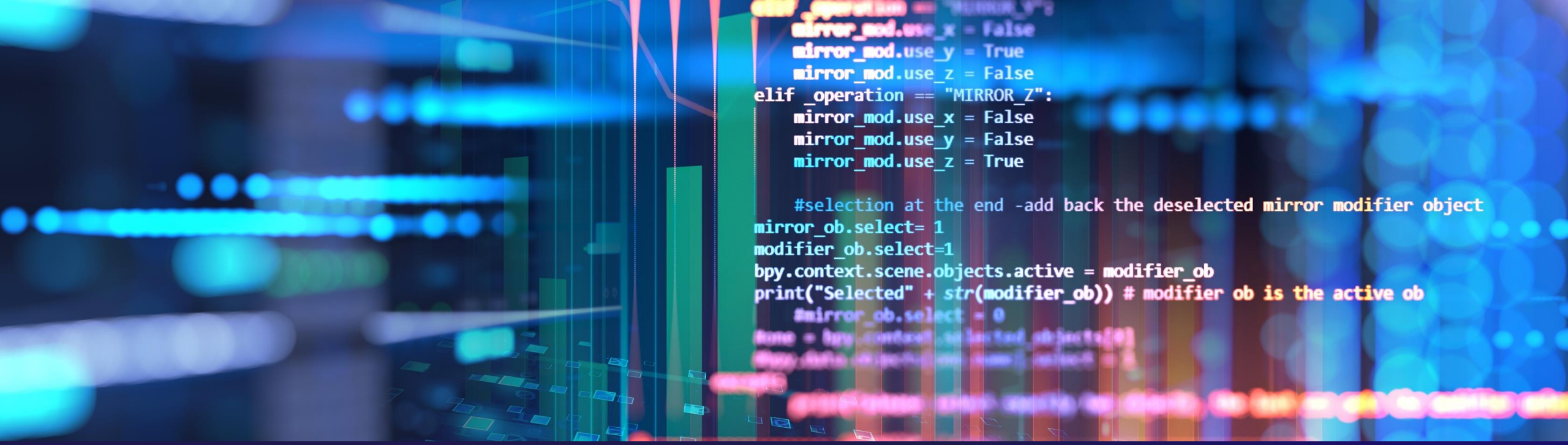


> TROOPERS 2024

Open Source Vulnerability Management

Stefan Fleckenstein

MAIBORNWOLFF

Stefan Fleckenstein

MaibornWolff GmbH

Head of Cybersecurity

stefan.fleckenstein@maibornwolff.de

+49 151 544 22 035

- 25 years mostly in software development projects
- 6 years mostly in cybersecurity
- 2 years maintainer of OWASP DefectDojo
- Founder and maintainer of SecObserve

Agenda

- Motivation
- Security testing in SW development
- How does it work?
- Does it work?

Motivation

Log4Shell (CVE-2021-44228)

Log4j is one of the most popular Java libraries

An attacker can load and execute malicious code via JNDI

Wide range exploitation

MOVEit Transfer (CVE-2023-34362)

Managed file transfer for sensitive data

SQL injection to get access to database

Ransomware attacks on BBC, BA, Nova Scotia and others

4 airports in Columbia and Peru (2022)

Publicly available S3 bucket with 3 TB of employee PII and other sensitive data

Discovered by security researchers

Microsoft Container Registry (2024)

A Microsoft employee accidentally published credentials to a public git repository, granting privileged access to an internal Azure Container Registry

