QuillAudits

Audit Report
September, 2024

For

Table of Content

Table of Content

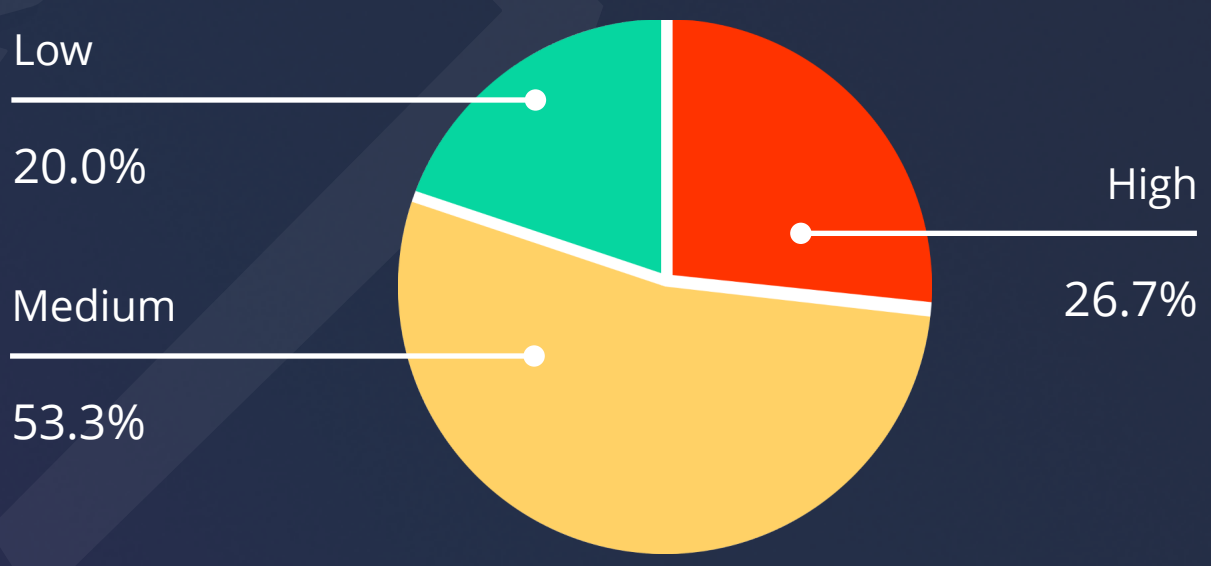
Overview

Overview	Triskel Capital is a global Web3 financial ecosystem where everyone can send, receive, earn money, and do much more, anytime, anywhere, in 173 countries. Triskel Capital's vision is to make Web3, decentralized finance the new normal for billions of people and millions of businesses world-wide. The decentralized world is not a dream, but a reality today.
Scope of Audit	The scope of this pentest was to analyze the Android App and Source code for quality, security, and correctness.
In Scope	com.triskel (sha:f603a61692cbd68b2969ddda8eec2efb48322beb)

Number of Issues per Severity

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	2	0	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	4	6	3	0

Security Issues



Checked Vulnerabilities

We scanned the application for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Improper Authentication
- Improper Resource Usage
- Improper Authorization
- Insecure File Uploads
- Insecure Direct Object References
- Client-Side Validation Issues
- Rate Limit
- Input Validation
- Injection Attacks
- Cross-Site Request Forgery
- Broken Authentication and Session Management
- Insufficient Transport Layer Protection
- Broken Access Controls
- Insecure Cryptographic Storage
- Insufficient Cryptography
- Insufficient Session Expiration
- Information Leakage
- Third-Party Components
- Malware
- Denial of Service (DoS) Attacks
- Cross-Site Scripting (XSS)
- Security Misconfiguration
- Unvalidated Redirects and Forwards

And more...

Techniques and Methods

Throughout the pentest of applications, care was taken to ensure:

- Information gathering – Using OSINT tools information concerning the web architecture, information leakage, web service integration, and gathering other associated information related to web server & web services.
- Using Automated tools approach for Pentest like Nessus, Acunetix etc.
- Platform testing and configuration
- Error handling and data validation testing
- Encryption-related protection testing
- Client-side and business logic testing

Tools and Platforms used for Pentest:

- Burp Suite
- DNSenum
- Dirbuster
- SQLMap
- Acunetix
- Neucli
- Nabbu
- Turbo Intruder
- Nmap
- Metasploit
- Horusec
- Postman
- Netcat
- Nessus and many more.

