ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

*Московский институт электроники и математики им. А.Н.Тихонова*

**Домашнее задание по теме 10 взаимодействие Python с WWW.   
Программирование на Python**

По направлению 10.04.01 – «Информационная безопасность»

Проверил:

преп. Антон Лукашов

Подпись \_\_\_\_\_\_\_\_\_\_\_\_\_

Выполнил:

Новиков В. С. МКБ 241

Подпись \_\_\_\_\_\_\_\_\_\_\_\_\_

Москва 2025

# Введение

Научиться использовать инструмент Scapy для анализа сетевого трафика и эксплуатации уязвимостей cross-site scripting (XSS) на учебном сайте Google Gruyere.

Описание тестовой среды: операционная система Linux, использовалась локальная версия Google Gruyere, адрес 127.0.0.1, порт 8008, инструменты – браузер firefox, библиотека python Scapy

Весь код, примеры скриншотов с сайта, файлы: http\_requests.txt, http\_responses.txt расположены по адресу:

<https://github.com/vit81g/Cybersecurity_HSE/tree/main/HomeWorks/Python_basics/HW_9_WWW>

# Этап 1. Изучение Scapy

Настройте Scapy для перехвата HTTP-трафика, используйте скрипт scapy для отправки HTTP-запросов.

Настройка:

*1. активация venv*

*source venv/bin/activate*

*2. установка*

*pip install scapy*

*3. запуск скрипта (с повышенными привелегиями)*

*sudo python script\_name.py*

Ссылка на код по захвату трафика с помощью scapy:

<https://github.com/vit81g/Cybersecurity_HSE/blob/main/HomeWorks/Python_basics/HW_9_WWW/hw09_01_v2.py>

# Этап 2. Анализ трафика

Ознакомьтесь с инструментом Google Gruyere и запустите его.

* Запустите Scapy и начните собирать трафик, взаимодействуя с сайтом Google Gruyere.
* Проанализируйте полученные данные, обращая внимание на запросы и ответы HTTP.

Решение:

Установка Google Gruyere на локальной машине. Запуск скрипта ./ gruyere.py

Сбор сетевого трафика с помощью Scapy, настройки:

*filter="tcp port 8008", iface="lo", prn=packet\_callback, store=0*

Запросы, перехваченные Scapy:

**HTTP-запрос:**

*GET /424434239301099530633532976913311439060/snippets.gtl HTTP/1.1*

*Host: 127.0.0.1:8008*

*User-Agent: Mozilla/5.0 (X11; Linux x86\_64; rv:128.0) Gecko/20100101 Firefox/128.0*

*Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,\*/\*;q=0.8*

*Accept-Language: en-US,en;q=0.5*

*Accept-Encoding: gzip, deflate, br, zstd*

*Referer: http://127.0.0.1:8008/424434239301099530633532976913311439060/editprofile.gtl%3Cscript%3Ealert(1)%3C/script%3E*

*DNT: 1*

*Connection: keep-alive*

*Cookie: GRUYERE=91049579|user||author*

*Upgrade-Insecure-Requests: 1*

*Sec-Fetch-Dest: document*

*Sec-Fetch-Mode: navigate*

*Sec-Fetch-Site: same-origin*

*Sec-Fetch-User: ?1*

*Priority: u=0, i*

**Ответы, перехваченные Scapy:**

*HTTP-ответ:*

*HTTP/1.0 200 OK*

# Этап 3. Эксплуатация XSS

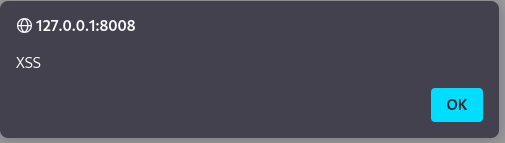
Исследование сайта <http://127.0.0.1:8008/424434239301099530633532976913311439060/editprofile.gtl>

1. Создав пользователя, перешли в создание постов. Где поле текста ввели XSS вида:

*<script>alert('XSS')</script>*

*<img src="nonexistent.jpg" onerror="alert('XSS')">*

После сохранения поста, при обновлении страницы мы видим XSS атаку

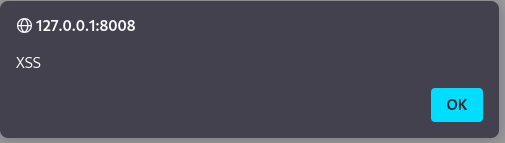


1. В адресной строке браузера в конце добавляем XSS:

*<script>alert('XSS')</script>*

[*http://127.0.0.1:8008/365409555492888778759821044345253631054/snippets.gtl<script>alert('XSS')</script*](http://127.0.0.1:8008/365409555492888778759821044345253631054/snippets.gtl%3cscript%3ealert('XSS')%3c/script)*>*

