
# Open-source SAST на максималках

Герасимов Александр

Основатель Awillix

Москва, 25.08.2022

# Awillix

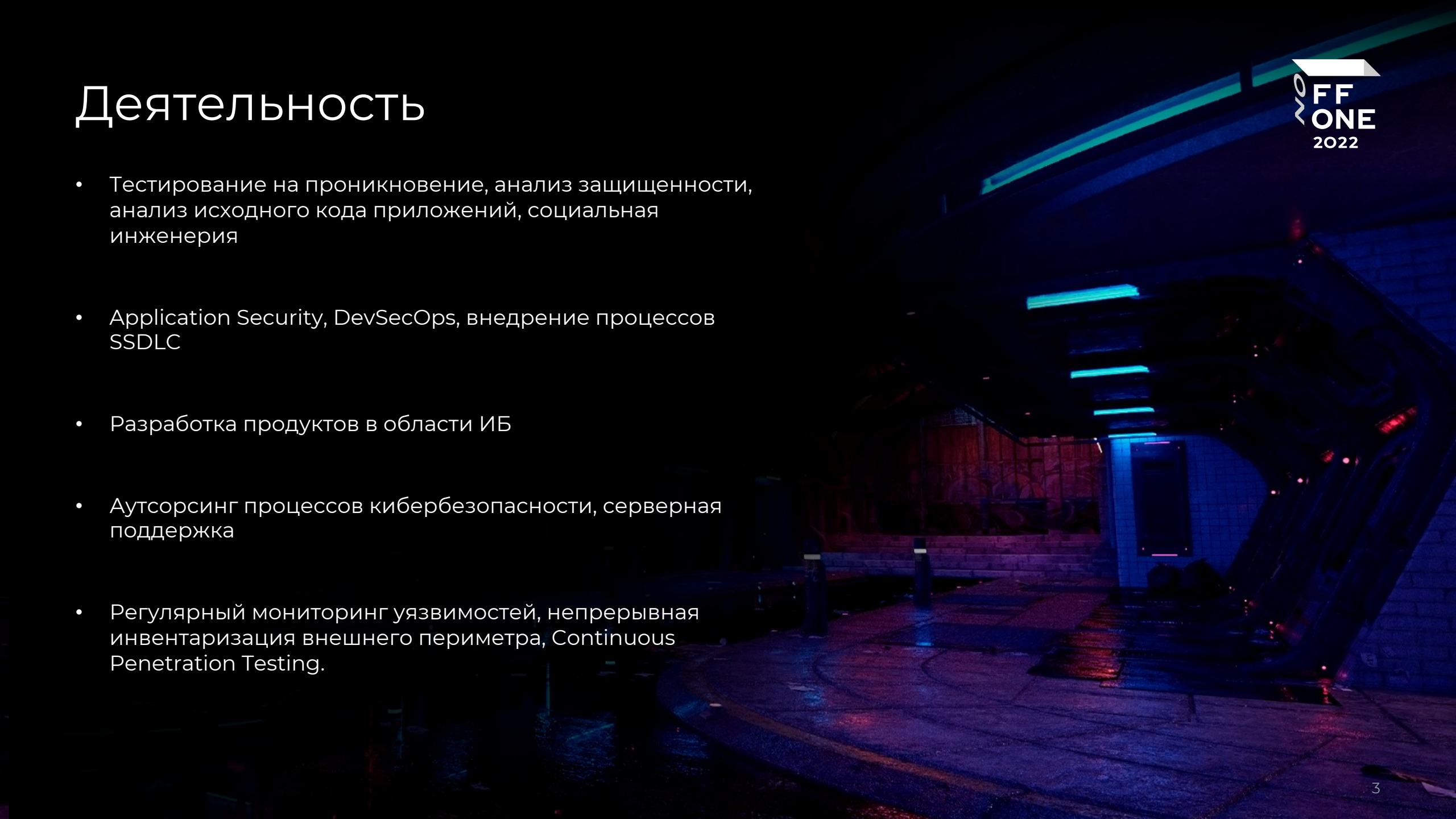
- Одна из лучших offensive-компаний на рынке кибербезопасности

- Мы гарантируем качество и конфиденциальность во всех пентестах. Делаем проекты любой сложности быстро, оставаясь всегда на связи с заказчиком

- Снимаем сомнения в отношении киберрисков клиентов, помогаем более эффективно расходовать бюджет на ИБ и доносить проблемы безопасности до руководителей

# Деятельность

- Тестирование на проникновение, анализ защищенности, анализ исходного кода приложений, социальная инженерия
- Application Security, DevSecOps, внедрение процессов SSDLC
- Разработка продуктов в области ИБ
- Аутсорсинг процессов кибербезопасности, серверная поддержка
- Регулярный мониторинг уязвимостей, непрерывная инвентаризация внешнего периметра, Continuous Penetration Testing.



