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

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

Тема 9
«Взаимодействие в WWW: взаимодействие с сервисами»

Домашняя работа № 8

по направлению 10.04.01 Информационная безопасность

курса «Программирование на Python»

Проверила:

преп. Дарья Погудина
Подпись _____

Выполнил:

Горбунов Г.Н. МКБ251

Подпись _____

Москва – 2026

Оглавление

Тема и цель работы

Тема домашней работы: «Взаимодействие в WWW».

Цель работы:

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

Оборудование, ПО

Устройство	Операционная система	IP адрес/ Маска	Шлюз	DNS
Личный ПК	Debian 12	192.168.1.6 (выдан DHCP)	192.168.1.1 (Маршрутизатор DHCP -сервер)	192.168.1.1

Выполнение

1. Определение XSS-уязвимости.

XSS (англ. Cross-Site Scripting — «межсайтовый скриптинг») — это уязвимость в веб-приложениях, которая позволяет злоумышленникам внедрять и исполнять произвольный скрипт (чаще всего JavaScript) в контексте сайта или веб-приложения, просматриваемого законопослушным пользователем. Суть проблемы в том, что такой вредоносный код обычно выполняется в браузере жертвы, получая доступ к элементам страницы, куки или токенам аутентификации. В результате злоумышленник может:

- Похитить конфиденциальную информацию (например, куки, сессии или личные данные).
- Провести фишинговые атаки, выдавая поддельные формы или ссылки за реальные.
- Заставить пользователя совершить незапланированные действия (например, сделать перевод средств в приложении банка).
- В некоторых случаях — распространять вредоносные программы, скрытые в коде.

Важно понимать, что XSS — это не единичный сценарий атаки, а целый класс уязвимостей, который проявляется в различных формах и с разным набором угроз. Часто люди, впервые столкнувшиеся с этим термином, предполагают, что XSS связан только с подделкой форм или всплывающими окнами. Но на практике спектр применений куда шире.

2. Описание принципа работы библиотеки Scapy в рамках исследования сайтов на наличие XSS.

В рамках домашней работы (тестирования веб-приложения Google Gruyere) использование сниффера на базе библиотеки Scapy позволяет проводить пассивный анализ сетевого трафика в реальном времени с целью выявления уязвимостей XSS.

Основная идея заключается в перехвате HTTP-ответов от сервера и проверке, отражает ли он пользовательский ввод без надлежащего экранирования. Если в теле HTML-ответа обнаруживаются неэкранированные специальные символы (например, <, >, ", ') или исполняемые конструкции JavaScript (такие как onerror=, alert(, document.cookie), это указывает на потенциальную уязвимость к reflected XSS. Scapy, будучи инструментом низкоуровневого анализа сетевых пакетов, позволяет перехватывать «сырые» TCP-сегменты, извлекать из них HTTP-данные и осуществлять поиск сигнатур XSS.

Такой подход особенно эффективен в контролируемой среде, где исследователь сам инициирует запросы с заранее заданными полезными нагрузками (например, <img src=x onerror=alert(document.cookie)>). Если сервер возвращает эту нагрузку в неизменённом виде внутри HTML-контекста, сниффер фиксирует её в теле ответа, что служит прямым признаком отсутствия механизмов санитизации (обезвреживания) входных данных.

3. Написание работы собственного сниффера с использованием Scapy.

Суть работы сниффера заключается в последовательном выполнении ряда действий:

1) Установить прослушивание сетевого трафика на всех интерфейсах с фильтрацией по целевому IP-адресу и порту (в нашем случае — 216.58.210.180:80), чтобы захватывать только релевантные TCP-пакеты, относящиеся к взаимодействию с сервисом Google Gruyere.

2) Перехватывать и декодировать HTTP-запросы, включая:

- строку запроса (например, GET /?q=<payload> HTTP/1.1);
- URL-параметры (извлекаемые из строки запроса);
- тело POST-запросов (если присутствует).

