Creating an AppSec Pipeline with containers in a week

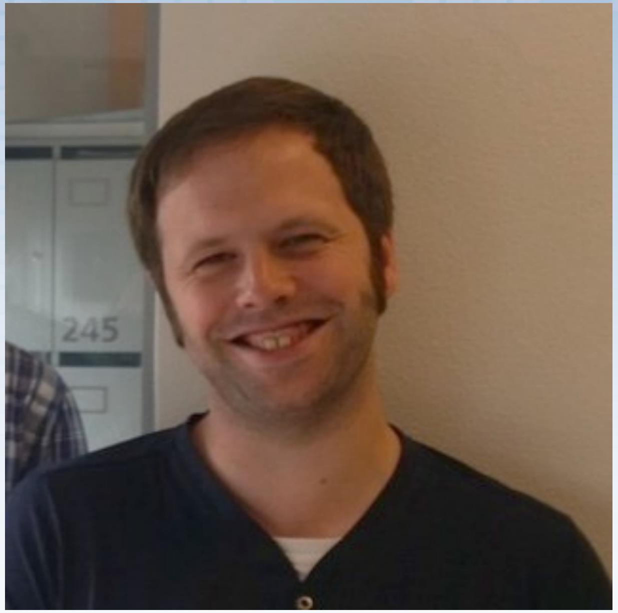
How we failed and succeeded

Jeroen Willemsen – OWASP benelux days

About me

Jeroen Willemsen
@commjoenie
jwillemsen@xebia.com

“Security architect”
“Full-stack developer”
“Mobile security”



Xebia

OWASP
Open Web Application Security Project

Agenda

- The challenge
- The solution
- Bumps on the road
- Recap

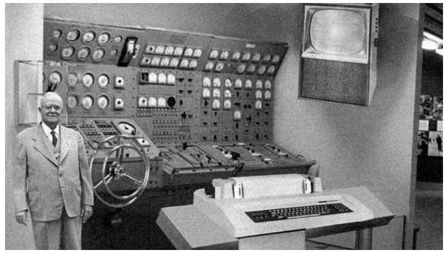
THE CHALLENGE

What could possibly go wrong?

