

# DevSecOps with OpenSource

Hello \$username!

Welcome to the w00d00 workshop where you will learn (or recap) on some tools and techniques helping you to automagically identify vulnerabilities in applications and infrastructures as well as to address and track them in manageable form.

Speaking of automation, tools in this field have always been expensive or ~~crappy~~ not efficient, or both, which usually left the most of us, vulnerability hunters, with occasional OpenVAS/Nessus/Burp scans and, maybe (for millionaires) some SAST for the code. And no, SonarQube is not a SAST, unless you enable the right plugins in it, which developers usually don't do because these produce even more noise in addition to thousands of alerts they already get from it.

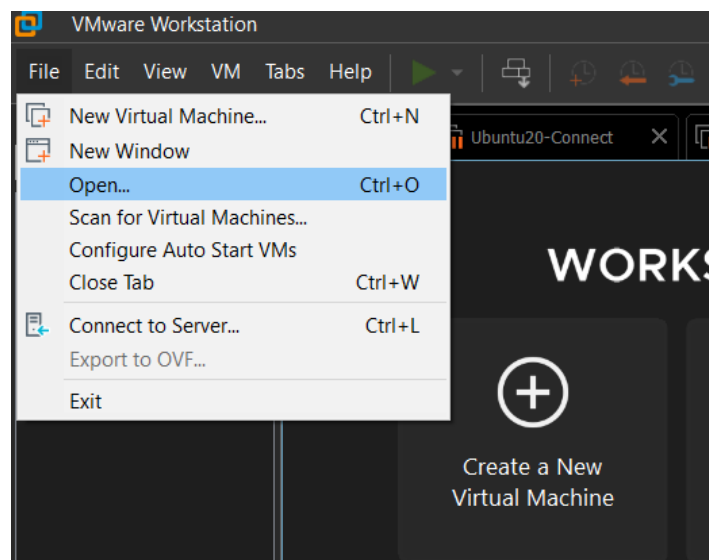
However, the time goes, and some good researchers have developed decent tools and even made them available for the public, so today I'd like to show you some ways you may want to build your vulnerability management process without spending too much time for developing own tools or money to buy expensive solutions.

## 1. Lab preps

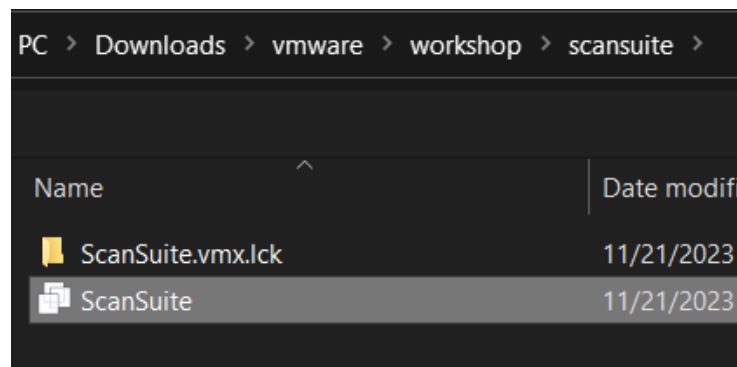
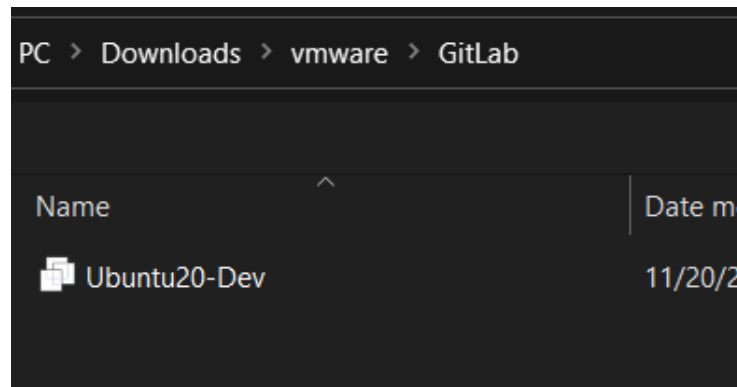
To run the labs please download **2 VMs - scansuite.zip and GitLab.zip** These are rather big images, and you may want to download them before coming to the conference as the Internet speed there may disappoint you.

To run VMs you'd need a VMWare Workstation/Fusion/Player/ESXi – something compatible with VMWare images.

Once you have downloaded the VMs, unzip the files and open the VMs via VMWare interface:



GitLab VM has a weird Ubuntu20-Dev name, that's OK just open it:



Boot the machines and select “I copied it” when asked by the VMWare.

Login to the **GitLab VM user dev with password gitlab1** over console and get it's IP address:

```

Gitlab
ScanSuite

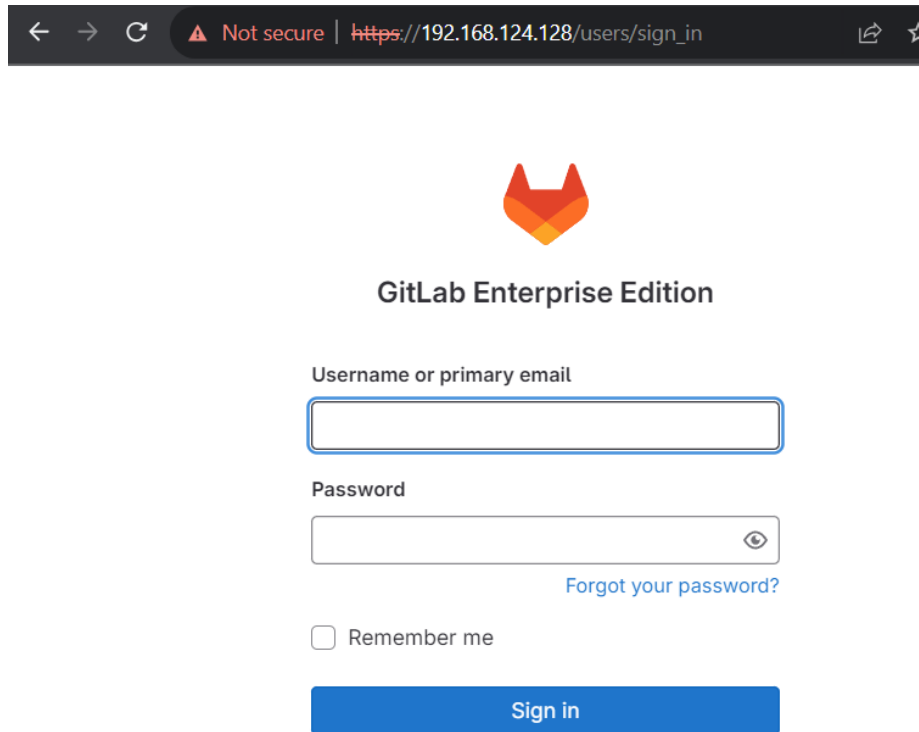
Usage of /: 50.3% of 23.70GB  Users logged in:
Memory usage: 79%          IPv4 address for
Swap usage: 25%

95 updates can be applied immediately.
To see these additional updates run: apt list --upg


*** System restart required ***
Last login: Mon Nov 20 10:59:06 UTC 2023 on tty1
dev@gitlab:~$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>
    inet 192.168.124.128 netmask 255.255.255.0
    inet6 fe80::20c:29ff:fe7b:3ef5 prefixlen 64
    ether 00:0c:29:7b:3e:f5 txqueuelen 1000 (
    RX packets 86719 bytes 101954360 (101.9 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 45348 bytes 30523711 (30.5 MB)
    TX errors 0 dropped 0 overruns 0 carrier

```

Open the browser and check if the GitLab server is loaded. That would be [https://YOUR\\_GITLAB\\_IP](https://YOUR_GITLAB_IP) :



← → ↻ ⚠ Not secure | https://192.168.124.128/users/sign\_in



## GitLab Enterprise Edition

Username or primary email

Password

[Forgot your password?](#)

☐ Remember me

**Sign in**

It may load 3-5 minutes after the machine is booted, just wait a bit if you don't see a login screen.

