

Vulnerability Management with DefectDojo

DevOps & Security – Course Project
Ahmed Nouralla

Content

- **Introduction**
 - Vulnerability Management
 - DefectDojo Architecture, Features, and Essential Concepts
- **Methods**
 - Server Deployment
 - Implemented use-case
- **Results**
- **Discussion**
- **References**

Intro: Vulnerability Management

Vulnerability management is the cyclical practice of identifying, evaluating (classifying/prioritizing), addressing (mitigating/remediating) and reporting software vulnerabilities.

Practical DevSecOps Pipeline:

1. Run one or more security scanners against the application source code
 - E.g., SAST, DAST, SCA, and Secret Detection tools.
2. Aggregate and evaluate reports from such tools
 - May need some cleaning/reformatting of findings.
3. Upload results to a central **vulnerability management** tool
 - E.g., Automatically in CI/CD using a script
4. Inspect results for further analysis and remediation

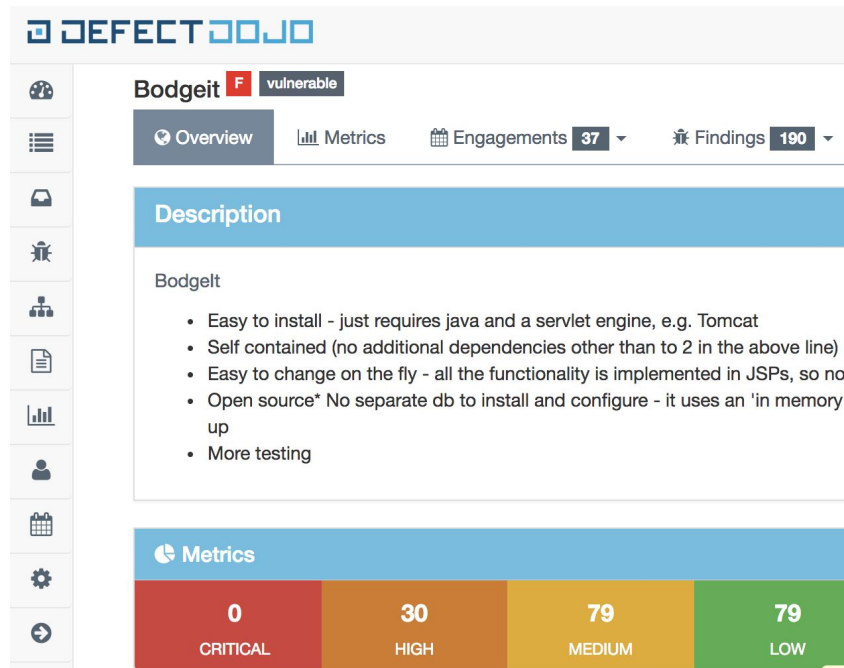


DefectDojo Features

A popular vulnerability management solution written in Python.

It provides:

- A centralized dashboard for inspecting security findings
- [Integrations](#) with 150+ scanners and security tools
 - SAST, DAST, SCA, and Infrastructure Scanners
- Automated deduplication of findings to reduce noise.
- Automatic scan imports in CI/CD through the API.
- Bi-directional integration with JIRA.

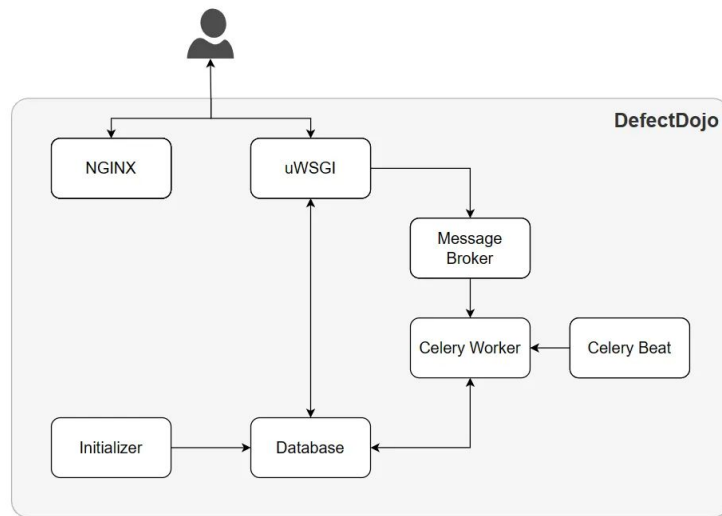


DefectDojo Interface

DefectDojo Architecture

DefectDojo server utilizes multiple components:

- **Nginx**: webserver delivering all static content
- **uWSGI**: application server serving all dynamic content.
- **Redis**: message broker for asynchronous execution of tasks.
- **Celery Worker**: performs tasks like finding deduplication or JIRA synchronization asynchronously in the background
- **Celery Beat**: to run periodic scheduled tasks.
- **PostgreSQL**: database storing all the application data.
- **Initializer**: setups/maintains the database and syncs/runs migrations after version upgrades. It shuts itself down after all tasks are performed.

