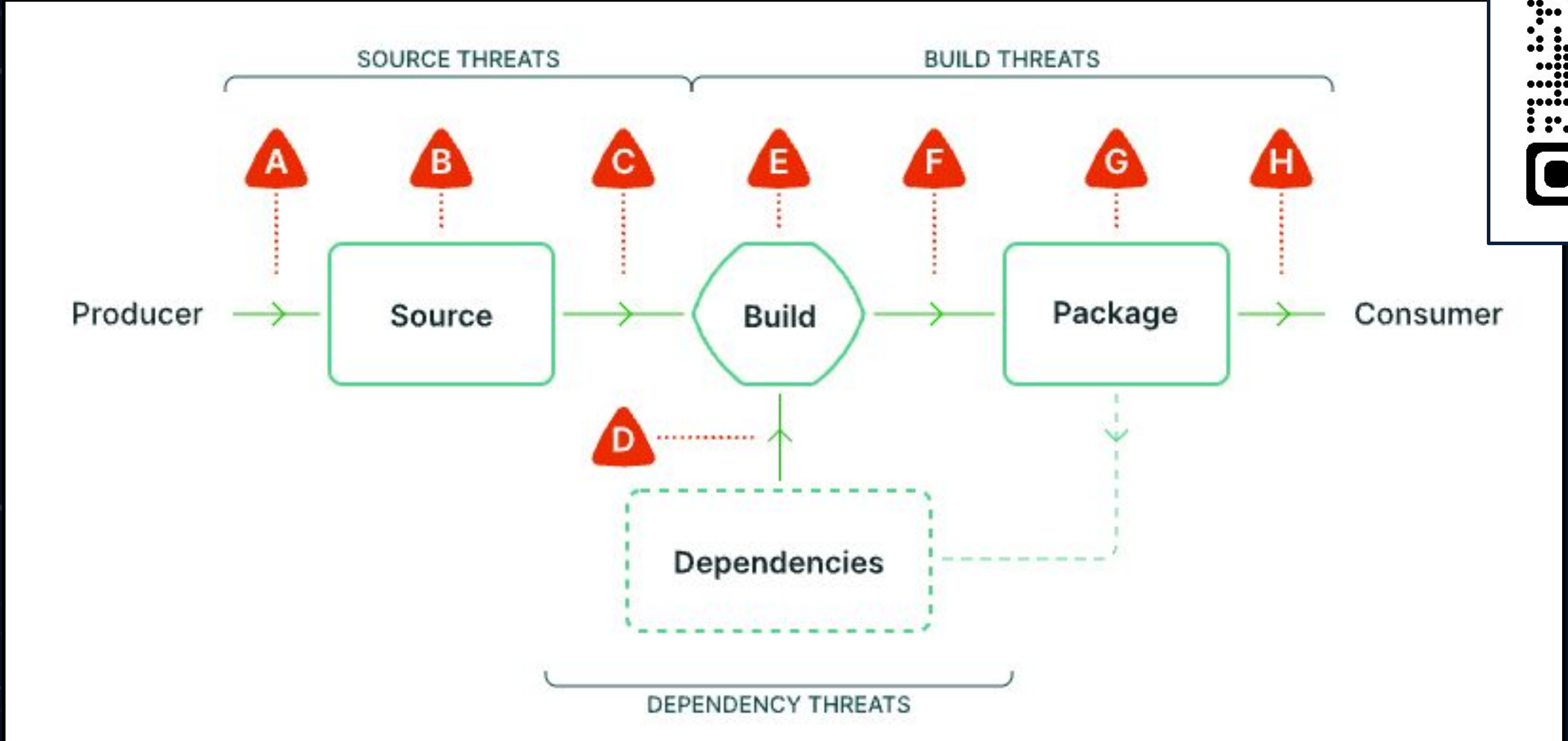


- 18,000 customers received an update that included malicious code with a backdoor
- ***Compromised file was digitally signed!***


4 Commit the changes
5 Take VCS changes
Version Control System
Dev. Team
1 Declare new dependencies
CI Servers
circleci
wercker
Bamboo
Hudson
Travis CI
Jenkins
CODESHIP
DRONE
Azure DevOps
JFrog PIPELINES
go
GitLab
6 Build
Build Tools/Dependency Managers
NuGet
Ruby
git
puppet
GO
rpm
python
Bower
Gradle
CHEF
ivy
debian
conan.io
MSBuild
sbt
maven
R
HELM
docker
npm
2 Resolve dependencies
7 Resolve dependencies
8 Deploy module Artifacts and Buildinfo BOM
ARTIFACTORY
10 Roll out for Internal Distribution
Analyze
9 Contribute Metadata
DISTRIBUTION
XRAY
ARTIFACTORY EDGE
ARTIFACTORY EDGE
ARTIFACTORY EDGE
aws
QA/QC
JMeter
Se
External Data Sources
NVD
ubuntu
VulnDB
debian
redhat
Remote Repositories
npm
R
3 Resolve dependencies
10 Deploy to Production
Provisioning Tools
docker
SALTSTACK
CLOUD FOUNDRY
ca
JFrog PIPELINES
puppet labs
CHEF
vmware
redhat
Production Servers