Obviously, you need to agree with the usage of self-signed certificates in every web app we will be using.

Login to the GitLab **user root** with **password NOaQYCqYlOwUzGGs9+v0Wogmy5bVmFA2bmlxa/GTBJQ=**

Now, let's move to the second VM. Boot it and login as **scanbot / scanbot**

Execute the script in the home folder to specify the GitLab VM IP address with sudo:

**sudo ./conn-gitlab.sh**

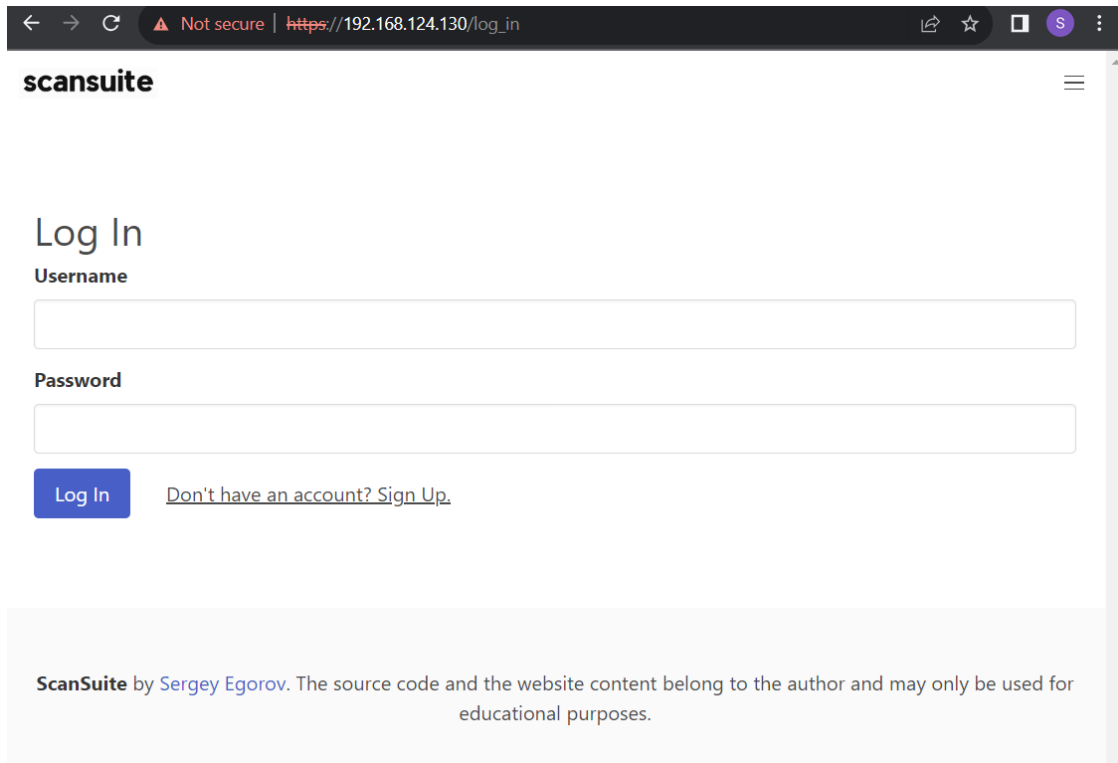
```
scanbot@scansuite:~$ sudo ./conn-gitlab.sh
[sudo] password for scanbot:
Enter GitLab IP address: 192.168.124.128
PING gitlab-local (192.168.124.128) 56(84) bytes of data.
64 bytes from gitlab-local (192.168.124.128): icmp_seq=1 ttl=64 time=0.307 ms
64 bytes from gitlab-local (192.168.124.128): icmp_seq=2 ttl=64 time=0.275 ms
64 bytes from gitlab-local (192.168.124.128): icmp_seq=3 ttl=64 time=1.23 ms
64 bytes from gitlab-local (192.168.124.128): icmp_seq=4 ttl=64 time=0.256 ms

--- gitlab-local ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3031ms
rtt min/avg/max/mdev = 0.256/0.516/1.227/0.410 ms
scanbot@scansuite:~$
```

At the end it should successfully ping the gitlab-local host, which is your GitLab VM.

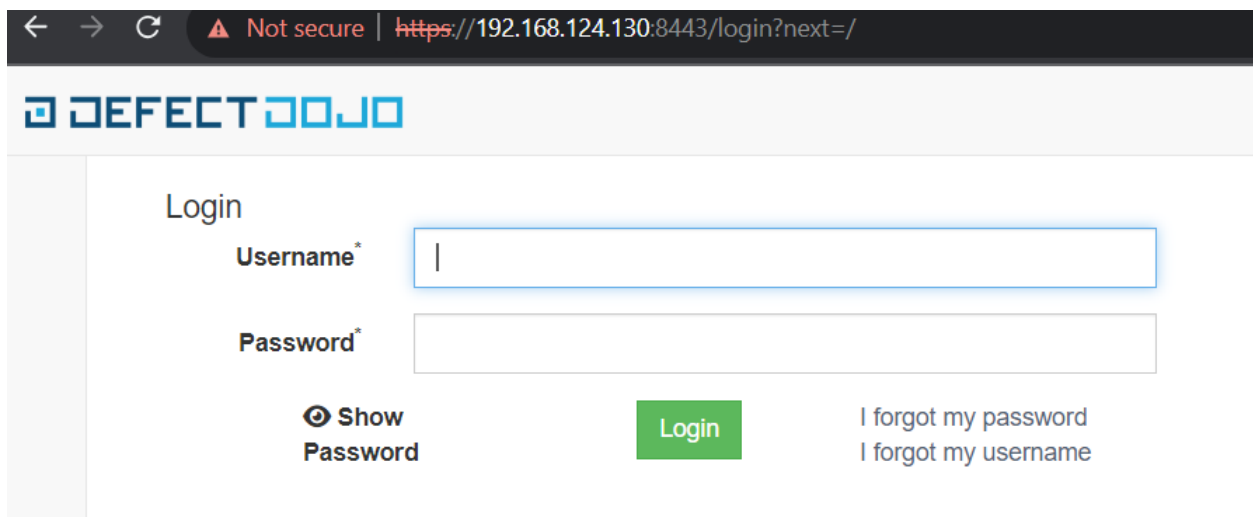
If gitlab-local is not reachable, try to troubleshoot by separately executing commands in the script with sudo privs

Now, open the ScanSuite web interface via [https://YOUR\\_SCANSUITE\\_IP](https://YOUR_SCANSUITE_IP) . Login with **user admin** and **password COzAVQwewpBVAMHNj**



