\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
|  | **ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ**  **Кафедра**  **«Криптология и кибербезопасность»** |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ОТЧЕТ**

**о лабораторной работе №2**

Проведения ручного поиска уязвимостей пакетов приложения «Remediation-Demo»

|  |  |  |
| --- | --- | --- |
| Исполнитель: студент гр. Б21-515 |  | Тимин А. С. |
| Преподаватель: | подпись, дата | Карапетьянц М. |
|  | подпись, дата |  |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Москва — 2025**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc192592984)

[1 Пакеты приложения 3](#_Toc192592985)

[1.1 blinker 1.9.0 4](#_Toc192592986)

[1.2 click 8.1.8 5](#_Toc192592987)

[1.3 Flask 3.1.0 5](#_Toc192592988)

[1.4 flask-cors 5.0.1 6](#_Toc192592989)

[1.4.1 Improper Access Control 7](#_Toc192592990)

[1.5 itsdangerous 2.2.0 9](#_Toc192592991)

[1.6 Jinja2 3.1.6 10](#_Toc192592992)

[1.6.1 Template Injection 11](#_Toc192592993)

[1.7 MarkupSafe 3.0.2 12](#_Toc192592994)

[1.8 Werkzeug 3.1.3 13](#_Toc192592995)

[1.9 wheel 0.45.1 14](#_Toc192592996)

[1.10 Итоги 15](#_Toc192592997)

1 Пакеты приложения

Для точного определения пакетов приложения и их версий приложение было запущено с помощью docker compose, после чего было осуществлено подключение к контейнеру и вывод всех пакетов, установленных с помощью пакетного менеджера pip (рисунок 1.1).



Рисунок 1 – пакеты приложения

Как видно из скриншота, в контейнере установлены следующие пакеты, представленные в таблице 1.1.

Таблица 1 – установленные пакеты

|  |  |
| --- | --- |
| **Пакет** | **Версия** |
| blinker | 1.9.0 |
| click | 8.1.8 |
| Flask | 3.1.0 |
| flask-cors | 5.0.1 |
| Itsdangerous | 2.2.0 |
| Jinja2 | 3.1.6 |
| MarkupSafe | 3.0.2 |
| pip | 24.0 |
| Werkzeug | 3.1.3 |
| wheel | 0.45.1 |

Проанализируем каждый из пакетов на наличие уязвимостей с помощью сервисов: [nvd](https://nvd.nist.gov/products/cpe/search/) и [cnyk](https://security.snyk.io/). Данные актуальны 11 марта 2025 года.

1.1 blinker 1.9.0

Поиск по базе данных [NVD](https://nvd.nist.gov/products/cpe/search/results?namingFormat=2.3&keyword=blinker+1.9.0) не выявил актуальных для данной версии пакета уязвимостей (рисунок 2).

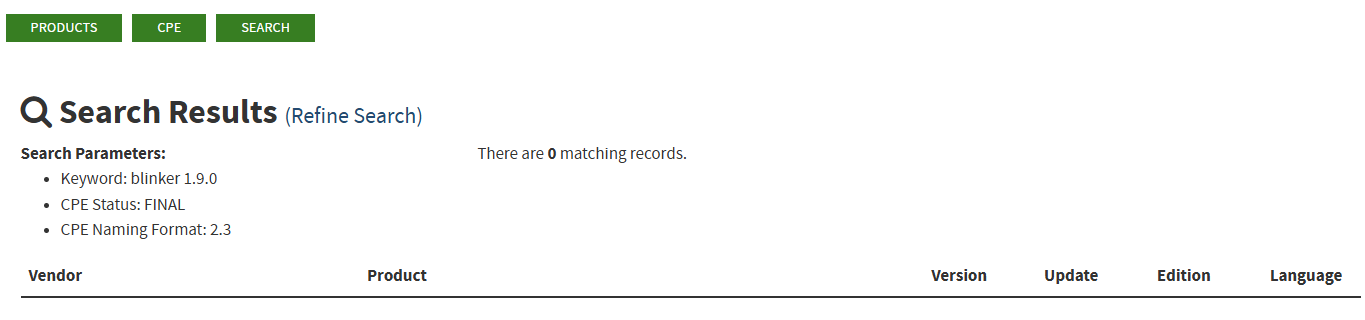


Рисунок 2 – актуальные уязвимости blinker 1.9.0 в NVD

Аналогичная ситуация и с поиском через сервис [SNYK](https://security.snyk.io/package/pip/blinker) (рисунок 3).