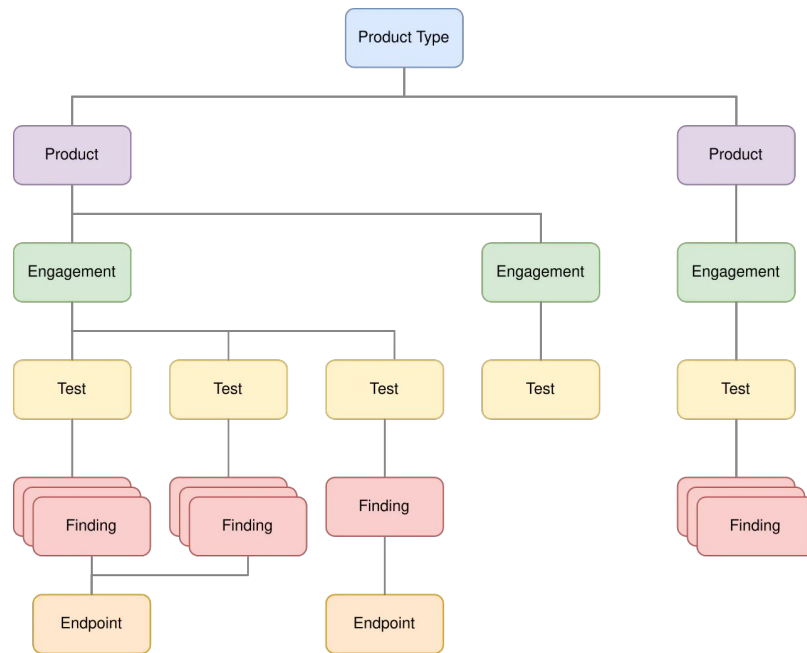


DefectDojo Components

Essential Concepts

Data classes in DefectDojo:

- **Product Type:** high-level categorization of products
- **Product:** individual unit to be tested (e.g., a certain version)
- **Engagement:** moments in time when testing takes place.
 - **Interactive engagement:** manual scan imports from the UI
 - **CI/CD engagements:** automatic result imports from pipelines.
- **Tests:** grouping of activities conducted to attempt to discover flaws in a product.
- **Findings:** specific flaws discovered while testing.
- **Endpoints:** hosts affected by a certain finding.



DefectDojo Product Hierarchy

Methods: Server Deployment

- DefectDojo recommends deployment through [Docker Compose](#)
- For enterprises, a managed [SaaS platform](#) is also offered
- [Local](#) or [K8s](#) installation is possible but not officially supported.

Instances (1/1) [Info](#)

Find Instance by attribute or tag (case-sensitive)

Instance state = running [X](#) [Clear filters](#)

<input checked="" type="checkbox"/>	Name ↗	Instance ID	Instance state ▼	Instance type ▼	Public IPv4 ... ▼
<input checked="" type="checkbox"/>	defectdojo	i-03f7a158148410542	Running 🔍 🔍	t2.medium	34.226.190.90

domain	current ip
defectdojo	34.227.52.215 update ip

```
ubuntu@ip-172-31-23-252:~/django-DefectDojo$ docker comp
[+] Running 7/7
✓ Container django-defectdojo-postgres-1 Started
✓ Container django-defectdojo-redis-1 Started
✓ Container django-defectdojo-celeryworker-1 Started
✓ Container django-defectdojo-initializer-1 Started
✓ Container django-defectdojo-celerybeat-1 Started
✓ Container django-defectdojo-nginx-1 Started
✓ Container django-defectdojo-uwsgi-1 Started
```

defectdojo.duckdns.org/dashboard

DEFECTDOJO

Cloud and On-Premise Subscriptions Now Available! [Click here for more details](#)