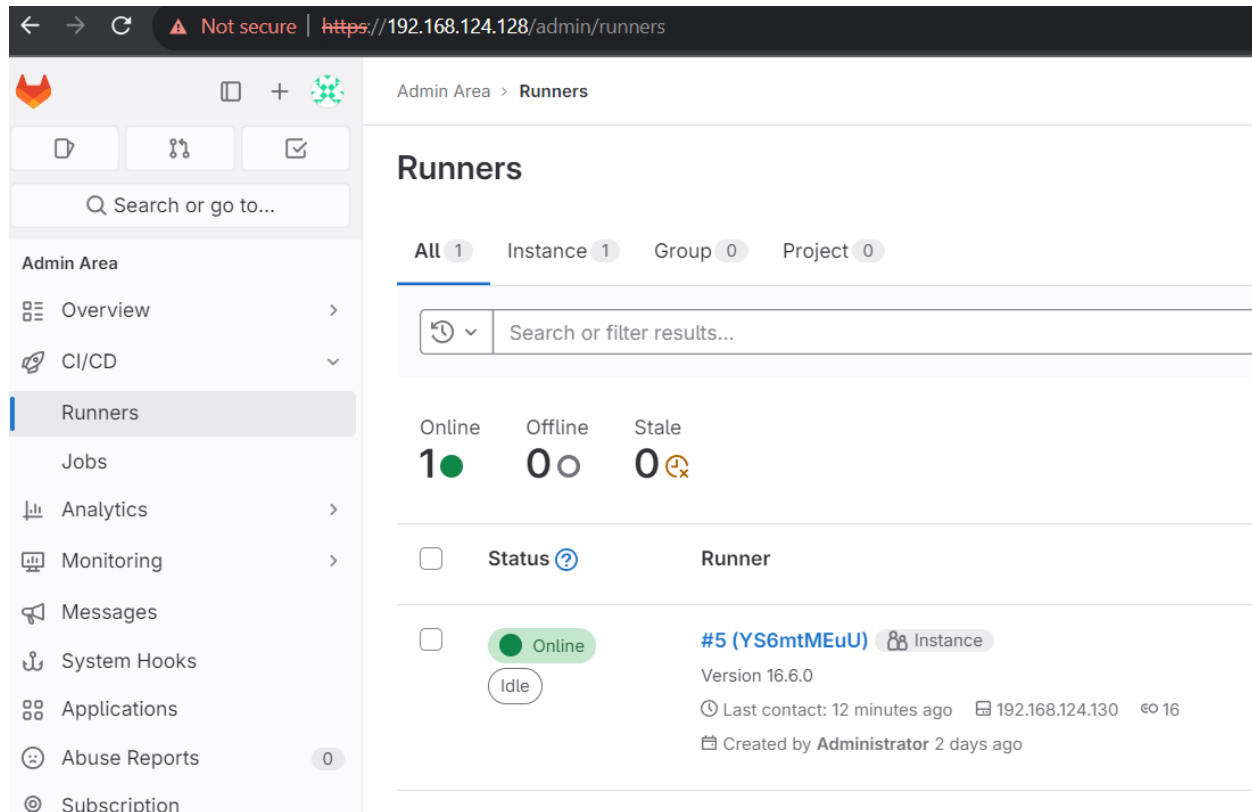
The screenshot shows a web browser window with the address bar displaying `https://192.168.124.130/log_in`. The page title is "scansuite". The login form includes a "Log In" heading, a "Username" label with an input field, a "Password" label with an input field, and a "Log In" button. A link "Don't have an account? Sign Up." is located next to the button. At the bottom, a footer states: "ScanSuite by Sergey Egorov. The source code and the website content belong to the author and may only be used for educational purposes."

Also open the DefectDojo via [https://YOUR\\_SCANSUITE\\_IP:8443](https://YOUR_SCANSUITE_IP:8443) The **username** here would be **admin** and the **password YUcfb8qUM6yCecD6C1PnS6**



The screenshot shows a web browser window with the address bar displaying `https://192.168.124.130:8443/login?next=`. The page title is "DEFECT DOJO". The login form includes a "Login" heading, a "Username\*" label with an input field, a "Password\*" label with an input field, a "Show Password" link, a "Login" button, and links for "I forgot my password" and "I forgot my username".

Now, get back to the GitLab web console and follow to the [https://YOUR\\_GITLAB\\_IP/admin/runners](https://YOUR_GITLAB_IP/admin/runners) page:



You should see one online / green runner. If you don't, try to reboot your scansuite (runner is there) host and check again.

Please ensure all above works before you start the labs. Otherwise approach me (contacts are on the last page) and we troubleshoot together.

In addition, please download the application **source codes vulnado.zip and django.nV.zip** These are just zipped repos for 2 well known vulnerable apps. You do not need to unzip them, just save them somewhere and upload the zip when requested.

## 2. Lab overview

Here is how our stack looks like. We have got a few open-source scanners, they can produce the results, exportable to the vulnerability management system, [DefectDojo](#), which is also free to use.

Although it not necessary, one may want to have some orchestration engine for the scanners, as the single point of interaction with some interfaces to it, like web, cli or callable via CI/CD pipeline. I have developed one and called it ScanSuite, you may use it exclusively for these labs because it nicely glues

the scanners with the respective systems and helps you to learn the tools without getting into the syntax details of every individual tool.



We won't be exporting issues to Jira this time, although this can be done from DefectDojo out of box.

So, let's warm up a bit.

### Lab 1 Vulnerable python application

As your developer peers may approach you and ask you to check the bugs in their code because “you are the security expert”, you can send them back to their salt mines OR you can run some checks and see what comes up.

First, we need to create a “Product” in terms of DefectDojo. That would be a “folder”, which will contain one or many “subfolders” called “Engagements”. Let's open the “Products” Tab in ScanSuite and create 3 products one by one: **DjangoNV**, **Vulnado** and **Infra**

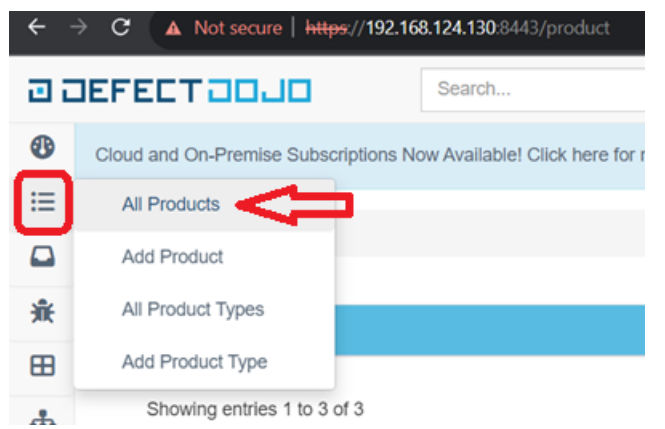
## Create new product

Product Name

Submit

ScanSuite by [Sergey Egorov](#). The source code and the website content belong to the author and may only be used for educational purposes.

