Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

«Информационная безопасность»

Работа 2:
Анализ и устранение уязвимости на примере реального CVE с использованием Vulhub

Выполнил:

Румский Александр Максимович

Группа:

P3407

Преподаватель:

Маркина Татьяна Анатольевна

Санкт-Петербург, 2025 год

Содержание

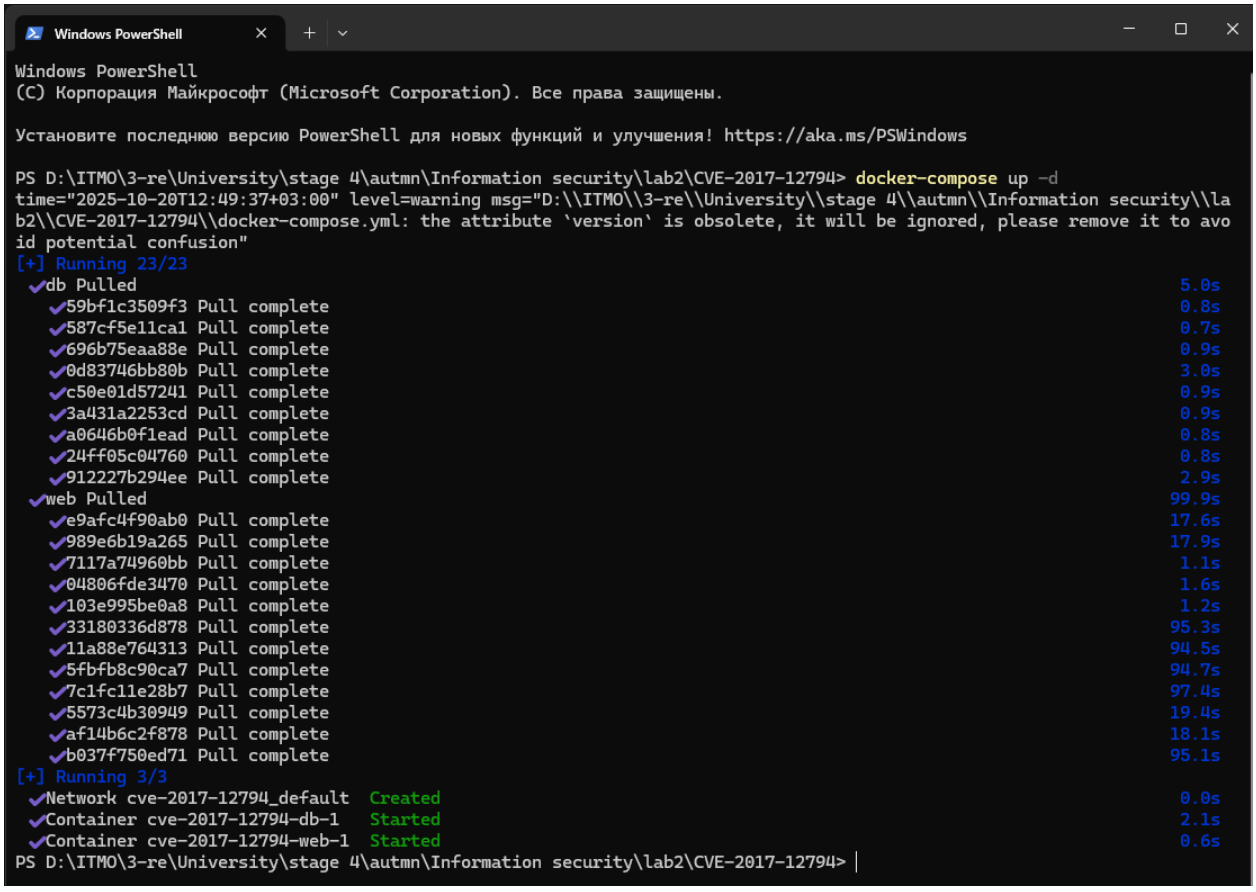
Название выбранной уязвимости (CVE ID) и краткое ее описание