Security testing in SW development

Vulnerability management is an important element of security testing

Security code reviews

› For every merge request or constantly with pair programming

› Cumbersome and large chance to miss problems

Vulnerability management

› With every run of the CI/CD pipelines or regularly

› Tool supported coverage of different perspectives

Security functional tests

› Repeatable unit-, integration-, system- and loadtests

› Verify that security measures are implemented correctly

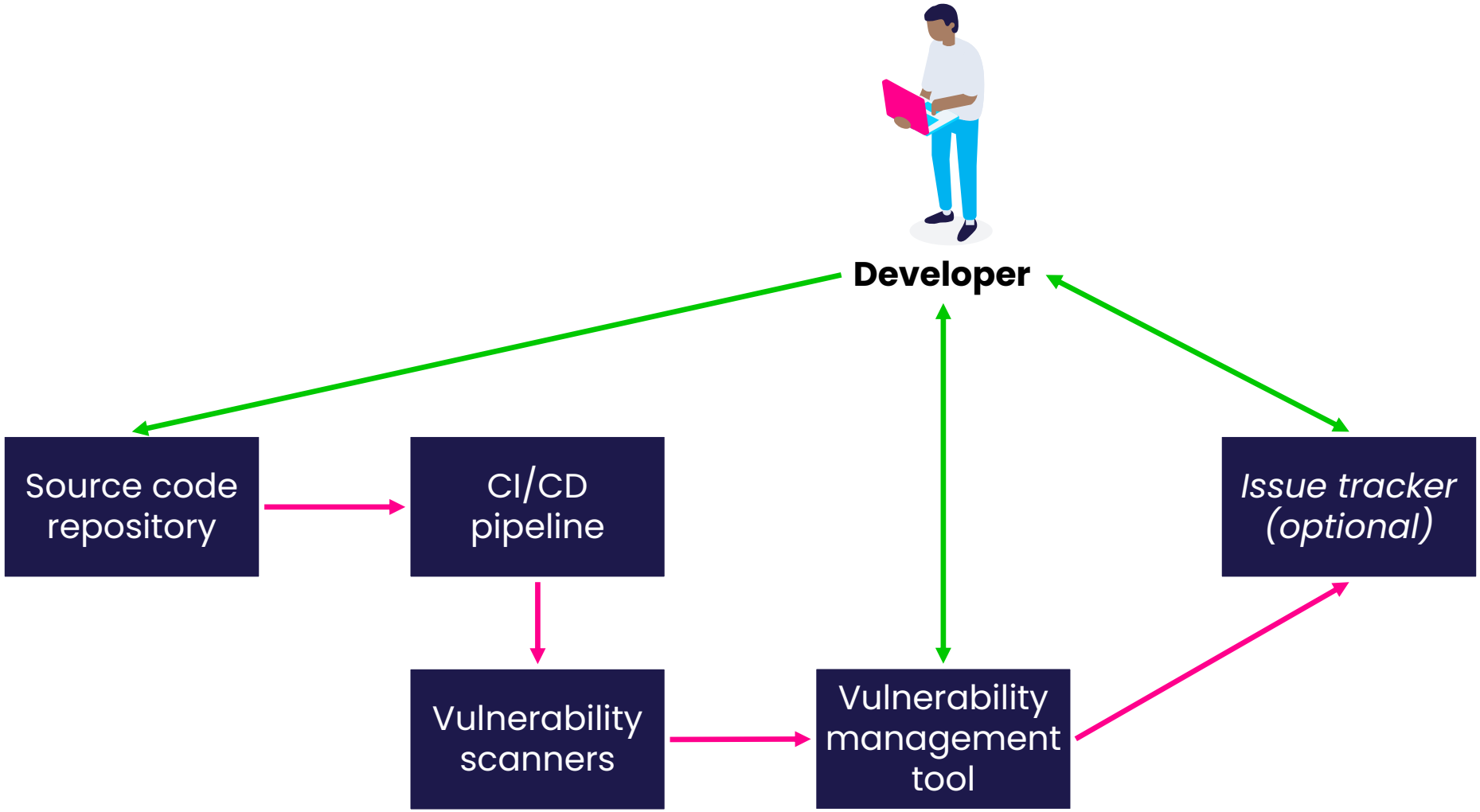
Penetration tests

› Typically for initial release and larger changes

› Outside view of the system

How does it work?

Vulnerability management workflow



› HOW DOES IT WORK?

No. 1 – Avoid vulnerabilities

Security by design

- Derive security requirements from protection needs, threat modelling and risk analysis
- Anchor security requirements in the architecture, user stories and definition of done

Keep it simple stupid

- As few libraries as possible, as many libraries as necessary
- Use small base images for Docker containers, e.g. Alpine or distroless.

Automated patch management

- Keep libraries and Docker base images regularly up-to-date
- Use tools such as the RenovateBot or DependaBot to automate the process

Security Champions and trained employees

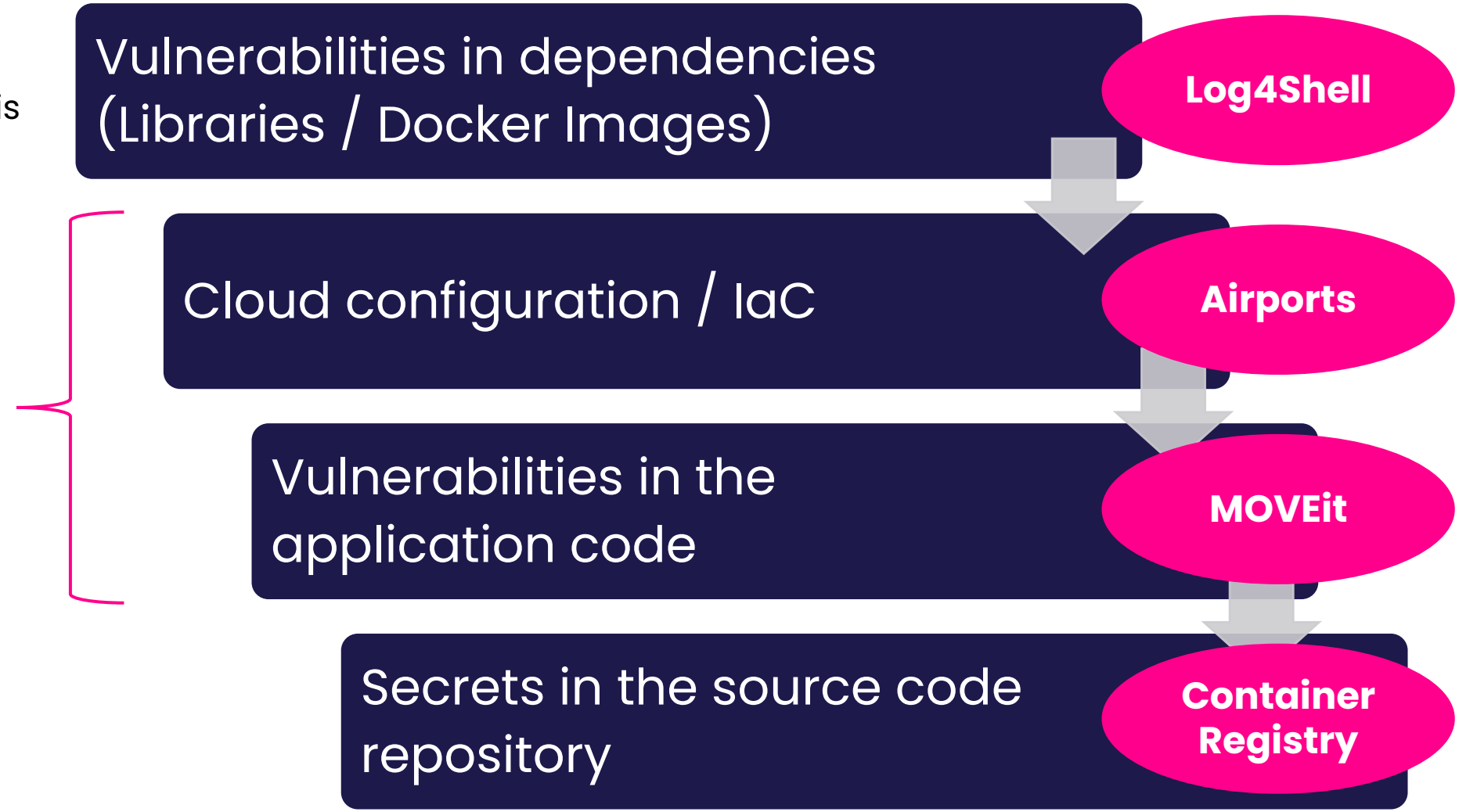
- Security Champions ensure that cybersecurity is deeply integrated into the development process
- Developers with a secure programming training avoid many pitfalls

What needs to be checked?

