STRIDE/DREAD Analysis

Threat Modelling of Trinity Wallet

M. Masoom Alam, PhD

CYBERSECURITY LAB

Department of Computer Science

COMSATS University Islamabad

Table of Contents

Management Summary

Trinity is a user-friendly IOTA wallet that has been in development for the past few months. It serves as a base wallet for the IOTA cryptocurrency and due to the decentralized nature of the IOTA platform, Trinity stores all user private information for accessing funds on the device (Mobile and desktop).

Considering the data gathered for analysis, the probability of risk has been determined by identifying the assets affected by different possible threats in the Trinity wallet. Each risk has a different impact upon the respective asset. We have used DREAD and STRIDE analysis for identification of threats and their risk rating in the Trinity wallet.

Threat Risk Modelling mainly comprises the following steps:

1. Identifying security objectives
2. Breaking down application features
3. Identifying threats and vulnerabilities

Identifying security objectives of the Trinity wallet mainly involves analyzing:

1. Security of the sensitive information stored on device.
2. Review of the third party libraries used.
3. Quantifying the loss of reputation derived from the application being misused.

Breaking down application features involves decomposing the application into small feature sets and categorizing/prioritizing them. The goal of this step is to identify the security impact, evaluated on the basis of data flow, entry points, components and boundaries. The cross platform nature of the Trinity wallet and its dependencies is analyzed thoroughly as some features are more secure if built natively.

Identifying threats involves formalizing known threats. This should include the risks associated with using the application or a certain feature.

Our threat analysis is done in three stages:

1. **Identify assets in Trinity:** Identify trust boundaries, data flow, entry points, privileged code and document the security profile.
2. **Identify threats:** Identification of threats in the Trinity wallet for Android, Linux, Windows and iOS platforms
3. **Rate threats:** Rating threats using STRIDE and DREAD analysis.

Identification of Assets

Identifying critical assets is crucial for information security. A threat process for the Trinity wallet is modeled as follows:

1. Assets for the Trinity wallet.
2. Overview of the Trinity Architecture.
3. Crumble Trinity wallet.

Assets for the Trinity Wallet

Trinity is a stateful wallet which means it stores client data. The most valuable data that the Trinity wallet stores is the seed, balance information, and transaction metadata. To ensure security, the user's seed is encrypted with two layers of AES encryption.

The valuable assets of user that wallet stores are:

- The seed, a string of 81 random characters using A-Z or number 9. [A-Z or 9].
- Balance information, which describes the amount of IOTA owned by the user.
- Public addresses, which are used to receive payments.
- Transaction metadata.

Overview of the Trinity Wallet Architecture

During this step, the following activities are performed

- What can Trinity do?
- Diagram for an overview of the Trinity architecture.
- Technology stack of the Trinity Wallet.

What can Trinity Wallet do?

Trinity offers several unique services that make it one of the most promising candidates for a user-friendly wallet. The IOTA community is growing swiftly, and there has been an increase in the number of non-technical users. Users are exasperated following snapshots, when they have to reattach addresses until their balance is recovered. To combat this situation, Trinity wallet provides an automatic balance recovery feature. Another nuisance of the current IOTA wallet is that users have to do manual reattachment or promotion if a transaction is pending for a while. The Trinity wallet performs automatic reattachment and promotion under the hood to make the wallet user-friendly.

Trinity wallet provides two factor authentication to combat the risk of theft. This feature creates an additional level of security because solely knowing the victim's password is

not enough to gain access to account. Trinity wallet also allows its users to scan addresses via QR codes for convenience.

Here are some unique features that distinguish Trinity wallet from the current official IOTA wallet:

- **Statefulness:** stores data locally and thus improves efficiency of the wallet.
- **Transaction auto-promotion/reattachment :** to increase the probability of transaction confirmation.
- **Two factor authentication:** achieved with Google Authenticator and Authy using Time-based One time Passwords (TOTP).
- **Fingerprint authentication:** A user can use mobile device fingerprint scan for authentication after first login.
- **Multi seed support:** multiple seeds can be stored and managed.
- **QR scanning:** the Trinity wallet allows users to scan QR codes when entering IOTA addresses.

The Trinity wallet accesses locally stored data at login since it is stateful wallet. The data obtained from full nodes by Trinity is secured by the HTTPS protocol.

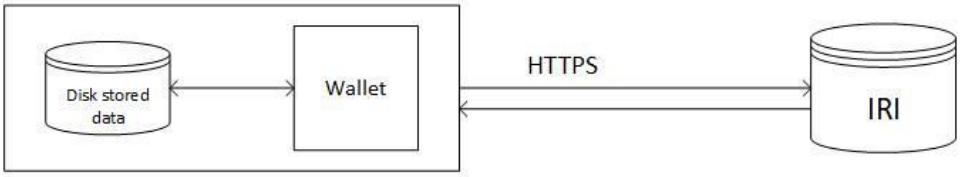

Figure:1 The Trinity Architecture Diagram

Technology/Platform	Implementation Details
Electron	An open source framework for the development of desktop applications.
React Native/ React framework	A javascript library for the development of mobile applications.

