

Relatório Final - Evidências de Segurança (Tarefa 1, 2, 3 e 4)

Introdução

Este relatório apresenta as evidências práticas das tarefas de segurança realizadas. Foram executadas e documentadas análises em diferentes camadas: SAST (Semgrep), DAST (OWASP ZAP), SCA (Dependency-Check) e implementação de CI/CD Security Pipeline no GitHub Actions.

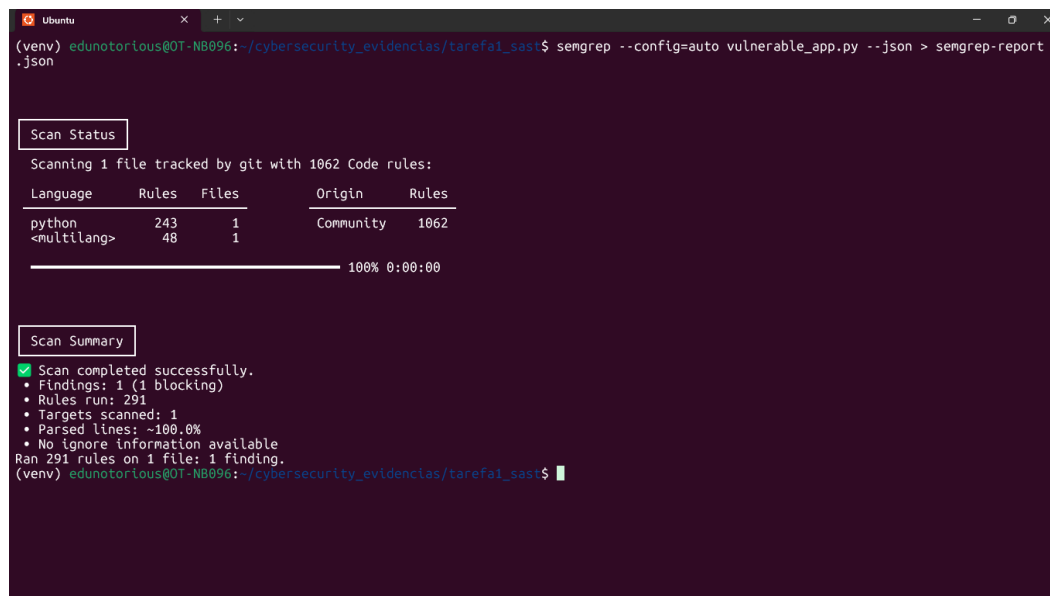
Tarefa 1 - SAST (Semgrep / Análise Estática)

Objetivo: Realizar análise estática de código usando Semgrep para identificar vulnerabilidades no código fonte (ex.: SQL Injection, logs sensíveis).

Comando utilizado (exemplo):

```
semgrep --config=auto vulnerable_app.py --json > semgrep-report.json  
semgrep --config=auto vulnerable_app.py
```

Evidências (prints do terminal mostrando execução do Semgrep e resumo dos achados):



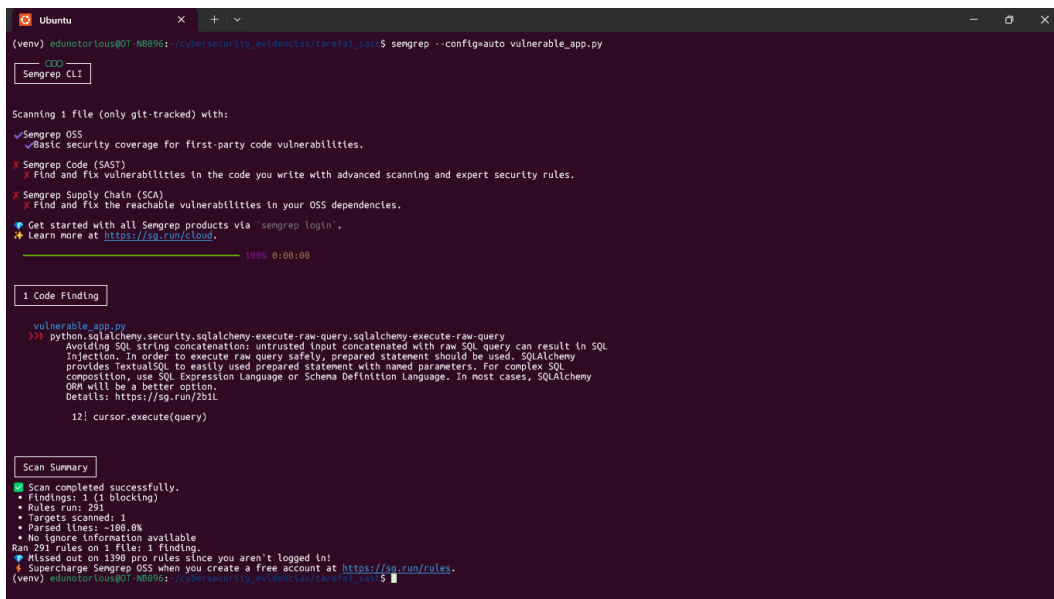
```
(venv) edunotorious@0T-NB096: ~/cybersecurity_evidencias/tarefa1_sast$ semgrep --config=auto vulnerable_app.py --json > semgrep-report.json
```

