**Раздел:** Анализ защищённости ПО

**Модуль 3:** Средства автоматизированного поиска уязвимостей в веб-приложениях

**Выполнил:** Василий Костюков

**Задание:** Пройдите следующие лабораторные работы в Juice Shop:

https://lms.skillfactory.ru/asset-v1:SkillFactory+MIFIIB+2022_DEC+type@asset+block@MIFIIB_m6.3_u7_1.pnghttps://lms.skillfactory.ru/asset-v1:SkillFactory+MIFIIB+2022_DEC+type@asset+block@MIFIIB_m6.3_u7_2.pnghttps://lms.skillfactory.ru/asset-v1:SkillFactory+MIFIIB+2022_DEC+type@asset+block@MIFIIB_m6.3_u7_3.png

Также проанализируйте фрагменты кода на наличие уязвимостей. Используйте дополнительные правила (любые) для сканирования кода:

* [Фрагмент № 1.](https://drive.google.com/file/d/1PF_J5Pmf166NEFlct_NaUmrTWpbN-Z9h/view?usp=drive_link)
* [Фрагмент № 2.](https://drive.google.com/file/d/1mPHdQlmt61uRsRql-_IoF7qUDRFLoAMT/view?usp=drive_link)
* [Фрагмент № 3.](https://drive.google.com/file/d/1WoYXpySLE9x2tQCuHK-rXBMqr3p8zLFl/view?usp=drive_link)

Условия реализации

В качестве отчёта предоставьте следующие скриншоты:

1. Скриншот выполненной работы CAPTCHA Bypass из таблицы Scoreboard Juice Shop.
2. Скриншот выполненной работы Database Schema из таблицы Scoreboard Juice Shop.
3. Скриншот выполненной работы Login Jim из таблицы Scoreboard Juice Shop

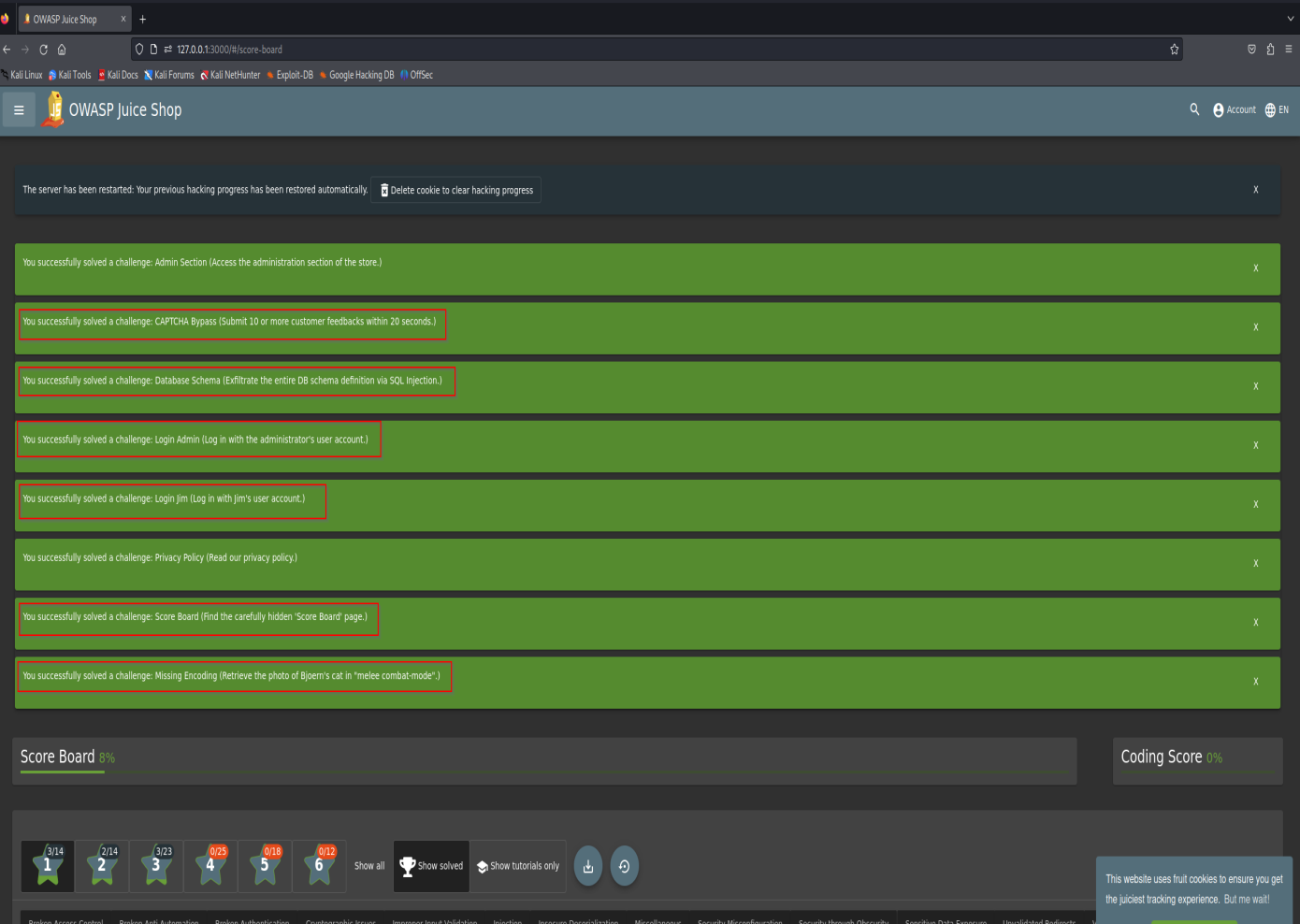
А также:

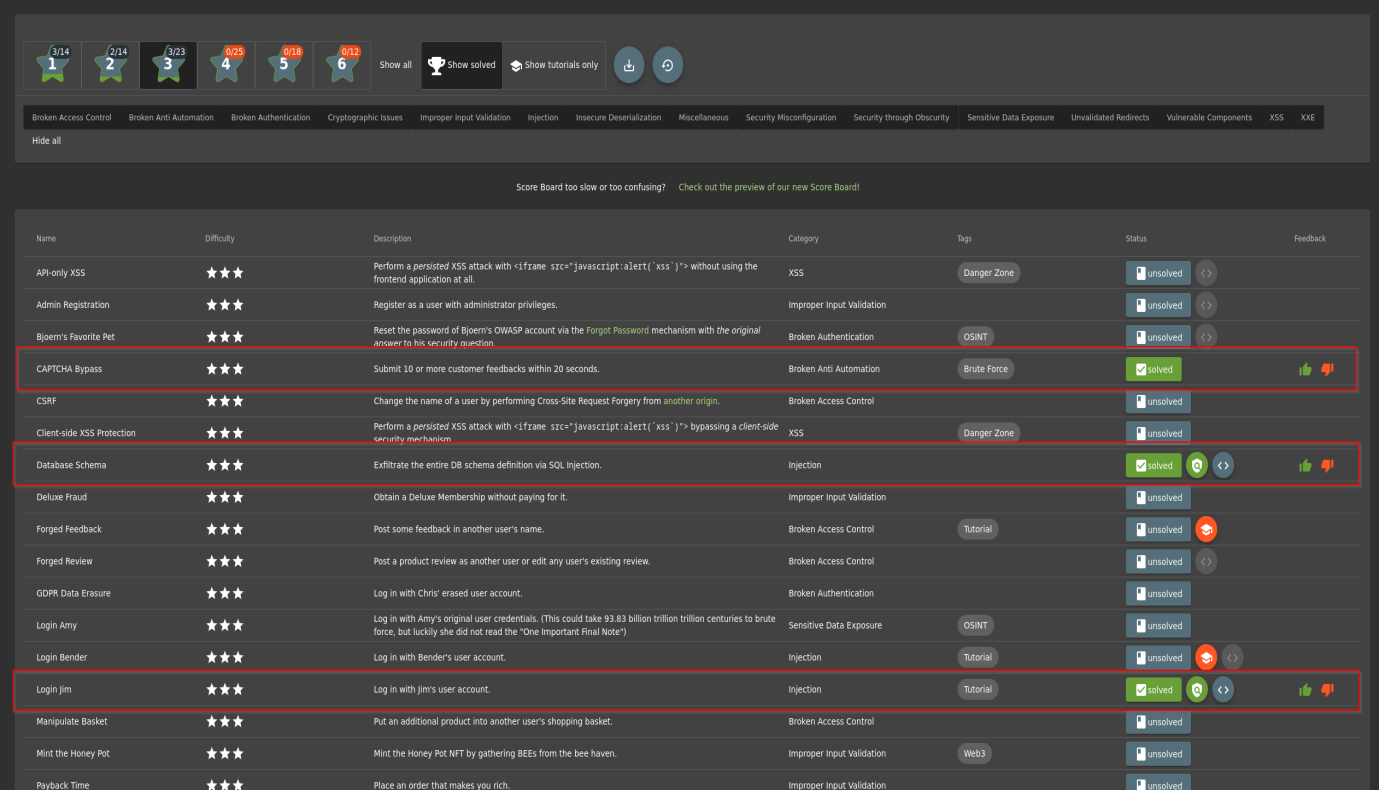
1. Команду для сканирования файлов на наличие уязвимостей.
2. Скриншот терминала с результатами сканирования.
3. Уязвимости, содержащиеся в приведённых фрагментах кода, в формате: «название файла» — «уязвимость».

