Análisis de seguridad de dispositivos hardware de firma de criptomonedas: El caso SeedSigner

Jordi Pradas-López

2 de julio de 2024

Resumen– Este artículo analiza la seguridad del SeedSigner, un dispositivo "hágalo usted mismo" para la firma de transacciones de Bitcoin, y evalúa su resistencia a ataques conocidos. El objetivo principal era determinar si el SeedSigner es seguro, a pesar de utilizar componentes estándar sin propósitos criptográficos. Las fases del trabajo han incluido la recopilación de información, el modelado de amenazas, el análisis de vulnerabilidades y la explotación de las mismas para desarrollar pruebas de concepto que ilustren los hallazgos. Se ha descubierto que el dispositivo presenta dos vulnerabilidades de severidad alta y media, respectivamente. Sin embargo, se ha determinado que, dadas las condiciones necesarias para explotar dichas vulnerabilidades, el SeedSigner puede considerarse un dispositivo seguro si se emplea adecuadamente.

Palabras clave– Análisis de seguridad, Bitcoin, jaqueo de equipo, *pentest*, pruebas de penetración, SeedSigner

Abstract– This paper analyzes the security of the SeedSigner, a do-it-yourself device for signing Bitcoin transactions, and evaluates its resistance to known attacks. The main objective was to determine whether the SeedSigner is secure, despite using standard components with no cryptographic purpose. Phases of the work have included information gathering, threat modeling, vulnerability analysis and exploitation of vulnerabilities to develop proof-of-concepts to illustrate the findings. The device has been found to have two vulnerabilities of high and medium severity, respectively. However, it has been determined that, given the conditions necessary to exploit these vulnerabilities, the SeedSigner can be considered a secure device when used properly.

Keywords– Bitcoin, hardware hacking, penetration testing, pentest, security analysis, SeedSigner

1 Introducción

El SeedSigner es un dispositivo *"Do It Yourself"* (DIY) que permite realizar firmas digitales, necesarias para autorizar pagos en criptomonedas. Su objetivo, según su página web [1], es reducir el coste y la complejidad de los monederos de firma múltiple utilizados en Bitcoin. Es por ello que es un dispositivo de código abierto y sus componentes electrónicos son económicos y accesibles. Sin embargo, el hecho de que no se utilice *hardware* diseñado para usos criptográficos y resistente a ciertos tipos de ataques, como el que usan los *hardware wallets* más famosos del mercado, genera dudas acerca de su seguridad.

En este Trabajo Final de Grado (TFG), se pretende comprender el funcionamiento del dispositivo y realizar un ejercicio de *penetration testing* para analizar la resistencia del SeedSigner a los ataques ya empleados en el pasado contra dispositivos similares y, en caso de detectar alguna vulnerabilidad, desarrollar una prueba de concepto que permita mostrar cómo se ha realizado el ataque e informar a los responsables del dispositivo para que puedan estudiar posibles mitigaciones.

La motivación para realizar este proyecto surge del in-

• E-mail de contacto: jordi.pradasl@autonoma.cat
• Mención realizada: Tecnologies de la Informació
• Trabajo tutorizado por: Cristina Pérez-Solà (DEIC)
• Curso 2023/24

terés por aprender acerca de *hardware hacking* y de la oportunidad para ello que proporcionan los dispositivos como los monederos *hardware*, vistos en la asignatura de Tecnología Blockchain y Criptomonedas [2].

1.1. Objetivos

Dada la naturaleza experimental de este trabajo de investigación, se han establecido: un objetivo principal, dos prioritarios y dos secundarios. Como se detallará a continuación, la consecución de los objetivos prioritarios y secundarios está condicionada por los resultados del principal.

- **O1: Realizar un análisis de seguridad del SeedSigner**
  El objetivo principal del proyecto es determinar, mediante un procedimiento similar al utilizado en las denominadas pruebas de penetración del campo de la seguridad informática, si el dispositivo se puede considerar seguro frente a algunos tipos de ataques conocidos pese a utilizar componentes no diseñados específicamente para realizar operaciones criptográficas.

- **O2: Robar una de las claves privadas generada al introducir la semilla (BIP-39 [3]) de generación de claves en el dispositivo**
  Es un objetivo secundario del proyecto, ya que depende de los resultados del principal, y consiste en encontrar la forma de acceder a una clave privada perteneciente al usuario legítimo del SeedSigner y extraerla, desde el punto de vista de un actor malicioso con acceso físico al dispositivo. Para ello, se emplearán técnicas utilizadas en pruebas de penetración de *hardware* y de *software*.

- **O3: Desarrollar una prueba de concepto**
  Si se diera el caso de que, a través del análisis de seguridad, se encontrasen vías que pudieran permitir a un atacante extraer una clave privada del dispositivo sin el consentimiento de su dueño, sería un objetivo secundario crear una prueba de concepto en el formato que se considere oportuno, con la finalidad de ilustrar la metodología empleada por el atacante para llevar a cabo dicha extracción ilícita.

- **O4: Documentar el estado de seguridad**
  Es un objetivo prioritario del proyecto documentar el estado de seguridad del SeedSigner con respecto a los vectores de ataque contemplados y clasificar cualitativamente la severidad de las vulnerabilidades encontradas, si fuera el caso, utilizando el *Common Vulnerability Scoring System (CVSS)* [4] definido por el *National Institute of Standards and Technology (NIST)*.

- **O5: Llevar a cabo una divulgación responsable**
  Reportar al equipo de desarrollo del SeedSigner las vulnerabilidades encontradas que se consideren mitigables o remediables, es un objetivo prioritario del proyecto. Esta divulgación deberá realizarse mediante canales privados para permitir que los responsables del producto lleven a cabo la correspondiente evaluación de riesgos y, si lo creen conveniente, la implementación de parches de seguridad.

La estructura del documento, a partir de aquí, es la siguiente: estado del arte, en el que se mencionan otros análisis de seguridad a monederos *hardware*; metodología y planificación, donde se detalla la forma de trabajar y se resume la planificación; descripción del dispositivo, donde se expone la información más relevante extraída en la fase de recopilación de información; modelado de amenazas, en el que se aprecian las primeras ideas surgidas de la adopción de la mentalidad de un actor malicioso; análisis de vulnerabilidades, en el que se detallan los procesos seguidos para intentar establecer una conexión con el dispositivo y/o extraer información sensible; prueba de concepto, sección en la que se narra cómo se ha demostrado que es factible realizar ciertos ataques; evaluación de resultados, donde se introduce el sistema de evaluación utilizado y se clasifican las vulnerabilidades encontradas; conclusiones; agradecimientos y, finalmente, bibliografía y apéndice.

2 Estado del Arte

El campo del análisis de seguridad de carteras de criptomonedas, en constante evolución, es esencial para garantizar la integridad y la privacidad de las transacciones digitales. Dado que la protección de las frases semilla y las claves privadas es la base de la seguridad de los fondos de los usuarios, la gestión que hacen las carteras de las mismas es una preocupación principal.

Estudios como *BlueWallet: The Secure Bitcoin Wallet* [5] enfatizan la importancia de la arquitectura de seguridad de los monederos de Bitcoin, destacando la necesidad de métodos sólidos para administrar las claves privadas.

Además, investigaciones sobre métodos de prueba de seguridad, como la obra *Evaluation of Security in Hardware and Software Cryptocurrency Wallets* [6], proporcionan marcos metodológicos esenciales para evaluar la seguridad de las implementaciones de los monederos de Bitcoin.

3 Metodología

En esta sección, se expone el método de trabajo empleado y se detallan las fases de la planificación, claves para obtener resultados consistentes y evitar aproximaciones caóticas.

3.1. Metodología de Trabajo

Dadas la magnitud, el enfoque y la tipología del proyecto, se ha decidido seguir la metodología de investigación experimental que Briony J. Oates adapta al campo de los sistemas de información y computación en su libro *Researching Information Systems and Computing* [7].

Este método facilita la investigación de relaciones causales entre variables mediante la manipulación de una variable independiente y la observación de los efectos que esta provoca en una variable dependiente, a la par que se controlan variables de confusión de forma sistemática. A continuación, se expondrán los elementos clave de dicha metodología.

Diseño experimental: Al aplicar la metodología, se diseñarán experimentos controlados con el objetivo de investigar fenómenos específicos. De esta forma, se identificarán

y definirán con precisión las variables independientes, dependientes y controladas.

En este caso, el diseño de los experimentos tiene en cuenta los tipos de ataques, la situación del atacante, el momento del ataque y la configuración del dispositivo.