SCA
Software Composition Analysis

SAST
Static Application Security Testing

DAST
Dynamic Application Security Testing



No. 2 – Curated list of vulnerability scanners

From the large amount of available vulnerability scanners, some tools have proven themselves in our projects:

SCA	SAST Application	SAST Infrastructure	Secrets	DAST	Cloud Infrastructure
Dependency Check	Bandit	Checkov	GitLeaks	DrHEADer	Prowler
Dependency Track	ESLint	KICS		Cryptolyzer	Azure Defender for Cloud
Grype	FindSecBugs	Trivy		OWASP ZAP	Trivy / Prometheus
Trivy	Semgrep				

No. 3 – Ready-made templates

- Integrating vulnerability scanners into a CI/CD pipeline can be tedious.
- Each tool has to be installed differently and is called with different parameters.
- A repository with GitLab CI templates and GitHub Actions makes the process of integrating the vulnerability scanners very easy by providing uniform methods for launching the tools.
- The tools are regularly updated in the repository, so that the latest features and bug fixes are always available.

Pipeline in GitLab

```yaml
include:
  - "https://.../vulnerability_scanner.yml"


stages:
...
  - test
...

vulnerability_scans_test_stage_dev:
  stage: test
  extends: .vulnerability_scanner
  variables:
    SO_CONFIGURATION: "so_config_sast_sca_secrets_dev.yml"
  needs: []
```

https://github.com/MaibornWolff/secobserve_actions_templates

No. 3 – Ready-made templates

- Integrating vulnerability scanners into a CI/CD pipeline can be tedious.
- Each tool has to be installed differently and is called with different parameters.
- A repository with GitLab CI templates and GitHub Actions makes the process of integrating the vulnerability scanners very easy by providing uniform methods for launching the tools.
- The tools are regularly updated in the repository, so that the latest features and bug fixes are always available.

Configuration file

```
bandit_backend:
  SCANNER: bandit
  RUN_DIRECTORY: SecObserve
  TARGET: backend
  REPORT_NAME: bandit_backend.sarif

...

trivy_image_backend:
  SCANNER: trivy_image
  RUN_DIRECTORY: "SecObserve"
  TARGET: "maibornwolff/secobserve-backend:dev"
  REPORT_NAME: "trivy_backend_image.json"

...

importer:
  SO_UPLOAD: "true"
  SO_API_BASE_URL: https://secobserve-backend.example.com
  SO_PRODUCT_NAME: SecObserve
  SO_BRANCH_NAME: dev
```

https://github.com/MaibornWolff/secobserve_actions_templates

› HOW DOES IT WORK?

No. 4 – Import in vulnerability management tool

- All vulnerability scanners write their results in JSON files
- But nobody likes to read and understand JSON files
- Therefore, the files are imported in an open-source vulnerability management tool
- There are API wrappers for both of them for easy integration in a pipeline

Photo by Ferenc Almasi on Unsplash

› HOW DOES IT WORK?

No. 5 – Assessment and remediation of vulnerabilities

- OWASP flagship project
- Mature, started in 2017
- Modern single page application
- Specialised on SCA (Software Composition Analysis) and licenses with CycloneDX SBOM's

- OWASP flagship project
- Mature, started in 2015
- Complex user interface
- Suited for security specialists to manage vulnerabilities on enterprise level

- Developed by MaibornWolff
- Started in 2023
- Modern, clean single page application
- Focused on development and maintenance of cloud-based software systems

› HOW DOES IT WORK?

No. 5 – Assessment and remediation of vulnerabilities

dependency track

- OWASP flagship project
- Mature, started in 2017
- Modern single page application
- Specialised on SCA (Software Composition Analysis) and licenses with CycloneDX SBOM's

DEFECTDOJO

- OWASP flagship project
- Mature, started in 2015
- Complex user interface
- Suited for security specialists to manage vulnerabilities on enterprise level

SecObserve

- Developed by MaibornWolff
- Started in 2023
- Modern, clean single page application
- Focused on development and maintenance of cloud-based software systems

› HOW DOES IT WORK?

No. 5 – Assessment and remediation of vulnerabilities

dependency track

- OWASP flagship project
- Mature, started in 2017
- Modern single page application
- Specialised on SCA (Software Composition Analysis) and licenses with CycloneDX SBOM's