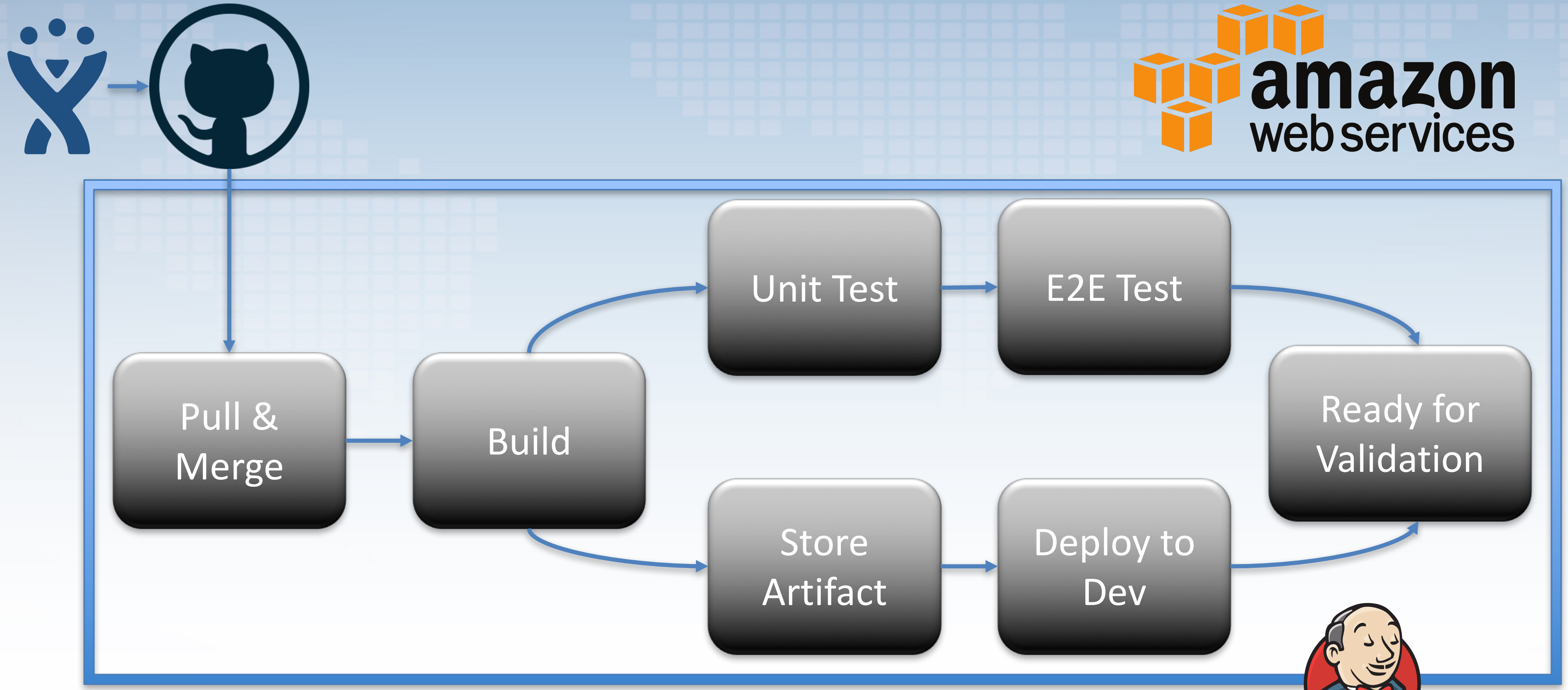
The Challenge

In a week?