CVE-2017-12794

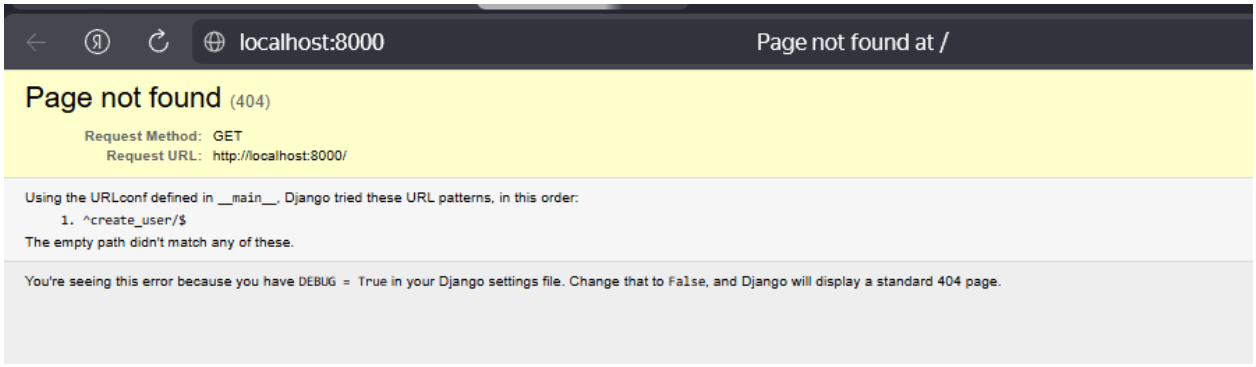
Уязвимость межсайтового скриптинга (XSS) на отладочных страницах с ошибками. При включении режима отладки в проекте, страницы с ошибкой могут потенциально раскрыть конфиденциальную информацию через неэкранированный HTML-код в сообщении об ошибке.

Последовательность действий по воспроизведению уязвимости

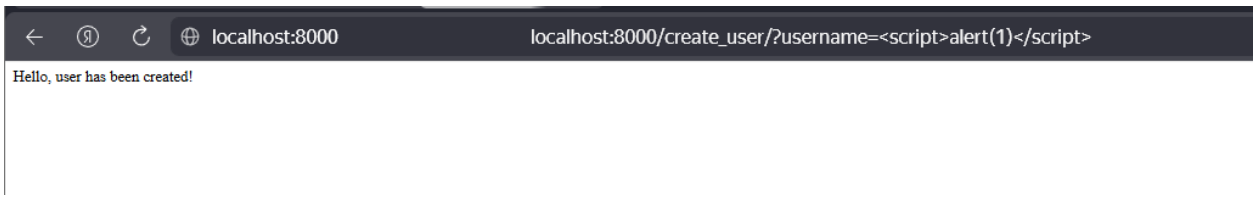
1. Создание docker-контейнера для запуска уязвимости



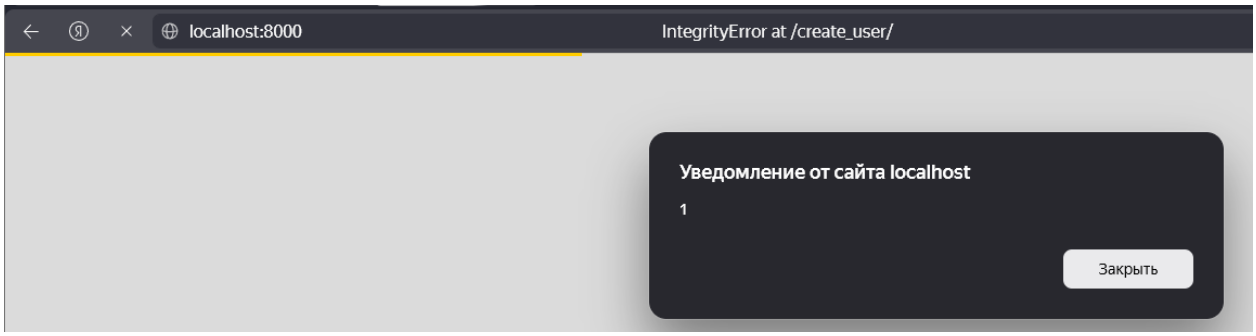
2. Проверить доступность страницы http://localhost:8000



3. Выполнить запрос к бэкенду, демонстрирующий уязвимость
   http://localhost:8000/create_user/?username=<script>alert(1)</script>
   Данный запрос, выполненный в первый раз, создаст пользователя



4. Повторный запрос вызовет срабатывание кода, переданного как имя пользователя
   Причина: Повторный вызов повлечет ошибку в БД, связанную с ограничением на уникальность имен пользователей, что, в свою очередь, вызовет страницу с отладочной информацией, на которой и сработает код, переданный в username.