Scan Status
Scanning 1 file tracked by git with 1062 Code rules:

Language	Rules	Files	Origin	Rules
python	243	1	Community	1062
<multilang>	48	1		

100% 0:00:00

Scan Summary
✔ Scan completed successfully.
• Findings: 1 (1 blocking)
• Rules run: 291
• Targets scanned: 1
• Parsed lines: ~100.0%
• No ignore information available
Ran 291 rules on 1 file: 1 finding.
(venv) edunotorious@0T-NB096: ~/cybersecurity_evidencias/tarefa1_sast\$



```
(venv) edumotorious@0T-NB096: /cybersecurity_evidencias/tarefa1_sas$ sengrep --config=auto vulnerable_app.py

Sengrep CLI

Scanning 1 file (only git-tracked) with:
✓ Sengrep OSS
  ✓ Basic security coverage for first-party code vulnerabilities.
✗ Sengrep Code (SAST)
  ✗ Find and fix vulnerabilities in the code you write with advanced scanning and expert security rules.
✗ Sengrep Supply Chain (SCA)
  ✗ Find and fix the reachable vulnerabilities in your OSS dependencies.
💡 Get started with all Sengrep products via 'sengrep login'.
🌟 Learn more at https://sg.run/cloud.

100% 0:00:00

1 Code Finding

vulnerable_app.py
12: cursor.execute(query)
Avoiding SQL string concatenation: untrusted input concatenated with raw SQL query can result in SQL Injection. In order to execute raw query safely, prepared statement should be used. SQLAlchemy provides TextualSQL to easily used prepared statement with named parameters. For complex SQL composition, use SQL Expression Language or Schema Definition Language. In most cases, SQLAlchemy ORM will be a better option.
Details: https://sg.run/2b1l

Scan Summary
✓ Scan completed successfully.
• Findings: 1 (1 blocking)
• Rules run: 291
• Targets scanned: 1
• Parsed lines: ~100.0%
• No ignore information available
Ran 291 rules on 1 file: 1 finding.
💡 Missed out on 1398 pro rules since you aren't logged in!
🚀 Supercharge Sengrep OSS when you create a free account at https://sg.run/rules.
(venv) edumotorious@0T-NB096: /cybersecurity_evidencias/tarefa1_sas$
```

Resumo dos achados e recomendações:

- Encontrado 1 achado (SQL concatenation leading to SQL Injection).
- Recomenda-se usar consultas parametrizadas e remover logs que exponham senhas.
- Exemplo de correção (SQL parametrizado e logs sem senha):

```
def get_user_by_id(user_id):
    cursor.execute("SELECT * FROM users WHERE id = ?", (user_id,))
    return cursor.fetchall()

def login(username, password):
    print(f"Login attempt for user: {username}")
    return True
```

Tarefa 2 - DAST (ZAP Baseline)


A tarefa 2 consistiu em rodar o OWASP ZAP (Zed Attack Proxy) em modo baseline contra a aplicação vulnerável (Juice Shop), gerando um relatório em HTML com os resultados.