Table:1 Technology of the Trinity Wallet

Crumble the Trinity wallet

In the next section we discuss the following aspects of the Trinity wallet:

- The trust boundaries.
- The data flow.
- The entry points.
- The privileged code.
- The security profile.

Boundaries of Trinity Wallet can be Trusted

The Trinity wallet validates the user-provided data at all entry points. Itvalidates user-provided addresses to check whether an address is of the required length: 81 trytes (non-checksum) or 90 trytes (checksum). The IOTA protocol works on trinary logic, so wallet functions check if the user-provided trytes are valid. Valid *trytes* are *[9, A-Z]*. Before creating the transaction, the wallet checks if the user-provided amount is an integer quantity of IOTA.

The other functions for validating data at entry points to Trinity are *isNum*, *isHash*, *isTransferArray*, *isArrayofHashes*, *isArrayofTrytes*, *isArrayofAttachedTrytes*, *isArrayofTxObjects*, *isInputs*, *isString*, *isArray*, *isObject* and *isUri*.

Data Flow

At the login, the IOTA stateless wallet generates addresses from index 0 upto the index of the last unspent address. That's why it takes a long time to login. However, the Trinity wallet is stateful. At login, it utilizes the locally stored data to provide user transaction history, balance information etc. When a user requests to generate a new receiving address, the wallet gets the latest address index from the data stored locally. The wallet then generates addresses and checks for associated transactions, before displaying the latest unspent address.

Entry Points

The entry point for a user is the password only. The user can access the wallet only if they have the valid password. The entry point for a user can also act as the entry point for attacks. An attacker can only access the user's seed if he has the valid password only. However, Trinity wallet hardens the security around the entry point by offering two factor authentication.

Privileged Code

The wallet requires permission to access the camera to scan QR codes.

Security Profile

The following table[1] answers what kinds of questions can be asked while analyzing each aspect of the design and implementation of the application.

Category	Consideration
Input validation	Q. Is all input data of the Trinity wallet validated? A. Yes. iota.lib.js library is used to validate all input data. Q. Could an attacker inject commands or malicious data into the Trinity wallet? A. No, because input data is validated perfectly using iota.lib.js library. Q. Is data validated as it is passed between separate trust boundaries (by the recipient entry point)? A. Full node data validation is outside the scope of this report. Q. Can data in the database be trusted? A. Not Applicable
Authentication	Q. Are credentials secured if they are passed over the network? A. Yes, through HTTPS Q. Are strong account policies used in the Trinity wallet? A. Currently, a 12 character password is required. A strong password should be enforced. Q. Are strong passwords enforced in the Trinity wallet? A. No. The Trinity team confirms they plan to enforce Dropbox's **zxcvbn** library. Q. Does Trinity use certificates? A. Nodes are required to use Transport Layer Security encryption Q. Are password verifiers (using one-way hashes) used for user passwords? A. Yes, the password hash is stored locally.

[1] https://msdn.microsoft.com/en-us/library/ff648644.aspx

Authorization	Q. What gatekeepers are used at the entry points of the Trinity wallet? A. Password is used. Q. How is authorization enforced at the database? A. Not Applicable Q. Does the wallet fail securely and only allow access upon successful confirmation of credentials? A. Yes.
Sensitive data	Q. What sensitive data is handled by the Trinity wallet? A. Seed, public addresses, transaction metadata. Q. How is it secured over the network and in persistent stores? A. TLS, Async storage, Keychain and Keystore Q. What type of encryption is used and how are encryption keys secured? A. Stanford Javascript Crypto Library (SJCL), TweetNaCl
Cryptography	Q. What algorithms and cryptographic techniques are used? A. Stanford Javascript Crypto Library (SJCL), TweetNaCl - AES for seed storage, SHA256 for password hashing. Q. How long are encryption keys and how are they secured? A. 32 bytes Q. Does the Trinity wallet put its own encryption into action? A. No, standard encryption.
Parameter manipulation	Q. Does the Trinity wallet detect tampered parameters? A. Yes. Trinity wallet detects tampered parameters for transactions, bundles, missing checksums. Strict type checking is done within iota.lib.js used by Trinity wallet.

	Q. Does it validate all parameters in form fields, view state and HTTP headers? A. Yes. iota.lib.js adds HTTP headers and API version for IRI automatically.
Exception management	Q. How does the Trinity wallet handle error conditions? A. Through exception handling. Q. Are exceptions ever allowed to propagate back to the client? A. Not Applicable
Auditing and logging	Q. Does Trinity audit activity across all tiers on all servers? A. Not Applicable. Q. How are log files secured? A. Log files are not stored by Trinity.