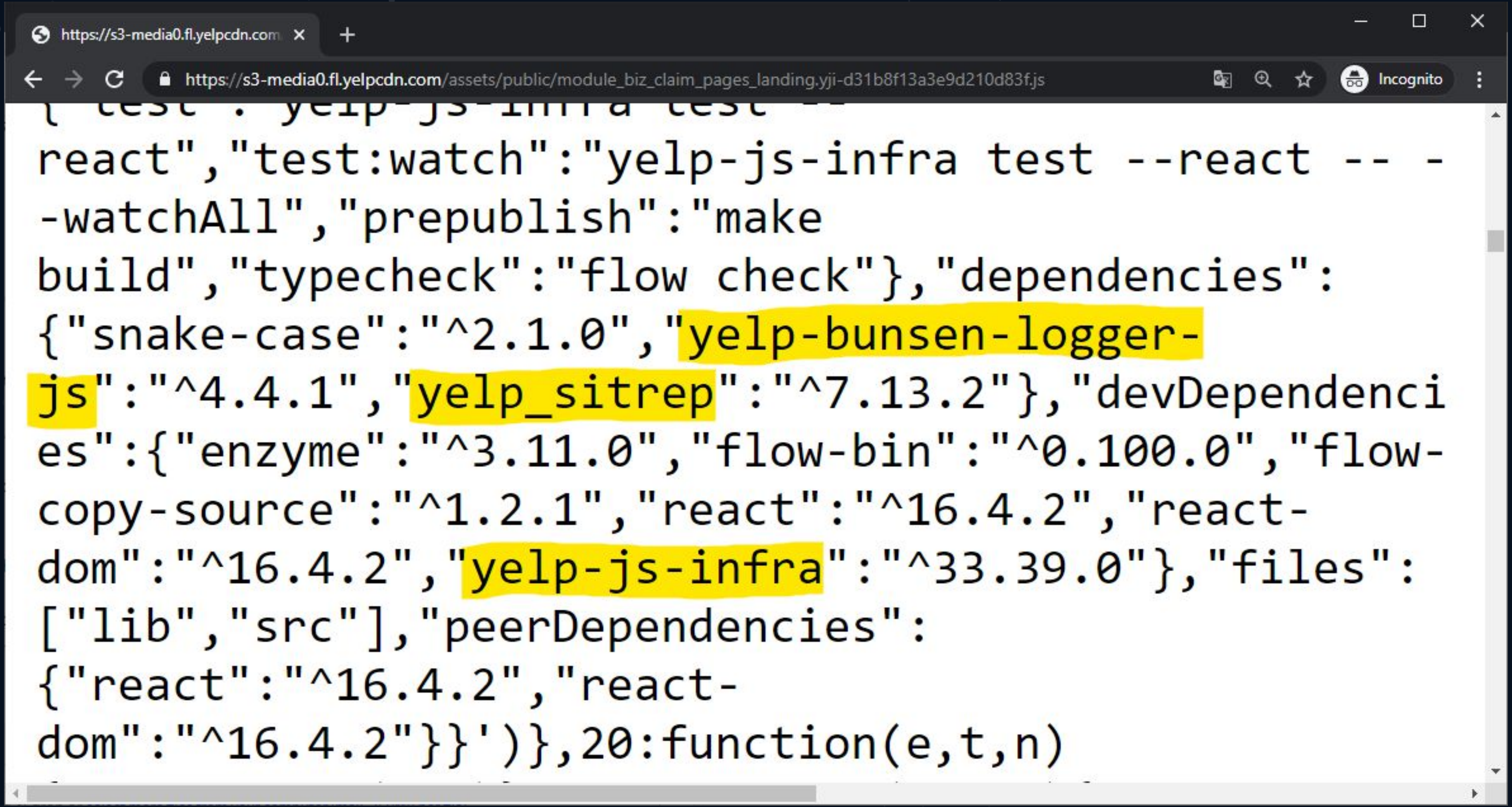
Supply-chain Levels for Software Artifacts

https://slsa.dev



SOURCE THREATS	DEPENDENCY THREATS	BUILD THREATS
A Submit unauthorized change	D Use compromised dependency	E Compromise build process
B Compromise source repo		F Upload modified package
C Build from modified source		G Compromise package repo
		H Use compromised package

