

Vulnerability Management with DefectDojo

DevOps and Security - Project Report

Ahmed Nouralla - a.shaaban@innopolis.university

Vulnerability Management with DefectDojo

Introduction

Methods

System Architecture

Supported Features

Deployment Options

Product Hierarchy

Use-case

Results

Discussion

References

Introduction

Vulnerability management is the "cyclical practice of identifying, classifying, prioritizing, remediating, and mitigating software vulnerabilities".

A typical DevSecOps CI pipeline may involve the following steps:

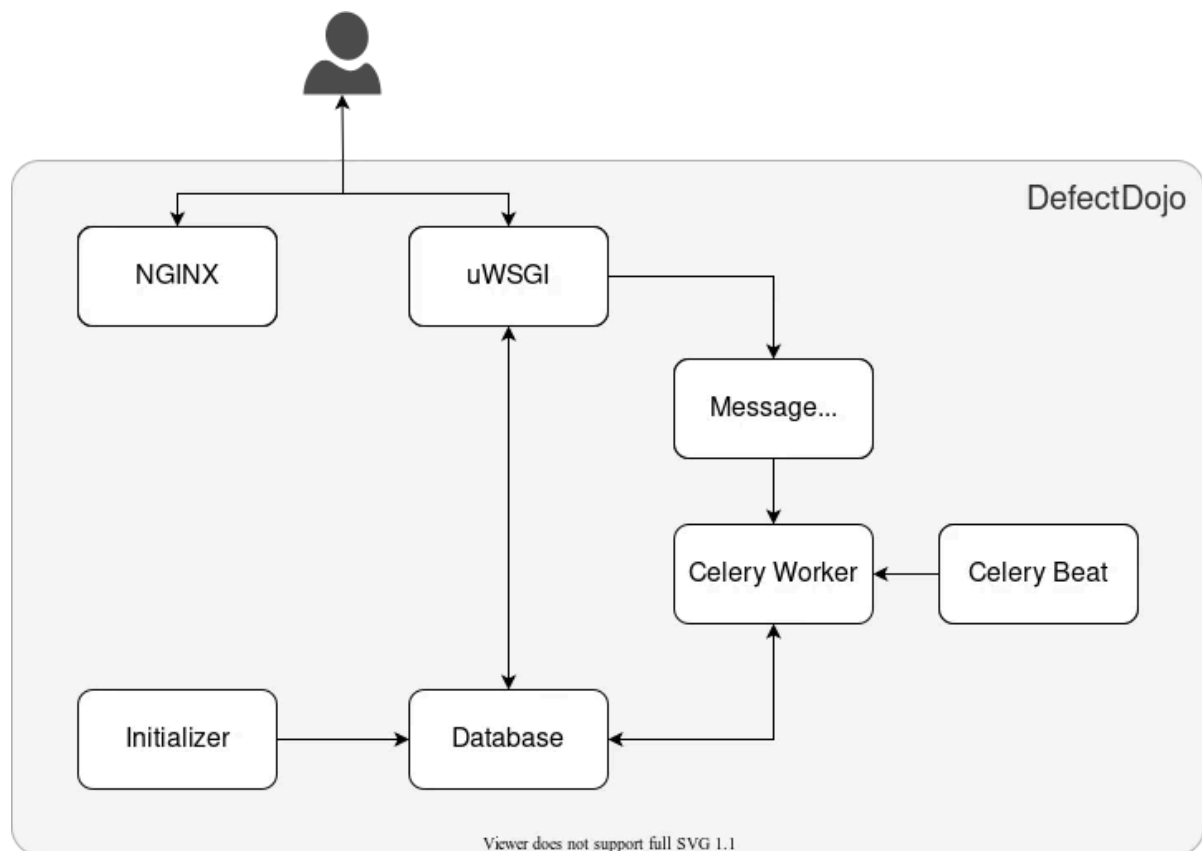
1. Running one or more security scanners against the application source code, this may include:
 - **Static Application Security Testing:** analyzing code for bad practices without running it.
 - **Dynamic Application Security Testing:** interactive security scanning by simulating attacks.
 - **Software Composition Analysis:** identification (e.g., generating SBOM) of 3rd party components used by a software to help reduce risks associated with using them. Tools for container/dependency scanning lie within this category.
 - **Secret Detection:** identifying leaked credentials or keys in source code.
2. Aggregate reports from such tools, cleaning/re-formatting them if needed.
3. Upload results to a central vulnerability management tool for an auditor to inspect results for further analysis and mitigation.