Table:2 Creating Security Profile

Identification of Threats

At this step , we identify the threats that “can affect” the Trinity wallet and compromise sensitive assets. To direct this identification proof process, we have identified the risks as a probabilistic outcome of threat to the Trinity wallet, and have determined how much loss can be expected from an incident. This leads us to have a better understanding on how to protect the system. Performance of risk assessment has outlined a considerable number of probable threats that can affect the Trinity wallet.

The first thing to realize is that there is no way to eliminate every threat that may affect Trinity. There is no such thing as absolute security. Making a facility absolutely secure would require excessive costs, and it would be so secure that no one would be able to perform the actions they need to. The goal is to manage risks, so that the problems resulting from them will be minimized.

Libraries/Dependencies Assessment

Dependency Management is more of a concern nowadays due to the popularity of frameworks including 3rd party modules and libraries. Security of dependencies is a time-consuming and ongoing task. Most projects make use of these libraries to speed up the development process and end up having large number of dependencies.

A major problem lies in the testing and security of project dependencies. Different vulnerabilities of different severity are found within these libraries and must be patched accordingly. A variety of commercial and open-source tools are available to test the project for finding vulnerable dependencies.

General Approach:

The libraries assessment was performed on the source code from the Gitlab Repository (wallet/tree/develop/src/mobile) develop branch - May 10, 2018. During the assessment, the following tools were used to test the project for vulnerable dependencies.

- **Snyk**

A commercial service that focuses on Javascript npm dependencies. It helps users detect and fix the known vulnerabilities in Javascript projects.

- **RetireJs**

An open-source Javascript dependency checker. It has multiple components including a command line scanner and plugins for multiple browsers.

What follows is a description of the **Table:3 Libraries dependency assessment** ,summarized using the tools mentioned above.

Table Description

- Dependency Used

    - Refers to the dependency used in the project.
- File Path
    - Path to the package.json file consumed by the tools
    - Mobile: **/wallet/src/mobile/package.json**
    - Desktop: **/wallet/src/desktop/package.json**
    - Shared: **/wallet/src/shared/package.json**
- Vulnerable Dependency
    - Dependency used has a vulnerable path to the dependency defined here.
- Info
    - Brief description of the vulnerability
- Risk
    - Severity of the vulnerability generated by the tool itself
- Description Link
    - Link to the detail description of the vulnerability

Dependency Used	File Path (package.json)	Vulnerable Dependency	Info	Risk	Description Link
keytar@4.2.1	Desktop	deep-extend@0.5.2	Prototype Pollution	Low	https://snyk.io/vuln/npm:deep-extend:20180409
tinycolor2@1.4.1	Shared	jquery@1.9.1	3rd party CORS request may execute	Medium	https://github.com/jquery/jquery/issues/2432 http://research.insecurelabs.org/jquery/test/
react-native-level-fs 3.0.1	Mobile	semver 2.3.2	semver_regular-expression-denial-of-service	Medium	http://nodesecurity.io/advisories/31

Table:3 Libraries dependency assessment

Recommendations:

It is important to monitor dependency vulnerability creep. Most of the time outdated dependencies become prone to vulnerabilities and the latest module versions could potentially be free of vulnerabilities. Developers should actively update all of their project modules dependencies once newer versions of them are published. However, this can involve a lot of work and repetition, particularly where there are a large number of dependencies in each application.

There are a number of tools available that help you keep your dependencies up to date.

Tools:

- Greenkeeper
- Snyk
- RetireJs

The identified threats in the Trinity wallet are catalogued in the following section:

- **Threat Description**: Gives the description of the threat within Trinity
- **Threat Target**: Refers to the component/process of Trinity
- **Attack Techniques**: Defines an attack approach
- **Countermeasures**: Actions to mitigate risk within the Trinity wallet
- **Risk**: CVS Scoring is used to calculate the risk

Threat 1:

Threat Description	**Attacker steals seed by monitoring the Android Clipboard**
Threat Target	Seed verification process
Risk	**High**
Attack Techniques	Use of a malicious app to monitor the Android Clipboard
Countermeasures	- Restrict user from copying/pasting the seed - Remove the copy/paste feature - Switch to a custom keyboard application during copy/paste of the seed, to sandbox the seed in that particular keyboard application. - Refrain from downloading apps from unfamiliar sites and only install apps from trusted sources

NB: Threat 1 has been mitigated by using a custom secure share function.