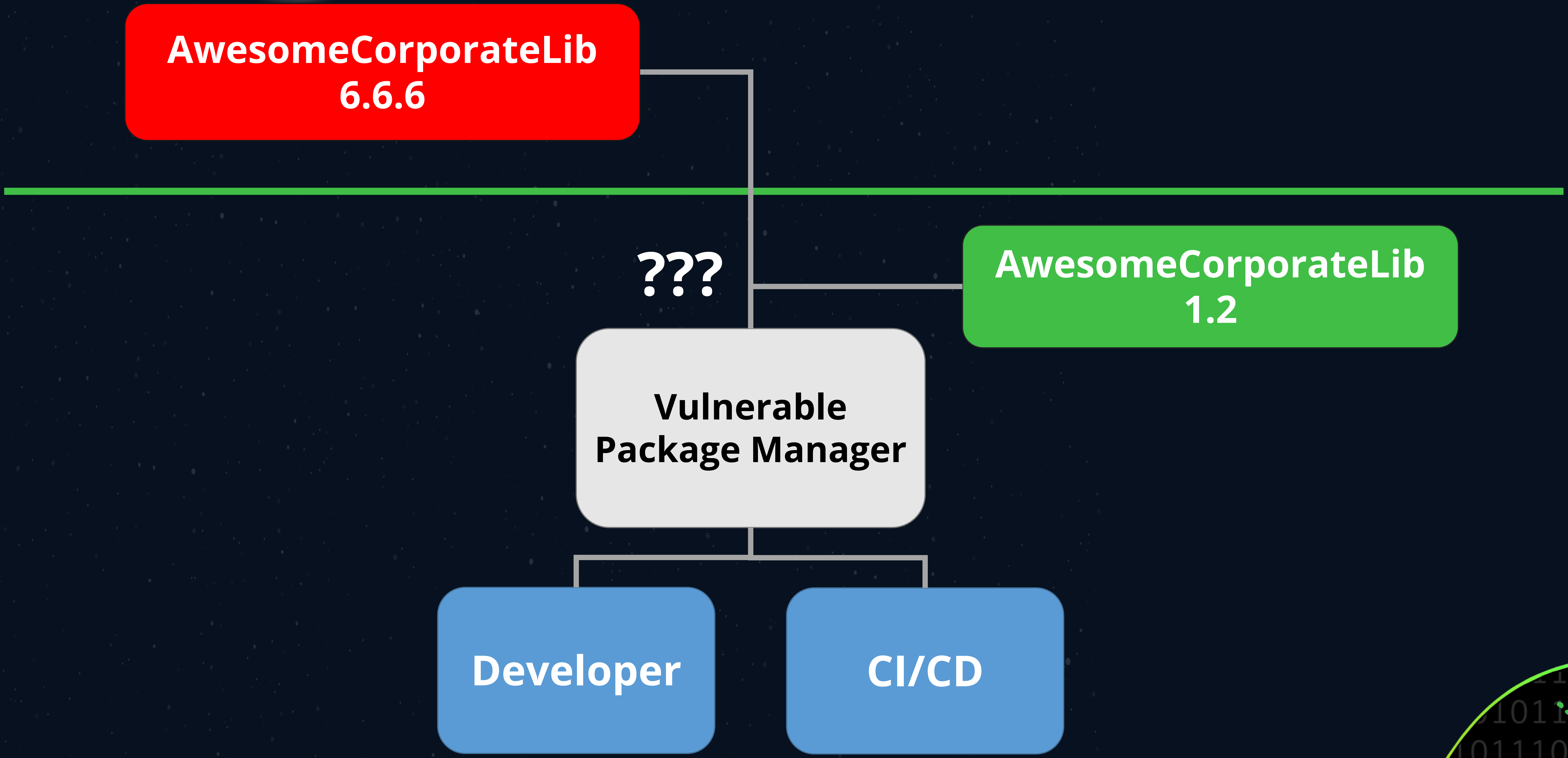
Dependency Confusion Attack - Package Mining


https://s3-media0.fl.yelpcdn.com
https://s3-media0.fl.yelpcdn.com/assets/public/module_biz_claim_pages_landing.yji-d31b8f13a3e9d210d83f.js
Incognito