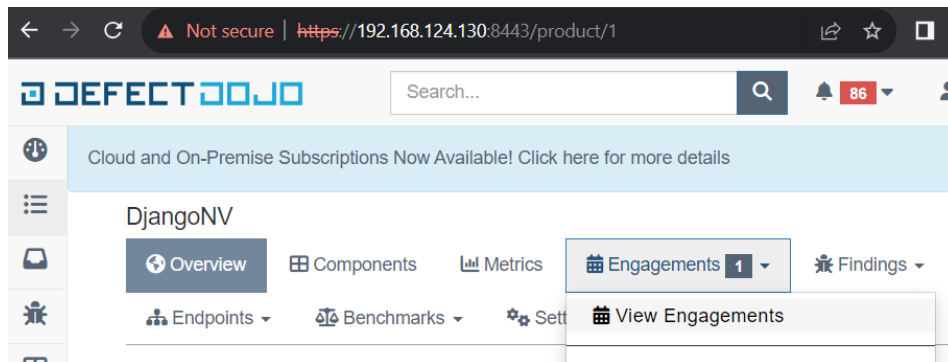
You won't get a result in ScanSuite because it just shoots the command to DefectDojo, so let's open it and check the created Products:



The screenshot shows the DefectDojo web interface. The 'All Products' menu item is highlighted with a red box and a red arrow. Below the menu, the 'Product List' table is visible, showing three products: DjangoNV, Infra, and Vulnado. The table has columns for Product, Tags, Criticality, Metadata, Eng., Active (Verified) Findings, Vulnerable Hosts / Endpoints, and Cor.

| Product  | Tags | Criticality | Metadata | Eng. | Active (Verified) Findings | Vulnerable Hosts / Endpoints | Cor |
|----------|------|-------------|----------|------|----------------------------|------------------------------|-----|
| DjangoNV |      |             |          |      | 0                          | 0                            |     |
| Infra    |      |             |          |      | 0                          | 0                            |     |
| Vulnado  |      |             |          |      | 0                          | 0                            |     |

Click on the DjangoNV one and open “Engagements - View Engagements” in it:

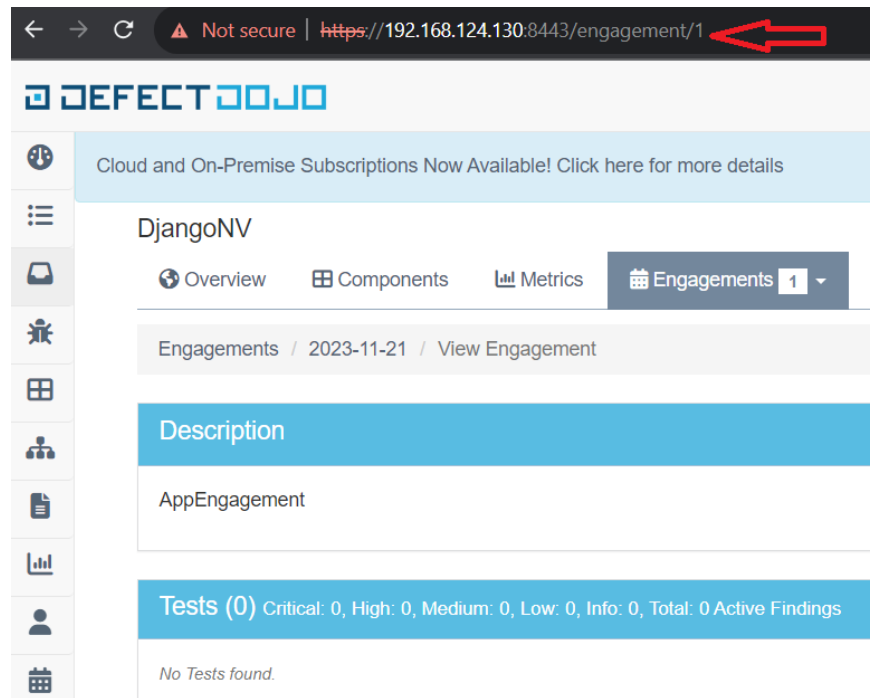


You will see the created Engagement with the current date as the name so just select it:

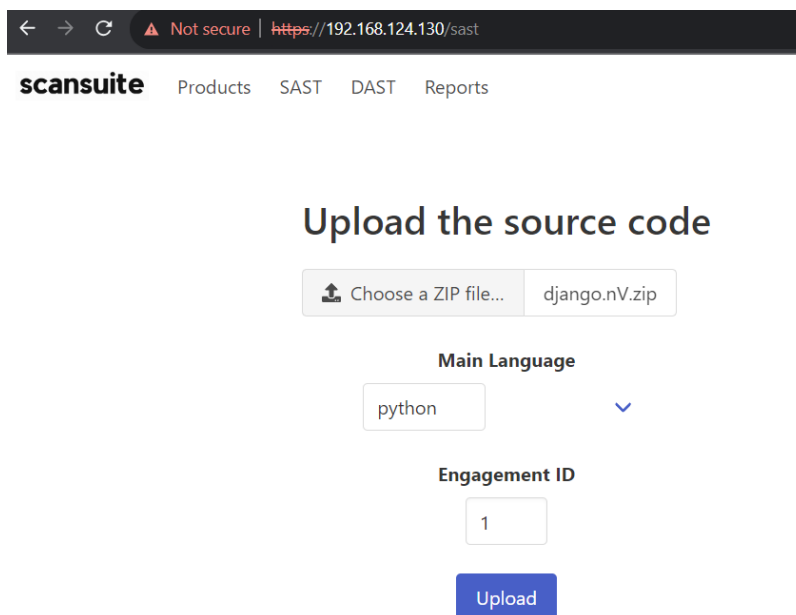
| Paused Engagements (1)   |            |             |      |                                     |         |       |
|--|------------|-------------|------|-------------------------------------|---------|-------|
| Showing entries 1 to 1 of 1  |            |             |      |                                     |         |       |
| <div>Column visibility</div> <div>Copy</div> <div>Excel</div> <div>CSV</div> <div>PDF</div> <div>Print</div> |            |             |      |                                     |         |       |
|  | Name       | Type        | Lead | Date                                | Length  | Tests |
| ⋮  | 2023-11-21 | Interactive |      | 21st November<br>- 21st<br>November | 2 years | 0     |
| Showing entries 1 to 1 of 1  |            |             |      |                                     |         |       |

Finally, we are inside the engagement where the results of our Python scans will come to. Check the number in the URL (most likely it will be 1 or 2). That's an Engagement ID you will need to provide during the scans:





Let's get back to ScanSuite and upload the django.nV.zip file. You should have it downloaded already during the prep stage. Go to the SAST page, select the file and specify Engagement ID. In my case it's 1:



Click "Upload" and select the SAST tab again. You should see the scan was started:

## Upload the source code



Main Language

python



Engagement ID

123

| Date                          | Scan Type | Target                    | Engagement Id | Status   | Scan Id  |
|-------------------------------|-----------|---------------------------|---------------|----------|----------|
| 2023-11-21<br>22:43:22.455639 | SAST      | django.nV.zip             | 1             | Started  | zyomdjwu |
| 2023-11-21<br>20:26:53.547972 | DAST      | https://ginandjuice.shop/ | 2             | Finished | lhdbnxvo |
| 2023-11-21<br>20:26:25.668894 | SAST      | django.nV.zip             | 2             | Finished | wabxjmy  |

Next you may want to run a DAST scan for the ready website so see some typical server misconfigs or exposed files to the public, so select the DAST tab and execute the scan.

You can target the <https://ginandjuice.shop/> site, which was purposely created for vulnerability scanning or you can scan the website, which belongs to me <https://myappsec.eu/> providing the same Engagement Id as you did for SAST:



## Scan the website

Engagement ID

1

Coming back to SAST tab we see another scan has been started:

python



## Engagement ID

123

Upload

| Date                          | Scan Type | Target                    | Engagement Id | Status   | Scan Id   |
|-------------------------------|-----------|---------------------------|---------------|----------|-----------|
| 2023-11-22<br>21:07:11.694027 | DAST      | https://ginandjuice.shop/ | 1             | Started  | wedqjfkjg |
| 2023-11-21<br>22:43:22.455639 | SAST      | django.nV.zip             | 1             | Finished | zyomdjwu  |

DAST scan is quite time consuming by its nature so for the purposes of this lab the quick / compliance ZAP scan is configured to run.