Threat 2:

Threat Description	**Attacker steals funds by manipulating receiving address copied to the clipboard.**

Threat Target	Generating and sending a receiving address process
Risk	**High**
Attack Techniques	Use of a malicious app to monitor and manipulate the Android Clipboard.
Countermeasures	- Restrict user from copying/pasting the receive address - Switch to a custom keyboard application during copy/paste of the receive address, to sandbox the address in that particular keyboard application. - Refrain from downloading apps from unfamiliar sites and only install apps from trusted sources

NB: Threat 2 has been mitigated by adding a warning when address paste is detected.

Threat 3:

Threat Description	**Changing date/time of the system prevents users from logging in to the application. Application gets stuck at the Loading Screen.**
Threat Target	Application Availability
Risk	**Low**
Attack Techniques	- Use AlarmManager Service and its method setTime() - Need **android.permission.SET_TIME** permission
Countermeasures	- Keep your software up to date - Refrain from downloading apps from unfamiliar sites and only install apps from trusted sources - Pay close attention to the permissions requested by apps

Threat 4:

Threat Description	**Attacker steals sensitive information using phishing attack**
Threat Target	Deep Link component
Risk	**Medium**
Attack Techniques	Use of a malicious app for a successful phishing attack
Countermeasures	- Keep your software up to date - Refrain from downloading apps from unfamiliar sites and only install apps from trusted sources - Pay close attention to the permissions requested by apps

NB: Threat 4 has been mitigated by temporarily disabling deep linking.

The Trinity Wallet registers a custom URL schema i.e **android:scheme="iota"** with the device. Whenever a URL or deep link matching the defined schema is called, the smartphone opens up the Trinity Wallet app. There lies the possibility for a phishing attack.

Any malicious app can have the same URL schema for "iota" defined within the AndroidManifest.xml file. So if a device has more than one application to handle the schema, it will show a dialog box to choose an application to perform the specific action defined in the deep link URL. Sensitive information might be disclosed if a user chooses the malicious app to open the specified URL.

Threat 5:

Threat Description	**Desktop-only: Attacker Obtains Seeds by monitoring the Volatile Memory during new seed setup**
Threat Target	Wallet Seed
Risk	**High**
Attack Techniques	Use of spyware to get the seed from memory
Countermeasures	- Save seed in encrypted form - Remove the seed from memory once there is no more use

NB: Threat 5 has been mitigated by clearing memory and using encryption.

Security Analysis per/Platform

(For Android Platform)

Identify Host Threats: Threat 1 can be identified as both a host and application threat, as the application is the reason the information is leaked into memory in the first place.

Issue: Clipboard Vulnerability
Platform: Android
Tested On:
Manufacturer: LG
Model: H818P
Android Version: 7.0
API Level: 24
Summary:

While creating an account for the first time, the user can copy/paste the seed for back-up and verification purposes. In this process, the seed gets copied to the clipboard and can be manipulated/stolen from the clipboard.

Details/Steps:

1) **Stealing a newly generated Seed**

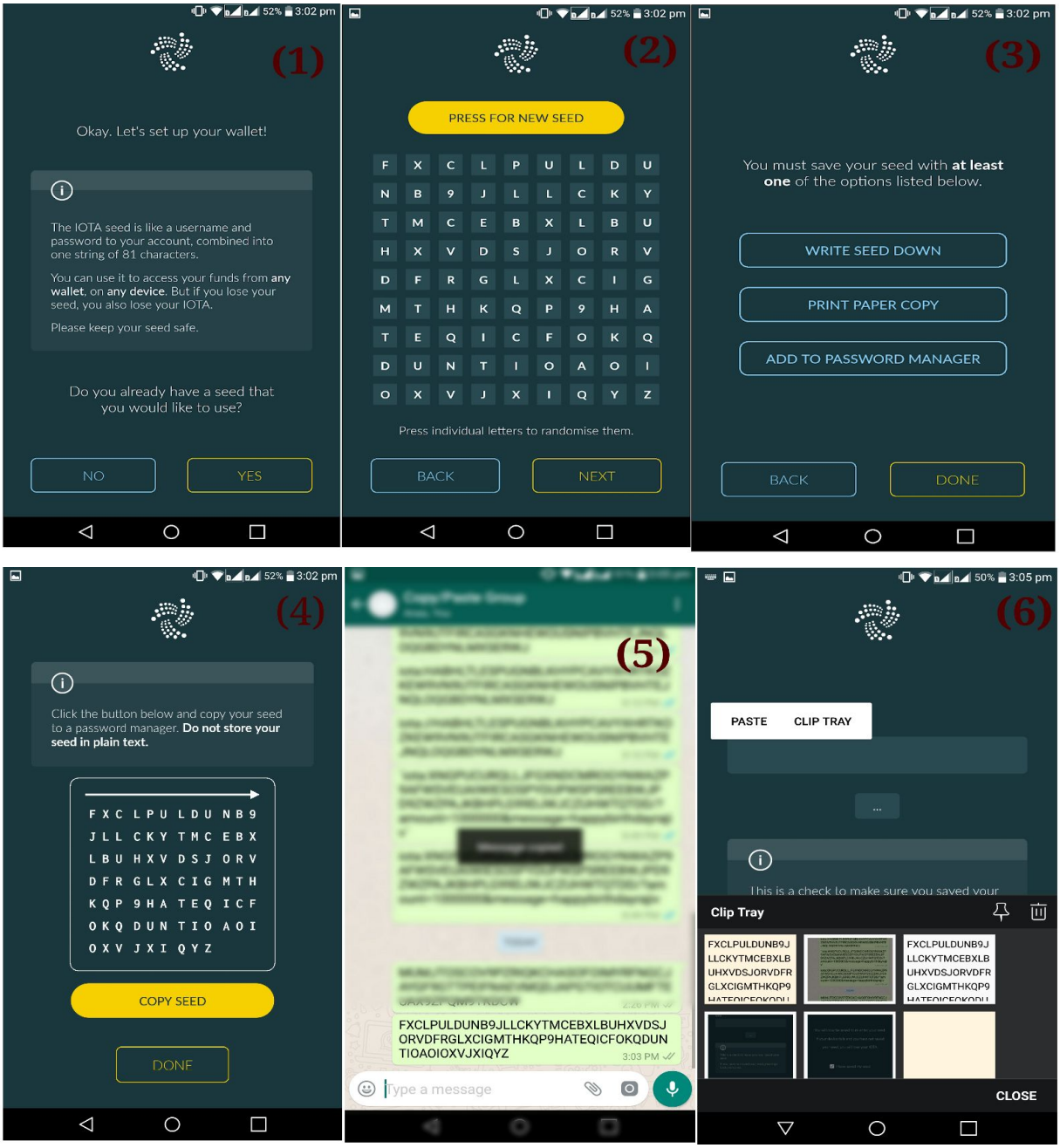

Figure:2 Stealing a newly generated seed

Steps:

1. User selects “NO”, if he/she does not have the seed.
2. User generates a new seed and press “Next”.
3. Application provides three options for saving the seed before seed re-entry verification.

4. User chooses “Add to Password Manager”, and He/She has to share the seed to a secure place by pressing “COPY SEED”.
5. User can save it anywhere. e.g User shares it on whatsapp as shown in image (5).
6. In the reentry process, when the user is asked to reenter the seed to verify it it is a tiresome task to type an 81-character seed. Hence, the user has to copy/paste it from the previously saved place. While copying, the seed gets added to Android Clipboard. Any other application can have a listener attach to the clipboard and the malicious application can manipulate or steal the seed.

2) Stealing funds by manipulating the clipboard

To receive funds from other users, the user has to generate a receive address and send it to the sender. In this process, the user has to copy the receive address which gets added to the clipboard and can be manipulated by the malicious app accordingly.

e.g The malicious application can attach a listener to the clipboard and check for a valid address. If an event occurs at the clipboard, the malicious application can manipulate the copied (desired) address to his receiving address.

When the user pastes the address while sending it to the sender, he might send the malicious address to the sender.

Issue: Date/Time Vulnerability
Platform: Android,Desktop(Linux)
Tested On:

Manufacturer: LG
Model: H818P
Android Version: 7.0
API Level: 24

Summary:

Changing date/time of the system does not let users log in to the application. Application gets stuck at the Loading Screen.

Details/Steps:

1. User opens the application
2. User gives his password and tries to log in.
3. The attacker changes the System Date to any date before 2018. i.e date < 01-01-2018
4. The Application gets stuck at the loading screen
5. User would not be able to log in to the application.

Security Analysis per/Platform.

Threat : Information Leakage
Platform : Linux Mint 64-bit
Tested On :

Manufacturer : Hp
Model : Hp 1000
Kernel Version : 4.4.0-119

Summary :
When setting up the wallet for the first time, the user has to generate a seed. In this process the seed gets copied to the physical memory which can be captured from physical memory image.

Details :

1). Stealing a New Generated Seed

a. Examine
We examined that when the seed is created, it resides in physical memory until the system is restarted, which is a major confidentiality breach.
b. Capture
After setting up the wallet, we generated a new seed, copied that seed to another file and then after a successful login,we took the image of physical memory through **LiME tool**, which allows the acquisition of volatile memory from Linux and Linux-based devices.

c. Analyze
We analyze the image of physical memory through **X-ways Winhex v19.6 tool**. We examined hexadecimal codes by which we found the seed in several places in the physical memory dump. This seed remains in memory even if the wallet is reset.

d. Recommendation
The seed should be stored in encrypted form with hash.

Screenshots: (The seed generated is for test purposes and hence contains zero IOTA.)

Figure 3 shows the settings screen after a successful login. Before login, the seed is not stored in memory, but when we log in successfully the seed is loaded into memory and hence can be captured.