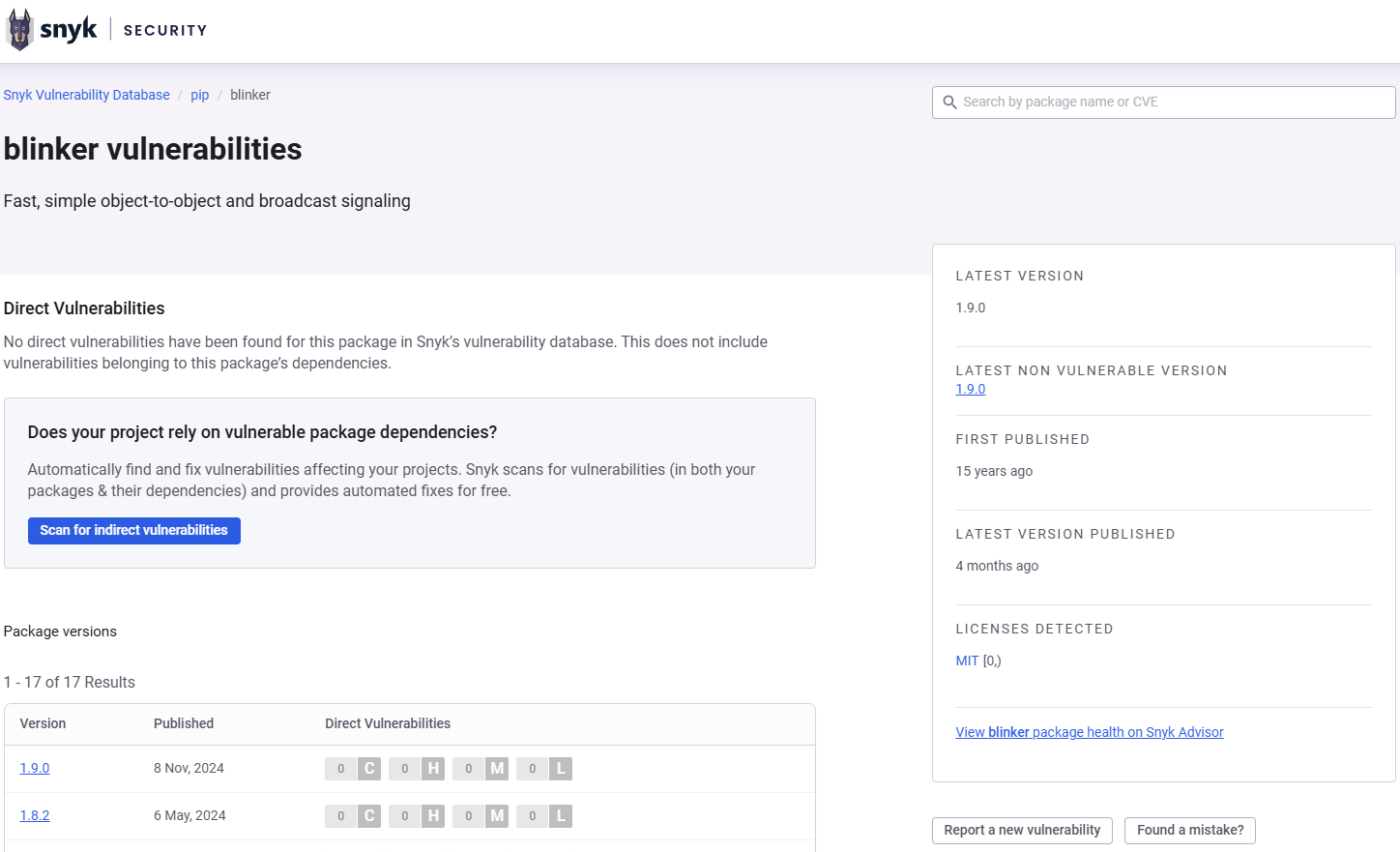


Рисунок 3 – актуальные уязвимости blinker 1.9.0 в SNYK

Выходит, что пакет blinker 1.9.0 не содержит актуальных выявленных уязвимостей.

1.2 click 8.1.8

С пакетом click 8.1.8 – аналогичный результат поиска по базам данных [NVD](https://nvd.nist.gov/products/cpe/search/results?namingFormat=2.3&keyword=click+8.1.8) и [SNYK](https://security.snyk.io/package/pip/click) (рисунки 4 и 5).

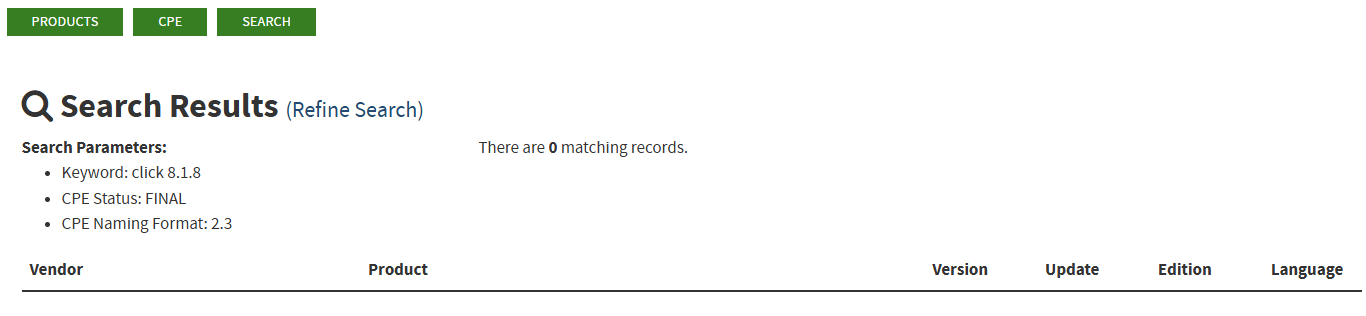


Рисунок 4 – актуальные уязвимости click 8.1.8 в NVD

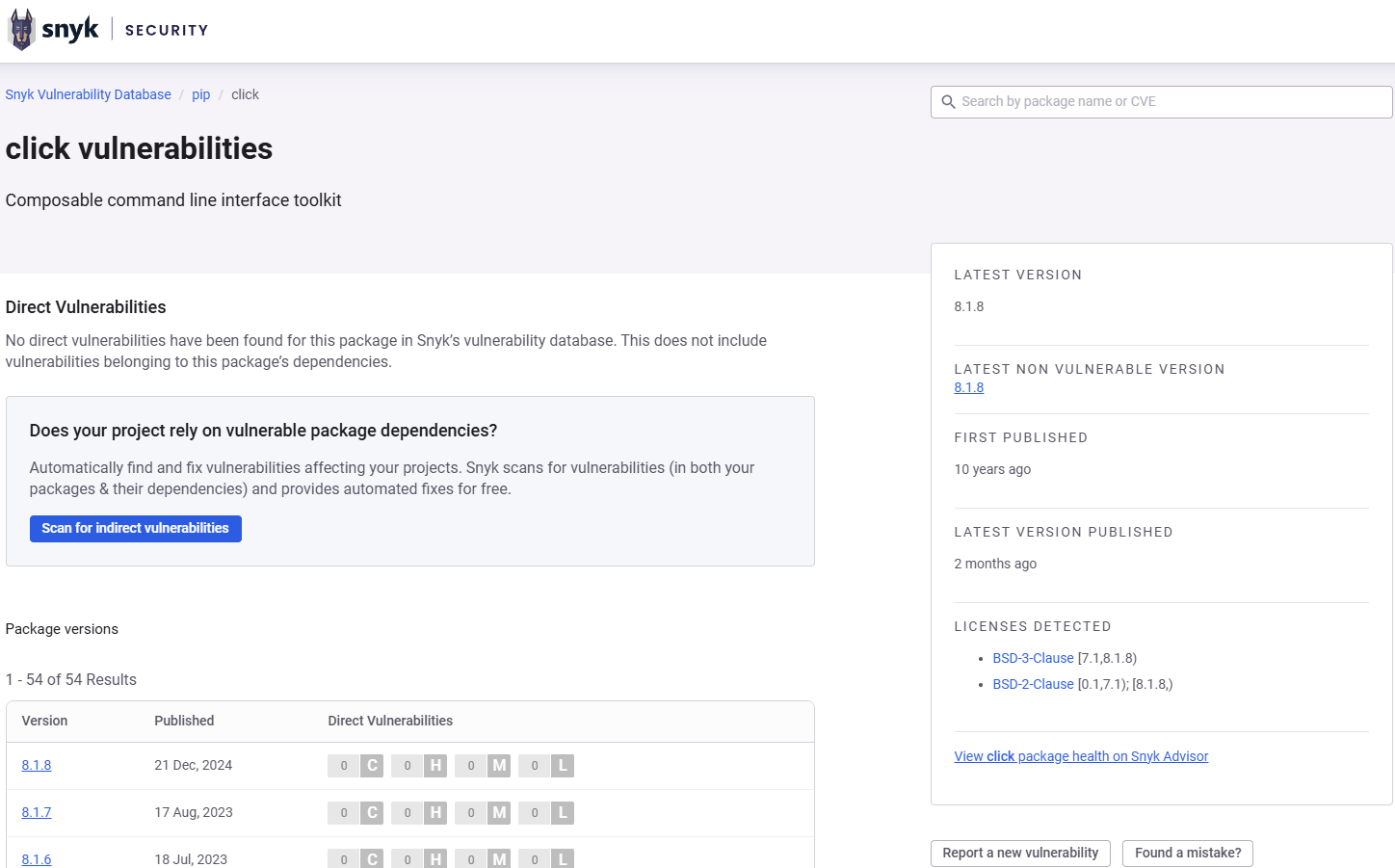


Рисунок 5 – актуальные уязвимости click 8.1.8 в SNYK

1.3 Flask 3.1.0

Касаемо пакета Flask 3.1.0, в ьазе данных [NVD](https://nvd.nist.gov/products/cpe/search/results?namingFormat=2.3&keyword=flask+3.1.0) нет сведений об уязвимостях самого пакета, хотя имеются уязвимости в проектах, использующих данный пакет (рисунок 6).