Let's get back to the DefectDojo and refresh the page. Scan results are coming:

The screenshot shows the DefectDojo web interface. The browser address bar indicates the URL is <https://192.168.124.130:8443/engagement/1>. The page title is "DjangoNV". The navigation bar includes "Overview", "Components", "Metrics", "Engagements 1", and "Findings 68". The main content area shows the engagement details for "DjangoNV". The "Description" section is titled "AppEngagement". The "Tests (4)" section shows a summary of findings: Critical: 2, High: 12, Medium: 53, Low: 1, Info: 0, Total: 68 Active Findings. Below this is a table showing the results of four scans:

| Title / Type                   | Date                          | Total Lead | Findings | Active (Verified) | Mitigated | Duplicates | Notes | Reimports |
|--------------------------------|-------------------------------|------------|----------|-------------------|-----------|------------|-------|-----------|
| Gitleaks Scan                  | Nov. 21, 2023 - Nov. 21, 2023 | 1          | 1 (1)    | 0                 | 0         | 0          |       |           |
| kics Scan (GitLab SAST Report) | Nov. 21, 2023 - Nov. 21, 2023 | 0          | 0 (0)    | 0                 | 0         | 0          |       |           |
| Semgrep JSON Report            | Nov. 21, 2023 - Nov. 21, 2023 | 50         | 50 (50)  | 0                 | 0         | 0          |       |           |
| Trivy Scan                     | Nov. 21, 2023 - Nov. 21, 2023 | 17         | 17 (17)  | 0                 | 0         | 0          |       |           |

Overall, that should be 4 SAST tests. The DAST one will come up later (10-20 mins depending on the website you scan and the Internet speed). Let's check the results by clicking the scan names or the corresponding active findings:

1. Secrets scan – [GitLeaks](#) Scan

The scan identified secrets in the code and git history, which is a big headache for many orgs, especially the cloud-based ones. One leaked API key to the cloud resources may lead to a full compromise of the cloud project or complete organization. So, keep an eye on these things and scan the code for these issues as often as needed.

In our case, however, this is truly a false positive, so we follow to the next scan.

2. Infrastructure as Code Scan (IACS) [kics](#) Scan

A neat way to identify vulnerabilities in your infra even before it gets deployed. Checks Terraform, Ansible, Docker & Kubernetes etc configs which we don't have in this repo, moving to the next one.

3. SAST – [Semgrep](#)

Here we see some issues in the code itself like XSS, SQLi, CSRF etc. Not a big deal compared to the number of issues the application has.

4. Software Composition Analysis (SCA) or just the dependency checks – [Trivy](#) Scan

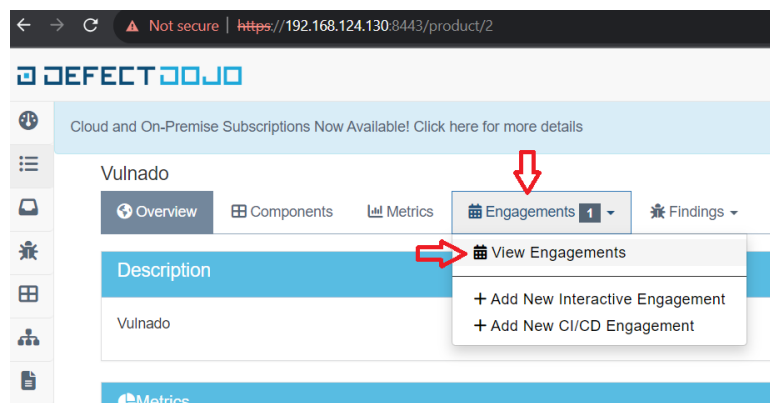
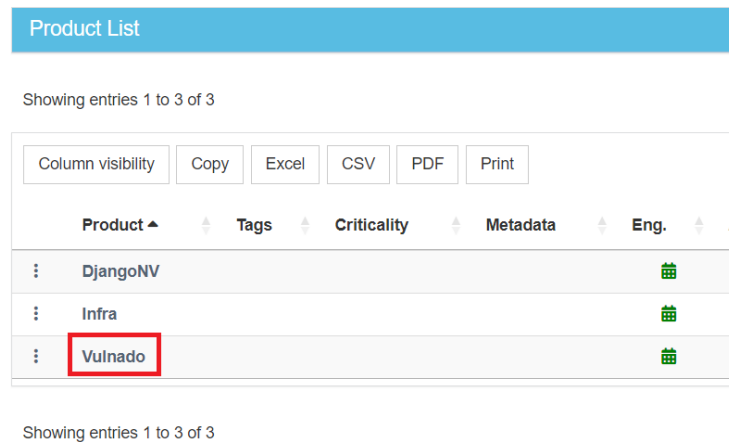
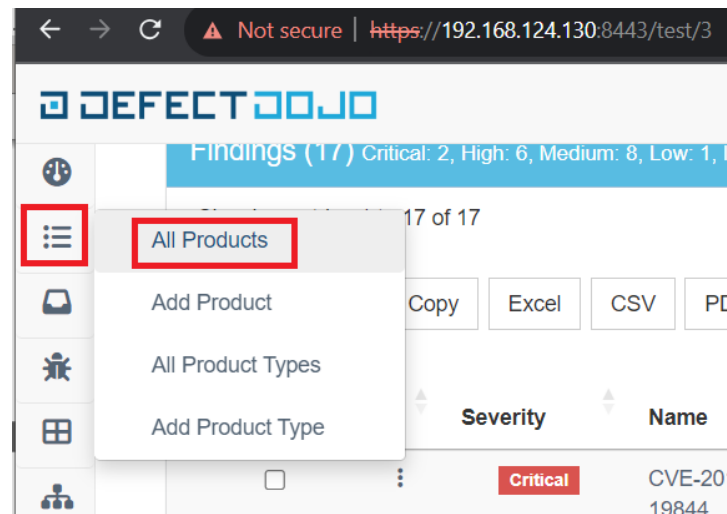
The dependency scan shows us known CVEs for the libs and imports used by the project.

Hope you have enjoyed this warm up, let's move to the second lab.

## Lab 2. CI/CD pipelines

Your dev team uses the CI/CD rails to test and deploy the code which is great because you can integrate the scans right into the pipeline. Let's see how to do that.

First, we switch to [Vulnado](#) product in DefectDojo to locate its Engagement Id:




Click of the Engagement with the current or the most recent date:

| Paused Engagements (1)   |             |      |
|--|-------------|------|
| Showing entries 1 to 1 of 1  |             |      |
| <div>Column visibility</div> <div>Copy</div> <div>Excel</div> <div>CSV</div> <div>PDF</div> <div>Print</div> |             |      |
| Name   | Type        | Lead |
| <div>⋮</div> 2023-11-21  | Interactive |      |
| Showing entries 1 to 1 of 1  |             |      |