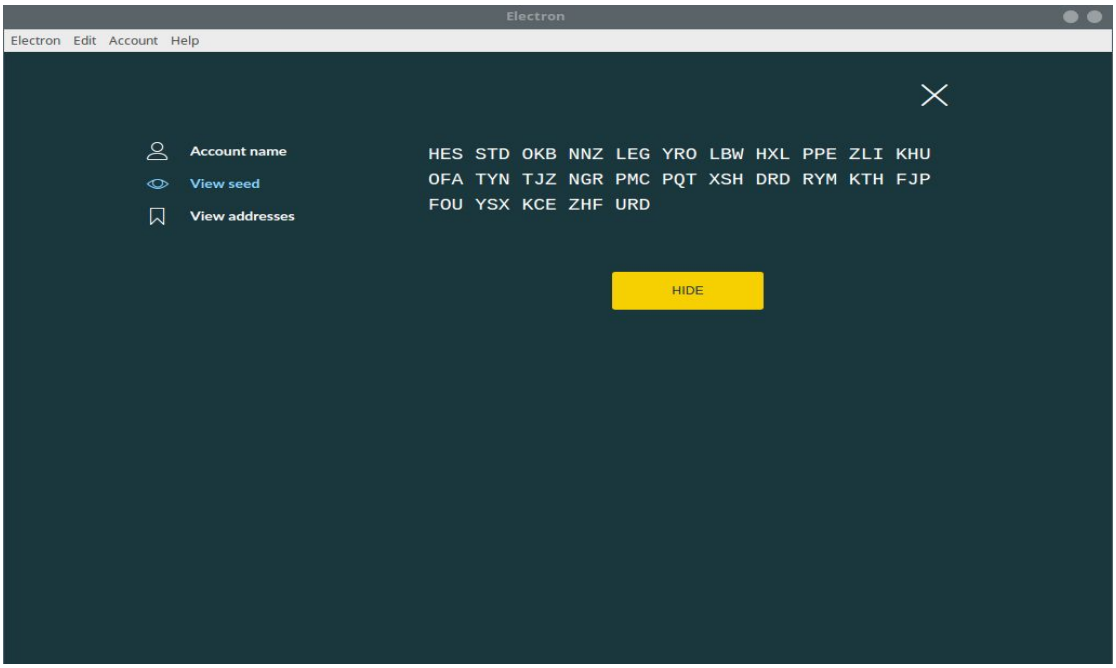

Figure:3 shows the seed generated

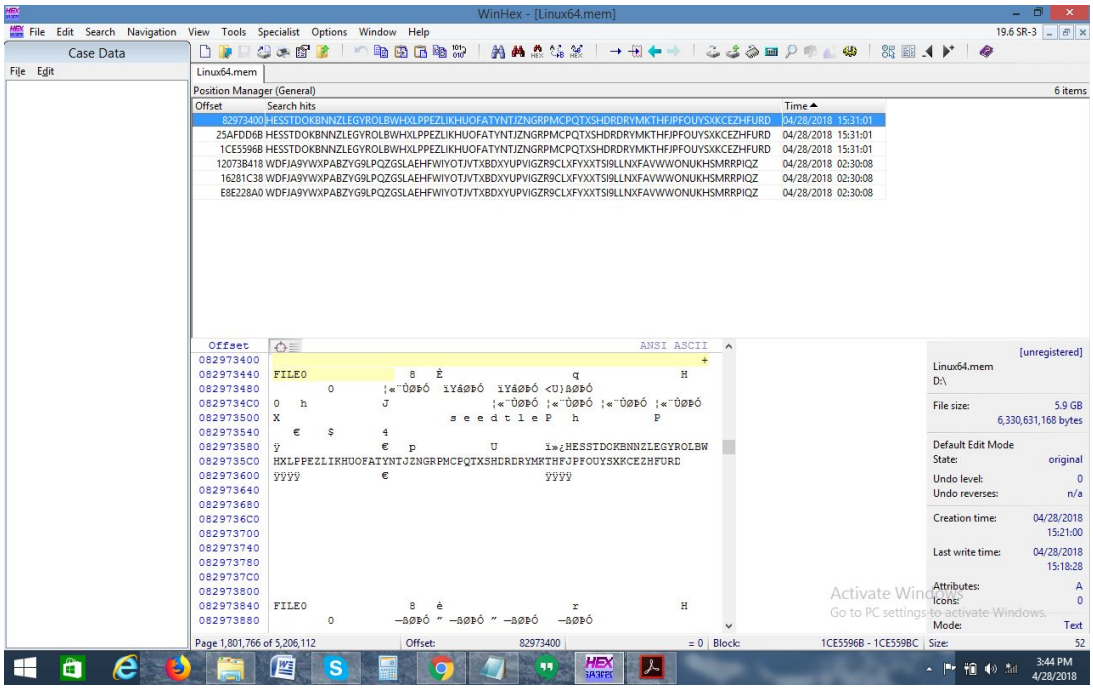

Figure:4 shows the same seed recovered by WinHex.

