

# DevSecOps, day 3



# 2. Lab: End to end testing

# Get the test suites

- On your Dev Workstation, clone:
  - <https://github.com/unixerius/selenium-juiceshop>
- The test suite defaults to local testing.
  - It runs Selenium and JuiceShop in Docker.

# Preparing the tests

- Using "*docker ps*",
  - Check that JuiceShop is not running locally.
- "*cd*" into "*selenium-juiceshop*" and run:

```
$ docker-compose -f docker-compose-v3.yml up
```

On Apple ARM, use docker-compose-arm.yml !

# Preparing the tests

- Once Docker Compose is ready,
  - Visit <http://localhost:4444>
  - This should show Selenium Grid UI,
  - ... with zero queued jobs and three browsers.
- Also test that <http://localhost:3000> *does* work now.

# Run the test

- In another terminal (tab),
  - Inside the "*selenium-juiceshop*" repo,
  - Run:

```
$ mvn test
```

# The results

- During the test, you can also watch the browser!
  - In Selenium Edge, click the "camera" icon.
  - The password is "*secret*".
- Hopefully, the tests will report that all's okay,
  - Or maybe 1-3 tests fail.

# Run this against Azure?

- You can!
  - In file "*JuiceShopTests.java*",
  - Change the "websiteLink" variable,
  - to <https://unixeriusdso-team1.azurewebsites.net>
- Adjust "team1" to your own team.



# Breaking down the lab

- Run:

```
$ docker-compose -f docker-compose-v3.yml \
down
```

- ... or <ctrl><c> on the running instances.

# 2. Lab: More E2E testing

# Cypress

- [Cypress](#) is another Unit and E2E testing tool.
  - Cases are written in JavaScript.
- The Juice Shop team write their tests in Cypress.
  - See the "*test/cypress/*" dir in our Git repo.

# Running the tests

- Run:

```
$ cd ~/Team1JS  
$ npx cypress run --config video=false
```

# Running the tests

- With a GUI you can watch the browser.
- You can create a video by setting "video" to true.
  - They appear in "cypress/videos" in the repo.

```
$ cd ~/Team1JS  
$ npx cypress run --config video=true
```

# 5. Lab: Software Comp. Analysis



# Different security tests

- Software composition
- Secrets Detection
- SAST, static analysis of code
- DAST, dynamic analysis of running app
- Pen-testing
- ... and more.

# NPM and OSV

- We will use two different tools to test SCA.
  - The "NPM" built-in audit option,
  - The Open Source Vuln DB scanner.
- Many alternatives are possible.
  - Snyk is popular.



# NPM audit

- As developer you can run this locally:

```
$ cd ~/Team1JS
```

```
$ npm audit
```

# NPM audit

- Azure DevOps needs a specific setup ([source](#)).
- Can be done with the NPM plugin, or with Bash.
- Runs the NPM built-in SCA checks.
- Sample code is available: *pipeline-step2-SCA.yml*

# OSV Scanner

- As developer you can run this locally:

```
$ cd ~/Team1JS
$ npm install          # Already done, right?

$ docker run --rm -it -v ${PWD}:/src \
ghcr.io/google/osv-scanner \
-L /src/package-lock.json
```

# OSV Scanner

- Does not use the NPM built-in scanner.
  - Can be used as a standalone binary, or via Docker.
  - We can run the Docker-based scan in the pipeline.
- Sample code is available: *pipeline-step2-SCA.yml*

See: [Using OWASP Dependency Check](#)

# Reports

- After the run, the reports should be downloadable.

The screenshot displays a build interface for a project named "ITVitae team 1 JuiceShop". The build is identified by the ID "#20220805.3 continue on error in SCA". The status is "Failed" (indicated by a red circle with a white 'X'). The build was manually run by "Tess Sluijter-Stek". The repository is "ITVitae team 1 JuiceShop" and the branch is "dev" with commit "2a7cc607". The build started "Today at 13:59" and took "6m 56s" to complete. The "Related" section, which is highlighted with a red box, shows "0 work items" and "1 published". The "Errors" section, which is also highlighted with a red box, shows "82" errors. The first error is "Bash wrote one or more lines to the standard error stream." and the second is "npm WARN deprecated topo@3.0.3: This module has moved and is now available at @hapi/topo. Please update your dependencies as this vers...".

