Выбираем open-source **SAST** для **Python** проектов

Максим Кобилев

27 сентября 2024 | PiterPy

цель?

$ whoami

$ whoami

- ИТ-лид в **SOLAR**

$ whoami

- ИТ-лид в **SOLAR**
- ~~разработчик~~

$ whoami

- ИТ-лид в **SOLAR**
- ~~разработчик~~
- ИБ практик / CTF player / org

$ whoami

- ИТ-лид в **SOLAR**
- ~~разработчик~~
- ~~ИБ практик / CTF player / org~~
- менеджер

$ whoami

- ИТ-лид в **SOLAR**
- ~~разработчик~~
- ~~ИБ практик / CTF player / org~~
- менеджер
- люблю ломать код

План доклада

План доклада

- Проблема

План доклада

- Проблема
- Обзор инструментов

План доклада

- Проблема
- Обзор инструментов
- Сравнение

План доклада

- Проблема
- Обзор инструментов
- Сравнение
- Выводы

План доклада

- Проблема
- Обзор инструментов
- Сравнение
- Выводы

* в конце ссылка на все ссылки

Кто такой этот ваш SAST?

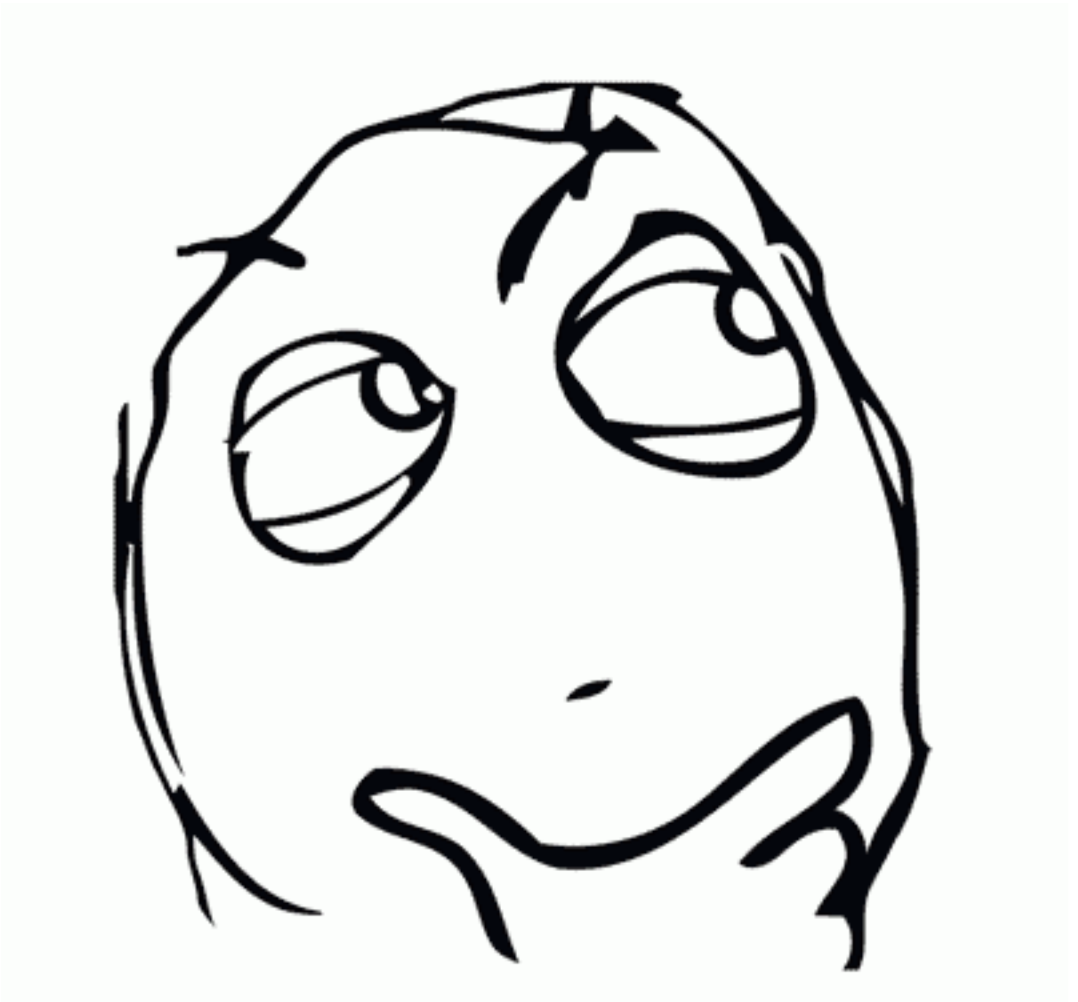
Кто знает что такое SAST?

Кто использует SAST?

Кто такой этот ваш SAST?