Threat Ratings

In the last section, we described a rundown of threats that apply to our specific application situation. In the last step of the procedure, we rate threats in light of the risks they cast. This leads us to address the threats that are most problematic, and afterwards resolve alternate threats. Truth be told, it may not be financially reasonable to address the greater part of the distinguished threats, and we may choose to overlook some if the chance of them happening is minimal and the harm that would come about on the off chance that they materialised is negligible.

Use of Threat-Model:

How it fits in with the Security System development life cycle.

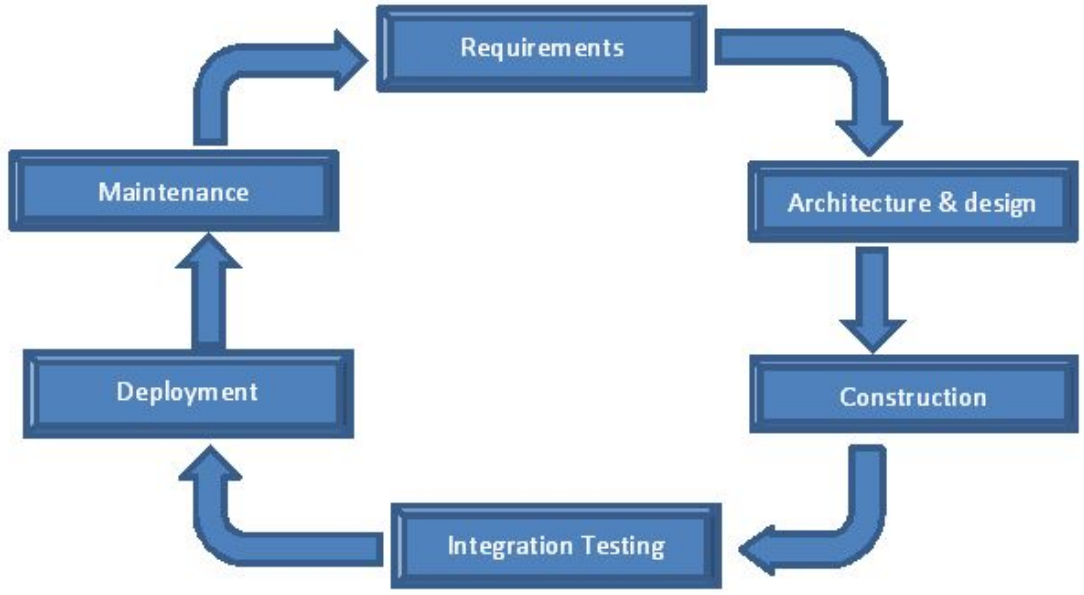

Figure:5 shows the Threat Model life cycle.

Rating Priority (High, Medium & Low)

We can utilize a straightforward High, Medium, or Low scale to organize threats. If a threat is appraised as high, a noteworthy risk is presented by our application and should be sorted out as soon as possible. Medium threats should be tended to, but are less critical. We may choose to overlook low threats contingent on “How much work required” and “Required cost to mitigate the risk”.

Using STRIDE-Model for Classification of Threats

After the vulnerabilities are distinguished, STRIDE procedure acquainted by Microsoft is utilized to characterize these vulnerabilities. Amid security commitment it is important to reinforce your cases (regarding vulnerabilities) with an established standard.

STRIDE stands for (along with the area of attack)

- **Spoofing:** Impact related to Authentication
- **Tampering:** Impact related to Integrity
- **Repudiation:** Impact related to Non-Repudiation
- **Information disclosure:** Impact related to Confidentiality
- **DOS:** Impact related to Availability
- **Elevation of Privilege:** Impact related to Authorization

Threats	Spoofing Identity	Tampering with data	Repudiation	Information Disclosure	Denial Of Service	Elevation of Privilege
Threat 1	✓		✓	✓		✓
Threat 2		✓	✓			
Threat 3					✓	
Threat 4	✓		✓	✓		
Threat 5	✓		✓	✓		✓

Table:4 STRIDE threat classification for analysis

As we go through the analysis from **STRIDE** threat classification (cf. Table:4), **Threat 1** is classified as **Spoofing** of the user's seed to get unauthorized access, **Repudiation** for the victim can deny the unauthorized transaction, **Information Disclosure** for disclosure of user seed information and **Elevation of Privilege** for an attacker can authorize himself by using the stolen user seed.

Threat 2 is classified as **Tampering with Data** for an attacker tampers with the address copied to the clipboard to have a successful attack and **Repudiation** for the victim can deny the unauthorized transaction.

