Trezor Crypto Wallet Product Security Report

1.0 PURPOSE

This document summarizes the results of the security-related activities performed as part of the Trezor Crypto Wallet Product Security Assessment project. Successfully completing this project satisfies the final requirement for successfully completing the University of Colorado Boulder's Embedded Cybersecurity graduate course (CYBR5830-002) taught by Mr. Garrett Schumacher during the Spring 2023 session in partnership with Velentium, LLC.

2.0 SCOPE

This assessment covers the Trezor Model One Blockchain Wallet – the "device under test" or "DUT". The DUT is a Trezor Model One. The DUT's identifying information is as follows:

- Manufacturer: Trezor
- Product Name: Trezor Model One
- Model: One
- Serial Number: N/A – Trezor doesn't use serial numbers as a security pre-caution
- Firmware Version: Universal 1.11.2

Other system components intended to interact with the DUT include the following. The interactions between the DUT and other components are under scope of the assessment.

- Desktop / Laptop Computer
- Trezor Suite Web Application
- Blockchain Nodes / Validators
- User PIN/Seed Phrase Management

3.0 ATTACHMENTS

Document Name	Description
Trezor-Data-Flow-Diagram.tm7	Raw threat modeling data flow diagram generated using Microsoft Threat Modeling Tool
Trezor_Vulnerability Assessment.xlsx	Pre-mitigation and post-mitigation CVSS scoring of vulnerabilities decomposed using STRIDE based upon the attached data flow diagram
CycloneDX-Trezor-Crypto-Wallet-SBOM-5-5-2023-11-50.xml	Machine-readable SBOM in CycloneDX format

4.0 TABLE OF CONTENTS

5.0 GOALS

The following security goals should be applied to all considerations when designing and implementing the system.

Non-Functional Goals

- Secure coding conventions should be defined and followed for each of the computer languages utilized in the system.
- Residual vulnerabilities negatively impacting the safety and efficacy of the device/system should be reviewed in the context of ISO 14971:2019+A11:2021 utilizing a benefit-risk assessment (BRA).

Functional Goals

- There should be encryption of critical data, including data in motion, data at rest, firmware updates, PII, etc.
- Encryption should utilize symmetric encryption with a minimum key bit width of 128 bits (256 bits or greater preferred), or asymmetric encryption with a minimum key bit width of 3072 bits (4096 bits preferred).
- Data integrity should be ensured via digital signatures, cryptographic hashes or message authentication codes (MACs, such as HMAC or CMAC).
- Data authorization should be ensured via access control lists or similar structures, and by limiting services and features to the minimum necessary in order to carry out the critical functionality of the system.
- There should be authorization control of devices when joined to a network.
- Forensically Secure Logging should be used to avoid repudiation and ensure integrity by protecting both the contents and order of log messages.
- Interruptions to ‘essential performance’ should, at a minimum, be reported to the user.
- Keys and shared secrets should be stored in a secure manner, refraining from storing them in plain text or in easily accessible memory.
- The device should not have any wireless communication capabilities to ensure data stored is only accessible with physical access to the device.
- There should be an additional pin code needed to use the device once physical access is gained.
- There will be a boardloader to load and check the integrity and signatures of the bootloader to prevent malicious code uploaded to the device from running.
- A seed phrase will be used to generate the private key using the BIP 32 standard.
- There will be an additional optional password that can be appended to this seed phrase to create an additional layer of security for accessing the accounts on the DUT.
- Hardware root of trust functionality should be utilized in favor of software-only solutions, inclusive of key and shared secrets storage and encryption/decryption acceleration.

6.0 PRODUCT OVERVIEW

The DUT and system under assessment are described in more detail below. The Data Flow Diagram depicts the system per Section 2.0 Scope and serves as the frame of reference for the Vulnerability Assessment.