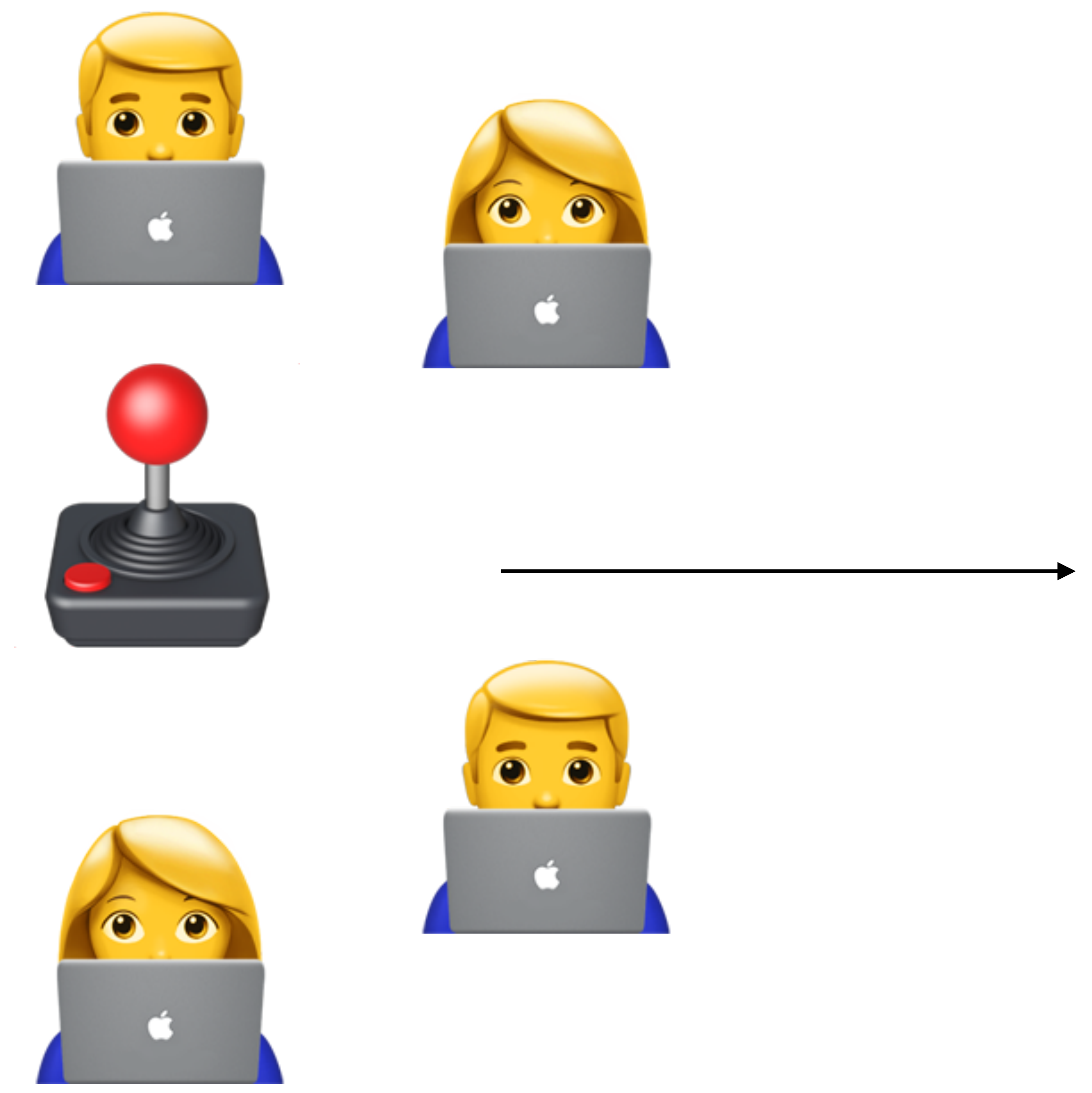
Static Application **Security** Testing

Проблема

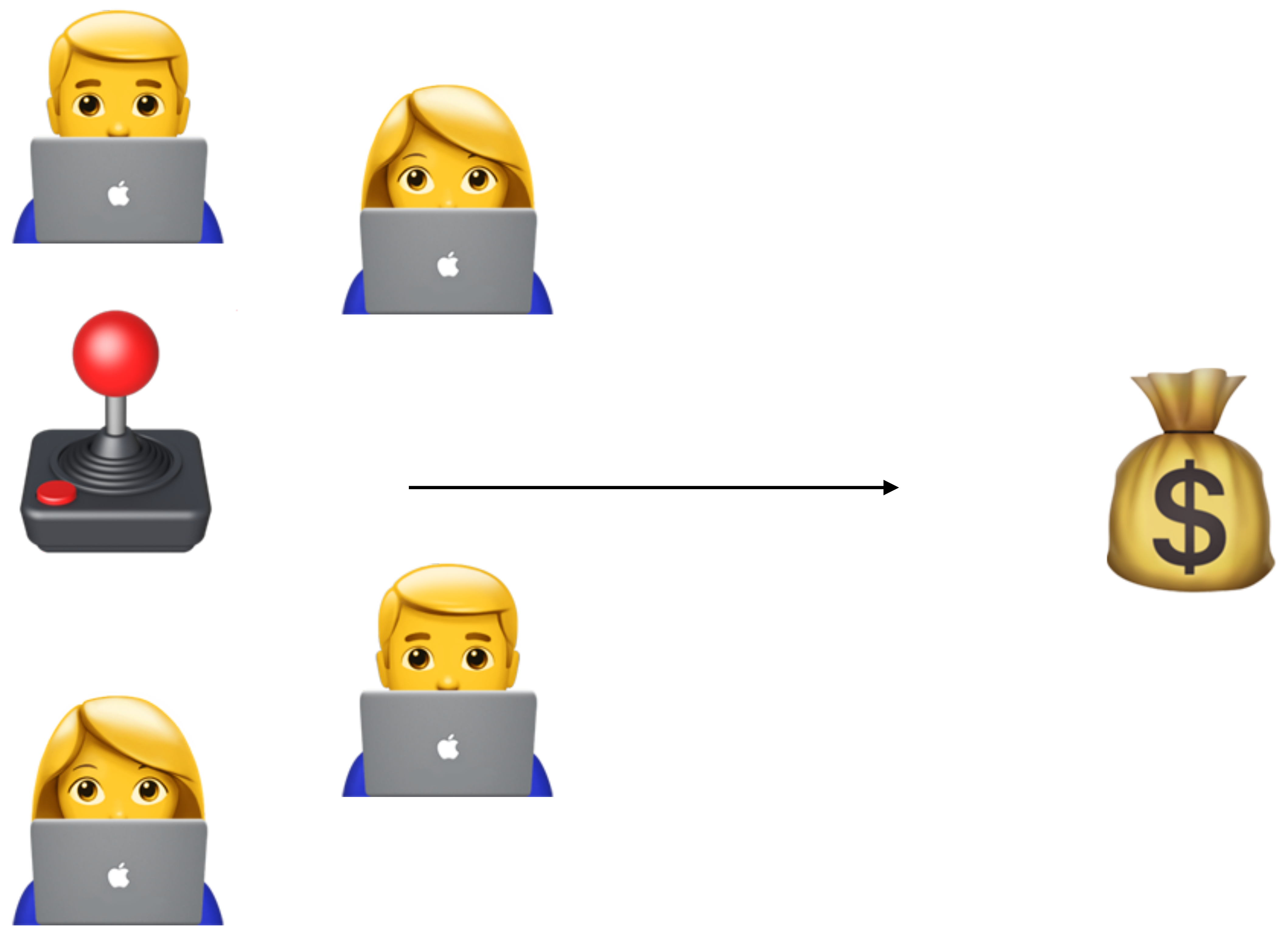


Жизненный цикл продукта

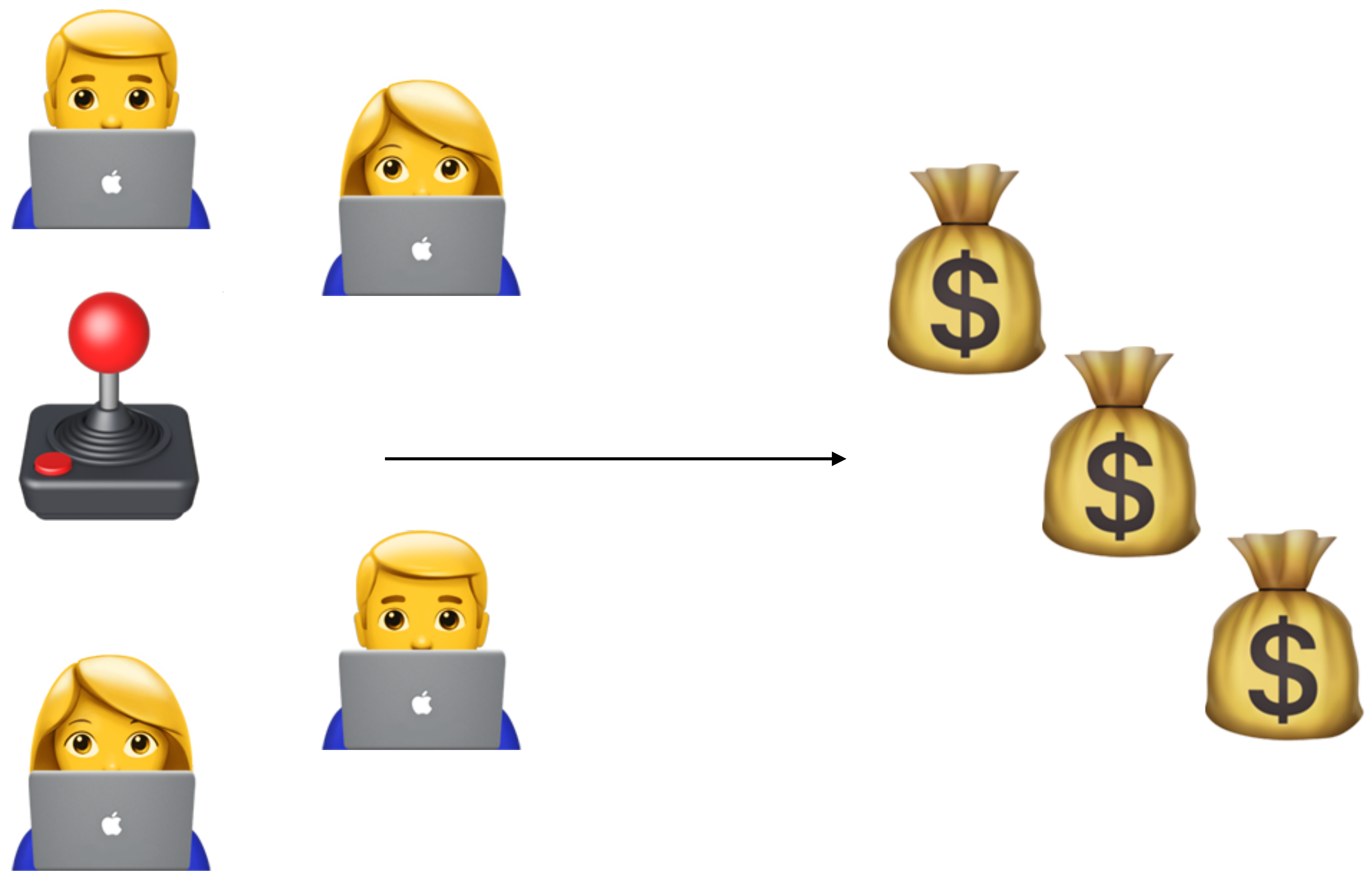
Жизненный цикл продукта



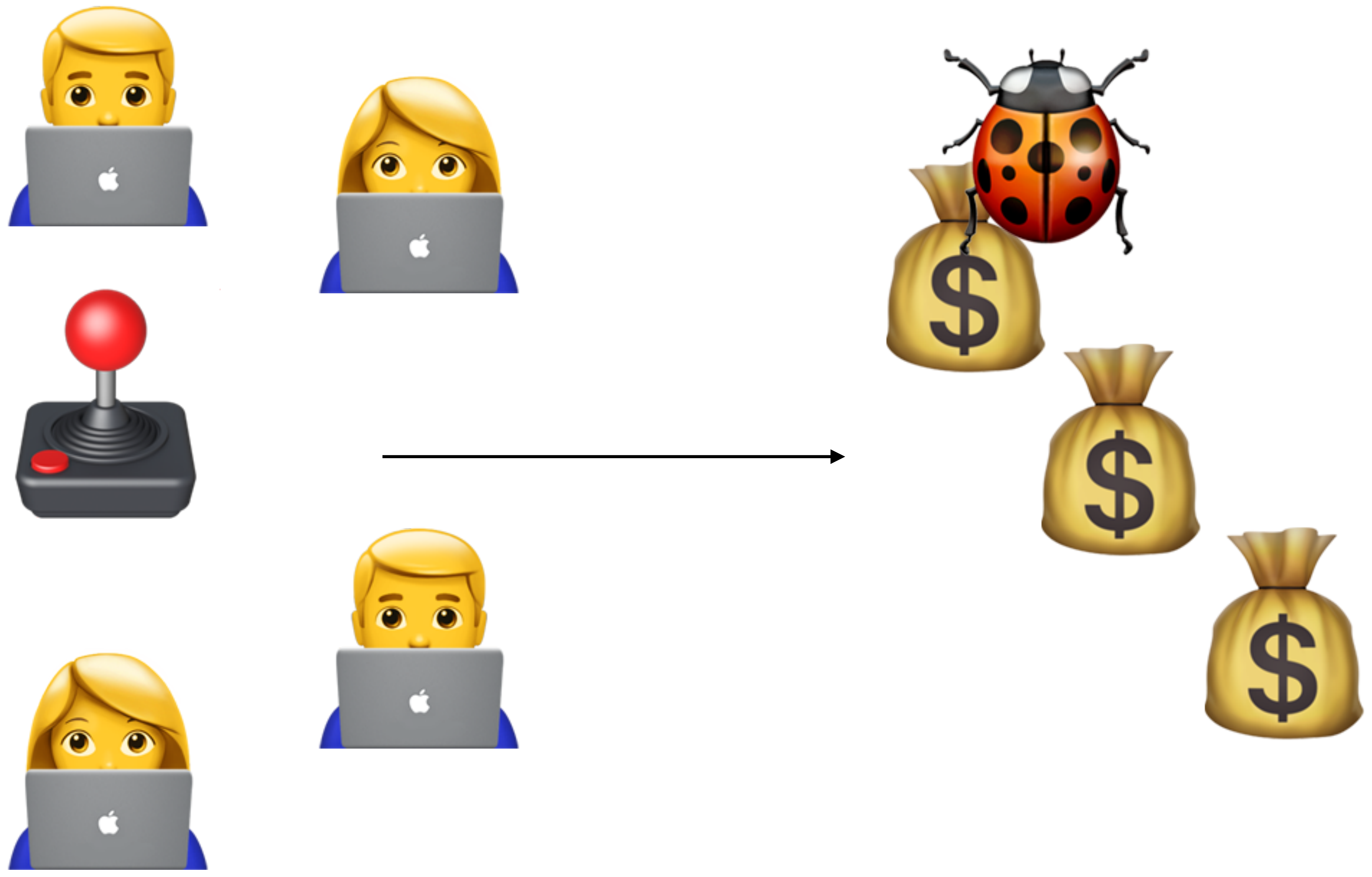
Жизненный цикл продукта



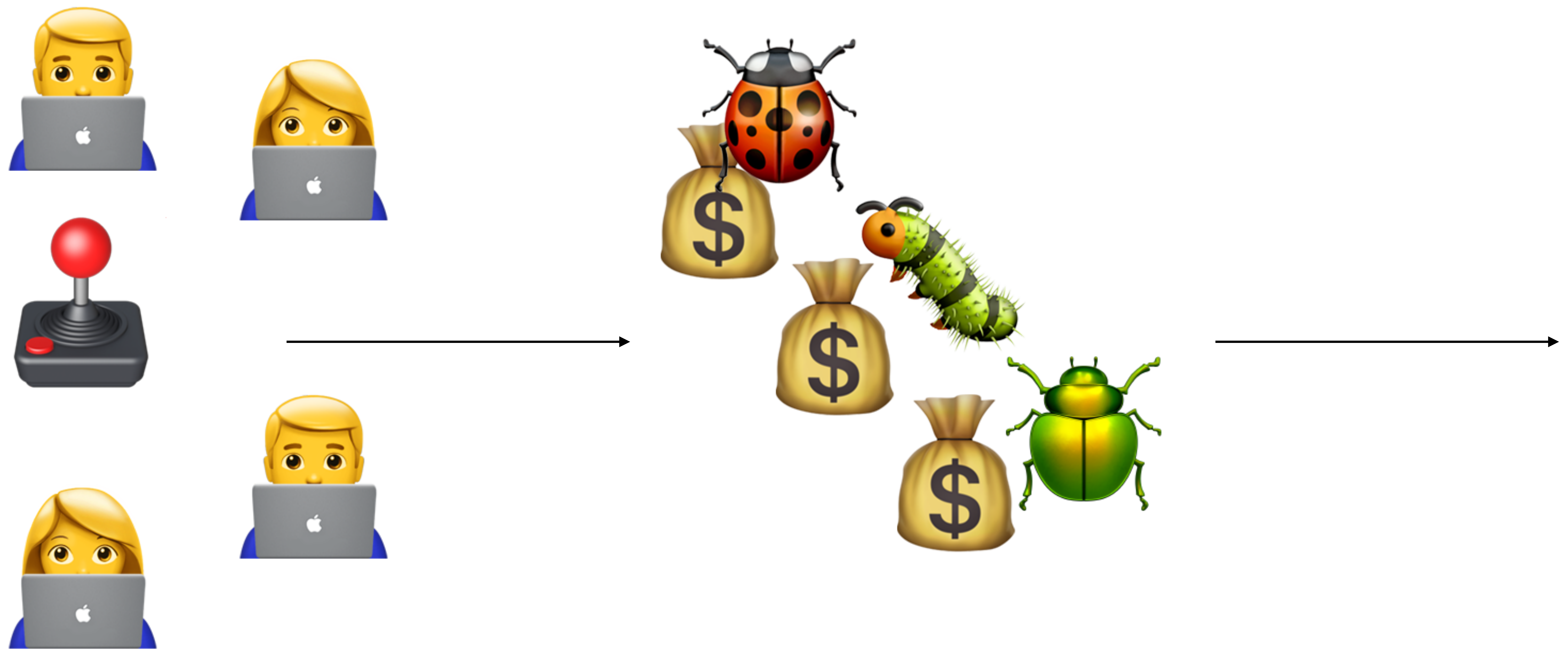
Жизненный цикл продукта



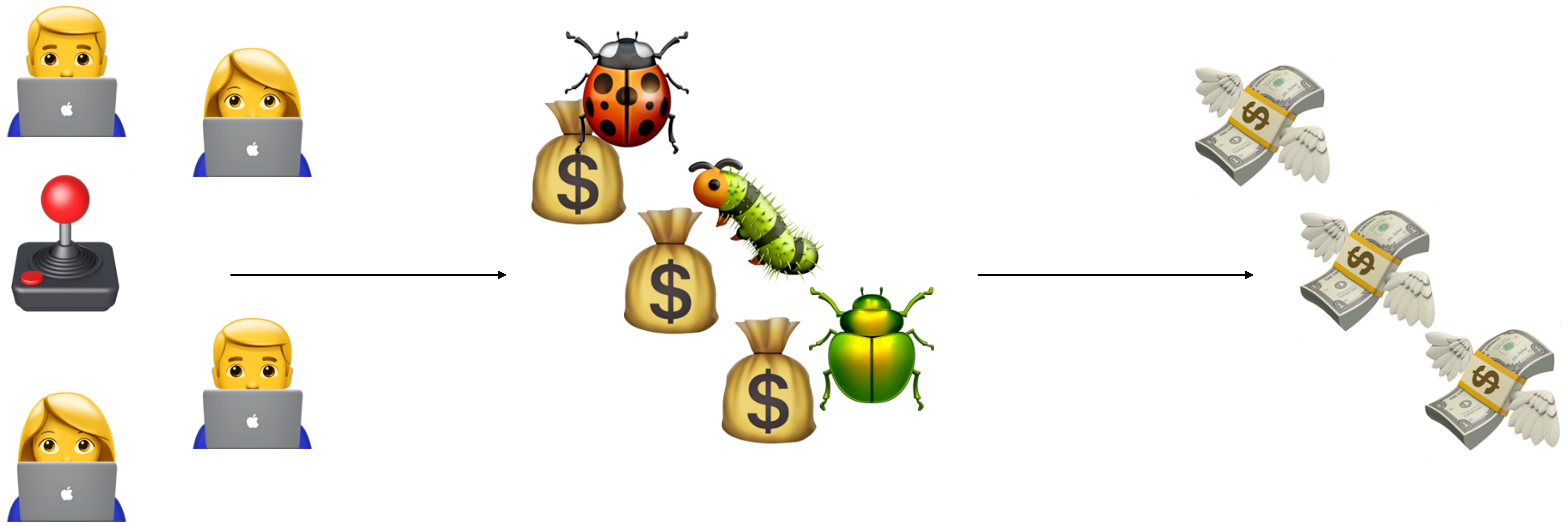
Жизненный цикл продукта



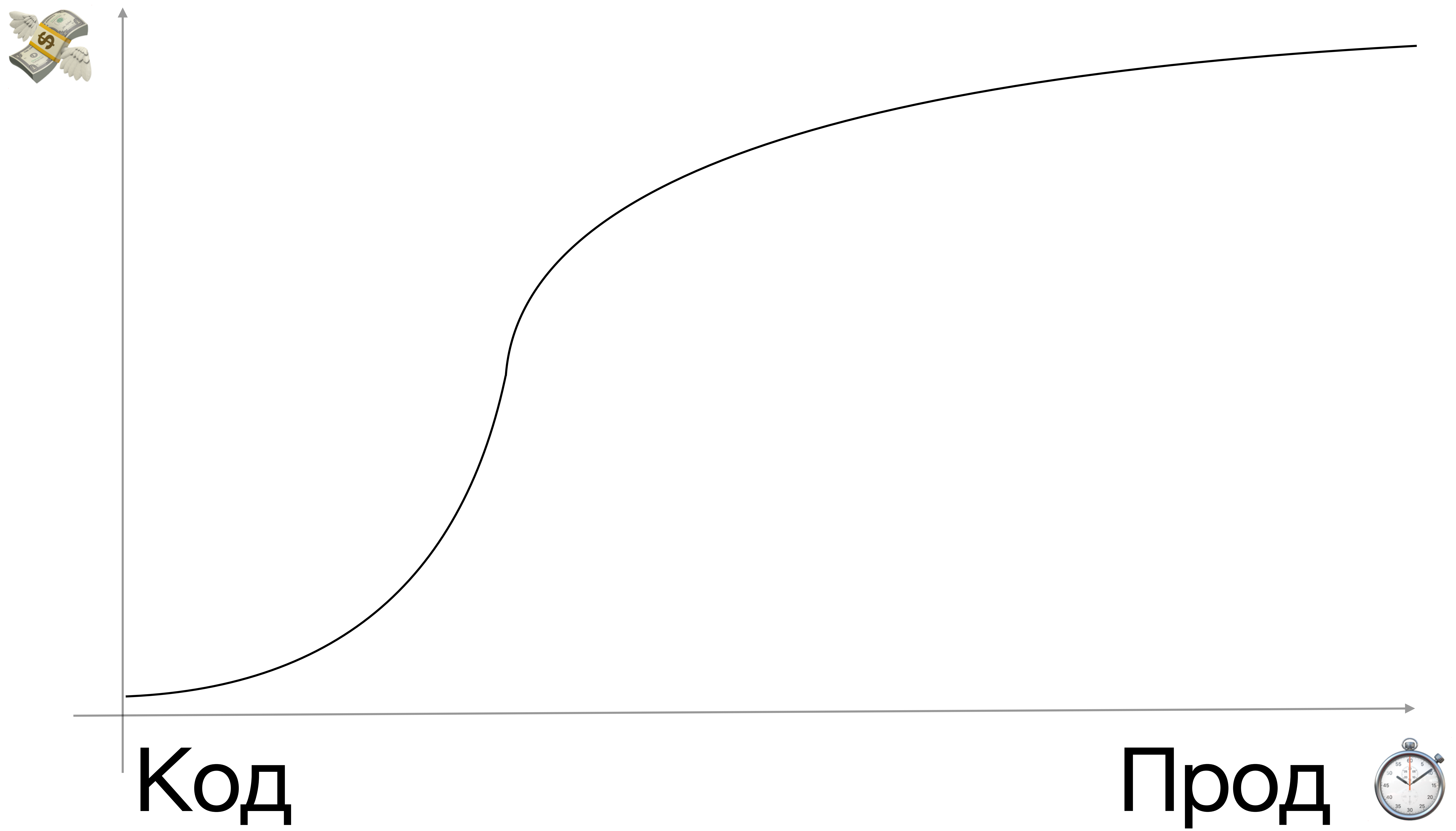
Жизненный цикл продукта



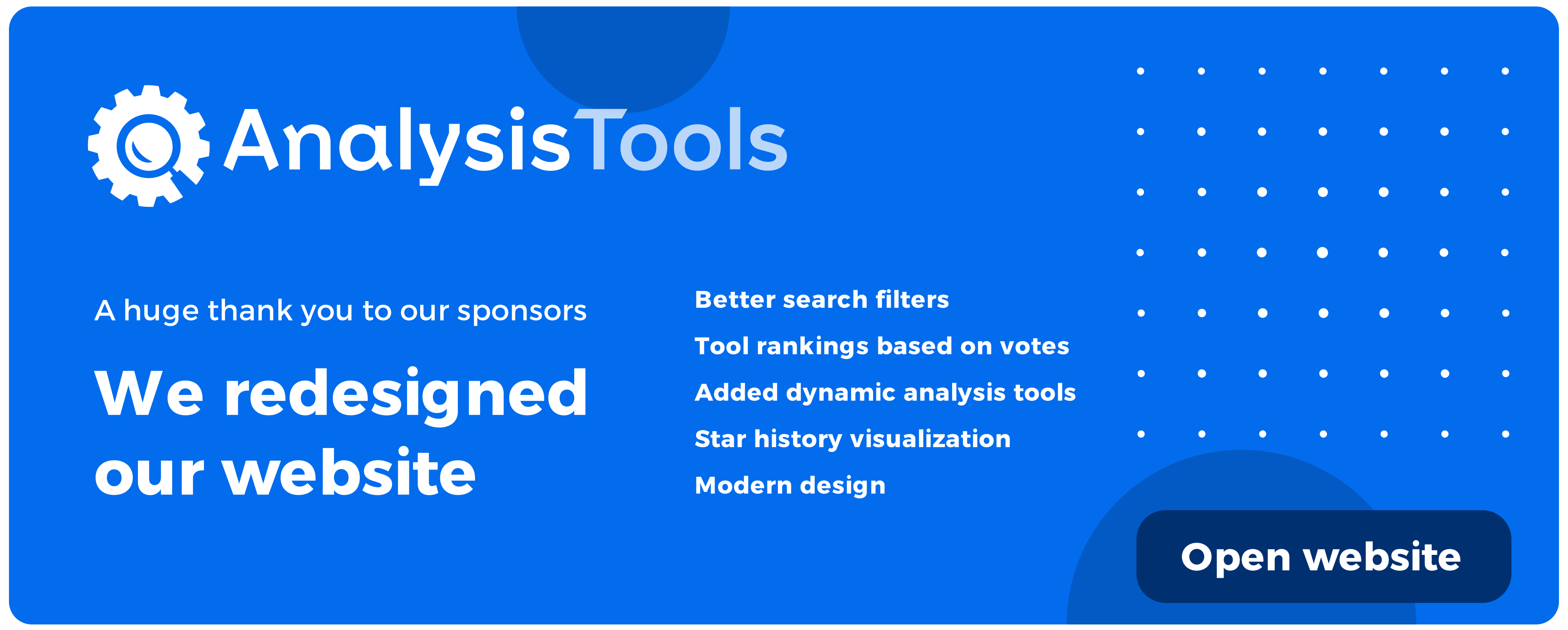
Жизненный цикл продукта



Важно находить уязвимости как можно раньше



1000 и 1 SAST


AnalysisTools

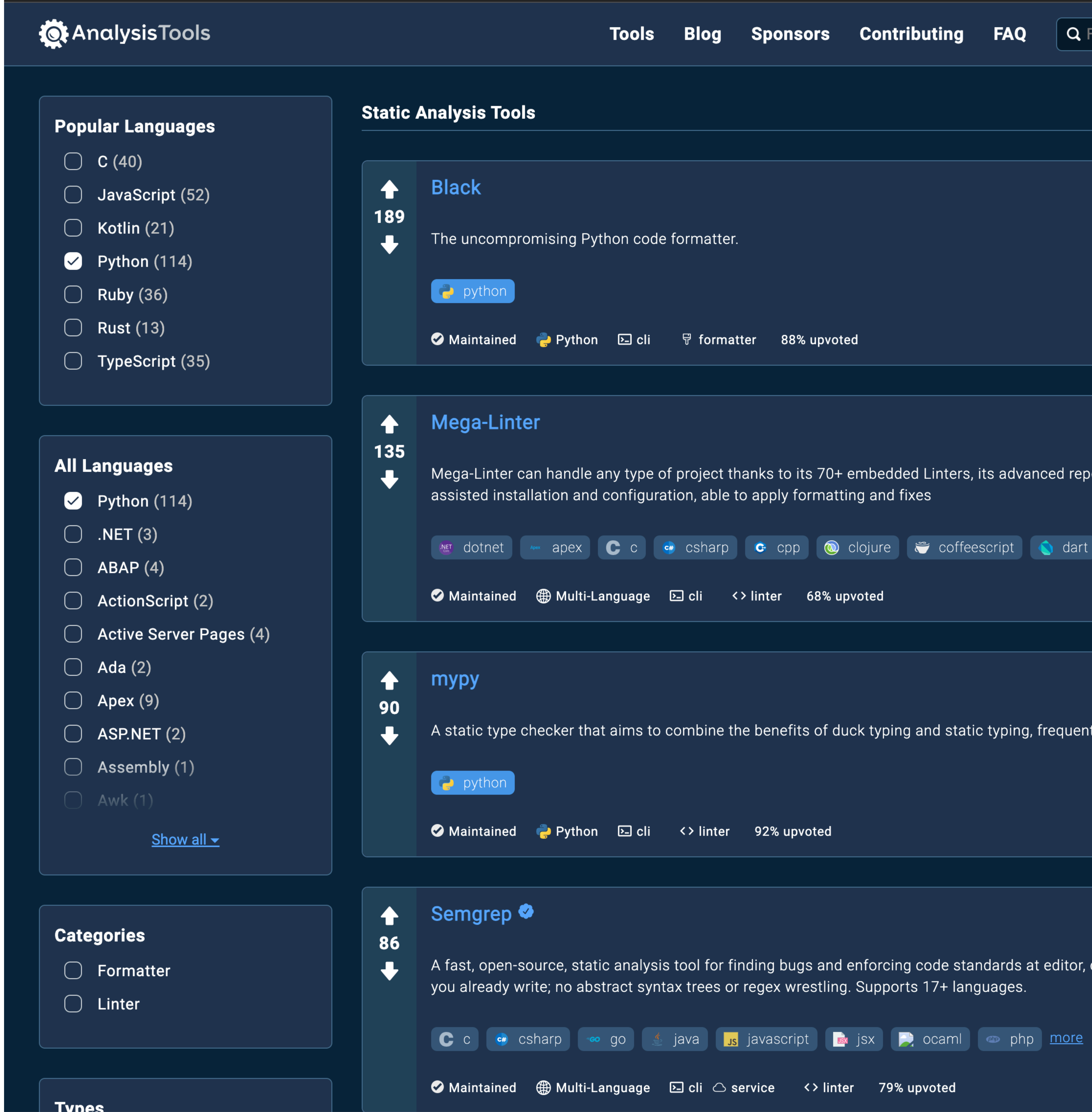
A huge thank you to our sponsors

We redesigned our website

Better search filters
Tool rankings based on votes
Added dynamic analysis tools
Star history visualization
Modern design

Open website

https://analysis-tools.dev



Popular Languages

- C (40)
- JavaScript (52)
- Kotlin (21)
- ✓ Python (114)
- Ruby (36)
- Rust (13)
- TypeScript (35)

All Languages

- ✓ Python (114)
- .NET (3)
- ABAP (4)
- ActionScript (2)
- Active Server Pages (4)
- Ada (2)
- Apex (9)
- ASP.NET (2)
- Assembly (1)
- Awk (1)

Show all

Categories

- Formatter
- Linter

Types

Static Analysis Tools

189 **Black**
The uncompromising Python code formatter.
python
Maintained · Python · cli · formatter · 88% upvoted

135 **Mega-Linter**
Mega-Linter can handle any type of project thanks to its 70+ embedded Linters, its advanced rep