DEFECTDOJO

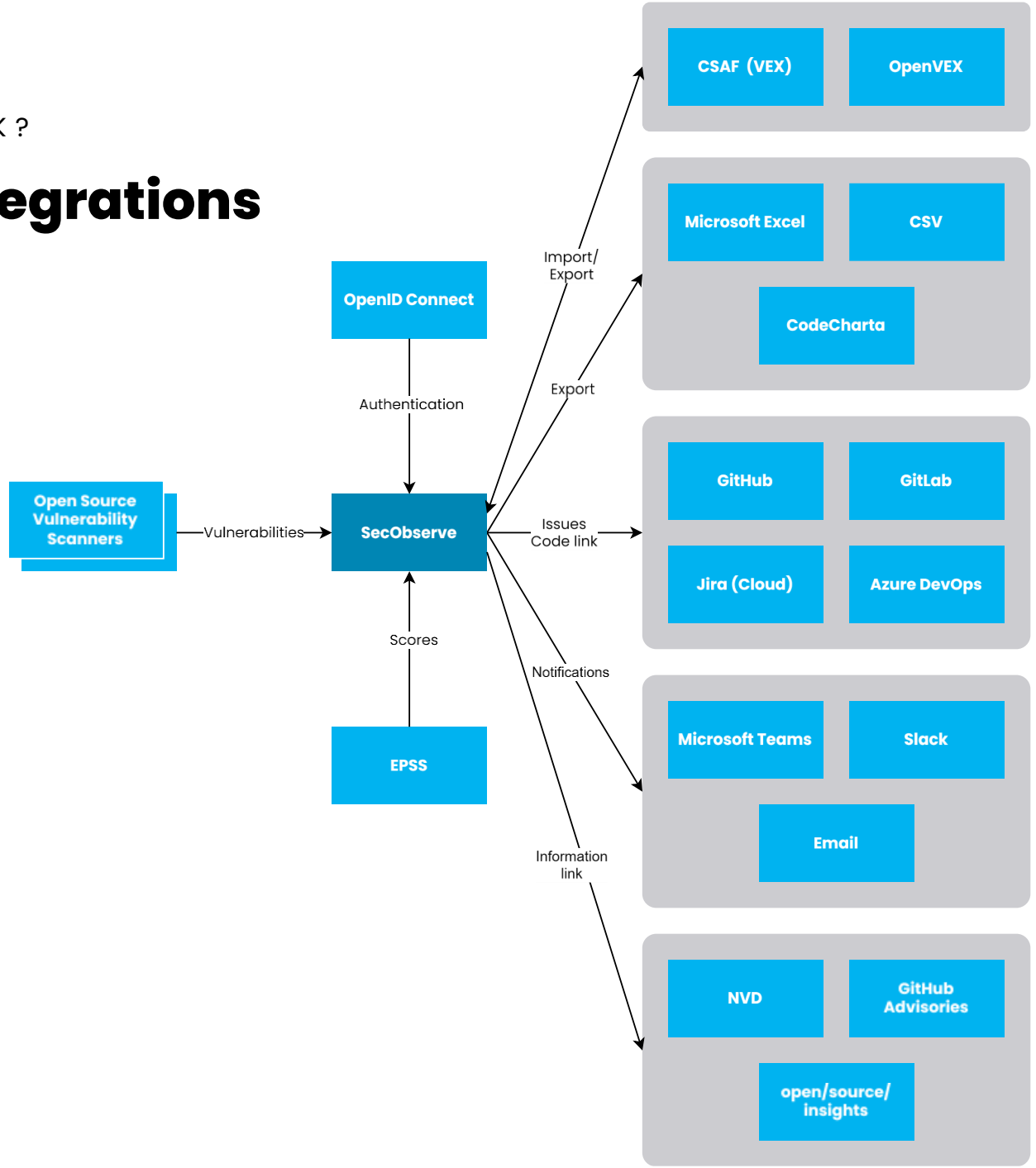
- OWASP flagship project
- Mature, started in 2015
- Complex user interface
- Suited for security specialists to manage vulnerabilities on enterprise level

SecObserve

- Developed by MaibornWolff
- Started in 2023
- Modern, clean single page application
- Focused on development and maintenance of cloud-based software systems

> HOW DOES IT WORK?

SecObserve integrations



Demo 1

Dashboard
Product Groups
Products
Observations
Notifications
Administration

Products

Name | Product group | Security gat... | Last observation change | + CREATE

Name	Product Group	Default branch / version	Security gate	Open observations	Last observation change
Troopers 2024		main	Failed	0 10 7 0 0 0	today

Rows per page: 25 1-1 of 1

Dashboard
Product Groups
Products
Observations
Notifications
Administration

Product name
Troopers 2024

Security gate
Failed

Open observations (main)
0 10 7 0 0 0

IMPORT EXPORT EDIT

OBSERVATIONS METRICS VULNERABILITY CHECKS BRANCHES / VERSIONS SETTINGS >>>

SHOW DEFAULT BRANCH / VERSION OBSERVATIONS
ADD OBSERVATION

Branch / Version: main | Title | Severity | Status: Open | Service | Component
Container | Host | Source | Resource | Scanner | Age
Duplicates

COLUMNS

Branch / Version	Title	Severity	Status	EPSS	Service	Component	Container	Host	Source	Resource	Scanner	Age	Dupl.
main	CVE-2024-4340	High	Open	0,045		sqlparse:0.4.4 (pypi)					trivy	yesterday	
main	Ensure that a user for the container has been created	High	Open						docker/backend/dev/django/Dockerfile		Checkov	yesterday	
main	Ensure that HEALTHCHECK instructions have been added to container images	High	Open						docker/backend/dev/django/Dockerfile		Checkov	yesterday	
main	CVE-2023-49083	High	Open	0,059		cryptography:41.0.5 (pypi)					trivy	yesterday	
main	CVE-2023-50782	High	Open	0,098		cryptography:41.0.5 (pypi)					trivy	yesterday	
main	CVE-2024-1135	High	Open	0,043		gunicorn:21.2.0 (pypi)					trivy	yesterday	
main	CVE-2024-24680	High	Open	0,081		django:4.2.8 (pypi)					trivy	yesterday	
main	CVE-2024-26130	High	Open	0,045		cryptography:41.0.5 (pypi)					trivy	yesterday	
main	CVE-2024-27351	High	Open	0,044		django:4.2.8 (pypi)					trivy	yesterday	

Dashboard
Product Groups
Products
Observations
Notifications
Administration

Observation

Severity: Medium
Status: Open
Title: **hardcoded_sql_expressions**

Description

Possible SQL injection vector through string-based query construction.

Snippet: `query = f"SELECT * FROM access_control_user WHERE access_control_user.username = '{username}' and password = '{password}'"`

Issue_Confidence: LOW

Issue_Severity: MEDIUM

Tags: ['security', 'external/cwe/cwe-89']

Precision: low

Origins

Source

Source file	Source line start	Source line end
backend/application/access_control/api/views.py	74	74

Log

User	Severity	Status	Comment	Created
-product-92-api_token-	Medium	Open	Set by parser	23.6.2024, 21:32:28

Rows per page: 25 1-1 of 1

Metadata

Product
Troopers 2024
Branch / Version
main
Parser name
SARIF
Parser type
SAST
Parser source
File
Scanner
Bandit / 1.7.8
Upload filename
bandit_backend.sarif
Import last seen
24.6.2024, 12:50:25
Created
23.6.2024, 21:32:28

References

https://bandit.readthedocs.io/en/1.7.8/plugins/b608_hardcoded_sql_expressions.html

Evidences

Result
Rule

Dashboard
Product Groups
Products
Observations
Notifications
Administration

Observation

Severity: High
Status: Open
Title: CVE-2023-49083

Description

cryptography is a package designed to expose cryptographic primitives and recipes to Python developers. Calling `load_pem_pkcs7_certificates` or `load_der_pkcs7_certificates` could lead to a NULL-pointer dereference and segfault. Exploitation of this vulnerability poses a serious risk of Denial of Service (DoS) for any application attempting to deserialize a PKCS7 blob/ certificate. The consequences extend to potential disruptions in system availability and stability. This vulnerability has been patched in version 41.0.6.

