

Отчет по практике "Рутокен".

Родионов Филипп

17.07.24

1 Формулировка вопроса

Предоставить отчет, описывающий, что такое Isolated Web App и реализовать IWA, использующий Web USB для доступа к Рутокен.

2 Постановка задачи

Часть 1: Разобраться, что такое Isolated Web App

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

РоС должен обнаруживать устройство, подключаться к нему, отправлять APDU-команду 00 CA 01 81 08 (это команда получения значения серийного номера токена) и получать корректный ответ (можно сравнить с тем, что отображается в Панели Управления Рутокен).

Артефакты:

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

3 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. С разрешением на использование функции «usb-unrestricted» изолированные веб приложения смогут получить доступ к устройствам из блок-листа и защищённым классам интерфейсов.

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. Иными словами, веб приложение (сами файлы) хостится на компьютере конечного пользователя.

4 Ход работы

4.1 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 качестве готового проекта, содержащего функции 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 при помощи 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.

4.2 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 *Рутокен* 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 я еще не знал, что ошибка заключается именно в этом, потому решил «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.
4. Использовать полученные данные для реализации взаимодействия с токеном через WebUSB API в изолированном веб-приложении.

4.3 Запуск Debian на UTM и Windows 11 на Parallels

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

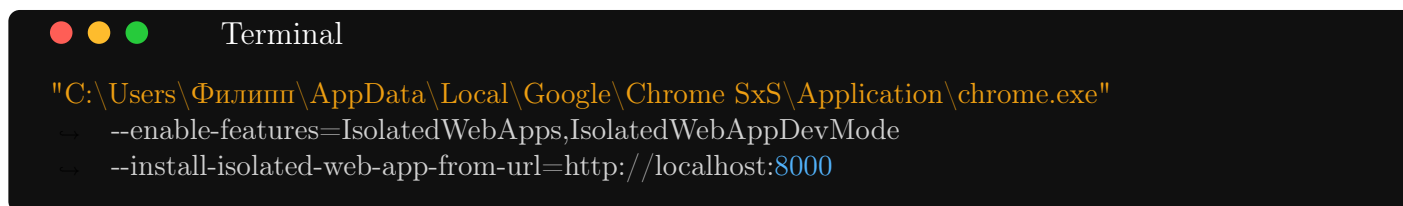
Проблема заключалась в том, что на ARM впринципе невозможно выполнить данное задание по причине того, что поддержка функции *usb-unrestricted* добавлена в Chrome 128, работающий на движке Blink. Но Chrome на macOS, на процессоре Apple Silicon (архитектура ARM), работает на движке WebKit \Rightarrow *usb-unrestricted*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

4.4 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 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 соединиться с токеном, нажав на кнопку *"connect reader"*, в консоли возникла ошибка: *"Failed to execute 'open' on 'USBDevice': Access denied"*. Как выяснилось, Windows ограничивает доступ к смарт-картам, как к USB-устройству.

Это 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."*

4.5 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};
```