Issue Categories

Every issue in this report has been assigned with a severity level. There are four levels of severity, and each of them has been explained below.

High Severity Issues

A high severity issue or vulnerability means that your web app can be exploited. Issues on this level are critical to the web app's performance or functionality, and we recommend these issues be fixed before moving to a live environment.

Medium Severity Issues

The issues marked as medium severity usually arise because of errors and deficiencies in the web app code. Issues on this level could potentially bring problems, and they should still be fixed.

Low Severity Issues

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are four issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

High Severity Issues

1. Private Keys Leaked

Description

During the penetration testing conducted on the repositories, it was identified that multiple secret keys were exposed in the source code and constant.js files. This is a critical security concern as it could potentially lead to unauthorized access and compromise of the Freshchat account associated with these keys.

Vulnerable Endpoint

https://github.com/antiergit/triskel_wallet_mobile/blob/5d49806f4b006ddc34dd774924c667e165e86d11/src/Constants.js

Keys Leaked

Web3Auth Key , Secret Widget Key, PVT_KEY, ALCHEMY_SECRET_KEY_MAINNET, CARD_KEY, CARD_PVT_KEY

Recommendation

Immediate Key Rotation:
Initiate an immediate rotation of the exposed API keys. This will invalidate the leaked keys and prevent any potential misuse

Implement Environment Variables:
Refactor the code to use environment variables for storing sensitive information. This ensures that critical credentials are not hard-coded within the source code.

Impact

The exposure of Private keys poses a severe risk to the confidentiality and integrity of the associated with the application. An attacker with access to these keys could potentially impersonate the application, eavesdrop on portfolio, or perform malicious actions on behalf of the compromised account. This could lead to a loss of trust among users and damage to the reputation of the affected organization.

Status

Resolved

2. Encryption Keys Leaked in Shared_pref

Description

During the penetration testing we were able to spot the __androidx_security_crypto_encrypted_prefs_key_keyset__ Keys in shared_Pref folder for com.triskel

Recommendation

Such keys should not be stored on the client side for encryption rather it should be embedded into code and taken on runtime for the app.

POC

<string name="__androidx_security_crypto_encrypted_prefs_key_keyset__"
>12a9012facc855ef25c2f24c0d269aaa3f3cd1cf3fc7ad50eec87a6c1cf0f2b32c3de901f981773
abdb318db7c5e4bc47caf31a6ed551f261e4cfa0da87bc2d93ad897c23a511e21a9a1c99e41c
7ee2f79e4d1808bc0052200914288e3924ccc49b5c88eb35b32e4e5a0bd6c727ddae3722d36
32060f0eceb481e744fcfc9780941c23c176945e1e46a713ca68b5acd7114a56d31425384dad3
0a4f58f517c7d82eb8217fd9601795b13a0c61a4408dae0a2f404123c0a30747970652e676f6f
676c65617069732e636f6d2f676f6f676c652e63727970746f2e74696e6b2e416573536976 4b6
579100118dae0a2f4042001</string>
<string name="__androidx_security_crypto_encrypted_prefs_value_keyset__"
>1288014ffb04ce2596be68a2055b6f65c18abe0dcbfdece67ff70dac3391b8e6d3d94c05a094
e9cee2ec2cdcf2c1545a31e55173b622911a34def425caac1c87a88ce58c02592a9cb20f80057
da04786fb30853c11f7bfc6ea5ec151f9e41cb7931312d4d048e98072b4429a4f25911ce8b890
ab1ab309c3956cb855f0d8190b3516989c5a305eb64fb7271a44089cd0fe8304123c0a307479
70652e676f6f676c65617069732e636f6d2f676f6f676c652e63727970746f2e74696e6b2e4165
7347636d4b65791001189cd0fe83042001</string>

Impact

The exposure of Private keys poses a severe risk to the confidentiality and integrity of the associated with the application. This could lead to a loss of trust among users and damage to the reputation of the affected organization.

Status

Resolved

3. Pin Bruteforce

Description

During the penetration testing, we were able to brute force the PIN during the login as it does not have a rate limit to block a user from using n times the pin which can result in account access without permission as the limit for PIN is not custom and only 6 digits so it can be brute-forced by a simple script injection or manually brute forcing it with N number of attempts.

Steps to Reproduce

1. Create an account
2. Set A pin (eg:123456)
3. It show’s as weak but allows us using this pin and then close the app
4. Try to open the app again and start bruteforcing the code.

Recommendation

Set Attempts Limit:
Set a fix 3 or 5 attempts to be made in PIN login and lock the account for a few minutes after a consecutive incorrect PIN to stop Brute-forcing attacks.

Impact

Account Takeover as it has Improper access due to lack of rate limit.

Status

Resolved

4. OTP Bypass using Response Manipulation

Description

During the penetration testing, during registering the card we had to register using OTP for the Mobile Number to avail the physical card but due to response manipulation, we were able to insert a dummy OTP and mark it as valid in the application.

Vulnerable Endpoint

https://api.triskel-cap.com/api/v1/card/verify_otp

Keys Leaked

- Go to Get a Physical Card Feature.
  Fill out the details
- Add a random Phone Number and click on get OTP
- In Burpsuite, Modify the response from 400 Bad Request and body of {“status”:false,” code”:400,”message”:”Invalid code!”} to status code 200 Ok and body to {“status”:true,” code”:200,”message”:”Success”}
- It will verify a dummy OTP won’t verify it on the backend in the later steps.

Recommendation

Set a response verification so that in critical steps like these it would check if the OTP was provided or not and if was valid or manipulated for bypassing the check.

Impact

Bypass a critical Step in Physical Card Verification leading to not-so-secure verification of the person’s credentials.

Status

Resolved

Medium Severity Issues

5. Insecure key store password

Description

The Java KeyStore (JKS) is a repository of security certificates, including private keys, that are used in Java applications, including Android apps, to establish secure connections and sign APKs. The security of these keys is paramount, as they are used to prove the authenticity of the application and to encrypt sensitive data. In this particular case, the passwords used for both the KeyStore and the key itself are weak (triskel123). The key alias is also simplistic (triskel). These passwords are composed of a common word (triskel) followed by a simple numeric sequence (123). This pattern is highly predictable and could be easily guessed by attackers through brute-force attacks or dictionary attacks.

Vulnerable Endpoint

https://github.com/antiergit/triskel_wallet_mobile/blob/5d49806f4b006ddc34dd774924c667e165e86d11/credentialsReadMe

key store password/key password - triskel123
key alias : triskel

Recommendation

To mitigate the risks associated with weak KeyStore and key passwords, the following actions are recommended:

Use Strong, Unique Passwords: Replace triskel123 with a strong, unique password for both the KeyStore and key. A strong password should:

- Be at least 12-16 characters long.
- Include a mix of upper and lower case letters, numbers, and special characters.
- Avoid easily guessable patterns, such as common words, sequences, or personal information.

Enhance Key Management Practices: Consider using hardware security modules (HSMs) or cloud-based key management services to store and manage cryptographic keys securely, providing additional protection against unauthorized access.

Impact

Using weak passwords for the KeyStore and key can have severe implications:

Compromised Integrity: If an attacker gains access to the KeyStore by guessing the password, they could sign malicious APKs with the legitimate certificate. This would allow them to distribute malicious versions of the application that appear to be from a trusted source.
Loss of Confidentiality: The private keys stored in the KeyStore are used to encrypt sensitive data. If an attacker gains access to these keys, they could decrypt and access this sensitive information, leading to data breaches.
Reputation Damage: If users download compromised applications signed with a stolen certificate, it could lead to a loss of trust in the organization. This could result in a significant reputational and financial loss.
Legal and Compliance Issues: Depending on the nature of the data and the jurisdictions involved, the compromise of private keys and sensitive information could lead to non-compliance with data protection regulations, resulting in legal penalties.

Status

Acknowledged

6. Open Redirection - Dapp

Description

An open redirection vulnerability has been identified in the decentralized application (dApp) due to inadequate input validation in the search field. Specifically, the search function is designed to redirect users to a Google search results page based on their input. However, the validation mechanism relies on a weak regular expression, which can be easily bypassed.

When a user inputs a search query, the application redirects the user to a Google search URL, showing the relevant search results. However, by appending an “@“ symbol followed by a domain (e.g., @evil.com) to the search query, an attacker can manipulate the redirection behavior. Instead of being redirected to Google, the user is redirected to a malicious website controlled by the attacker (e.g., evil.com).