Отчёт создайте в формате DOCX или PDF и загрузите в свой гит. Ссылку на гит отправьте ментору.

**Выполнение задания.**

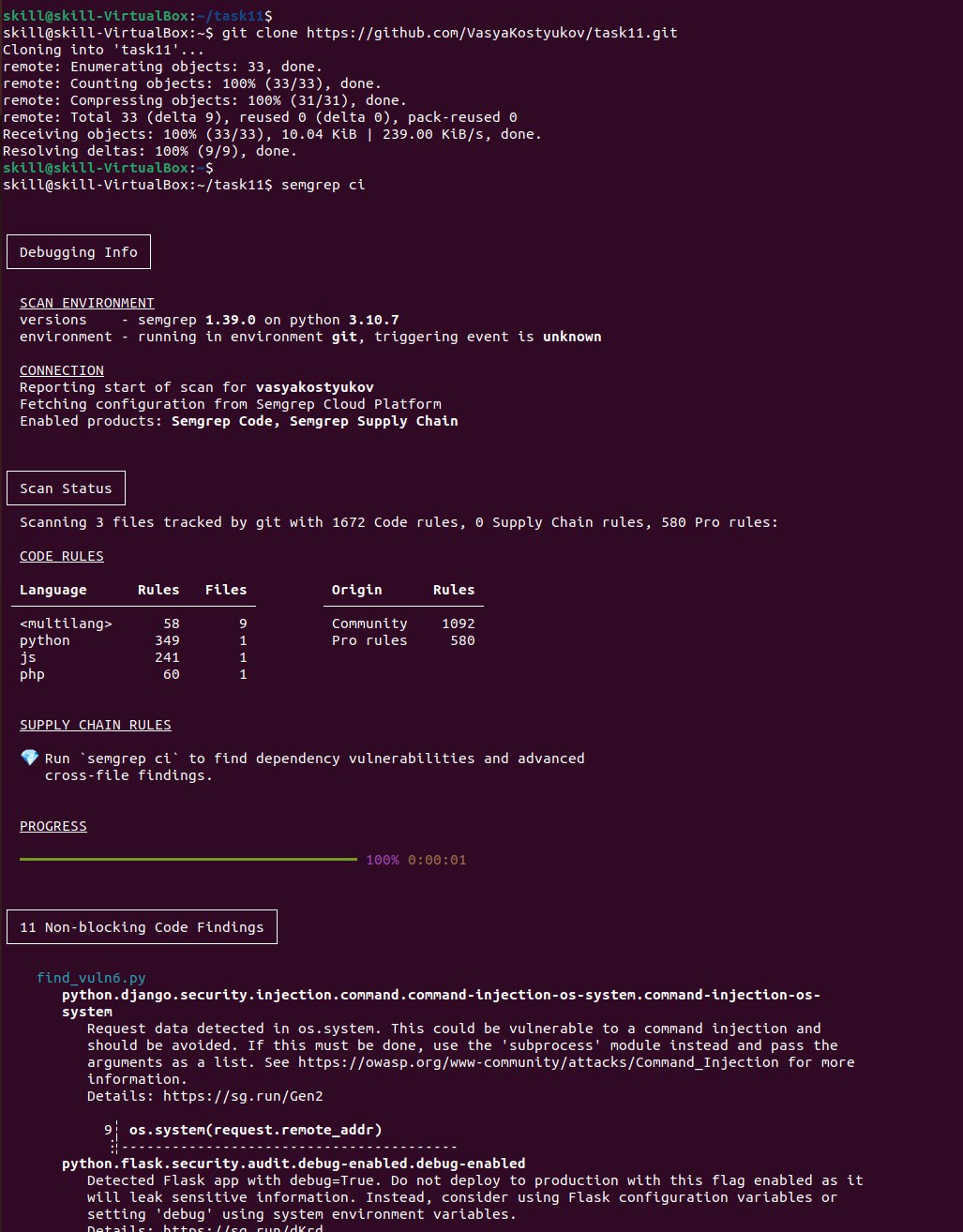
1. **Лабораторные работы в Joice Shop:**