The Challenge: The Landscape



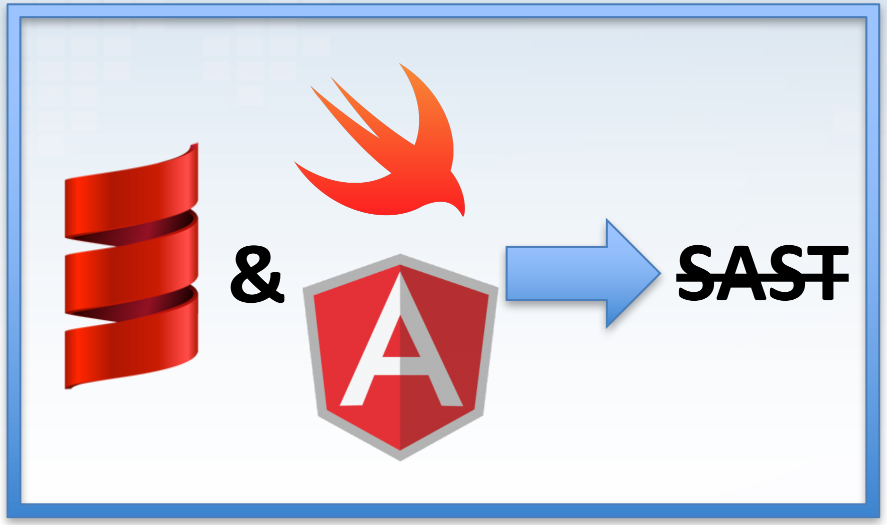
The Challenge: Existing workflow



The Challenge: New entries

- OWASP Dependency-Check
- License checkers
- clair
-

- **Etc…**



THE SOLUTION

We got there…kind off

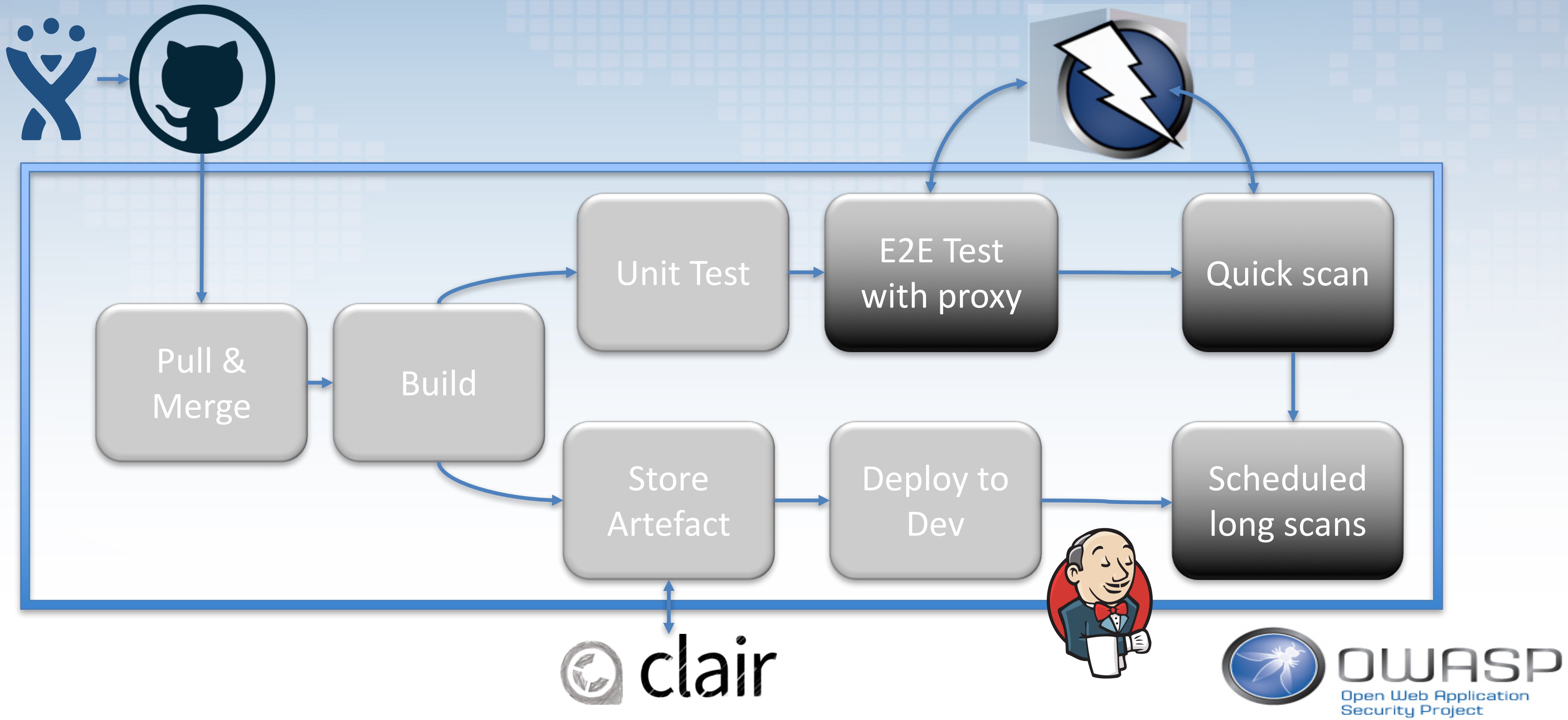
The Solution: Extend the build step

Add dependency & license checkers on top of quality tooling.

↓

Get feedback FAST!

