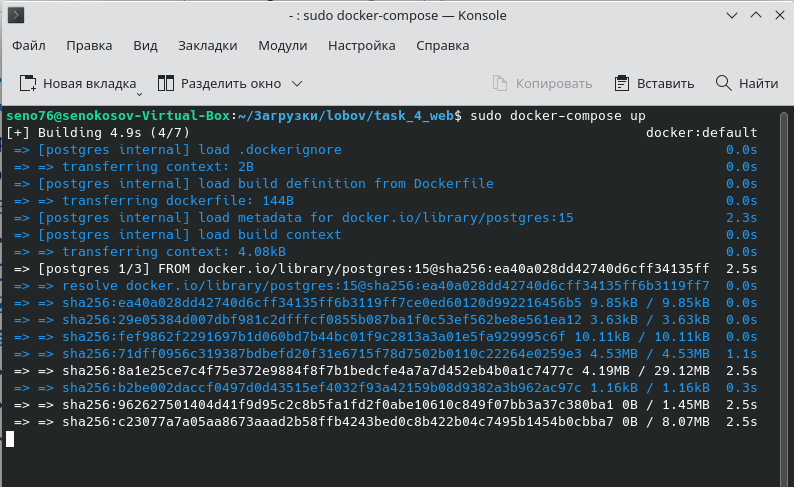
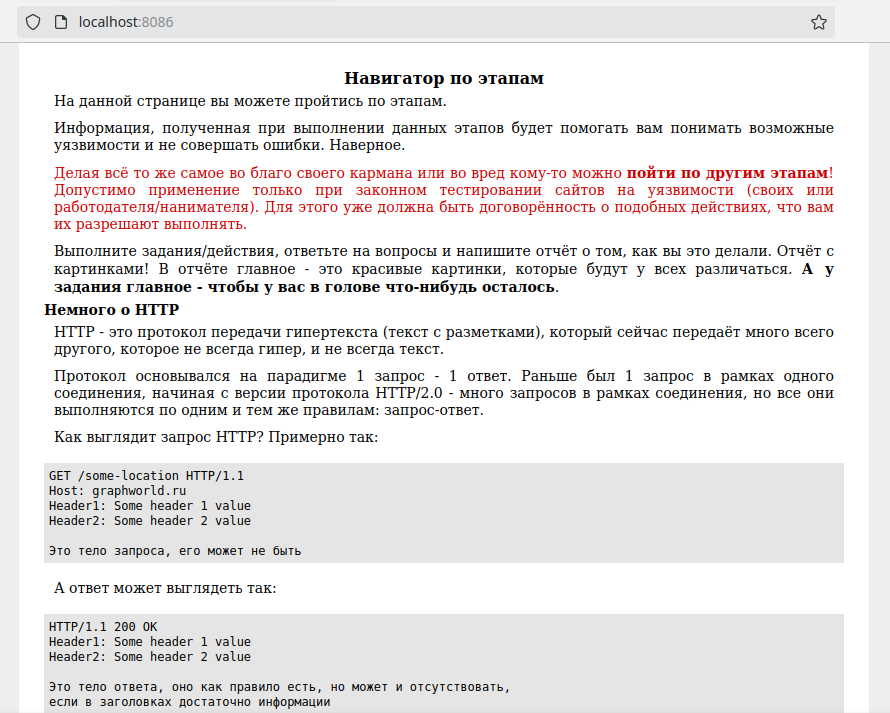
**Лабораторная работа 5**

**Выполнил: Сенокосов Владислав Владимирович 431**

Запустили docker-compose и установили необходимые компоненты:

****

После установки успешно зашли на тестовый полигон:

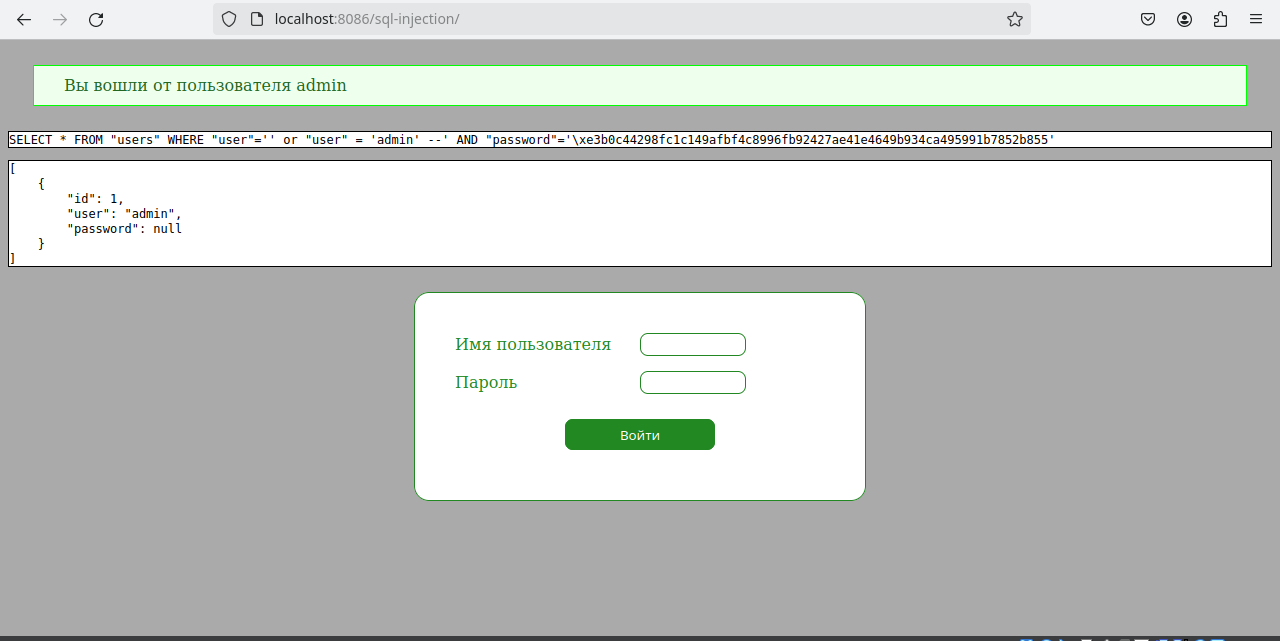
****

А) Инъекция SQL-кода:

Так как мы знаем имя пользователя, под которым хотим зайти, то в данном случае была применена идея экранирования запроса SQL с помощью символов --, а также экранирования строк при помощи символа `. Запрос, который был послан в поле «Имя пользователя» имеет следующий вид:

' or "user" = 'admin' --""

Ввиду того, что SQL запросы обрабатываются при условной конструкции до первого верного значения. То в нашем запросе мы сразу получаем верный результат, так как пользователь admin действительно существует в таблице users. В результате проделанной работы мы смогли зайти от имени admin:



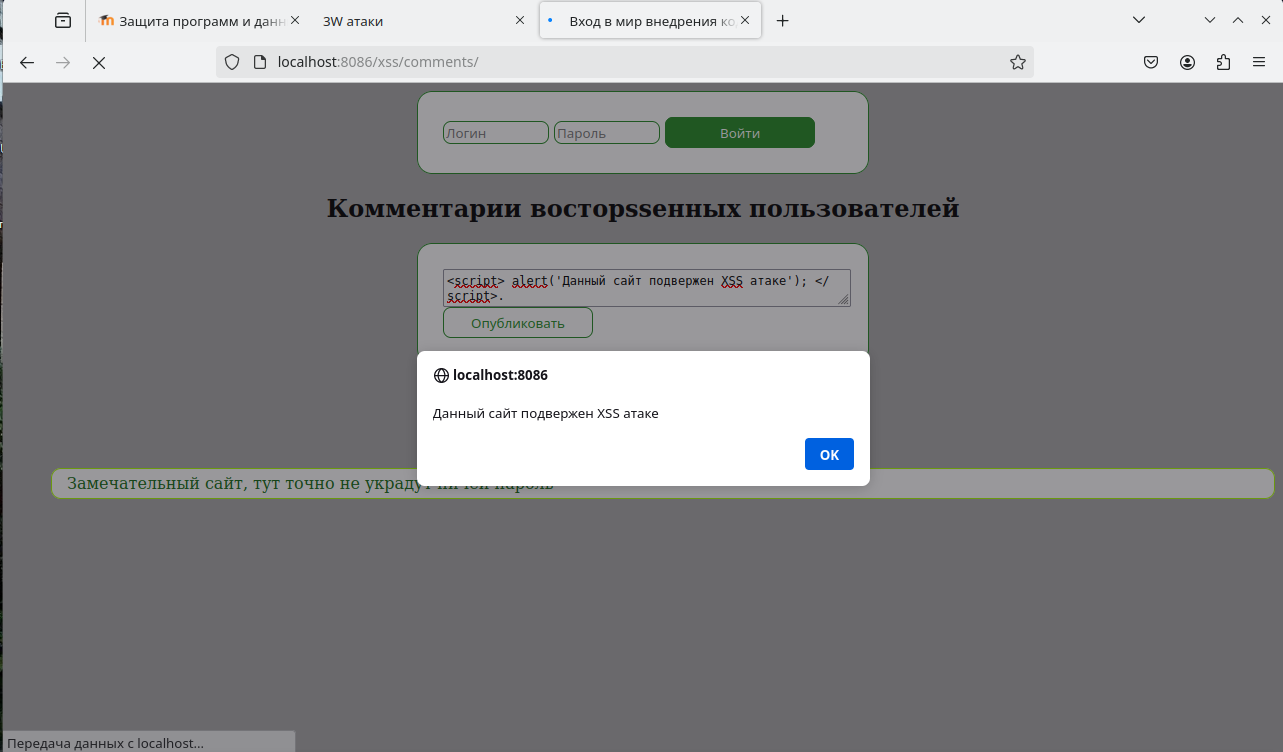
1. Каков базовый механизм атаки?

