

Правительство Российской Федерации

Федеральное государственное автономное образовательное учреждение
высшего профессионального образования «Национальный
исследовательский университет «Высшая школа экономики»

Московский институт электроники и математики Национального исследовательского
университета "Высшая школа экономики"

Отчет по производственной практике В компании АО «Актив-Софт»

Работу выполнил студент группы СКБ-201
Работу проверил

Родионов Ф. Д.
Лось А. Б.

Москва - 2024

1 Информация о прохождении практики

Место проведения практики: АО "Актив-Софт".

Сроки проведения практики: 05.07.24 - 30.07.24

2 Формулировка вопроса

Выполнить обзор текущего состояния возможности доступа к смарт-картам из браузера через интерфейс WebUSB.

3 Постановка задачи

Часть 1: Разобраться, что такое Isolated Web App

Часть 2: Реализовать простейший Isolated Web App, использующий WebUSB API для доступа к Рутокен.

В рамках задания, в зависимости от статуса возможности доступа в описываемом окружении: предоставить исчерпывающие объяснения причин невозможности доступа или предоставить прототип, реализующий доступ к смарт-карте.

Артефакты:

1. Отчет, описывающий, что такое Isolated Web App.
2. PoC Isolated Web App с доступом к Рутокен.
3. Небольшая документация к PoC, поясняющая ключевые решения.

4 Актуальность с точки зрения задач образовательной программы и новизна результатов

Результаты проведенного исследования, касающиеся особенностей работы WebUSB в Isolated Web App с различными комбинациями операционных систем и архитектур процессоров, вносят новый вклад в область изучения совместимости и безопасности изолированных веб-приложений с различными устройствами. Это исследование уточняет существующие данные о влиянии аппаратных и программных платформ на функциональность и защищенность веб-приложений, что является критически важным для разработчиков и специалистов по информационной безопасности.

Проделанное исследование и рабочий прототип описывают работу технологии IWA, вышедшей в Chrome 128, а так же ее интеграции с WebUSB API. С точки зрения новизны результатов, реализованное веб-приложение позволяет подключаться к девайсам из списка защищенных устройств через USB из браузера внутри Isolated Web App, что является новым и оригинальным исследованием.

5 Isolated Web App

Web USB – это JavaScript API, который даёт возможность веб-приложениям взаимодействовать с локальными USB-устройствами на компьютере. Согласно спецификации WebUSB, некоторые классы интерфейсов защищены от доступа веб-приложений, чтобы предотвратить доступ вредоносных скриптов к потенциально чувствительным данным.

Спецификация WebUSB определяет **блок-лист уязвимых устройств** и таблицу защищённых классов интерфейсов, доступ к которым заблокирован через WebUSB (Аудио, видео, HID, запоминающие устройства, смарт-карты, беспроводные контроллеры (Bluetooth и беспроводной USB)). С разрешением на использование функции «usb-unrestricted» изолированные веб приложения смогут получить доступ к устройствам из блок-листа и защищённым классам интерфейсов.

API WebUSB требует, чтобы приложение запрашивало разрешение у пользователя для каждого устройства, к которому оно хочет получить доступ. Доступ к функции WebUSB также контролируется политикой разрешений (Permissions Policy) с помощью функции «usb».

Политика безопасности контента (CSP) обеспечивает надежную защиту от уязвимостей межсайтового скриптинга (XSS). Протокол TLS и механизм проверки целостности подресурсов (SRI) защищают ресурсы от подмены во время передачи или при размещении на сторонних серверах.

Это предложение вводит новую функцию политики разрешений «usb-unrestricted». Приложению с функцией «usb-unrestricted» разрешен доступ к защищенным USB-интерфейсам и USB-устройствам, внесенным в заблокированный список.

Функция «usb-unrestricted» раскрывает возможности устройства, которые считаются слишком опасными для доступа недоверенных приложений (untrusted application). Недоверенные приложения не смогут запрашивать «usb-unrestricted». Согласно этому предложению, «usb-unrestricted» можно использовать только в изолированных веб-приложениях, включив эту функцию в поле манифеста «permissions policy».