assisted installation and configuration, able to apply formatting and fixes
dotnet · apex · c · csharp · cpp · clojure · coffeescript · dart
Maintained · Multi-Language · cli · linter · 68% upvoted

90 **mypy**
A static type checker that aims to combine the benefits of duck typing and static typing, frequent
python
Maintained · Python · cli · linter · 92% upvoted

86 **Semgrep**
A fast, open-source, static analysis tool for finding bugs and enforcing code standards at editor,
you already write; no abstract syntax trees or regex wrestling. Supports 17+ languages.
c · csharp · go · java · javascript · jsx · ocaml · php · more
Maintained · Multi-Language · cli · service · linter · 79% upvoted

OWASP: Source Code Analysis Tools

OWASP

Open Web Application Security Project

21 про security

https://owasp.org/www-community/Source_Code_Analysis_Tools

Какие бывают анализаторы?

Какие бывают анализаторы?

Линтеры

Какие бывают анализаторы?

Линтеры **/ Flake**

Какие бывают анализаторы?

Линтеры / **Flake**

Форматтеры

Какие бывают анализаторы?

Линтеры **/ Flake**

Форматтеры **/ Black**

Какие бывают анализаторы?

Линтеры / **Flake**

Форматтеры / **Black**

Проверка аннотаций типов

Какие бывают анализаторы?

Линтеры / **Flake**

Форматтеры / **Black**

Проверка аннотаций типов / **MyPy**

Какие бывают анализаторы?

Линтеры / **Flake**

Форматтеры / **Black**

Проверка аннотаций типов / **MyPy**

Статический анализ безопасности кода

Какие бывают анализаторы?