Dashboard

Products

Engagements

Findings

Components

Endpoints

0

Active Engagements

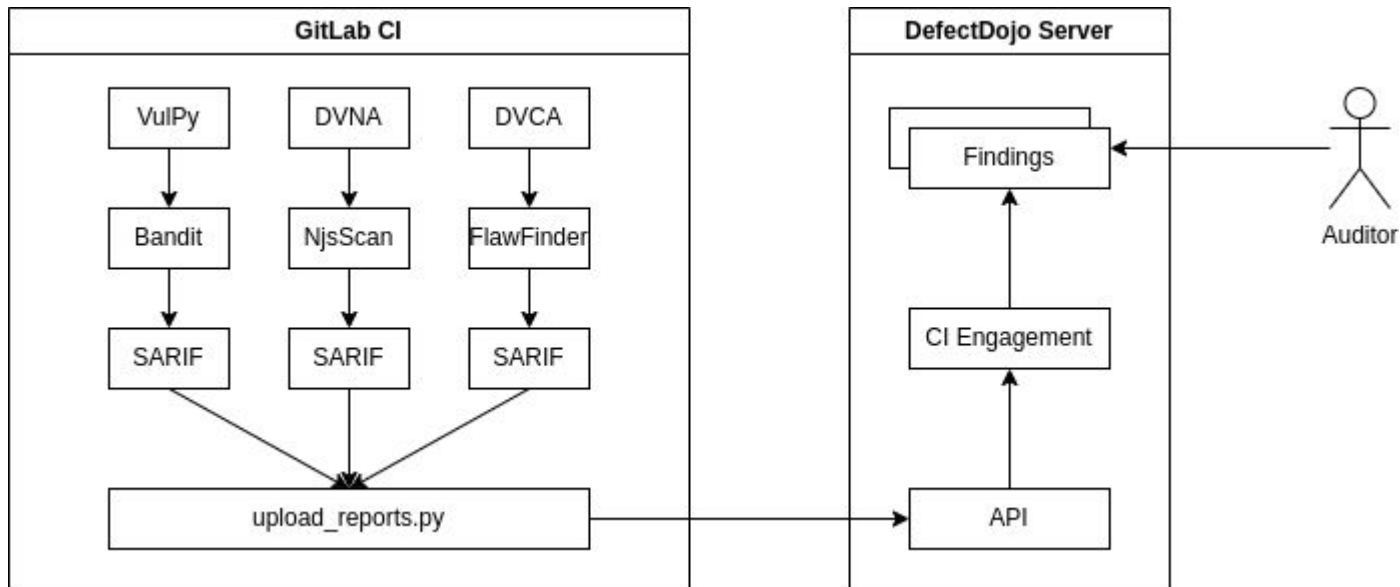
[View Engagement Details](#)

Last S

[View Finding Details](#)

Use Case

- Cloned three sample vulnerable-by-design projects: [VulPy](#) (Python), [DVNA](#) (NodeJS), and [DVCA](#) (C).
- Ran three SAST tools ([Bandit](#), [NjsScan](#), and [FlawFinder](#)) to scan the projects in CI and produce [SARIF](#) reports.
- Wrote a Python script to automatically import scan reports (CI engagements) to DefectDojo via its API.



Technical Details: Pipeline and Script

🐱 .gitlab-ci.yml > [] stages

```
gitlab-ci - GitLab CI Configuration file (ci.json)
1  stages:
2    - security-scan
3    - upload-reports
4
5  security-scan:
6    stage: security-scan
7    image: python:3-alpine
8    before_script:
9      - pip3 install bandit njsscan flawfinder
10   script:
11     - bandit -r vulpy/ -f sarif --exit-zero -o bandit_report.sarif
12     - njsscan --sarif dvna/ -o njsscan_report.sarif || true
13     - flawfinder --sarif dvca/ > flawfinder_report.sarif
14   artifacts:
15     paths:
16       - bandit_report.sarif
17       - njsscan_report.sarif
18       - flawfinder_report.sarif
19
20  upload-reports:
21    stage: upload-reports
22    image: python:3-alpine
23    needs: ["security-scan"]
24    before_script:
25      - pip3 install requests
26    script:
27      - python3 upload-reports.py bandit_report.sarif
28      - python3 upload-reports.py njsscan_report.sarif
29      - python3 upload-reports.py flawfinder_report.sarif
```