#20220805.3 continue on error in SCA  
ITVitae team 1 JuiceShop

Cancel

Summary

Manually run by Tess Sluijter-Stek

View 3 changes

Repository and version  
 ITVitae team 1 JuiceShop  
 dev 2a7cc607

Time started and elapsed  
 Today at 13:59  
 6m 56s

Related  
 0 work items  
 1 published

Tests and coverage  
 [Get started](#)

Errors 82

Bash wrote one or more lines to the standard error stream.  
build • npm\_audit\_sca • Bash

npm WARN deprecated topo@3.0.3: This module has moved and is now available at @hapi/topo. Please update your dependencies as this vers...  
build • npm\_audit\_sca • Bash

# Any differences?

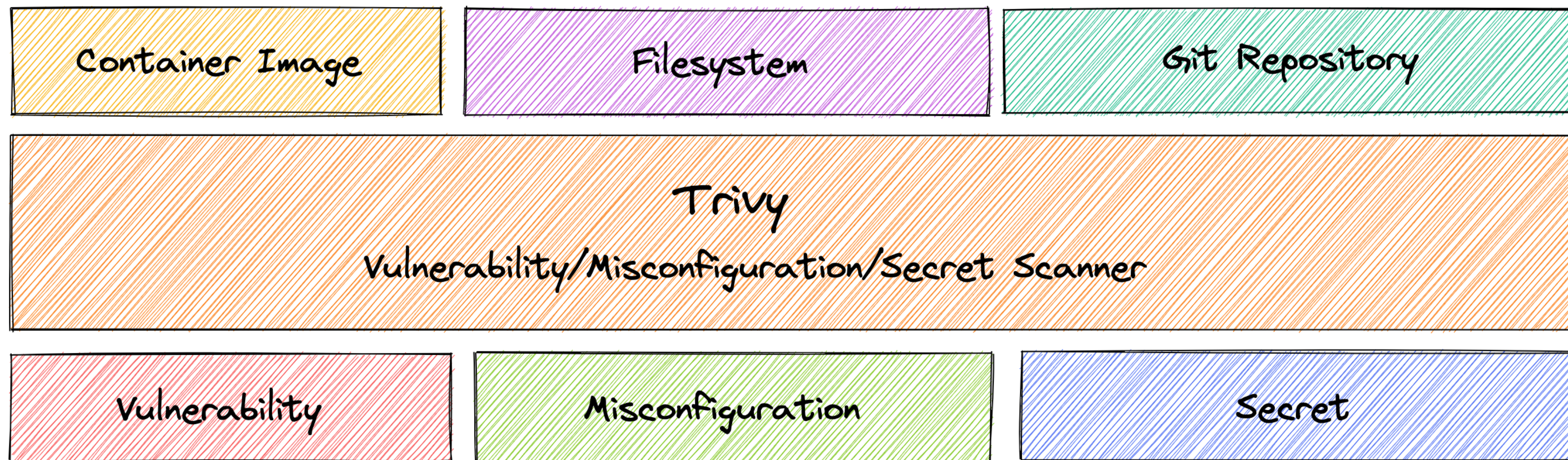
- OSV scans many languages and platforms.
- Both offer JSON and other output formats,
- Both offer flexible configuration.
- There is no single, right choice.

# 6. SCA for container images



# Trivy: another SCA approach

- Trivy is a cool tool, with lots of functionality!
  - Container images, configs, secrets, deps and libs



See: [Trivy documentation](#)



# Trivy

- As developer you can run this locally.
  - Here we use the public image, for demonstration.
  - Normally you use your own image.

```
$ docker run aquasec/trivy \
image bkimminich/juice-shop:v15.0.0
```

# Adding Trivy

- Trivy can scan local directories, but also
  - Container images, local or on a repository.
- There's a sample pipeline: *pipeline-step3-trivy.yml*

# Reports

- The sample makes JSON, but there's also tables.