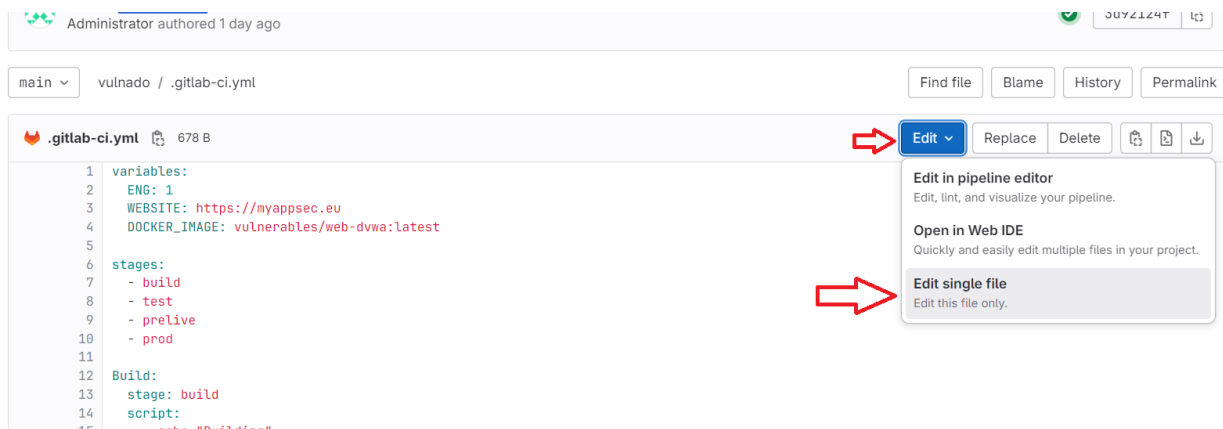
Check the number in the URL and note it down:

The screenshot shows a web browser window with the URL <https://192.168.124.130:8443/engagement/2>. The page is titled "Vulnado" and displays engagement details for the date "2023-11-21". The interface includes a sidebar with navigation icons and a main content area with sections for "Description" (AppEngagement) and "Tests (0)". A red arrow points to the URL in the address bar.

Open the [https://YOUR\\_GITLAB\\_IP/root/vulnado](https://YOUR_GITLAB_IP/root/vulnado) page where you will find the vulnerable Java application code. Click on .gitlab-ci.yml file:

|  |                       |
|--|-----------------------|
| reverse_shell  | init commit           |
| src  | init commit           |
| .gitignore   | init commit           |
|  .gitlab-ci.yml | Update .gitlab-ci.yml |
| Dockerfile   | init commit           |
| LICENSE  | init commit           |
| README.md  | init commit           |


And then Edit – Edit single file:



Administrator authored 1 day ago

main vulnado / .gitlab-ci.yml

Find file Blame History Permalink

 .gitlab-ci.yml 678 B

```

1 variables:
2   ENG: 1
3   WEBSITE: https://myappsec.eu
4   DOCKER_IMAGE: vulnerables/web-dvwa:latest
5
6 stages:
7   - build
8   - test
9   - prelive
10  - prod
11
12 Build:
13   stage: build
14   script:
15     - echo "Building"

```

Edit Replace Delete

**Edit in pipeline editor**  
Edit, lint, and visualize your pipeline.

**Open in Web IDE**  
Quickly and easily edit multiple files in your project.

**Edit single file**  
Edit this file only.

This is a yaml config file which describes the pipeline to be executed. GitLab is monitoring code commits to the git main/master branch and autoruns the pipeline described in this file. Contents of this file are easily readable. Let's change the ENG variable value to the Engagement Id we just got in DefectDojo, will be 2 in my case:

## Edit file

Write Preview changes

main .gitlab-ci.yml Apply a template

```

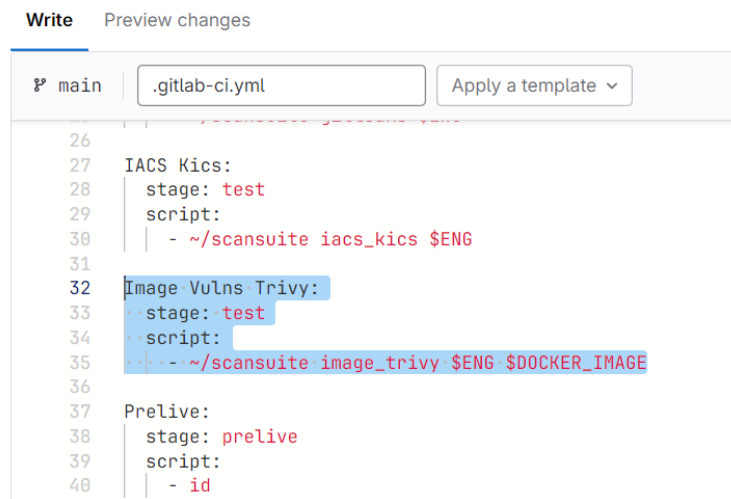
1 variables:
2   ENG: 2
3   WEBSITE: https://myappsec.eu
4   DOCKER_IMAGE: vulnerables/web-dvwa:latest
5
6 stages:
7   - build

```

And add another test to the config right after the IACS Kics scan. That would be a docker image scan, just Ctrl+C – Ctrl+V the text below and paste into your web page:

Image Vulns Trivy:

```
stage: test
script:
  - ~/scansuite image_trivy $ENG $DOCKER_IMAGE
```



Then scroll down and commit the changes:

```

38   stage: prelive
39   script:
40     - id
41
42 DAST ZAP:
43   stage: prelive
44   script:
45     - ~/scansuite zap_base $ENG $WEBSITE
46
47 Production:
48   stage: prod

```

Commit message

Update .gitlab-ci.yml

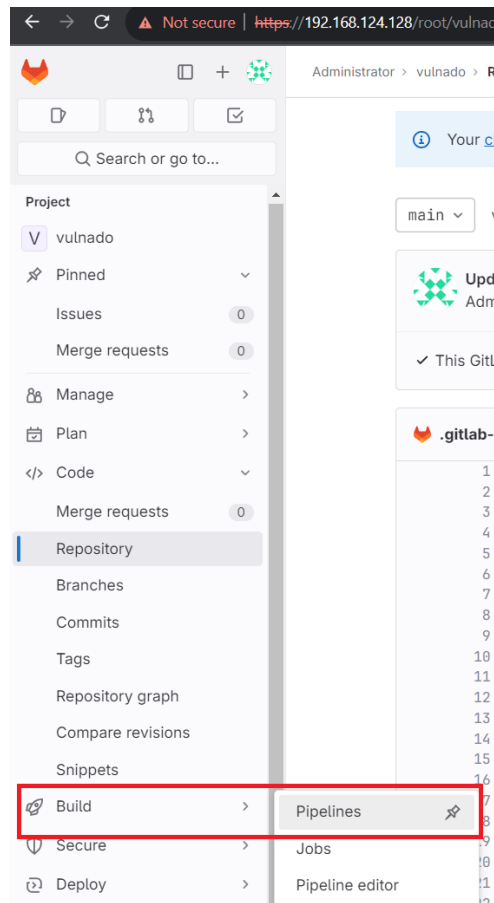
Target Branch

main

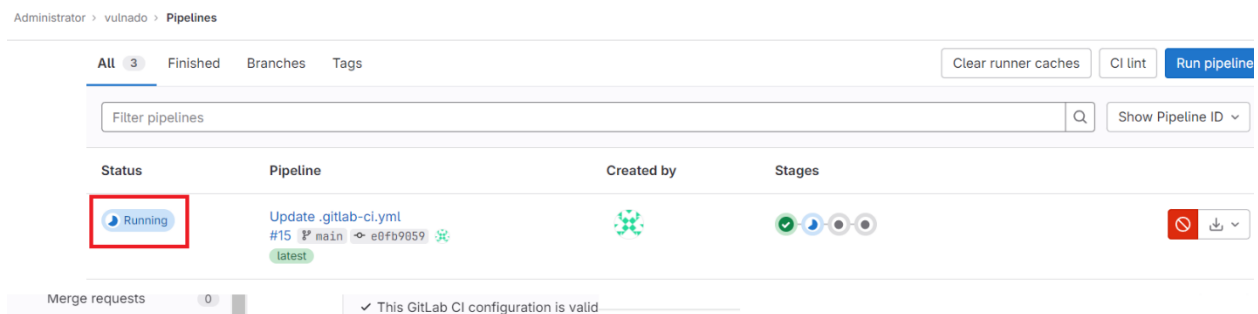
Commit changes Cancel

Now moving to the Build – Pipelines:





The pipeline is executing, click on the Running logo to see the details of it:



Here we have stages and tasks, executed by our Runner, installed on the scansuite VM host. If tasks are not picked by the runner, try to reboot the **scansuite** machine, hopefully it comes back.