Анализ root cause

Ключевым условием для эксплуатации этой уязвимости является запуск Django-приложения в режиме отладки: DEBUG = True в app.py.

Уязвимость заключалась в обработке и выводе информации о связанных исключениях. В шаблоне technical_500.html был блок кода, который отвечал за отображение причины исходного исключения.
В уязвимых версия она выглядела так:

```
{% autoescape off %}

...

{% ifchanged frame.exc_cause %}{% if frame.exc_cause %}

  ...

    The above exception ({{ frame.exc_cause }}) was the direct cause of
the following exception:

  ...

{% endif %}{% endifchange %}

...

{% endautoescape %}
```

{% autoescape off %} отключает автоматическое экранирование HTML-тегов для всего заключенного в нее блока. Это означает, что любые переменные, выводимые внутри этого блока, будут отображаться "как есть", без преобразования символов <,> и других в безопасные HTML-элементы(прим. > станет >)

{{ frame.exc_cause }} содержит текстовое описание исключения, которое стало причиной возникшей ошибки 500.

Django имеет механизм для "связывания" исключений, чтобы предоставить разработчику более полную картину ошибки. Например, если код приложения вызывает ошибку целостности данных в базе (например, нарушение уникальности поля), драйвер базы данных генерирует свое исключение. Django перехватывает его и создает свое, более высокоуровневое исключение, при этом сохраняя исходное исключение от драйвера в атрибуте __cause__, что и позволяет встроить код.

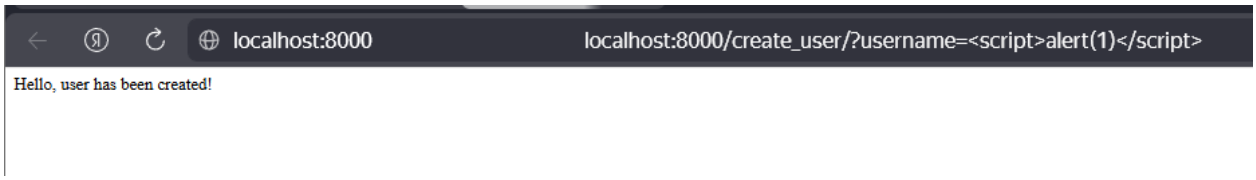
Описание примененного исправления

Данная уязвимость исправляется достаточно просто – установкой флага DEBUG в настройках приложения в значение False. После этого страницы с ошибками перестают формироваться с использованием шаблона technical_500.html, как при DEBUG = True, что позволяет избежать данной уязвимости. Сервер просто возвращает Server Error (500).

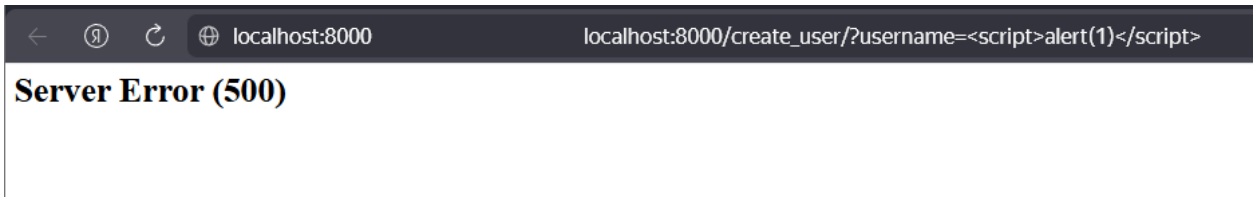
Доказательство устранения уязвимости

Остановка и перезапуск контейнера с внесенными изменениями

Запрос на создание пользователя:



Повторный запрос:



Как можно видеть, сервер вернул только код ошибки, без выполнения каких-либо скриптов.