Recommendation

Upgrade cryptography to version 41.0.6

Vulnerability

Vulnerability ID	CVSS3 score	CVSS3 vector	CWE	EPSS score (%)	EPSS percentile (%)
CVE-2023-49083	7,5	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H	476	0,059	25,249

Origins

Component

Component name	Component version	Component PURL
cryptography	41.0.5	pkg:pypi/cryptography@41.0.5

Component dependencies

backend/poetry.lock --> poetry.lock --> cryptography:41.0.5
backend/poetry.lock --> poetry.lock --> django-encrypted-model-fields:0.6.5 --> cryptography:41.0.5

Log

User	Severity	Status	Comment	Created
-product-92-api_token-	High	Open	Set by parser	23.6.2024, 21:33:33

Rows per page: 25 1-1 of 1

Metadata

Product
Troopers 2024
Branch / Version
main
Parser name
CycloneDX
Parser type
SCA
Parser source
File
Scanner
trivy / 0.52.0
Upload filename
trivy_backend_poetry.json
Import last seen
24.6.2024, 12:51:31
Created
23.6.2024, 21:33:33

References

https://avd.aquasec.com/nvd/cve-2023-49083
http://www.openwall.com/lists/oss-security/2023/11/29/2
https://access.redhat.com/errata/RHSA-2024:2337
https://access.redhat.com/security/cve/CVE-2023-49083
https://bugzilla.redhat.com/2255331
https://bugzilla.redhat.com/show_bug.cgi?id=2255331
https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2023-49083
https://errata.almalinux.org/9/ALSA-2024-2337.html
https://errata.rockylinux.org/RLSA-2024:2337
https://github.com/pyca/cryptography
https://github.com/pyca/cryptography/commit/f09c261ca10a31fe41b1262306db7f8f1da0e48a
https://github.com/pyca/cryptography/pull/9926
https://github.com/pyca/cryptography/security/advisories/GHSA-jfhm-5ghh-2f97
https://github.com/pypa/advisory-database/tree/main/vulns/cryptography/PYSEC-2023-254.yaml

Dashboard
Product Groups
Products
Observations
Notifications
Administration

Observation

Severity: High
Status: Open
Title: **Ensure that a user for the container has been created**

Description

Ensure that a user for the container has been created

Snippet:

```
FROM python:3.11.6-alpine as python

# Python build stage
FROM python as python-build-stage

ARG BUILD_ENVIRONMENT=dev

# Install packages
# kics-scan ignore-block
# versions of dependencies from distribution are ok
RUN apk add --no-cache --virtual .build-deps \
    ca-certificates gcc postgresql-dev linux-headers musl-dev libffi-dev mariadb-dev

# install dependencies with poetry
RUN pip install poetry==1.7.1

ENV POETRY_NO_INTERACTION=1 \
    POETRY_VIRTUALENVS_IN_PROJECT=1 \
    POETRY_VIRTUALENVS_CREATE=1 \
    POETRY_CACHE_DIR=/tmp/poetry_cache

WORKDIR /

COPY backend/pyproject.toml backend/poetry.lock ./
RUN poetry install --without prod,unittests --no-root && rm -rf $POETRY_CACHE_DIR


# Python 'run' stage
FROM python as python-run-stage

ARG BUILD_ENVIRONMENT=dev
ARG APP_HOME=/app
ARG COMMIT_ID=undefined

ENV PYTHONUNBUFFERED 1
ENV PYTHONDONTWRITEBYTECODE 1
ENV BUILD_ENV ${BUILD_ENVIRONMENT}

WORKDIR ${APP_HOME}

# Install binaries and libs for PostgreSQL
# kics-scan ignore-block
# versions of dependencies from distribution are ok
```

Metadata

Product
Troopers 2024

Branch / Version
main

Parser name
SARIF

Parser type
SAST

Parser source
File

Scanner
Checkov / 3.2.93

Upload filename
checkov.sarif

Import last seen
24.6.2024, 12:51:12

Created
23.6.2024, 21:33:14

References

https://docs.prismacloud.io/en/enterprise-edition/policy-reference/docker-policies/docker-policy-index/ensure-that-a-user-for-the-container-has-been-created

Evidences

Result

Rule

No. 5 – Assessment and remediation of vulnerabilities

- Undesired results can be **false positives**, **not affected** or **not security related**
- Elimination by changing the configuration of the scanner, manual assessment, rule-based assessment

- The development team weighs the degree of risk of the remaining results
- The threat-based risk analysis helps to decide whether the risk of a vulnerability can be accepted in the current context

- Now the development team has the list of remaining vulnerabilities that need to be fixed
- With the next scan, the fixed vulnerabilities will no longer be reported and will automatically be set to completed in SecObserve.

Demo 2

Dashboard
Product Groups
Products
Observations
Notifications
Administration

Observation

Severity: Medium
Status: Open
Title: **hardcoded_sql_expressions**

Description

Possible SQL injection vector through string-based query construction.

Snippet: `query = f"SELECT * FROM access_control_user WHERE access_control_user.username = '{username}' and password = '{password}'"`

Issue_Confidence: LOW

Issue_Severity: MEDIUM

Tags: ['security', 'external/cwe/cwe-89']

Precision: low

Origins

Source

Source file	Source line start	Source line end
backend/application/access_control/api/views.py	74	74

Log

User	Severity	Status	Comment	Created
-product-92-api_token-	Medium	Open	Set by parser	23.6.2024, 21:32:28