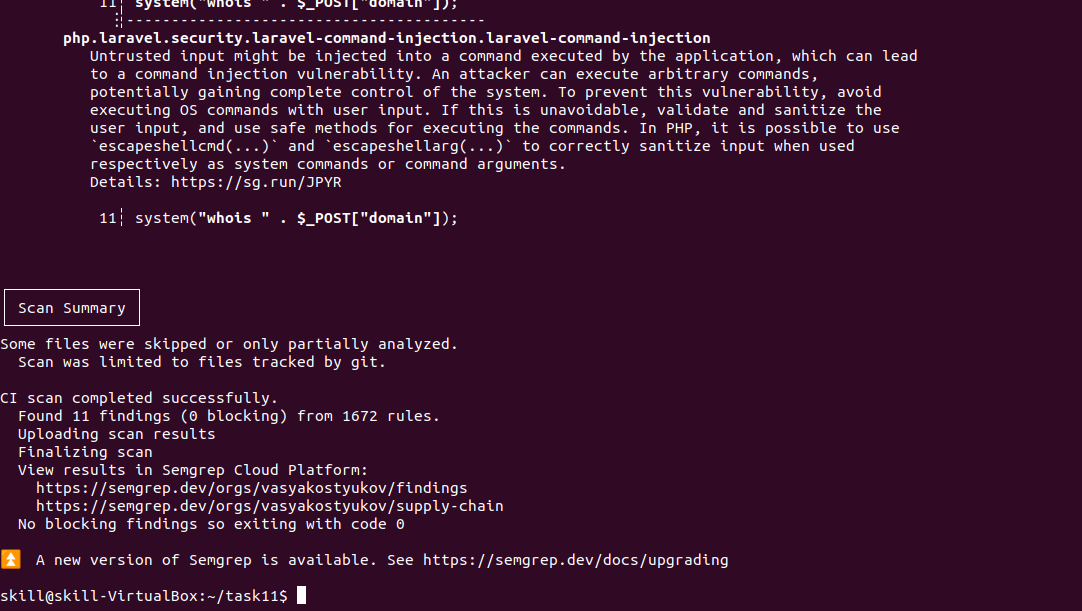


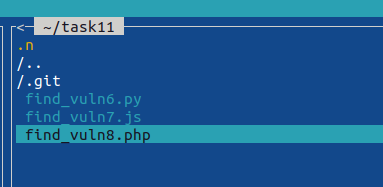
1. **Сканирование кода на уязвимости**

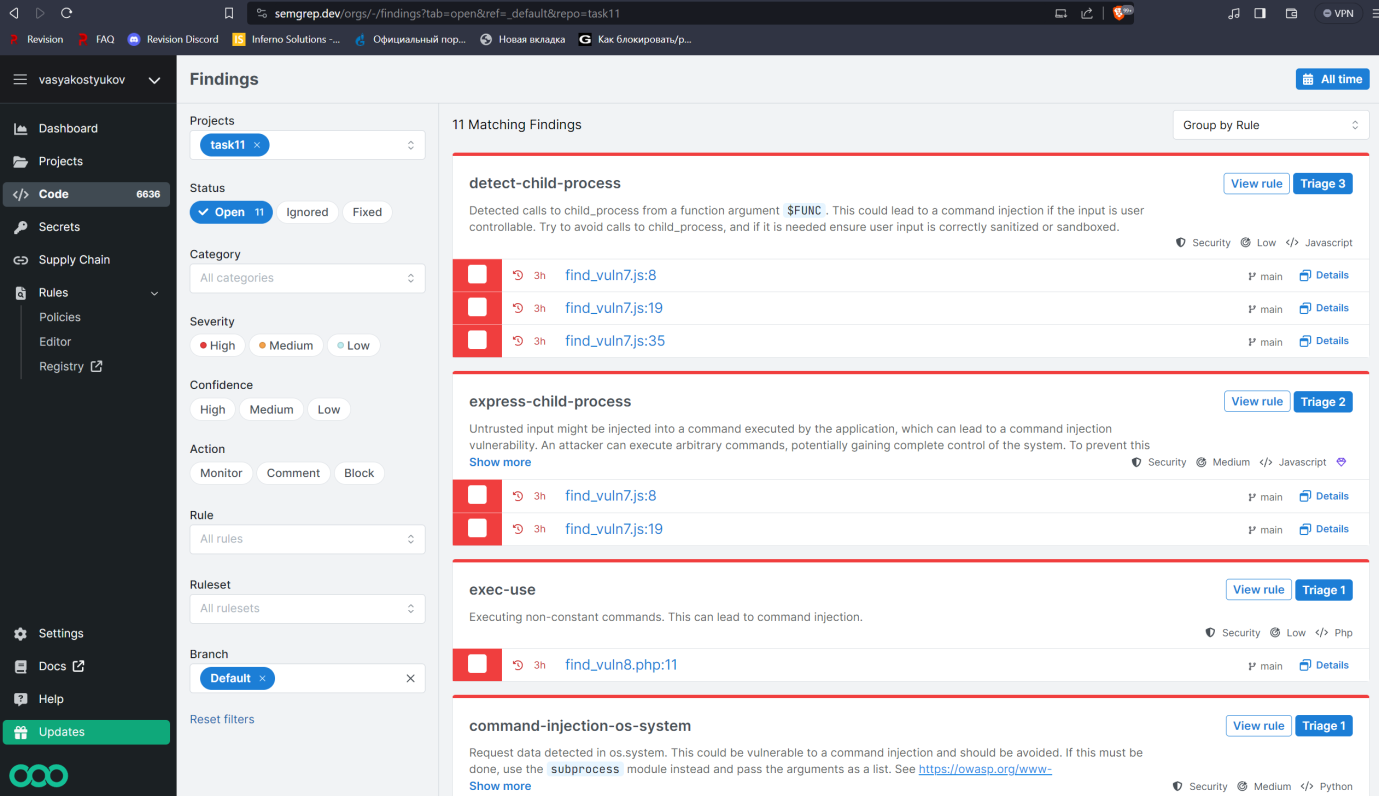
Я залил файлы с кодом на свой гитхаб. Затем скачал этот репозиторий на виртуальную машину с Ubuntu и установленным и настроенным Semgrep. Запустил сканирование командой semgrep ci чтобы видеть результат не только в терминале, но и в веб приложении semgrep.

