IT Security

Andrea Ida Malkah Klaura, MA BSc
1710303005 / ic17m005

SCRAP

Secure Code Review Automated Platform.

Prototype of an automated feedback platform to provide secure coding feedback for introductory programming courses

Story Points

1. Problem context
2. Aim and approach
3. Results of the review
4. The prototype
5. Conclusion

The monkey factor

The prototyping librarian

L-space findings

Raising code

The next level

Website (incl. presentation): https://scrap.tantemalkah.at

1. The monkey factor

Establishing the problem context

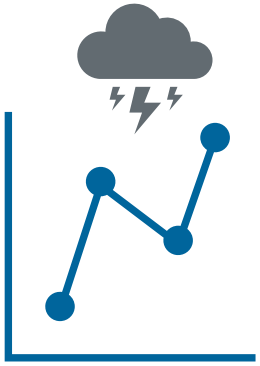
From the reports

- Web app attacks:
  - are primary cause of breaches
  - are main problem on network perimeter
    - 75% of penetration vectors caused by poor protection of web resources
- Over 85% of apps have at least one vulnerability
- Over 13% at least one critical severity flaw

OWASP Top 10

Veracode State of Software Security

CWE/SANS Top 25



Positive Technologies: Web Application Vulnerabilities Statistics

F5 Application Protection Report

Improving overall code security

Improving overall code security

Cannot be done

even with an

Army of Code Monkeys

(also not with ever more expensive and awesomely improved CI/CD pipelines)

Improving overall code security

Cannot be done

even with an

Army of Code Monkeys

(also not with ever more expensive and awesomely improved CI/CD pipelines)

We need

secure coding aware

Developers

who have evolved to at least the level of secure coding acolytes (apes or higher primates)

Teaching (secure) coding

Teaching (secure) coding

- Learning to code is hard
- Teaching students to code is hard
- Teaching them to code securely? **Try harder!**
- Introductory programming courses:
  - Teaching too much in too little time

2. The prototyping librarian

Aim and approach of the research

Main research questions

Main research questions

- Can we build a toolchain of open source secure code analysis tools, to analyse programming exercises and to automatically create qualitative feedback?

Main research questions

- Can we build a toolchain of open source secure code analysis tools, to analyse programming exercises and to automatically create qualitative feedback?
- What are the contextual (that is, organisational and educational) requirements for applying such an approach to gain secure coding awareness among students?

Research Approach

1. Map the research field
2. Evaluate analysis tools
3. Create and evaluate a prototype

3. L-space findings

Results of the review

Code vulnerabilities

- The human factor in coding: developers
- SQL injections still most prevalent in PHP based F/LOSS projects (Schuckert et al., 2017)

Static analysis tools

- PHP static code analysis:
  - many F/LOSS tools but most are abandoned
- 7 currently maintained tools for easy integration
- most promising:
  - PHP_CodeSniffer + php_security_audit
  - SonarQube
  - PHP Malware Finder's YARA rules

Software security education

- No comprehensive overview of the field
- Several studies and projects out there
  - … until funding ends
- Most recent and promising:
  - Security Injections @ Towson (apparent end: 2017)

4. Raising Code

The prototype

Key requirements

- Web service with well documented API
- Additional lightweight web client
- Good documentation (SCRAP != scrap)
- Modular and adaptable design, to:
  - integrate different programming languages
  - and scanner rules and explanations