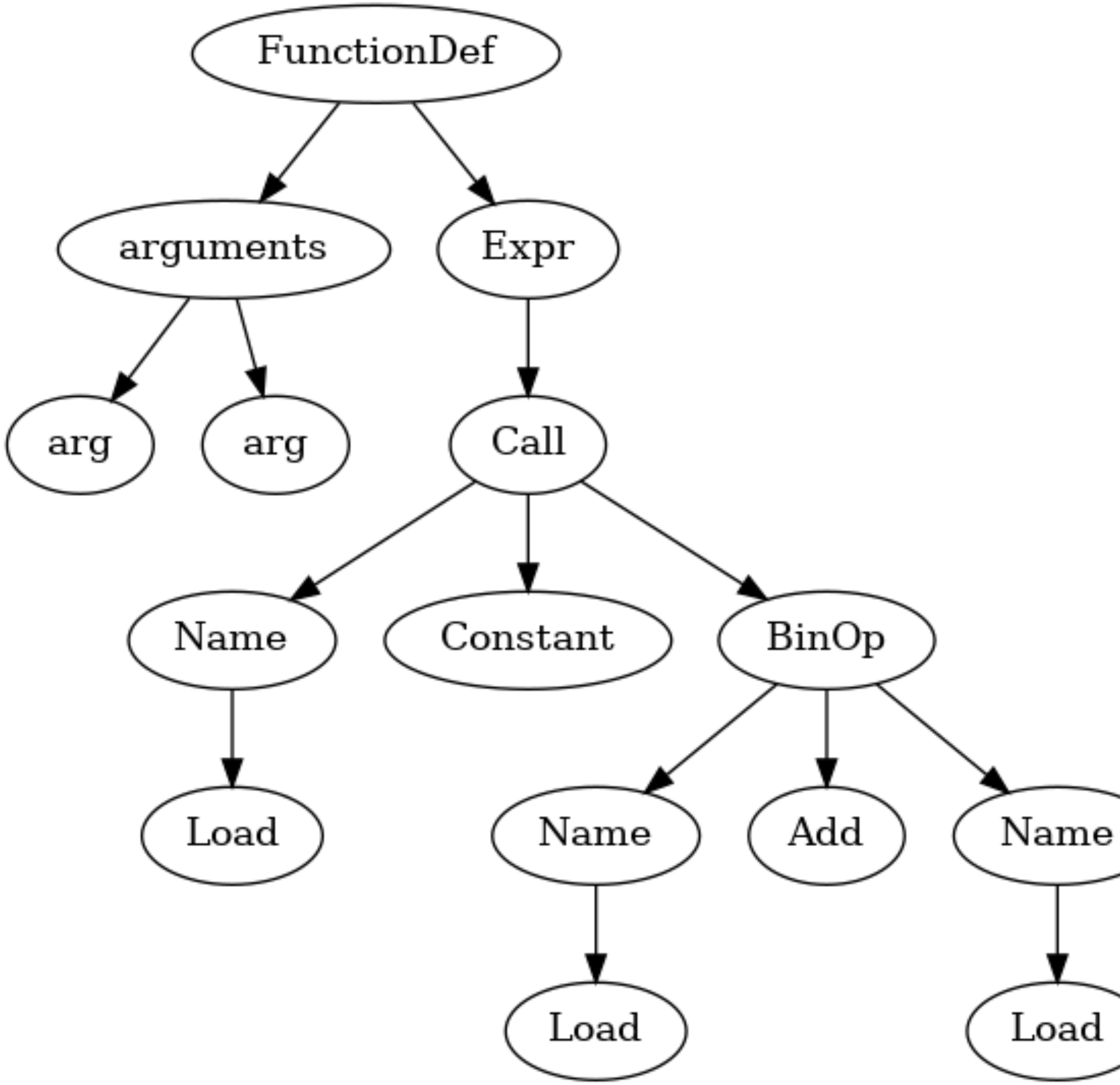
Линтеры / **Flake**

Форматтеры / **Black**

Проверка аннотаций типов / **MyPy**

Статический анализ безопасности кода / **Bandit**

Как работает SAST?

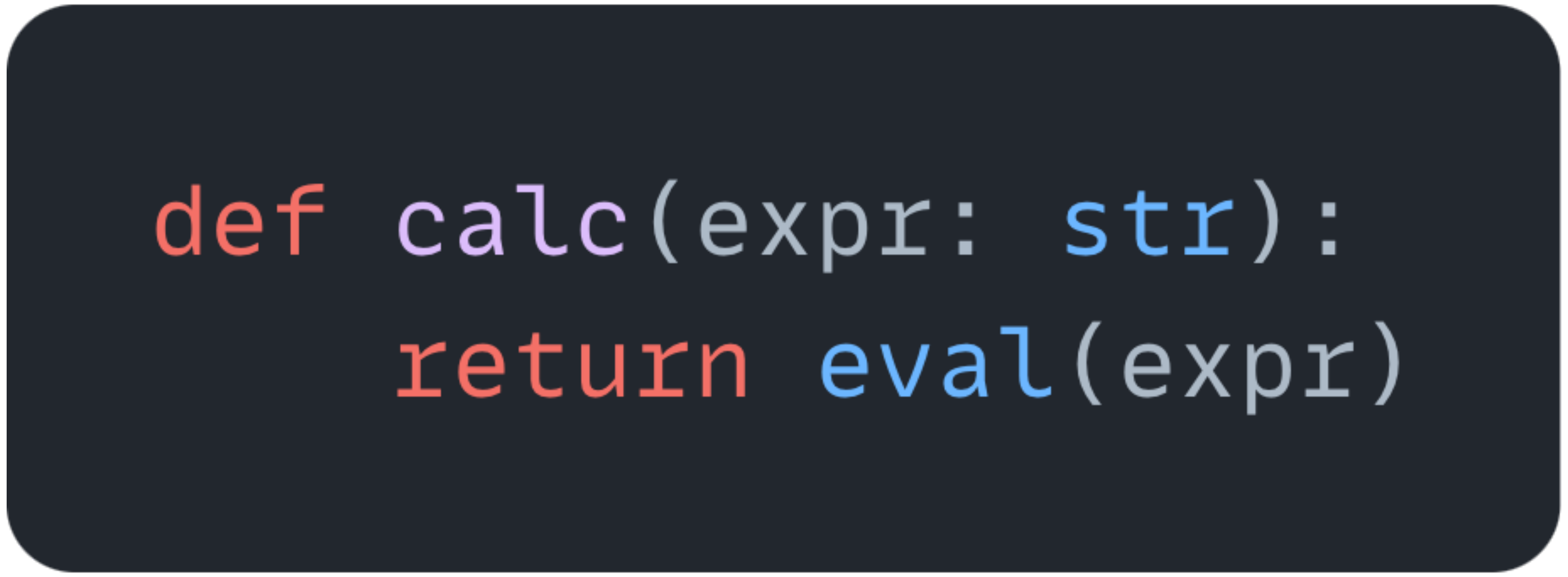


```python
def calc(expr: str):
    return eval(expr)
```

```python
def calc(expr: str):
    return eval(expr)
```

pattern matching

grep eval



```python
def calc(expr: str):
    return eval(expr)
```

regex

\b(eval)\s*\(

```python
def calc(expr: str):
    return eval(expr)
```

AST
CFG
Data-flow
Taint analysis

Taint-анализ

```python
@app.get('/select')
async def sql_return_users_from_username(username: str):
    resp = await run_sql_query(f'SELECT * FROM users WHERE username = "{username}";')
    return resp
```

Taint-анализ: Taint

Tainted Data (загрязненные данные): Данные из ненадежных источников

```
@app.get('/select')
async def sql_return_users_from_username(username: str):
    resp = await run_sql_query(f'SELECT * FROM users WHERE username = "{username}";')
    return resp
```

Taint-анализ: Propagation

Taint Propagation: Процесс, при котором загрязненные данные передаются через функции, методы или переменные.

```
@app.get('/select')
async def sql_return_users_from_username(username: str):
    resp = await run_sql_query(f'SELECT * FROM users WHERE username = "{username}";')
    return resp
```

Taint-анализ: Sinks

Taint Sinks: Места в коде, где загрязненные данные могут привести к уязвимостям

```python
@app.get('/select')
async def sql_return_users_from_username(username: str):
    resp = await run_sql_query(f'SELECT * FROM users WHERE username = "{username}";')
    return resp
```

DEVELOPMENT

Один анализатор, чтобы управлять ими всеми

ВЛАДИМИР КОЧЕТКОВ

Бенчмарк: история

Бенчмарк: OWASP TOP 10

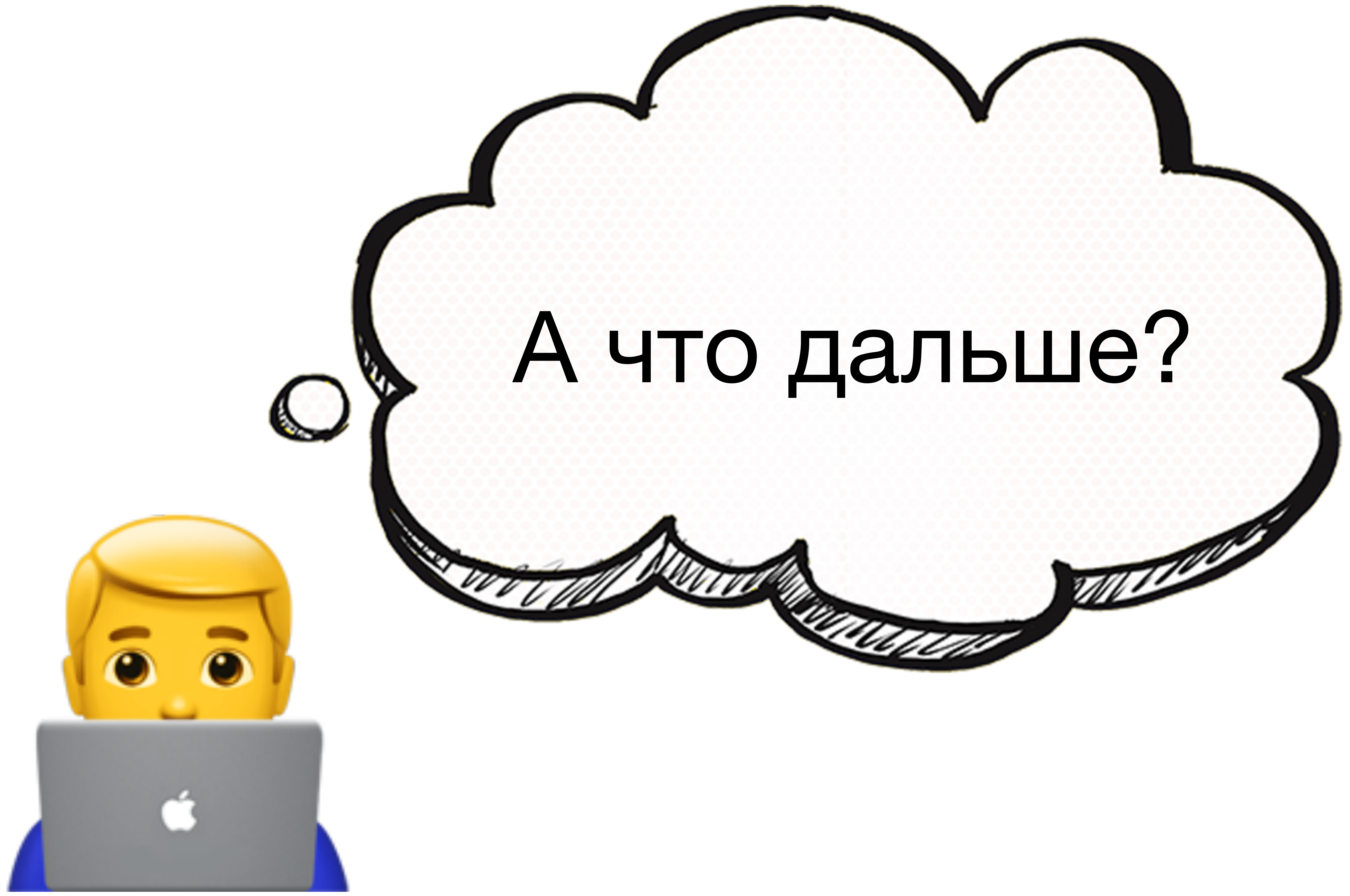
Бенчмарк: OWASP TOP 10

1. JWT Auth - CWE-347
2. LFI - CWE-98
3. NoSQL injection - CWE-134
4. SQL injection - CWE-89
5. Pickle RCE - CWE-502
6. Exec RCE - CWE-78
7. Eval RCE - CWE-95
8. SSTI - CWE-95
9. XXE - CWE-611
10. Off by Slash - CWE-22

Синтетическое приложение для оценки

python-sec-101

/mkobilev/python-sec-101


А что дальше?

Bandit

Bandit is a tool designed to find common security issues in Python code.