Popular open-source tools providing vulnerability management dashboards include DefectDojo, Faraday, ArcheySec, and Wazuh.

Methods

This work implements a simple DevSecOps pipeline scenario with GitLab, DefectDojo, and Three SAST tools (Bandit, NjsScan, Flawfinder).

System Architecture



- [Nginx](#): the webserver delivering all static content, e.g. images, JavaScript files or CSS files.
- [uWSGI](#) is the application server that runs the DefectDojo platform, written in Python/Django, to serve all dynamic content.
- [Redis](#): the application server sends tasks to a [Message Broker](#) for asynchronous execution.
- [Celery](#) Worker: tasks like deduplication or the JIRA synchronization are performed asynchronously in the background by the Celery Worker.
- [Celery Beat](#): In order to identify and notify users about things like upcoming engagements, DefectDojo runs scheduled tasks using Celery Beat.
- [PostgreSQL](#): database storing all the application data of DefectDojo.
- **Initializer**: setups/maintains the database and syncs/runs migrations after version upgrades. It shuts itself down after all tasks are performed.

Supported Features

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

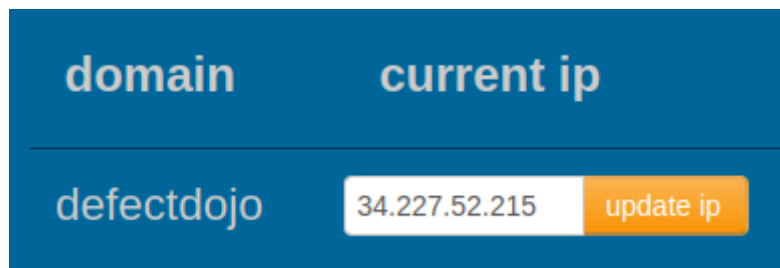
Deployment Options

- The recommended option to deploy DefectDojo is thorough docker compose or their managed SaaS platform.
- Running DefectDojo in Docker in a cloud VM following the [docs](#)

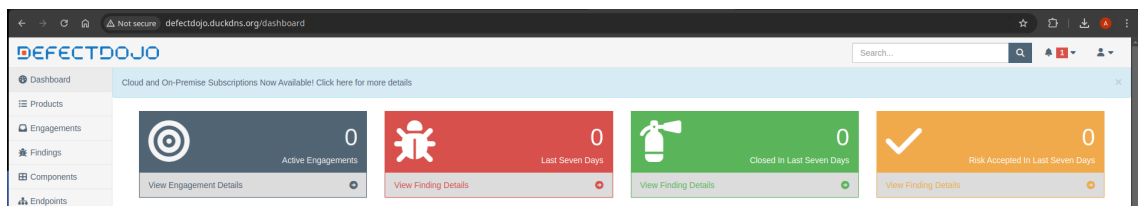
```
git clone https://github.com/DefectDojo/django-DefectDojo
cd django-DefectDojo
docker compose build && docker compose up -d
docker compose logs -f initializer | grep "Admin password:"
```

```
ubuntu@ip-172-31-23-252:~/django-DefectDojo$ docker compose up -d
[+] 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
```

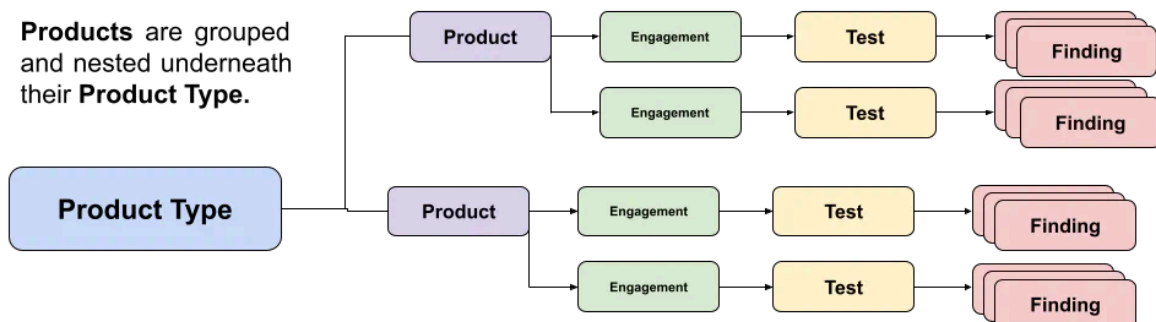
- Added a dynamic DNS record for the server to be accessible at <http://defectdojo.duckdns.org>



- Logged in with `admin` and the password from initializer logs.