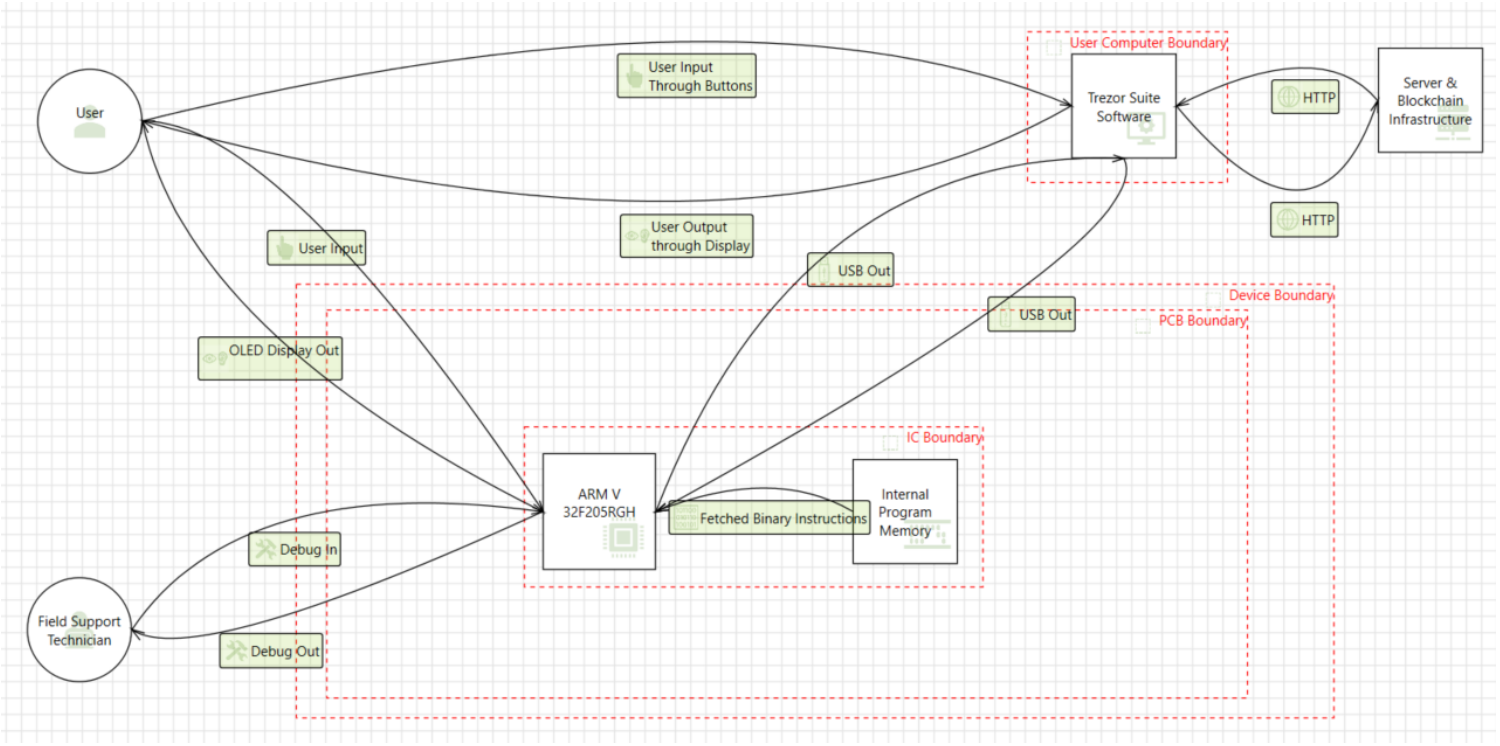
Product Description and Use Case

The Trezor One Wallet is a cryptocurrency wallet used to access and transact digital assets on blockchain-based transaction networks. The system comprises of the physical Trezor device and a micro-USB to USB A cable to connect the DUT to a computer. This computer will contain the digital asset transaction application called the Trezor Suite that connects the device to the backend blockchain infrastructure where the transaction information is processed, stored, and accessed. This blockchain infrastructure consists of several peer-to-peer transaction networks of nodes that operate on distributed consensus mechanisms. The Trezor interoperates with the Bitcoin, Ethereum, Ripple, Cardano, and Litecoin blockchains as well as a variety of other similar protocols. The specific architecture of each of these networks varies in levels of decentralization and the consensus mechanism used to validate transactions, however all of them share certain properties such as a transparent distributed ledger of transactions and distributed read/write privileges to various degrees.

