QuillAudits

Audit Report
November, 2023

For

Table of Content

Executive Summary

Project Name	REAP
Project URL	*https://play.google.com/store/apps/details?id=com.reapaprill*
Overview	REAP (Recovery Encrypted Access Pass) is an encrypted private storage service for cryptocurrency wallet seed phrases developed by April Token. It is used to recover wallets in case of loss or damage. The app allows users to split their wallet seed phrase into multiple parts and store them separately for added security.
Audit Scope	https://play.google.com/store/apps/details?id=com.reapaprill https://github.com/Apriloracle/reapapp
Method	Manual Code Analysis, Automated Review
Review 1	3rd November 2023 - 17th November 2023
Updated Code Received	30th November 2023
Review 2	1st December 2023

Number of Issues per Severity

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	1	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	0	2	1	0

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
- Broken Access Controls
- Insecure Cryptographic Storage
- Insufficient Cryptography
- Insufficient Session Expiration
- Information Leakage
- Third-Party Components
- Rate Limit
- Input Validation
- Injection Attacks
- Cross-Site Request Forgery
- Broken Authentication and Session Management
- Insufficient Transport Layer Protection
- 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 REAP, 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.

Issues Found

Medium Severity Issues

1. API Key’s Leaked

Description

During the penetration testing conducted on the repository "https://github.com/Apriloracle/reapapp," it was identified that Freshchat API keys and Infura api key were exposed in the source code and configuration 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/Apriloracle/reapapp/blob/a34cafc7fdc026706c8da16b8bfb31ab1ba0f328/src/config/secrets.ts

https://github.com/Apriloracle/reapapp/blob/a34cafc7fdc026706c8da16b8bfb31ab1ba0f328/src/App.tsx#L52

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 Freshchat API keys poses a severe risk to the confidentiality and integrity of the associated Freshchat account. An attacker with access to these keys could potentially impersonate the application, eavesdrop on conversations, 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. Chat Bot Accessible without Authentication

Description

The chatbot service is accessible without requiring any authentication, posing a significant security risk. This vulnerability allows unauthorized users to interact with the chatbot, potentially leading to unauthorized access, data leakage, and misuse of the system.

Steps to Reproduce

Connect the phone through adb

- adb shell
- Su
- am start –n com.reapapril/com.freshchat.consumer.sdk.activity.ChannelListActivity

Recommendation

Check for authentication before starting the activity.

POC

```
tissot:/ # am start -n com.reapapril/com.freshchat.consumer.sdk.activity.ChannelListActivity
Starting: Intent { cmp=com.reapapril/com.freshchat.consumer.sdk.activity.ChannelListActivity }
```

Status

Resolved

Low Severity Issues

1. 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

- @babel/traverse
- crypto-js
- browserify-sign
- Json5
- Semver
- Activesupport
- xml2js
- node-fetch
- react-devtools-core
- axios

Recommended Fix

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

Acknowledged

2. HTML Injection

Description

HTML Injection is a security vulnerability that occurs when an attacker is able to inject malicious HTML code into a web application. This can happen when user input is not properly validated or sanitized before being displayed on a application. The injected code can be executed by other users, leading to various security risks.

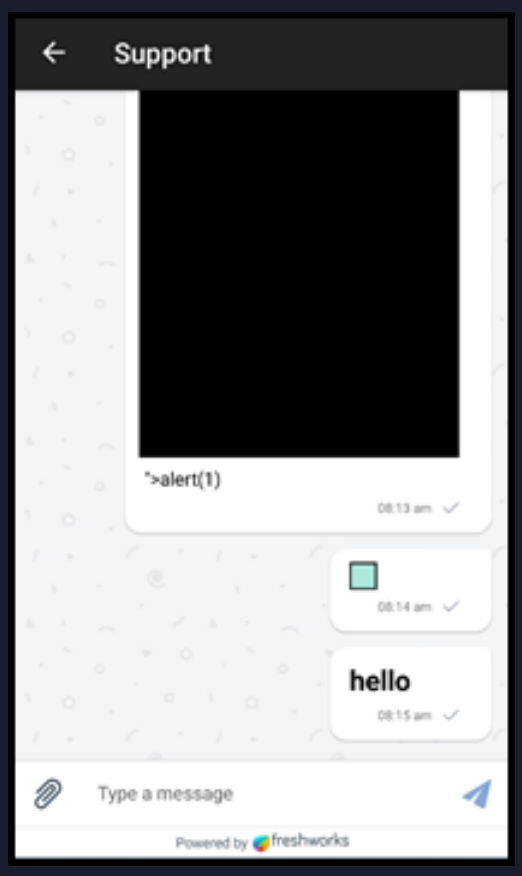
Steps to Reproduce

1) Open Support / Chat bot page in the reap application
2) Add html payloads like <h1>Hello</h1> or <img src=x onerror=alert(1)> or something similar
3) Payload will be rendered in the chat history

Recommendation

- Input Validation: Implement strict input validation on both client and server sides to ensure that user inputs are sanitized and do not contain malicious code.
- Output Encoding: Encode user inputs before rendering them in HTML to prevent the execution of injected scripts. Use functions such as htmlspecialchars() or equivalent in the respective programming language.

POC



Status

Resolved

Closing Summary

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

Some issues of 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/Wallet Pentesting Security Audit provides services to help identify and mitigate potential security risks in REAP Wallet. 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 REAP Wallet. 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 REAP to implement the recommendations provided in our audit reports and to take appropriate steps to mitigate potential security risks.

About QuillAudits

QuillAudits is a secure smart contracts audit platform designed by QuillHash Technologies. We are a team of dedicated blockchain security experts and smart contract auditors determined to ensure that Smart Contract-based Web3 projects can avail the latest and best security solutions to operate in a trustworthy and risk-free ecosystem.

850+
Audits Completed

$30B
Secured

$30B
Lines of Code Audited

Follow Our Journey

Audit Report
November, 2023

For

QuillAudits

Canada, India, Singapore, UAE, UK

www.quillaudits.com

audits@quillhash.com