Рисунок 6 – актуальные уязвимости Flask 3.1.0 в NVD

Поиск по базе данных сервиса [CNYK](https://security.snyk.io/package/pip/flask) также не выявил актуальных уязвимостей, однако некоторые старые версии пакета были не так безопасны (рисунок 7).

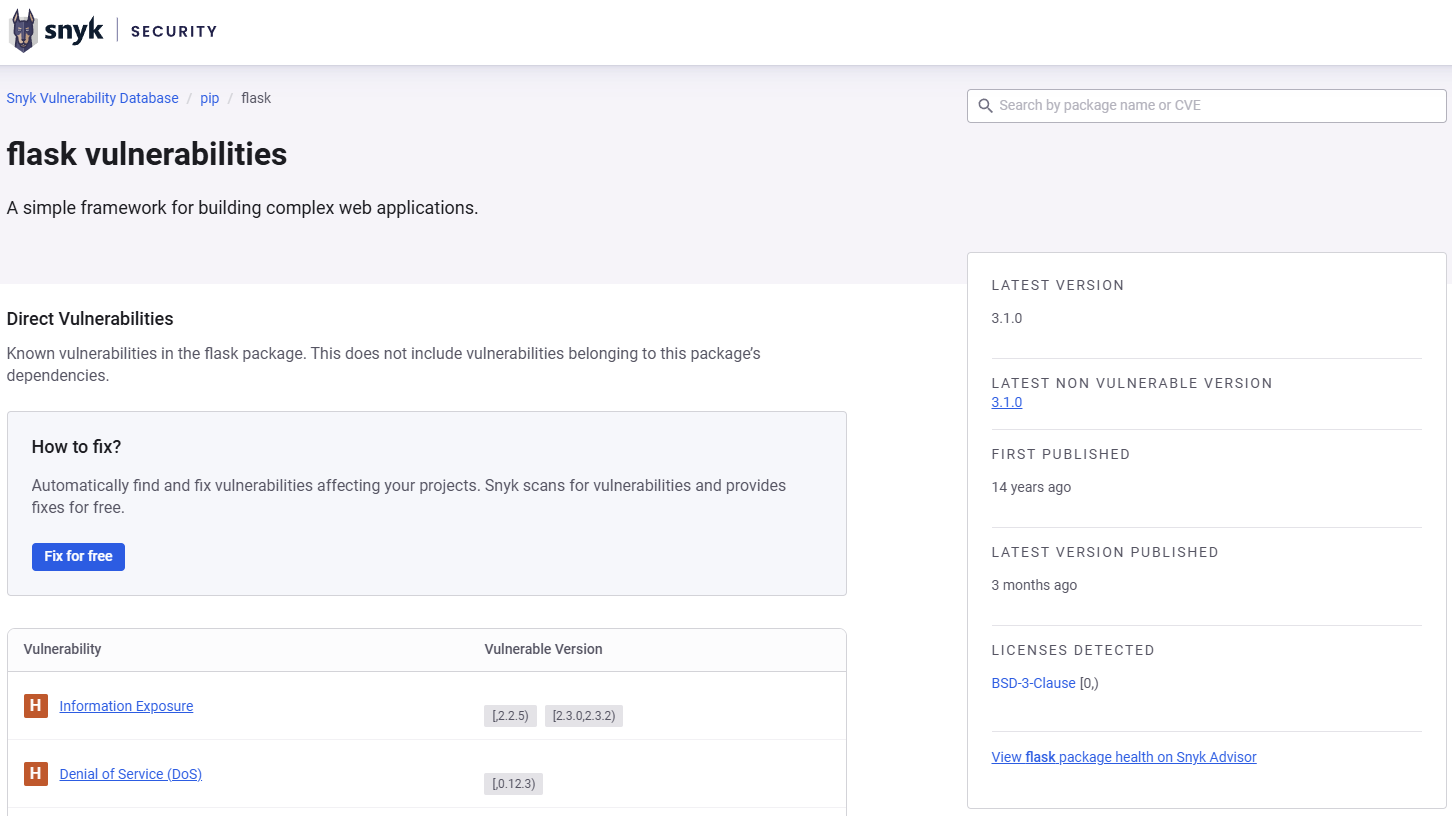


Рисунок 7 – актуальные уязвимости Flask 3.1.0 в SNYK

1.4 flask-cors 5.0.1

База данных сервиса [NVD](https://nvd.nist.gov/products/cpe/search/results?namingFormat=2.3&keyword=flask-cors+5.0.1) не содержит информации об актуальных уязвимостях пакета flask-cors 5.0.1 (рисунок 8).

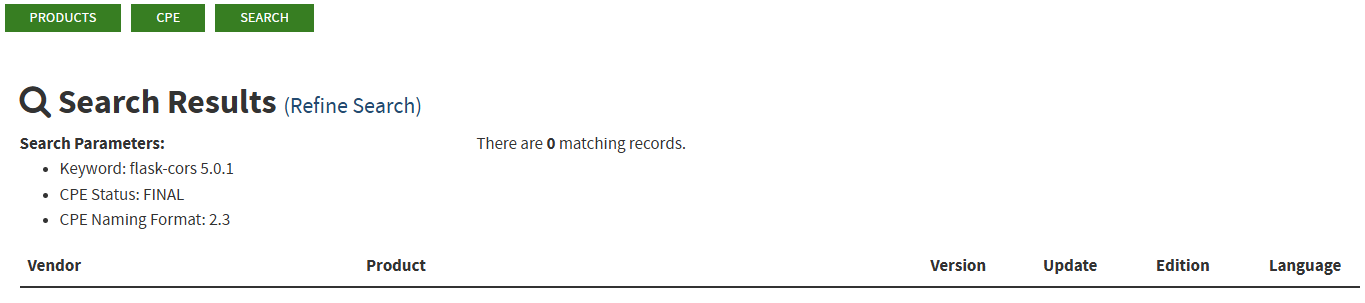


Рисунок 8 – актуальные уязвимости flask-cors 5.0.1 в NVD

Аналогичная ситуация и с результатом поиска на сервисе [SNYK](https://security.snyk.io/package/pip/flask-cors) (рисунок 9).

