DevSecOps, day 5

1. Lab prep

Lab preparation

- You were asked to:
  - Request Nessus Essentials activation code

- We will work with Nessus Essentials.
- Startup takes very long, so we'll do that now.

Lab preparation (x86_64)

- On your lab VM, *cd* into "*~/Nessus*".
- Edit the "*docker-compose.yml*" file,
  - Add your activation code in the right place.
  - Set a username and password.

- Run: *docker-compose up*
- Once ready, it's at https://localhost:8834

Lab preparation (ARM)

- Run:
  - *docker run --restart=always -ti -p 8834:8834 nessus*

- Visit *https://localhost:8834*
  - Use the setup wizard for **Nessus Essentials**.
  - This will ask for your activation code.

Lab preparation

- The Docker container might restart.
  - After a few minutes, login to the web UI.
  - It should say "*plugins are compiling*".

- That's good; now we can do some theory. 🤓

3. Lab: Vulnerability scanning

Before we start

- Make sure that you have JuiceShop running.
  - Either run "*npm start*",
  - Or use your "*team1:dev*" container,
  - Or use the official container.

- Also test that SSH to user "vagrant" on the VM works.

Before we start

- These Nessus scans can take a long time. 🕰️⏳
  - We will start three at the same time.
  - Let's tune the scan, start it...
  - And then grab a drink. ☕

Logging in

- By now, Nessus should be up and running.
- Your username and password were setup,
  - In the *docker-compose.yml* file.
  - So go to https://localhost:8834

- **SKIP** the first discovery scan.

Start: webapp scan

- We will scan our local Juice Shop.

- In Nessus, define a new "*Webapp*" scan,
  - Set target to your lab VM's IP (<u>not localhost)</u>.
  - Set a custom discovery, limit to port 3000.

Start: network scan

- Define a new "Network scan".
  - Set the target to your lab VM's IP.
  - Set custom discovery to ports 1-1000.

- This takes a lot longer and runs in the background.

Start: credentialed scan

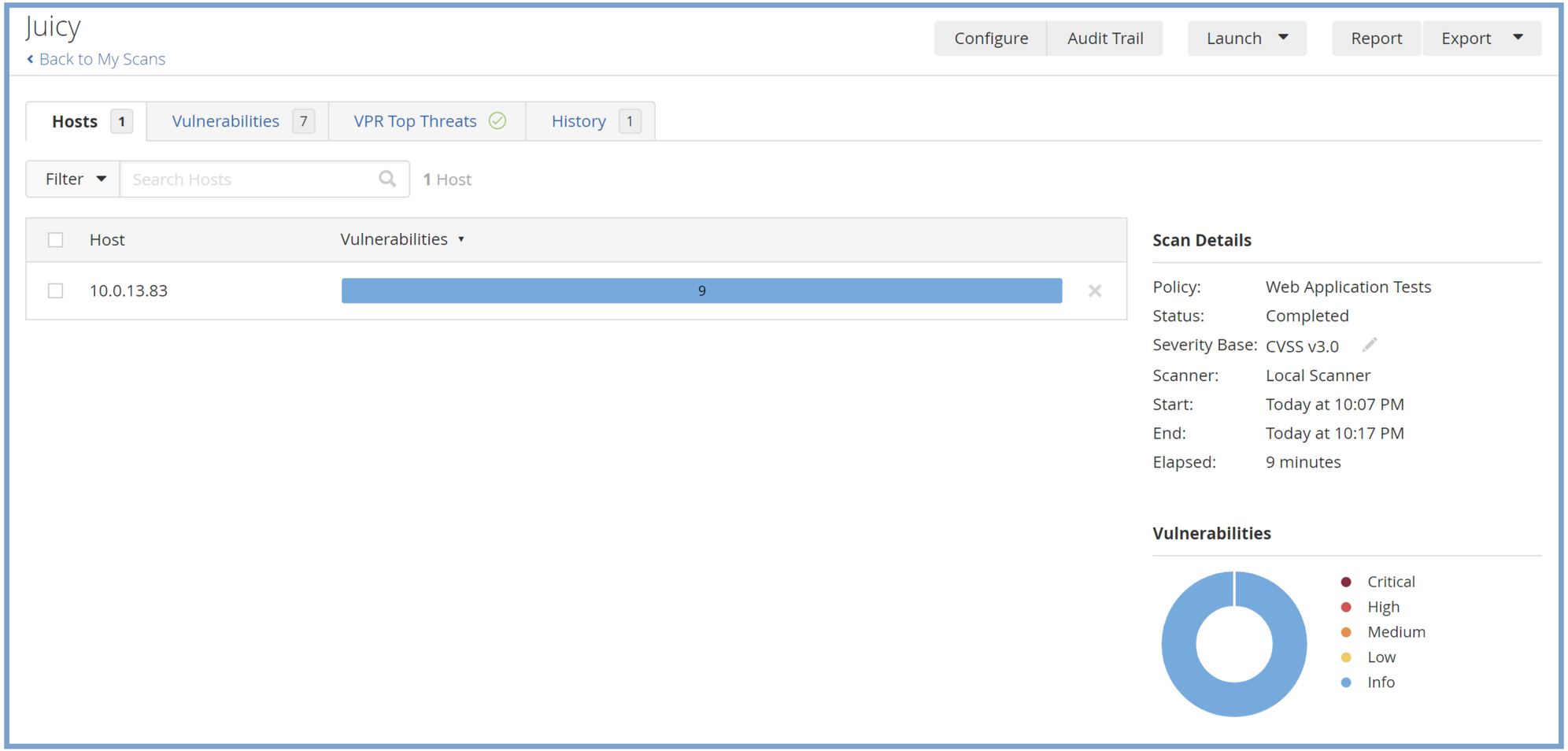
- Create a new "network scan".
  - Set the target to the lab VM's IP.
  - Set discovery to custom, ports 1-1000.