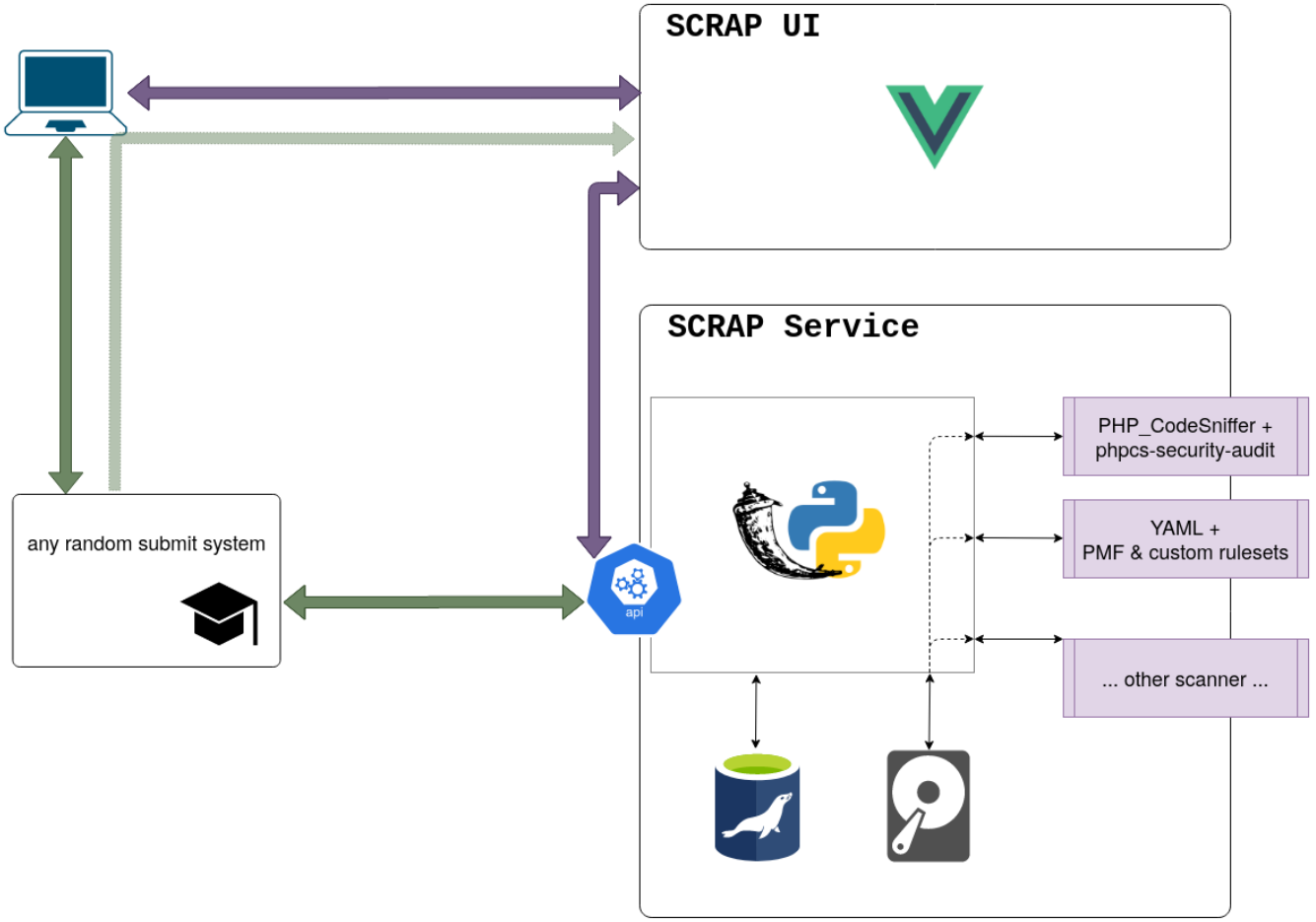
All the docs you need

https://scrap.tantemalkah.at

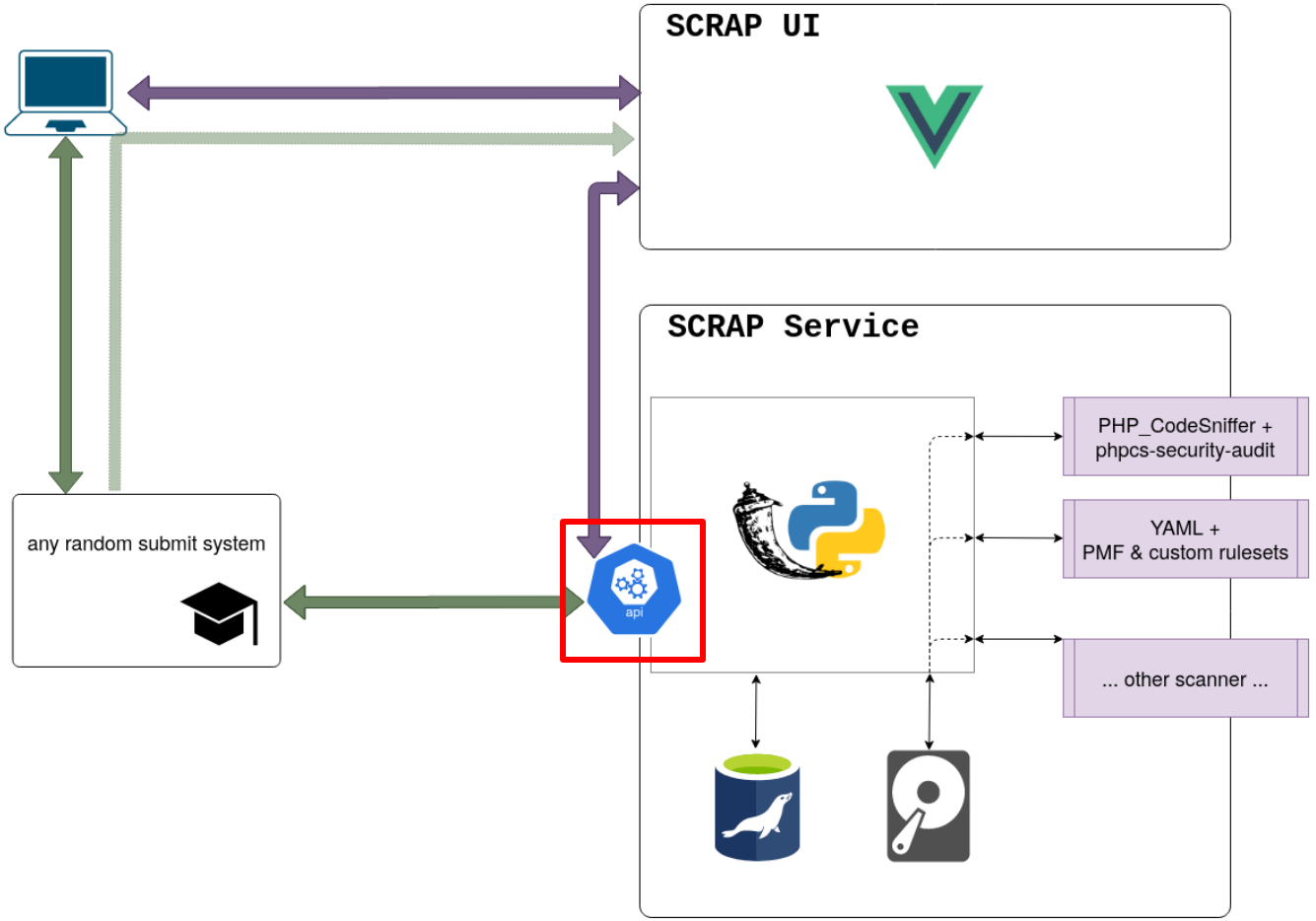
Includes:

- Links to component repos
  - including the scanner analysis
- Public demo
- Background and usage info