The Trezor system creates an operating condition in which the private keys used for authorizing transactions on these blockchains are not stored on a device directly connected to the internet, thereby decreasing the attack surface of the private keys used for these transactions. This is seen as the most optimal form of security for these transactions due to the potential for large amounts of financial value stored in these networks.

These keys are stored on the Trezor Wallet and only used for transaction authorization when the physical buttons on the Trezor Wallet are pressed to verify these transactions, however the Trezor Suite App must be used to interface with this wallet. The Trezor Model One uses an ARM Cortex-M3 processor at 120 MHz with custom software. Two buttons on the device are used to approve or reject transactions and move between screens during setup phase. This 128x64 pixel OLED screen provides enough space to display the transaction details, confirmation requests, warnings, or even simple images. A user-selected PIN is required to access the physical device, which is input with an interface on the Trezor Suite. This interface obscures the numbers on the image of the keypad on the computer, only displaying the numerical values in a random order on the physical device, so even if a computer's screen were being remotely recorded, the PIN number would not be knowable to an adversary.

Data Flow Diagram


User
User Input Through Buttons
User Computer Boundary
Trezor Suite Software
HTTP
Server & Blockchain Infrastructure
HTTP
User Input
User Output through Display
USB Out
Device Boundary
USB Out
PCB Boundary
OLED Display Out
IC Boundary
ARM V 32F205RGH
Fetched Binary Instructions
Internal Program Memory
Debug In
Field Support Technician
Debug Out

7.0 SECURITY RISK MANAGEMENT PLAN

The following security-related activities will be performed as part of the project.

Product and System Data Flow Diagram

The product will be modeled in the Microsoft Threat Modeling Tool. All processes, data stores, users, and interactions will be diagramed. The resulting threat model data flow diagram will be attached to this report. A screenshot of the diagram will be provided in this report, and a description of the product and its use case will be provided within this report.

Threat Modeling

Threat modeling will be performed to identify possible vulnerabilities that could impact the product during the time of design. Vulnerability assessment is a form of threat modeling, in which a data flow diagram is used to decompose the system and identify potential vulnerabilities using the STRIDE methodology. The resulting STRIDE vulnerabilities are then scored in two phases using the Common Vulnerability Scoring System (CVSS).

The first phase, known as the “Pre-Mitigation Vulnerability Assessment”, does not consider the use of any mitigating controls. Any vulnerabilities with a CVSS score of 4 or greater require mitigation controls to be applied in order to manage the security risk introduced by each vulnerability. The second phase, known as the “Post-Mitigation Vulnerability Assessment”, does consider the mitigating controls that are to be implemented in the system. Any vulnerabilities with a Post-Mitigation CVSS score of 4 or greater will be considered a residual design vulnerability in the fielded product/system.

The vulnerability assessment is attached to this report, and a summary is included in this report.

Security Architecture and Requirements

The Threat Modeling, or Vulnerability Assessment, will inform the necessary requirements needed in the system for the product to be a robust and resilient product per the product’s use case.

Both System and Product-level Requirements will be included in this report.

Security Testing and Research

The device testing plan consists of a combination of both hardware and software analysis. Initially, we plan to gain access to the device internals by removing the plastic shell. It’s then that device hardware data relating to the layout and internal components of our device can be discovered. From there we plan to begin researching potential hardware vulnerabilities focused on related security vulnerabilities associated with the device or individual components it uses. Additionally, this stage is where we will research device firmware and perform static and dynamic code analysis along with any known device vulnerabilities discovered by other security researchers.

Once these initial stages of research have been concluded we can then begin our physical testing. Our goal is to gain access to the device PIN which is a 4-9 digit integer that is the sole method of device authentication preventing access to any locally stored private keys for a user’s cryptocurrency. So first we plan to set up the device with a known PIN that will become our target for future testing. Potential testing will include side-channel attacks, power analysis, power glitching, fuzzing, and any additional potentially effective methods discovered during our research phase. Possible limitations during testing may consist of time limitations for time-intensive testing methods such as fuzzing or limited access to required power analysis tools.