Rows per page: 25 1-1 of 1

Metadata

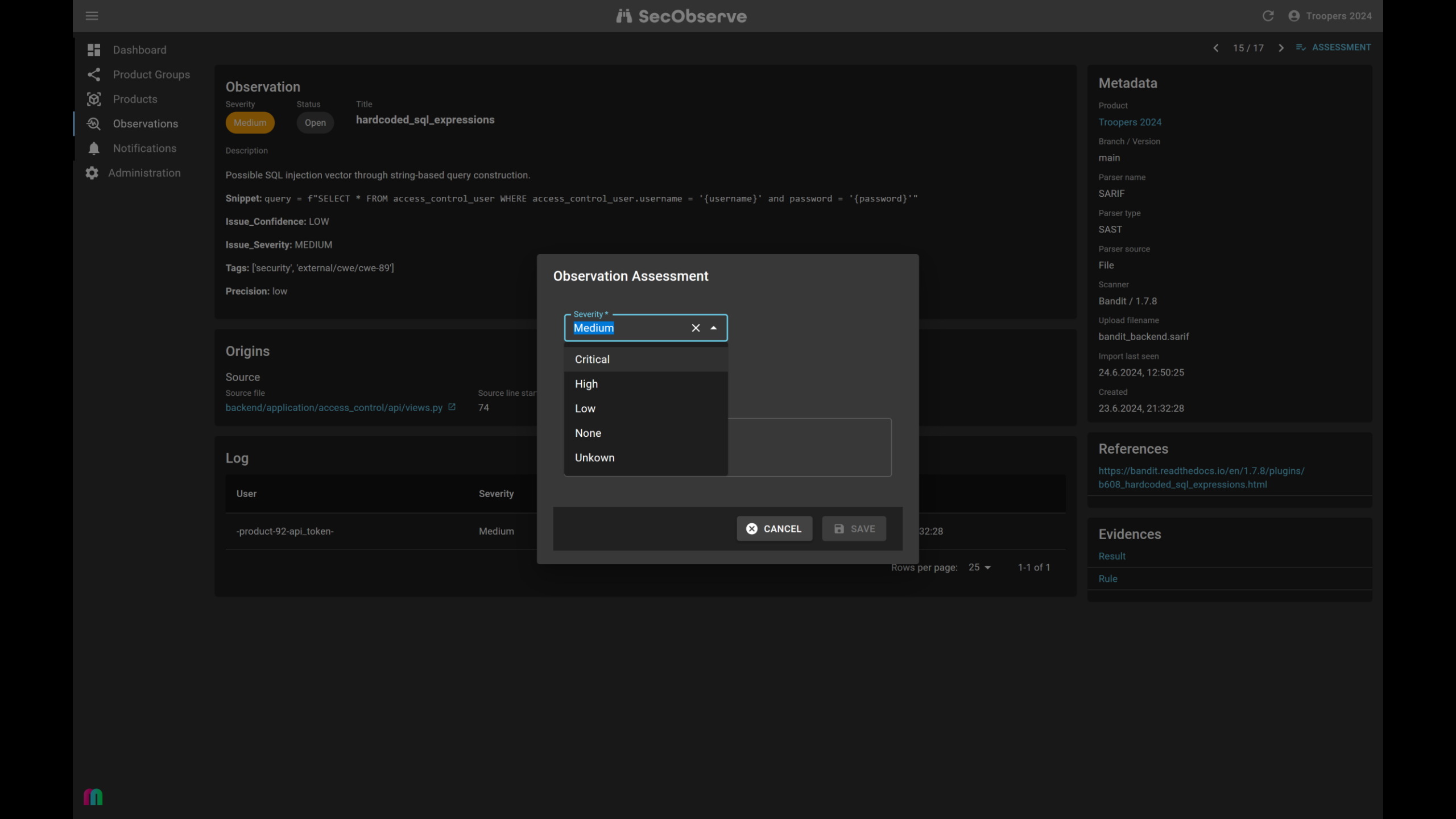
Product
Troopers 2024
Branch / Version
main
Parser name
SARIF
Parser type
SAST
Parser source
File
Scanner
Bandit / 1.7.8
Upload filename
bandit_backend.sarif
Import last seen
24.6.2024, 12:50:25
Created
23.6.2024, 21:32:28

References

https://bandit.readthedocs.io/en/1.7.8/plugins/b608_hardcoded_sql_expressions.html

Evidences

Result
Rule


SecObserve
Troopers 2024
Dashboard
Product Groups
Products
Observations
Notifications
Administration
15 / 17
ASSESSMENT
Observation
Severity
Medium
Status
Open
Title
hardcoded_sql_expressions
Description
Possible SQL injection vector through string-based query construction.
Snippet: query = f"SELECT * FROM access_control_user WHERE access_control_user.username = '{username}' and password = '{password}'"
Issue_Confidence: LOW
Issue_Severity: MEDIUM
Tags: ['security', 'external/cwe/cwe-89']
Precision: low
Origins
Source
Source file
backend/application/access_control/api/views.py
Source line start
74
Log
User
Severity
-product-92-api_token-
Medium
32:28
Rows per page: 25
1-1 of 1
Metadata
Product
Troopers 2024
Branch / Version
main
Parser name
SARIF
Parser type
SAST
Parser source
File
Scanner
Bandit / 1.7.8
Upload filename
bandit_backend.sarif
Import last seen
24.6.2024, 12:50:25
Created
23.6.2024, 21:32:28
References
https://bandit.readthedocs.io/en/1.7.8/plugins/b608_hardcoded_sql_expressions.html
Evidences
Result
Rule
Observation Assessment
Severity *
Medium
Critical
High
Low
None
Unkown
CANCEL
SAVE

Dashboard
Product Groups
Products
Observations
Notifications
Administration

Observation

Severity: Medium
Status: Open
Title: **hardcoded_sql_expressions**

Description

Possible SQL injection vector through string-based query construction.

Snippet: `query = f"SELECT * FROM access_control_user WHERE access_control_user.username = '{username}' and password = '{password}'"`

Issue_Confidence: LOW

Issue_Severity: MEDIUM

Tags: ['security', 'external/cwe/cwe-89']

Precision: low

Origins

Source

Source file: backend/application/access_control/api/views.py
Source line start: 74

Log

User	Severity
-product-92-api_token-	Medium

32:28

Rows per page: 25
1-1 of 1

Metadata

Product: Troopers 2024
Branch / Version: main
Parser name: SARIF
Parser type: SAST
Parser source: File
Scanner: Bandit / 1.7.8
Upload filename: bandit_backend.sarif
Import last seen: 24.6.2024, 12:50:25
Created: 23.6.2024, 21:32:28

References

https://bandit.readthedocs.io/en/1.7.8/plugins/b608_hardcoded_sql_expressions.html

Evidences

Result
Rule

Observation Assessment

Severity *: Critical
Status *: Open
Comment *: Everyone can login to application without knowing user or password!!!