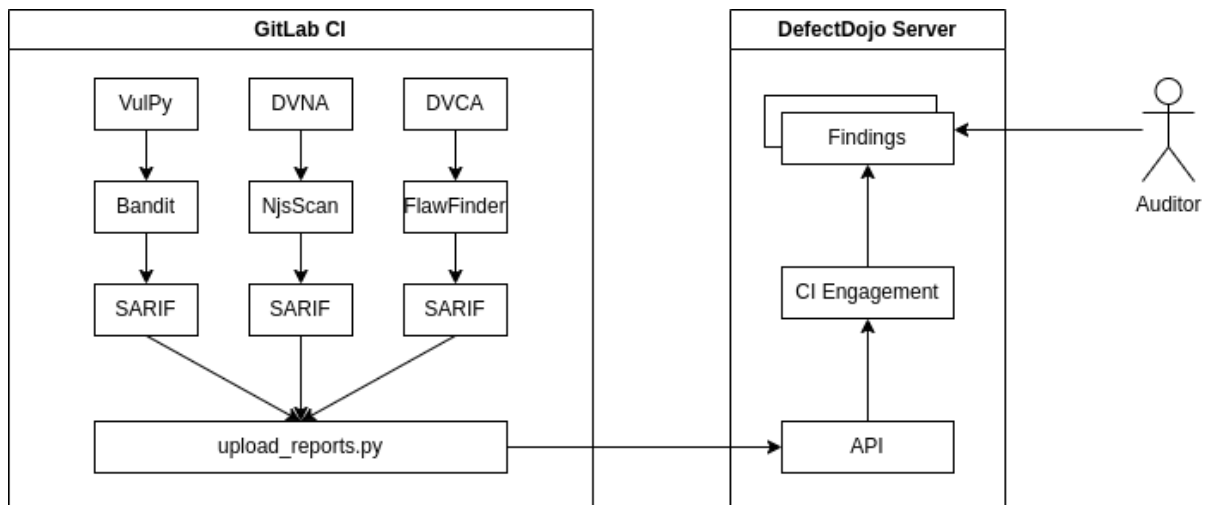


Product Hierarchy

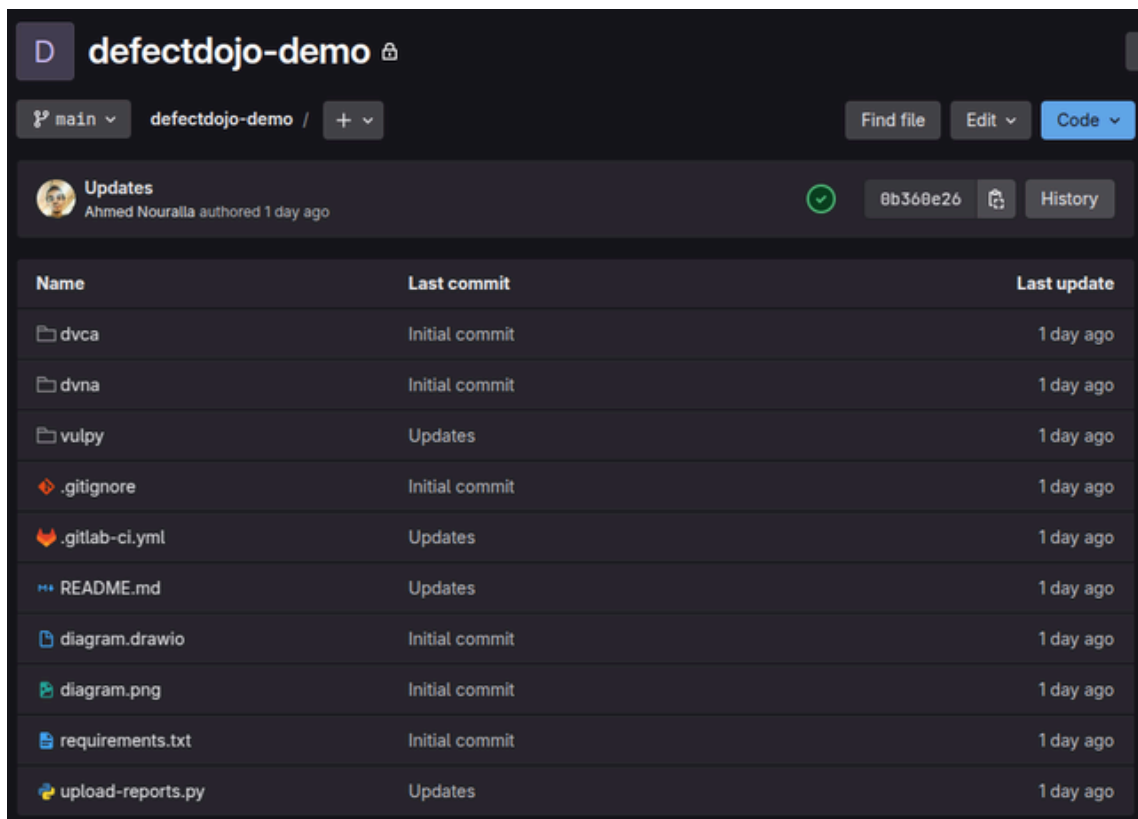


- Product Type:** high-level categorization of products
- Product:** individual unit to be tested (e.g., a certain version/build)
 - Engagement:** represents moments in time when testing takes place
 - Interactive engagement: manual scan imports from the UI
 - CI/CD engagements: automatic result imports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

Use-case



1. Cloned three sample vulnerable-by-design projects: [VulPy](#), [DVNA](#), and [DVCP](#).



- **Vulpy:** Flask (Python) WebApp with SQLite database.
 - **DVNA:** Express (NodeJS) WebApp with MySQL database.
 - **DVCA:** C Program with utilities to read and process images.
2. Added a `.gitlab-ci.yml` that utilizes three SAST tools ([Bandit](#), [NjsScan](#), [FlawFinder](#)) to scan the respective projects in CI.

```
stages:
  - security-scan
  - upload-reports

security-scan:
  stage: security-scan
  image: python:3-alpine
  before_script:
```

```

- pip3 install bandit njsscan flawfinder
script:
- bandit -r vulpy/ -f sarif --exit-zero -o bandit_report.sarif
- njsscan --sarif dvna/ -o njsscan_report.sarif || true
- flawfinder --sarif dvca/ > flawfinder_report.sarif
...

```

3. Scan reports are collected as artifacts in [SARIF](#) format

```

...
artifacts:
  paths:
    - bandit_report.sarif
    - njsscan_report.sarif
    - flawfinder_report.sarif
...

```

4. A Python script is used to connect to DefectDojo API to upload scan results.

- API token was obtained from server and configured in GitLab as a secret variable `DD_API_TOKEN` to be used by the Python script.

```

...
upload-reports:
  stage: upload-reports
  image: python:3-alpine
  needs: ["security-scan"]