Manipulación de variables: El uso de esta metodología implica la manipulación deliberada de una o más variables independientes con el objeto de observar cómo afectan a la variable dependiente. Briony J. Oates contextualiza las variables independientes en el campo de la informática, pudiendo ser estas diferentes configuraciones de *software*, *hardware* u otros parámetros de sistemas informáticos. En nuestro caso, la variable independiente será el tipo de ataque utilizado y la variable dependiente podría ser una medida del rendimiento del sistema, la seguridad, la vulnerabilidad explotada o el acceso no autorizado obtenido.

Control de variables de confusión: Las variables de confusión son aquellas que distorsionan la medida de la asociación entre dos variables. Por lo tanto, tienen la capacidad de influir en los resultados del experimento creando una correlación falsa entre las variables dependientes e independientes.

Para evitar esto, Briony J. Oates propone controlar las variables de confusión mediante técnicas como la aleatorización de los participantes, la asignación aleatoria a diferentes condiciones experimentales o el uso de grupos de control que reciban el mismo tratamiento en todas las variables a excepción de la independiente.

Recopilación y análisis de datos: La metodología de investigación experimental propone que se lleve a cabo una recopilación de datos sobre la variable dependiente en todas las condiciones experimentales. Posteriormente, sugiere el análisis de los datos recopilados con el fin de identificar patrones, tendencias y relaciones causales entre las variables.

Interpretación de resultados: Como última etapa de este enfoque, se interpretan los resultados en el contexto de las preguntas de investigación y se discuten las implicaciones de los hallazgos, en nuestro caso, en materia de seguridad informática. Adicionalmente, se pueden identificar recomendaciones para mejorar la seguridad del sistema analizado y proceder a su reporte si se considera oportuno.

3.2. Fases de la Planificación

La planificación del proyecto se ha dividido en 5 fases, adaptando algunas de las mencionadas en un estándar del sector, llamado *Penetration Testing Execution Standard (PTES)* [8], a la magnitud y las condiciones de este trabajo.

Recopilación de información: En esta fase se investigará y recopilará toda la información posible del SeedSigner, incluyendo su arquitectura, componentes, documentación técnica, código, funcionalidades, protocolos de comunicación y posibles vulnerabilidades conocidas. Además, cualquier información adicional que se considere relevante será almacenada para su posterior análisis y utilización. Para llevarla a cabo, se usarán técnicas de obtención de información de fuentes abiertas y se utilizará el dispositivo como lo haría un usuario legítimo del mismo.

Modelado de amenazas: Esta fase consiste en identificar posibles vectores de ataque contra el dispositivo, que permitan comprometer información sensible del usuario o sus claves privadas, tomar el control del propio dispositivo o afectar a la integridad de las transacciones realizadas por el usuario. Adicionalmente, se puede hacer una primera evaluación del impacto potencial que pueden tener las amenazas modeladas sobre el usuario y sus transacciones.

Análisis de vulnerabilidades: En esta fase se utilizarán técnicas de análisis de vulnerabilidades para identificar puntos débiles en el dispositivo. Entre estas técnicas se encontrarán las usadas en pruebas de penetración de caja blanca y caja negra, como análisis de la configuración del dispositivo y revisión de código. Además, se utilizarán técnicas propias de la rama del *hardware hacking*, como el análisis de consumo y otros ataques de canal lateral.

Explotación y desarrollo de la prueba de concepto: Si durante la fase de análisis se identifican vulnerabilidades explotables, se realizarán pruebas adicionales centradas en dichas vulnerabilidades y se desarrollará una prueba de concepto en un formato que permita ilustrar al público cómo un actor malicioso podría llegar a aprovechar los fallos de seguridad encontrados para comprometer la seguridad del dispositivo o la de los fondos y transacciones del usuario de la *blockchain*.

Reporte y realización del informe: La última fase del proyecto consistirá en redactar el artículo que documenta todo el proceso del trabajo y los resultados de la evaluación de seguridad del SeedSigner, así como el estado del arte, los objetivos, la metodología empleada, la planificación seguida, las líneas de investigación futuras y la bibliografía. También se preparará una presentación oral para defender el trabajo delante de un tribunal compuesto por miembros del profesorado y un póster que resuma el proyecto de forma rápida y visual. En paralelo, si se han encontrado vulnerabilidades cuyo nivel de severidad pueda afectar gravemente a los usuarios del dispositivo de firma de transacciones, se contactará con los responsables del SeedSigner para realizar una divulgación responsable y permitirles remediar las vulnerabilidades y/o advertir a sus usuarios de las mismas.

4 Descripción del Dispositivo

En esta sección se mostrarán las características principales del *hardware*, del SO y de la aplicación del SeedSigner.

4.1. Hardware: Raspberry Pi Zero W

A continuación se verá el hardware recomendado por los desarrolladores de SeedSigner para su correcto uso.

4.1.1. Esencial

El proceso de montaje precisa de, cómo mínimo, los siguientes elementos:

- Raspberry Pi Zero (preferiblemente v1.3, W aceptable).
- Pantalla LCD WaveShare 1.3 pulgadas con controles (240x240 píxeles).

- Módulo de cámara compatible con Raspberry Pi Zero.
- Tarjeta MicroSD con un mínimo de 64MB de espacio de almacenamiento.

4.1.2. Raspberry Pi Zero W

Para llevar a cabo este proyecto, se ha utilizado una Raspberry Pi Zero W cuyos principales componentes son:

- **CPU:** Broadcomm BCM2835 ARM11 a 1GHz.
- **GPU:** VideoCore IV GPU.
- **RAM:** 512MB.
- **Almacenamiento:** ranura microSD.
- **Salida de vídeo:** salida mini HDMI y vídeo compuesto sin montar el conector.
- **Puerto *Universal Serial Bus (USB)* de datos:** micro USB On-The-Go (OTG).
- **Wi-Fi y *Bluetooth*:** 802.11 b/g/n *wireless Local Area Network (LAN)*, *Bluetooth* 4.1 y *Bluetooth Low Energy (BLE)* en el chip Cypress CYW43438.
- ***General-Purpose Input/Output (GPIO)*:** Cabezal de 40 patillas compatible con *Hardware Attached on Top (HAT)*.
- **Alimentación:** 5V vía micro USB.
- **Extra:** conector de cámara *Camera Serial Interface (CSI)*.

Los pines *General-Purpose Input/Output (GPIO)*, comunes en la mayoría de modelos de Raspberry Pi, ofrecen la posibiliad de conectar dispositivos externos a la Raspberry y establecer comunicación entre chips para diversas finalidades. Algunos de estos pines permiten el uso de protocolos de comunicación serie, empleados habitualmente en una gran cantidad de sistemas embebidos, como *routers* o dispositivos *Internet of Things (IoT)*.

Además de los protocolos ofrecidos en los pines GPIO, también encontramos otros protocolos de comunicación hardware como el conocido *Universal Serial Bus (USB)* o el protocolo de transferencia de datos entre dispositivos de imagen, *Mobile Industry Processor Camera Serial Interface 2 (MIPI CSI-2)*.

Antes de realizar las pruebas de acceso físico, es importante conocer el funcionamiento de aquellos protocolos de comunicación hardware que, en caso de estar activados, podrían brindar información de las comunicaciones entre los componentes del dispositivo o, incluso, permitir el acceso al sistema de ficheros del SeedSigner. Dado que, durante las pruebas de acceso físico, conocer los protocolos de los pines GPIO 1 será de utilidad, se han resumido los más relevantes para este trabajo en el apéndice B.1.

4.2. SO: SeedSigner OS

El SeedSigner OS es un sistema operativo basado en Linux y diseñado para reducir la superficie de ataque, ya que los desarrolladores optaron por incluir únicamente aquellos módulos del *kernel* y librerías absolutamente imprescindibles para realizar las funciones que ofrece el dispositivo [1].

4.2.1. Configuración e Instalación

La página oficial de SeedSigner [1] ofrece una gran cantidad de documentación y recursos gráficos que muestran cómo configurar e instalar el sistema, por lo que se ha utilizado para preparar el dispositivo para el análisis.

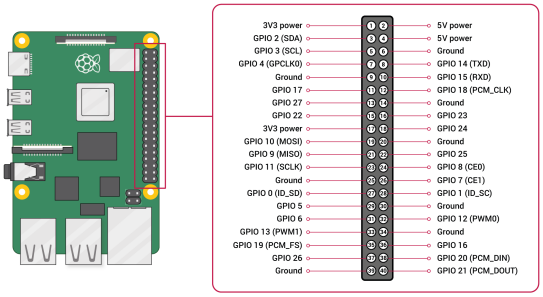

Fig. 1: *Raspberry Pi GPIO Pinout* [9]

Gracias a la lectura de la documentación, se ha descubierto que el programa de código abierto que adaptaron y utilizaron para construir el SeedSigner OS es Buildroot, lo que puede resultar relevante para construir imágenes modificadas en el futuro.