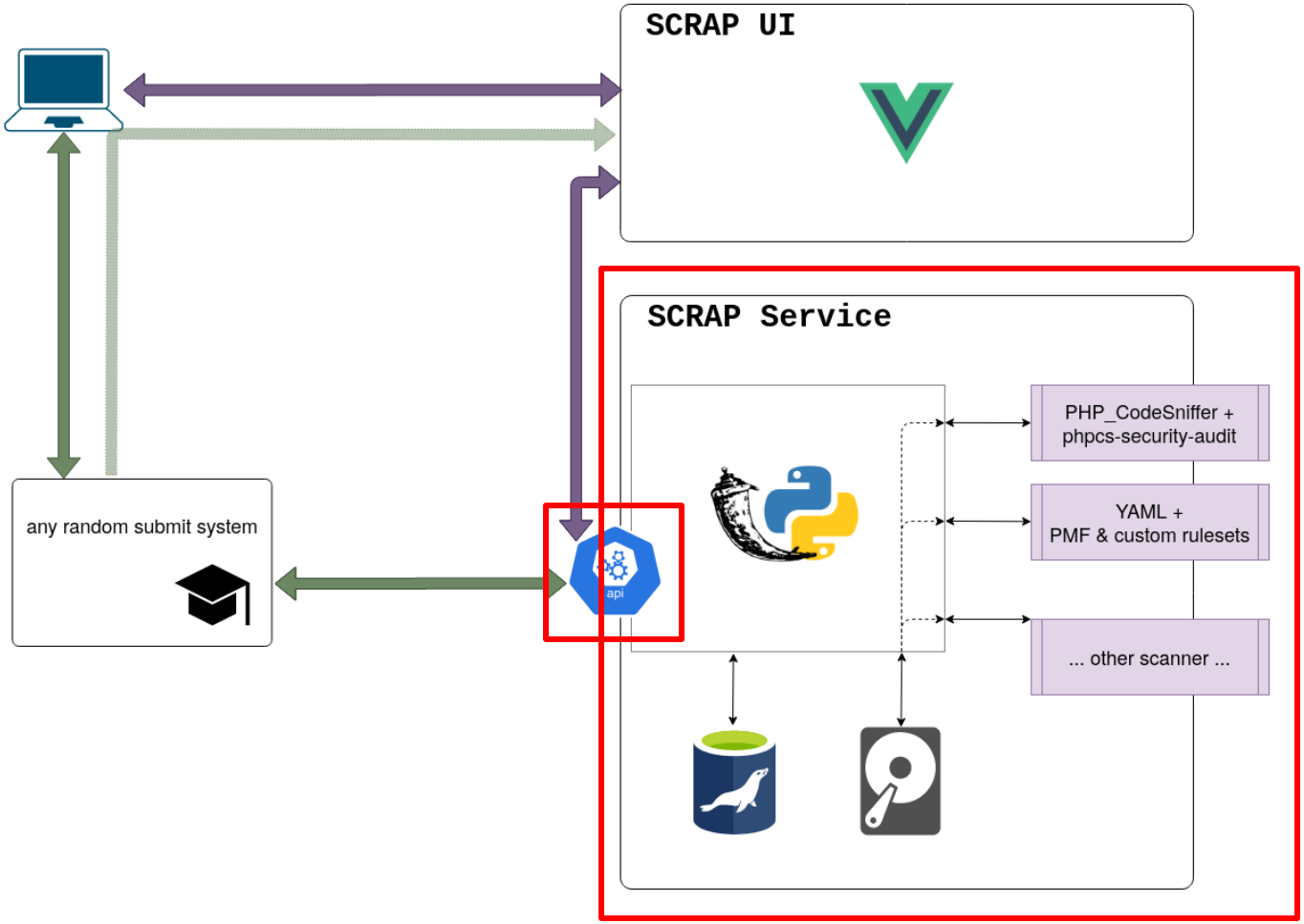
SCRAP Architecture



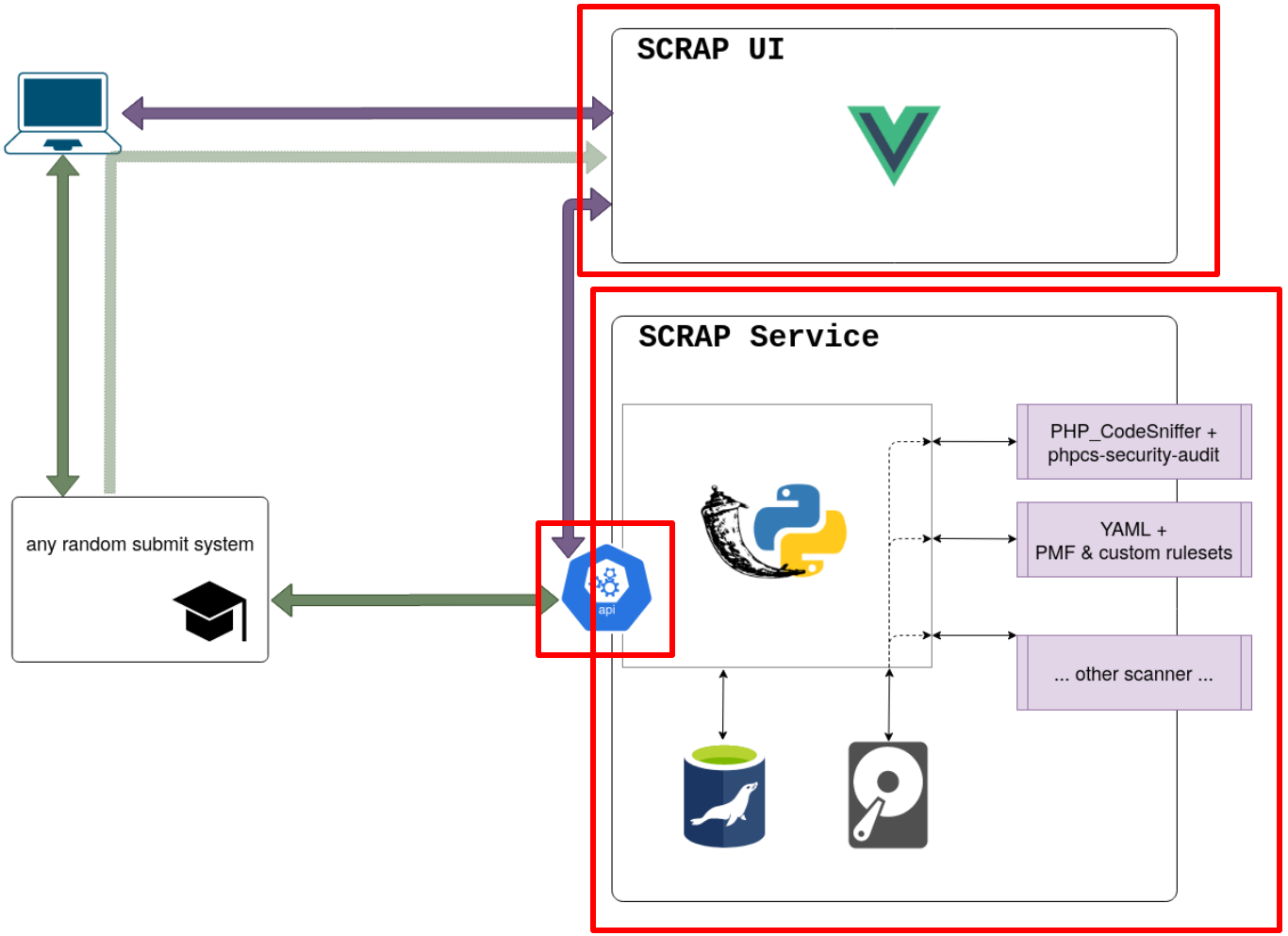
SCRAP Architecture



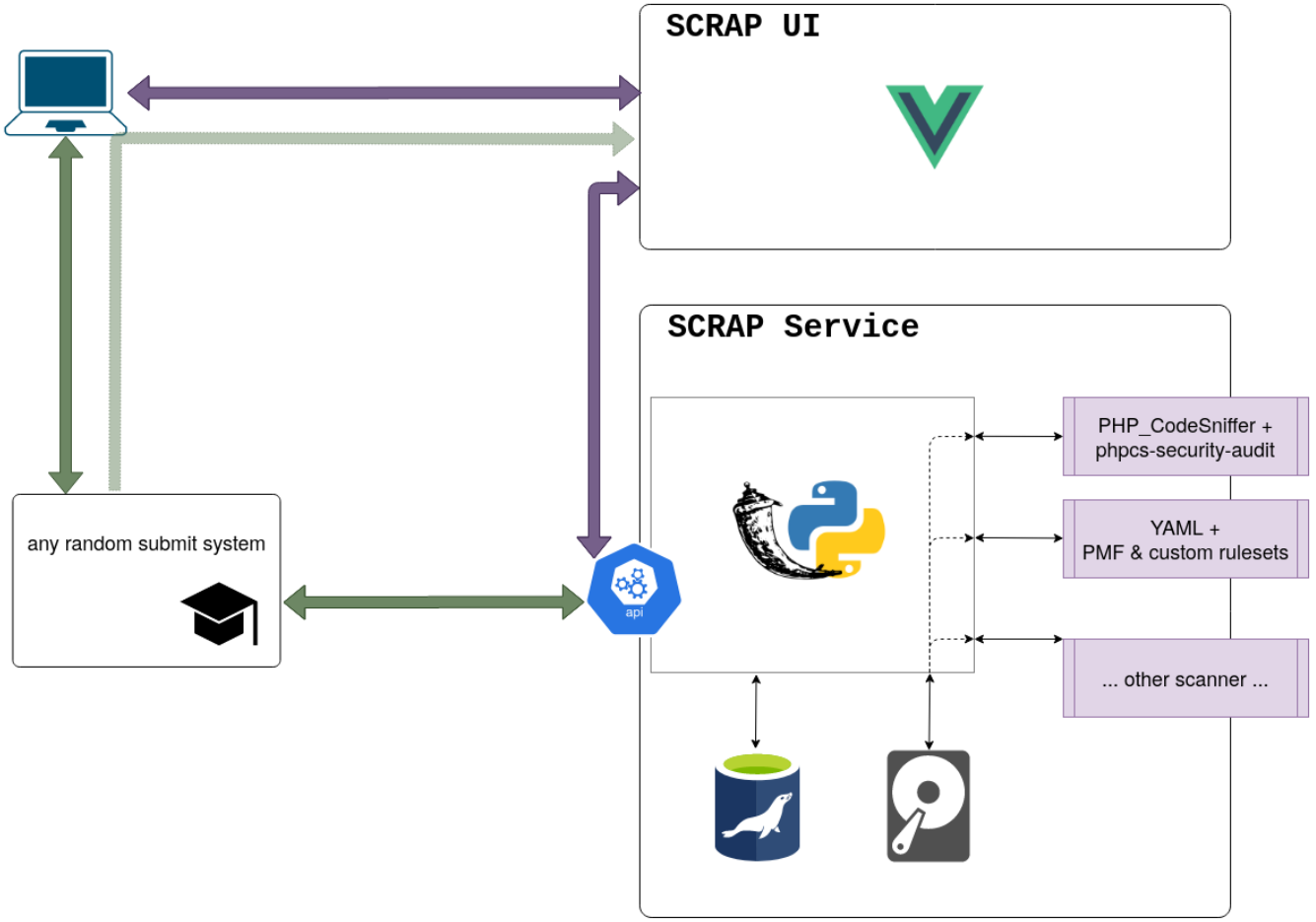
SCRAP Architecture



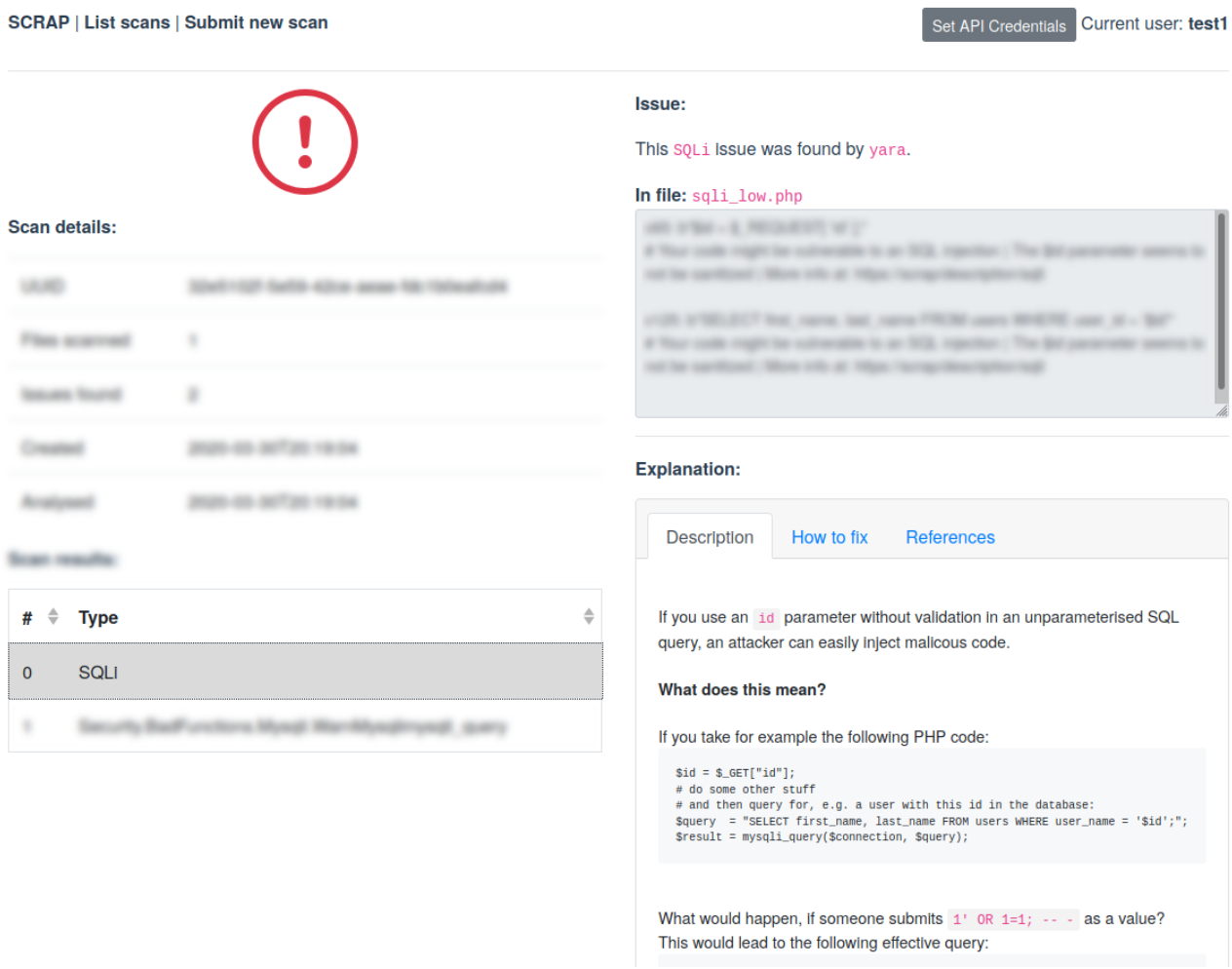
SCRAP Architecture



SCRAP Architecture



SCRAP UI: Explanations



5. The Next Level

Conclusion and future research

Feasibility of the tool chain

- Yes, we can use F/LOSS tools and integrate them rather easily
- No, they do not provide enough context as-is
- Major work needed to write scanner rules and explanations
- New scanner integration just takes ~1 programmer day

Feasibility of the automated feedback approach

- Just adding SCRAP does not solve a lot
- Non-technical adaptations needed
  - Policies (e.g. coding standards)
  - Educational strategies
  - Didactic approaches

Where to go from here

- SCRAP could be extended
- But so could SonarQube

Sustainability is hard to achieve when everyone is trying to solve their problem on their own and has to compete for limited funding

Level 2: Evaluation in introductory programming course

Andrea Ida Malkah Klaura, MA BSc
1710303005 / ic17m005

SCRAP

Secure Code Review Automated Platform.

https://scrap.tantemalkah.at