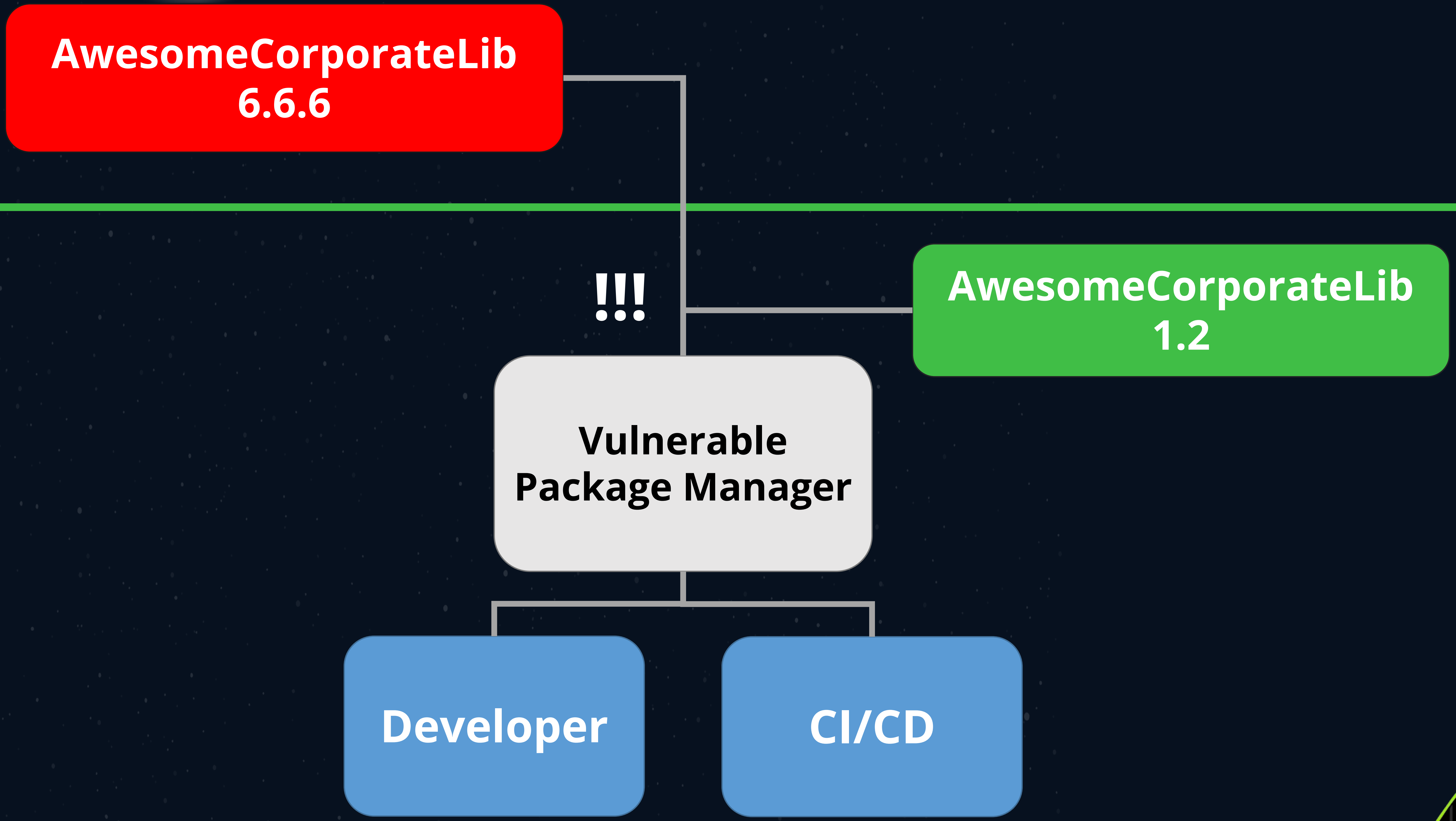
```
{"test":"yelp-js-infra test --
react","test:watch":"yelp-js-infra test --react -- -
-watchAll","prepublish":"make
build","typecheck":"flow check"},"dependencies":
{"snake-case":"^2.1.0","yelp-bunsen-logger-
js":"^4.4.1","yelp_sitrep":"^7.13.2"},"devDependenci
es":{"enzyme":"^3.11.0","flow-bin":"^0.100.0","flow-
copy-source":"^1.2.1","react":"^16.4.2","react-
dom":"^16.4.2","yelp-js-infra":"^33.39.0"},"files":
["lib","src"],"peerDependencies":
{"react":"^16.4.2","react-
dom":"^16.4.2"}}')},20:function(e,t,n)
```