Evidências:

```
Ubuntu
(venv) edunotorious@101-NB096:~/cybersecurity_evidencias/tarefa2_dast$ xdg-open reports/zap-report.html
WARNING: You don't seem to have any mimeinfo.cache files.
Try running the update-desktop-database command. If you
don't have this command you should install the
desktop-file-utils package. This package is available from
http://freedesktop.org/wiki/Software/desktop-file-utils/
No applications found for mimetype: text/html
./usr/bin/xdg-open: 882: x-www-browser: not found
./usr/bin/xdg-open: 882: firefox: not found
./usr/bin/xdg-open: 882: iceweasel: not found
./usr/bin/xdg-open: 882: seamonkey: not found
./usr/bin/xdg-open: 882: mozilla: not found
./usr/bin/xdg-open: 882: epiphany: not found
./usr/bin/xdg-open: 882: konqueror: not found
./usr/bin/xdg-open: 882: chromium: not found
./usr/bin/xdg-open: 882: chromium-browser: not found
./usr/bin/xdg-open: 882: google-chrome: not found
./usr/bin/xdg-open: 882: www-browser: not found
./usr/bin/xdg-open: 882: links2: not found
./usr/bin/xdg-open: 882: elinks: not found
./usr/bin/xdg-open: 882: links: not found
./usr/bin/xdg-open: 882: lynx: not found
./usr/bin/xdg-open: 882: w3m: not found
xdg-open: no method available for opening 'reports/zap-report.html'
(venv) edunotorious@101-NB096:~/cybersecurity_evidencias/tarefa2_dast$ explorer.exe .
(venv) edunotorious@101-NB096:~/cybersecurity_evidencias/tarefa2_dast$
```

ZAP Scanning Report

ws1localhost/Ubuntu/home/edunotorious/cybersecurity_evidencias/tarefa2_dast/reports/zap-report.html

**ZAP Scanning Report**

Site: <http://juice:3000>

Generated on Sun, 21 Sept 2025 19:18:31

ZAP Version: 2.16.1

ZAP by [Checkmarx](#)

Summary of Alerts

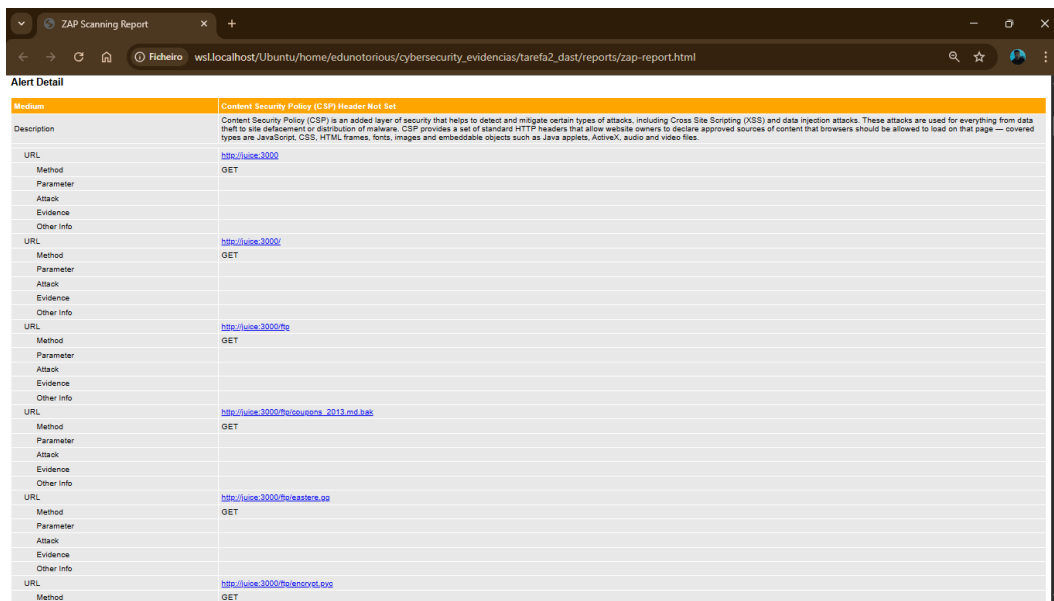
Risk Level	Number of Alerts
High	0
Medium	2
Low	5
Informational	5
False Positives	0

Summary of Sequences

For each step result (Pass/Fail): risk (of highest alert(s) for the step, if any).