CANCEL
SAVE

Dashboard
Product Groups
Products
Observations
Notifications
Administration

15 / 17 ASSESSMENT REMOVE ASSESSMENT

Observation

Severity: Critical
Status: Open
Title: **hardcoded_sql_expressions**

Parser severity
Medium

Assessment severity
Critical

Description

Possible SQL injection vector through string-based query construction.

Snippet: `query = f"SELECT * FROM access_control_user WHERE access_control_user.username = '{username}' and password = '{password}'"`

Issue_Confidence: LOW

Issue_Severity: MEDIUM

Tags: ['security', 'external/cwe/cwe-89']

Precision: low

Origins

Source

Source file	Source line start	Source line end
backend/application/access_control/api/views.py	74	74

Log

User	Severity	Status	Comment	Created
Troopers 2024	Critical	---	Everyone can login to application without knowing user or password!!!	24.6.2024, 14:39:38
-product-92-api_token-	Medium	Open	Set by parser	23.6.2024, 21:32:28

Rows per page: 25 1-2 of 2

Metadata

Product
Troopers 2024

Branch / Version
main

Parser name
SARIF

Parser type
SAST

Parser source
File

Scanner
Bandit / 1.7.8

Upload filename
bandit_backend.sarif

Import last seen
24.6.2024, 12:50:25

Created
23.6.2024, 21:32:28

References

https://bandit.readthedocs.io/en/1.7.8/plugins/b608_hardcoded_sql_expressions.html

Evidences

Result

Rule

Dashboard
Product Groups
Products
Observations
Notifications
Administration

Observation

Severity: Medium
Status: False positive
Title: python.lang.maintainability.is-function-without-parentheses.is-function-without-parentheses

Parser status: Open

Rule status: False positive

Description

Is "is_superuser" a function or an attribute? If it is a function, you may have meant user.is_superuser() because user.is_superuser is always true.

Snippet: `if user.is_superuser:`

Precision: very-high

Origins

Source

Source file	Source line start	Source line end
backend/application/access_control/services/jwt_authentication.py	18	18

Log

User	Severity	Status	Comment	Created
Troopers 2024	---	False positive	Django user has attributes that look like functions	23.6.2024, 21:43:51
-product-92-api_token-	Medium	Open	Set by parser	23.6.2024, 21:32:51

Rows per page: 25 1-2 of 2

Metadata

Product: Troopers 2024
Branch / Version: main
Parser name: SARIF
Parser type: SAST
Parser source: File
Scanner: Semgrep OSS / 1.75.0
Upload filename: semgrep_backend.sarif
Product rule name: Semgrep / Function without parenthesis
Import last seen: 24.6.2024, 12:50:48
Created: 23.6.2024, 21:32:51

References

https://semgrep.dev/r/python.lang.maintainability.is-function-without-parentheses.is-function-without-parentheses

Evidences

Result
Rule

Dashboard
Product Groups
Products
Observations
Notifications
Administration

Product name
Troopers 2024

Security gate
Failed

Open observations (main)
1 10 6 0 0 0

IMPORT EXPORT EDIT

OBSERVATIONS METRICS VULNERABILITY CHECKS BRANCHES / VERSIONS SETTINGS RULES API CONFIGURATIONS MEMBERS API TOKEN

ADD PRODUCT RULE APPLY RULES

Name Parser

Name	New severity	New status	Enabled	Parser	Scanner prefix	Observation title
Semgrep / Function without parenthesis		False positive	✓		Semgrep	python\.lang\.maintainability\.is-function-without-parentheses\.is-function-without-parentheses

Rows per page: 25 1-1 of 1

Dashboard
Product Groups
Products
Observations
Notifications
Administration

EDIT

Product Rule

Product
Troopers 2024

Name
Semgrep / Function without parenthesis

Description
Django user has attributes that look like functions

New status
False positive

Enabled
✓

Last changed by
Troopers 2024

Observation

Parser

Scanner prefix
Semgrep

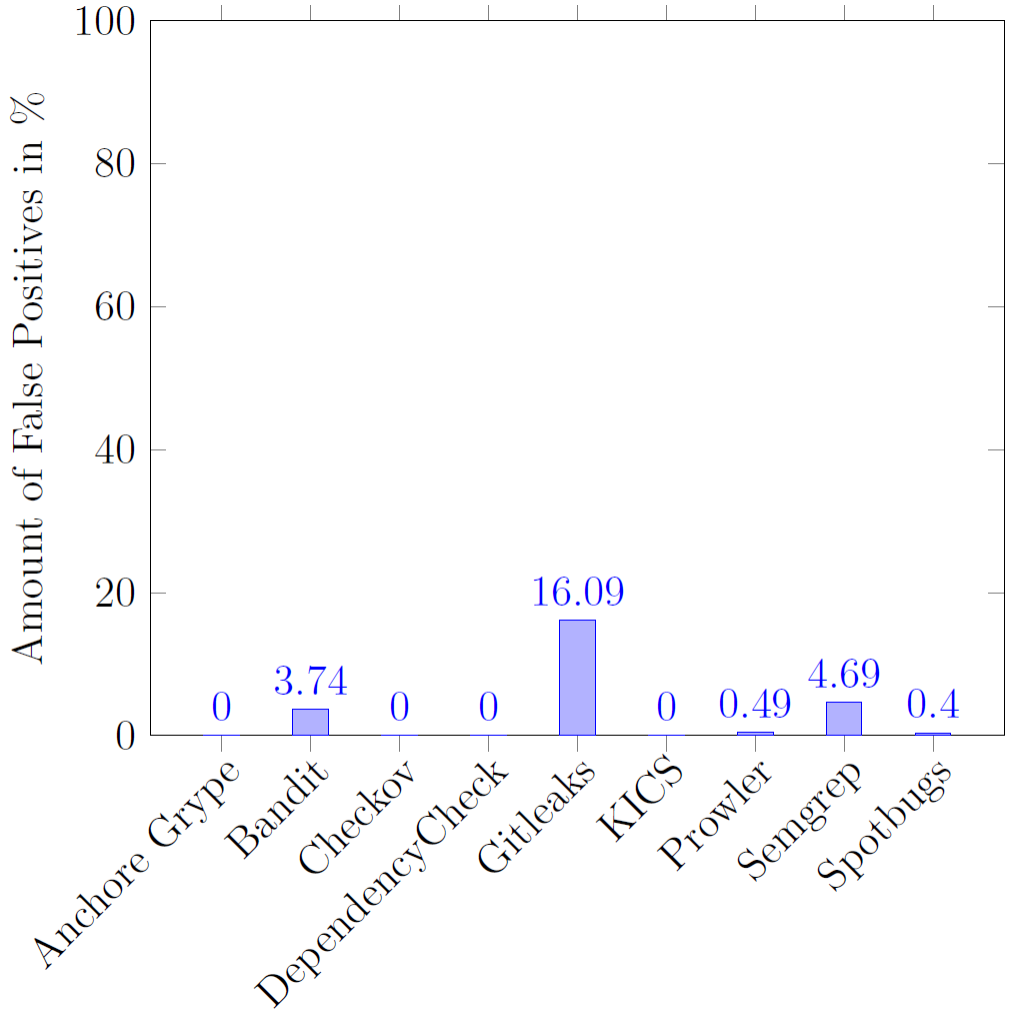
Title
python\.lang\.maintainability\.is-function-without-parentheses\.is-function-without-parentheses