# Немножко про SAST

## Что это такое?

Решение, которое позволяет обеспечивать соответствие стандартам, находить недостатки и уязвимости в исходном коде без запуска самого приложения.

## Что хотим решить с помощью SAST?

- Соответствие кода практикам введёнными в компании
- Поиск недостатков конфигурации (например, nginx, Docker)
- Поиск секретов в коде (API-ключи, токены, пароли и т.д.)
- Уязвимости и недостатки в исходном коде

# Почему говорим про Open-source

- Если не использовали SAST-решения до этого, Open-source может помочь сформировать требования
- Хорошая поддержка от сообщества
- Можно создать собственные правила сканирования
- Популярные инструменты  имеют интеграции с IDE
- Не попадут под санкции

# Semgrep

## Особенности решения

- Поддержка 25+ языков программирования
- Более 2400 шаблонов
- Проверка конфигурационных файлов
- Правила описываются как код - легко написать свои, под конкретные задачи
- Интеграции:  IDE, CI/CD, Slack, Jira
- Возможность создания автоисправления
- Поиск недостатков по Data flow
- Поиск секретов по энтропии
- Может запускаться локально, без доступа в Интернет
- Имеет собственную веб-платформу по управлению уязвимостями

# Варианты использования  Semgrep

## 1. Semgerp CLI

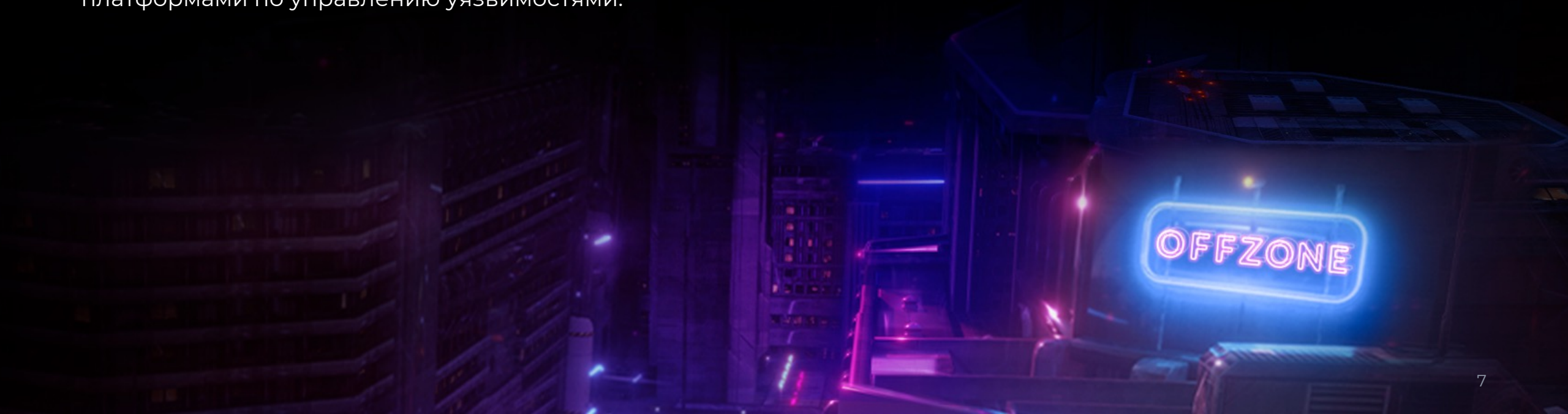
Консольный вариант использования Semgrep. Позволяет построить необходимый флоу в любом CI/CD.

Удобно использовать в случае self-managed SCM и, когда необходимы собственные интеграции с платформами по управлению уязвимостями.