https://medium.com/@alex.birsan/dependency-confusion-4a5d60fec610

Dependency Confusion Attack - Package Mining

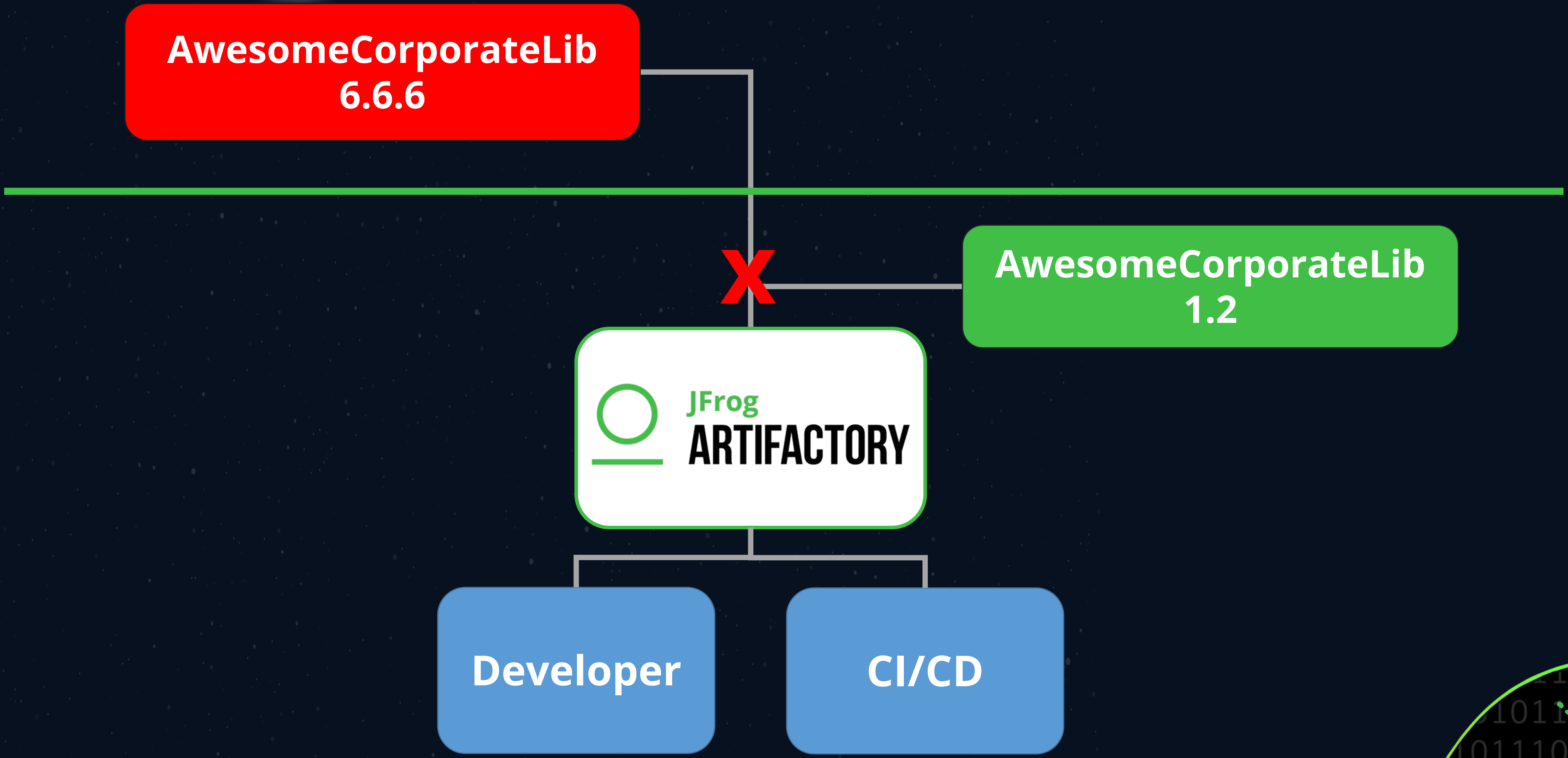


Dependency Confusion Attack - Package Mining



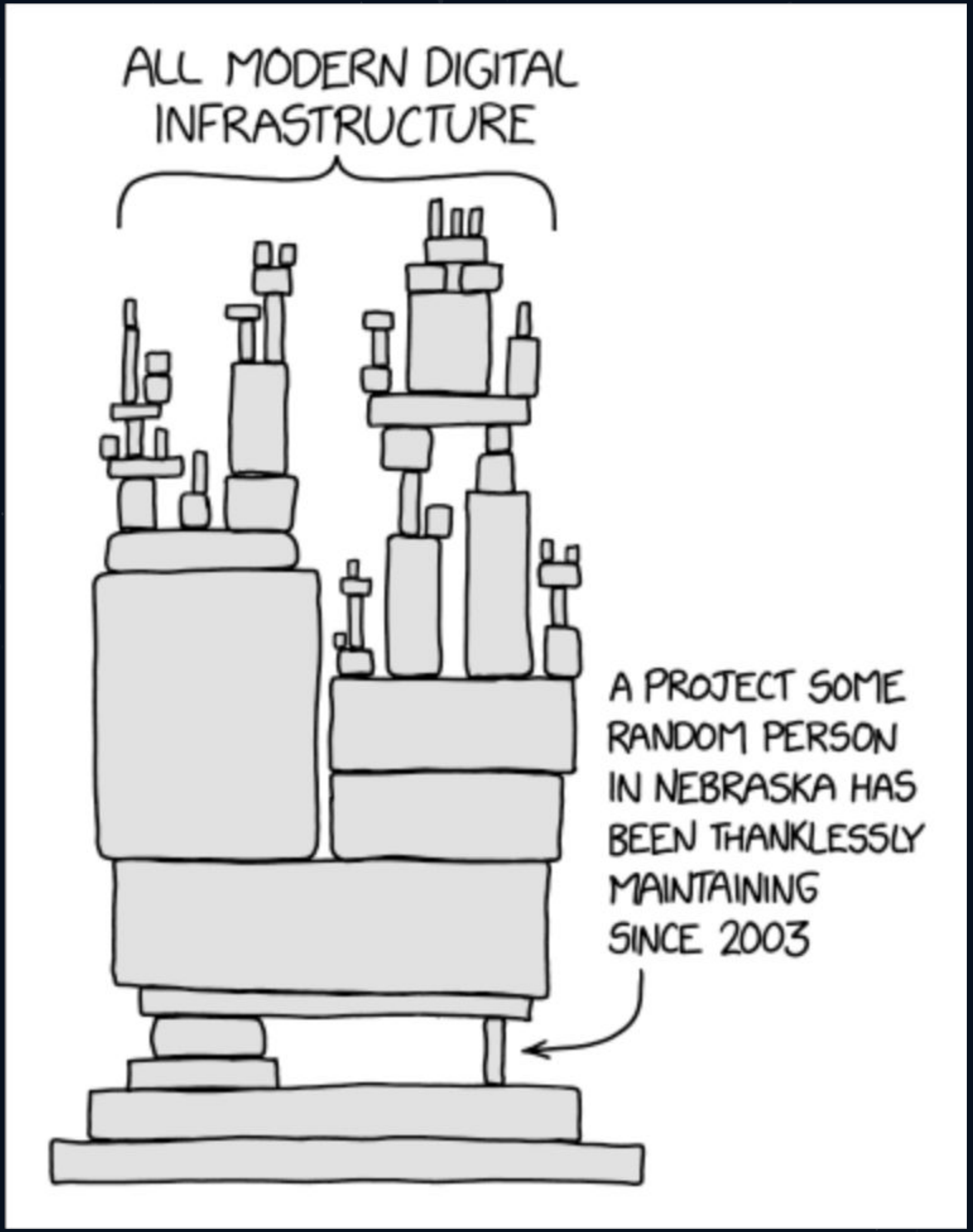
Dependency Confusion Attack - Package Mining