3) Для каждого из этих элементов выполняется проверка на наличие сигнатур XSS (таких как onerror=, alert(, document.cookie и др.) с помощью заранее определённых регулярных выражений.

4) Перехватывать HTTP-ответы от сервера, извлекать из них HTML-тело (расположенное после разделителя заголовков \r\n\r\n) и анализировать его на предмет отражения пользовательского ввода без экранирования.

5) Фиксировать все подозрительные события: как попытки отправки XSS-полезной нагрузки (в запросах), так и подтверждение уязвимости (в ответах) — с выводом в консоль и одновременной записью в лог-файл result.log.

Таким образом, сниффер обеспечивает сквозной пассивный мониторинг клиент-серверного взаимодействия, позволяя не только обнаружить потенциальные точки внедрения, но и подтвердить эксплуатируемость уязвимости XSS на основе анализа ответа сервера.

4. Исследование Google Gruyere

Исходный текст скрипта приведен в файле hw_8.py.
Работа Scapy в режиме просмотра Raw-данных требует привилегий root, в связи с этим запуск скрипта следует производить командой:

```
sudo ./hw_8.py
```

после чего скрипт отчитается о начале работы.

Откроем инстанс Google Gruyere (Google Gruyere генерирует новый инстанс для каждого нового запуска – в моем случае это http://google-gruyere.appspot.com/485216650750532078476376784740994309881/)