The Solution: Feeding ZAP & BURP


Pull & Merge

Build

Unit Test

E2E Test with proxy

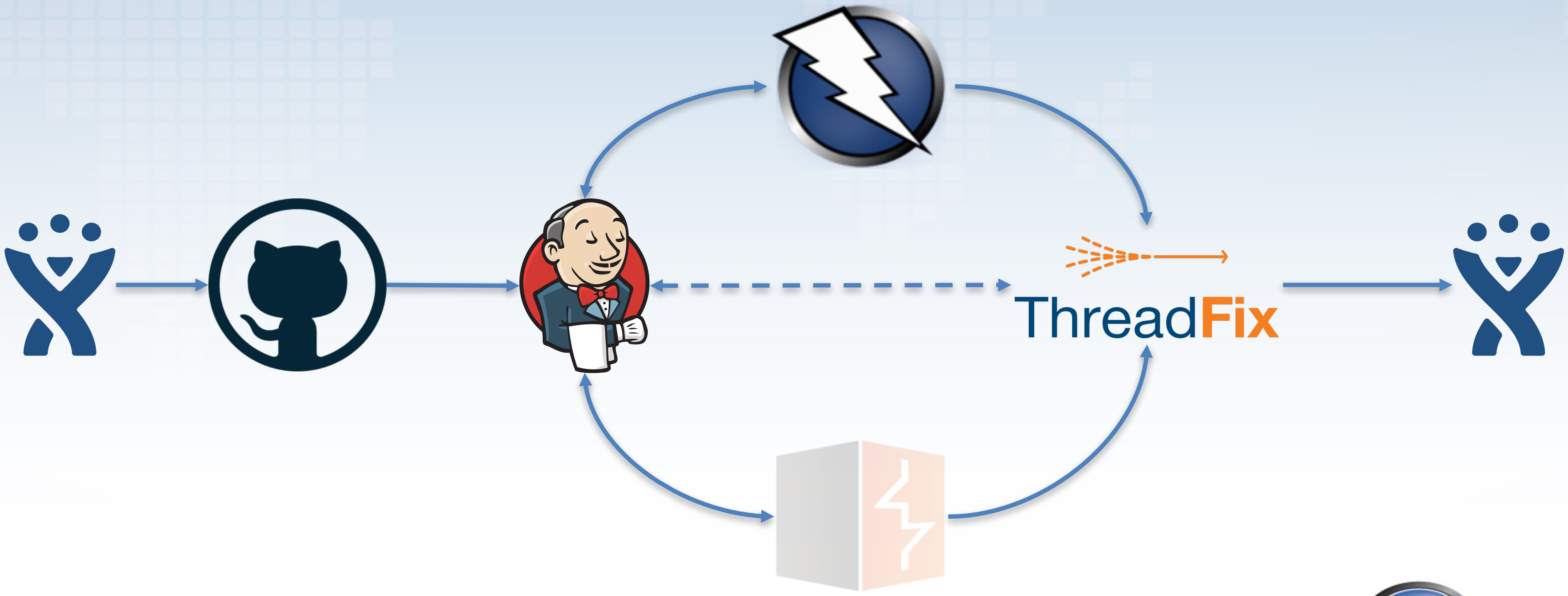
Quick scan

Store Artefact

Deploy to Dev

Scheduled long scans

The Solution: DAST & reporting



The Solution: Clair

- Run Clair on the created containers.

- *Todo: run Clair regularly on the registry, add whitelists & integrate with Threadfix.*
  - *By now this could be done differently using the clair-scanner from ArminC.*

The Solution: Containerize!

- Our tools embedded in containers:
  - + Less additional platform complexities
  - + Can run anywhere (locally / deployed)
  - + Easy to scale
  - - Still need to manage the data!
  - - More assets that might contain vulnerabilities
  - - Not perfect: still have to harden our assets

The Solution: A starting point

Example scan with a later version of the clair-scanner by Armin Coralic:

```
./clair-scanner app/threadfix example-whitelist.yaml
http://10.200.98.63:6060 10.200.98.63
```

```
2017-05-12 10:50:19.712897 I | Analyzing
014fdc7e45e4e7c5967856fc65d7bb5ff0b324fe4ef1ac8ce448843ab310416a
```

And 9 other layers...

Giving:

```
2017-05-12 10:50:19.854789 I | Image contains unapproved vulnerabilities:
[CVE-2017-6508]
```

The Solution: A starting point

- *2017-05-12 10:50:19.854789 I | Image contains unapproved vulnerabilities: [CVE-2017-6508]*
  - A vulnerability when creating the container
  - Not used during runtime
  - Clair cannot pick up the layers in which you create your own custom tooling (your own jar’s, executables, etc.)

The Solution: Did it work?