## 2. Semgrep APP

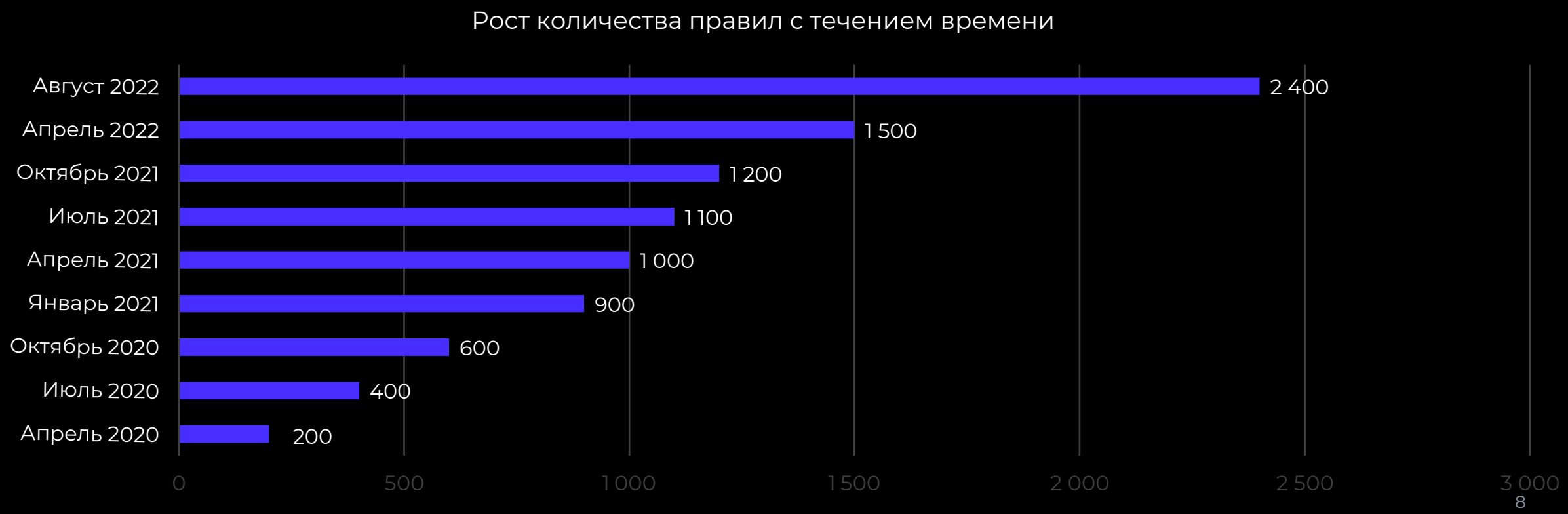
Веб-сервис, из которого осуществляется сканирование, управление уязвимостями  в репозиториях Github / Gitlab.

Удобно использовать, когда организация использует и доверяет SaaS решениям



# Правила сканирования

- Где брать правила:
  - https://github.com/returntocorp/semgrep-rules
  - https://semgrep.dev/r
- Playground: https://semgrep.dev/playground



# Поиграем с Semgrep CLI

## Локальная установка

Любая система с python:

$ python3 -m pip install semgrep

Любая система с Docker:

$ docker pull returntocorp/semgrep

# Как выглядят правила

```
# python-insecure-crypto.yaml
id: md5-usage
languages:
  - python
message: Found md5 usage
pattern: hashlib.md5(...)
severity: ERROR
```

```
# md5.py
import hashlib
import sys


password = sys.argv[1].encode("utf-8")
h_password = hashlib.md5(password).hexdigest()


if  h_password == ”d8578edf8458ce06fbc5bb76a58c5ca4":
    ....
```

# Запуск Semgrep CLI


```
$ semgrep --config "./rules/python-insecure-crypto.yaml" samples/md5.py
Scanning 1 file.

Findings:
  samples/md5.py
    rules.md5-usage
      Found md5 usage
        4┆ h_password = hashlib.md5(password).hexdigest()

Ran 1 rule on 1 file: 1 finding.
```

# Основные поля в правилах сканирования

| Поле | Тип | Описание |
|---|---|---|
| id | string | Уникальный идентификатора. *md5-usage |
| message | string | Сообщение с указанием причины срабатывания правила и способов устранения |
| severity | string | INFO, WARNING, ERROR |
| languages | array | Список языков программирования, которые затрагивает правило. * cpp: [.cpp, .h] |
| pattern | string | Поиск по паттерну |
| patterns | array | Логическое «и» по списку паттернов |
| pattern-either | array | Логическое «или» по списку паттернов |
| pattern-regex | string | Поиск по регулярному выражению |