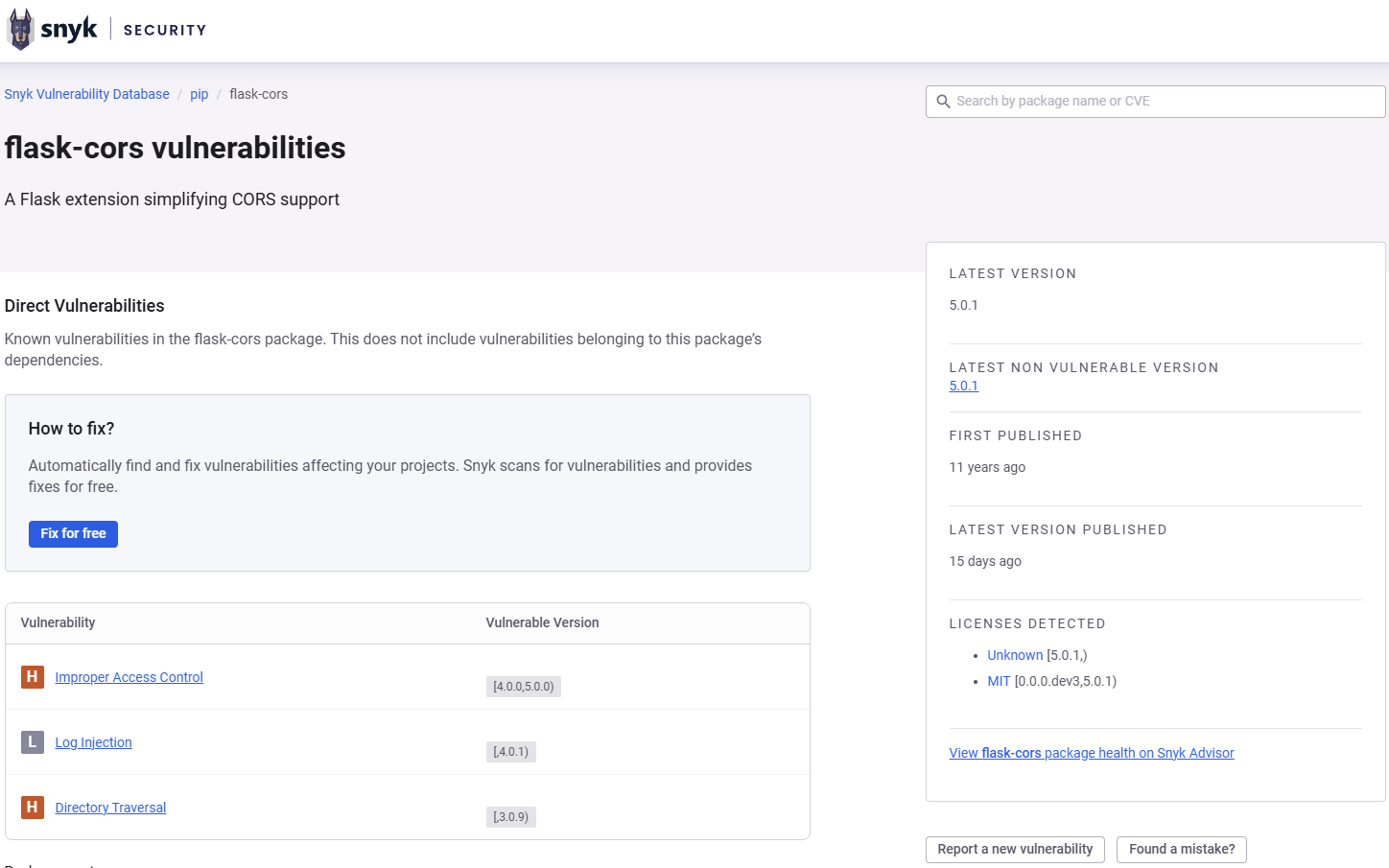


Рисунок 9 – актуальные уязвимости flask-cors 5.0.1 в SNYK

Из скриншота видно, что до версии 5.0.0 пакет flask-cors имел уязвимость [Improper Access Control](https://security.snyk.io/vuln/SNYK-PYTHON-FLASKCORS-7707876). Проанализируем ее подробнее.

1.4.1 Improper Access Control

Кратко об уязвимости Improper Access Control:

* уязвимость: неправильный контроль доступа;
* затронутые версии: 4.0.0–5.0.0;
* исправлено в версии: 5.0.0 и выше;
* критичность: CVSS 4.0 – 7.1 (High).

Данная уязвимость связана с неправильной настройкой CORS в расширении Flask-Cors. По умолчанию в уязвимых версиях включена поддержка заголовка Access-Control-Allow-Private-Network, что может привести к разглашению ресурсов внутренней сети (Intranet) через браузерные запросы. Это позволяет злоумышленнику взаимодействовать с внутренними сервисами, если пользователь случайно переходит по вредоносной ссылке.

Если приложение с Flask-Cors обрабатывает запросы от неизвестных источников, злоумышленник может отправить запросы к локальным ресурсам (например, 192.168.x.x, 10.x.x.x, localhost) через браузер жертвы.

Эксплуатация уязвимости позволяет:

* собирать информацию о внутренних сервисах;
* получать доступ к защищенным API;
* обходить меры защиты, такие как Same-Origin Policy.

Если жертва откроет вредоносный сайт, он сможет выполнить запрос, обращающийся к внутренним ресурсам:

fetch("http://192.168.1.1/admin", { credentials: "include" })

.then(response => response.text())

.then(data => console.log(data));

Если сервер с Flask-Cors на уязвимой версии разрешает такие запросы, злоумышленник получит доступ к ресурсам внутренней сети.

Существует несколько способов устранить уязвимость:

* обновить Flask-Cors до версии 5.0.0 или выше. В этой версии изменена обработка Access-Control-Allow-Private-Network, что исключает автоматический доступ к внутренним ресурсам;
* настроить CORS вручную, если обновление невозможно. Если обновить библиотеку нельзя, в коде нужно жестко ограничить источники запросов;
* запретить заголовок Access-Control-Allow-Private-Network в middleware перед отправкой ответа.

1.5 itsdangerous 2.2.0

В базах данных [NVD](https://nvd.nist.gov/products/cpe/search/results?namingFormat=2.3&keyword=itsdangerous+2.2.0) и [SNYK](https://security.snyk.io/package/pip/itsdangerous) не выявлено актуальных уязвимостей пакета itsdangerous версии 2.2.0 (рисунки 10 и 11).