4.2.2. Proceso de Arranque

La figura 11 resume el proceso de arranque de la Raspberry Pi aunque, en el caso del SeedSigner, cambia ligeramente:

1. Al encenderse la Raspberry Pi, la *Graphics Processing Unit (GPU)* inicia el *bootloader* de primera fase y lee la microSD. La *Synchronous Dynamic Random-Access Memory (SDRAM)* está desactivada.
2. La GPU lee el archivo bootcode.bin e inicia el *bootloader* de segunda fase, encargado de activar la SDRAM.
3. El *firmware* en start_x.elf, los archivos config.txt, cmdline.txt y el núcleo del sistema son leídos.
4. La *Central Processing Unit (CPU)* empieza a funcionar y arranca el núcleo del sistema.

4.2.3. Seguridad

Dado que el *SeedSigner OS* está diseñado para reducir la superficie de ataque y activar funcionalidades adicionales para las aplicaciones, se ha limitado la cantidad de módulos, librerías y binarios presentes en el sistema, reduciendo el número de ataques posibles. Algunas características de seguridad y ventajas funcionales destacables del SeedSigner OS son [10]:

- **Arranque completo desde la RAM:** la tarjeta microSD puede ser retirada tras ver la pantalla de inicio de SeedSigner, ya que no se necesita entrada/salida de disco después del arranque.
- **Una partición FAT32 en la tarjeta microSD:** los sistemas operativos más comunes para Raspberry Pi suelen tener 2 particiones, una para el arranque y otra para el sistema de ficheros.
- **Eliminados los siguientes módulos estándar del *kernel* de Raspberry Pi OS:** Bluetooth y Redes; SWAP; *Inter-Integrated Circuit (I2C)*; *Universal Asynchronous Receiver-Transmitter (UART)*; USB; y Modulación por Ancho de Pulsos (PWM).
- **Sin soporte HDMI.**

- **Sin soporte para conexión serie *Transistor-Transistor Logic (TTL)*.**
- **Ningún software compatible con chips inalámbricos o de red.**
- **Un único archivo zImage de sólo lectura en la partición de arranque que contiene todo el núcleo de Linux y el sistema de archivos.**

4.3. Aplicación: SeedSigner

La aplicación SeedSigner permite generar claves privadas de Bitcoin, sin la necesidad de confiar en sistemas externos, configurar monederos multifirma y facilitar las transacciones a través de un modelo de firma con intercambio de códigos QR y aislamiento de la red [1].

Entre las funcionalidades más destacables de la aplicación, se encuentran:

- Calcular las 12/24 palabras de una frase semilla BIP39 [3].
- Crear una frase semilla BIP39 realizando tiradas consecutivas de un dado de 6 caras.
- Crear una frase semilla BIP39 con la entropía derivada de realizar una fotografía digital.
- Crear un SeedQR [11] de forma manual y guiada.
- Añadir contraseña a la frase semilla.
- Generar clave pública extendida para Segwit nativo multifirma [3].
- Escanear y analizar datos de transacciones de códigos QR animados.
- Firmar transacciones y transferir la clave pública extendida usando códigos QR animados.
- Establecer ajustes de usuario con persistencia opcional.
- Retirar la tarjeta microSD una vez ha cargado el *software*.
- Disponer de vista previa en vivo durante la conversión de imagen a semilla y durante el escaneo de códigos QR.
- Introducir palabras semilla con una interfaz optimizada.
- Utilizar las redes principal (*mainnet*), de pruebas pública (*testnet*) y de pruebas privada (*regtest*) de Bitcoin.
- Utilizar rutas de derivación personalizadas.
- Derivar frases semilla BIP85 [3].
- Verificar direcciones de recepción y de cambio bajo demanda.
- Configurar densidad y brillo de los códigos QR.

En el apartado C del anexo se expondrán todas las funcionalidades de la aplicación, con su nombre original (en inglés), y se describirán una por una.

4.4. Ejemplo de uso legítimo

En este apartado se mostrará cómo utilizar el SeedSigner [12], con el objetivo de estudiar cómo lo haría un usuario estándar y, posteriormente, encontrar posibles vectores de ataque. A modo de ejemplo, se configurarán tanto el SeedSigner como la aplicación de monedero utilizada para trabajar en la *testnet*.

Antes de empezar, se recomienda desactivar WiFi y Bluetooth por hardware [13].

4.4.1. Encendido

Para encender el sistema basta con introducir la microSD y alimentar uno de los 2 puertos microUSB de la Raspberry Pi Zero W.

4.4.2. Generación de Semilla (BIP-39)

Para crear una nueva frase semilla, se puede utilizar el método de generación de entropía a partir de múltiples lanzamientos de un dado o a partir de la toma de una imagen. En este caso, se usará el método de la fotografía por ser más sencillo e igualmente efectivo:

1. Realizar una fotografía con la cámara instalada.
2. Elegir la longitud del mnemotécnico: 12 o 24 palabras.
3. Apuntar en la plantilla [11] y guardar en algún sitio muy seguro (preferiblemente con lápiz) las palabras semilla. Ejemplo en la figura 2.

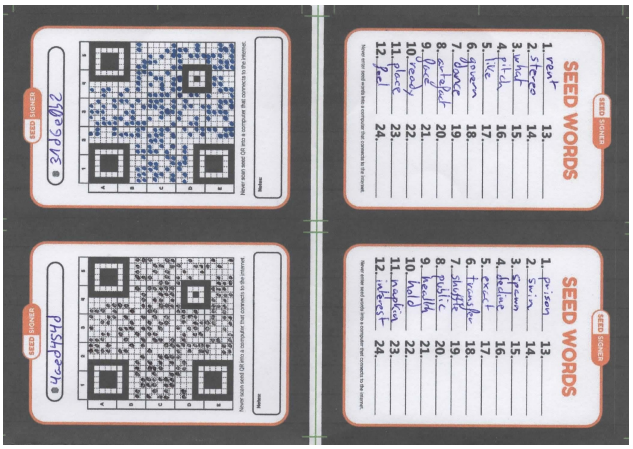

Fig. 2: Frases semilla en formato SeedQR

4. Verificar que se ha copiado bien la frase semilla.
5. (Opcional) Añadir una contraseña [3] como medida extra de seguridad ante el robo de la frase semilla.
6. Elegir “Export Xpub” entre las opciones que han aparecido.

Se debe tener en cuenta que la frase semilla no es persistente, por lo que se borrará al reiniciar el dispositivo.

4.4.3. Importar la Clave Pública Extendida (Xpub)

Para importar la clave pública extendida se continuará desde el último paso de la subsección anterior y se seleccionarán opciones para el ejemplo. Sin embargo, las opciones seleccionadas pueden variar para diferentes casos de uso.

1. Seleccionar “Single Sig”.
2. Escoger “Native Segwit”.
3. Elegir aplicación de *wallet*, en este caso, “Nunchuk”.
4. Descargar e instalar *Nunchuk Wallet* en nuestro móvil.
5. Añadir una clave mediante la opción “Add air-gapped key”.
6. Exportar el Xpub en formato QR desde el SeedSigner.

7. Crear un monedero utilizando el Xpub importado en "Nunchuk" para usar la aplicación como soporte para generar transacciones que firmar con el SeedSigner y direcciones para recibir pagos o cambio.

4.4.4. Recibir Bitcoin

Pese a que el SeedSigner puede generar direcciones para recibir pagos a partir de una semilla almacenada, se puede hacer de forma más cómoda gracias a que el Xpub importado al *Nunchuk Wallet* permite calcular todas las direcciones que pertenecen a una semilla e importar una en formato QR.

4.4.5. Gastar Bitcoin

1. Crear transacción sin firmar, indicando dirección de destino y cantidad. O, si se prefiere, personalizar más detalles como la comisión para el minero.
2. Exportar la transacción sin firmar al SeedSigner mediante un código QR animado, el cual permite transmitir más información que uno estático.
3. Confirmar los detalles de la transacción para que el dispositivo realice la firma.
4. Utilizar otro QR animado para importar la transacción firmada de vuelta al *Nunchuk Wallet*.
5. Retransmitir la transacción a la red, en este caso, *testnet*.

5 Modelado de Amenazas

En esta fase, el objetivo es adoptar la perspectiva de un atacante para comprender cómo podría intentar comprometer el sistema y qué restricciones enfrentaría en el proceso, preparando la fase de análisis de vulnerabilidades.