🐍 upload-reports.py > ...

```
1  import requests
2  from os import environ
3  from sys import argv
4
5  headers = {
6    'Authorization': f'Token {environ["DD_API_TOKEN"]}'
7  }
8
9  url = 'http://defectdojo.duckdns.org/api/v2/import-scan/'
10
11  data = {
12    'active': True,
13    'verified': True,
14    'scan_type': 'SARIF',
15    'auto_create_context': True,
16    'product_name': 'demo',
17    'engagement_name': 'CI Engagement',
18    'test_title': 'CI Scan',
19  }
20
21  files = {
22    'file': open(argv[1], 'r')
23  }
24
25  res = requests.post(url, headers=headers, data=data, files=files)
26
27  if res.status_code == 201:
28    print(f"Report {argv[1]} uploaded successfully!")
29  else:
30    print(f"Report {argv[1]} failed to upload.\n{res.content}")
31  ~
```

Results: Automatic Scans and Imported Findings

```
Ahmed Nouralla / defectdojo-demo / Jobs / #9283594672

security-scan
✓ Passed Started just now by Ahmed Nouralla

Search visible log output

1 Running with gitlab-runner 17.7.0-pre.103.g896916a8 (896916a8)
2   on green-5.saas-linux-small-amd64.runners-manager.gitlab.com/default xS6Vzpvo, system ID: s_6ble4f06cfcd
3   ✓ Preparing the "docker+machine" executor
4   Using Docker executor with image python:3-alpine ...
5   Pulling docker image python:3-alpine ...
6   Using docker image sha256:e0566e0d46dcfa5a8b8ada82bba53d8f84d0629d8460d1d41197e4c66bc24a51 for python:3-alpine
7   256:323a717dc4a018fec21e3f1aac738ee10bb485de4e7593ce242b36ee48d6b352 ...
8   ✓ Preparing environment
9   Running on runner-xs6vzpvo-project-67594403-concurrent-0 via runner-xs6vzpvo-s-l-s-amd64-1740859798-ed910347.
10  ✓ Getting source from Git repository
11  Fetching changes with git depth set to 20...
12  Initialized empty Git repository in /builds/Sh3B0/defectdojo-demo/.git/
13  Created fresh repository.
14  Checking out ed92b0d7 as detached HEAD (ref is main)...
15  Skipping Git submodules setup
16  $ git remote set-url origin "${CI_REPOSITORY_URL}"
17  ✓ Executing "step_script" stage of the job script
18  ✓ Uploading artifacts for successful job
19  Uploading artifacts...
20  bandit_report.sarif: found 1 matching artifact files and directories
21  njsscan_report.sarif: found 1 matching artifact files and directories
22  flawfinder_report.sarif: found 1 matching artifact files and directories
23  WARNING: Upload request redirected location=https://gitlab.com/api/v4/jobs/9283594672/artifacts
24  artifact_type=archive new-url=https://gitlab.com
25  WARNING: Retrying... context=artifacts-uploader error=request redirected
26  Uploading artifacts as "archive" to coordinator... 201 Created id=9283594672 responseStatus=201 Created token=
27  ✓ Cleaning up project directory and file based variables
28  Job succeeded
```

```
Ahmed Nouralla / defectdojo-demo / Jobs / #9283609102

upload-reports
✓ Passed Started 24 seconds ago by Ahmed Nouralla

1 Running with gitlab-runner 17.7.0-pre.103.g896916a8 (896916a8)
2   on green-6.saas-linux-small-amd64.runners-manager.gitlab.com/default YKxHNyexq, system ID: s_a201ab37b78a
3   ✓ Preparing the "docker+machine" executor
4   ✓ Preparing environment
5   ✓ Getting source from Git repository
6   ✓ Downloading artifacts
7   ✓ Executing "step_script" stage of the job script
8   Using docker image sha256:e0566e0d46dcfa5a8b8ada82bba53d8f84d0629d8460d1d41197e4c66bc24a51 for python:3-alpine
9   $ pip3 install requests
10  Collecting requests
11    Downloading requests-2.32.3-py3-none-any.whl.metadata (4.6 kB)
12  Collecting charset-normalizer<4,>=2 (from requests)
13    Downloading charset-normalizer-3.4.1-cp313-cp313-muslinux_1_2_x86_64.whl.metadata (35 kB)
14  Collecting idna<4,>=2.5 (from requests)
15    Downloading idna-3.10-py3-none-any.whl.metadata (10 kB)
16  Collecting urllib3<3,>=1.21.1 (from requests)
17    Downloading urllib3-2.3.0-py3-none-any.whl.metadata (6.5 kB)
18  Collecting certifi-2017.4.17 (from requests)
19    Downloading certifi-2025.1.31-py3-none-any.whl.metadata (2.5 kB)
20  Downloading requests-2.32.3-py3-none-any.whl (64 kB)
21  Downloading certifi-2025.1.31-py3-none-any.whl (166 kB)
22  Downloading charset-normalizer-3.4.1-cp313-cp313-muslinux_1_2_x86_64.whl (145 kB)
23  Downloading idna-3.10-py3-none-any.whl (70 kB)
24  Downloading urllib3-2.3.0-py3-none-any.whl (128 kB)
25  Installing collected packages: urllib3, idna, charset-normalizer, certifi, requests
26  Successfully installed certifi-2025.1.31 charset-normalizer-3.4.1 idna-3.10 requests-2.32.3 urllib3-2.3.0
27  WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system
28  instead: https://pip.pypa.io/warnings/venv. Use the --root-user-action option if you know what you are doing and
29  [notice] A new release of pip is available: 24.3.1 -> 25.0.1
30  [notice] To update, run: pip install --upgrade pip
31  $ python3 upload-reports.py bandit_report.sarif
32  Report bandit_report.sarif uploaded successfully!
33  $ python3 upload-reports.py njsscan_report.sarif
34  Report njsscan_report.sarif uploaded successfully!
35  $ python3 upload-reports.py flawfinder_report.sarif
36  Report flawfinder_report.sarif uploaded successfully!
37  ✓ Cleaning up project directory and file based variables
38  Job succeeded
```