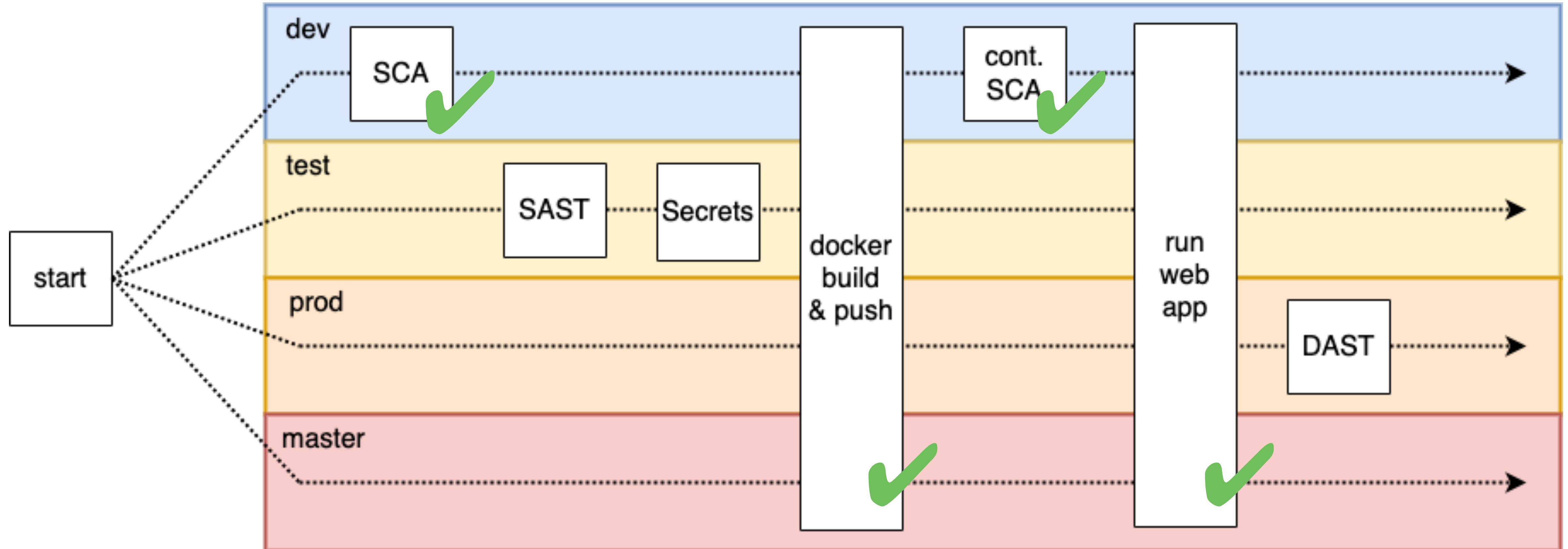
```
23 Total: 14 (UNKNOWN: 0, LOW: 0, MEDIUM: 7, HIGH: 6, CRITICAL: 1)
24
25 +-----+-----+-----+-----+-----+-----+
26 | LIBRARY | VULNERABILITY ID | SEVERITY | INSTALLED VERSION | FIXED VERSION | TI
27 +-----+-----+-----+-----+-----+-----+
28 | libc6   | CVE-2019-1010022 | CRITICAL | 2.31-13+deb11u3   |                | glibc: stack guard
29 |         |                  |          |                   |                | -->avd.aquasec.com/
30 +-----+-----+-----+-----+-----+-----+
31 |         | CVE-2018-20796   | HIGH    |                   |                | glibc: uncontrolled
32 |         |                  |          |                   |                | function check_dst_
33 |         |                  |          |                   |                | in posix/regexec.c
34 |         |                  |          |                   |                | -->avd.aquasec.com/
35 +-----+-----+-----+-----+-----+-----+
36 |         | CVE-2019-1010023 |          |                   |                | glibc: running ldd
37 |         |                  |          |                   |                | leads to code execu
38 |         |                  |          |                   |                | -->avd.aquasec.com/
39 +-----+-----+-----+-----+-----+-----+
```

# Checkpoint!

- Does everyone have:
  - A pipeline on the "dev" branch.
  - Which still builds + runs the webapp,
  - Plus which runs Trivy and NPM?
- Have you tested this?



# Our final pipeline goal



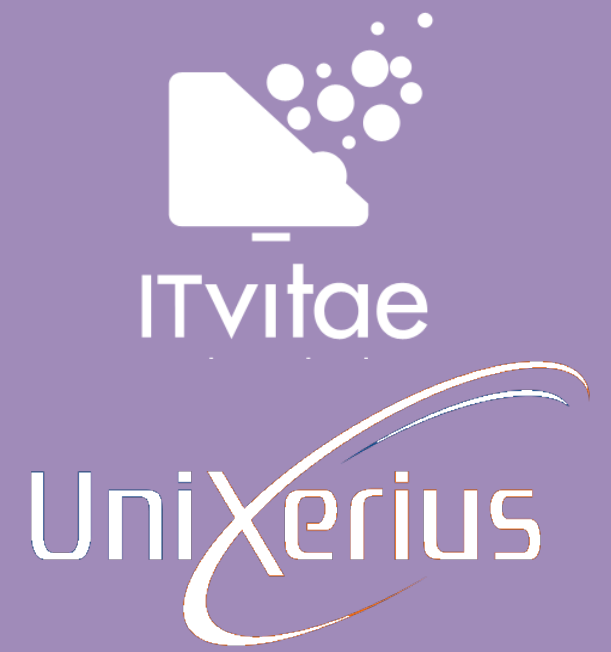
# Closing



# Tomorrow

- Static Analysis Security Testing (SAST)
- Secrets detection
- Dynamic Analysis Security Testing (DAST)

# Reference materials





# Resources

- [Atlassian - Types of software testing](#)
- [About Selenium](#)
- [Writing your first Selenium test script](#)
- [FIRST CVSS 3.1 Calculator](#)
- [CISA KEV catalog](#)
- [Software Composition Analysis](#) (in-depth)
- [Automate dependency updates with Renovate](#)
- [Renovate Github](#)

# Resources

- [The threat modelling field guide](#)
- [The threat modelling manifesto](#)
- [PluralSight learning path: threat modelling](#)
- [PDSO certified threat modelling professional](#)
- [Crowd sourcing the creation of persona non-grata](#)
- [Nixu CyberBogies](#) (PnG cards)