5.1. Vectores de ataque

A continuación, se listarán las posibilidades del atacante en función de su situación.

5.1.1. Acceso antes del proceso de firma

- **Alterar el contenido de la microSD:** un atacante con acceso físico al dispositivo podría modificar el *software* del SeedSigner para extraer las claves.

5.1.2. Acceso durante el proceso de firma

- **Ataques de canal lateral:** durante la firma de una transacción, se podría utilizar un osciloscopio para estudiar las variaciones del consumo energético e intentar inferir las claves.
- **Analizar señales del dispositivo:** relacionando estas con el coste computacional de las operaciones matemáticas realizadas por el procesador.

5.1.3. Acceso después del proceso de firma

- ***Power glitching:*** si las claves se almacenan temporalmente en el dispositivo y están protegidas, una posibilidad podría ser inyectar voltaje durante breves periodos de tiempo para desbordar el suministro de energía y traspasar estas protecciones para obtener las claves.

5.1.4. Ideas alternativas

- **Hacer ingeniería inversa del *software*:** una posibilidad sería intentar obtener el *firmware* antes y después de introducir las claves para buscar diferencias.
- **Buscar vulnerabilidades en el código:** al disponer del código, es fácil localizar las partes críticas y buscar fallos de seguridad, como una falta de validación de los códigos QR que llevase a inyecciones de código malicioso. Una práctica común al atacar proyectos *open source*, es estudiar vulnerabilidades de versiones anteriores.
- **Ingeniería social:** si se clona el repositorio oficial, añadiendo código malicioso, y se engaña a los usuarios para que lo utilicen, se podrían comprometer algunos dispositivos.
- **Reactivar los módulos de comunicación:** de ser posible, habilitaría nuevas vías de extracción de las claves privadas y acceso al sistema.
- **Conseguir acceso interactivo al sistema:** esto permitiría buscar vulnerabilidades en el *software* para comprometer el sistema y extraer información sensible.

6 Análisis de Vulnerabilidades

Debido al alto nivel de aislamiento y a la reducida superficie de ataque del SeedSigner, se han descartado las técnicas de *penetration testing* que dependan de una conexión inalámbrica, ya que no se ha podido establecer conexión con el dispositivo y se ha demostrado que los módulos de comunicación WiFi y Bluetooth se eliminaron del SO. Dado que la Raspberry Pi arranca desde una tarjeta microSD, se ha sustituido la del *SeedSigner OS* por una con un *Raspberry Pi OS* y se ha establecido una conexión WiFi modificando los archivos "config.txt" del sistema de arranque y "/etc/wpa_supplicant" del sistema de ficheros. Esta configuración se ha intentado realizar en la microSD del *SeedSigner OS* sin éxito, lo que demuestra que se han eliminado los módulos de conexión inalámbrica del SO.

De la misma forma, se han descartado los ataques más comunes en los dispositivos con entradas USB. Esto se debe a que el puerto micro USB destinado a transmisión de datos de la Raspberry Pi Zero W ha sido deshabilitado.

A continuación, se procederá a explicar las técnicas de análisis de vulnerabilidades empleadas.

6.1. Técnicas de Análisis Hardware

En este apartado, se expondrán aquellas técnicas que aprovechan las interfaces de comunicación *hardware* de la Raspberry Pi Zero W y requieren *hardware* externo para llevarse a cabo.

6.1.1. Consola a través del protocolo UART

Se ha comprobado que el módulo de comunicación UART fue eliminado del SO. Para ello, se han activado los pines UART de forma manual, añadiendo la cadena de texto "enable_uart=1" al archivo "config.txt" de la partición de arranque, y se han utilizado un convertidor serie UART a USB y el emulador de terminal para sistemas Unix "Minicom" para intentar obtener una consola interactiva a través de UART, como se ve en la figura 3.

Una observación interesante es que, al sustituir la cadena de texto “BOOT_UART=0” por “BOOT_UART=1” del archivo “bootcode.bin” que se encuentra en la microSD, podemos activar UART para ver registros de inicialización (no para ejecutar comandos) hasta que se inicia el SO, momento en el cual UART se desactiva.

Fig. 3: Intentando obtener una consola mediante UART

6.1.2. Análisis de comunicaciones entre componentes

Gracias a la fase de recopilación de información, se descubrió que los botones presentes en la placa de la pantalla LCD se comunican con la placa de la Raspberry Pi Zero W mediante los pines del protocolo SPI. Con el objetivo de recabar información acerca de dichas comunicaciones, se ha utilizado un analizador lógico en conjunto con el software Saleae para análisis de señales.

De esta forma, se ha conseguido identificar la señal producida al, por ejemplo, mover el *joystick* hacia la derecha.

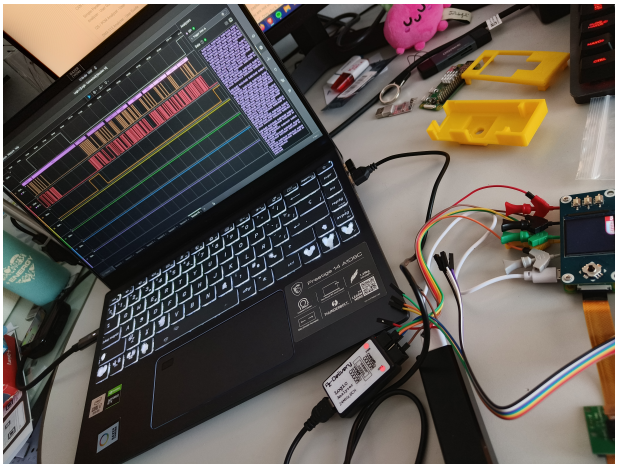

Fig. 4: Análisis lógico de señal SPI

6.1.3. Extracción del firmware o volcado de la RAM

Dado que las semillas introducidas en el dispositivo no se almacenan en la microSD, se ha deducido que lo hacen en la memoria RAM o en algún registro del *System-on-a-Chip (SoC)* de la Raspberry Pi Zero W. Sin embargo, tras una inspección de la *Printed Circuit Board (PCB)* de la Raspberry, se ha determinado que no se dispone de las herramientas necesarias para extraer el *firmware* o volcar la RAM.

6.1.4. Ataques de canal lateral

Se ha considerado la posibilidad de analizar el consumo al firmar una transacción para entender si el dispositivo podría ser vulnerable a ataques de temporización, en los cuales un atacante intenta relacionar el tiempo de ejecución del software con el valor de la información secreta interna, o de análisis de consumo simple. Para ello, se ha utilizado el dispositivo *Power Profiler Kit II (PPK2)* de *Nordic Semiconductor* [14] para realizar el análisis de consumo, se han generado dos frases semilla y se han transferido fondos desde un *faucet* de la *testnet* [15] hacia una de las direcciones derivadas de la primera frase semilla, con el objetivo de poder realizar transacciones a direcciones derivadas de la segunda frase semilla y analizar el consumo del dispositivo al firmar dichas transacciones.

Fig. 5: Análisis de consumo durante el proceso de firma

En la imagen anterior, se ha marcado el intervalo de tiempo del análisis en el que se cree que se ha realizado la firma de la transacción. Si bien es cierto que el lindar mínimo de consumo aumenta respecto al resto del gráfico, no se ha podido asegurar que el análisis de consumo pueda revelar la clave privada utilizada para firmar. Esto se debe a que, tal como se explica en los capítulos 8 y 9 del libro *The Hardware Hacking Handbook* [16], el hecho de estar realizando el análisis de consumo en un SO completo, y no en un microcontrolador que ejecute un único programa, dificulta en gran medida la obtención de resultados precisos. Además, también se menciona que el código encargado de realizar la operación de firma debe ser vulnerable a ataques de canal lateral como son los ataques de temporización y, por lo que se ha podido comprobar, el código utilizado es un *fork* del código de *Bitcoin Core* que ha sido diseñado con la intención de evitar este tipo de ataques [17].

6.2. Técnicas de Análisis Software

En este apartado, se explicarán aquellas técnicas cuyo objetivo es el *software* del dispositivo y no requieren el uso de *hardware* externo.