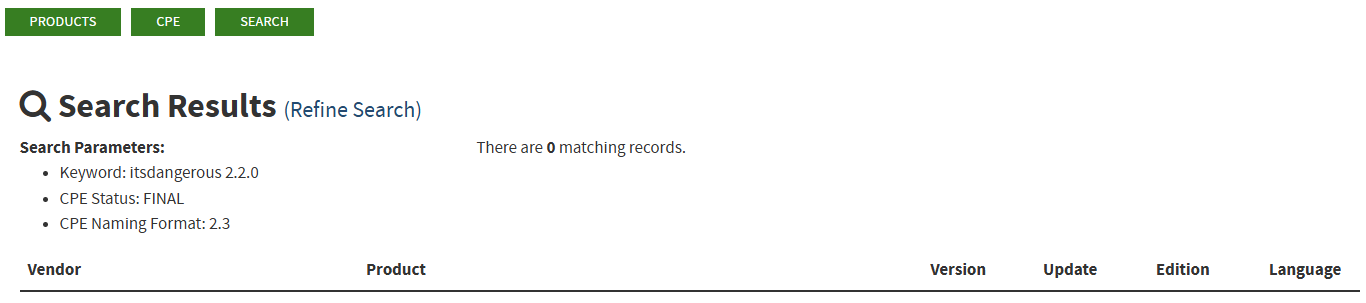


Рисунок 10 – актуальные уязвимости itsdangerous 2.2.0 в NVD

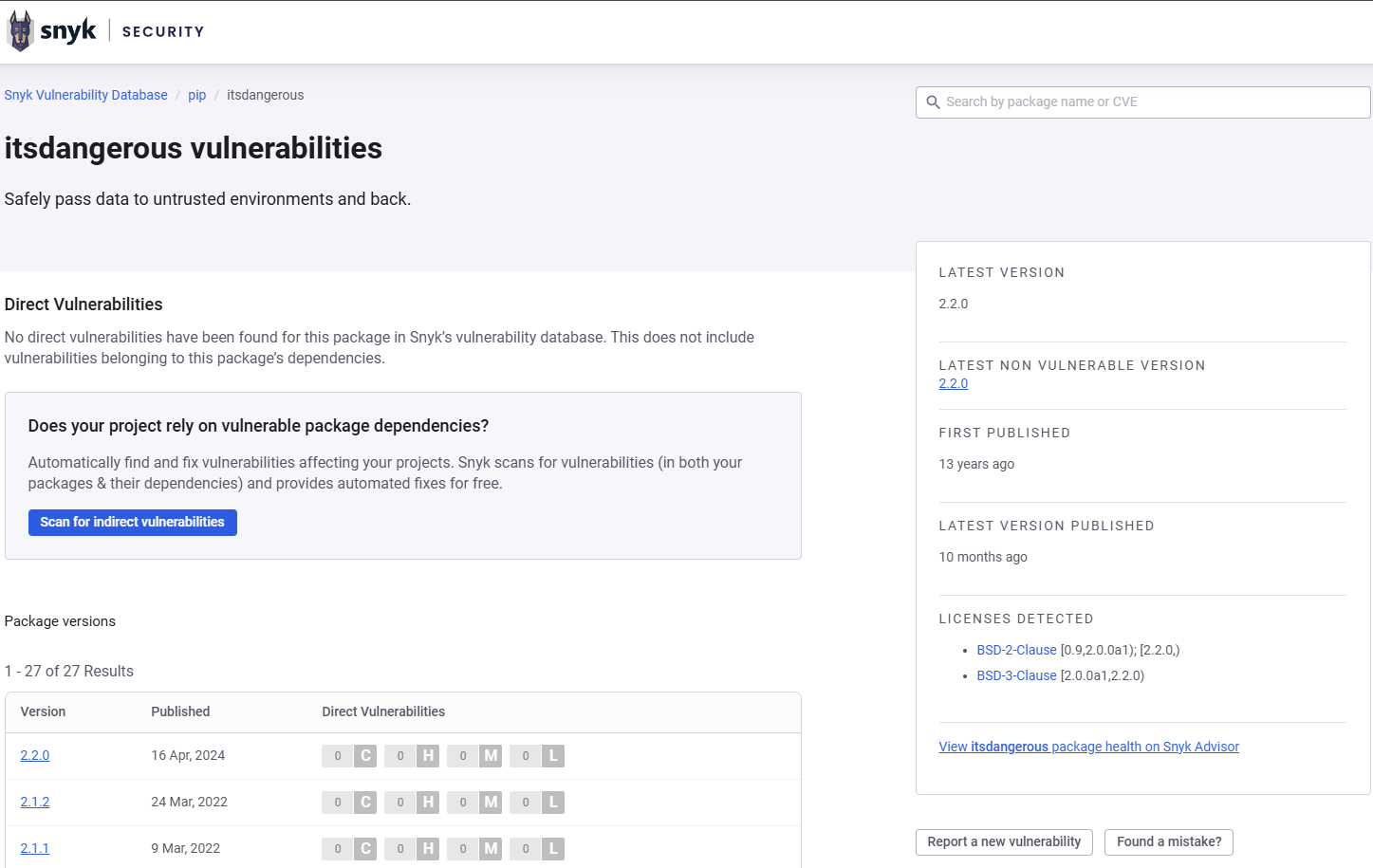


Рисунок 11 – актуальные уязвимости itsdangerous 2.2.0 в SNYK

1.6 Jinja2 3.1.6

Аналогичный результат поиска уязвимостей пакета Jinja2 3.1.6 в базах данных [NVD](https://nvd.nist.gov/products/cpe/search/results?namingFormat=2.3&keyword=Jinja2+3.1.6) и [SNYK](https://security.snyk.io/package/pip/jinja2) (рисунки 12 и 13).

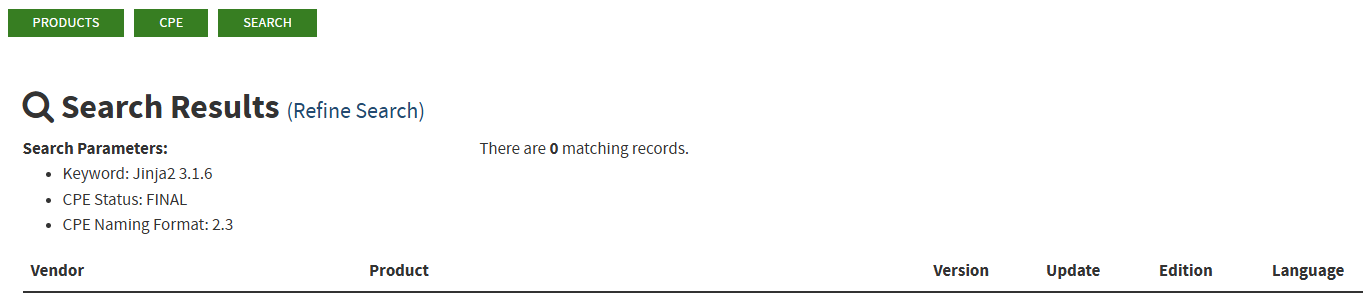


Рисунок 12 – актуальные уязвимости Jinja2 3.1.6 в NVD

Однако лишь в текущей версии пакета была исправлена уязвимость [Template Injection](https://security.snyk.io/vuln/SNYK-PYTHON-JINJA2-9292516), рассмотрим ее подробней.