/PyCQA/bandit

https://bandit.readthedocs.io

Bandit: установка

```
pip install bandit
```

Bandit: запуск

text

json

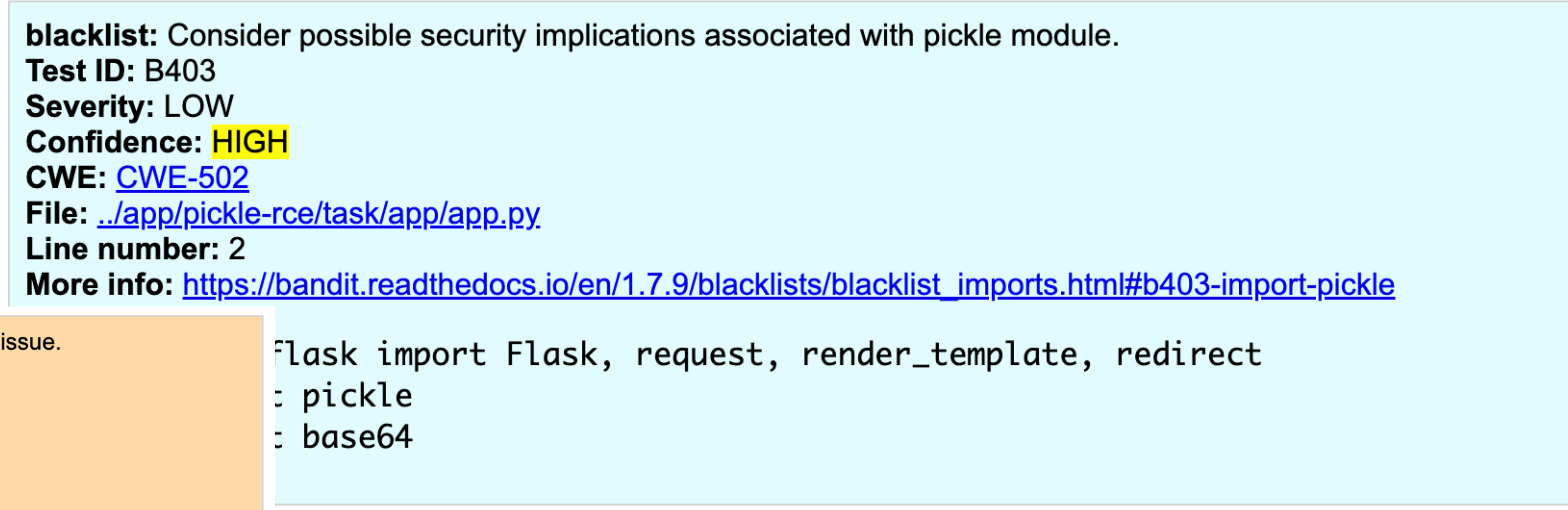
csv

```
bandit -r ../app -f html -o report.html
```

xml

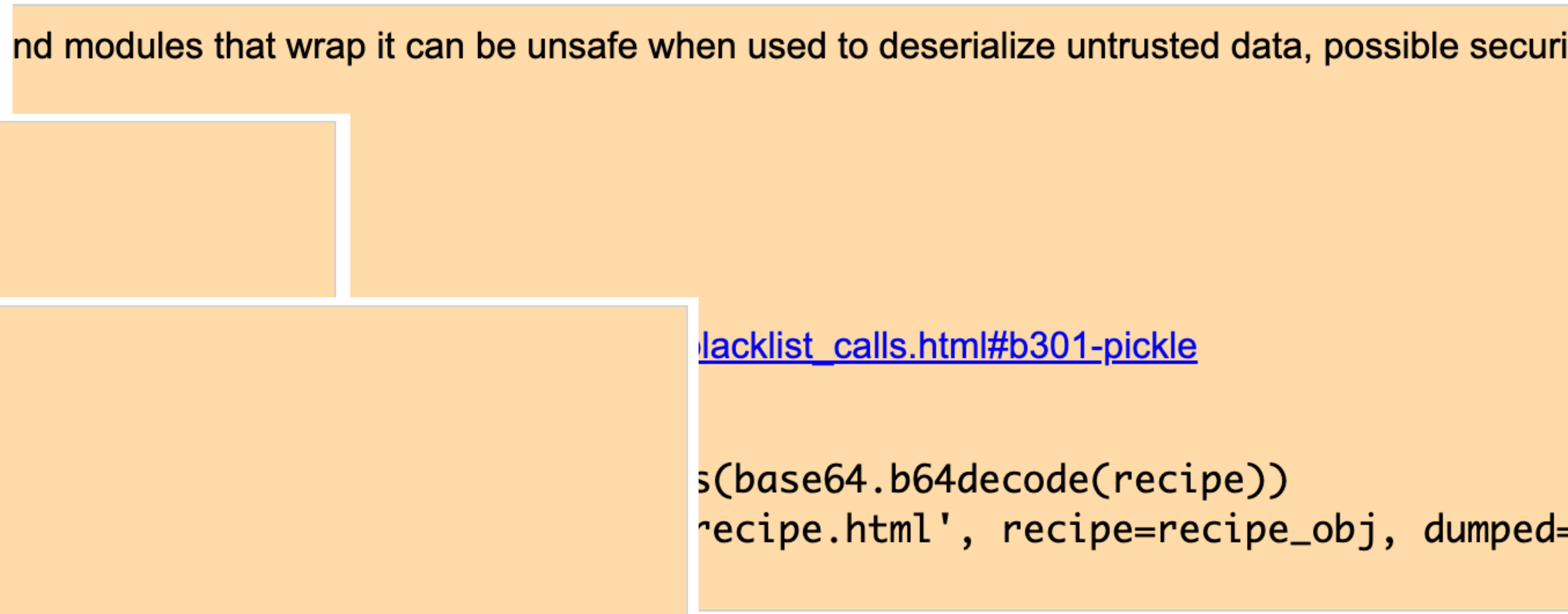
yaml

Bandit: результаты



blacklist: Consider possible security implications associated with pickle module.
Test ID: B403
Severity: LOW
Confidence: HIGH
CWE: CWE-502
File: ../app/pickle-rce/task/app/app.py
Line number: 2
More info: https://bandit.readthedocs.io/en/1.7.9/blacklists/blacklist_imports.html#b403-import-pickle

```
flask import Flask, request, render_template, redirect
: pickle
: base64
```



nd modules that wrap it can be unsafe when used to deserialize untrusted data, possible securi

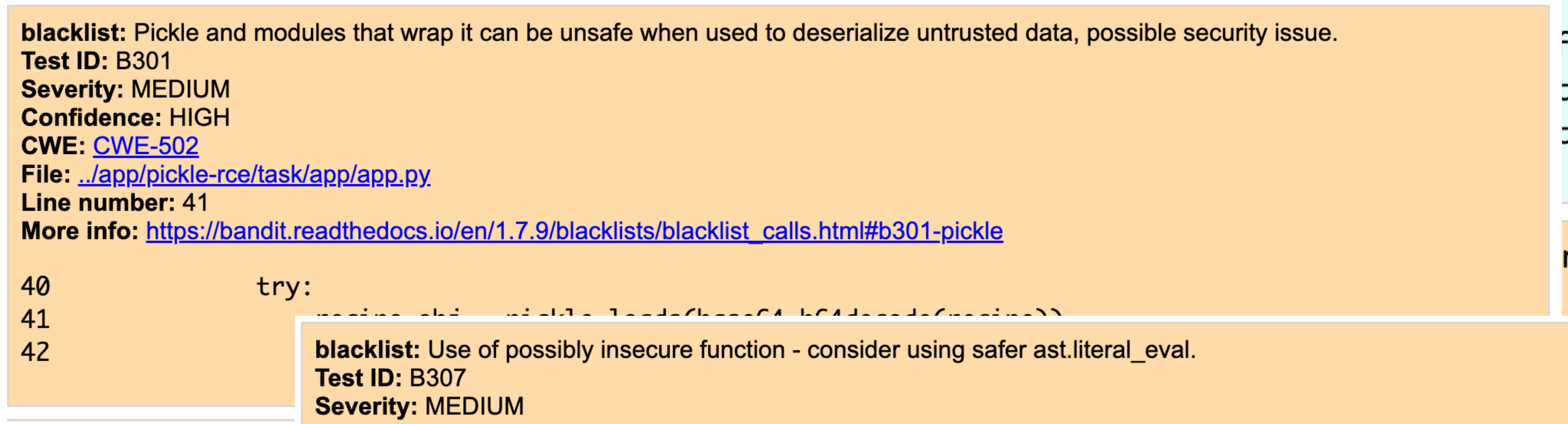
lacklist_calls.html#b301-pickle

```
s(base64.b64decode(recipe))
recipe.html', recipe=recipe_obj, dumped=
```

k/app/app.py

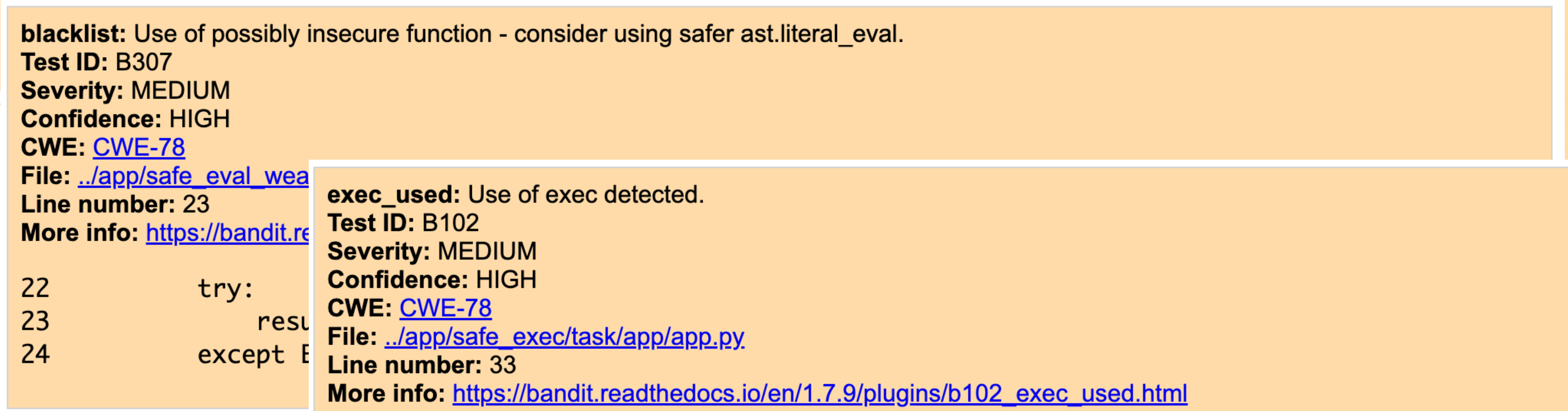
ndit.readthedocs.io/en/1.7.9/plugins/b113_request_without_timeout.html

```
:
  response = requests.get(url)
  return render_template('proxy.html', content=response.content.decode
```



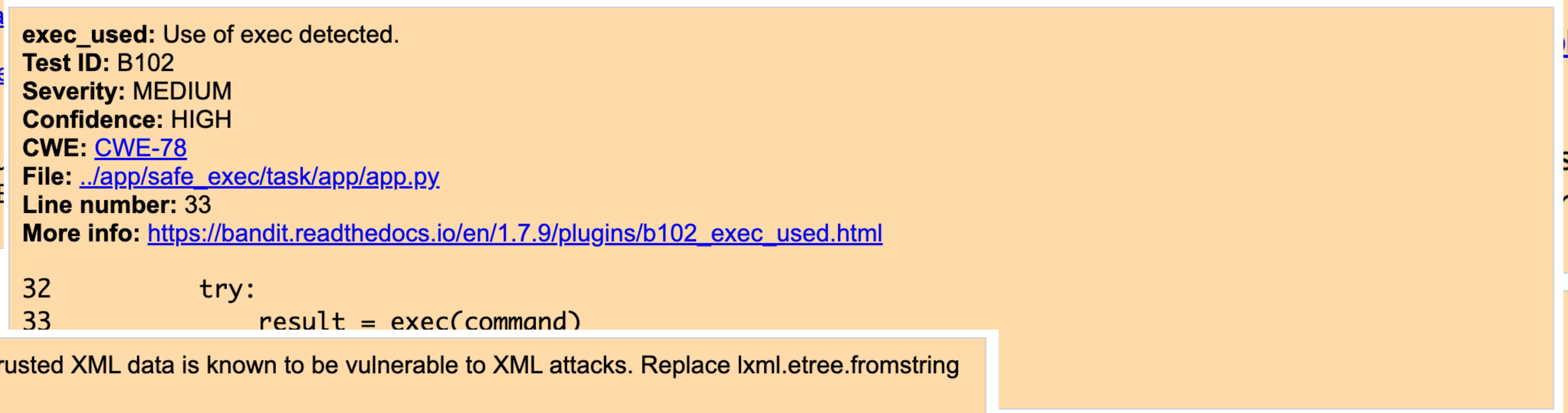
blacklist: Pickle and modules that wrap it can be unsafe when used to deserialize untrusted data, possible security issue.
Test ID: B301
Severity: MEDIUM
Confidence: HIGH
CWE: CWE-502
File: ../app/pickle-rce/task/app/app.py
Line number: 41
More info: https://bandit.readthedocs.io/en/1.7.9/blacklists/blacklist_calls.html#b301-pickle

```
40              try:
41                  recipe_obj = pickle.loads(base64.b64decode(recipe))
42
```