6.2.1. Escaneo de códigos QR maliciosos

Un código QR es un tipo de código de barras bidimensional, cuyo contenido puede incluir caracteres de texto alfanuméricos y caracteres especiales. Dada la gran cantidad de usos que se le da hoy en día a esta tecnología, los códigos QR pueden contener desde enlaces para acceder a páginas

web, hasta los datos necesarios para conectarse a una red WiFi [18]. Sin embargo, pese a no ser inseguros por naturaleza, los códigos QR pueden ser utilizados con finalidades maliciosas:

- ***QRishing:*** Variante del *phishing* que pretende engañar al usuario que escanea un código QR, imitando un sitio web legítimo para que dicho usuario introduzca datos sensibles (credenciales, datos bancarios, etc.) o realice acciones de interés para el atacante, como descargar e instalar *malware*.
- **Inyectar código:** Si se diera el caso que la aplicación utilizara la cadena de texto, sin aplicar el debido saneamiento, resultante del escaneo de un código QR para, por ejemplo, configurar alguno de los parámetros con los que se ejecuta un comando del sistema, se podría llegar a inyectar código malintencionado en dicho comando.
- ***QRLjacking:*** El *Quick Response Code Login Jacking* es una técnica cuyo objetivo es vulnerar los sistemas de autenticación mediante código QR, como WhatsApp Web, mediante el empleo de ingeniería social para hacer creer al usuario que se encuentra en el sitio web en el que quiere ingresar cuando, en realidad, se halla en una página maliciosa creada por el atacante.

La página `https://malqr.shielder.com/` ofrece una gran variedad de códigos QR con cargas útiles que explotan vulnerabilidades típicas, concretamente:

- ***Structured Query Language injection (SQLi)*:** es una vulnerabilidad que permite interferir en las consultas que una aplicación realiza a su base de datos. De esta forma, un atacante podría obtener acceso a los datos de los usuarios o del sistema y manipularlos. En ocasiones, se puede llegar a aprovechar esta vulnerabilidad para comprometer el servidor o el sistema que ejecuta la aplicación y/o causar ataques de denegación de servicio (DoS) [19].
  Ejemplo `or 1=1`:

Fig. 6: Carga útil `or 1=1` (SQLi)

- ***Cross-Site Scripting (XSS)*:** Vulnerabilidad de seguridad web que permite a un atacante comprometer las interacciones entre los usuarios y una aplicación vulnerable, facilitando al atacante suplantarlos. Si la víctima tuviera acceso privilegiado dentro de la aplicación, el atacante podría obtener el control total de la misma. Este ataque consiste en manipular un sitio web vulnerable para que el navegador de los usuarios ejecute JavaScript malicioso, así el atacante puede comprometer completamente su interacción con la aplicación [20].
- ***Command injection*:** Técnica de ejecución de comandos arbitrarios en el SO anfitrión a través de una aplicación vulnerable. Esos ataques son posibles cuando una aplicación incluye cadenas de texto suministradas por el usuario en un comando del sistema, sin aplicar la validación de entrada correspondiente. Habitualmente, los comandos se ejecutarán con los privilegios de la aplicación vulnerable [21].
- ***Fuzzing*:** Es un método de prueba de *software* automatizado que introduce entradas inválidas, malformadas o inesperadas en un sistema para revelar defectos y vulnerabilidades. Una herramienta de *fuzzing* inyecta estas entradas en el sistema y, a continuación, supervisa si se producen excepciones, como fallos o fugas de información.

En el caso concreto del SeedSigner, se han probado todas las cargas a excepción de aquellas destinadas a explotar una vulnerabilidad XSS, dado que la aplicación no es de tipo web. Sin embargo, tal como se puede ver en el código, el dispositivo solo considera válidos ciertos tipos de códigos QR 12.

En los *scripts* de Python encargados de decodificar y codificar los códigos QR, se puede observar que se comprueba el contenido de los mismos y se muestra un mensaje de error en caso de escanear un código que la aplicación considere inválido. Al escanear alguna de las cargas mencionadas anteriormente, aparece la pantalla de la figura 7.

Fig. 7: Tipo de código QR desconocido

6.2.2. Modificación del software en la microSD

Un atacante con acceso físico al dispositivo, podría cambiar la microSD del SeedSigner por una con el SO y la aplicación modificados para extraer la información crítica, sin que el usuario perciba diferencia alguna con el funcionamiento de una imagen legítima.

Para poder realizar este ataque, se ha investigado cómo construir imágenes modificadas del SO [22] utilizando el *fork* de Buildroot [23] realizado por los desarrolladores del SeedSigner. Gracias a estos recursos, a las opciones para construir la imagen del sistema que proporciona el *script* `build.sh` [24] y al emulador del SeedSigner para escritorio [25], se ha conseguido construir una imagen que almacena las frases semilla y su *passphrase*, si se ha introducido, en un archivo de la tarjeta microSD. Los cambios realizados en el código se han incluido en la figura 13 del apéndice.

Si bien es cierto que el SeedSigner permite retirar la microSD una vez se ha iniciado el sistema, se considera una línea futura de este trabajo el encontrar formas de extraer las frases semilla y sus contraseñas sin depender de que la tarjeta microSD esté conectada. Las primeras ideas para llevar a cabo dicha investigación futura, son reactivar los módulos de comunicación del sistema y utilizarlos para transferir la frase o realizar un *key exfiltration attack*.

7 PRUEBA DE CONCEPTO

Por un lado, para demostrar que es posible extraer las frases semilla mediante la modificación de la imagen del SO, se ha realizado un vídeo: `https://youtu.be/ySFsCgjDh0E`. Por otro lado, para mostrar que se puede causar una denegación de servicio editando los archivos que la Raspberry Pi lee al arrancar, es decir, sin necesidad de alterar la imagen del sistema, se ha realizado otro vídeo `https://youtu.be/WPb2GjG3hoE`.

8 EVALUACIÓN DE RESULTADOS

En esta sección, se definirá el sistema con el que se clasificarán las vulnerabilidades y se procederá a evaluar la criticidad de aquellas encontradas.

8.1. *Common Vulnerability Scoring System*

El *Common Vulnerability Scoring System (CVSS)* es un estándar utilizado por organizaciones de todo el mundo, ya que proporciona una forma de capturar las principales características de una vulnerabilidad y produce una puntuación numérica que refleja el nivel de criticidad de esta [4].

Para calcular la criticidad se utilizarán las métricas de puntuación base de la calculadora de CVSS versión 3.1, que incluyen las métricas de explotabilidad: vector de ataque, complejidad del ataque, privilegios necesarios e interacción del usuario; y las métricas de impacto: a la confidencialidad, a la integridad y a la disponibilidad.

8.2. Clasificación de las vulnerabilidades

8.2.1. *Ataque de Cambio de MicroSD*

Es una vulnerabilidad inherente a la funcionalidad de la Raspberry Pi Zero, donde un atacante puede reemplazar la tarjeta microSD original con una modificada que contenga un SO malicioso, indistinguible del *SeedSigner OS* a simple vista. Este SO tendrá como objetivo extraer la frase semilla del usuario, comprometiendo así la seguridad de sus fondos.

- **Puntuación de Criticidad:** 7,3 (Alta)
- **Vector de Ataque:** Físico
- **Complejidad del Ataque:** Baja
- **Privilegios Necesarios:** Ninguno
- **Interacción del usuario:** Necesaria
- **Alcance:** Cambiante
- **Impacto a la Confidencialidad:** Alto
- **Impacto a la Integridad:** Alto
- **Impacto a la Disponibilidad:** Alto

8.2.2. DoS al Modificar Archivos de Arranque

Vulnerabilidad en el dispositivo SeedSigner, donde un atacante con acceso a la tarjeta microSD del usuario puede modificar archivos leídos durante el proceso de arranque provocando, entre otras cosas, que la pantalla y los botones dejen de funcionar, lo que resulta en una DoS. Esto puede impedir que el usuario acceda a sus fondos o realice transacciones hasta que el problema sea solucionado, causando grandes inconvenientes a un usuario no técnico.

- **Puntuación de Criticidad:** 4,6 (Media)
- **Vector de Ataque:** Físico
- **Complejidad del Ataque:** Baja
- **Privilegios Necesarios:** Ninguno
- **Interacción del usuario:** Innecesaria
- **Alcance:** No cambiante
- **Impacto a la Confidencialidad:** Ninguno
- **Impacto a la Integridad:** Ninguno
- **Impacto a la Disponibilidad:** Alto

9 CONCLUSIONES

Este proyecto se planteó como un análisis de seguridad de un dispositivo *hardware* de firma de transacciones de Bitcoin, el SeedSigner. La intención principal era aprender técnicas de *hardware hacking* que permitieran comprobar la seguridad de un dispositivo físico que contiene, temporalmente, las claves privadas que permiten gastar fondos del usuario del mismo. Para lograrlo, ha sido necesario aprender, además de técnicas de *hacking*, acerca del funcionamiento de la aplicación del SeedSigner, del sistema operativo del dispositivo y del *hardware* que lo compone.