Software Bill of Materials

Third-party software components utilized in the product will be listed in this report in a human-readable table, including metadata such as name, origin/vendor, software type, intended use, latest version, implemented version, support status, security description and impact. A machine-readable SBOM in CycloneDx format as a XML is attached to this report.

Security Risk Management Report

Any residual vulnerabilities or security gaps will be described within the report. A residual vulnerability or gap is defined as:

- Any vulnerability in the Post-Mitigation Vulnerability Assessment with a CVSS score remaining above the threshold of 4.0
- Any identified requirement or recommended control/mitigation that was lacking in the product's implementation, or any identified issue or weakness in the implementation of the product
- Any disclosed vulnerabilities or threats for the product or any of its third-party components

An overall summary will be provided by the project team members with high-level recommendations for developers or users of the product to securely maintain and use the product given the Product Security Assessment results.

8.0 ASSET THREAT MODELING

Based on the asset threat modeling exercise we conducted using the Microsoft Threat Modeling tool, there were several areas of concern that we identified in the system. Given the system's use case and the results of the vulnerability assessment, our biggest security concerns were around the potential for unauthorized access and data breaches.

The system handles sensitive information of customers, so ensuring the security and confidentiality of this data is of utmost importance. One of the biggest concerns we identified was around the use of weak or easily guessable passwords. This is a common security risk and could potentially allow attackers to gain unauthorized access to the system. Another area of concern was around the security of the network infrastructure and the potential for network-based attacks.

Fortunately, the appropriate controls were identified and implemented as mitigation to address these vulnerabilities. For example, multi-factor authentication was implemented to strengthen the system against spoofing attacks, and encryption was put in place to limit the potential impact of sniffing-based attacks. Through such methods, mitigations are implemented and the post-mitigation score of each attack is on the lower end.

While the implemented controls addressed most of the identified vulnerabilities, there may still be residual vulnerabilities in the system. As the threat landscape evolves, new vulnerabilities may emerge that were not previously identified or addressed. Therefore, it is important to continually monitor and assess the security of the system to ensure that it remains secure over time.

9.0 SECURITY REQUIREMENTS

- The product shall restrict the authorization of blockchain transactions associated with the DUT to the individual with physical possession of the device.
- The product shall produce authorization keys in accordance with the BIP 32 standard to ensure account recovery by the user in the case of a lost or damaged physical device.
- The product shall require a user to input their PIN to access the device even with physical access to the device.
- The product shall enable the user to authorize and receive transactions of financial value on several blockchain networks.
- The product shall store authorization keys in air-gapped DUT hardware memory such that internet access to the user's computer will not compromise the accounts of the user.
- The product shall require encryption of all data in transit within the device using secure modern encryption methods.
- The product shall retain accurate data in memory during extended periods without a power source.

10.0 SECURITY TESTING AND RESEARCH

Our testing began by extracting the device from its shell. This step proved easy as we discovered the device had no physical security layer that prevented the removal of the shell. At this point, we compiled a list of all the screen printing that could be read off the board and notes of what interfaces were available including a set of six small through-hole vias, the micro-USB port, and exposed pins coming off the main microcontroller. Using this list of device components, supported by specifications released by the manufacturer, we identified the microcontroller as an Arm Cortex-M3 Microcontroller Unit. Additionally, we discovered that the previously mentioned set of vias was a JTAG debug interface. At this point, we decided that our next step of physical testing would be to analyze the JTAG interface.

Next, we attempted to gain access to the debug interface. This started with attempting to insert standard header pins into each via but we quickly discovered that the header pins were too large to fit within each via. We then considered our options of connecting wires to the board either with or without soldering to get usable leads coming off the board. We decided to go with the solder-free option as we were concerned about the potential quality of soldering with our tools at such a small level. Once the wire was cut to size it was inserted into each of the six vias and then up to four of the wires at a time were connected to the KEYSIGHT InfiniiVision DSOX4024A Oscilloscope. The oscilloscope was used to monitor the power and signals of each output during multiple states of device operation. We ran through an analysis of the device with normal power and no power. This analysis identified the likely purpose of each via in the device. From here we were able to short the device using two of the identified leads which triggered a reboot of the device. Based on findings by Joe Grand, a security researcher who was able to retrieve the PIN on an earlier version of firmware through power glitching, it may have been possible for us to gain access to the memory contents stored on the device. However, during our static analysis of the device firmware, we were unable to identify any substantial security-related changes between our version (1.11.2) and the current version (1.12.2).