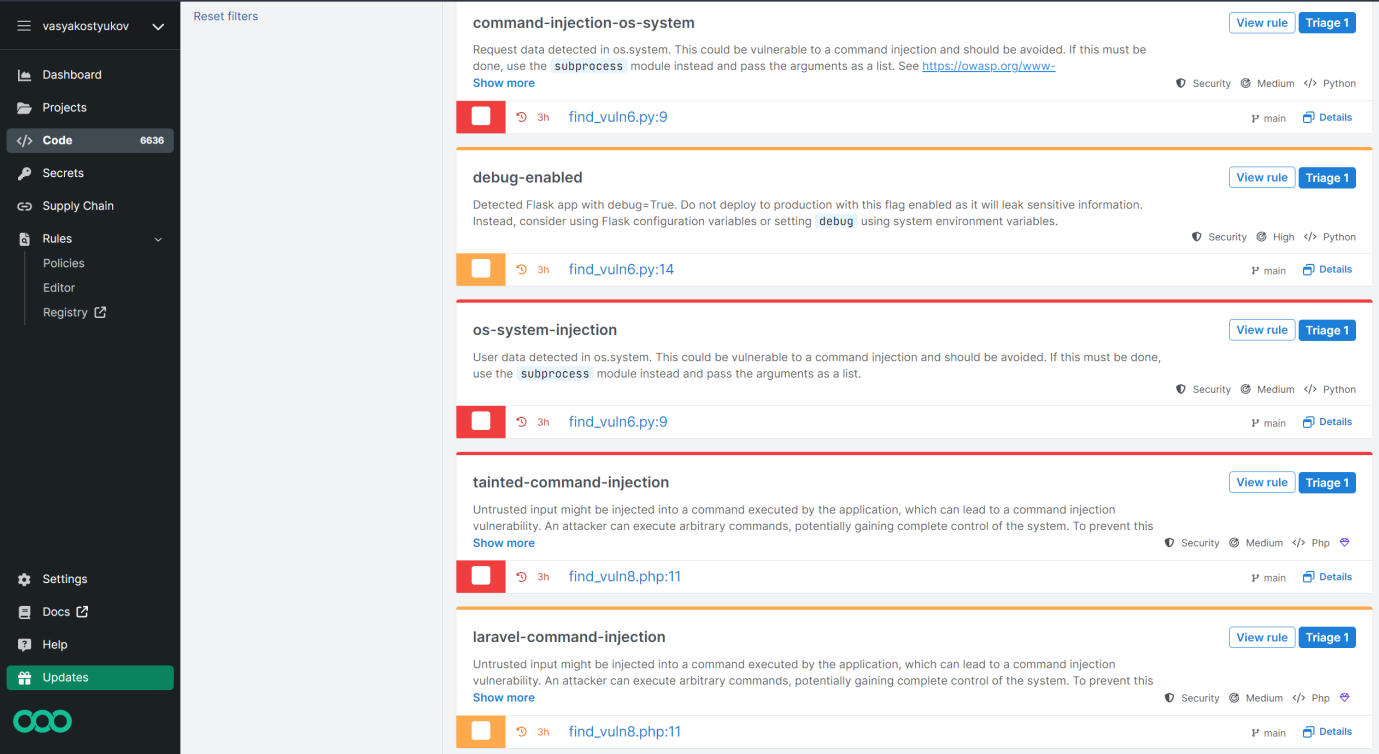




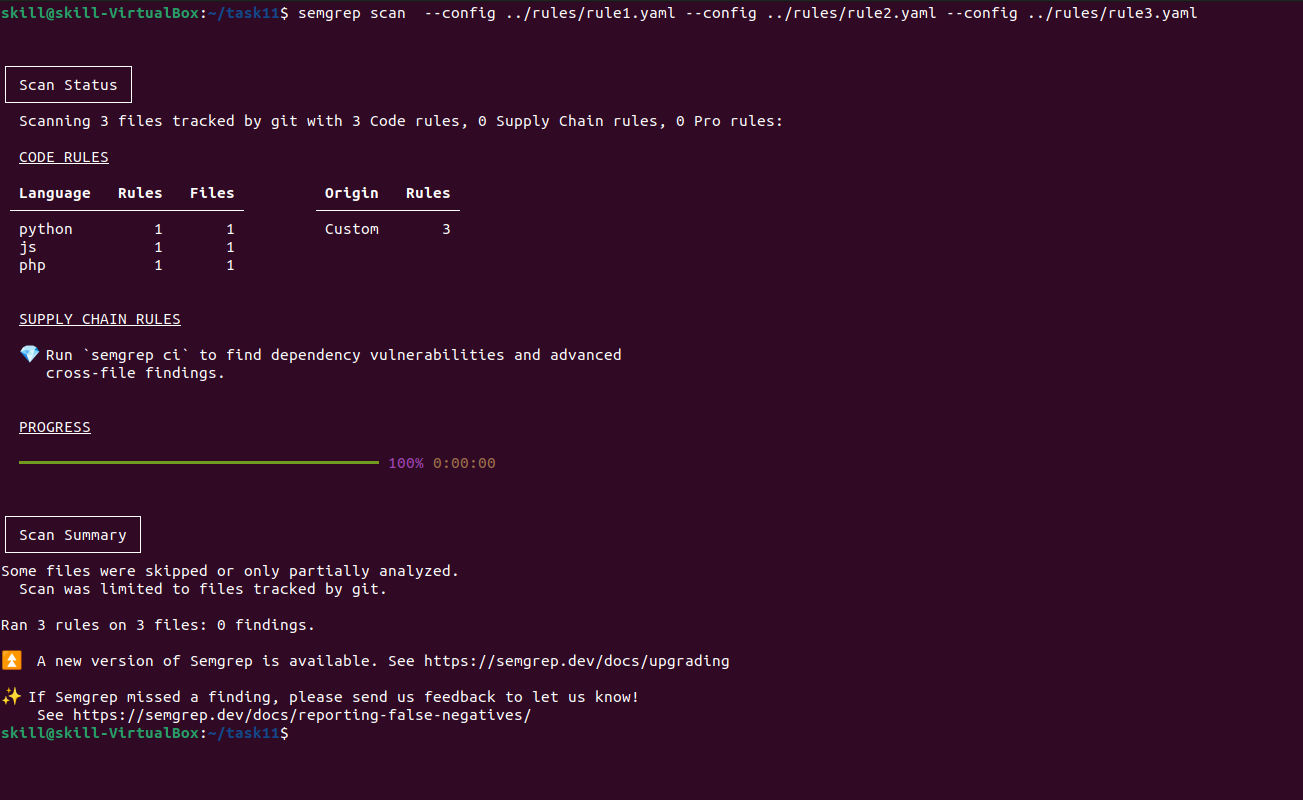








Уязвимостей Open Redirect в результате не нашлось ни в одном файле. Скачал из интернета несколько правил по поиску данной уязвимости, но результат остался прежний.



**Содержимое файла rule1.yaml:**

rules:

- id: open-redirect

patterns:

- pattern-inside: |

@$APP.route(...)

def $X(...):

...

- pattern-not-inside: |

@$APP.route(...)

def $X(...):

...

if <... werkzeug.urls.url\_parse($V) ...>:

...

- pattern-either:

- pattern: flask.redirect(<... flask.request.$W.get(...) ...>, ...)

- pattern: flask.redirect(<... flask.request.$W[...] ...>, ...)

- pattern: flask.redirect(<... flask.request.$W(...) ...>, ...)

- pattern: flask.redirect(<... flask.request.$W ...>, ...)

- pattern: |

$V = flask.request.$W.get(...)

...

flask.redirect(<... $V ...>, ...)

- pattern: |

$V = flask.request.$W[...]

...

flask.redirect(<... $V ...>, ...)

- pattern: |

$V = flask.request.$W(...)

...

flask.redirect(<... $V ...>, ...)

- pattern: |

$V = flask.request.$W

...

flask.redirect(<... $V ...>, ...)

- pattern-not: flask.redirect(flask.request.path)

- pattern-not: flask.redirect(flask.request.path + ...)

- pattern-not: flask.redirect(f"{flask.request.path}...")

message: Data from request is passed to redirect(). This is an open redirect and

could be exploited. Consider using 'url\_for()' to generate links to known

locations. If you must use a URL to unknown pages, consider using

'urlparse()' or similar and checking if the 'netloc' property is the same