  before_script:
    - pip3 install requests
  script:
    - python3 upload-reports.py bandit_report.sarif
    - python3 upload-reports.py njsscan_report.sarif
    - python3 upload-reports.py flawfinder_report.sarif

```

Results

- CI Run

The screenshot shows a GitLab CI/CD pipeline run for a job named 'security-scan'. The pipeline is in a 'Passed' state, started by user 'Ahmed Nouralla'. The main log area shows the execution steps: running on a green-5.saas-linux-small-amd64 runner, preparing the Docker executor, pulling the python:3-alpine image, and then uploading artifacts. The artifacts section lists three files: bandit_report.sarif, njsscan_report.sarif, and flawfinder_report.sarif, each with a matching artifact file and directory. The pipeline summary on the right indicates a duration of 59 seconds, finished 1 minute ago, and a timeout of 1h. The commit is 'ed92b8d7' and the pipeline is '1694847058'.

```

1 Running with gitlab-runner 17.7.0-pre.183.g896916a8 (896916a8)
2   on green-5.saas-linux-small-amd64.runners-manager.gitlab.com/default xs6Vzpvo, system ID: s_6b1e4f6fcfd
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:c856e0d6d6cfa5a8b8ada82bba53d8f84d0629d8460d1d41197e4c66bc24a51 for python:3-alpine with digest python:sha
7   256:323a717dc4a010fee21e3f1aac738ee10bb485de4e7593ce242b36ee48d0b352 ...
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 ed92b8d7 as detached HEAD (ref is main)...
15  Skipping Git submodule 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?artifact_format=zip&
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... 281 Created id=9283594672 responseStatus=281 Created token=glcblt-66
27  Cleaning up project directory and file based variables
28  Job succeeded

```

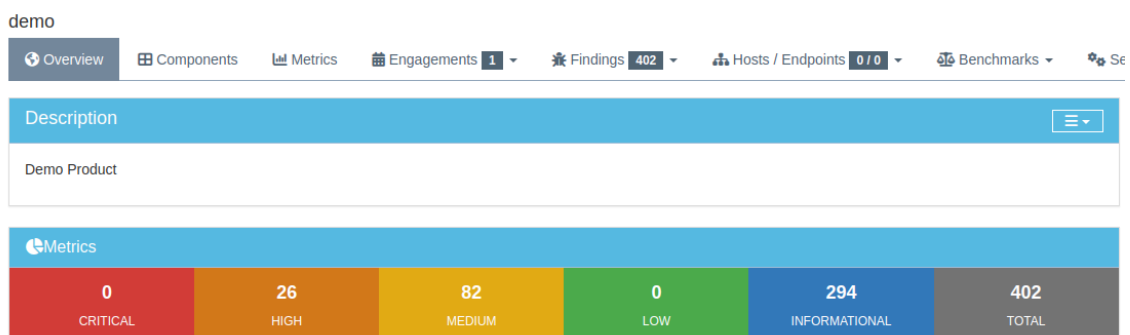
```
Ahmed Nouralla / defectdojo-demo / Jobs / #9283609102

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 YKxHNYeq, system ID: s_a201ab37b78a
> 3 Preparing the "docker+machine" executor
> 7 Preparing environment
> 9 Getting source from Git repository
> 16 Downloading artifacts
✓ 19 Executing "step_script" stage of the job script
20 Using docker image sha256:e0566e0d46dcfa5a8b8ada82bba53d8f84d0629d8460d1d41197e4c66bc24a51 for python:3-alpine
21 $ pip3 install requests
22 Collecting requests
23   Downloading requests-2.32.3-py3-none-any.whl.metadata (4.6 kB)
24 Collecting charset-normalizer<4,>=2 (from requests)
25   Downloading charset_normalizer-3.4.1-cp313-cp313-musllinux_1_2_x86_64.whl.metadata (35 kB)
26 Collecting idna<4,>=2.5 (from requests)
27   Downloading idna-3.10-py3-none-any.whl.metadata (10 kB)
28 Collecting urllib3<3,>=1.21.1 (from requests)
29   Downloading urllib3-2.3.0-py3-none-any.whl.metadata (6.5 kB)
30 Collecting certifi>=2017.4.17 (from requests)
31   Downloading certifi-2025.1.31-py3-none-any.whl.metadata (2.5 kB)
32 Downloading requests-2.32.3-py3-none-any.whl (64 kB)
33 Downloading certifi-2025.1.31-py3-none-any.whl (166 kB)
34 Downloading charset_normalizer-3.4.1-cp313-cp313-musllinux_1_2_x86_64.whl (145 kB)
35 Downloading idna-3.10-py3-none-any.whl (70 kB)
36 Downloading urllib3-2.3.0-py3-none-any.whl (128 kB)
37 Installing collected packages: urllib3, idna, charset-normalizer, certifi, requests
38 Successfully installed certifi-2025.1.31 charset-normalizer-3.4.1 idna-3.10 requests-2.32.3 urllib3-2.3.0
39 WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the sys
instead: https://pip.pypa.io/warnings/venv. Use the --root-user-action option if you know what you are doing an
40 [notice] A new release of pip is available: 24.3.1 -> 25.0.1
41 [notice] To update, run: pip install --upgrade pip
42 $ python3 upload-reports.py bandit_report.sarif
43 Report bandit_report.sarif uploaded successfully!
44 $ python3 upload-reports.py njsscan_report.sarif
45 Report njsscan_report.sarif uploaded successfully!
46 $ python3 upload-reports.py flawfinder_report.sarif
47 Report flawfinder_report.sarif uploaded successfully!
✓ 48 Cleaning up project directory and file based variables
49 Job succeeded
```

- Findings were imported to DefectDojo.



- Tests

Tests (3) Critical: 0, High: 13, Medium: 41, Low: 0, Info: 147, Total: 201 Active Findings									
Showing entries 1 to 3 of 3									
Title / Type	Date	Lead	Total Findings	Active (Verified)	Mitigated	Duplicates	Notes	Reimports	
CI Scan (Bandit Scan (SARIF))	March 1, 2025 - March 1, 2025	49	49 (49)	0	0	0		0	
CI Scan (Flawfinder Scan (SARIF))	March 1, 2025 - March 1, 2025	134	134 (134)	0	0	0		0	
CI Scan (nodejsscan Scan (SARIF))	March 1, 2025 - March 1, 2025	18	18 (18)	0	0	0		0	
Showing entries 1 to 3 of 3									

- Finding view

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.

Sec-35

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://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://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://docs.oasis-open.org/sarif/sarif/v2.0/csprd01/sarif-v2.0-csprd01.html>