## Update .gitlab-ci.yml

✓ Passed Administrator created pipeline for commit `e0fb9859` finished just now

For `main`

latest 60 8 Jobs ⌚ 2 minutes 49 seconds, queued for 3 seconds

Pipeline Needs Jobs 8 Tests 0

| build          | test                | prelive    | prod         |
|----------------|---------------------|------------|--------------|
| ✓ Build        | ✓ IACS Kics         | ✓ DAST ZAP | ✓ Production |
| ✓ SAST Semgrep | ✓ Image Vulns Trivy | ✓ Prelive  |              |
| ✓ Secrets Scan |                     |            |              |

Feel free to click on any of the tasks where you will see the terminal output of the executed command.

```

30 java 120 10
31 dockerfile 5 3
32 htal 1 3
33 js 179 2
34 terraform 101 2
35 yaml 28 2
36
37
38
39
40
41
42 Scan Summary
43
44 Some files were skipped or only partially analyzed.
45 Scan was limited to files tracked by git.
46 Scan skipped: 1 files matching .semgrepignore patterns
47 For a full list of skipped files, run semgrep with the --verbose flag.
48 Ran 508 rules on 46 files: 24 findings.
49 A new version of Semgrep is available. See https://semgrep.dev/docs/upgrading
50 Uploading Results to DefectDojo ...
51 % Total % Received % Xferd Average Speed Time Time Time Current
52 100 59253 100 1219 100 58834 1451 69117 --:--:-- --:--:-- --:--:-- 78539
53 [{"minimum_severity":"Info","active":true,"verified":true,"scan_type":"Semgrep JSON Report","endpoint_to_add":null,"file":null,"engagement":2,"auto_create_context":false,"deduplication_on_engagement":false,"lead":null,"close_old_findings":false,"close_old_findings_product_scope":false,"push_to_jira":false,"api_scan_configuration":null,"create_finding_groups_for_all_findings":true,"test":6,"test_id":6,"engagement_id":2,"product_id":2,"product_type_id":1,"statistics":{"after":{"info":{"active":0,"verified":0,"duplicate":0,"false_p":0,"out_of_scope":0,"is_mitigated":0,"risk_accepted":0,"total":0},"low":{"active":0,"verified":0,"duplicate":0,"false_p":0,"out_of_scope":0,"is_mitigated":0,"risk_accepted":0,"total":0},"medium":{"active":16,"verified":16,"duplicate":0,"false_p":0,"out_of_scope":0,"is_mitigated":0,"risk_accepted":0,"total":16},"high":{"active":6,"verified":6,"duplicate":0,"false_p":0,"out_of_scope":0,"is_mitigated":0,"risk_accepted":0,"total":6},"critical":{"active":0,"verified":0,"duplicate":0,"false_p":0,"out_of_scope":0,"is_mitigated":0,"risk_accepted":0,"total":0},"total":{"active":22,"verified":22,"duplicate":0,"false_p":0,"out_of_scope":0,"is_mitigated":0,"risk_accepted":0,"total":22}}}]
54 Cleaning up project directory and file based variables

```

Timeout: 1h (from project) ⓘ

Runner: #5 (Y56mtMEuU)

Commit `e0fb9859` ⓘ

Update .gitlab-ci.yml

Pipeline #15 ✓ Passed for main ⓘ

build

Related jobs

- ✓ Build
- ✓ SAST Semgrep
- ✓ Secrets Scan

Let's check the results in DefectDojo. Here we have 5 scan results, both static and (eventually) dynamic:

## Vulnado

[Overview](#)
[Components](#)
[Metrics](#)
[Engagements 1](#)
[Findings 981](#)
[Endpoints 1](#)

Engagements / 2023-11-21 / View Engagement

## Description

AppEngagement

Tests (5) Critical: 40, High: 140, Medium: 251, Low: 509, Info: 41, Total: 981 Active Findings

Showing entries 1 to 5 of 5

| Title / Type                     | Date                          | Lead | Total Findings | Active (Verified) | Mitigat |
|----------------------------------|-------------------------------|------|----------------|-------------------|---------|
| ⋮ Gitleaks Scan                  | Nov. 22, 2023 - Nov. 22, 2023 |      | 2              | 2 (2)             | 0       |
| ⋮ kics Scan (GitLab SAST Report) | Nov. 22, 2023 - Nov. 22, 2023 |      | 72             | 72 (72)           | 0       |
| ⋮ Semgrep JSON Report            | Nov. 22, 2023 - Nov. 22, 2023 |      | 22             | 22 (22)           | 0       |
| ⋮ Trivy Scan                     | Nov. 22, 2023 - Nov. 22, 2023 |      | 870            | 870 (870)         | 0       |
| ⋮ ZAP Scan                       | Nov. 22, 2023 - Nov. 22, 2023 |      | 15             | 15 (15)           | 0       |

Showing entries 1 to 5 of 5

Check the results and reach me if you have any questions. This time results should be more actionable with much less false positives.

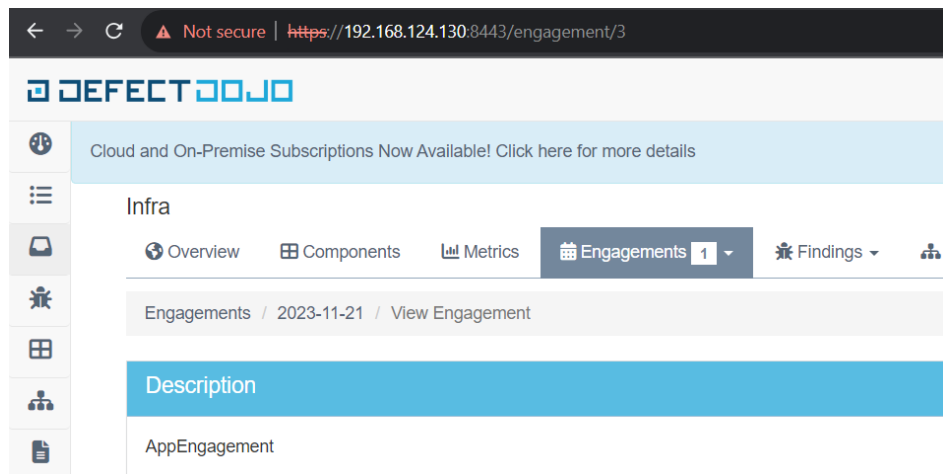
Worth noting, that Trivy Image scan brought 870+ findings as the examined Docker image “web-dvwa” is old and vulnerable, as it should be.

Let’s switch the gears and examine how we can check local systems for infrastructure vulnerabilities.

### Lab 3 Infrastructure scanning

During the previous labs we have used IACS (Infra as Code Scanning) and Docker image scanning tools which allowed us to gather some environmental vulnerabilities in which applications are supposed to be executed.