Dependency Confusion Attack - Package Mining



Dependency Confusion Attack 130,000 USD

Managing Open Source Dependencies


ALL MODERN DIGITAL INFRASTRUCTURE
A PROJECT SOME RANDOM PERSON IN NEBRASKA HAS BEEN THANKLESSLY MAINTAINING SINCE 2003

Attribution: https://xkcd.com/2347/

The Left-Pad Incident

1. Developer and *kik* organization couldn't come to an agreement on an npm package named *kik*
2. *npm* sided with the *kik organization*
3. Developer unpublished his *kik* package and **272** other packages! One of these was *left-pad*

Cameron Westland stepped in and published a functionally identical version of left-pad. v1.0.0, but many projects were explicitly requesting v0.0.3

https://qz.com/646467/how-one-programmer-broke-the-internet-by-deleting-a-tiny-piece-of-code

The Left-Pad Incident

```
module.exports = leftpad;
function leftpad (str, len,
ch) {
  str = String(str);
  var i = -1;
  if (!ch && ch !== 0) ch = '
';
  len = len - str.length;
  while (++i < len) {
    str = ch + str;
  }
  return str;
}
```

Tuesday, March 22, 2016

2:30 PM Pacific Time

Container Development

```
FROM untrustedParentImage
RUN apt update && apt install -y \
    somepackage \
    oldandvulnerablepackage=0.5
WORKDIR /myapp
COPY . .
RUN curl -sL \
    https://somewhere.com/script.sh | bash -
ENTRYPOINT [“start.sh”]
```

76%

Containers run as root

Sysdig 2022 Cloud-Native Security And Usage Report

(out of 3 million containers)