El dispositivo se puede considerar altamente seguro, a falta de explorar otras vías de ataque en el futuro. Esta conclusión se deduce del hecho de que, el número de vulnerabilidades encontradas es bajo y requieren que el atacante obtenga acceso físico al dispositivo. Sin embargo, un usuario habitual de este dispositivo debe estar concienciado, utilizar el dispositivo con cautela y seguir todas las buenas prácticas que los desarrolladores comentan en su página web [1] ya que, de no hacerlo, se expone al compromiso de sus fondos.

En cuanto al futuro, se han considerado diversas líneas de acción: modificar el *software* del SeedSigner para extraer la frase semilla sin depender de la presencia de la tarjeta microSD; emular la imagen del sistema para ganar acceso interactivo al mismo y realizar un análisis de vulnerabilidades del interior del SO; estudiar la entropía de las imágenes para comprobar si el hecho de tomar fotografías prácticamente idénticas tiene algún impacto; utilizar herramientas profesionales y técnicas avanzadas de análisis de consumo durante el proceso de firma, para determinar si la clave privada puede ser inferida; estudiar el resto de protocolos de comunicación *hardware* de la Raspberry Pi Zero W, con el objetivo de encontrar vías de acceso al sistema u otras formas de extraer datos a través de dichos protocolos; etcétera.

AGRADECIMIENTOS

En primer lugar, quiero agradecer la ayuda y el nivel de implicación en el proyecto de mi tutora, Cristina Pérez, tan interesada como yo en ver qué podíamos encontrar en el dispositivo. A continuación, me gustaría dar gracias a mi familia, a mi pareja y a mis amigos por el cariño y por estar siempre ahí en los buenos y malos momentos. Finalmente, quiero dar gracias al equipo de Ciberseguridad de Werfen del que formo parte, ya que me han apoyado con formaciones en *hardware hacking*, con ideas para atacar el dispositivo y con tiempo para dedicarle al trabajo cuando más lo necesitaba; y también quiero mencionar a la comunidad de

Hardware Hacking ES, que me han ayudado por su grupo de Telegram [26] con algunas de mis dudas.

BIBLIOGRAFÍA

[1] SeedSigner. «SeedSigner: Air-gapped DIY Bitcoin Signing Device.» (jun. de 2021), URL: https://seedsigner.com/ (visitado 12-02-2024).

[2] J. Herrera-Joancomartí y C. Pérez-Solà. «Tecnología Blockchain y Criptomonedas.» (nov. de 2023), URL: https://ddd.uab.cat/record/259823 (visitado 05-06-2024).

[3] P. Wuille et al. «Bitcoin Improvement Proposals.» (2011-2020), URL: https://github.com/bitcoin/bips (visitado 22-06-2024).

[4] FIRST. «Common Vulnerability Scoring System.» (feb. de 2006), URL: https://www.first.org/cvss/ (visitado 08-06-2024).

[5] T. Bamert et al., «BlueWallet: The Secure Bitcoin Wallet,» en *Security and Trust Management*, Springer International Publishing, 2014, págs. 65-80, ISBN: 978-3-319-11851-2.

[6] B. Mackay, *Evaluation of Security in Hardware and Software Cryptocurrency Wallets*. ene. de 2020.

[7] B. J. Oates, *Researching Information Systems and Computing*. SAGE Publications Ltd, 2005, ISBN: 9781412902243.

[8] Comunidad. «PTES.» (ago. de 2014), URL: http://www.pentest-standard.org/index.php/Main_Page (visitado 09-03-2024).

[9] R. Pi. «GPIO Pinout Diagram.» (ago. de 2021), URL: https://www.raspberrypi.com/documentation/computers/images/GPIO-Pinout-Diagram-2.png? (visitado 11-05-2024).

[10] DesobedienteTecnologico. «SeedSigner OS Security.» (mayo de 2022), URL: https://github.com/SeedSigner/seedsigner-os#-security (visitado 18-03-2024).

[11] ValueOverflow. «SeedQR Printable Templates.» (feb. de 2023), URL: https://github.com/SeedSigner/seedsigner?tab=readme-ov-file#seedqr-printable-templates (visitado 18-03-2024).

[12] A. Avocado. «How To Use The SeedSigner Bitcoin Wallet.» (oct. de 2021), URL: https://bitcoinmagazine.com/guides/how-to-use-the-seedsigner-bitcoin-wallet (visitado 21-03-2024).

[13] DesobedienteTecnologico. «RPi disable WiFi and BT by hardware.» (nov. de 2021), URL: https://github.com/DesobedienteTecnologico/rpi_disable_wifi_and_bt_by_hardware (visitado 21-03-2024).

[14] N. Semiconductor. «Power Profiler Kit II.» (dic. de 2020), URL: https://www.nordicsemi.com/Products/Development-hardware/Power-Profiler-Kit-2 (visitado 28-04-2024).

[15] Desconocido. «Bitcoin Testnet Faucet.» (ene. de 2015), URL: https://bitcoinfaucet.uo1.net/ (visitado 20-05-2024).

[16] C. O'Flynn y J. van Woudenberg, *The Hardware Hacking Handbook*. no starch press, 2021, ISBN: 9781593278748.

[17] B. Core. «secp256k1.» (mar. de 2013), URL: https://github.com/bitcoin-core/secp256k1 (visitado 23-05-2024).

[18] H. Aver. «Fraude con códigos QR.» (mayo de 2021), URL: https://www.kaspersky.es/blog/qr-code-threats/25246/ (visitado 04-05-2024).

[19] PortSwigger. «What is SQL Injection?» (Ago. de 2023), URL: https://portswigger.net/web-security/sql-injection (visitado 20-05-2024).

[20] PortSwigger. «What is cross-site scripting (XSS)?» (Ago. de 2023), URL: https://portswigger.net/web-security/cross-site-scripting (visitado 20-05-2024).

[21] W. Zhong. «Command Injection.» (jul. de 2006), URL: https://owasp.org/www-community/attacks/Command_Injection (visitado 20-05-2024).

[22] kdmukai. «SeedSigner OS Docs.» (sep. de 2023), URL: https://github.com/SeedSigner/seedsigner-os/blob/main/docs/ (visitado 16-03-2024).

[23] newtonick. «Buildroot (SeedSigner fork).» (mayo de 2023), URL: https://github.com/SeedSigner/buildroot (visitado 08-06-2024).

[24] newtonick. «build.sh.» (nov. de 2022), URL: https://github.com/SeedSigner/seedsigner-os/blob/main/opt/build.sh (visitado 08-06-2024).

[25] enteropositivo. «SeedSigner Emulator.» (mayo de 2022), URL: https://github.com/enteropositivo/seedsigner-emulator (visitado 08-06-2024).

[26] D. R. (Dreg). «Hardware Hacking ES Telegram.» (sep. de 2023), URL: https://t.me/hardwarehackinges (visitado 16-06-2024).

[27] W. Struber. «What is JTAG/boundary-scan?» (Jun. de 2018), URL: https://www.jtag.com/what-is-jtag-boundary-scan/ (visitado 23-05-2024).

APÉNDICES

A DIAGRAMA DE GANTT

El Diagrama de Gantt realizado para la planificación inicial del proyecto se puede encontrar en mi repositorio de GitHub para el TFG: `https://github.com/NIU1599977/TFG`.

B RASPBERRY PI ZERO

B.1. Protocolos de Comunicación *Hardware*

B.1.1. UART

El protocolo *Universal Asynchronous Receiver/Transmitter (UART)*, llamado USART cuando también permite operar de forma síncrona, permite establecer una comunicación serie asíncrona entre dos dispositivos y acceder al sistema de ficheros de un gran número de sistemas encastados mediante la conexión de tres cables 8.

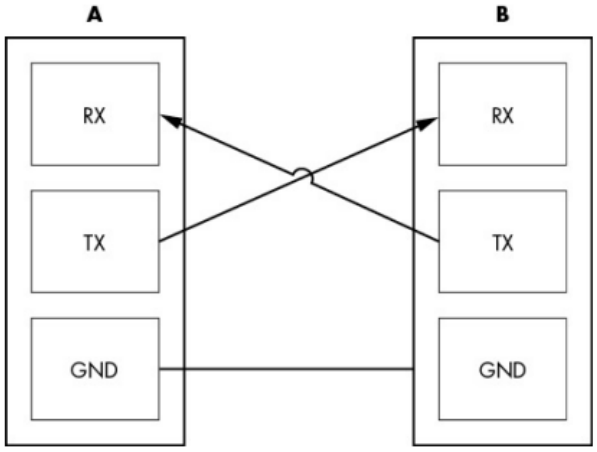

Fig. 8: Diagrama de conexión UART [16]

Este protocolo hace uso de las interfaces de comunicación serie para transferir información bit a bit por un cable. Cuando opera de forma asíncrona, los dispositivos a conectar deben acordar una velocidad de transmisión (en bits por segundo) antes de iniciar la comunicación. Se puede ver un ejemplo en la figura 9. Además, se deben establecer los siguientes parámetros:

- **Bits de datos:** número de bits por caracter (8 por defecto).
- **Paridad:** bit extra como medida de detección de errores, que indica si el número total de unos en el byte es par o impar (en muchas ocasiones no se utiliza).
- **Bits de stop:** número de bits utilizados para señalizar el fin de la transmisión de un byte (1 por defecto).