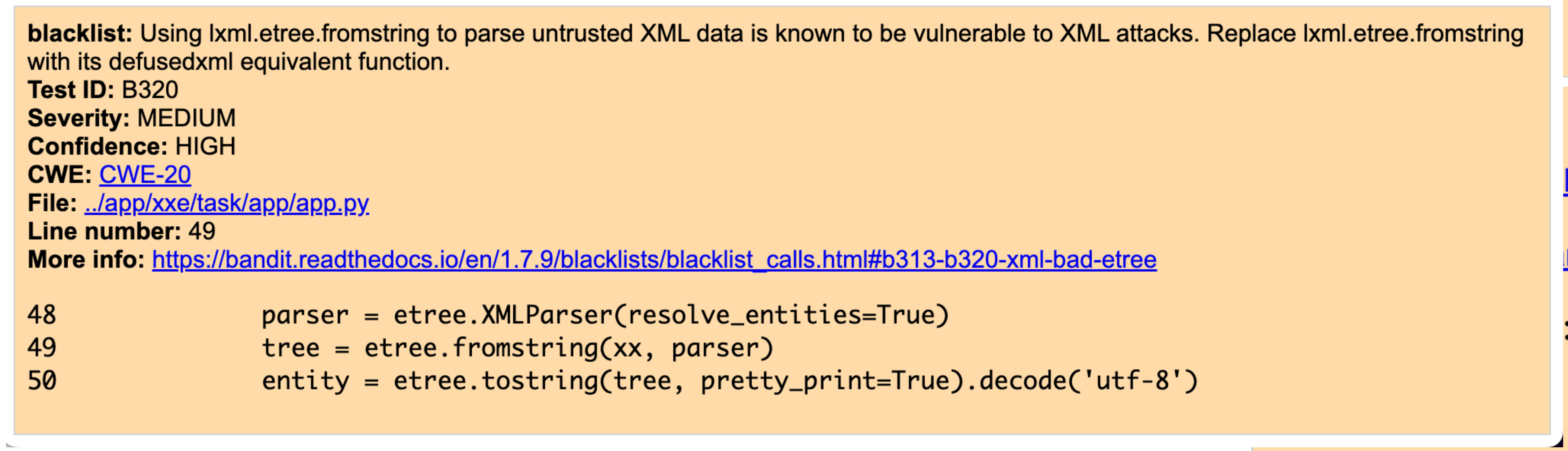
blacklist: Use of possibly insecure function - consider using safer ast.literal_eval.
Test ID: B307
Severity: MEDIUM
Confidence: HIGH
CWE: CWE-78
File: ../app/safe_eval_wea
Line number: 23
More info: https://bandit.re

```
22          try:
23              resu
24          except E
```



exec_used: Use of exec detected.
Test ID: B102
Severity: MEDIUM
Confidence: HIGH
CWE: CWE-78
File: ../app/safe_exec/task/app/app.py
Line number: 33
More info: https://bandit.readthedocs.io/en/1.7.9/plugins/b102_exec_used.html

```
32          try:
33              result = exec(command)
```



blacklist: Using lxml.etree.fromstring to parse untrusted XML data is known to be vulnerable to XML attacks. Replace lxml.etree.fromstring with its defusedxml equivalent function.
Test ID: B320
Severity: MEDIUM
Confidence: HIGH
CWE: CWE-20
File: ../app/xxe/task/app/app.py
Line number: 49
More info: https://bandit.readthedocs.io/en/1.7.9/blacklists/blacklist_calls.html#b313-b320-xml-bad-etree

```
48              parser = etree.XMLParser(resolve_entities=True)
49              tree = etree.fromstring(xx, parser)
50              entity = etree.tostring(tree, pretty_print=True).decode('utf-8')
```

Bandit: Результаты

1. ~~JWT Auth - CWE-347~~
2. ~~LFI - CWE-98~~
3. NoSQL injection - CWE-134
4. SQL injection - CWE-89
5. Pickle RCE - CWE-502
6. Exec RCE - CWE-78
7. Eval RCE - CWE-95
8. ~~SSTI - CWE-95~~
9. XXE - CWE-611
10. ~~Off by Slash - CWE-22~~

6/10

SonarQube

Continuous Inspection

/SonarSource/sonarqube

SonarQube

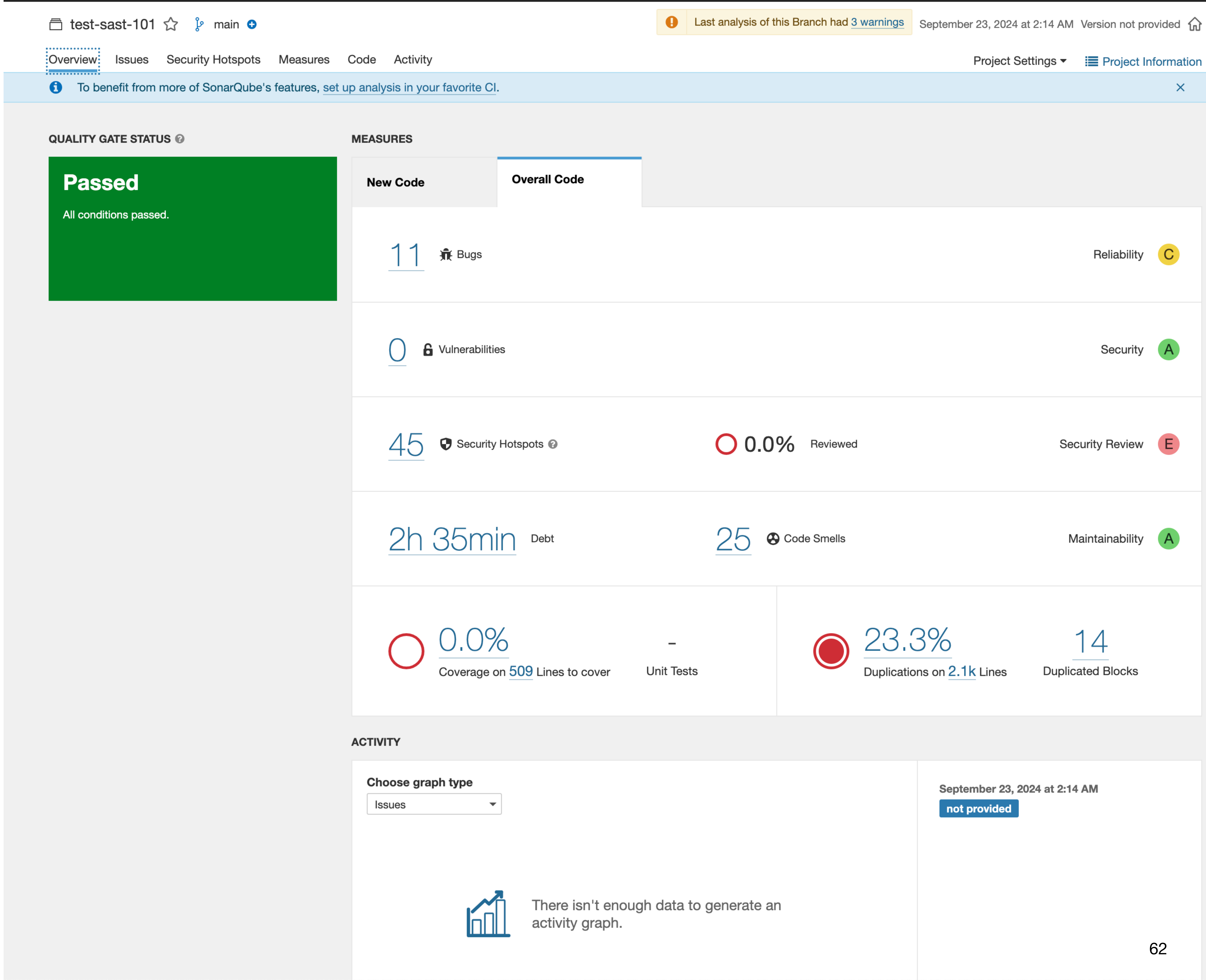
Установка

```yaml
version: "3"

services:
  sonarqube:
    image: sonarqube:lts-community
    depends_on:
      - sonar_db
    environment:
      SONAR_JDBC_URL: jdbc:postgresql://sonar_db:5432/sonar
      SONAR_JDBC_USERNAME: sonar
      SONAR_JDBC_PASSWORD: sonar
    ports:
      - "9001:9000"
    volumes:
      - sonarqube_conf:/opt/sonarqube/conf
      - sonarqube_data:/opt/sonarqube/data
      - sonarqube_extensions:/opt/sonarqube/extensions
      - sonarqube_logs:/opt/sonarqube/logs
      - sonarqube_temp:/opt/sonarqube/temp

  sonar_db:
    image: postgres:13
    environment:
      POSTGRES_USER: sonar
      POSTGRES_PASSWORD: sonar
      POSTGRES_DB: sonar
    volumes:
      - sonar_db:/var/lib/postgresql
```

SonarQube

```
sonar-scanner \
  -Dsonar.projectKey=test-sast-101 \
  -Dsonar.sources=. \
  -Dsonar.host.url=http://localhost:9001 \
  -Dsonar.login=sqp_094a749b0917dda75b28c515898d1568f20aff1f
```

SonarQube



test-sast-101 main

Last analysis of this Branch had 3 warnings September 23, 2024 at 2:14 AM Version not provided

Overview Issues Security Hotspots Measures Code Activity Project Settings Project Information

To benefit from more of SonarQube's features, set up analysis in your favorite CI.

QUALITY GATE STATUS

Passed

All conditions passed.

MEASURES

New Code | Overall Code

11 Bugs — Reliability C

0 Vulnerabilities — Security A

45 Security Hotspots — 0.0% Reviewed — Security Review E

2h 35min Debt — 25 Code Smells — Maintainability A

0.0% Coverage on 509 Lines to cover — - Unit Tests

23.3% Duplications on 2.1k Lines — 14 Duplicated Blocks

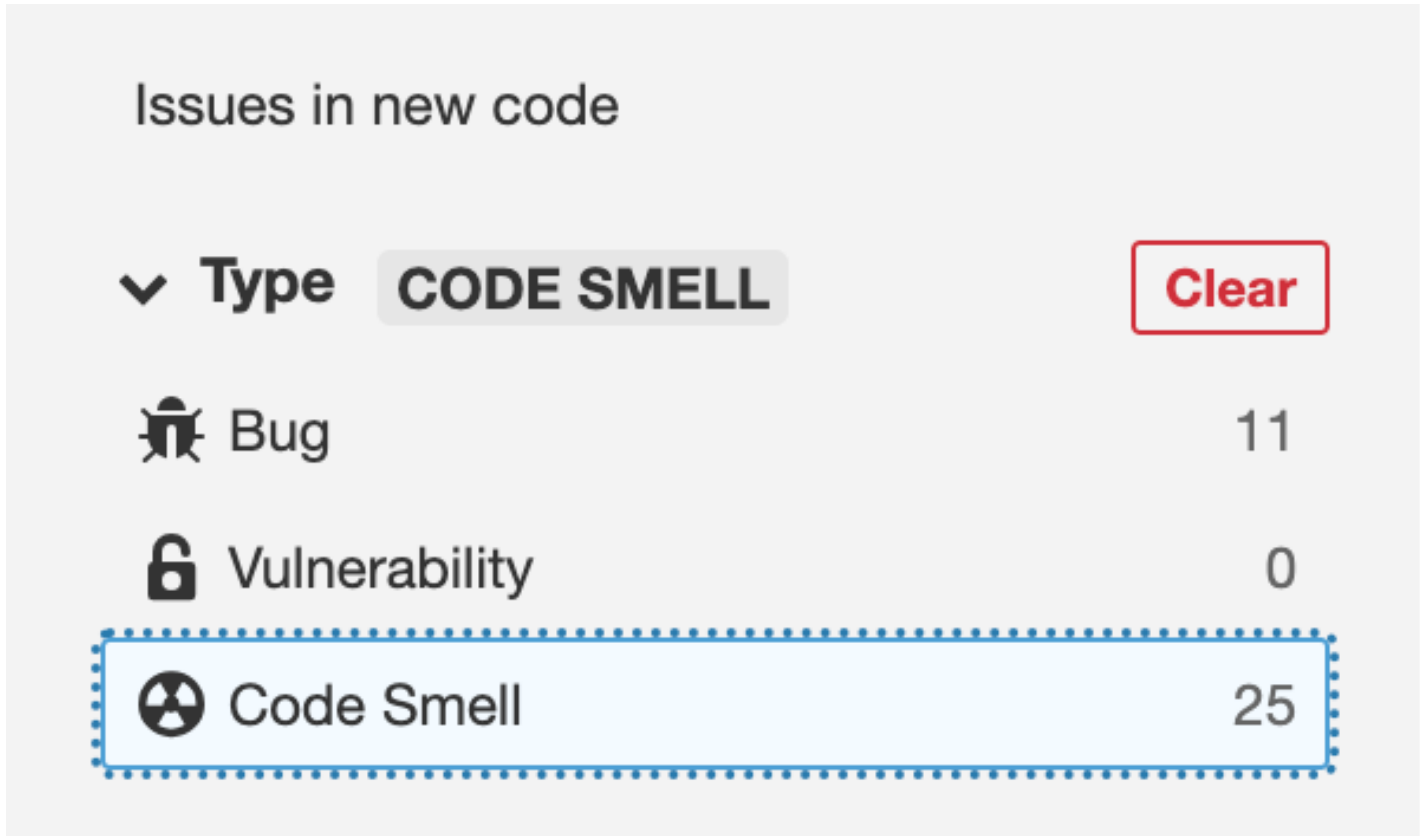
ACTIVITY

Choose graph type: Issues

There isn't enough data to generate an activity graph.