Steps to Reproduce

1. Open the App and click on Dapp
2. Click on search and search for ABC and it show google response of ABC
3. Search for @evil.com and it will directly show you evil.com and not go to google.com

Recommendation

Implement robust regular expression checks that strictly validate user input in the search field. Sanitize and encode all user input to prevent manipulation of redirection URLs. This includes removing or escaping characters that could be used to bypass validation.Instead of allowing arbitrary redirects, consider implementing an allowlist of safe, predefined URLs to which users can be redirected. This limits the possibility of redirection to malicious sites

Impact

If users are redirected to malicious sites from the dApp, it could damage the reputation of the application and its developers, leading to a loss of user trust and engagement. Attackers can craft URLs that appear to be legitimate but, when clicked, redirect users to harmful sites. This can be used in social engineering campaigns to deceive users into performing actions that compromise their security.

Status

Resolved

7. Pin history check missing

Description

A vulnerability has been identified in the application related to the absence of a PIN history check mechanism. Specifically, users are allowed to reuse previous PINs when setting a new PIN. This lack of enforcement allows users to reset their PIN to one of the last three previously used PINs, significantly weakening the security of the application.
In secure applications, especially those handling sensitive financial information like banking applications, it is crucial to prevent users from reusing recent PINs. This ensures that even if an attacker obtains a previous PIN, they cannot simply use it again if they gain access to the PIN reset functionality.
The absence of a PIN history check means that a user can continuously cycle through the same set of PINs. This reduces the effectiveness of PIN expiration policies and increases the risk of unauthorized access.

Recommendation

Implement PIN History Enforcement: Develop and enforce a PIN history policy that prevents users from reusing the last three (or more) PINs. This should be implemented at the server level to ensure consistency across all devices.

Secure Storage of PIN History : Store the hash of the previous PINs securely using strong cryptographic methods. Ensure that this data is protected and cannot be tampered with or accessed by unauthorized entities.

Impact

Increased Risk of Unauthorized Access: If an attacker manages to obtain a user's previous PIN, they could potentially reuse it to gain unauthorized access to the account, as there is no restriction on reusing old PINs.

Weakening of Security Measures: Security policies requiring regular PIN changes are undermined if users can revert to a previously used PIN. This negates the security benefits intended by periodic PIN updates.

Compliance Violations: Many regulatory standards for financial and sensitive data applications require enforcing PIN history checks. Failure to implement this can lead to non-compliance, resulting in fines or other legal penalties.

Status

Resolved

8. Fingerprint Bypass using Frida

Description

The vulnerability has been identified in the Triskel application, allowing unauthorized users to bypass the fingerprint authentication feature. This flaw poses a significant risk to the security and integrity of user data within the application. Immediate action is recommended to address and remediate this vulnerability.

Steps to Reproduce

1. Connect the device using adb and run frida.
2. Frida –U —codeshare ax/universal-android-biometric-bypass –f com.triskel

Recommendation

To implement secure biometric authentication, developers must use the Keychain/Keystore to access objects only when a valid biometric is used.

Impact

Login with biometrics can be bypassed due to this. The severity is reduced as it requires Android device to be rooted in order to be exploited.

Status

Acknowledged

9. Email Address leaked in Response

Description

In some API endpoints it leaks the user email address directly in response as a crypto investing application it should directly leak email address in multiple api endpoint and only in Profile section rather it should use encrypted one or just a user ID for all other API calls

Vulnerable Endpoint

https://api.triskel-cap.com/api/v1/wallet/portfolio

https://otc.triskel-cap.com/api/v1/otc/portfolio

https://api.triskel-cap.com/api/v1/wallet/getWatchlist

https://api.triskel-cap.com/api/v1/card/view_status

Status

Resolved

10. Weak User ID and Wallet ID

Description

User ID and Wallet ID should not be 4 or 5 character ID which can be enumerated by brute-forcing it should be a UUID of sorts that can help keep user information secure from brute-forcing PII data.

Vulnerable Endpoint

https://api.triskel-cap.com/api/v1/wallet/portfolio

https://otc.triskel-cap.com/api/v1/otc/portfolio

https://api.triskel-cap.com/api/v1/wallet/getWatchlist

https://api.triskel-cap.com/api/v1/card/view_status

Status

Resolved

11. Able to access the portfolio without app pin or biometrics

Description

When the app is just opened up and the PIN isn't provided it requests the backend to fetch the portfolio and details like user id and holdings and such details even before logging into the app. The flow should be that the User Provides the PIN and then the app requests API for the portfolio endpoint.

Steps to Reproduce

Open the app and see the requests before even providing the PIN.