Alerts

Name	Risk Level	Number of Instances
Content Security Policy (CSP) Header Not Set	Medium	11
Cross-Domain Misconfiguration	Medium	12
Cross-Domain JavaScript Source File Inclusion	Low	12
Dangerous JS Functions	Low	2
Deprecated Feature Policy Header Set	Low	13
Insufficient Site Isolation Against Scripts Vulnerability	Low	10
Timingating Disclosure - Unix	Low	16
Information Disclosure - Suspicious Comments	Informational	2
Modern Web Application	Informational	11
Non-Scriptable Content	Informational	2
Storable and Cacheable Content	Informational	1
Storable but Non-Cacheable Content	Informational	8



O relatório apontou vulnerabilidades de nível Médio, Baixo e Informacional, como a ausência de Content Security Policy (CSP) e configuração incorreta de cabeçalhos HTTP.

Tarefa 3 - SCA (Dependency Check)

A tarefa 3 consistiu em realizar a análise de dependências utilizando o OWASP Dependency Check, que identificou bibliotecas vulneráveis no projeto Node.js.

Evidências:

```

edunotorious@OT-NB096:~/cybersecurity_evidencias/tarefa3_sca$ cd ~/cybersecurity_evidencias/tarefa3_sca
edunotorious@OT-NB096:~/cybersecurity_evidencias/tarefa3_sca$ ls -lh reports/dependency-check-report.html
-rw-r--r-- 1 100999 100999 2.0M Sep 21 17:16 reports/dependency-check-report.html
edunotorious@OT-NB096:~/cybersecurity_evidencias/tarefa3_sca$ explorer.exe reports
edunotorious@OT-NB096:~/cybersecurity_evidencias/tarefa3_sca$

```



Dependency-Check is an open source tool performing a best effort analysis of first party dependencies. False positives and false negatives may exist in the analysis performed by the tool. Use of the tool and the reporting provided constitutes acceptance for use in an AS-IS condition, and there are NO warranties, implied or otherwise, with regard to the analysis or its use. Any use of the tool and the reporting provided is at the user's risk. In no event shall the copyright holder or OWASP be held liable for any damages whatsoever arising out of or in connection with the use of this tool, the analysis performed, or the resulting report.

[How to read the report](#) | [Suppressing false positives](#) | [Getting Help: oitubh issues](#)

[Sponsor](#)

Project: mynode

Scan Information ([show all](#)):

- dependency-check version: 12.1.6
- Report Generated On: Sun, 21 Sep 2025 20:16:10 GMT
- Dependencies Scanned: 1297 (1187 unique)
- Vulnerable Dependencies: 9
- Vulnerabilities Found: 32
- Vulnerabilities Suppressed: 0
- ..

Analysis Exceptions

Error initializing OSS Index analyzer due to missing user/password credentials. Authentication is now required: <https://ossindex.sonatype.org/doc/auth-required>

Summary

Summary of Vulnerable Dependencies ([click to show all](#))

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count
body-parser:1.18.2		pkg:npm/body-parser@1.18.2	HIGH	1		5
cookie:0.3.1		pkg:npm/cookie@0.3.1	LOW	1		6
express:4.16.0		pkg:npm/express@4.16.0	MEDIUM	2		7
lodash:4		pkg:javascript/lodash@4.17.0	CRITICAL	7		3
lodash:4.17.0	cpe:2.3:a:lodash:lodash:4.17.0:*:*:*:*:*	pkg:npm/lodash@4.17.0	CRITICAL	14	Highest	7
path-to-regexp:0.1.7		pkg:npm/path-to-regexp@0.1.7	HIGH	2		5
qs:0.5.1	cpe:2.3:a:qs:qs:0.5.1:*:*:*:*:*	pkg:npm/qs@0.5.1	HIGH	2	Highest	6

Summary

Summary of Vulnerable Dependencies ([click to show all](#))