- Under the "Credentials" tab, select SSH.
  - Use the settings for your "vagrant" user.
  - Username, password, sudo, sudo password, etc.

The web app scan

- It will take a few minutes to do a quick scan.
- The results will be disappointing!
  - Juiceshop is bug ridden, with lots of vulnerabilities.
  - But these are not CVEs that Nessus detects.

The web app scan



The web app scan

Sev	Score	Name	Family	Count
INFO	...	2 HTTP (Multiple Issues)	Web Servers	2
INFO	...	2 Web Server (Multiple Issues)	Web Servers	2
INFO		External URLs	Web Servers	1
INFO		Missing or Permissive Content-Security-...	CGI abuses	1
INFO		Nessus Scan Information	Settings	1
INFO		Nessus SYN scanner	Port scanners	1
INFO		Web Application Sitemap	Web Servers	1

The web app scan

Hosts 1 | Vulnerabilities 7 | **VPR Top Threats** | History 1

Assessed Threat Level: **None**

No vulnerabilities have been found as prioritized by Tenable's patented Vulnerability Priority Rating (VPR) system.
To learn more about Tenable's VPR scoring system, see Predictive Prioritization.

No prioritized vulnerabilities found.

Why such bad results?

- Nessus can only look at the outside of the host.
  - It can only see running services that are open.

- Better results will be had with a "credentialed scan".
  - Nessus will check all installed packages!

Credentialed scan results

Hosts 1 | Vulnerabilities 39 | Remediations 28 | VPR Top Threats | History 1

Filter | Search Hosts | 1 Host

Host	Vulnerabilities					
10.0.2.15	9	14	10		56	

MIXED	...	33	Canonical Ubuntu Linux (Multip...	Ubuntu Local Security Checks	33	
MIXED	...	5	SSL (Multiple Issues)	General	5	
INFO	...	6	SSH (Multiple Issues)	General	6	
INFO	...	4	SSH (Multiple Issues)	Misc.	4	

Credentialed scan results

VPR Severity	Name	Reasons	VPR Score ▾	Hosts
CRITICAL	Ubuntu 18.04 LTS / 20.04 LTS : Linux kernel vulnerabilities (...	No recorded events	9.4	1
HIGH	Ubuntu 18.04 LTS / 20.04 LTS / 21.10 / 22.04 LTS : OpenSSL v...	No recorded events	8.4	1
HIGH	Ubuntu 18.04 LTS / 20.04 LTS / 21.10 : NSS vulnerabilities (U...	No recorded events	8.4	1
HIGH	Ubuntu 18.04 LTS / 20.04 LTS : Linux kernel vulnerabilities (...	No recorded events	7.4	1
HIGH	Ubuntu 16.04 ESM / 20.04 LTS / 21.10 : Linux kernel vulnera...	No recorded events	7.4	1
HIGH	Ubuntu 16.04 ESM / 18.04 LTS / 20.04 LTS : rsync vulnerabilit...	No recorded events	7.4	1
HIGH	Ubuntu 18.04 LTS / 20.04 LTS / 22.04 LTS : Libxslt vulnerabili...	No recorded events	7.4	1

7. Lab: Dynamic Analysis (DAST)

DAST: ZAP and Nuclei

- DAST scans have varying quality and outcomes.
  - Let's compare two tools in their baseline setting.

- We will compare:
  - OWASP ZAP
  - Nuclei

What is your IP?

- We will run the DAST tools in Docker.
  - Here, "localhost" will not work as target.
  - Make sure you have your Dev Workstation IP.

- For example, we will use:
  - http://192.168.56.11:3000

On your Dev Workstation

- We will use the Docker-based solution:

```
$ docker pull ghcr.io/zaproxy/zaproxy

$ docker run --rm ghcr.io/zaproxy/zaproxy \
  zap-baseline.py \
  -t http://192.168.56.11:3000
```

On your Dev Workstation

- We will use the Docker-based solution:

```
$ docker pull projectdiscovery/nuclei

$ docker run --rm projectdiscovery/nuclei \
  -u http://192.168.56.11:3000
```

Compare the results

- Did any of the scans trigger new flags in JuiceShop?