Базовый механизм атаки заключается во внедрении SQL кода в поля ввода данных от пользователя. При введении такого вредоносного SQL кода, злоумышленник может получить различную информацию о структуре базы данных (имена таблиц, полей, значений в полях, имена пользователей, версию базы данных и т.д). То есть в общем случае проблема заключается в конкатенации кусков запроса.

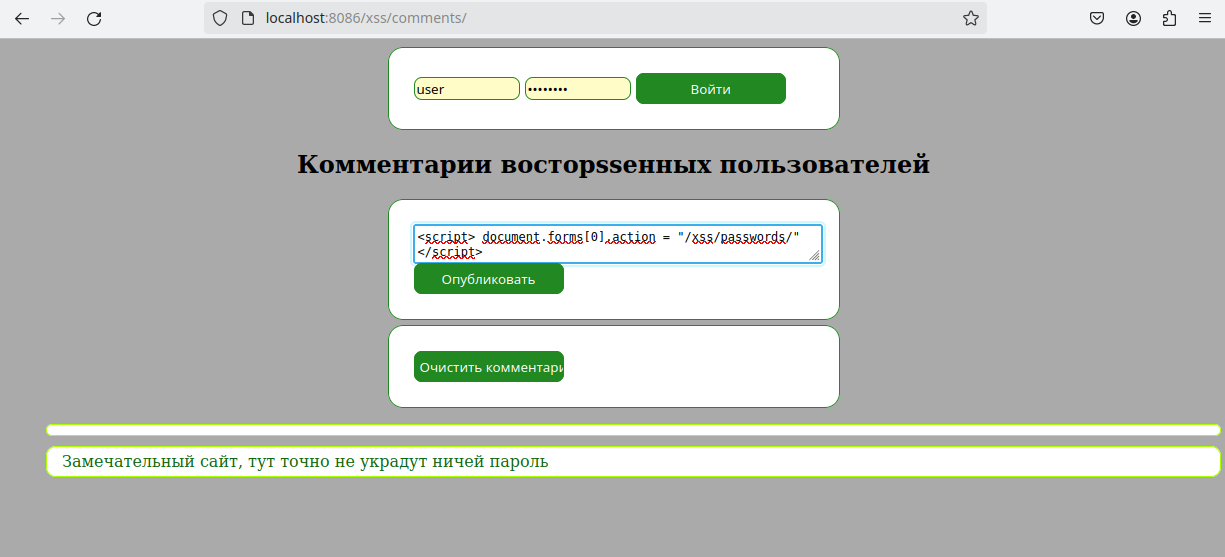
1. Какие есть способы защиты от неё?
   1. Использование параметризированных запросов – передача данных от пользователя происходит независимо от построения SQL запроса.
   2. Экранирование передаваемых данных и их предварительный анализ передаваемых строк на наличие тех или иных конструкций или типов данных
   3. Уменьшение количество вводимых символов. Если разработчик знает особенности передаваемых данных, то есть смысл ограничить число символов, так как обычно SQL инъекции имеют большую длину.
2. Какие инструменты применяются для защиты на уровне разработчика.
   1. Использование файрволлов прикладного уровня, некоторые из которых могут осуществлять перечисленные методы.
   2. Безопасные ORM (Object-Relational Mapping) библиотеки: такие библиотеки предоставляют абстракции для работы с базой данных и обеспечивают безопасную генерацию SQL запросов.
   3. Использование баз данных, которые имеют в своем арсенале возможные механизмы защиты
   4. Конструктор запросов — это инструмент или библиотека, предназначенная для динамического формирования SQL запросов с использованием программного интерфейса (API) на языке программирования. Вместо написания SQL запросов вручную в виде строковых выражений, разработчик может использовать конструктор запросов для создания SQL запросов с помощью методов и функций языка программирования.

Б) XSS:

Введем скрипт, который поможет нам узнать если защита от XSS или нет. В результате при вводе данных у нас появляется окошко с переданным текстом, который был сгенерирован нашим скриптом, что говорит о том, что данный сайт подвержен XSS уязвимости:



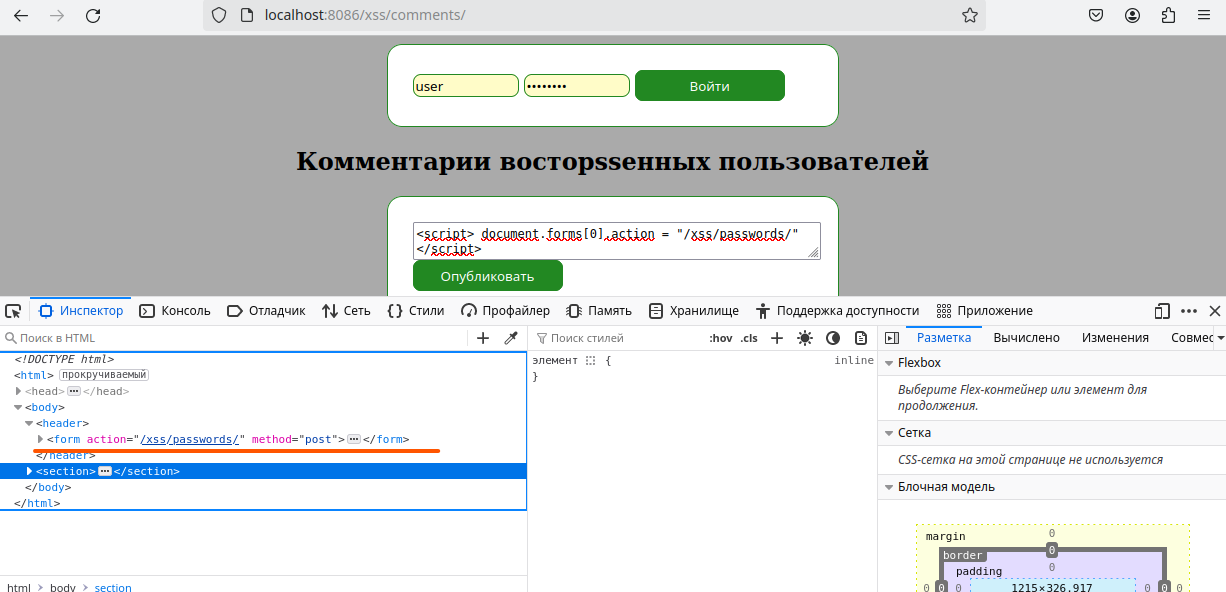
Теперь осуществим отправку данных о логине и пароле пользователя, который хочет произвести авторизацию на сайте:



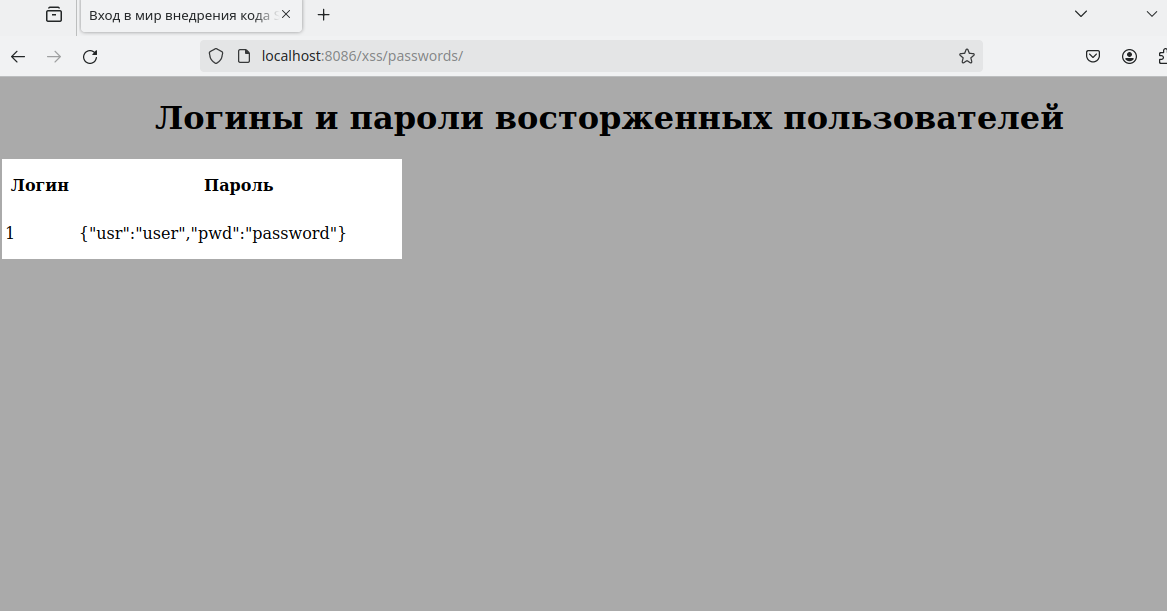
Теперь напишем скрипт, который позволил бы перенаправить данные со страницы xss/somments на xss/passwords/:

<script> document.forms[0].action = "/xss/passwords/" </script>

Можно заметить, что значение действительно изменилось на сайте:



В итоге мы отправили данные о логине и пароле на адрес злоумышленника при нажатии «Вход» в форме:



1. Каков механизм атаки?

Механизм атаки XSS (Cross-Site Scripting) заключается во внедрении вредоносного JavaScript кода на веб-страницу, которая затем выполняется в браузере клиента. Это позволяет злоумышленнику получить доступ к данным пользователя, выполнять действия от его имени и т. д.

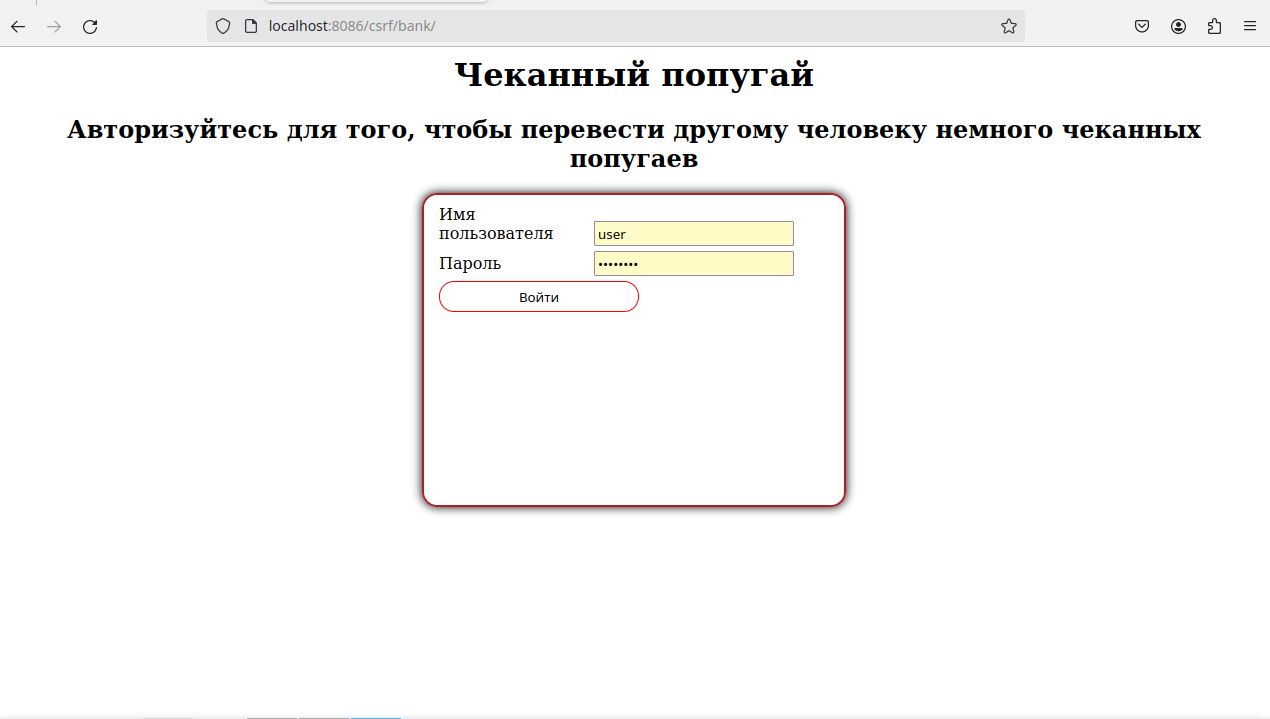
1. Внедрение какого языка происходит?

Внедрение происходит на языке JavaScript

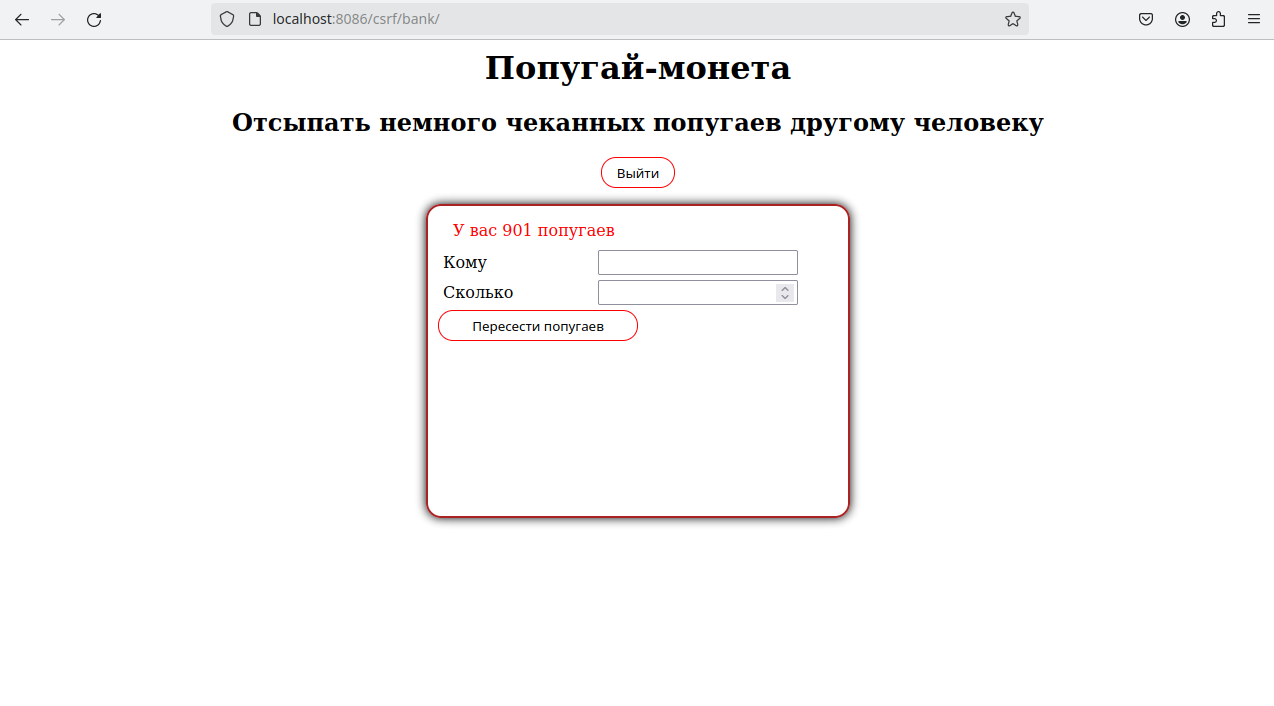
1. Какие есть способы защиты от неё на уровне администратора web-сервера?
   1. Фильтрация (то есть просто всякую подозрительную последовательность сразу выкидываем)
   2. Экранирование символов (< = &lt; , > = &gt; , & = &amp;)
   3. Использовать CORS (Cross-Origin Resource Sharing) – заранее настроенные политики безопасности
   4. Использовать CSP (Content-Security-Policy) - то механизм, который позволяет вам указать браузеру, откуда загружать ресурсы (такие как скрипты), что может помочь предотвратить XSS-атаки.
2. Как защититься от неё на уровне разработчика.
   1. Правильная шаблонизация (Twig, React, Angular, Vue ….)
   2. Неправильная шаблонизация (PHP, CSHTML, JSP ….)

В) CSRF:

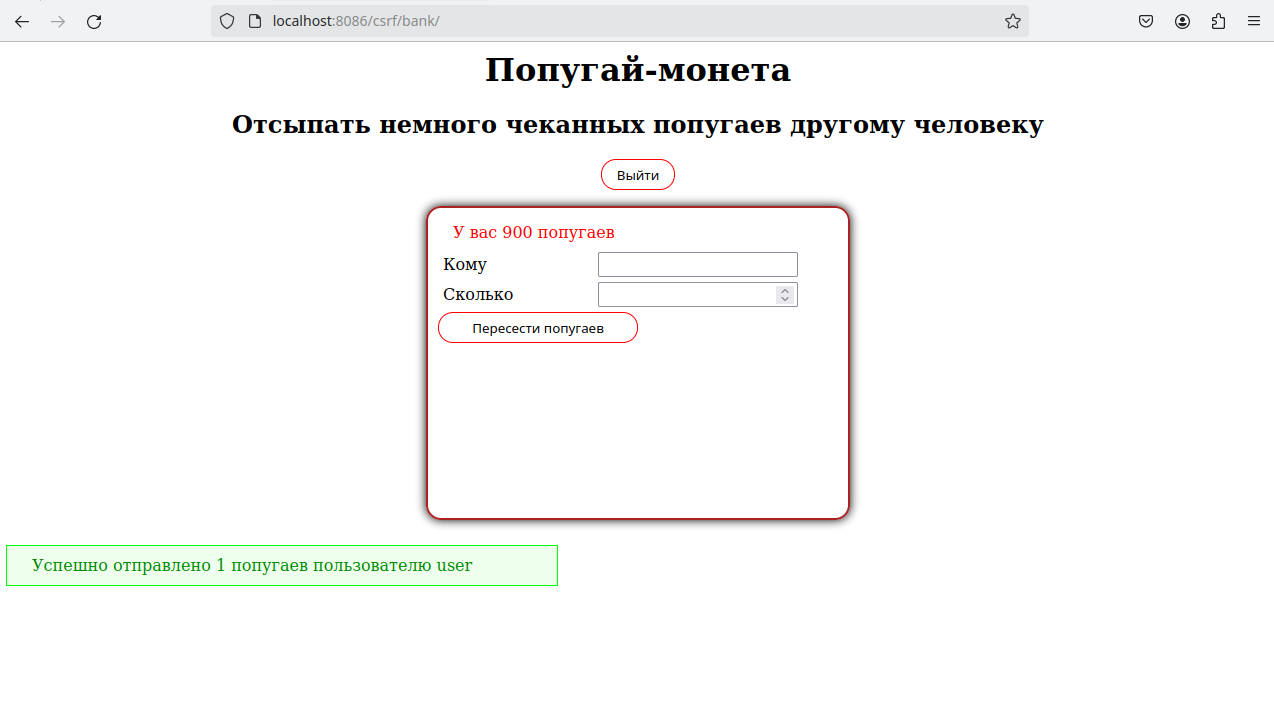
Произведем вход в банк под пользователем user и паролем password:



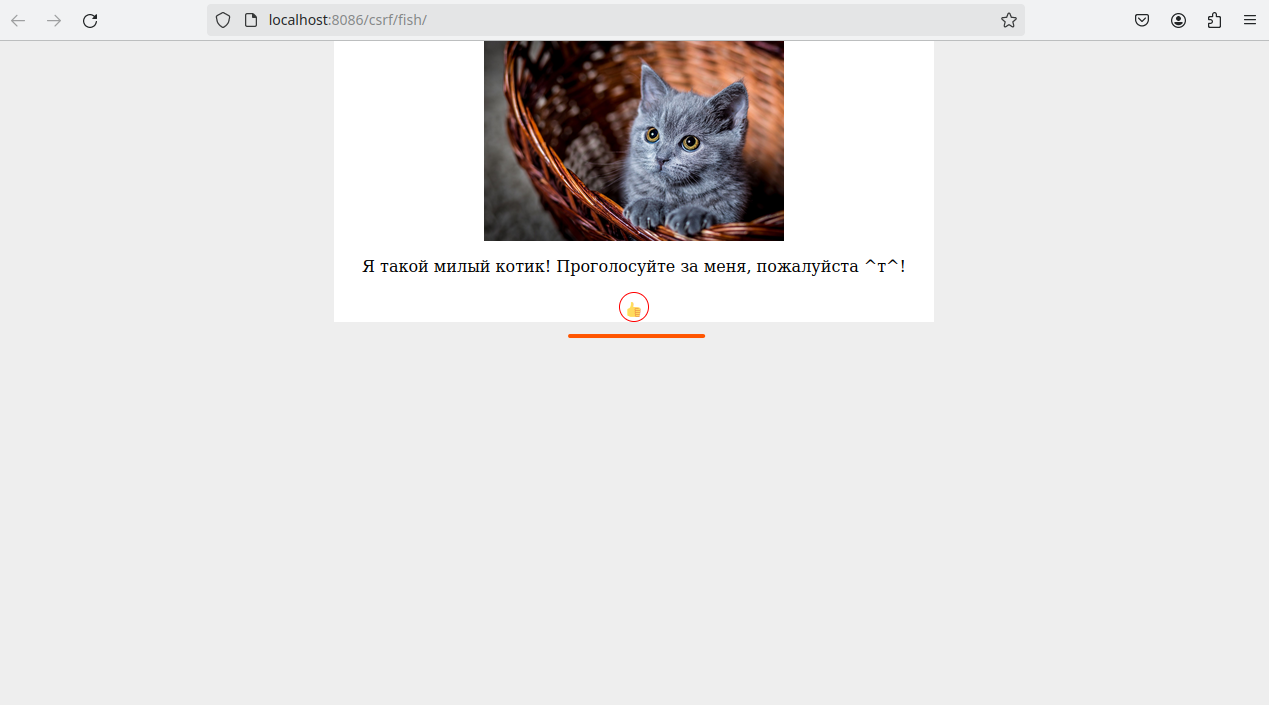
Видим стартовое окно с возможностью перевода денег:

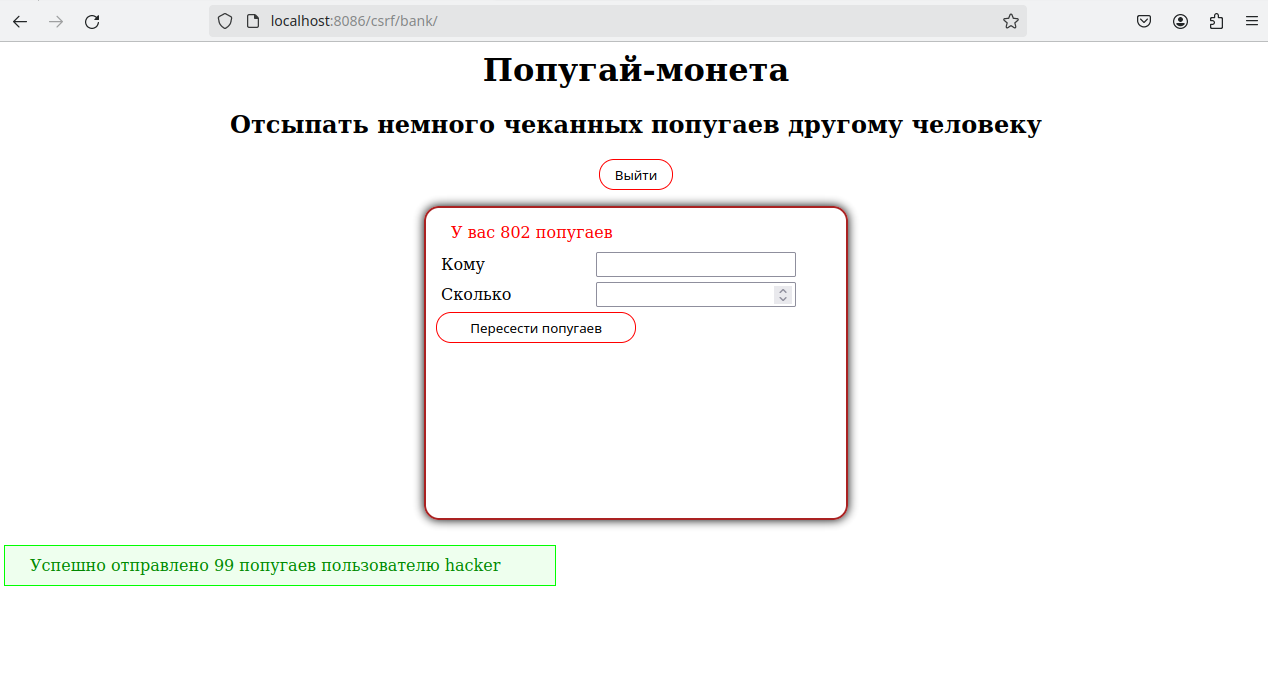


Переведем 1 попугая для проверки работы:

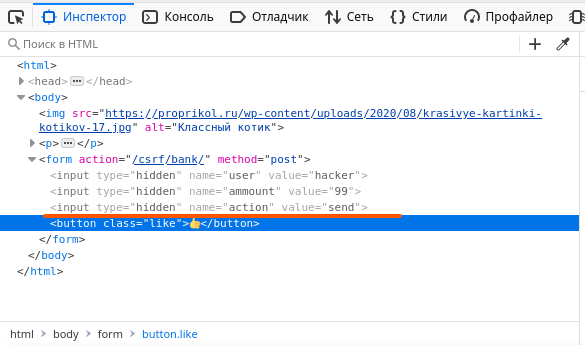


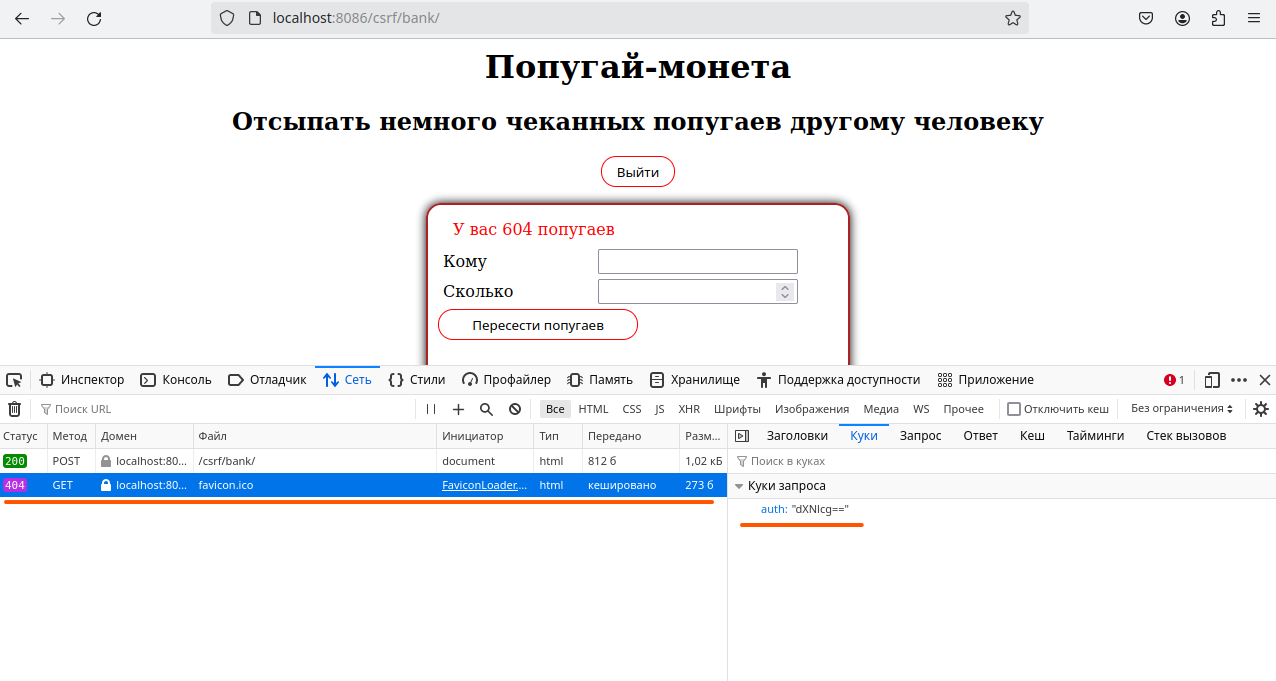
Теперь зайдем на сайт, который выступает в роли фишинга, и останемся авторизированными на сайте банка, и нажмем на кнопку поставить лайк, в результате чего переведутся средства со счета в банке в размере 99 шакалов:





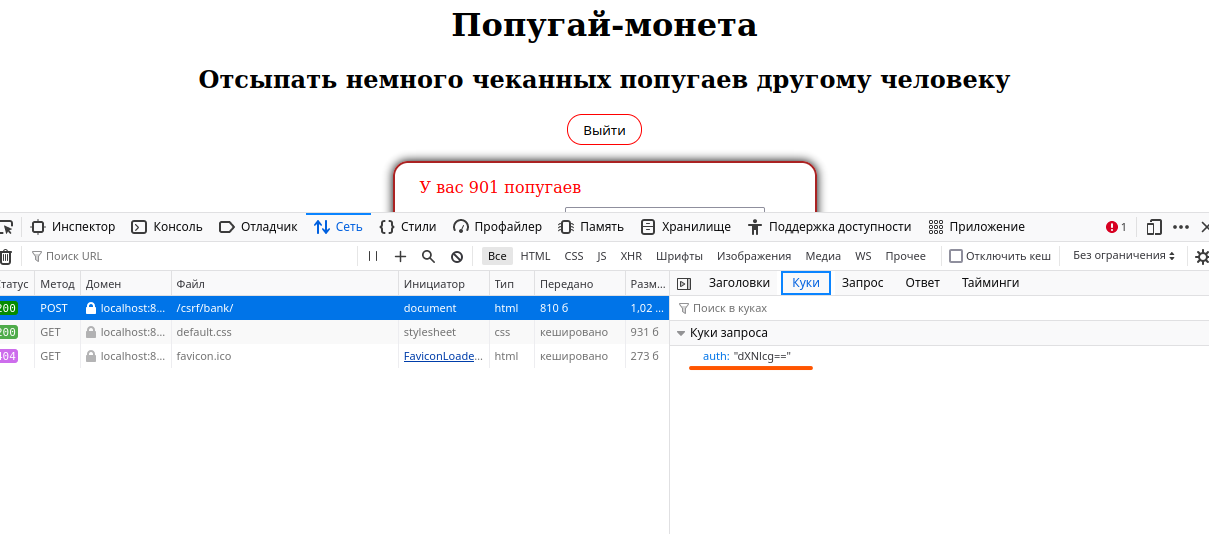
Структура запроса была следующая на фишинге c использованием куки пользователя в банке:



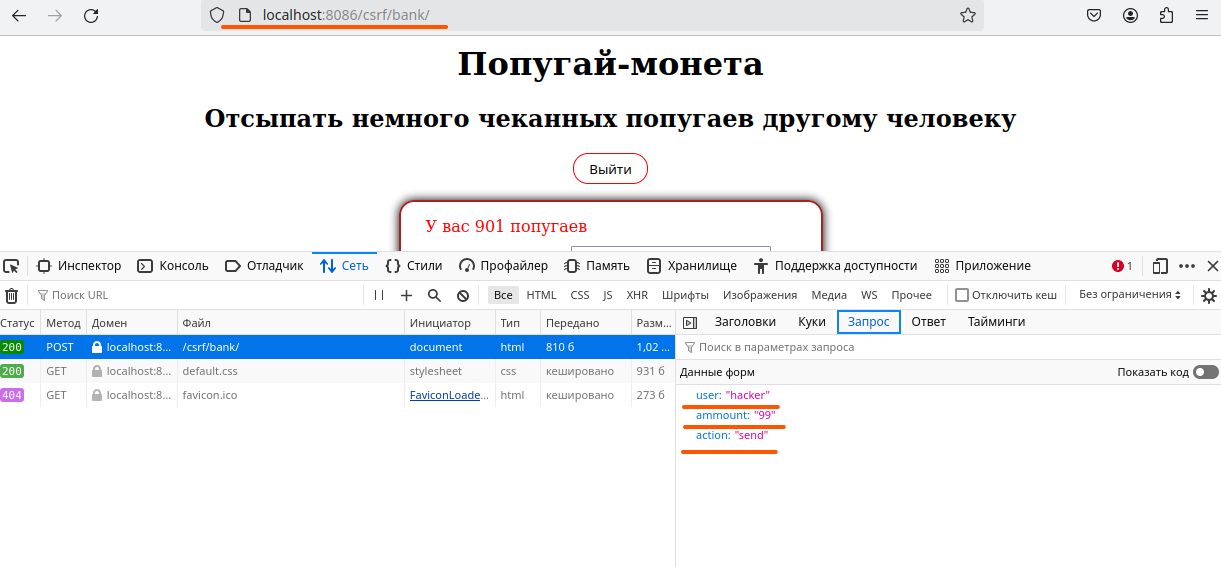


Сформируем curl запрос со следующими параметрами.

Куки:



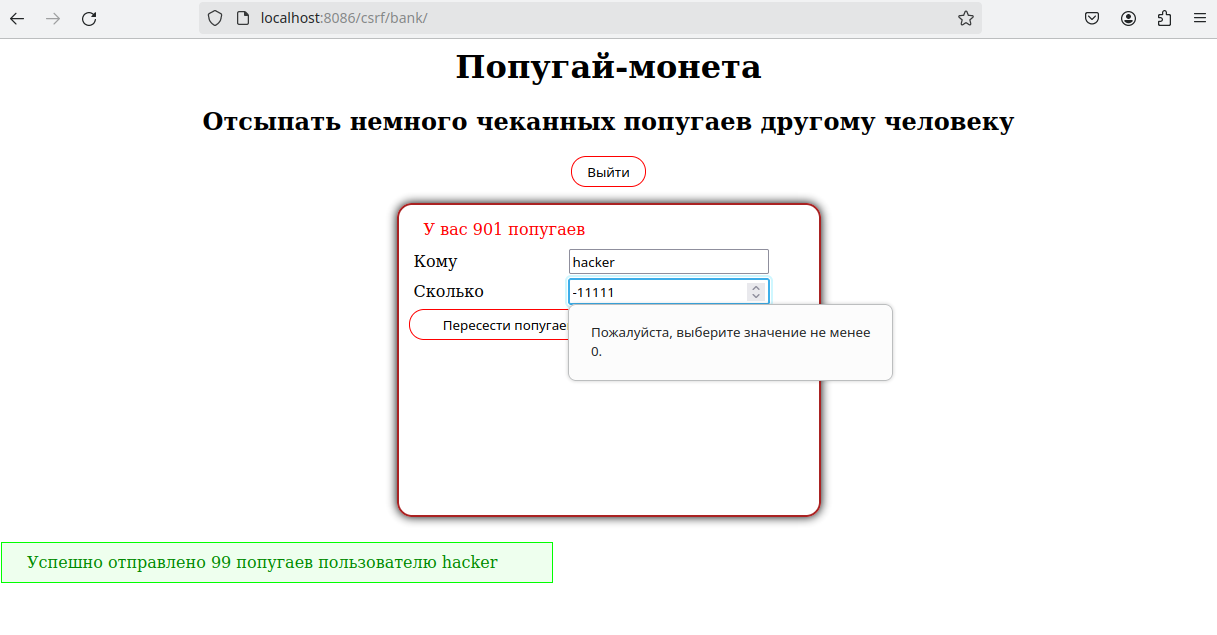
Параметры запроса и адрес, на который будем отправлять данные:



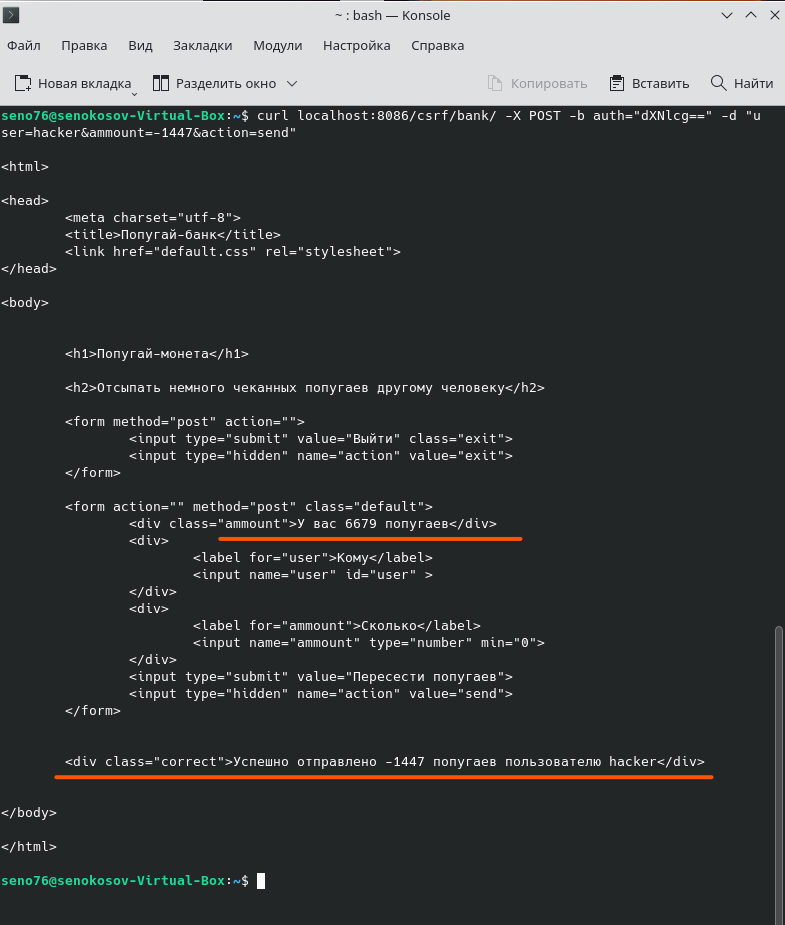
В итоге получим следующий запрос curl:

curl localhost:8086/csrf/bank/ -X POST -b auth="dXNlcg==" -d "user=hacker&ammount=-1000&action=send"

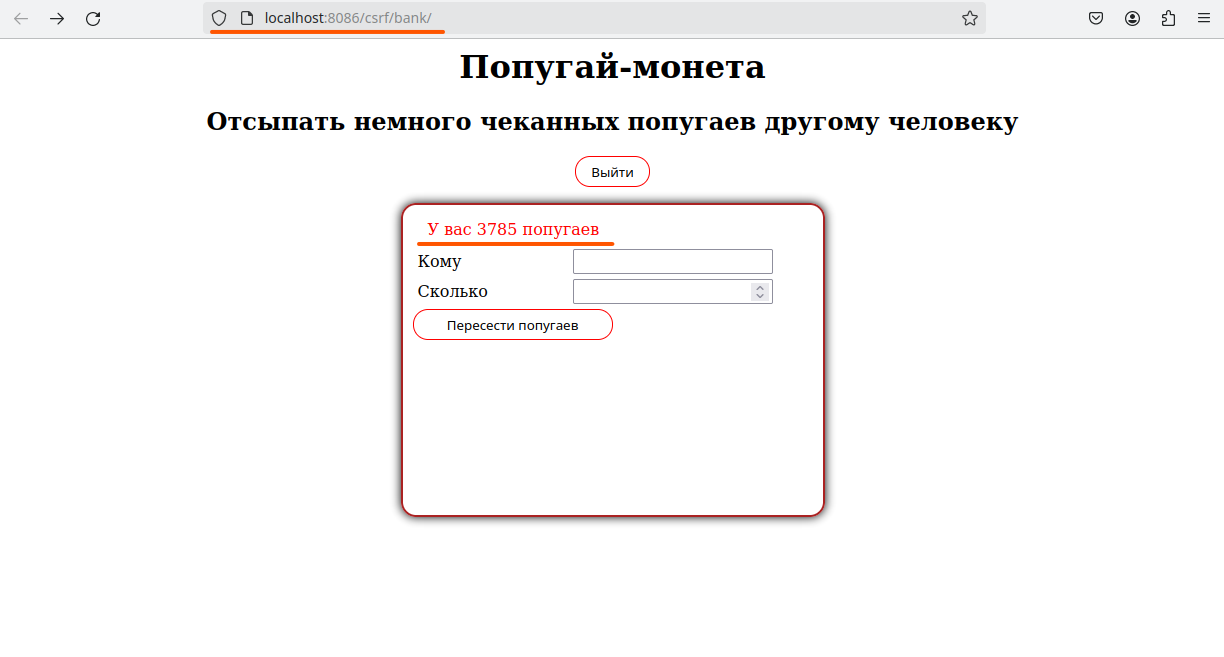
Предварительно проверим можем ли мы отослать отрицательное значение amount и убедимся, что нельзя:



Обратимся к банку через curl и убедимся, что мы можем снимать отрицательное значение, в результате чего растет счет пользователя:



На сайте также можно убедиться в этом:



1. Каков механизм атаки?

Подделка запросов авторизированных пользователей, то есть выполнение нелигитимный действий от лица имеющего на то права. Важной особенностью является, то, что само деяние осуществляет не злоумышленник, а сам пользователь, не подозревающий этого. Часто злоумышленник может проанализировать интересующий его сервис (например банк) на формат отправляемых запросов, понять его структуру и попытаться украсть данные пользователей, которые в данный момент авторизированы на интересующем сервисе, но для того, чтобы осуществить действия внутри браузера жертвы применяется фишинг. После перехода на фишинговый сайт пользователь должен как-то взаимодействовать с данными или формами и в результате этих действий будет отправлен вредоносный запрос, в котором будут куки сервиса и необходимые параметры запроса.

1. Где хранятся куки, из чего состоят и при каком условии передаются.

Куки — это небольшие текстовые файлы, которые веб-сайты сохраняют на компьютере пользователя. Они используются для хранения информации о действиях пользователя на сайте. Куки могут содержать различные данные, такие как идентификатор сеанса, предпочтения пользователя, информацию о входе и т. д.

Куки передаются между сервером и пользователем при каждом его запросе, это позволяет синхронизировать действия пользователя на wed ресурсе. Обычно куки передаются в заголовках HTTP. Устанавливаются с помощью заголовка Set-Cookie.

В большинстве случаев куки сохраняются в виде файлов на компьютере пользователя. Расположение этих файлов зависит от используемого браузера и операционной системы.

Некоторые куки могут храниться в оперативной памяти браузера вместо файловой системы. Это временное хранилище и обычно используется для временных куки, которые должны быть удалены после закрытия браузера.

Куки состоят из параметров запросов, которые в последствии будут обработаны сервером и выданы пользователю. Эти параметры соответствуют предпочтениям или настройкам пользователей, и тд.

Устанавливаются сервером заголовком или через JavaScript на клиенте Set-Cookie: Ключ=Значение; Атрибут…

document.cookie = 'cookie-name:cookie-value'

Отправка на сервер заголовком:

Cookie: Ключ=Значение

1. Почему атака работает при аутентификации с помощью куки.

В аутентификации с помощью куки, после того как пользователь успешно аутентифицирован на веб-сайте, сервер создает уникальный идентификатор сеанса (обычно в виде куки), который сохраняется на стороне клиента и отправляется обратно серверу с каждым запросом. Этот идентификатор сеанса используется для идентификации пользователя на протяжении его сеанса взаимодействия с веб-сайтом. Если подделать идентификационные данные в куки, то злоумышленник может получить данные от легитимного пользователя используя свой идентификатор сеанса.

1. Как защищаться от данной атаки на стороне разработчика. На что направлены данные атаки.

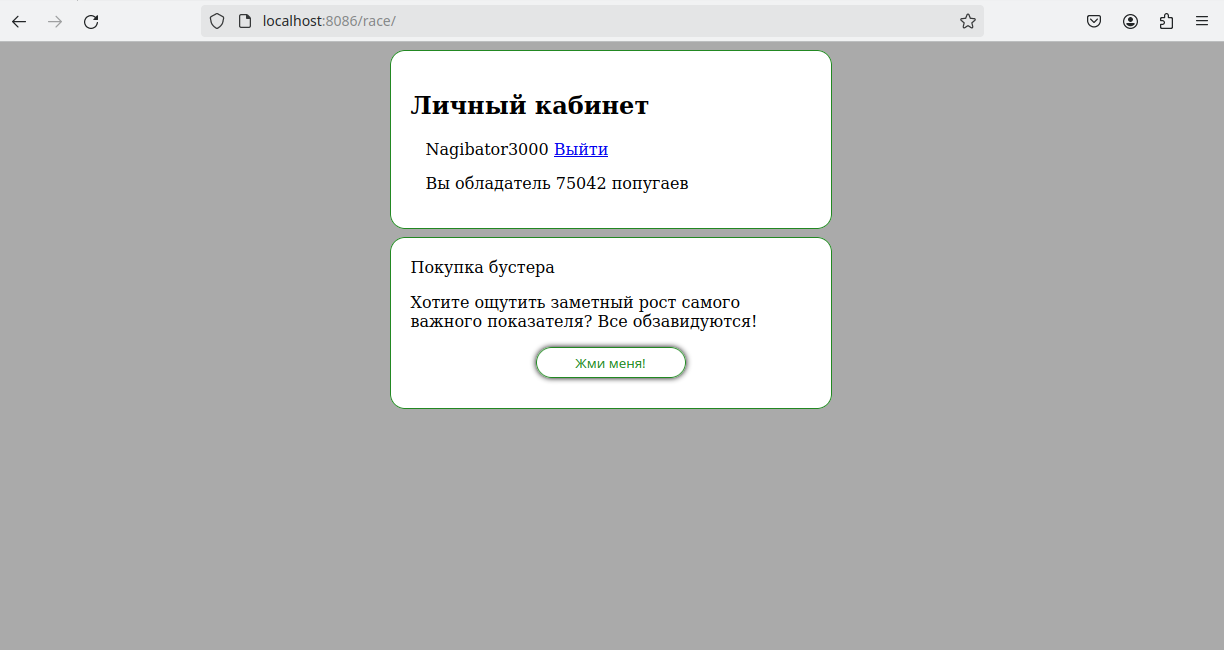
Как правило, такие атаки направлены на получения конфиденциальной информации о пользователе, паролей, и любой другой информации. Также это позволяет осуществлять различные операции на web ресурсах от лица легитимного пользователя, что может значительно упростить получение необходимых для злоумышленника данных.

Для защиты мы можем использовать следующие подходы:

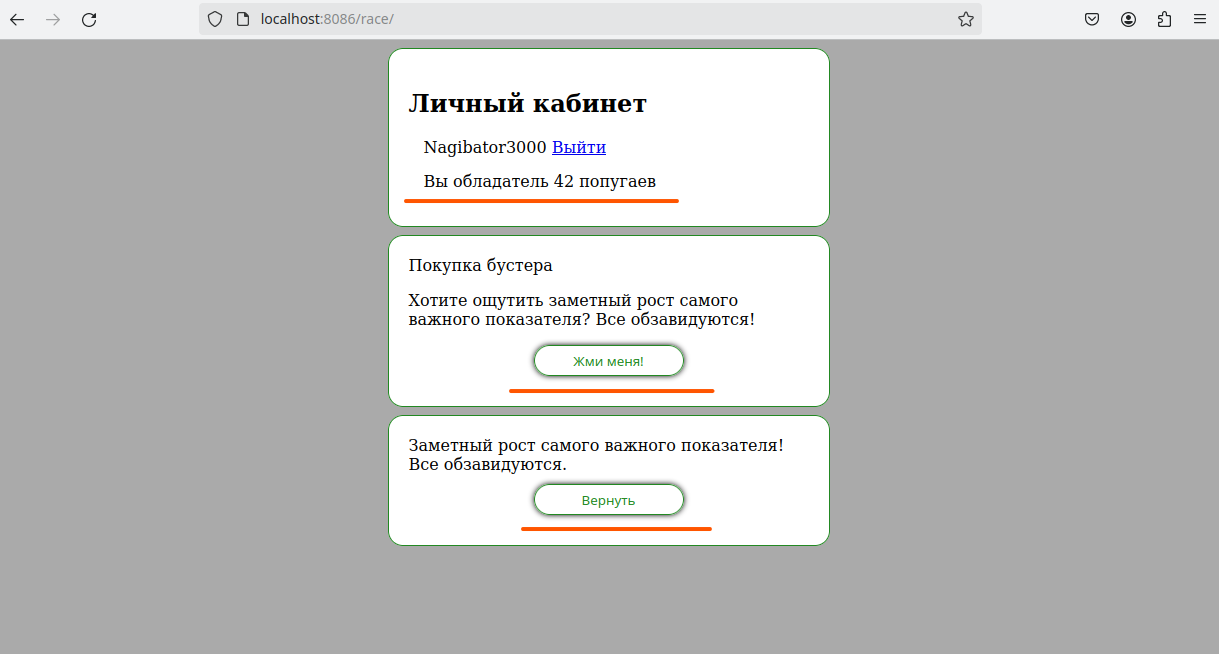
1. Использовать HttpOnly – атрибут, который запрещает доступ к конкретной куке через JavaScript. Также можно использовать secure – атрибут, указывающий на то, что данные будут передаваться в зашифрованном виде. SameSite = None|Lax|Strict – это достаточно новый атрибут, который имеет 3 значения, None – никак не регулируется, Lax – это значит, если обращение происходит к элементу странички, а не переходим куда-то то у нас кука посылаться не будет, а если переходим по ссылке то будет посылаться, Strict – кука не посылается вообще никогда.
2. Использование случайных и уникальных идентификаторов сеанса: При создании идентификаторов сеанса следует использовать достаточно длинные случайные значения, которые трудно подделать. Например, можно использовать криптографически безопасные генераторы случайных чисел.
3. Привязка идентификатора сеанса к IP-адресу: Это может предотвратить атаки, когда злоумышленник пытается использовать чужой идентификатор сеанса с другого IP-адреса.
4. Использование HTTPS: Шифрование передаваемых данных с помощью HTTPS защищает идентификатор сеанса от перехвата злоумышленником.
5. Ограничение времени жизни сеанса: Устанавливайте короткий срок действия сеанса, чтобы уменьшить возможное время, в течение которого злоумышленник может использовать поддельный идентификатор сеанса.
6. CSRF токен – это такой токен, который прикрепляется к форме отправки данных (дополнительные данные которые сверяются на сервере). Что это дает? Это дает то, что формируются дополнительные параметры при передаче данных и злоумышленник не знает о таких параметрах.

Г) Условие гонки:

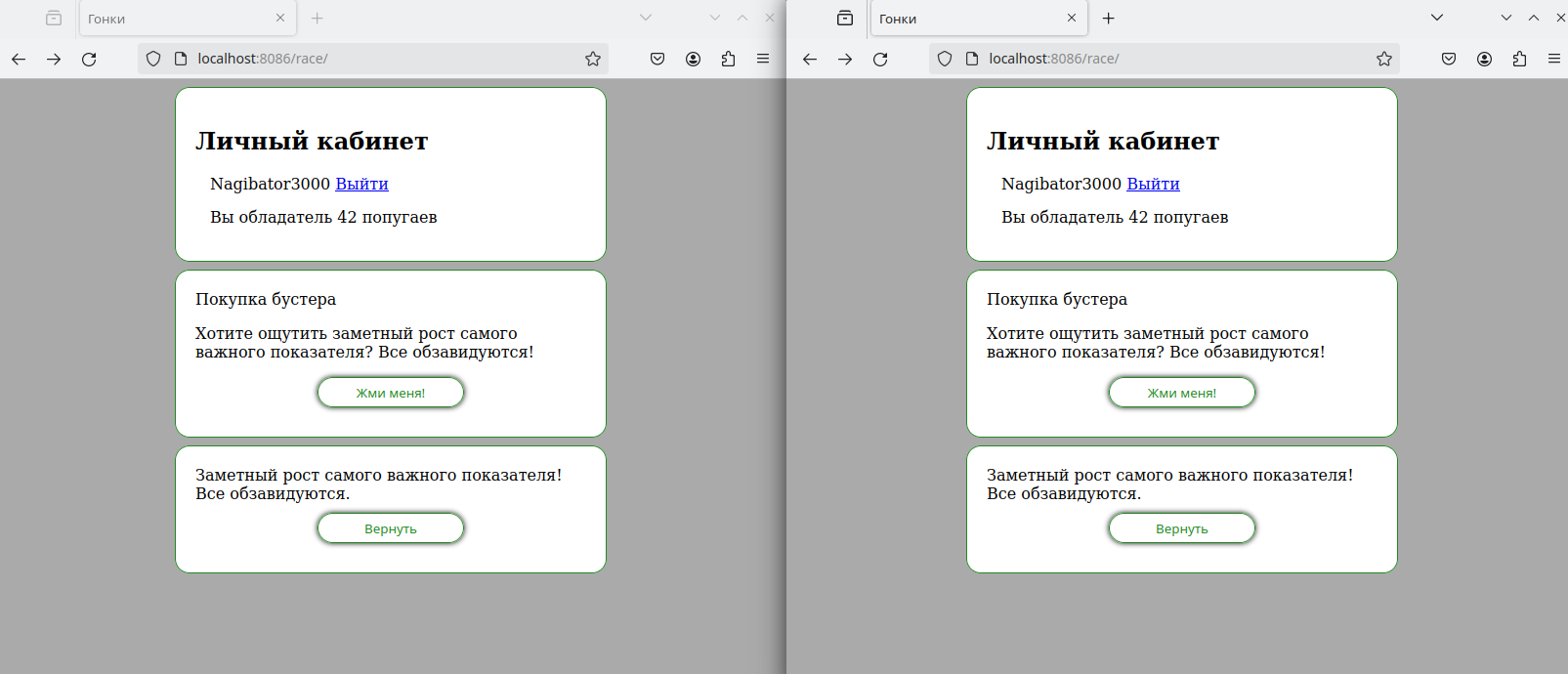
Авторизировались под пользователем Nagibator3000:



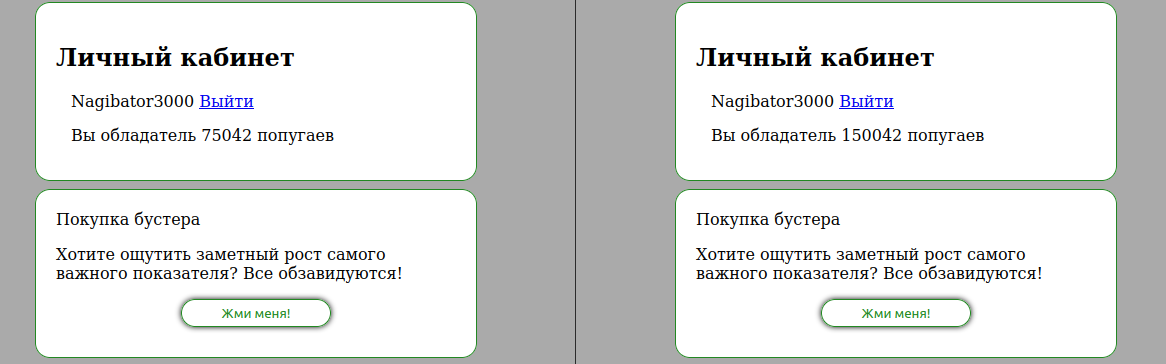
Теперь нажмем на кнопку и получаем следующую страницу:



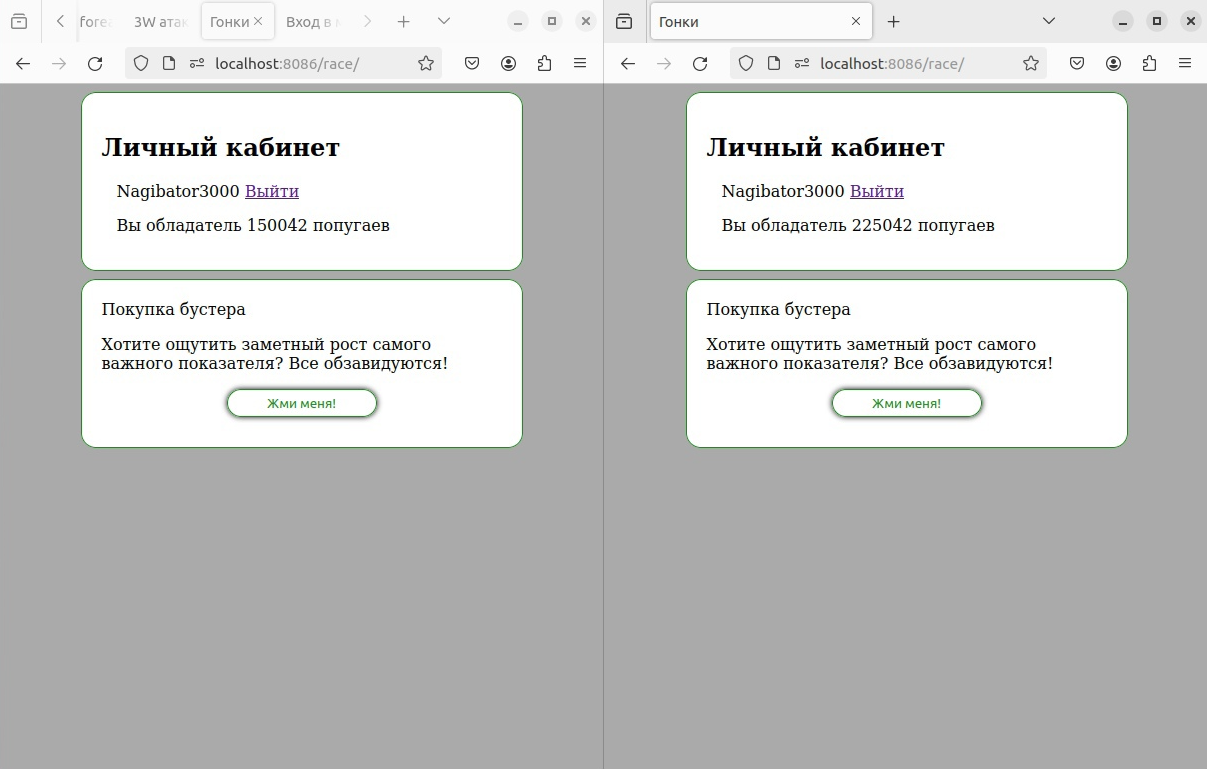
Продублируем эту страницу с целью попытки нарушения ее работы:



Теперь осуществим быстрое нажатие, на кнопку вернуть в двух окнах, и получим, что средства были увеличены:



Теперь попробуем снова осуществить возврат средств и получим опять увеличенное значение:



1. Механизм подобной уязвимости и проведения атаки.

Запрос разбит на части: проверка и выполнение, и любые другие операции, осуществляемые в приложении. Проведение атаки заключается в нарушении синхронизации операций, в результате чего происходят операции над неправильными данными.

1. Как защищаться?

Использовать механизм транзакций (очень ресурсоемкая операция)

Использовать очередь запросов\исполнений задач