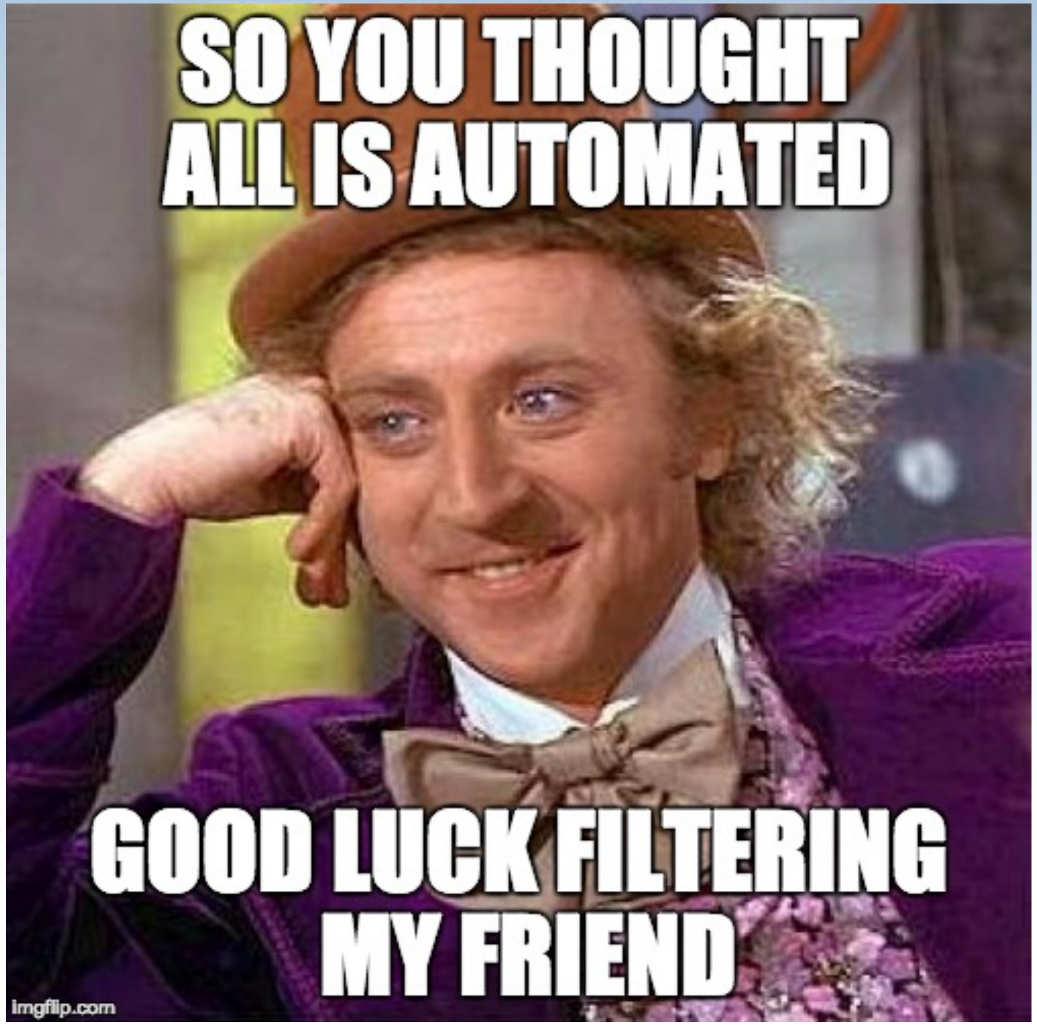
YES!