This is great, but we can do better. Let’s investigate the local Docker setup deficiencies and how to analyze them using [docker-bench-security](#) tool. First, goto DefectDojo and locate the Engagement ID of Infra product. You should be already familiar how to do that:



Now, login to the scansuite host over SSH (**login scanbot** with **password scanbot**) or via VMWare console:

```
PS C:\Users\v0id> ssh scanbot@192.168.124.130
scanbot@192.168.124.130's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-88-generic x86_64)
```

Execute the following command:

**sudo ~/scansuite docker\_bench 3**

Here **3** is my Engagement ID, yours might be different (but most likely will be the same)

```
scanbot@scansuite:~$ sudo ~/scansuite docker_bench 3
[sudo] password for scanbot:

----- ScanSuite v2.0 -----
-- Author: Sergey Egorov --

# -----
# Docker Bench for Security v1.6.0
#
# Docker, Inc. (c) 2015-2023
#
# Checks for dozens of common best-practices around deploying
# Based on the CIS Docker Benchmark 1.6.0.
# -----

Initializing 2023-11-25T20:49:13+00:00

Section A - Check results

[INFO] 1 - Host Configuration
[INFO] 1.1 - Linux Hosts Specific Configuration
[WARN] 1.1.1 - Ensure a separate partition for containers ha
```

The scan was executed, and report uploaded to DefectDojo.

Before analyzing the results let's run a couple of more local infra scans. One would be compliance check of the OS hardening to [CIS Benchmarks](#). We will be using popular [Open SCAP](#) analyzer for it:

**sudo ~/scansuite oscap\_ubuntu 3 22**

Again, 3 is an Engagement ID

```
scanbot@scansuite:~$ sudo ./scansuite oscap_ubuntu 3 22

----- ScanSuite v2.0 -----
-- Author: Sergey Egorov --

Title   Package "prelink" Must not be Installed
Rule    xccdf_org.ssgproject.content_rule_package_prelink_removed
Result  pass

Title   Install AIDE
Rule    xccdf_org.ssgproject.content_rule_package_aide_installed
Result  fail

Title   Build and Test AIDE Database
Rule    xccdf_org.ssgproject.content_rule_aide_build_database
Result  fail
```

The scan will take 3-5 minutes to complete, some checks will take a while. At the end the results will be uploaded to DefectDojo.

We can also check local packages for vulnerabilities using OVAL checks of the same Open SCAP tool. sudo is not required for these checks:

**~/scansuite oscap\_ubuntu\_vuln**

It will download the latest OVAL definitions from Ubuntu and check the local setup.

```
scanbot@scansuite:~$ ~/scansuite oscap_ubuntu_vuln

----- ScanSuite v2.0 -----
-- Author: Sergey Egorov --

--2023-11-25 19:02:58-- https://security-metadata.canonical.com/oval/com.ubuntu.jammy.usn.oval.xml.bz2
Resolving security-metadata.canonical.com (security-metadata.canonical.com)... 185.125.190.21, 185.125.190.29, 185.125.190.20, ...
Connecting to security-metadata.canonical.com (security-metadata.canonical.com)|185.125.190.21|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 263942 (258K) [application/x-bzip2]
Saving to: 'com.ubuntu.jammy.usn.oval.xml.bz2'

com.ubuntu.jammy.usn.oval.xml 100%[=====] 257.76K --.-KB/s in 0.07s

2023-11-25 19:02:58 (3.60 MB/s) - 'com.ubuntu.jammy.usn.oval.xml.bz2' saved [263942/263942]

Definition oval:com.ubuntu.jammy:def:981000000: false
Definition oval:com.ubuntu.jammy:def:971000000: false
Definition oval:com.ubuntu.jammy:def:961000000: false
Definition oval:com.ubuntu.jammy:def:951000000: false
```

Results will not be uploaded to DefectDojo due to the absence of parser and will be saved to **oscap\_scansuite\_vuln.html** in the same folder. Transfer the file to your local machine and open it:

```
PS C:\Users\v0id> scp scanbot@192.168.124.130:oscap_scansuite_vuln.html .
scanbot@192.168.124.130's password:
oscap_scansuite_vuln.html 100% 653KB 47.7MB/s 00:00
```

Some packages need to be patched:

| ID                                    | Result | Class     | Reference ID  | Title   |
|---------------------------------------|--------|-----------|---|---|
| oval:com.ubuntu.jammy:def:65051000000 | true   | patch     | [USN-6505-1], [CVE-2023-44487]  | USN-6505-1 – nhttp2 vulnerability                 |
| oval:com.ubuntu.jammy:def:64961000000 | true   | patch     | [USN-6496-1], [CVE-2023-25775], [CVE-2023-31085], [CVE-2023-45871]  | USN-6496-1 – Linux kernel vulnerabilities         |
| oval:com.ubuntu.jammy:def:100         | true   | inventory |   | Check that Ubuntu 22.04 LTS (jammy) is installed. |
| oval:com.ubuntu.jammy:def:981000000   | false  | patch     | [LSN-0098-1], [CVE-2023-3776], [CVE-2023-3609], [CVE-2023-21400], [CVE-2023-4004], [CVE-2023-3777], [CVE-2023-40283], [CVE-2023-3090], [CVE-2023-3995], [CVE-2023-3567] | LSN-0098-1 – Kernel Live Patch Security Notice    |


Switch to the DefectDojo and see how we can do better in Docker and OS hardening:

| Tests (2) Critical: 0, High: 37, Medium: 90, Low: 22, Info: 7, Total: 156 Active Findings |                                  |      |                |                   |           |            |       |           |
|---|----------------------------------|------|----------------|-------------------|-----------|------------|-------|-----------|
| Showing entries 1 to 2 of 2   |                                  |      |                |                   |           |            |       |           |
| Title / Type  | Date                             | Lead | Total Findings | Active (Verified) | Mitigated | Duplicates | Notes | Reimports |
| docker-bench-<br>: security Scan  | Nov. 25, 2023 -<br>Nov. 25, 2023 |      | 54             | 54 (54)           | 0         | 0          |       | 0         |
| Openscap<br>: Vulnerability Scan  | Nov. 25, 2023 -<br>Nov. 25, 2023 |      | 102            | 102 (102)         | 0         | 0          |       | 0         |
| Showing entries 1 to 2 of 2   |                                  |      |                |                   |           |            |       |           |

In this lab we did not investigate the remote infrastructure scanning, which should be a mandatory part of any vulnerability management program. This is due to most tools in this area are commercial, and OpenVAS is pretty resource consuming to set for these labs.

Feel free to set it up in your spare time (using for example [this guide](#)) and upload the results using the **Reports** tab of ScanSuite interface so you can aggregate such findings in DefectDojo too:

## Upload the report file

 Choose a ZIP file...

No file uploaded

### Report Type

OpenVAS CSV



### Engagement ID

3

Upload

### 3. Wrap-up

This is it, hope you enjoyed this intro to vulnerability scanning. Obviously, there are more scanners one can use to get better results for analyzing specific language or framework. You can check the complete list of scanners I am using along with other [ScanSuite functionality here](#).

Try to clone some repos to **scansuite** host and run these scanners using the syntax suggested in Readme. See if you can identify new vulnerabilities in open source projects.

Please reach out to me if you have any questions about the lab walkthrough, the DevSecOps topic or would like to share the worthwhile scanners I am missing. Also, if you want to collaborate on ScanSuite or use it for your commercial projects. Will be happy to learn from your experience too.

Otherwise enjoy your days at the conference and good luck with battling vulnerabilities!

My contacts:

Sergey Egorov

[cepexo@gmail.com](mailto:cepexo@gmail.com)

Threema STR295ER