Recommendation

Rectify the flow of the Application to avoid such issues.

Impact

Details Disclosed even before loggin into the application.

Status

Resolved

12. User can directly see for the Virtual CARD

Description

When applying for a physical card user can change the card_id parameter from 2 to 4 or 5 to get a card for null price in the case of card_id 5 which we can't obtain directly from the application.

Steps to Reproduce

Change the parameter in the request from

https://api.triskel-cap.com/api/v1/card/cardFeesData?cardId=2 to https://api.triskel-cap.com/api/v1/card/cardFeesData?cardId=5

Recommendation

Improve the card_id value to a complex value that can be hard to enumerate to avoid such issues where users can see internal details that are only available to higher privilege users.

Status

Resolved

Low Severity Issues

13. Multiple Deprecated Libraries in package-lock.json

Description

Package-lock and yarn.lock Stores files that can be useful for the dependency of the application. This is used for locking the dependency with the installed version. It will install the exact latest version of that package in your application and save it in package. This arises a problem if the dependency used has an exploit in the version mentioned. It can create a backdoor for an attacker.

Vulnerable Dependencies

ejs
loader-utils
crypto-js
immer
lodash
decode-uri-component
nth-check
minimatch
webpack-dev-middleware
braces
ws
semver
browserify-sign
axios
shell-quote

Recommendation

1) Update all the above mentioned Dependencies
2) Remove any Library Not needed.

Impact
Multiple of these libraries have public exploits and CVE-registered issues that have been patched and can help your application stay more secure from any dependency-vulnerable issues.

Status
Resolved

14. Cleartext Transmission of Sensitive Information

Description
http.createServer uses HTTP which is an insecure protocol and should not be used in code due to cleartext transmission of information. Data in cleartext in a communication channel can be sniffed by unauthorized actors. Consider using the https module instead..

Vulnerable URL
https://github.com/antiergit/triskel_backend/blob/8cf5d55489bfe7426ff53713ada9cdb72cd73e7d/src/index.ts

Recommendation
Use httpsServer instead of httpServer

Status
Resolved

15. OLD pin can be same as NEW pin

Description

Due to the lack of PIN storage and validation mechanism, user can set the new pin as the old pin. In Financial application this shouldn’t be the case and should not allow user to use the same pin multiple time as it can cause compliance issues and lack of security.

Steps to Reproduce

1. Click on Reset PIN under Security Tab
2. Confirm the old secret and new secret code as the same.
3. It allows user to set the same pin multiple times.

Description

Don’t let user reuse pin’s multiple times.

Status

Resolved

Closing Summary

In this report, we have considered the security of the Triskel Mobile wallet app. We performed our audit according to the procedure described above.

Some issues of High, medium, low, and Informational severity were found, Some suggestions and best practices are also provided in order to improve the code quality and security posture.

Disclaimer

QuillAudits Dapp security audit provides services to help identify and mitigate potential security risks in Triskel. However, it is important to understand that no security audit can guarantee complete protection against all possible security threats. QuillAudits audit reports are based on the information provided to us at the time of the audit, and we cannot guarantee the accuracy or completeness of this information. Additionally, the security landscape is constantly evolving, and new security threats may emerge after the audit has been completed.

Therefore, it is recommended that multiple audits and bug bounty programs be conducted to ensure the ongoing security of Triskel. One audit is not enough to guarantee complete protection against all possible security threats. It is important to implement proper risk management strategies and stay vigilant in monitoring your smart contracts for potential security risks.

QuillAudits cannot be held liable for any security breaches or losses that may occur subsequent to and despite using our audit services. It is the responsibility of the Triskel to implement the recommendations provided in our audit reports and to take appropriate steps to mitigate potential security risks.

About QuillAudits

QuillAudits is a leading name in Web3 security, offering top-notch solutions to safeguard projects across DeFi, GameFi, NFT gaming, and all blockchain layers. With six years of expertise, we've secured over 1000 projects globally, averting over $30 billion in losses. Our specialists rigorously audit smart contracts and ensure DApp safety on major platforms like Ethereum, BSC, Arbitrum, Algorand, Tron, Polygon, Polkadot, Fantom, NEAR, Solana, and others, guaranteeing your project's security with cutting-edge practices.

1000+
Audits Completed

$30B
Secured

1M+
Lines of Code Audited

Follow Our Journey

Audit Report
September, 2024

For

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillhash.com