as your site's host name. See the references for more information.

metadata:

cwe:

- "CWE-601: URL Redirection to Untrusted Site ('Open Redirect')"

owasp:

- A01:2021 - Broken Access Control

references:

- https://flask-login.readthedocs.io/en/latest/#login-example

- https://cheatsheetseries.owasp.org/cheatsheets/Unvalidated\_Redirects\_and\_Forwards\_Cheat\_Sheet.html#dangerous-url-redirect-example-1

- https://docs.python.org/3/library/urllib.parse.html#url-parsing

category: security

technology:

- flask

subcategory:

- audit

likelihood: LOW

impact: MEDIUM

confidence: LOW

license: Commons Clause License Condition v1.0[LGPL-2.1-only]

vulnerability\_class:

- Open Redirect

languages:

- python

severity: ERROR

**Содержимое файла rule2.yaml:**

rules:

- id: open-redirect-deepsemgrep

message: The application builds a URL using user-controlled input which can lead

to an open redirect vulnerability. An attacker can manipulate the URL and

redirect users to an arbitrary domain. Open redirect vulnerabilities can

lead to issues such as Cross-site scripting (XSS) or redirecting to a

malicious domain for activities such as phishing to capture users'

credentials. To prevent this vulnerability perform strict input validation

of the domain against an allowlist of approved domains. Notify a user in

your application that they are leaving the website. Display a domain where

they are redirected to the user. A user can then either accept or deny the

redirect to an untrusted site.