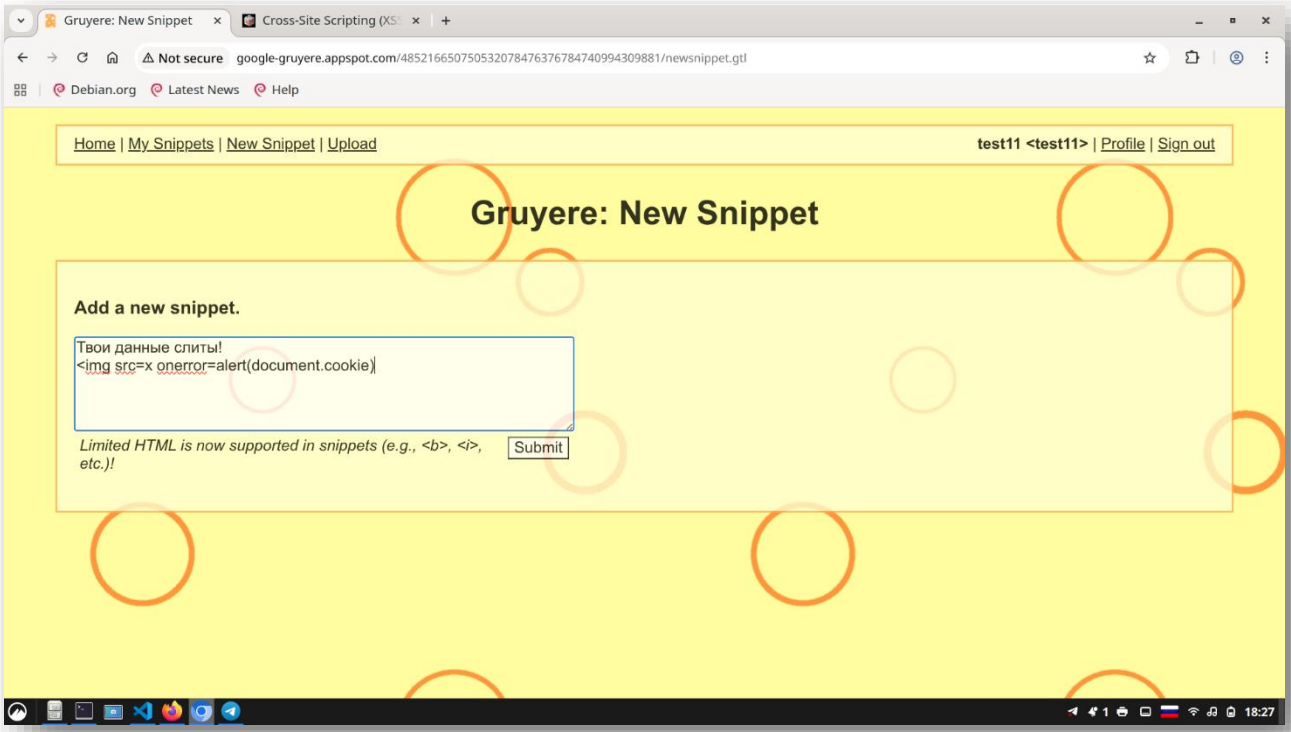
Придумаем XSS-нагрузку, такую как

<img src=x onerror=alert(document.cookie)>

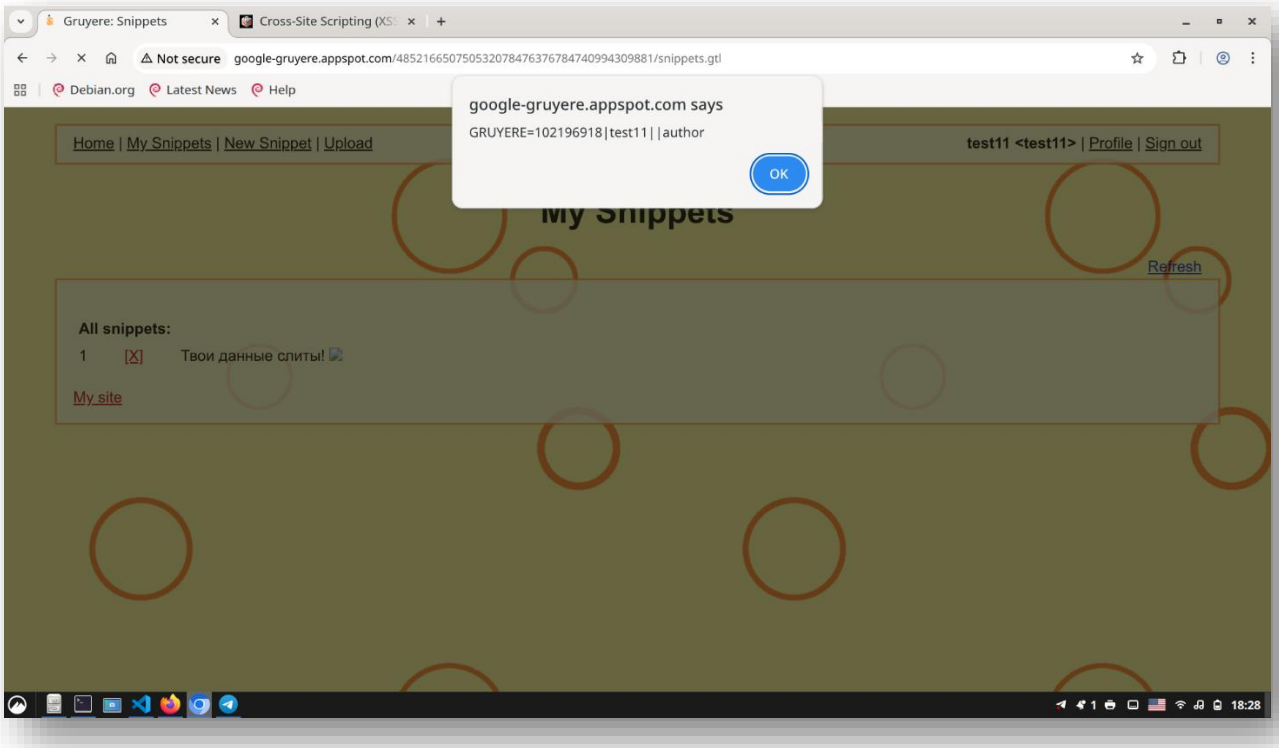
*(Атрибут onerror — это **встроенный JavaScript-обработчик события**, который выполняется **автоматически при ошибке загрузки**. Так как мы передаем «x» в качестве ссылки на рисунок, то браузер не обнаружит изображения по такому адресу и сработает обработчик – код, который выведет cookie-данные текущего пользователя.)*

Начнем вставлять XSS-нагрузку во все текстовые поля, доступные для ввода и будем отслеживать – где XSS сработает (в некоторых случаях придется обновлять страницу).