Does it work?

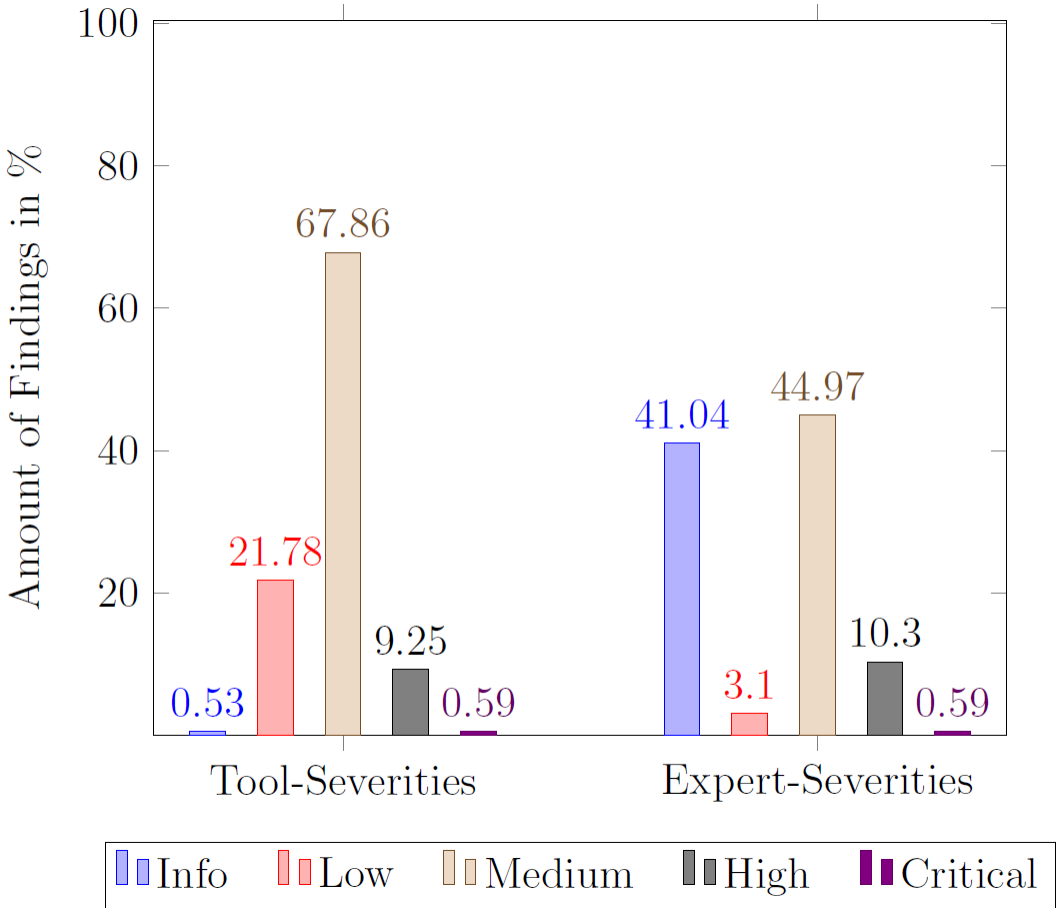
Quality of scan results – false positives

Results of the master thesis "*Security analysis and comparison of different static application security testing tools*" by Christian Näther.

In the master's thesis, various open-source vulnerability scanners were evaluated through their use in three software development projects and an assessment of the results by security experts.



Quality of scan results – severities



Deviation of tools and expert's severities

	Info	Low	Medium	High	Critical
Info	100%	0%	0%	0%	0%
Low	81.5%	13.7%	0.5%	4.3%	0%
Medium	33.5%	0.2%	64.9%	1.4%	0%
High	0%	0%	8.9%	91.1%	0%
Critical	0%	0%	0%	0%	100%

Rows: Tool results
Columns: Experts assessment

› DOES IT WORK?

Open source vs. commercial products

- Open-source vulnerability scanners are often the basis of commercial products or developed by well respected communities
- Open-source vulnerability management can often be introduced faster without the need for lengthy vendor selection and procurement processes
- Commercial products often have better support of developers for fixing the vulnerabilities
- In the future, AI might give commercial products a huge advantage in identifying vulnerabilities and helping to fix them

Summary

> SUMMARY

Open source vulnerability management in 5 steps

Avoidance of vulnerabilities

Curated list of Open Source vulnerability scanners

Ready-made templates for CI/CD pipelines

Automated import into SecObserve

Transparency, assessment and elimination of vulnerabilities

Thank you for your attention

Questions?