Threat 3 is classified as only **Denial of Service** for changing date/time of the system that prevents the user from logging into the app.

Threat 4 is classified as **Spoofing** of the user's seed to get unauthorized access, **Repudiation** for the victim can deny the unauthorized transaction and **Information Disclosure** for disclosure of user sensitive information, as the sensitivity of information depends upon the phishing attack.

Threat 5 is classified as **Spoofing** of the user's seed to get unauthorized access, **Repudiation** for the victim can deny the unauthorized transaction, **Information Disclosure** for disclosure of the user's seed and **Elevation of Privilege** for an attacker can authorize himself by using the stolen user seed.

Using DREAD-Model for Rating Risk

The issue with an oversimplified rating framework is that colleagues typically won't agree on the chosen ratings. To help alleviate this issue, additional measurements are included to figure out the effect of a security threat. The DREAD model is utilized to figure out the probability of risk, which is abbreviated as Damage Potential, Reproducibility, Exploitability, Affected Users and Discoverability. The threats are rated for a given risk by following the accompanying inquiries[2]:

- Damage potential: How awesome is the harm if the vulnerability is misused?
  - ➢ 1 = Nothing
  - ➢ 2 = Compromised or affected individual user data.
  - ➢ 3 = Compromised or affected every user data.
- Reproducibility: How easy is it to repeat the assault?
  - ➢ 1 = Very hard or impossible, even for administrators of the application.
  - ➢ 2 = One or two steps required, may need to be an authorized user.
  - ➢ 3 = Just a malicious app, No need of authentication.
- Exploitability: How simple is it to dispatch an assault?
  - ➢ 1 = Advanced programming and deep knowledge, with custom or advanced attack tools.
  - ➢ 2 = Malware exists on the Internet, or an exploit is easily performed, using available attack tools.
  - ➢ 3 = Just a malware or native application
- Affected clients: As an unpleasant rate, what number of clients are influenced?
  - ➢ 1 = None
  - ➢ 2 = Some users, but not all
  - ➢ 3 = All users

[2] https://www.owasp.org/index.php/Threat_Risk_Modeling

- Discoverability: How simple is it to discover the vulnerability?
  - ➢ 1 = Very hard to impossible; requires source code or administrative access.
  - ➢ 2 = Can figure it out by guessing or by analyzing the application data flow.
  - ➢ 3 = Details of faults like this are already in the public domain and can be easily discovered using a search engine.

We have the above scale to rate every threat. We can likewise extend the above inquiries to address further issues. For instance, we have included an inquiry regarding Potential harm:

1) Evaluations don't need to utilize a substantial scale since this makes it hard to rate dangers reliably nearby each other. We can utilize a straightforward plan, for example, High (3), Medium (2), and Low (1).
2) Characterizing risk rate in our framework makes it easy to perform risk analysis.

Procedure for DREAD-Model Risk Analysis

After you make the above inquiries, check the qualities (1 to 3) for a given threat. The outcome can fall in the scope of (5 to 15). At that point threat dangers with general evaluations can be given as:

- ➢ (12 to 15) => High Risk.
- ➢ (8 to 11) => Medium Risk.
- ➢ (5 to 7) => Low Risk.

Prioritizing the threat in the following way to analyze the risk factor from first to last:

- First High Risk.
- Second Medium Risk.
- Last Low Risk.

We have considered our five threats:

1. Attacker steals seed by monitoring the Android Clipboard.
2. Attacker steals funds by manipulating receiving address copied to the clipboard.
3. Changing date/time of the system would not let users to login to the application. Application gets stuck at the Loading Screen.
4. Attacker steals sensitive information using phishing attack.
5. Attacker Obtains Seeds by monitoring the Volatile Memory.

Threats	D	R	E	A	D	Total	Rating
Threat 1	2	3	3	2	3	13	High
Threat 2	2	3	3	2	2	12	High
Threat 3	1	1	1	3	1	7	Low
Threat 4	2	2	2	2	3	11	Medium
Threat 5	2	3	2	3	3	13	High

Table:5 shows DREAD rating for all Threats

Conclusion

While we can mitigate the risk of an attack, we can never wholly eliminate risk from any complex application. The truth in the world of security is that we recognize the nearness of threats and we deal with our risks. Threat analysis enables us to examine and impart security throughout our work.

In our threat analysis, we have: identified sensitive system assets, identified ways that an attacker could compromise the system and gain access to the sensitive assets, and prioritised these threats by categorizing them as “High”, “Medium”, or “Low” risk.

Our threat analysis has demonstrated that the IOTA Foundation and the Trinity team have taken considerable measures for hardening the security of the Trinity wallet. Application end users who follow the instructions provided in the setup guide should not encounter any significant risks, provided that they do not download apps from unknown sources. We appreciate the Trinity wallet team for putting great efforts in making Trinity happen and wish them all the best.