Not all components are in,
but feedback is already of great value

BUMPS ON THE ROAD

And their countermeasures

Bump 1: False positives

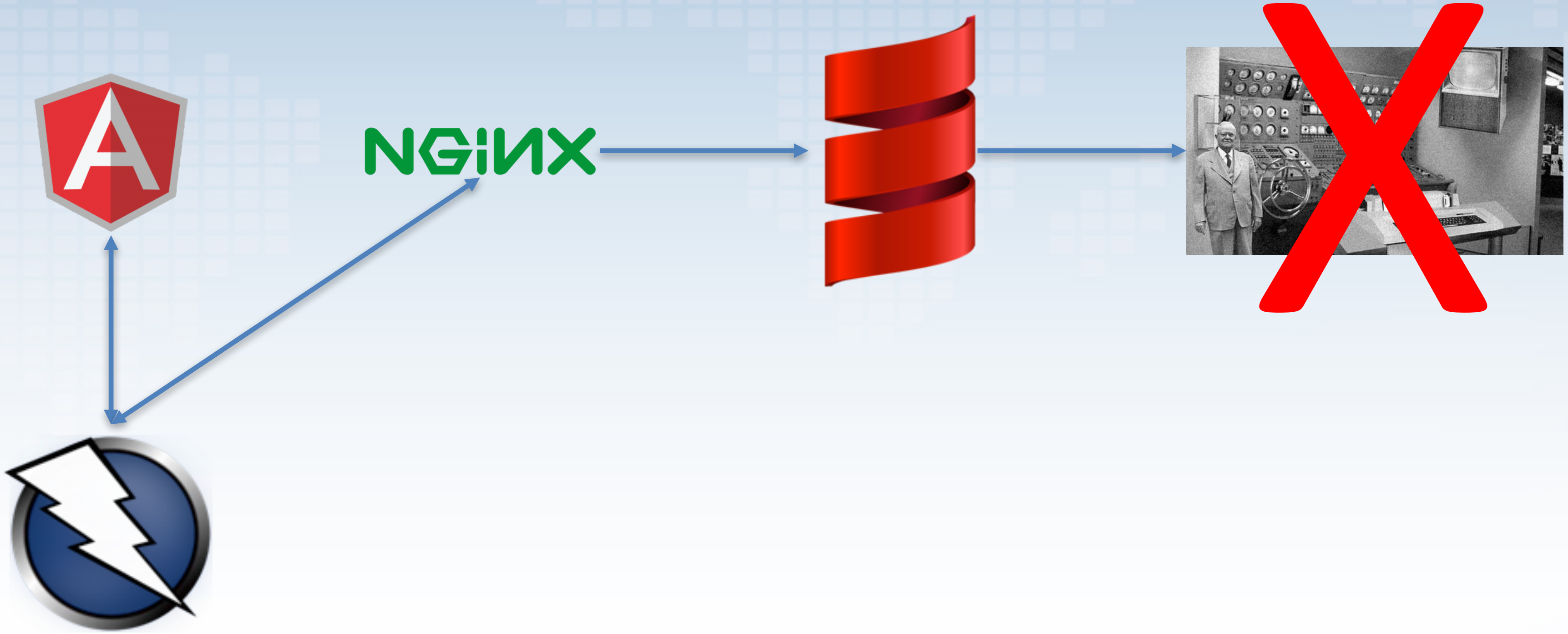


OWASP
Open Web Application
Security Project

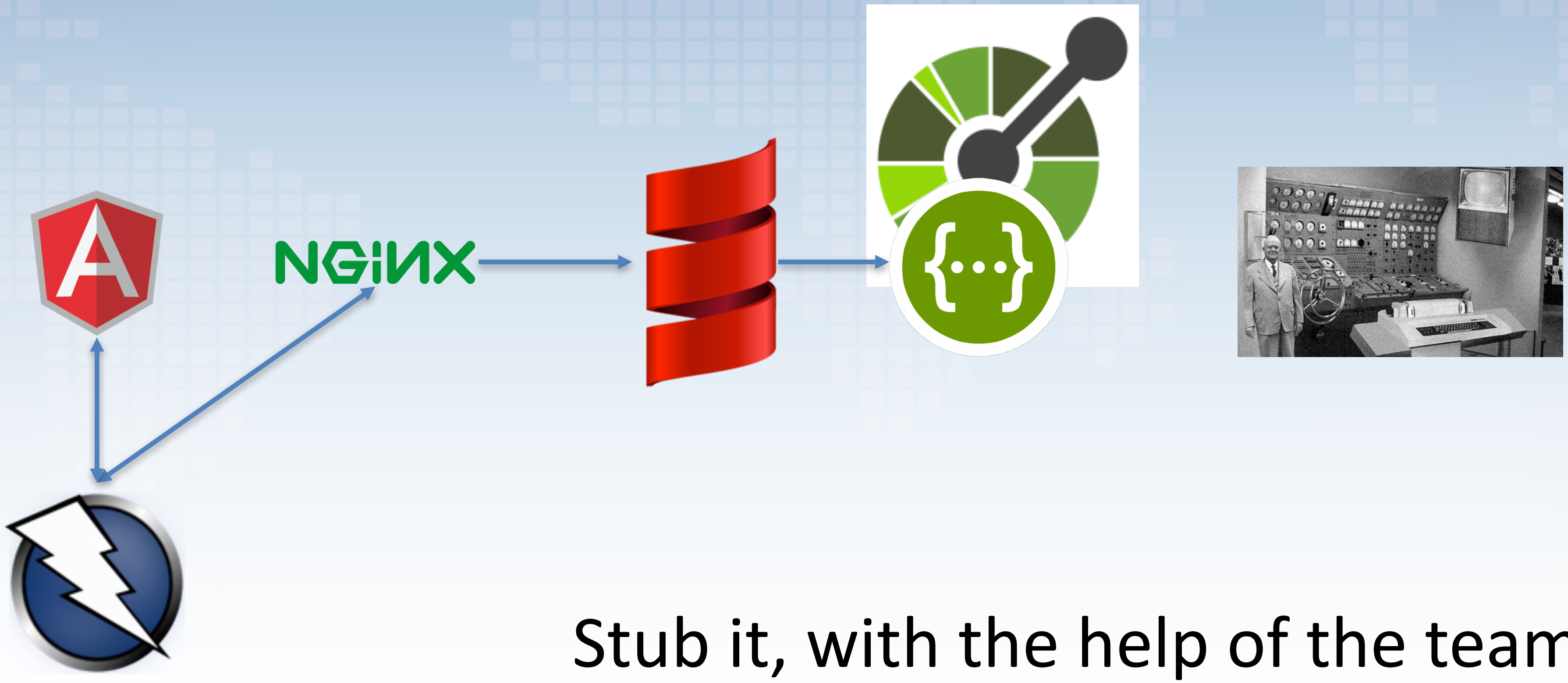
Bump 1: False positives

- Use settings/plugins in app → no scaling.
- Use a DB with a framework: BDD-Security
- Have an API ThreadFix & DEFECTdojo