# Погружаемся глубже: pattern-either


```
id: missing-user
patterns:
  - pattern-either:
      - pattern: CMD ...
      - pattern: ENTRYPOINT ...
  - pattern-not-inside: |
      USER $USER
      ...
      ...
message: By not specifying a USER, a programs in the container may run as 'root'.
severity: ERROR
languages: [generic]
paths:
  include: ["*Dockerfile*"]
```

# Погружаемся  глубже: metavariable-comparison


id: insufficient-rsa-key-size

patterns:

**- pattern-either:**

**- pattern: Cryptodome.PublicKey.RSA.generate(..., bits=$SIZE, ...)**

**- pattern: Cryptodome.PublicKey.RSA.generate($SIZE, ...)**

**- metavariable-comparison:**

**metavariable: $SIZE**

**comparison: $SIZE <= 2048**

message: Detected an insufficient key size for RSA

languages: [python]

severity: WARNING

# Погружаемся  глубже: metavariable-comparison


Оператор сравнения метапеременных использует базовые выражением сравнения Python

```
id: web-ports
patterns:
  - pattern: server.run(port=$PORT, env=$ENV)
  - metavariable-comparison:
      metavariable: $PORT, $ENV
      comparison: not ($PORT in [8000, 8080, 8443]) and ($ENV in ["dev", "test"])
message: Web server in dev or test should use [8000, 8080, 8443] ports
languages:
  - python
severity: ERROR
```

# Автоматическое исправление


```
rules:
  - id: use-dict-get
    patterns:
      - pattern: $DICT[$KEY]
    fix: $DICT.get($KEY)
    message: "Use `.get()` method to avoid a KeyNotFound error"
    languages: [python]
    severity: ERROR
```

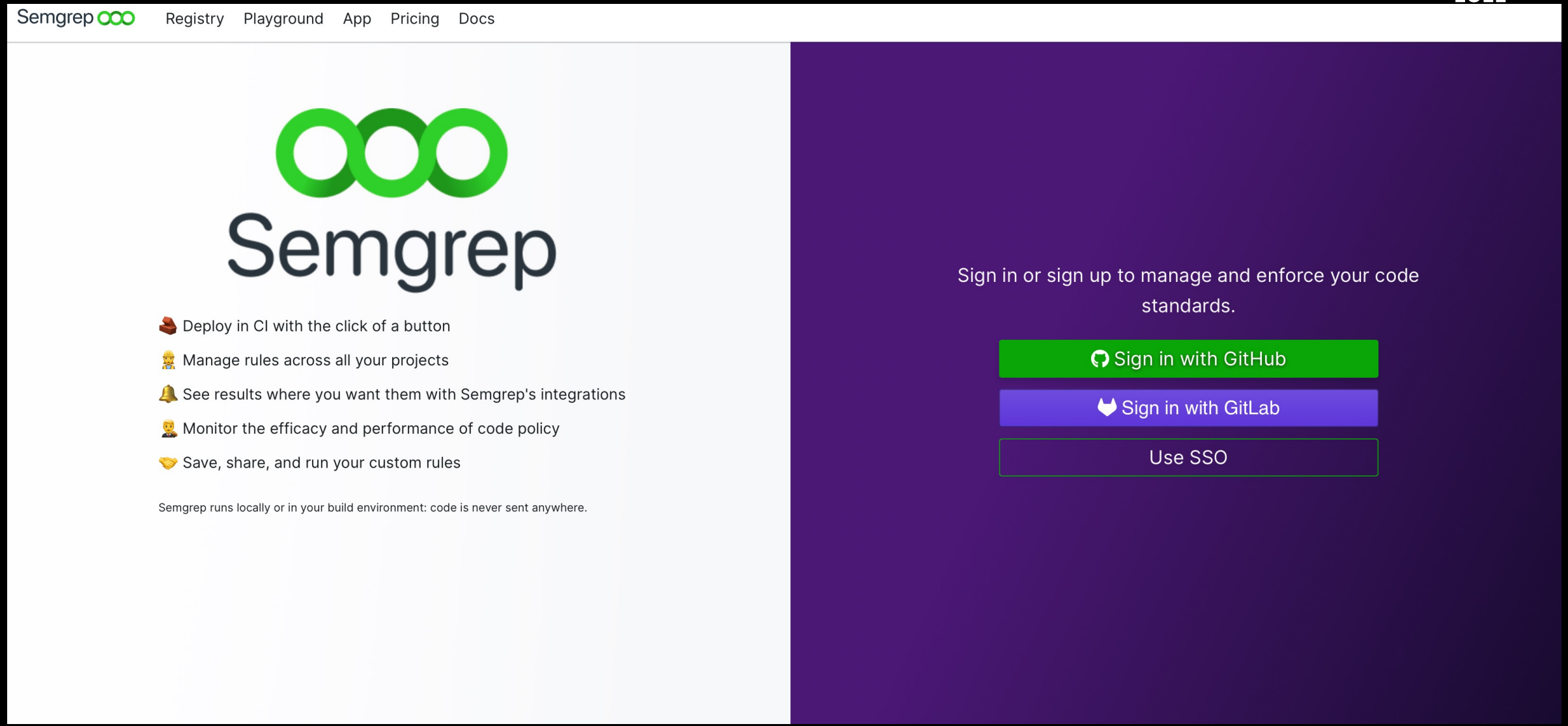
# Как добавить полезную нагрузку?


```
rules:
  - id: next_js-CVE-2020-5284
    patterns:
      - ...
metadata:
      cwe: " CWE-89: Improper Neutralization of Special Elements used in an SQL Command "
      owasp: " A03:2021-Injection"
      references:
        - https://owasp.org/Top10/A03_2021-Injection/
        - https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
      category: security
      remediation: >-
          1. use PreparedStatement() with bind variables
          2. Escaping All User-Supplied Input
```

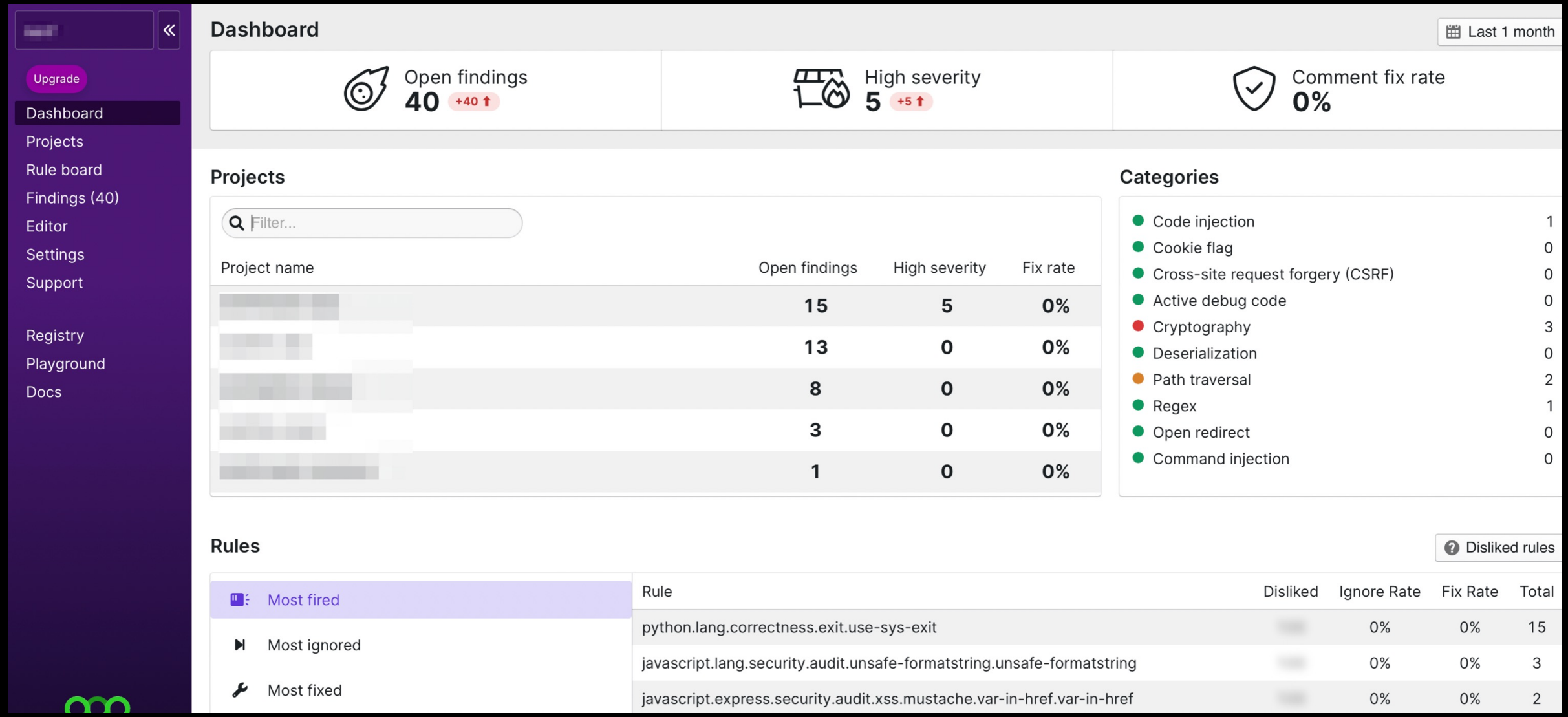
# Интеграция в CI/CD на примере Gitlab

```
semgrep:
  image: returntocorp/semgrep
  script: semgrep ci

  variables:
    SEMGREP_RULES: p/default
    GITLAB_TOKEN: $CI_BOT
    SEMGREP_GITLAB_JSON: "1” # Need to push findings to GitLab SAST Dashboard

  script: semgrep ci --gitlab-sast > gl-sast-report.json || true # Push findings to GitLab SAST Dashboard
   artifacts:
   reports:
     sast: gl-sast-report.json
```