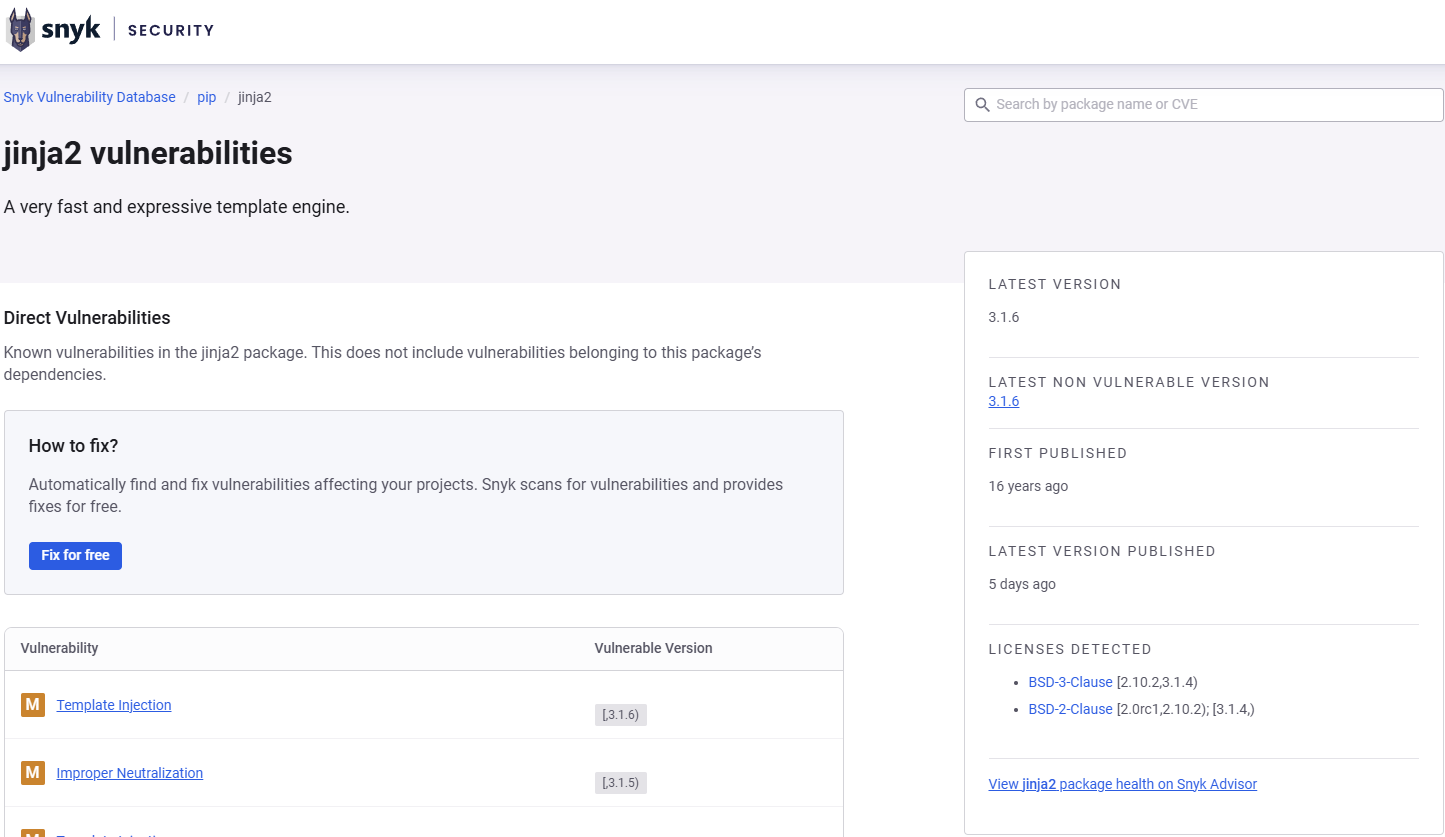


Рисунок 13 – актуальные уязвимости Jinja2 3.1.6 в SNYK

1.6.1 Template Injection

Обнаруженная уязвимость в пакете Jinja2 связана с инъекцией шаблонов (Template Injection). Уязвимые версии (до 3.1.6) допускают выполнение произвольного кода через фильтр |attr, который позволяет злоумышленнику обойти защитные механизмы среды и получить доступ к критическим функциям Python. Критичность: CVSS 4.0: 5.4 (Medium).

Уязвимость может привести к выполнению произвольного кода на сервере, если злоумышленник имеет возможность передавать пользовательские шаблоны в движок Jinja2. Это делает атаку особенно опасной в веб-приложениях, которые рендерят шаблоны с динамическим вводом.

Условия эксплуатации:

* приложение использует Jinja2 и позволяет пользователям загружать или изменять шаблоны;
* приложение не ограничивает список доступных фильтров в Jinja2;
* отсутствуют дополнительные меры защиты, такие как строгая валидация вводимых данных.

Следующий код приводит к выполнению команды id на сервере:

{{ "".format.\_\_globals\_\_['os'].system('id') }}

С учетом использования фильтра |attr, злоумышленник может вызвать уязвимые методы следующим образом:

{{ ''|attr('\_\_class\_\_')|attr('\_\_mro\_\_')[-1]|attr('\_\_subclasses\_\_')() }}

Этот вызов позволяет получить список всех классов в системе, после чего можно найти и использовать subprocess.Popen для выполнения команд.

Существует ряд способов исправления уязвимости:

* обновить Jinja2 до версии 3.1.6 или выше;
* ограничить использование пользовательских шаблонов: запрещать передачу динамически изменяемых шаблонов от пользователей или использовать предопределенные шаблоны, хранящиеся в безопасных местах;
* отключить фильтр attr через Environment.

1.7 MarkupSafe 3.0.2

В базах данных [NVD](https://nvd.nist.gov/products/cpe/search/results?namingFormat=2.3&keyword=MarkupSafe+3.0.2) и [SNYK](https://security.snyk.io/package/pip/markupsafe) не обнаружено сведений об актуальных уязвимостях пакета MarkupSafe 3.0.2 (рисунки 14 и 15).