Bump 2: Legacy APIs



Bump 2: Legacy APIs



Stub it, with the help of the teams

Test legacy APIs separately ☹

Bump 3: Not frustrate developers

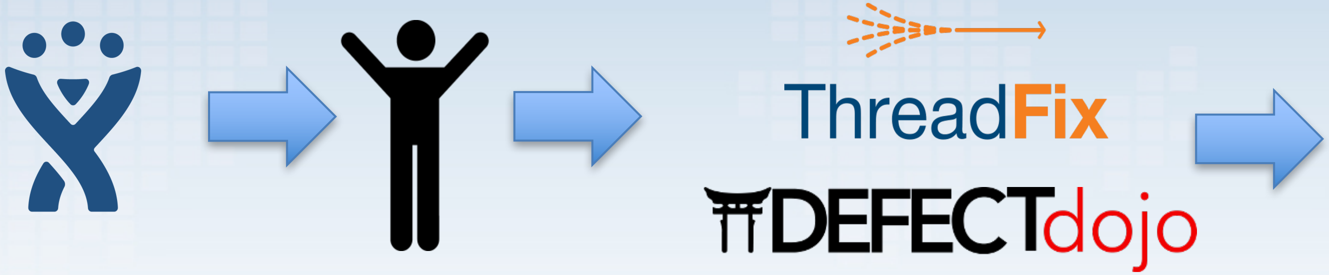
- Give feedback fast!
- Automate all the things!
- Be part of the team
- Filter & suppress false positives ASAP
- Use known tooling

Bump 4: Integrating Burpproxy

- Integration with Burp is not completed
  - Custom builds for containers
  - At time of testing: Additional extensions necessary to have a proper REST API

Bump 5: False negatives….

Security automation does not mean: no manual pentesting.



Even when you add more tools (which we have to…).

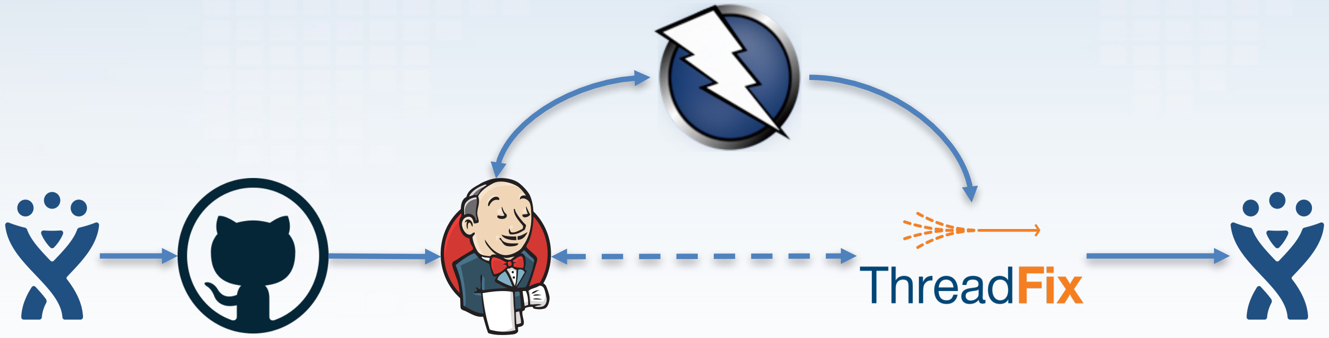
Bump 6: Platform team availability

OWASP
Open Web Application
Security Project

Lesson learned later on….