Findings are shown in DefectDojo UI



Inspecting a Finding

demo

Overview Components Metrics Engagements 1 Findings 402 Hosts / Endpoints 0 / 0 Benchmarks Settings

CI Engagement / CI Scan (Bandit Scan (SARIF)) / A Flask App Appears to Be Run With Debug=True, Which Expose... / View Finding

A Flask App Appears to Be Run With Debug=True, Which Exposes the Werkzeug Debugger and Allows the Execution of Arbitrary Code. cwe-94 security Last Reviewed today by Admin User (admin), Last Status Update today, Created today, Last Mentioned in (Re)import: today as created



ID	Severity	SLA	Status	Type	Date discovered	Age	Reporter	CWE	Vulnerability id	Found by	Vuln ID from tool
32	High	30	Active, Verified	Static	March 1, 2025	0 days	Admin User (admin)	94		Bandit Scan (SARIF)	B201

Location	Line Number
vulpy/good/vulpy.py	53

Similar Findings (401)



Import History (1)



Description



Result message: A Flask app appears to be run with debug=True, which exposes the Werkzeug debugger and allows the execution of arbitrary code.

Snippet:

```
app.run(debug=True, host='127.0.1.1', port=5001, extra_files='csp.txt')
```

Rule name: flask_debug_true

Discussion

Workflow improvements:

- Integrate email notification for new findings
- Merge and deduplicate similar findings with different contexts.
- Create different user accounts and groups for different types of users
- Optimize tool settings to avoid reporting false positives

Server Monitoring:

- One can expose metrics from Django used by DefectDojo services by setting the variable ``DD_DJANGO_METRICS_ENABLED``.
- These metrics can later be consumed by other tools (e.g., Prometheus)
- An open-source [exporter](#) for collecting metrics is also available.

References

- <https://www.microsoft.com/en-us/security/business/security-101/what-is-vulnerability-management>
- https://docs.defectdojo.com/en/open_source/installation/architecture/s
- https://docs.defectdojo.com/en/open_source/installation/running-in-production/
- https://docs.defectdojo.com/en/working_with_findings/organizing_engagements_tests/product_hierarchy/
- <https://docs.oasis-open.org/sarif/sarif/v2.0/csprd01/sarif-v2.0-csprd01.html>
- https://docs.gitlab.com/ci/quick_start/
- <https://github.com/PyCQA/bandit>
- <https://github.com/ajinabraham/njsscan>
- <https://github.com/david-a-wheeler/flawfinder>
- <https://github.com/fportantier/vulpy>
- <https://github.com/appsecco/dvna>
- [https://github.com/hardik05/Damn Vulnerable C Program](https://github.com/hardik05/Damn_Vulnerable_C_Program)