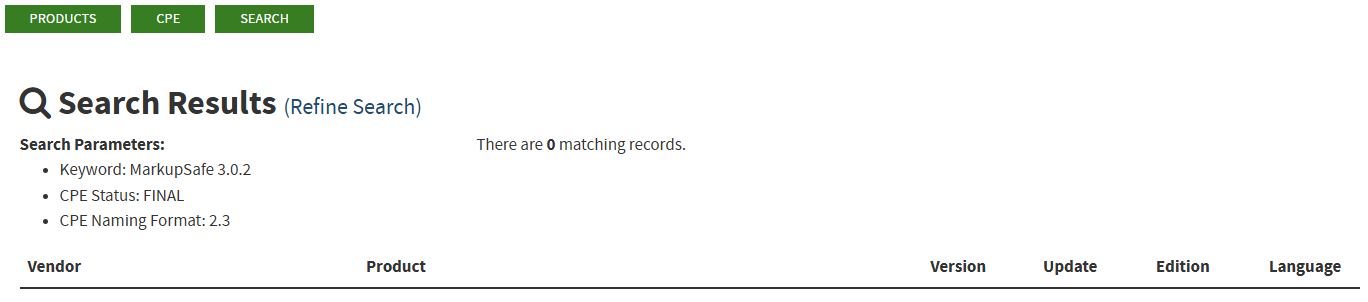


Рисунок 14 – актуальные уязвимости MarkupSafe 3.0.2 в NVD

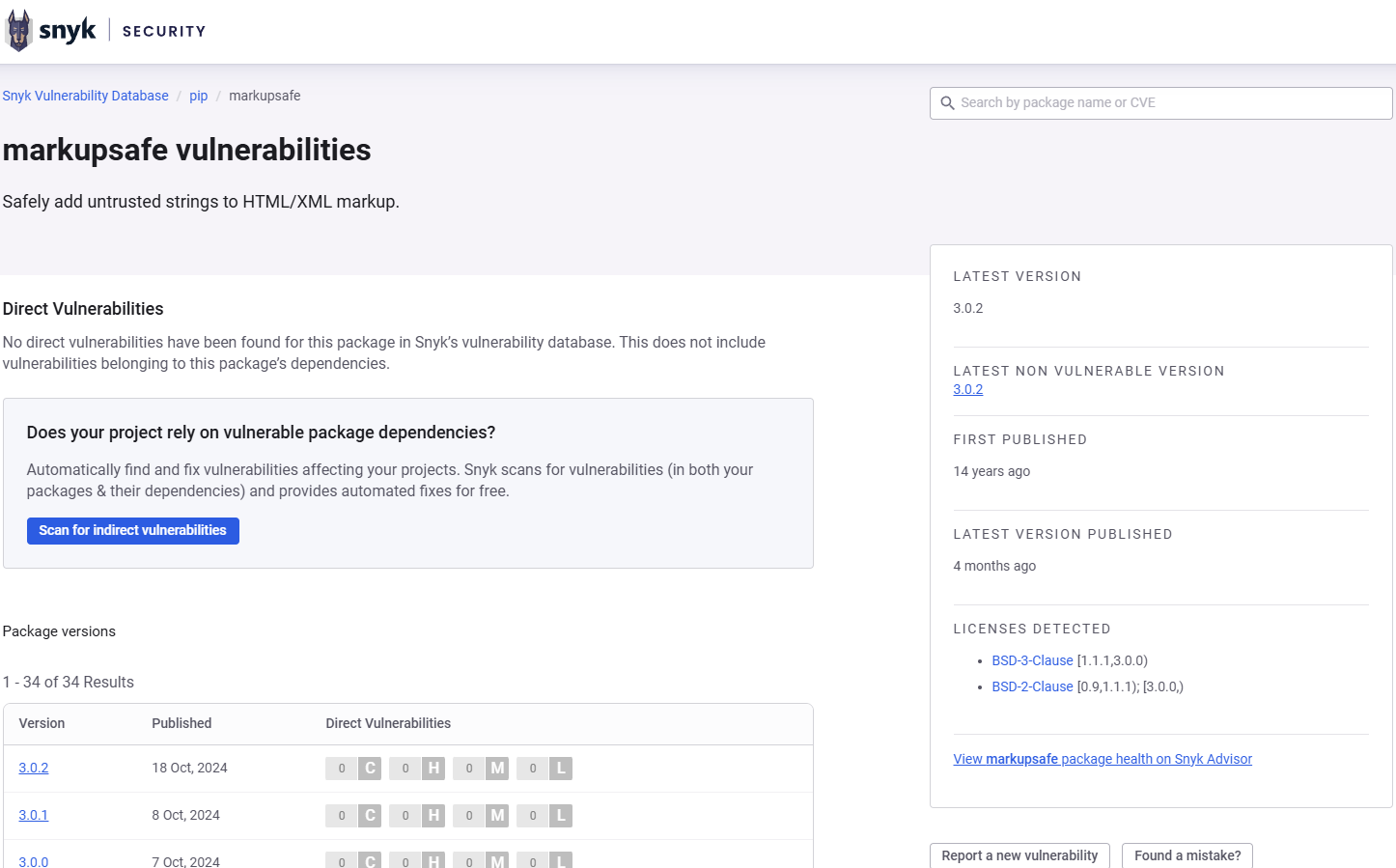


Рисунок 15 – актуальные уязвимости MarkupSafe 3.1.2 в SNYK

1.8 Werkzeug 3.1.3

Аналогичный результат по поиску уязвимостей пакета Werkzeug 3.1.3 по базам данных [NVD](https://nvd.nist.gov/products/cpe/search/results?namingFormat=2.3&keyword=Werkzeug+3.1.3) и [SNYK](https://security.snyk.io/package/pip/werkzeug) (рисунки 16 и 17).