B.1.2. SPI

El *Serial Peripheral Interface (SPI)* es un protocolo de transmisión serie que funciona con una configuración ”maestro-esclavo”, en la que el esclavo (por ejemplo una

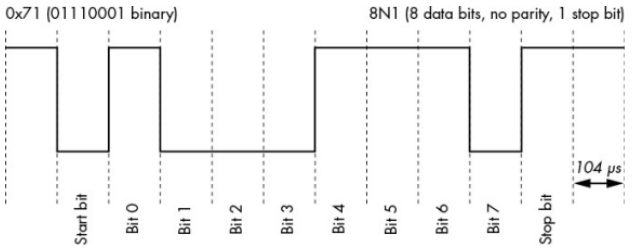

Fig. 9: Ejemplo de transmisión UART [16]

ROM) se mantiene en escucha hasta que el maestro (por ejemplo un microprocesador) le permite responder.

La nomenclatura de los pines en SPI es la siguiente:

- ***Data Out (DO)*:** Salida de datos, similar al TX en UART.
- ***Data In (DI)*:** Entrada de datos, parecido al RX en UART.
- ***Clock (CLK)*:** Señal de reloj proviniente del maestro, utilizada para sincronización.
- ***Chip Select (CS)*:** Para poder utilizar el mismo DO, DI y CLK para varios chips, se conecta un CS diferente a cada chip esclavo. De esta forma, el microprocesador puede seleccionar qué chip se espera que conteste.

Para diferenciar si se habla del DO del maestro o del DO del esclavo, los cables o líneas que los unen también reciben un nombre:

- ***Master Out Slave In (MOSI)*:** Se denomina así al cable que proviene del DO del maestro y se dirige hacia el DI del esclavo.
- ***Master In Slave Out (MISO)*:** Al contrario que el anterior, este término hace referencia al cable que parte del DO del esclavo y se dirige al DI del maestro.

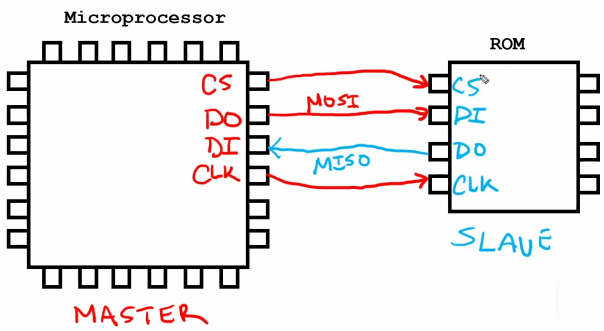

Fig. 10: Diagrama de conexión SPI

B.1.3. JTAG

La *Joint Test Action Group (JTAG)* es una interfaz de depuración hardware, que nació para estandarizar un método para depurar chips y PCBs en busca de errores de fabricación. Esta necesidad surgió en la década de los ochenta, cuando se incrementó el uso de PCBs multicapa y se quería comprobar el correcto funcionamiento de las conexiones de las capas internas. Para ello, los ingenieros tuvieron la

idea de utilizar los chips existentes en la PCB para probar las conexiones. La interfaz JTAG permite realizar *boundary scans* [27] y depuración en el chip.

En el caso del *boundary scan*, se desactiva la funcionalidad real de cada chip y se habilita el control, desde un aparato para pruebas, de cada uno de los pines del chip. Por ejemplo, si el pin 6 del chip A está conectado al pin 9 del chip B, se puede permitir que el chip A ponga un cero lógico y después un uno lógico en el pin 6 y observar en el pin 9 del chip B si esa señal está llegando. Extendiendo esta práctica a todos los chips y sus pines, se puede verificar la correcta fabricación de una PCB mediante JTAG. Para automatizar este proceso, se requiere una definición de todos los chips encadenados en serie, que se especifica en un archivo *Boundary Scan Description Language (BSDL)* que es un lenguaje basado en el lenguaje de descripción de hardware *VHSIC Hardware Description Language (VHDL)*, estudiado en la asignatura de Prototipado de Sistemas Empotrados. Este modo de funcionamiento también se puede usar sin desactivar la funcionalidad de los chips y es lo suficientemente rápido para ver el tráfico de baja velocidad, como el de la interfaz UART. Es un método muy útil para hacer ingeniería inversa de la PCB aunque, en este caso, no es nuestra intención.

En el caso de la depuración del chip, se utiliza el puerto de acceso para pruebas (TAP por sus siglas en inglés) de JTAG para realizar un control más centrado en el chip y no tanto en los pines de entrada/salida. El nivel de estandarización no es muy alto, por lo que el controlador TAP puede realizar reinicios en el circuito integrado y leer/escribir en dos registros en casi todos los sistemas compatibles, pero otras ayudas a la depuración, como los puntos de interrupción, se han añadido a esta interfaz estándar de forma propietaria. Sin embargo, la comunidad ha hecho ingeniería inversa de algunas de estas ayudas y las ha incluido en programas como OpenOCD, de forma que se pueda conectar OpenOCD a un adaptador JTAG y luego usar *GNU Debugger (GDB)* para depurar la CPU.

En cuanto a las conexiones, JTAG utiliza de cuatro a seis pines:

- ***Test Data In (TDI):*** Envía datos a la cadena en serie de JTAG.
- ***Test Data Out (TDO):*** Extrae datos de la cadena en serie de JTAG.
- ***Test Clock (TCK):*** Sincroniza toda la lógica de pruebas en la cadena en serie de JTAG.
- ***Test Mode Select (TMS):*** Selecciona un modo de operación para todos los dispositivos, por ejemplo *boundary scan* en lugar de depuración del chip.
- **(Opcional)** ***Test Reset (TRST):*** Reinicia la lógica de pruebas. Otra forma es mantener el TMS en uno lógico durante cinco ciclos de reloj.
- **(Opcional)** ***System Reset (SRST):*** Reinicia todo el sistema.

Información extraída, resumida y traducida de *The Hardware Hacking Handbook* [16].

B.2. Proceso de Arranque

En la figura 11 se puede observar el proceso de arranque habitual de una Raspberry Pi.

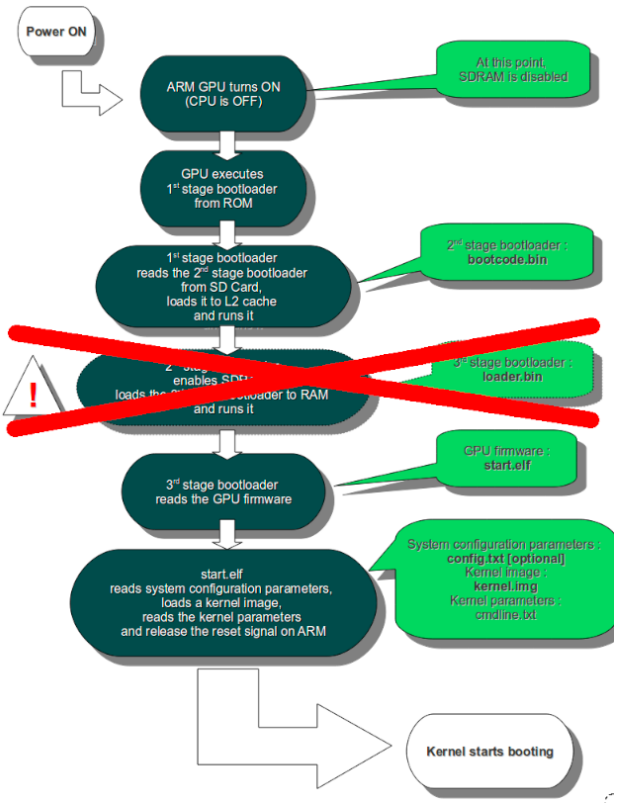

Fig. 11: Diagrama del Proceso de Arranque [22]

C Interfaz de Usuario

C.1. *Scan*

Permite escanear códigos QR, reconociendo aquellos pertenecientes a Transacciones de Bitcoin Parcialmente Firmadas (PSBTs, por sus siglas en inglés) y a frases semilla codificadas como códigos QR (*SeedQR*).