Further device testing should focus on more targeted power glitching with the ability to define the power levels and the duration supplied. Additional testing of the device should include fuzz testing through the micro-USB input along with static and dynamic code analysis. We expect future successful attacks on the device to require multiple vulnerabilities stemming from both hardware and firmware implementations.

11.0 SOFTWARE BILL OF MATERIALS (SBOM)

The following table itemizes the third-party software components (TPSC) utilized in the product's software. This table is capable of being included in text documents to be communicated to end users and consumers, such as in Instructions for Use.

The National Institute of Standards and Technology's (NIST) National Vulnerability Database (NVD) is the primary threat source used for assessing the below TPSC. If any disclosed vulnerabilities or threats are found for any TPSC, an impact assessment for the findings will be documented below the table in the Third-Party Threat and Vulnerability Impact Analysis section.

A machine-readable CycloneDX SBOM in XML format is attached to this report that can be made available for end users and developers to monitor threat sources against the SBOM for post-market surveillance.

Human-Readable SBOM

TPSC Name	Vendor	Software Type	Intended use in Product	Version Utilized	Current Version	Support Status	Identified Known Vulnerabilities
Doodle	Doodle INC	Application	Web application for the product	4.5.5	4.5.5	Supported	N/A
Embedded Workbench for ARM	IAR	Application	Development environment for ARM	8.42.2	9.32.1	Supported	N/A
Scintilla	Neil Hodgson	Application	Source code editing	5.1.3	5.3.4	Supported	CVE-2019-16294
IBM ICU	Unicode Consortium	Standard	Software international standardization	1.8.1	15.1.0	Supported	CVE-2016-10578
Apache Thrift	Apache	Framework	Cross-language development	0.9.2	0.18.1	Supported	CVE-2015-3254
LLVM	University of Illinois	Compiler	Static and dynamic compilation of programming languages	13.0.0	16.0.3	Supported	N/A
Boost C++ Libraries	Boost	Library	C++ library for software development	1.0	1.82.0	Supported	CVE-2008-0171
OpenSSL Toolkit	OpenSSL Project	Communication stack	Cryptography and secure communication within the device	3.0.0	3.1.0	Supported	CVE-2023-0464
dlmalloc	Doug Lea	Application	Dynamic memory allocation	1.0	2.8.6	Supported	N/A
FreeRTOS	Amazon Web Services	OS	Operating system for the microcontroller	10.3.1	202212.01	Supported	CVE-2021-43997
BlueNRG 1 Bluetooth Stack/SDK	STMicroelectronics	Communication stack	Bluetooth communication between internal components	3.2.1	3.2.3	Supported	CVE-2019-19192
STM32Cuber MCU Package	STMicroelectronics	Framework	MCU package for the microcontroller on the device	1.15.1	1.17.2	Supported	N/A
EmCrypt Cryptographic Library	SEGGER Microcontroller	Library	Enable cryptography on the device	2.36.0	2.36.0	Supported	N/A

Third-Party Threat and Vulnerability Impact Analysis

CVE-2019-16294: Scintilla in Notepad++ prior to v7.7 allows remote code execution or denial of service. This vulnerability is not applicable to our product, as we do not use Scintilla within Notepad++.

CVE-2016-10578: Unicode before v9.0.0 downloads binary resources over HTTP, which leaves it vulnerable to person in the middle attacks. This vulnerability could affect our product, as it uses a very early version of Unicode (1.8.1) before this vulnerability is fixed.

CVE-2015-3254: Apache Thrift versions before 0.9.3 potentially allow attackers to cause a denial of service through the skip function. To avoid this risk, during implementation our team could disable the skip function.

CVE-2008-0171: Boost.Regex in versions Boost 1.33 and 1.34 allows attackers to cause a denial of service through invalid user expression. This vulnerability is not applicable to our product, as we are using version 1.0.