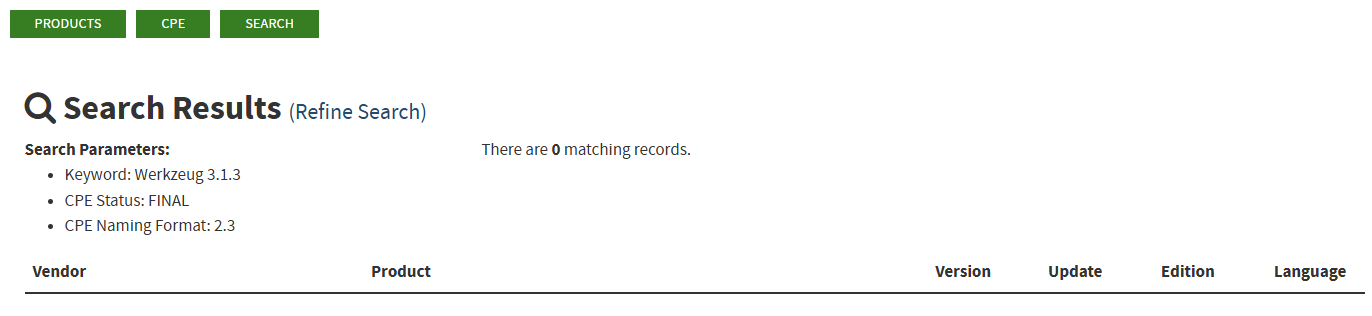


Рисунок 16 – актуальные уязвимости Werkzeug 3.1.3 в NVD

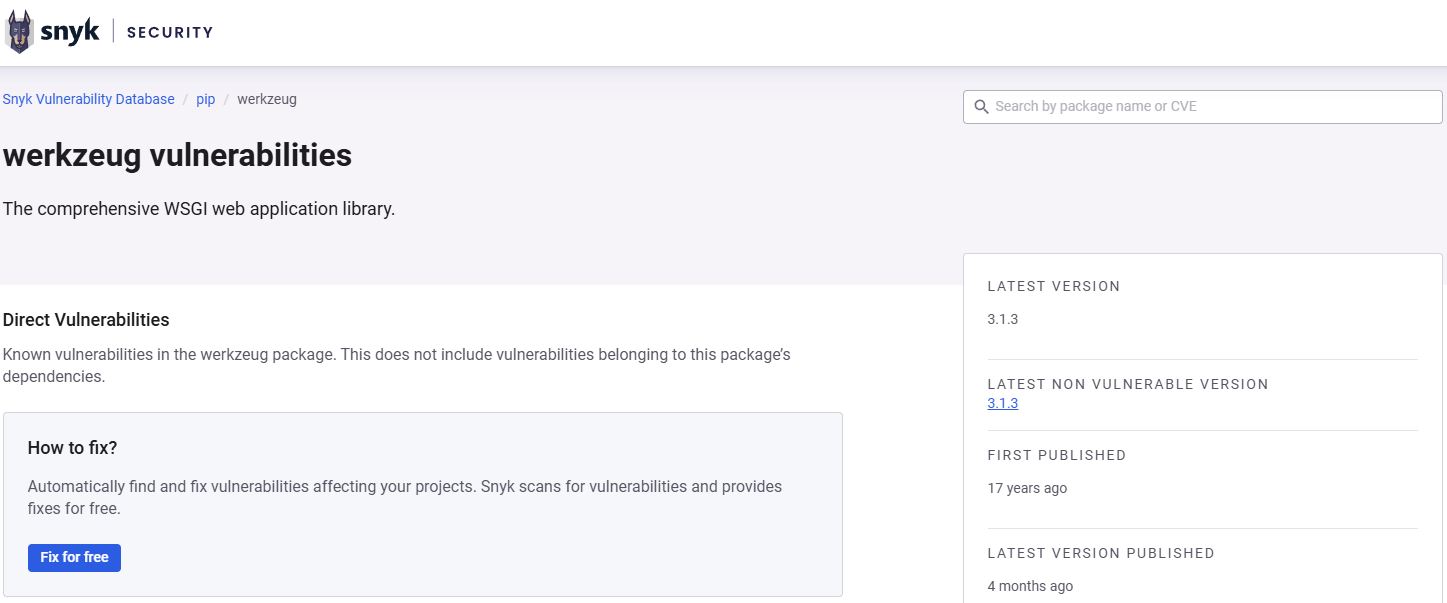


Рисунок 17 – актуальные уязвимости Werkzeug 3.1.3 в SNYK

1.9 wheel 0.45.1

Такая же ситуация и с пакетом wheel 0.45.1 – поиск по базам данных [NVD](https://nvd.nist.gov/products/cpe/search/results?namingFormat=2.3&keyword=wheel+0.45.1) и [SNYK](https://security.snyk.io/package/pip/wheel) не выявил актуальных уязвимостей (рисунки 18 и 19).

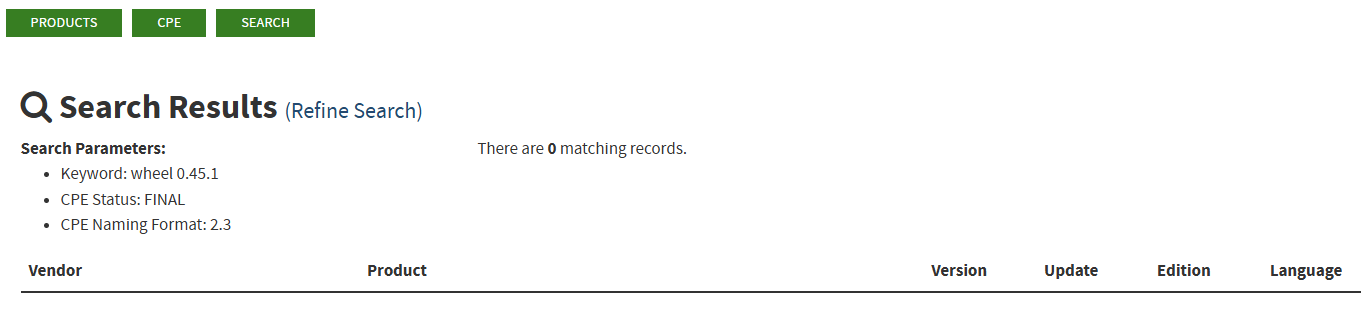


Рисунок 18 – актуальные уязвимости wheel 0.45.1 в NVD

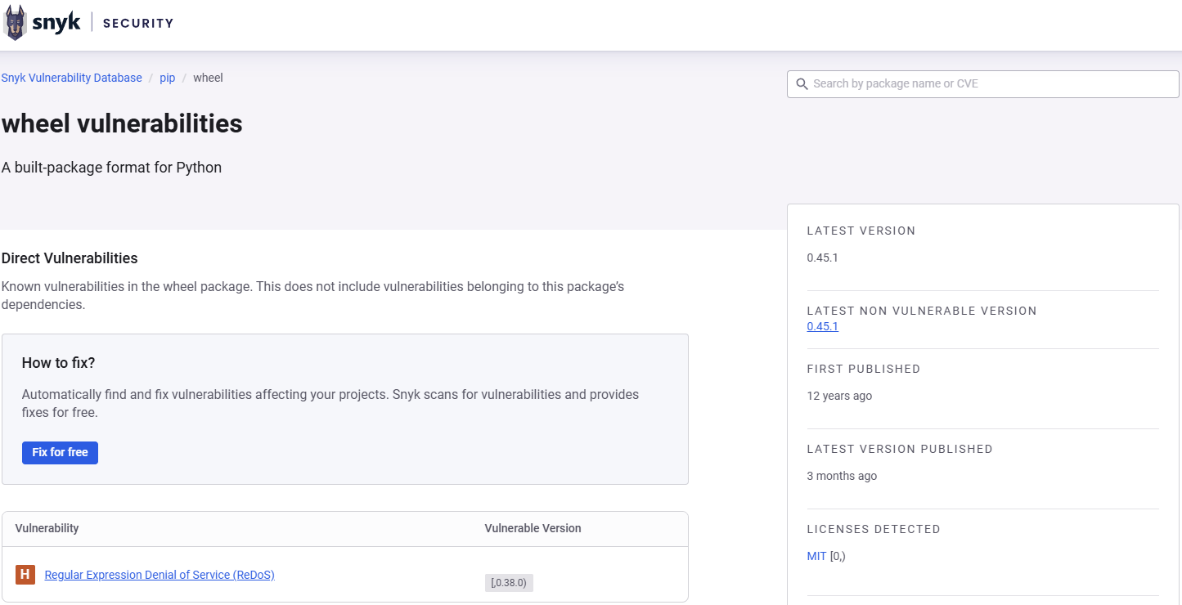


Рисунок 19 – актуальные уязвимости wheel 0.45.1 в SNYK

1.10 Итоги

Проанализировав по открытым источникам пакеты, используемые в приложении Remediation-Demo, можно сделать вывод, что оно безопасно, так как в своей работе использует новейшие версии зависимостей. При этом назвать его абсолютно безопасным никак нельзя, потому что существует вероятность, что ряд уязвимостей просто еще не выявлен, а также имеются серьезные ошибки в исходном коде, позволяющие эксплуатировать уязвимости: SQL-инъекции и утечку конфиденциальных данных (однако эти уязвимости выходят за рамки темы настоящей лабораторной работы).