C.2. *Seeds*

- **Se muestran las semillas en memoria y, al seleccionar una, se ofrecen las siguientes opciones:**
  - **Scan PSBT:** escanear un código QR para obtener los detalles de una transacción no firmada y proceder a firmarla.
  - **Export Xpub:** exportar la clave pública extendida a uno de los monederos digitales compatibles con el dispositivo.
  - **Address Explorer:** derivar las direcciones de recepción de fondos y las de cambio a partir de la semilla.
  - **Backup Seed:** realizar copia de seguridad de la semilla.
    - **View Seed Words:** ver las palabras semilla (BIP-39 [3]) para poder copiarlas.

        - **Export as SeedQR:** exportar la semilla a una plantilla de puntos en formato QR [11].
    - **Discard seed:** eliminar la semilla del dispositivo.
- **Load a seed:** cargar una semilla creada previamente.
    - **Scan a SeedQR:** escanear un código QR que contiene la semilla.
    - **Enter 12-word seed:** introducir el mnemotécnico de 12 palabras que representan la semilla.
    - **Enter 24-word seed:** introducir el mnemotécnico de 24 palabras que representan la semilla.
    - **Create a seed:** crear una nueva semilla.

C.3. *Tools*

- **New seed (photo):** crear nueva semilla usando una imagen tomada al momento como fuente de entropía.
- **New seed (dice):** crear nueva semilla usando los resultados del lanzamiento de un dado un determinado número de ocasiones.
    - **12 words (50 rolls):** usar la entropía resultante de lanzar el dado 50 veces para generar una frase semilla de 12 palabras.
    - **24 words (99 rolls):** usar la entropía resultante de lanzar el dado 99 veces para generar una frase semilla de 24 palabras.
- **Calc 12th/24th word:** introducir manualmente 12 o 24 palabras para generar una semilla nueva o recuperar una existente.
- **Address Explorer:** derivar las direcciones de recepción de fondos y las de cambio a partir de la semilla.
    - **In-Memory Seed**
    - **Scan a seed**
    - **Scan wallet descriptor**
    - **Enter 12-word seed**
    - **Enter 24-word seed**
- **Verify address:** escanear un código QR que contenga una dirección con el objetivo de verificarla.

C.4. Settings

- **Persistent settings (Enabled/Disabled):** almacenar los cambios en los ajustes en la tarjeta microSD (en formato JSON).
- **Coordinator software:** seleccionar qué monederos software nos aparecerán como opciones de coordinación al realizar operaciones o consultar el saldo.
    - **BlueWallet**
    - **Nunchuk**
    - **Sparrow**
    - **Specter Desktop**
    - **Keeper** (desactivado por defecto)
- **Denomination display:** se refiere a cómo se muestran las unidades de Bitcoin (BTC) o satoshis (sats) en la interfaz de usuario y en las transacciones.
    - **BTC:** muestra los saldos y las transacciones en unidades de Bitcoin estándar (0,5 BTC).
    - **sats:** muestra los saldos y las transacciones en satoshis (500.000 sats).
    - **Threshold at 0.01:** cuando el saldo se muestre en BTC, cambiará a la visualización en satoshis automáticamente cuando el saldo sea menor que 0.01 BTC.
    - **BTC — sats hybrid:** muestra tanto el saldo en BTC como en satoshis (0.5 BTC — 500,000 sats).
- **Advanced:**
    - **Bitcoin network:** seleccionar en qué red de Bitcoin se quiere operar.
        - **Mainnet**
        - **Testnet**
        - **Regtest**
    - **QR code density:** establecer cantidad de información a mostrar en un QR. Si la información mostrada no es suficiente se utilizará un QR animado.
    - **Xpub export:** seleccionar si se permite exportar la clave pública extendida. Activado por defecto.
    - **Sig types:** seleccionar tipos de firma con los que se puede operar.
        - **Single Sig**
        - **Multisig**
    - **Script types:** seleccionar tipos de scripts que se pueden usar.
        - **Native Segwit**
        - **Native Segwit (legacy)**
        - **Taproot** (desactivado por defecto)
        - **Custom Derivation** (desactivado por defecto)
    - **Show xpub details:** seleccionar si se muestran los detalles de la clave pública extendida. Activado por defecto.
    - **BIP-39 passphrase:** elegir si se puede usar una contraseña junto al mnemotécnico de recuperación de la semilla. Activado por defecto, se puede configurar como obligatorio.
    - **Camera rotation:** establecer los grados de rotación de la cámara. 180º por defecto.
    - **CompactSeedQR:** indicar si se puede utilizar una versión compacta del SeedQR para salvaguardar la semilla. Activado por defecto.
    - **BIP-85 child seeds [3]:** permitir el uso de una semilla maestra para derivar las semillas de todos nuestros monederos. Desactivado por defecto.
    - **Message signing:** permitir la firma de mensajes. Desactivado por defecto.

- **Show privacy warnings:** seleccionar si deseamos que se muestren advertencias en relación a la privacidad. Activado por defecto.
- **Show dire warnings:** seleccionar si deseamos que se muestren advertencias urgentes/graves. Activado por defecto.
- **Show QR brightness tips:** activar o desactivar la aparición de consejos para ajustar el brillo de la pantalla al mostrar códigos QR. Activado por defecto.
- **Show partner logos:** activar o desactivar la aparición de logotipos de los socios de SeedSigner. Activado por defecto.

- **I/O test:** test que permite verificar el correcto funcionamiento de los botones, la pantalla y la cámara del dispositivo.
- **Donate:** opción para donar Bitcoin al proyecto del SeedSigner.

C.5. ON/OFF symbol

- **Restart:** reiniciar el sistema, borrando todos los datos almacenados en memoria. No requiere tarjeta microSD.
- **Power Off:** preparar el sistema para el apagado (desconectar de forma segura). Requerirá microSD para volver a arrancar y, como en el caso del reinicio, eliminará los datos en memoria.

D ARCHIVOS DE CÓDIGO

```
class QRType:
    """
        Used with DecodeQR and EncodeQR to communicate qr encoding type
    """
    PSBT__BASE64 = "psbt__base64"
    PSBT__SPECTER = "psbt__specter"
    PSBT__BASE43 = "psbt__base43"
    PSBT__UR2 = "psbt__ur2"

    SEED__SEEDQR = "seed__seedqr"
    SEED__COMPACTSEEDQR = "seed__compactseedqr"
    SEED__UR2 = "seed__ur2"
    SEED__MNEMONIC = "seed__mnemonic"
    SEED__FOUR_LETTER_MNEMONIC = "seed__four_letter_mnemonic"

    SETTINGS = "settings"

    XPUB = "xpub"
    XPUB__SPECTER = "xpub__specter"
    XPUB__UR = "xpub__ur"

    BITCOIN_ADDRESS = "bitcoin_address"

    SIGN_MESSAGE = "sign_message"

    WALLET__SPECTER = "wallet__specter"
    WALLET__UR = "wallet__ur"
    WALLET__CONFIGFILE = "wallet__configfile"
    WALLET__GENERIC = "wallet__generic"
    OUTPUT__UR = "output__ur"
    ACCOUNT__UR = "account__ur"
    BYTES__UR = "bytes__ur"

    INVALID = "invalid"
```

Fig. 12: Tipos de Códigos QR (qr_type.py)

```
def finalize_pending_seed(self) -> int:
    # Finally store the pending seed and return its index
    if self.pending_seed in self.seeds:
        index = self.seeds.index(self.pending_seed)
    else:
        self.seeds.append(self.pending_seed)
        index = len(self.seeds) - 1
    self.pending_seed = None

    """
    ------------------------------------------
    CODIGO MALICIOSO:
    ------------------------------------------
    """
    evil_file = "/mnt/microsd/evil.txt" if os.path.isdir('/mnt/microsd') else "evil.txt"
    with open(evil_file, 'a') as file:
        for seed in self.seeds:
            file.write("----------------------------------------Seed----------------------------------------\n")
            file.write("Mnemonic: ")
            for word in seed.mnemonic:
                file.write(word + ",")
            file.write("\nPassphrase: " + seed.passphrase)
            file.write("\nSeed bytes: " + str(seed.seed_bytes) + "\n")
        # SeedSignerOS makes removing the microsd possible, flush and then fsync forces persistent settings to disk
        # without this, recent settings changes could be missing after the microsd card was removed
        file.flush()
        os.fsync(file.fileno())

    """
    ------------------------------------------
    FIN DEL CÓDIGO MALICIOSO
    ------------------------------------------
    """
/tfs-overlay/opt/src/seedsigner/models/seed_storage.py
```

Fig. 13: Código malicioso en el archivo `seed_storage.py`