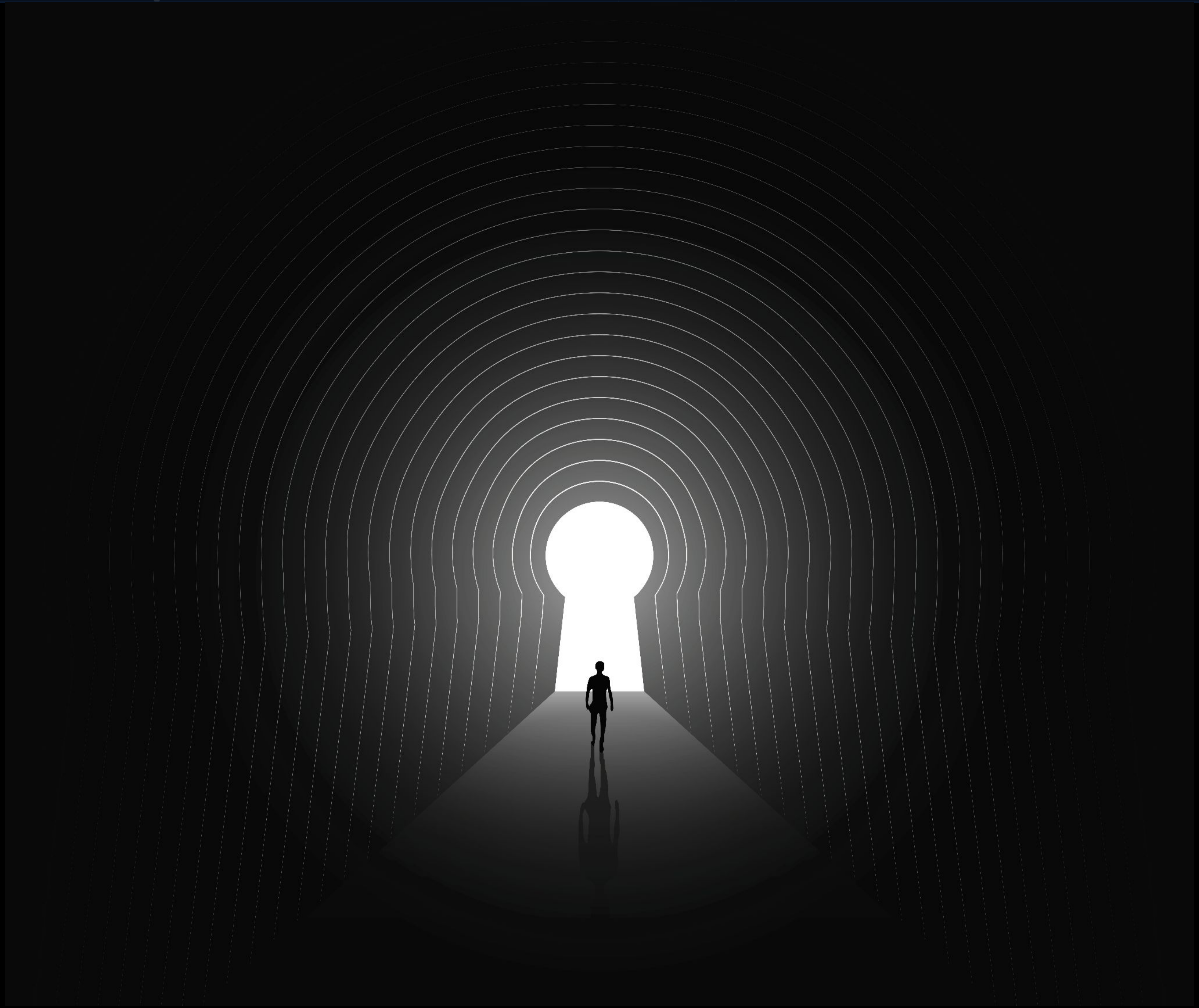
83%

Containers run as root

Sysdig **2023** Cloud-Native Security And Usage Report

(out of 7 million containers)

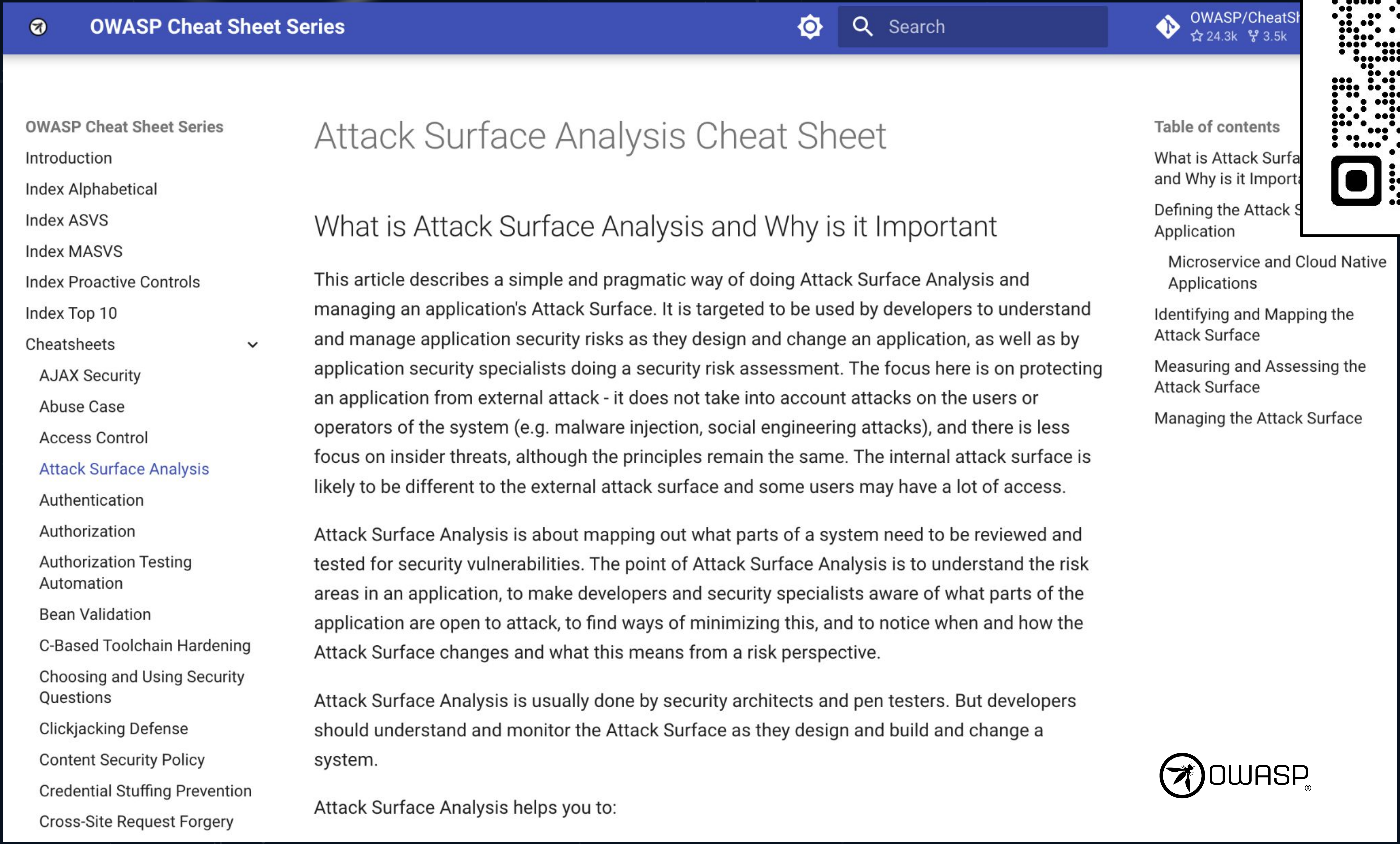
Is There Any Hope???