После перехода по данной ссылки срабатывает XSS атака



# Этап 4. Анализ результатов

Для перехвата трафика будем использзовать Scapy, результаты будут вывдены в два файла:

http\_requests.txt

http\_responses.txt

Код программы на странице github.com, имя файла hw09\_02\_v1.py

*> sudo python3 hw09\_02\_v1.py*

Открываем сайт:

[*http://127.0.0.1:8008/365409555492888778759821044345253631054/snippets.gtl*](http://127.0.0.1:8008/365409555492888778759821044345253631054/snippets.gtl)

Переходим в посты, ждем запуска XSS. Запущенный скрипт hw09\_02\_v1.py зафиксирует наши действия (запросы и ответы)

Для анализа тела ответа в браузере нажать F12, перейдем в режим разработчика.

*<body>*

*...*

*<div>*

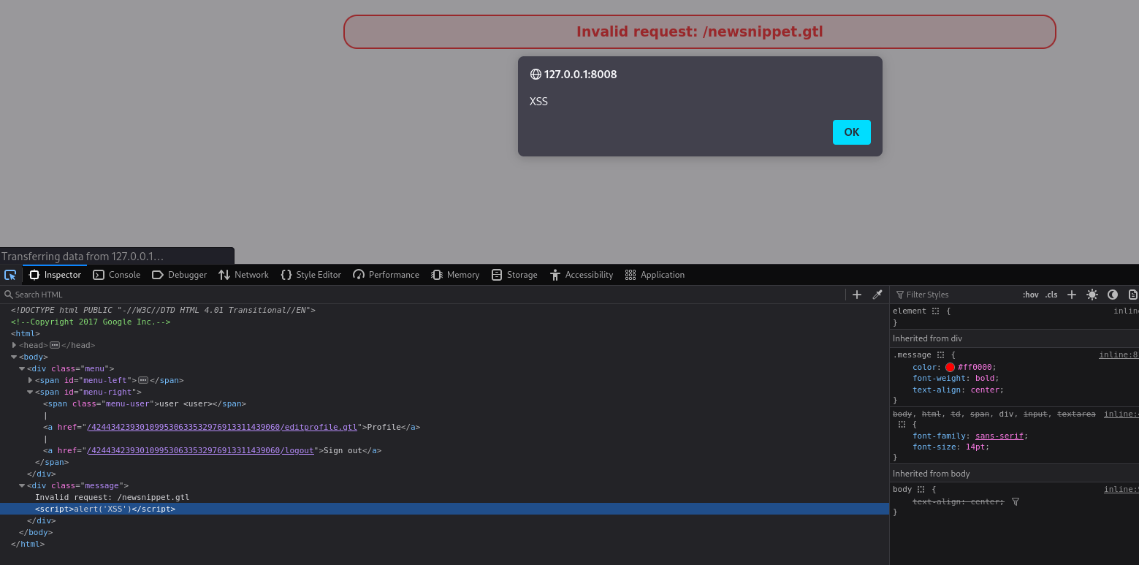
*…*

*<script>alert('XSS')</script> # это XSS*

*</div>*

*…*

*</body>*



# Этап 5. Отчёт

Проведён анализ страниц тестового сайта Google Gruyere. Особое внимание уделено формам редактирования профиля, где пользовательский ввод отображается без надлежащей фильтрации.

Обнаружена уязвимость в поле, отвечающем за личную информацию (например, параметр private\_snippet), где вводимые данные впоследствии выводятся на странице.

Подготовка и внедрение пейлоада:

* Для тестирования использован классический XSS-пейлоад:

*<script>alert('XSS')</script>*

* Пейлоад был внедрён в параметр запроса (например, через URL или в форме редактирования профиля). В URL этот пейлоад кодировался следующим образом:

*%3Cscript%3Ealert%28'XSS'%29%3C%2Fscript%3E*

После отправки формы/обновления профиля произошёл рефлекс (отражение) вредоносного кода на странице – в браузере появилось всплывающее окно с сообщением alert('XSS').

На сайте наблюдалось изменение HTML-кода: в тех местах, где ожидался вывод безопасного текста, появился внедрённый JavaScript-код.

# Заключение

В результате проведённого исследования было выявлено, что веб-приложение Google Gruyere уязвимо к XSS-атакам. Анализ HTTP-трафика с использованием Scapy показал, что вредоносный ввод отражается в HTML-ответах без должной фильтрации, что приводит к выполнению встроенного JavaScript-кода. В отчёте приведены рекомендации по устранению уязвимости, включающие экранирование ввода, использование CSP и регулярный аудит безопасности.

# Пример скриншота с сайта:

