QuillAudits

Audit Report
October, 2022

For

CrowdPad

Table of Content

Executive Summary 01

Checked Vulnerabilities 03

Techniques and Methods 04

Manual Testing 05

High Severity Issues 05

A.1 Android Debug is Enabled in Android Application 05

Medium Severity Issues 06

A.2 API Key Leaked 06

A.3 Multiple Social Media Links Hardcoded 06

A.4 Multiple Deprecated Libraries in Package-Lock.json 07

A.5 "usesCleartextTraffic" flag is True 08

Low Severity Issues 09

A.6 Cross-Origin-Resource-Sharing. 09

Closing Summary 10

About QuillAudits 11

Executive Summary

Project Name	CrowdPad
Timeline	October 6, 2022 - October 28, 2022
Scope of Audit	The scope of this pentest was to analyze the Debug-version Adndroid App for quality, security, and correctness.
In Scope	Android App (app.crowdpad.crowdpad) GitHub Source-Code *https://github.com/CrowdPad/crowdpad/tree/mainnet-test* (Last Commit Audited :- f0035dc664714958e8a8f529adf08dec300aac88)

High
Medium
Low
Informational

	High	Medium	Low	Informational
Open Issues	0	0	0	0
Acknowledged Issues	0	0	0	0
Partially Resolved Issues	0	0	0	0
Resolved Issues	1	4	1	0

Types of Severities

High

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

Medium

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

Low

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 severity 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.

Types of Issues

Open

Security vulnerabilities identified that must be resolved and are currently unresolved.

Resolved

These are the issues identified in the initial audit and have been successfully fixed.

Acknowledged

Vulnerabilities which have been acknowledged but are yet to be resolved.

Partially Resolved

Considerable efforts have been invested to reduce the risk/impact of the security issue, but are not completely resolved.

Checked Vulnerabilities

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- Exception Disorder
- Gasless Send
- Use of tx.origin
- Compiler version not fixed
- Address hardcoded
- Divide before multiply
- Integer overflow/underflow
- Dangerous strict equalities
- Tautology or contradiction
- Return values of low-level calls
- Missing Zero Address Validation
- Private modifier
- Revert/require functions
- Using block.timestamp
- Multiple Sends
- Using SHA3
- Using suicide
- Using throw
- Using inline assembly

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis

In this step, we have analysed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

Static Analysis

Static analysis of smart contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

Code Review / Manual Analysis

Manual analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analysed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

Gas Consumption

In this step, we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

Manual Testing

High Severity Issues

A1. Android Debug is Enabled in Android Application

Description

Consider a situation when your mobile is stolen and it is not rooted. If an application is marked as debuggable then any attacker can access the application data by assuming the privileges of that application or can run arbitrary code under that application permission. In the case of non-debuggable application, an attacker would first need to root the device to extract any data.

Vulnerable File

Decompile App using Jadx.
Check Android Manifest.xml File and line 56 and you can see android:debuggable="true"

Recommended Fix

Fix is very simple, just set android:debuggable flag to false in AndroidManifest.xml of the application.

Status

Resolved

Medium Severity Issues

A2. API Key Leaked

Description

These keys are usually used for internal purposes and so it shold not be leaked like this. Google API keys usually are used to update information on services or for analytics purposes.

Vulnerable File

Visit crowdpad-main/ios/GoogleService-Info.plist on GitHub Source Code

Recommended Fix

When use Google API key is recommended use vault or environment variable encrypted for the best security.

Status

Resolved

A3. Multiple Social Media Links Hardcoded

Description

Social Media accounts can be changed via username and so, in android applications, it should be fetching such information dynamically and should not be hardcoded. This arises a problem of broken link hijacks and such other security issues. If an attacker gets access to the such accounts of png endpoints they can manipulate the app content in their own way

Vulnerable File

Visit crowdpad-main/assets/tokens_list.json on GitHub Source Code

Recommended Fix

Keep Dynamic URL fetching mechanisms for such files

Status

Resolved

A4. Multiple Deprecated Libraries in Package-Lock.json

Description

Package-lock.json Stores files that can be useful for 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 that if the dependency used has an exploit in the version mentioned. It can be create a backdoor for an attacker.

Vulnerable File

Visit crowdpad-main/functions/package-lock.json

Vulnerable Dependencies

1) dicer
2) node-fetch
3) protobufjs
4) ansi-regex
5) node-forge
6) jose

Recommended Fix

When use Google API key is recommended use vault or environment variable encrypted for the best security.

Status

Resolved

A5. "usesCleartextTraffic" flag is True

Description

An unsecured communications channel between an app and any back-end services can expose the data transmitted between them. Similar to the App Transport Security (ATS) feature in iOS (see also best practice 6.5 Implement App Transport Security), Android 6.0 and later makes it easier to prevent an app from using cleartext network traffic (e.g., HTTP and FTP without TLS) by adding the android:usesCleartextTraffic attribute to the Android Manifest.

Vulnerable File

Decompile App using Jadx.
Check Android Manifest.xml File and line 56 and you can see
android:usesCleartextTraffic="true"

Recommended Fix

Set the usesCleartextTraffic flag to false in the Android Manifest. This will result in the app asking the platform and third-party libraries to prevent the app from using cleartext traffic.
Not all APIs will honor the flag, however, and developers/security analysts will need to determine what APIs will or will not honor the flag.
Also note that WebView will not honor the usesCleartextTraffic flag (if you must use WebView, see also best practice 7.9 - Follow WebView Best Practices).
If specific services used by the app require cleartext, developers can use the Network Security Configuration feature (android:networkSecurityConfig) to manually add exceptions if absolutely necessary.

Status

Resolved

Low Severity Issues

A6. Cross-Origin-Resource-Sharing

Description

Cross-origin resource sharing is a mechanism that allows restricted resources on a web page to be requested from another domain outside the domain from which the first resource was served. A web page may freely embed cross-origin images, stylesheets, scripts, iframes, and videos. An Attacker can use this for the malicious purpose to fetch content from internal pages.

Vulnerable File

Visit crowdpad-main/lib/Screens/Wallet/utils/base_account.dart in Github Source Code.
Visit rowdpad-main/lib/Screens/Wallet/utils/tracker.dart in Github Source Code.

Both above mentioned files have “headers['Access-Control-Allow-Origin'] = '*';”
Cors Allow Origin Wild Card
Cross-Origin Resource Sharing (CORS) allows a service to disable the browser’s Same-origin policy, which prevents scripts on an attacker-controlled domain from accessing resources and data hosted on a different domain. The CORS Access-Control-Allow-Origin HTTP header specifies the domain with permission to invoke a cross-origin service and view the response data. Configuring the Access-Control-Allow-Origin header with a wildcard (*) can allow code running on an attacker-controlled domain to view responses containing sensitive data.

Recommended Fix

When use Google API key is recommended use vault or environment variable encrypted for the best security.

Status

Resolved

Closing Summary

In this report, we have considered the security of the CrowdPad Android Application. We performed our audit according to the procedure described above.

Some issues of High, low and Informational severity were found, Some suggestions and best practices are also provided in order to improve the code quality and security posture. At the end, the CrowPad Team will resolve all issues.

Disclaimer

QuillAudits Dapp audit is not a security warranty, investment advice, or an endorsement of the CrowdPad Platform. This audit does not provide a security or correctness guarantee of the audited crowdPad Android and IOS Application.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multi-step process. One audit cannot be considered enough. We recommend that the CrowdPad Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

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.

600+
Audits Completed

$15B
Secured

600K
Lines of Code Audited

Follow Our Journey

Audit Report
October, 2022

For

CrowdPad

QuillAudits

Canada, India, Singapore, United Kingdom

audits.quillhash.com

audits@quillhash.com