Dependency	Vulnerability IDs	Package	Highest Severity	CVE Count	Confidence	Evidence Count
body-parser:1.18.2		pkg:npm/body-parser@1.18.2	HIGH	1		5
cookie:0.3.1		pkg:npm/cookie@0.3.1	LOW	1		6
express:4.16.0		pkg:npm/express@4.16.0	MEDIUM	2		7
lodash:4		pkg:javascript/lodash@4.17.0	CRITICAL	7		3
lodash:4.17.0	cpe:2.3:a:lodash:lodash:4.17.0:*:*:*:*:*	pkg:npm/lodash@4.17.0	CRITICAL	14	Highest	7
path-to-regexp:0.1.7		pkg:npm/path-to-regexp@0.1.7	HIGH	2		5
qs:0.5.1	cpe:2.3:a:qs:qs:0.5.1:*:*:*:*:*	pkg:npm/qs@0.5.1	HIGH	2	Highest	6
serve:0.16.0	cpe:2.3:a:serve:serve:0.16.0:*:*:*:*:*	pkg:npm/serve@0.16.0	MEDIUM	2	Highest	6
serve-static:1.13.0	cpe:2.3:a:serve-static:serve-static:1.13.0:*:*:*:*:*	pkg:npm/serve-static@1.13.0	MEDIUM	1	Highest	6

Dependencies (vulnerable)

body-parser:1.18.2

Description:
Node.js body parsing middleware

License:
MIT

File Path: /src/package-lock.json?express=4.16.0/body-parser:1.18.2

Referenced In Projects/Scopes:

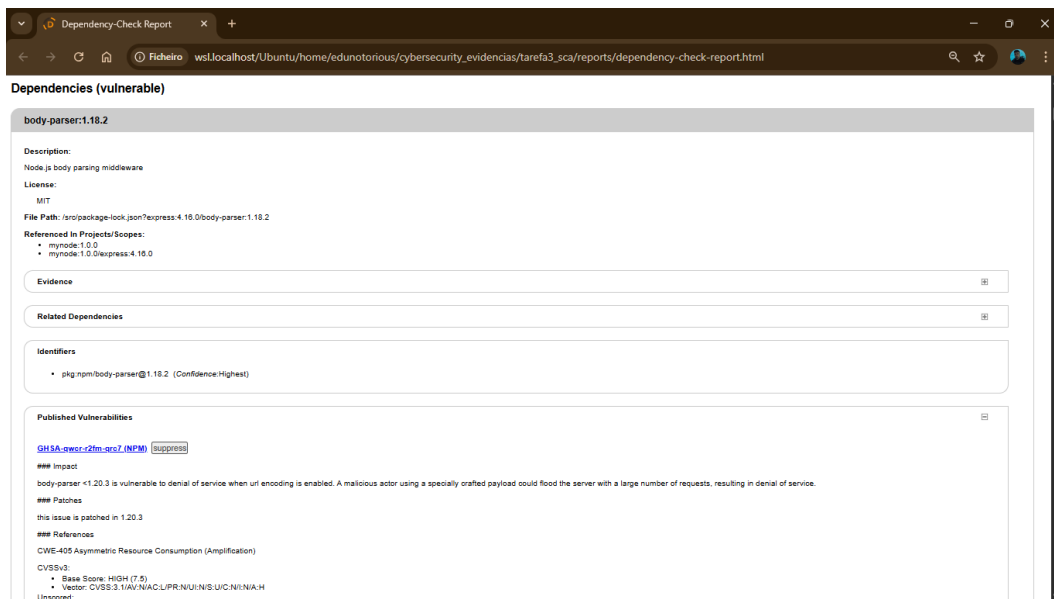
- mynode:1.0.0
- mynode:1.0.0/express:4.16.0

Evidence

Related Dependencies

Identifiers

- pkg:npm/body-parser@1.18.2 (Confidence: Highest)