CVE-2023-0464: Security vulnerability related to certification validation. Attackers can create malicious certificate chains that trigger exponential use of computational resources, leading to denial-of-service attacks. During implementation, policy processing can be enabled, which

should mitigate this vulnerability.

CVE-2021-43997: FreeRTOS versions 10.2.0 through 10.4.5 do not prevent non-kernel code from elevating privileges. This would affect our device, as we use version 10.3.1.

CVE-2019-19192: BLE implementation doesn't properly handle consecutive ATT requests, allowing attackers to cause a deadlock or crash. This vulnerability is not applicable to our product, as it only impacts through version 1.3.1 and our device uses 3.2.1.

Post-Market Surveillance

Once the product has been released to the market, our team will use the SBOM to monitor for any new vulnerabilities associated with the TPSCs above. Whenever there is a new vulnerability, our team will conduct an Impact Analysis to determine whether the vulnerability is applicable to our product. If it is applicable to our system, we will work to mitigate the vulnerability as the situation requires.

12.0 SECURITY RISK MANAGEMENT REPORT

Product Security Overview

Overall, the Trezor One hardware wallet is designed to reduce the attack surface an adversary may use to obtain the sensitive private keys needed to authorize transactions on a blockchain transaction network. The most effective method of doing this is achieved by storing the user's Seed Phrase or Private Keys on an air-gaped computing device. By only storing the Private Keys in the memory of the Trezor One hardware wallet, this means that even if the user's personal computer were completely compromised by an adversary, the adversary would not be able to access these authorization codes over the Internet.

With additional security systems, such as the user-input PIN, and the hidden wallet functionality that adds an additional custom word to the users' seed phrase, this device offers state-of-the-art protection against many of the most prevalent threats in cryptocurrency theft.

However, as with all systems, the Trezor One hardware wallet still has security issues. There have been instances of highly sophisticated hardware hackers able to produce the access PIN necessary to access the account on one of these wallets, however, this requires a great deal of time and technical knowledge.

An additional risk is the potential for loss or theft of the Seed Phrase used to cryptographically generate the accounts associated with this hardware device. The memorization and storage of this phrase remains the key threat to this system, as it does not require physical access to the device and can allow access to all of the funds associated with the user if it is obtained. In addition, if this phrase is lost, the user will not be able to recover their own funds, should anything happen to the device.

Residual Design Vulnerabilities in the Fielded System

As mentioned in the Trezor crypto wallet's threat modeling and vulnerability assessment report, several vulnerabilities were identified during the assessment process, and appropriate controls were implemented to address them. However, it is important to note that while the implemented controls may address most of the identified vulnerabilities, there may still be residual vulnerabilities in the system that were not identified or adequately addressed.

Given the evolving threat landscape, it is essential to continually monitor and assess the security of the system to ensure that it remains secure over time. This can be achieved by conducting regular security assessments, such as penetration testing and vulnerability scanning, to identify any new vulnerabilities that may have emerged.

In addition to the above, it is important for us to also monitor for attacks against softwares mentioned in our SBOM (Software Bill of Materials). An SBOM is a comprehensive list of all software components in a system, including their versions and dependencies. By monitoring for attacks against these software components, we can identify and address any vulnerabilities that may be present in the system.

Furthermore, it is crucial to stay up-to-date with the latest security trends and best practices and implement them where necessary to enhance the security posture of the system. This can

include implementing multi-factor authentication, using secure coding practices, and keeping the system and its components up-to-date with the latest security patches and updates.

Having an incident response plan in place is also essential to quickly and effectively respond to any security incidents that may occur. This plan should outline the steps to be taken in the event of a security incident, including containment, investigation, and recovery.

Overall, while the implemented controls may have addressed most of the identified vulnerabilities in the Trezor crypto wallet, it is crucial to continually assess and monitor the security of the system to ensure that it remains secure in the face of an ever-evolving threat landscape, and that any residual vulnerabilities are identified and addressed in a timely manner.