What Else Can We Do?

- **Educate ourselves**
- Don’t rely solely on public repos
- Manage dependencies!
- Manage permissions!
- Regularly scan your libraries & packages
- Keep up with maintenance
- Optimize CI/CD processes

OWASP Resources (Cheat sheets)



OWASP Cheat Sheet Series

Search

OWASP/CheatSh
24.3k 3.5k

OWASP Cheat Sheet Series
Introduction
Index Alphabetical
Index ASVS
Index MASVS
Index Proactive Controls
Index Top 10
Cheatsheets
AJAX Security
Abuse Case
Access Control
Attack Surface Analysis
Authentication
Authorization
Authorization Testing Automation
Bean Validation
C-Based Toolchain Hardening
Choosing and Using Security Questions
Clickjacking Defense
Content Security Policy
Credential Stuffing Prevention
Cross-Site Request Forgery

Attack Surface Analysis Cheat Sheet

What is Attack Surface Analysis and Why is it Important

This article describes a simple and pragmatic way of doing Attack Surface Analysis and managing an application's Attack Surface. It is targeted to be used by developers to understand and manage application security risks as they design and change an application, as well as by application security specialists doing a security risk assessment. The focus here is on protecting an application from external attack - it does not take into account attacks on the users or operators of the system (e.g. malware injection, social engineering attacks), and there is less focus on insider threats, although the principles remain the same. The internal attack surface is likely to be different to the external attack surface and some users may have a lot of access.

Attack Surface Analysis is about mapping out what parts of a system need to be reviewed and tested for security vulnerabilities. The point of Attack Surface Analysis is to understand the risk areas in an application, to make developers and security specialists aware of what parts of the application are open to attack, to find ways of minimizing this, and to notice when and how the Attack Surface changes and what this means from a risk perspective.

Attack Surface Analysis is usually done by security architects and pen testers. But developers should understand and monitor the Attack Surface as they design and build and change a system.

Attack Surface Analysis helps you to:

Table of contents

OpenSSF Trio of Free Courses

The Linux Foundation: **Secure Software Development: Requirements, Design, and Reuse**

The Linux Foundation: **Secure Software Development: Implementation**

The Linux Foundation: **Secure Software Development: Verification and More Specialized Topics**

What Can We Do???

- Educate ourselves
- **Don’t rely solely on public repos**
- Manage dependencies!
- Manage permissions!
- Regularly scan your libraries & packages
- Keep up with maintenance
- Optimize CI/CD processes

What Can We Do???

- Educate ourselves
- Don’t rely solely on public repos
- **Manage dependencies!**
- Manage permissions!
- Regularly scan your libraries & packages
- Keep up with maintenance
- Optimize CI/CD processes

What Can We Do???

- Educate ourselves
- Don’t rely solely on public repos
- Manage dependencies!
- **Manage permissions!**
- Regularly scan your libraries & packages
- Keep up with maintenance
- Optimize CI/CD processes

90%
of granted permissions are not used

JFrog

What Can We Do???

- Educate ourselves
- Don’t rely solely on public repos
- Manage dependencies!
- Manage permissions!
- **Regularly scan your libraries & packages**
- Keep up with maintenance
- Optimize CI/CD processes

What Can We Do???

- Educate ourselves
- Don’t rely solely on public repos
- Manage dependencies!
- Manage permissions!
- Regularly scan your libraries & packages
- **Keep up with maintenance**
- Optimize CI/CD processes

What Can We Do???

- Educate ourselves
- Don’t rely solely on public repos
- Manage dependencies!
- Manage permissions!
- Regularly scan your libraries & packages
- Keep up with maintenance
- **Optimize CI/CD processes**

QUESTIONS?

Melissa McKay
Developer Advocate, JFrog

@melissajmckay

linkedin.com/in/melissajmckay