- The need for multiple pipelines…

Appsec-pipeline:



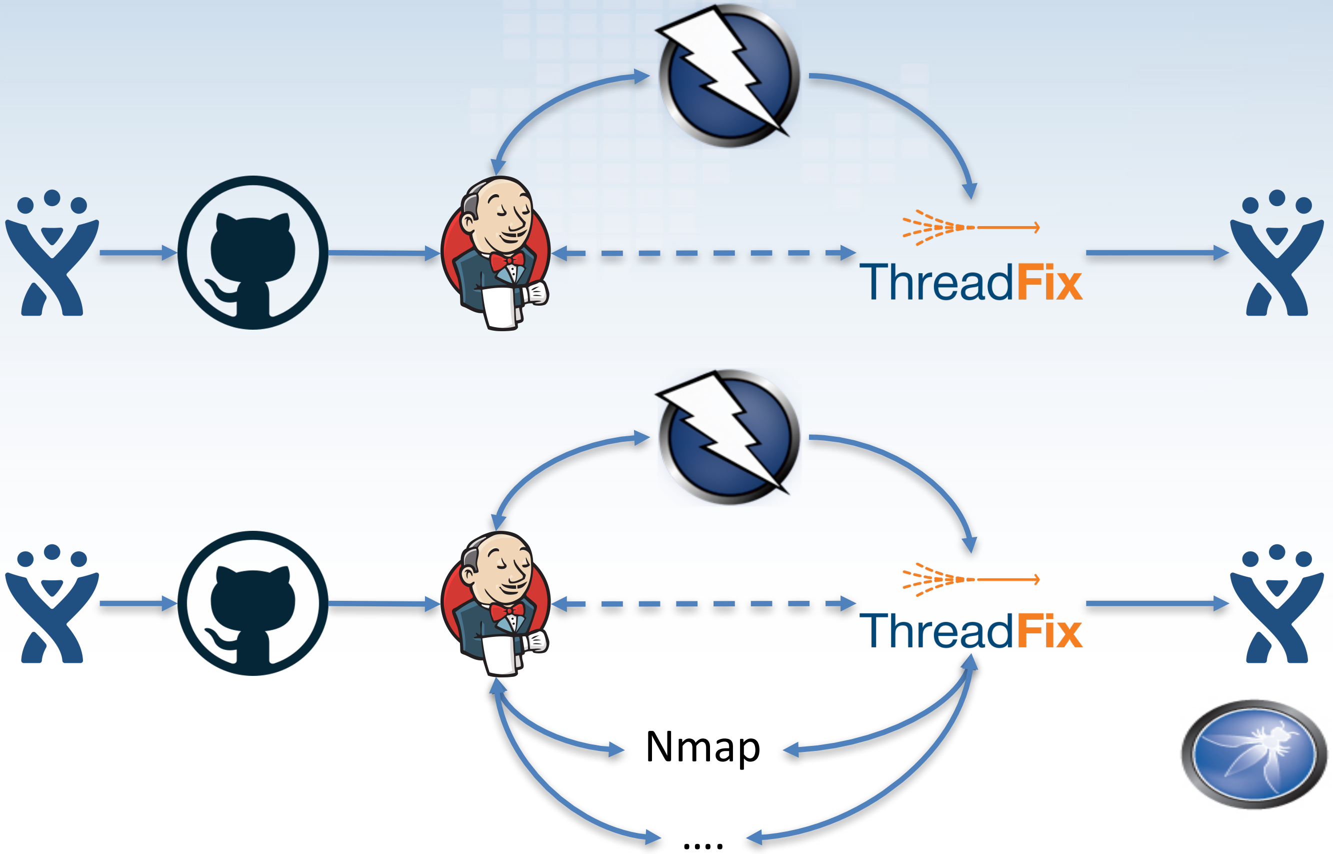
Open Web Application Security Project

Lesson learned later on….

- The need for multiple pipelines…

Appsec-pipeline:

Security pipeline:



Lesson learned later on

- Use the SWAGGER Api if possible
- Sooooooooo many tools to use:
  - Docker? Think of Docker Bench, OpenSCAP, Anchore, etc…
  - Infrastructure? Start with OpenVAS, OpenSCAP, Inspec
  - Inspect certificates: SSLlabs, testSSL.sh
  - Every language has its quality & security tooling

RECAP

To sum up

Recap

- Automate all the things: get feedback FAST.
- Containerize
- Filter false positives
- Stub legacy APIs
- HELP developers, DO NOT frustrate!
- Still a need for manual pentesting & reviewing.
- Get platform-team support!
- Every part of the pipeline is a blessing!

QUESTIONS?

THANK YOU!

OWASP
Open Web Application
Security Project