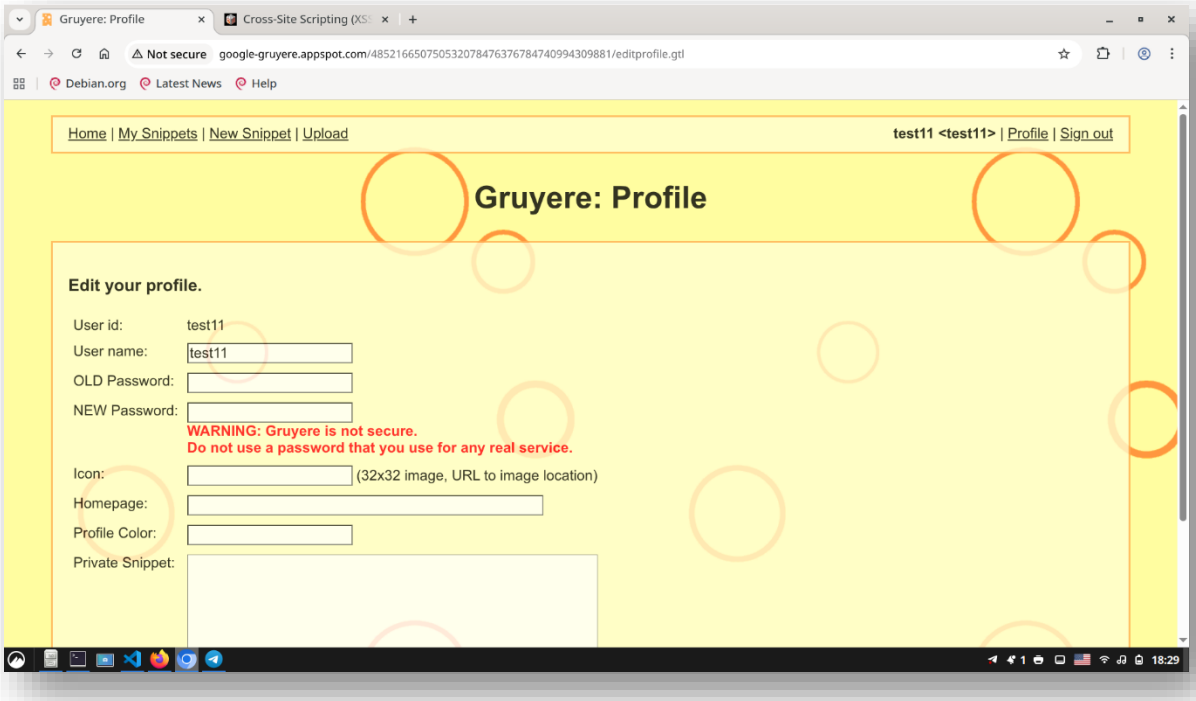
4.1 Создаем новый Сниппет



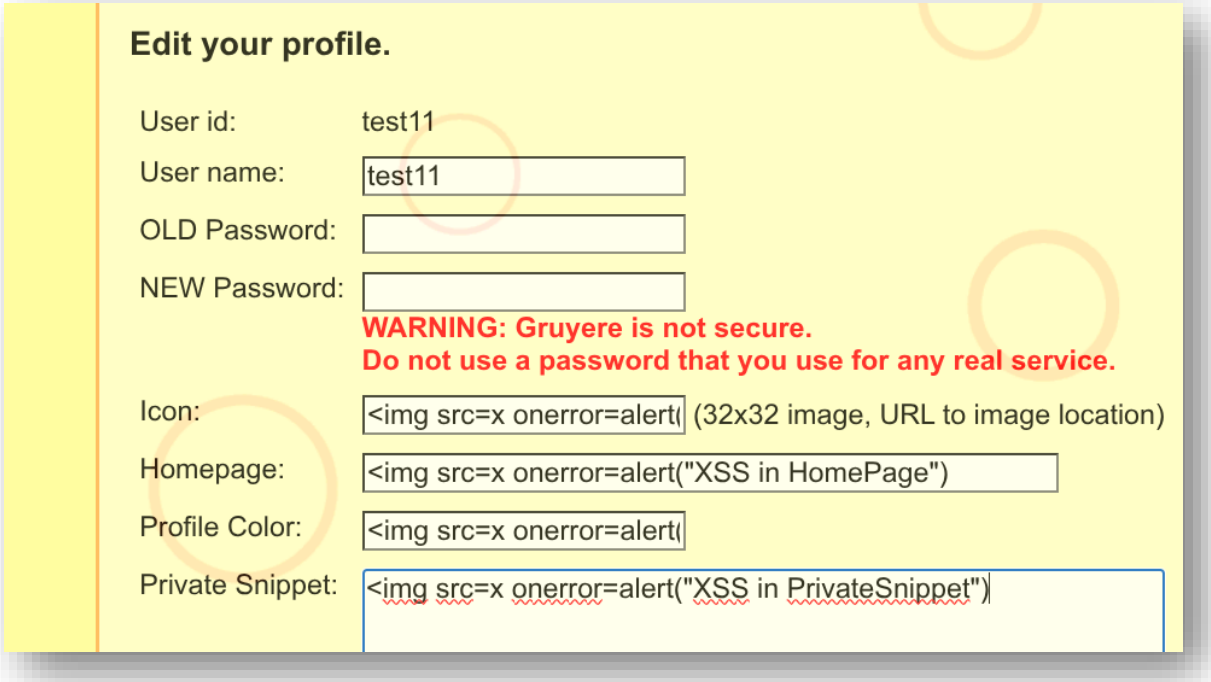
При сохранении он тут же сработал