- How different are the results?
  - In amounts... in severity... in quality?

- Want to optimize Nuclei? Read this article.

In Azure DevOps

- With our pipeline we can use Docker,
  - It's a lot better than getting all dependencies!

- There's a sample pipeline: *pipeline-step6-dast.yml*

Pipeline additions

```
- job: owasp_zap
    steps:
   - task: Bash@3
      continueOnError: true
      displayName: run_zap
      inputs:
        targetType: 'inline'
        workingDirectory: '$(Build.SourcesDirectory)'
        script: |
          docker run --rm owasp/zap2docker-stable zap-baseline.py \
          -t '${{ variables.webappurl }}' | tee zap-result.txt

    - publish: '$(Build.SourcesDirectory)/zap-result.txt'
      artifact: zap-result.txt
```

Pipeline additions

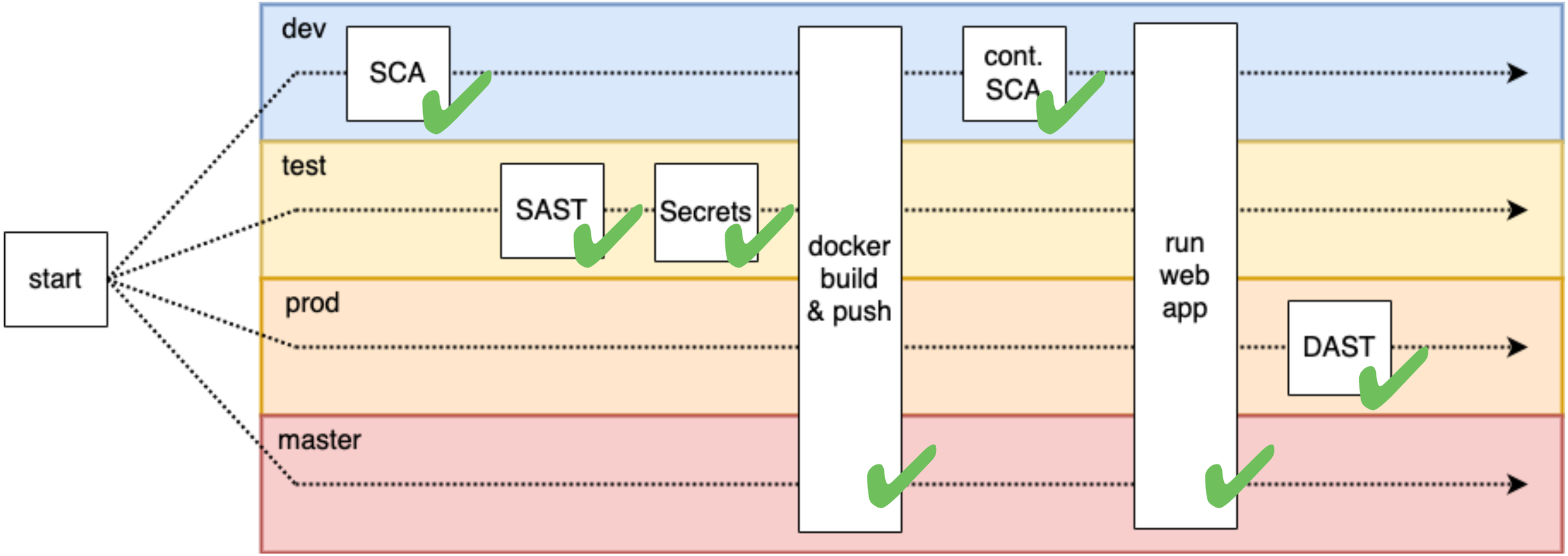
```
- job: nuclei
    steps:
    - task: Bash@3
      continueOnError: true
      displayName: run_nuclei
      inputs:
        targetType: 'inline'
        workingDirectory: '$(Build.SourcesDirectory)'
        script: |
          docker run --rm projectdiscovery/nuclei \
          -u '${{ variables.webappurl }}' | tee nuclei-result.txt

    - publish: '$(Build.SourcesDirectory)/nuclei-result.txt'
      artifact: nuclei-result.txt
```

Checkpoint!

- Does everyone have:
  - A pipeline on the "dev" branch.
  - Which still builds + runs the webapp,
  - And adds DAST after deployment?

- Have you tested this?

Our final pipeline goal



Closing

Where to, from here?

- If you really enjoyed this class,
  - There's a new job opportunity to explore!
  - Plus there's more training and even certification!

- For example: PDSO CDP

Thank you!

- It's been an awesome week!
  - I really enjoyed working with you.

Reference materials

Resources

- How to give the best pentest of your life
- Professor Messer - Pentesting
- "Pwning JuiceShop" ebook
- Debunking 5 DAST myths
- Vulnerability scanning vs pen-testing
- 7 Myths of AppSec automation

Resources

- Setting up an Azure WAF
- The Swiss cheese model
- PDSO CDP
- SBOM and VEX
- VEX use cases