severity: WARNING

metadata:

likelihood: HIGH

impact: MEDIUM

confidence: HIGH

category: security

subcategory:

- vuln

cwe:

- "CWE-601: URL Redirection to Untrusted Site ('Open Redirect')"

functional-categories:

- web::source::cookie

- web::source::http-params

- web::source::http-body

- web::source::header

- web::sink::redirect

owasp:

- A01:2021 - Broken Access Control

references:

- https://cheatsheetseries.owasp.org/cheatsheets/Unvalidated\_Redirects\_and\_Forwards\_Cheat\_Sheet.html

technology:

- express

license: Copyright 2023 Semgrep, Inc.

vulnerability\_class:

- Open Redirect

languages:

- javascript

- typescript

mode: taint

pattern-sources:

- label: TAINTED

patterns:

- pattern-either:

- pattern-inside: function ... (...,$REQ, ...) {...}

- pattern-either:

- pattern: $REQ.query

- pattern: $REQ.body

- pattern: $REQ.params

- pattern: $REQ.cookies

- pattern: $REQ.headers

- label: TAINTED

patterns:

- pattern-either:

- pattern-inside: |

(...,{..., $REQ,... }: Request,...) => {...}

- pattern-inside: |

(...,{..., $REQ,... }: $EXPRESS.Request,...) => {...}

- focus-metavariable: $REQ

- pattern-either:

- pattern: params

- pattern: query

- pattern: cookies

- pattern: headers

- pattern: body

- label: CONCAT

requires: TAINTED

patterns:

- pattern-either:

- pattern: |

`...${$X}...`

- pattern: |

$F + $X

- pattern-not: |

`${$X}...`

- pattern-not: |

"https://" + $X

- pattern-not: |

"http://" + $X

- pattern-not: |

"//" + $X

- pattern-not: |

`http://${$X}...`

- pattern-not: |

`https://${$X}...`

- pattern-not: |

`//${$X}...`

- pattern-not: |

$X + $F

- focus-metavariable: $X

pattern-sinks:

- requires: TAINTED and not CONCAT

patterns:

- pattern-either:

- patterns:

- pattern-either:

- pattern-inside: $RES.redirect($QUERY,...)

- focus-metavariable: $QUERY

- patterns:

- pattern-either:

- pattern-inside: |

(...,{..., redirect,... }: Response,...) => {...}

- pattern-inside: >

(...,{..., redirect,... }: $EXPRESS.Response,...) =>

{...}

- pattern-inside: redirect($QUERY,...)

pattern-sanitizers:

- by-side-effect: true

patterns:

- pattern-either:

- pattern-inside: |

if (!$F.$REGEX($URL)) {

...

return ...

}

...

- pattern-inside: |

if ($F.$REGEX($URL)) {

...

}

- metavariable-regex:

metavariable: $REGEX

regex: ^(has|test)$

- focus-metavariable: $URL

**Содержимое файла rule3.yaml:**

rules:

- id: symfony-non-literal-redirect

patterns:

- pattern: $this->redirect(...)

- pattern-not: $this->redirect("...")

- pattern-not: $this->redirect()

message: The `redirect()` method does not check its destination in any way. If

you redirect to a URL provided by end-users, your application may be open

to the unvalidated redirects security vulnerability. Consider using

literal values or an allowlist to validate URLs.

languages:

- php

metadata:

references:

- https://symfony.com/doc/current/controller.html#redirecting

- https://cheatsheetseries.owasp.org/cheatsheets/Unvalidated\_Redirects\_and\_Forwards\_Cheat\_Sheet.html

owasp:

- A01:2021 - Broken Access Control

cwe:

- "CWE-601: URL Redirection to Untrusted Site ('Open Redirect')"

category: security

technology:

- symfony

subcategory:

- audit

likelihood: LOW

impact: MEDIUM

confidence: LOW

license: Commons Clause License Condition v1.0[LGPL-2.1-only]

vulnerability\_class:

- Open Redirect

severity: WARNING