September 23, 2024 at 2:14 AM
not provided

SonarQube: результаты

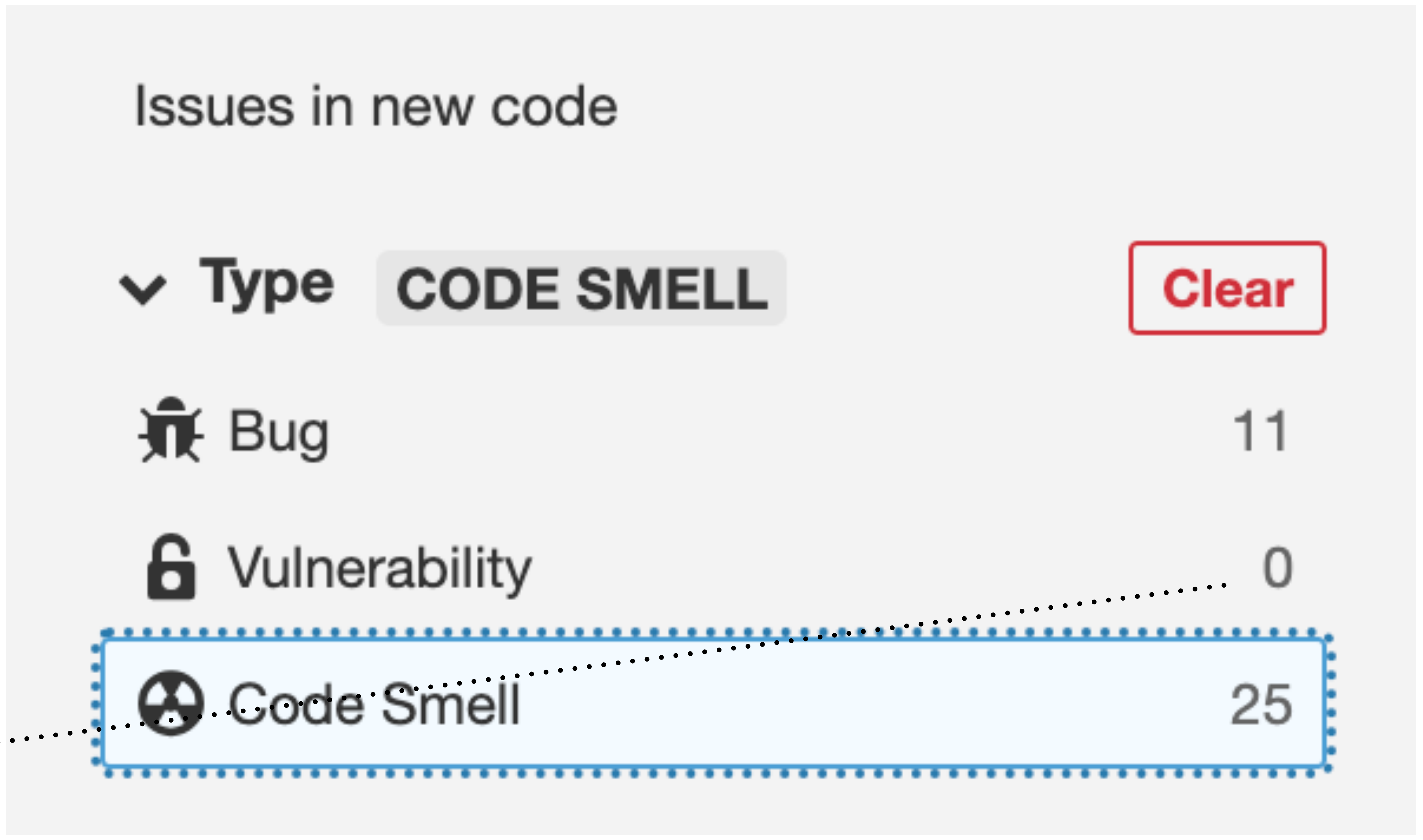


Issues in new code

Type CODE SMELL Clear

Bug	11
Vulnerability	0
Code Smell	25

SonarQube: результаты


Issues in new code

Type CODE SMELL Clear

Bug	11
Vulnerability	0
Code Smell	25

SonarQube: Результаты

1. ~~JWT Auth - CWE-347~~
2. ~~LFI - CWE-98~~
3. NoSQL injection - CWE-134
4. SQL injection - CWE-89
5. Pickle RCE - CWE-502
6. Exec RCE - CWE-78
7. Eval RCE - CW
8. ~~SSTI - CWE-95~~
9. XXE - CWE-61
10. ~~Off by Slash - CWE-22~~

SemGrep

Semgrep

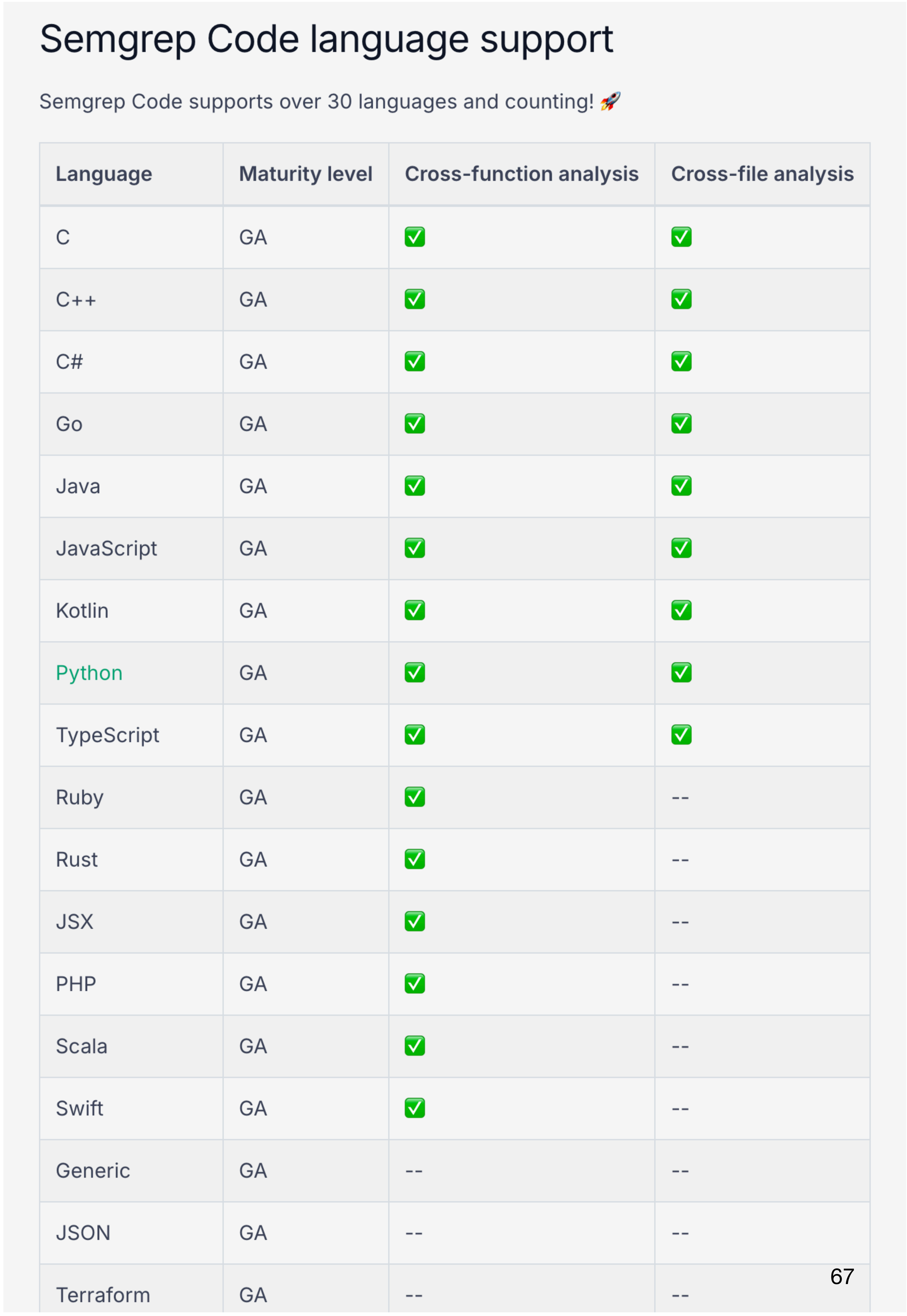
/semgrep/semgrep

https://semgrep.dev/

SemGrep:
поддержка других ЯП

SemGrep OSS Engine
(Community rules)

Semgrep Code
(Pro rules + Pro Engine)



Semgrep Code language support

Semgrep Code supports over 30 languages and counting! 🚀

Language	Maturity level	Cross-function analysis	Cross-file analysis
C	GA	✅	✅
C++	GA	✅	✅
C#	GA	✅	✅
Go	GA	✅	✅
Java	GA	✅	✅
JavaScript	GA	✅	✅
Kotlin	GA	✅	✅
Python	GA	✅	✅
TypeScript	GA	✅	✅
Ruby	GA	✅	--
Rust	GA	✅	--
JSX	GA	✅	--
PHP	GA	✅	--
Scala	GA	✅	--
Swift	GA	✅	--
Generic	GA	--	--
JSON	GA	--	--
Terraform	GA	--	--

SemGrep: правила

```yaml
rules:
  - id: python.avoid-eval
    patterns:
      - pattern: eval($EXPR)
    message: Avoid using `eval()`, as it can lead to code injection vulnerabilities.
             Consider using `ast.literal_eval()` for safe evaluation of expressions.
    languages: [python]
    severity: error
    metadata:
      cwe: "CWE-94"
      remediation: "Use `ast.literal_eval()` instead of `eval()`
                    for safely evaluating Python expressions."
```

SemGrep: Результаты

1. ~~JWT Auth - CWE-347~~
2. ~~LFI - CWE-98~~
3. ~~NoSQL injection - CWE-134~~
4. ~~SQL injection - CWE-89~~
5. Pickle RCE - CWE-502
6. Exec RCE - CWE-78
7. ~~Eval RCE - CWE-95~~
8. ~~SSTI - CWE-95~~
9. XXE - CWE-611
10. ~~Off by Slash - CWE-22~~

SemGrep: Pro Engine

Fix vulnerabilities today and prevent tomorrow's with secure guardrails Learn more →

mkobilev-perso...

- Dashboard
- Projects
- Code
- Secrets
- Supply Chain
- Rules

Get Started 8%

Code

Group by Rule | All time

Projects and branches: All projects with primary branches

Tags: All project tags

Status: Open (3)

Severity: [✓ High] Medium Low

Confidence: [✓ High] Medium Low

Pro findings only

Category: All categories

Action: Monitor Comment Block

Rule: All rules

Ruleset

3 Matching Findings

Sort by highest severity | Analyze (0) | Triage (0)

tainted-code-stdlib-flask (2) — Pro · Security · Python

The application might dynamically evaluate untrusted input, which can lead to a code injection vulnerability. An attacker can execute arbitrary code, potentially gaining complete control of the system. To prevent this vulnerability, avoid executing code containing user input. If this is unavoidable, validate and sanitize the input, and use safe alternatives for evaluating user input.

- 2m safe_eval_weak/task/app/app.py:23 — app develop Details
- 2m safe_exec/task/app/app.py:33 — app develop Details

tainted-pickle-flask (1) — Pro · Security · Python

The application may convert user-controlled data into an object, which can lead to an insecure deserialization vulnerability. An attacker can create a malicious serialized object, pass it to the application, and take advantage of the deserialization process to perform Denial-of-service (DoS), Remote code execution (RCE), or bypass access control measures. The C implementations

Show more

- 2m pickle-rce/task/app/app.py:41 — app develop Details

Per page: 10 | 1

SemGrep: Pro Engine

1. ~~JWT Auth - CWE-347~~
2. LFI - CWE-98
3. NoSQL injection - CWE-134
4. SQL injection - CWE-89
5. Pickle RCE - CWE-502

9/10

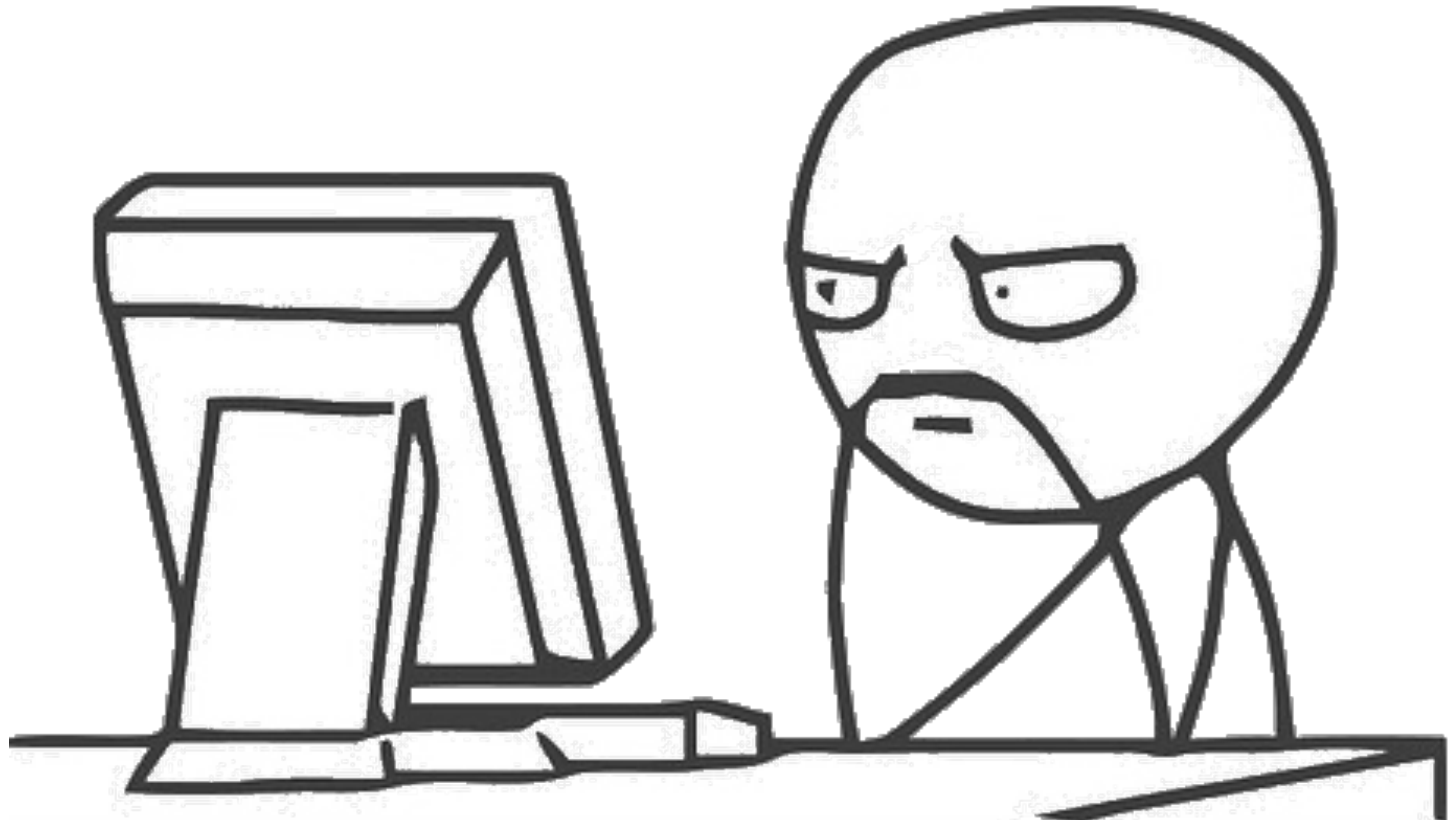
6. Exec RCE - CWE-78
7. Eval RCE - CWE-95
8. SSTI - CWE-95
9. XXE - CWE-611
10. Off by Slash - CWE-22

CodeQL

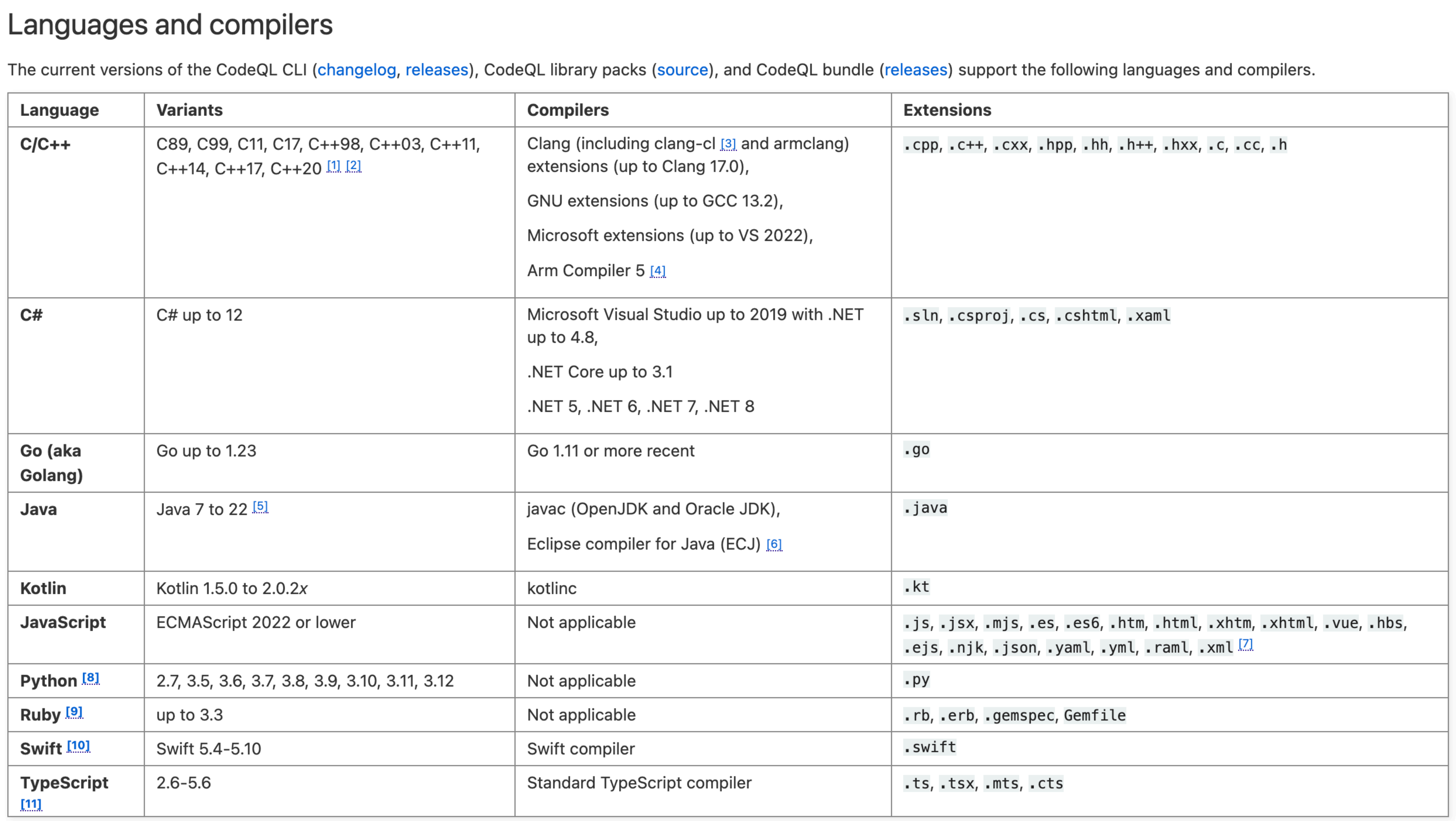
CodeQL

/github/codeql

https://codeql.github.com



CodeQL: поддержка других ЯП



Languages and compilers

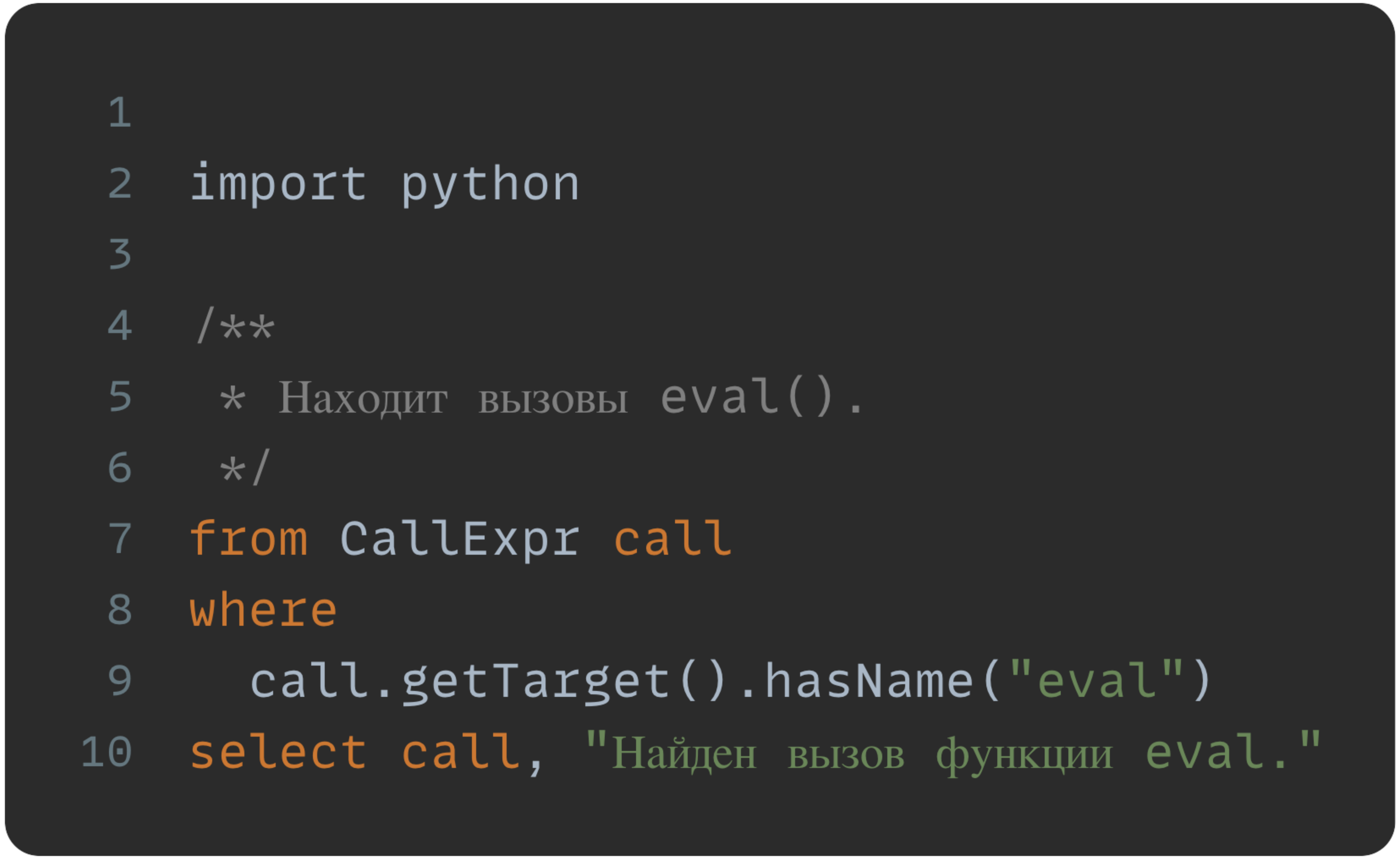
The current versions of the CodeQL CLI (changelog, releases), CodeQL library packs (source), and CodeQL bundle (releases) support the following languages and compilers.

Language	Variants	Compilers	Extensions
C/C++	C89, C99, C11, C17, C++98, C++03, C++11, C++14, C++17, C++20 [1] [2]	Clang (including clang-cl [3] and armclang) extensions (up to Clang 17.0), GNU extensions (up to GCC 13.2), Microsoft extensions (up to VS 2022), Arm Compiler 5 [4]	`.cpp`, `.c++`, `.cxx`, `.hpp`, `.hh`, `.h++`, `.hxx`, `.c`, `.cc`, `.h`
C#	C# up to 12	Microsoft Visual Studio up to 2019 with .NET up to 4.8, .NET Core up to 3.1 .NET 5, .NET 6, .NET 7, .NET 8	`.sln`, `.csproj`, `.cs`, `.cshtml`, `.xaml`
Go (aka Golang)	Go up to 1.23	Go 1.11 or more recent	`.go`
Java	Java 7 to 22 [5]	javac (OpenJDK and Oracle JDK), Eclipse compiler for Java (ECJ) [6]	`.java`
Kotlin	Kotlin 1.5.0 to 2.0.2*x*	kotlinc	`.kt`
JavaScript	ECMAScript 2022 or lower	Not applicable	`.js`, `.jsx`, `.mjs`, `.es`, `.es6`, `.htm`, `.html`, `.xhtm`, `.xhtml`, `.vue`, `.hbs`, `.ejs`, `.njk`, `.json`, `.yaml`, `.yml`, `.raml`, `.xml` [7]
Python [8]	2.7, 3.5, 3.6, 3.7, 3.8, 3.9, 3.10, 3.11, 3.12	Not applicable	`.py`
Ruby [9]	up to 3.3	Not applicable	`.rb`, `.erb`, `.gemspec`, `Gemfile`
Swift [10]	Swift 5.4-5.10	Swift compiler	`.swift`
TypeScript [11]	2.6-5.6	Standard TypeScript compiler	`.ts`, `.tsx`, `.mts`, `.cts`

CodeQL: запуск

- Сбор данных о коде
- Создание базы данных CodeQL
- Выполнение запросов CodeQL
- Анализ результатов

CodeQL: синтаксис



```

import python

/**
 * Находит вызовы eval().
 */
from CallExpr call
where
  call.getTarget().hasName("eval")
select call, "Найден вызов функции eval."
```

CodeQL: результаты

1. ~~JWT Auth - CWE-347~~
2. LFI - CWE-98
3. NoSQL injection - CWE-134
4. SQL injection - CWE-89
5. Pickle RCE - CWE-502
6. Exec RCE - CWE-78
7. Eval RCE - CWE-95
8. SSTI - CWE-95
9. XXE - CWE-611
10. Off by Slash - CWE-22

Выводы

False positives and False negatives

Обработка результатов

Придётся писать правила

как быть?

👌

Bandit

👌

Bandit

~~SonarQube~~

👌

Bandit

🔥

SemGrep

~~SonarQube~~

CodeQL

🔥

SemGrep

👌

Bandit

→ 💸

~~SonarQube~~

CodeQL

Слайд для вопросов и ответов

Выбираем open-source **SAST** для **Python** проектов

Максим Кобилев

mkobilev/python-sast-101