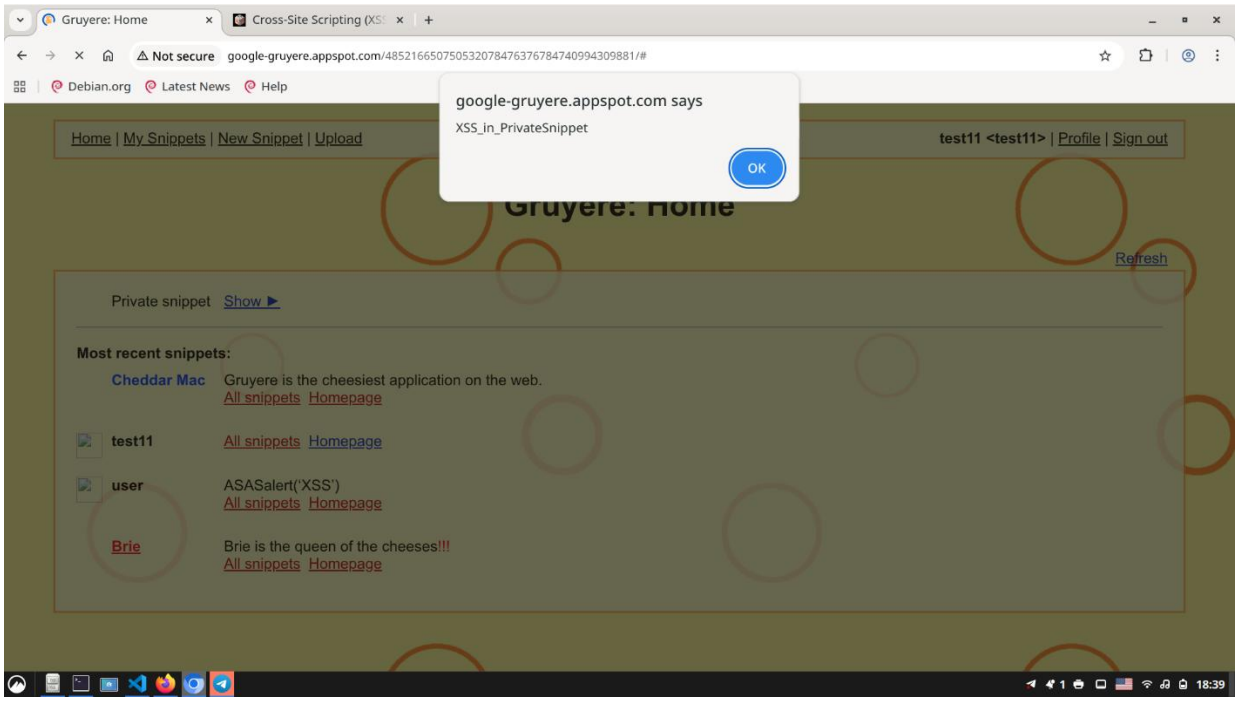
4.2 Посмотрим, какие поля доступны в ЛК



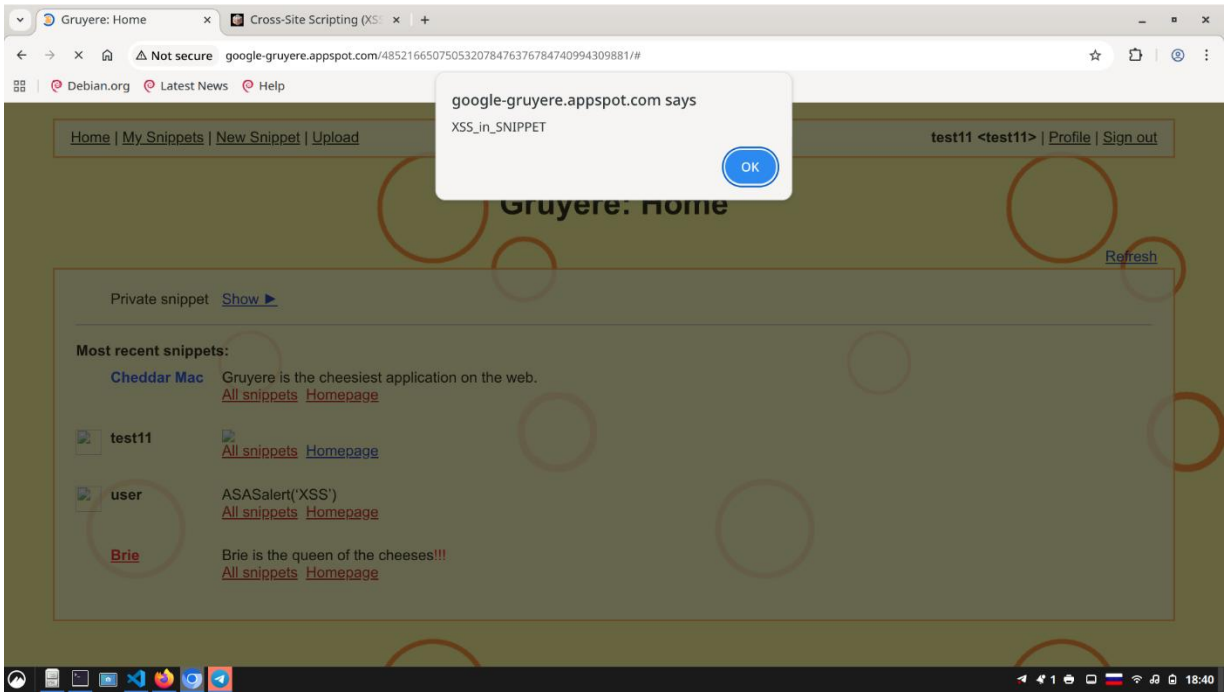
Заполним поля подготовленными нагрузками:



Проверим, какие срабатывают, гуляя по всем вкладками страницы:



Сработал для Приватных Сниппетов



Тут же сработал для публичных (страндартных)

5. Вывод