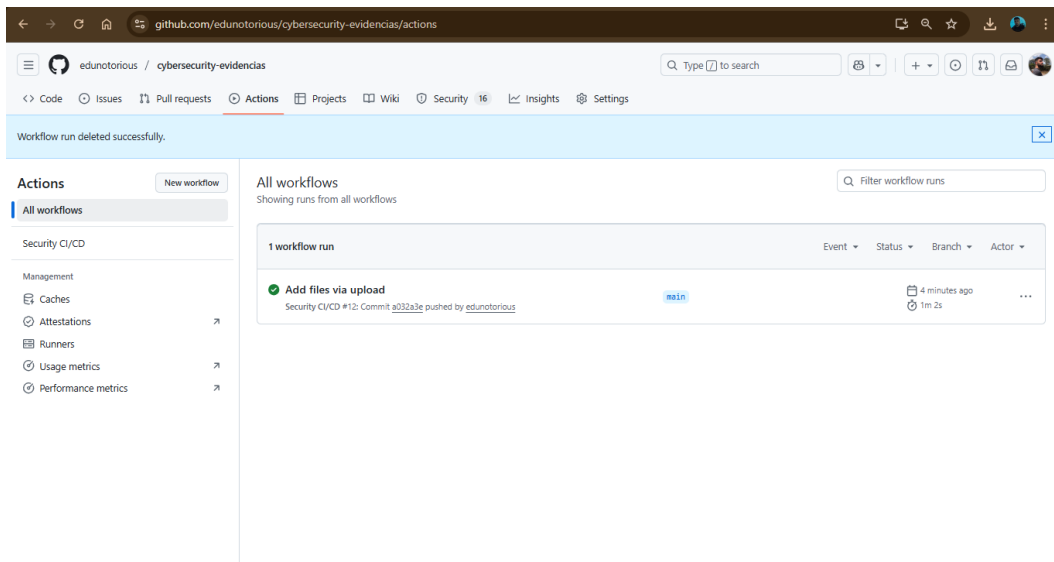


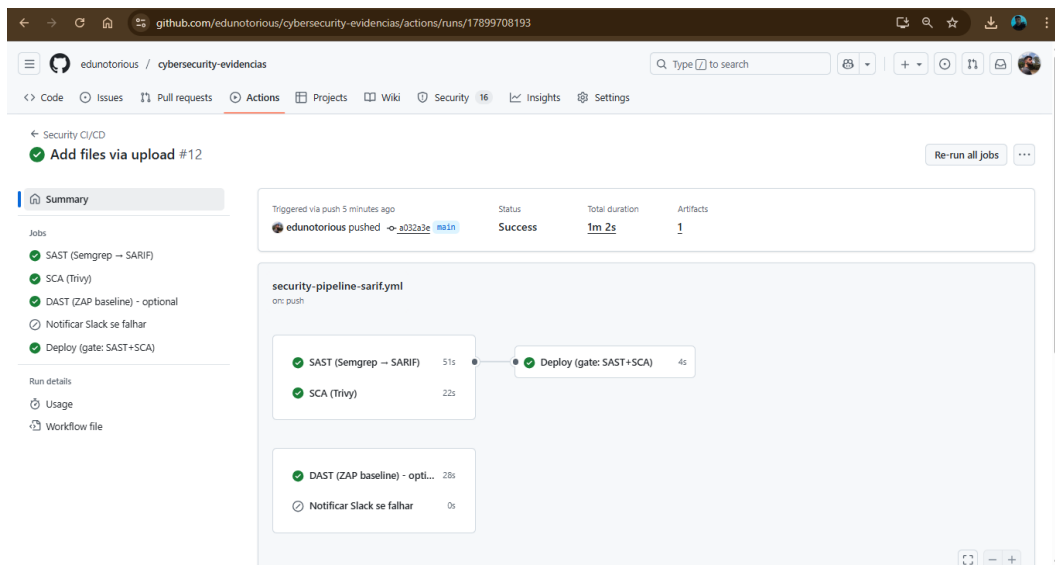
Foram identificadas dependências vulneráveis como 'lodash', 'body-parser' e 'express', com severidades variando entre LOW, MEDIUM, HIGH e CRITICAL.

Tarefa 4 - CI/CD Security Pipeline (GitHub Actions)

Na tarefa 4 foi implementado um pipeline de CI/CD de segurança no GitHub Actions. O pipeline executa SAST (Semgrep → SARIF), SCA (Trivy), DAST (ZAP baseline opcional) e realiza o deploy apenas se os testes passarem.

Evidências:





O pipeline foi executado com sucesso, integrando diferentes camadas de análise de segurança e garantindo que vulnerabilidades sejam detectadas antes do deploy.

Conclusão

As tarefas realizadas demonstraram a aplicação prática de ferramentas de segurança em diferentes níveis:

- SAST para análise estática do código (Semgrep);
- DAST para análise dinâmica da aplicação (OWASP ZAP);
- SCA para verificação de dependências (Dependency-Check);
- CI/CD Security Pipeline para automação e integração contínua de verificações.

Com isso, foi possível implementar um ciclo de desenvolvimento mais seguro, identificando e mitigando riscos potenciais.