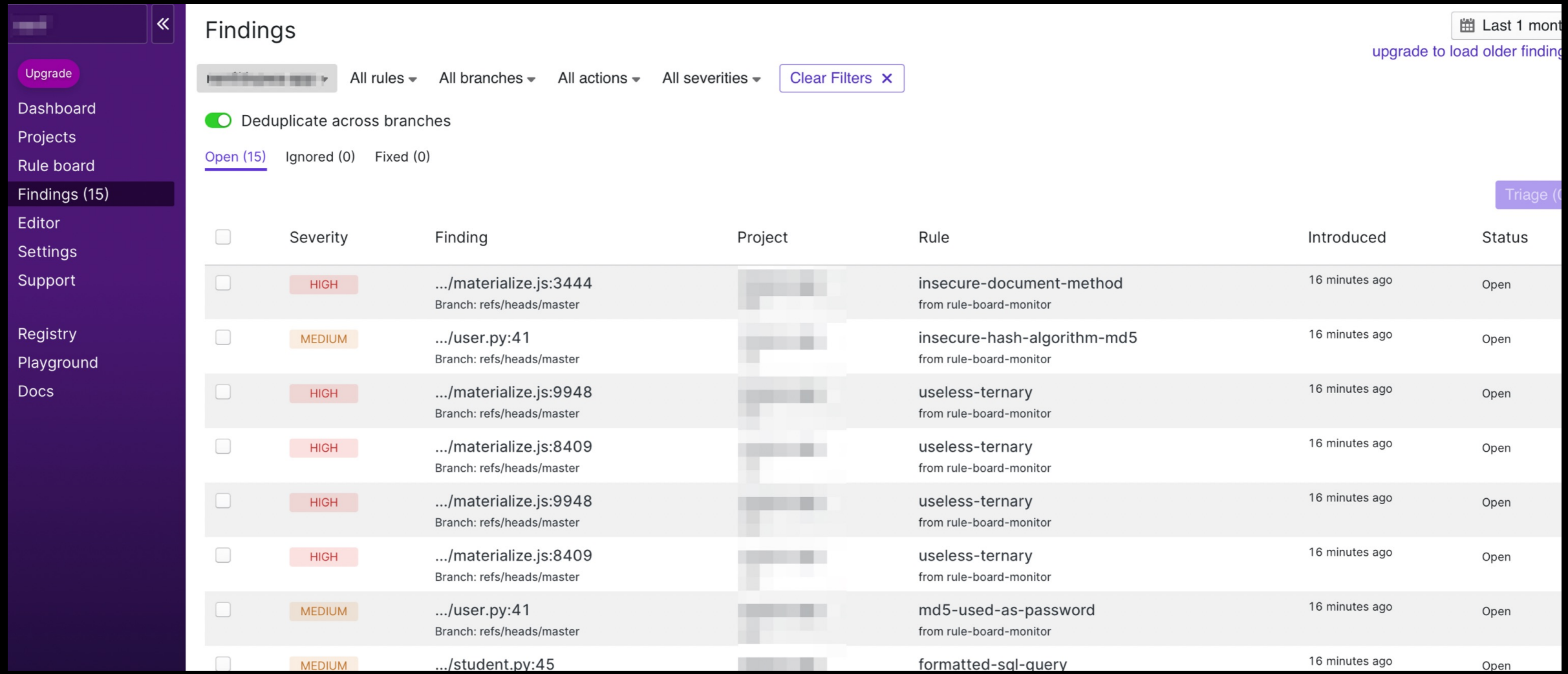
# Как насчет Semgrep APP



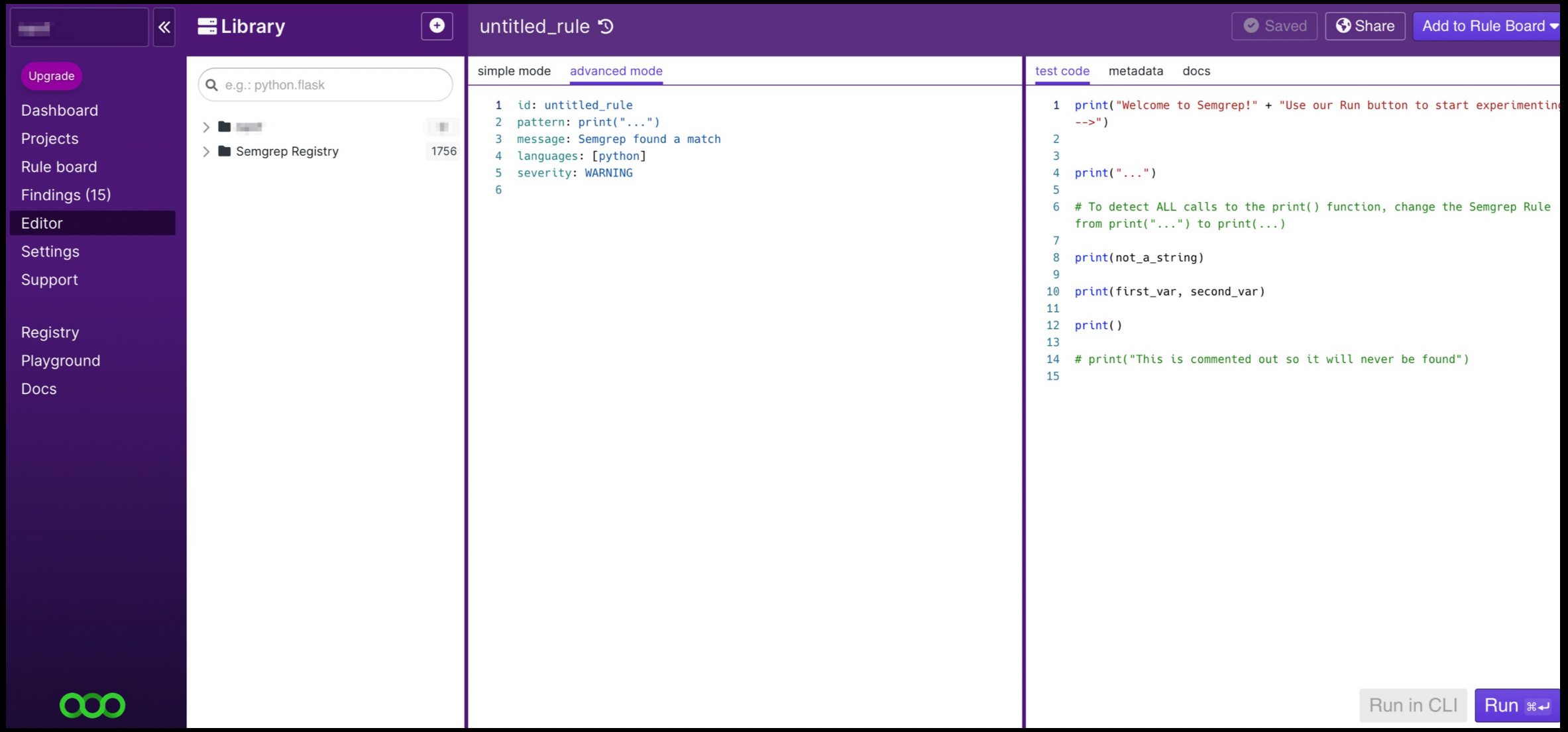
# Основной Dashboard



# Работа с дефектами

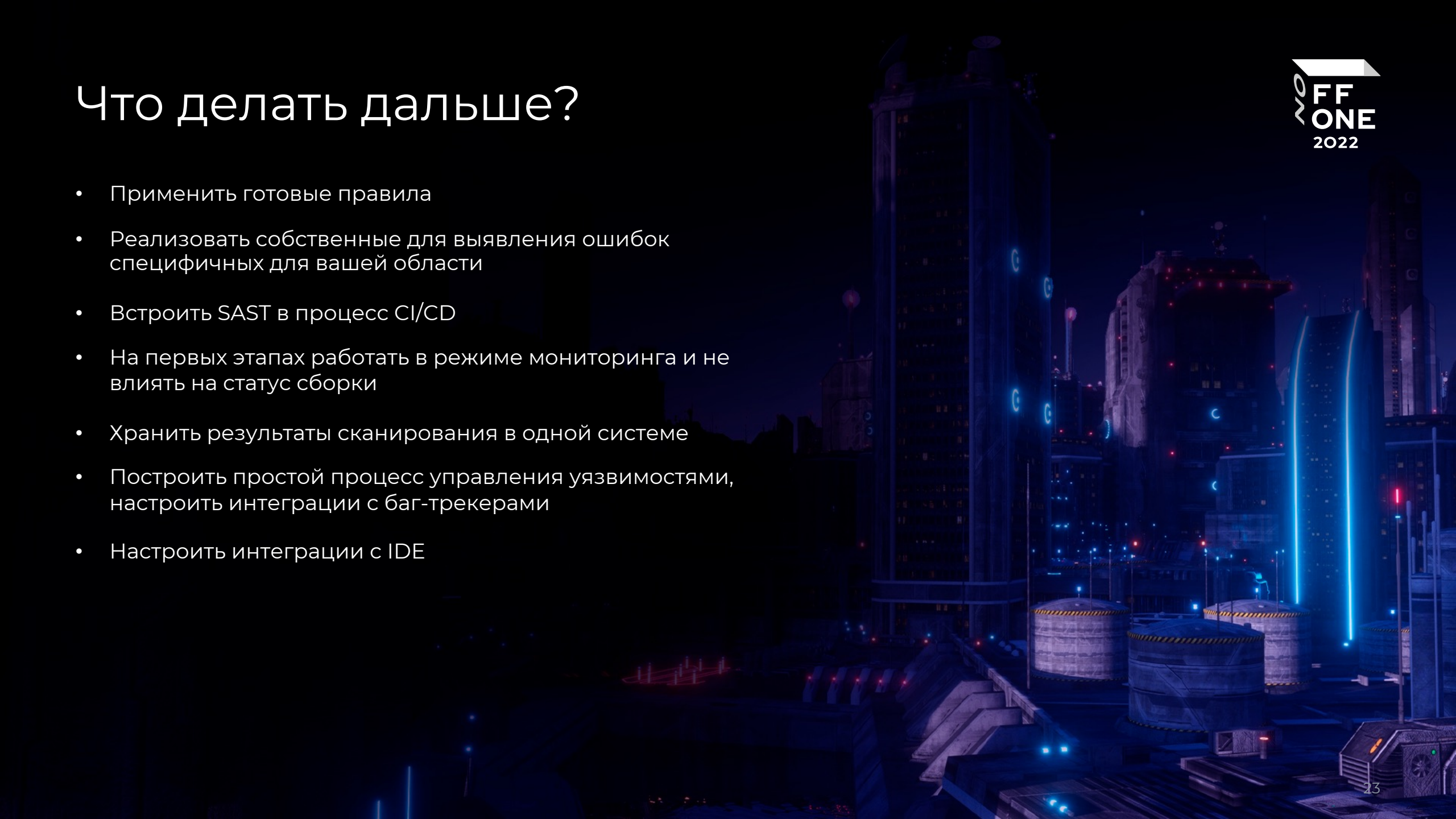


# Управление шаблонами



# Что делать дальше?

- Применить готовые правила
- Реализовать собственные для выявления ошибок специфичных для вашей области
- Встроить SAST в процесс CI/CD
- На первых этапах работать в режиме мониторинга и не влиять на статус сборки
- Хранить результаты сканирования в одной системе
- Построить простой процесс управления уязвимостями, настроить интеграции с баг-трекерами
- Настроить интеграции с IDE



# Контакты

Telegram: @n3nff

Telegram-канал: t.me/justsecurity


OFF
ONE
2022