Изолированные веб-приложения (IWA) - это развитие существующих технологий установки прогрессивных веб-приложений (PWA) и веб-упаковки. Веб-пакеты позволяют объединять группы веб-ресурсов для их совместной передачи. Эти пакеты можно подписать, чтобы установить их подлинность. IWA обеспечивают усиленную защиту от взлома сервера и других видов несанкционированного доступа, что особенно важно для разработчиков приложений, работающих с конфиденциальными данными.

Isolated Web App – это приложения, не размещённые на реальных веб-серверах, а упакованные в Web Bundles, подписанные их разработчиком и распространённые среди конечных пользователей. Обычно они создаются для внутреннего использования компаниями.

Ключевое отличие изолированных веб-приложений (IWA) от обычных веб-страниц заключается в том, что IWA не полагаются на разрешение доменных имен (DNS) и сертификаты HTTPS. Вместо этого они должны быть явно загружены и установлены пользователем, например, из магазина приложений или через корпоративную конфигурацию. Браузер может проверить целостность IWA, проверив подпись и сравнив соответствующий открытый ключ с списком известных доверенных открытых ключей.

6 Ход работы

6.1 WebUSB API внутри IWA

Данное исследование направлено на изучение работы WebUSB API с устройствами, входящими в блок-лист устройств, с помощью изолированных веб приложений (IWA).

В ориентира для создания IWA, был взят **telnet**.

Было обнаружено, что часть задачи, совпадающей с той, что была получена в начале уже выполнена в проекте **WebUSBAuth**.

Проект демонстрирует взаимодействие со смарт-картой через WebUSB API с помощью APDU команд. Но, поскольку смарт карта входит в список запрещенных устройств, было предложено использовать архитектуру изолированного веб-приложения, реализованного в проекте **telnet**. Использование изолированного приложения позволяет обойти ограничения черного списка WebUSB API, предоставляя разработчикам доступ к функционалу смарт-карт.

6.2 WebUSBAuth тестовый запуск

Было решено протестировать проект **WebUSBAuth** на MacOS. Далее - подробная инструкция по сборке и запуску проекта в Chrome 129.

1. Предварительно скачать архив проекта.
2. Предварительно скачать **python**.
3. С помощью терминала перейти в директорию проекта.
4. Запустить python сервер.
5. В браузере перейти по адресу <http://localhost:8000/demo.html>

При нажатии на кнопку connect reader и выборе нужного устройства *Pytoken* в панели выводится ошибка при вызове функции *claimInterface*, которая происходит при попытке получения доступа к устройствам из блок-листа WebUSB API.

Для решения проблемы необходимо реализовать этот проект в изолированном приложении.

Но на данном этапе еще не было известно, что ошибка заключается именно в этом, потому было решено «подсмотреть» как Linux общается с токеном с помощью специальных инструментов:

1. Для этого установить Debian (ARM64) в виртуальной машине UTM.
2. Использовать следующие инструменты:
 - (a) **libccid**: Библиотека для работы со смарт-картами и токенами в Linux.
 - (b) **pcscd**: Фоновая программа для работы с USB-ридерами смарт-карт (необходимо настроить максимальное логирование).
 - (c) **opensc-tool**: Утилита командной строки для отправки APDU-команд на токен.
3. Проанализировать логи pcscd и определить формат ccid-пакетов, которые отправляются на токен (там залоггируются ccid-пакеты, если установлен LIBCCID ifdLogLevel=0x0F).
4. Полученные ccid-пакеты отправлять на токен в BULK-IN через WebUSB; вычитывать результат из BULK-OUT.

6.3 Запуск Debian на UTM и Windows 11 на Parallels

Было решено использовать виртуальную машину UTM и с ее помощью запустить Debian 12 на macOS.

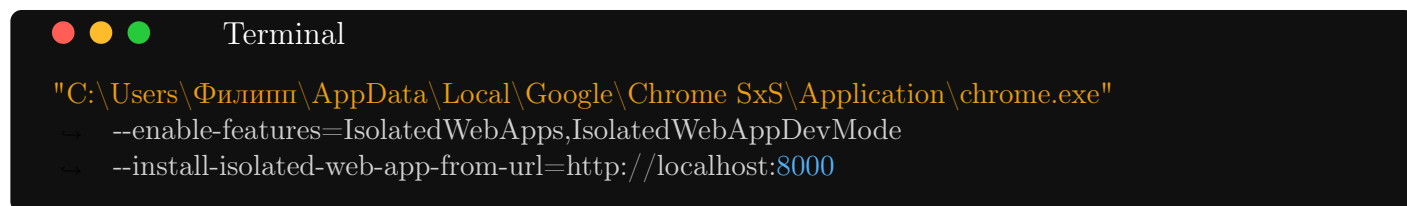
Проблема заключалась в том, что на ARM впринципе невозможно выполнить данное задание по причине того, что поддержка функции *usb-unrestricted* добавлена в Chrome 128, работающий на движке Blink. Но Chrome на macOS, на процессоре Apple Silicon (архитектура ARM), работает на движке WebKit \Rightarrow *usb-unrestricted* на нем не функционирует.

Так же была предпринята попытка запустить Windows 11 на виртуальной машине Parallels для проверки работы проекта WebUSBAuth, но, как позже выяснилось: "В настоящее время невозможно подключить какое-либо USB-устройство к виртуальной машине MacOS Arm. из документации Parallels.

6.4 Запуск проекта на Windows 10 (x64)

Следующим шагом мной был предпринят запуск проекта на ОС Windows 10 на процессоре x86-64. В отличие от запуска в Linux, где необходима установка Google Chrome unstable (версия Chrome для разработчиков), на Windows потребовалось установить Chrome Canary.

Ниже представлен измененный вариант запуска и установки IWA под Windows.



```
Terminal
"C:\Users\Филипп\AppData\Local\Google\Chrome SxS\Application\chrome.exe"
--enable-features=IsolatedWebApps,IsolatedWebAppDevMode
--install-isolated-web-app-from-url=http://localhost:8000
```

После установки изолированного приложения Chrome не разрешал работать встроенным JS скриптам внутри html разметки файла *index.html* и в Dev Tools появлялась ошибка *"Refused to execute inline event handler because it violates the following Content Security Policy directive: 'script-src 'self''. Either the 'unsafe-inline' keyword..."*. Она решилась путем добавления Content Security Policy в *manifest.json* и в *index.html*, а так же выносом JS скрипта в отдельный файл *main.js*.

Далее, при попытке соединения с токеном, возникает ошибка : *"Failed to execute 'open' on 'USBDevice': Access denied"*. Windows ограничивает доступ к смарт-картам как к USB-устройству, что подтверждено в summary описания ограничений доступа к классам устройств из WebUSB. *"These interface classes are already mostly blocked by an operating system's built-in class drivers."*

6.5 Успешный запуск проекта на Ubuntu VirtualBox (x64)

Финальным этапом было решено загрузить бесплатную виртуальную машину VirtualBox на компьютер с процессором x86-64 с дистрибутивом Ubuntu (24.04).

Проект успешно запустился и на токен стало возможным отправлять APDU-команды. Был изменен файл *cardu.js*, чтобы при отправке команды в консоль выводился серийный номер токена, по условию задания. Ниже представлена часть, где была изменена команда.

```
1
2
3 let apdus = {
4   GET_CHALLENGE: (length) => {
5     let apdu = new Uint8Array( [0x00, 0xCA, 0x01, 0x81, 0x08] );
6     return apdu;
7   },
8 };
```

```
9  
10 export {buildExtendedAPDU, apdus};
```

7 Заключение

В ходе прохождения практики были реализованы:

1. Работающий прототип Web USB Isolated выполняющий запрос серийного номера смарт-карты из браузера.
2. Отчет, описывающий текущее состояние возможности доступа к смарт-картам из браузера через интерфейс WebUSB в различных средах.
3. Готовый собранный проект, выложенный на GitHub.

8 Источники

1. API WebUSB
2. Поддержка API WebUSB в различных браузерах
3. Блок лист устройств, которым ограничен доступ API WebUSB
4. Использование любых классов usb-устройств возможно из Isolated Web Apps, получивших разрешение "usb-unrestricted"
5. Подробная информация об Isolated Web App