Residual Implementation Vulnerabilities in the Fielded System

There will always be the potential for implementation vulnerabilities, but as of now, there are no known vulnerabilities for the Trezor Model One. All the software for Trezor is open source, meaning anyone can see it in GitHub. Doing this allows for people to check the code for themselves and helps harden the software, since so many different people are looking over the code and looking for vulnerabilities.

Residual Third-Party Component Vulnerabilities in the Fielded System

There are four third-party component vulnerabilities that could be exploited in the fielded product. CVE-2016-10578, CVE-2015-3254, CVE-2023-0464, and CVE-43997 are all applicable to our product based on the SBOM and the use of the software in the product. Almost all of these vulnerabilities could be avoided by upgrading the software used to the most recent versions. Much of the software listed in our SBOM uses older versions of the products, which leaves our device much more susceptible to attacks via third-party components, since the most recent updates have not been incorporated into the device.

Recommendations for Developers and Manufacturers of the Product

Seed phrases seem to be a key issue in the secure usage of this product, so if there were a better way to cryptographically generate private keys in a way that offered a better user experience, this would add a greater deal of security and accessibility to the overall system.

Currently the method for proving that the device has not been tampered with upon purchase is simply a sticker that shows that the box was not opened upon receipt. We recommend additional proof that the device has not been tampered with when the user first purchases the device. This could involve some shrink wrap on the device or other such physical barriers once manufactured.

It is also not possible to prove that the seed phrase initially generated by the device is not known to the manufacturer. If there were a way for the individual user to add additional cryptographic entropy into the system to prove that better forms of randomness are introduced when these seed phrases are generated, this would be effective in ensuring the end user that the seed phrases are not initially known or cannot be approximated or easily discovered by the manufacturer of the device.

Recommendations for End Users and Consumers of the Product

The end user must keep their PIN and Seed phrase accessible to them. The PIN is necessary to get access to the physical device, however if this is lost, the user can buy another Trezor One Wallet and access their accounts using their initial seed phrase. In case of loss of the device, hardware error, physical device damage, or any other potential issue rendering the device unusable, the Seed Phrase is the only way to recover these accounts should there be an issue

with the device.

13.0 APPROVALS

By approving this report, we, the undersigned, acknowledge that all activities necessary to assess the security posture of the DUT, as outlined in Section 7.0 Assessment Plan and described and summarized in Sections 8.0-13.0, have been successfully completed in order to satisfy the requirements per the final project guidelines provided by the instructor.

Name: Kevin Ecke	Role: Blockchain Consultant	Signature: X	Date 5-6-2023
Name: Shelbi Davenport	Role: Developer	Signature	Date: 5/7/2023
Name: Deekshith Reddy Patil	Role: Developer	Signature: Deekshith	Date: 5/7/2023
Name: Keith Bates	Role	Signature: KB	Date

14.0 APPENDIX A – REFERENCES

Build software better, together. GitHub. (n.d.). Retrieved April 24, 2023, from https://github.com/orgs/trezor/discussions?discussions_q=is%3Aopen%2Blabel%3A%22Model%2BOne%22

EEVblog Electronics Community Forum. EEVblog #1006 - Trezor Bitcoin Hardware Wallet Teardown - Page 1. (n.d.). Retrieved April 24, 2023, from https://www.eevblog.com/forum/blog/eevblog-1006-trezor-bitcoin-hardware-wallet-teardown/msg1255268/#msg1255268

Factory reset: Wiping the trezor model one in the TREZOR suite application. Factory reset: wiping the Trezor Model One in the Trezor Suite application. (n.d.). Retrieved April 24, 2023, from https://trezor.io/learn/a/how-to-wipe-your-trezor-model-one

Trezor Company. (n.d.). *Privacy & Security*. Trezor. Retrieved May 6, 2023, from https://trezor.io/security

TREZOR One Dev Kits. Trezor one Dev Kits. (n.d.). Retrieved April 24, 2023, from https://mcudev.github.io/trezor-dev-kit/index.html

Trezor. (n.d.). *Trezor/Trezor-firmware: Trezor Firmware Monorepo*. GitHub. Retrieved May 6, 2023, from https://github.com/trezor/trezor-firmware