Defly Secure Code Review

Report Executive Summary Presented to:

Blockshake.io

July 12, 2022
Version: 2.1

Presented by:

Kudelski Security, Inc.
5090 North 40th Street, Suite 450
Phoenix, Arizona 85018

EXECUTIVE SUMMARY

Overview

Blockshake.io engaged Kudelski Security to perform a Defly Secure Code Review.

The assessment was conducted remotely by the Kudelski Security Team. Testing took place on May 16, 2022- June 10, 2022, and focused on the following objectives:

- Provide the customer with an assessment of their overall security posture and any risks that were discovered within the environment during the engagement.
- To provide a professional opinion on the maturity, adequacy, and efficiency of the security measures that are in place.
- To identify potential issues and include improvement recommendations based on the result of our tests.

This executive summary provides an overview of the engagement, tests performed, and identified findings.

Key Findings

Overall, Kudelski found that the provided source implemented a high level of security across the reviewed security domains. No issues were identified that would result in direct exposure of application data or functionality.

The following are themes and issues identified during the testing period. These, along with other items, within the findings section, should be prioritized for remediation to reduce the risk they pose.

- Weakness in Metadata Verification Workflow – Some application security-sensitive processes are executed on developer machines. Architectural reconfiguration of this workflow lowered and mitigates most of the risk identified during the review.
- Denial of Service – A low-risk finding identified from rate-limiting which could be exploited to undermine application availability. This has been mitigated in a later review and verified by Kudelski Security.
- Insufficient Logging – Some low-risk findings related to conditions where shared token usage or logging configurations led to an inability to use logs to replay security incidents and identify users performing sensitive operations.

During the test, the following positive observations were noted regarding the scope of the engagement:

- Background Snapshot protections were enabled by default
- Application disallows screenshots, which bolsters security for users
- Application provides option for user to password/code-lock the application
- Newly implemented signature scheme reduces threat landscape for asset and pool verification.
- Wallet private keys are stored securely.

Scope and Rules of Engagement

Kudelski performed a security assessment for the Defly mobile application, related libraries, and services. The following table documents the targets in scope for the engagement. No additional systems or resources were in scope for this assessment.

In-Scope Applications	
Application	Purpose
Defly App	Mobile Application acting as wallet storage and an Algorand trading platform

Table 1: Scope

FINDINGS SUMMARY

During the Defly Secure Code Review, we discovered one finding that had a medium-severity rating, as well as four low-risk findings. After Defly engineers implemented mitigations, Kudelski validated all medium- and low-risk findings were remediated. Other identified items did not result in risk to the application and are listed as informational.

The following chart displays the findings by severity.

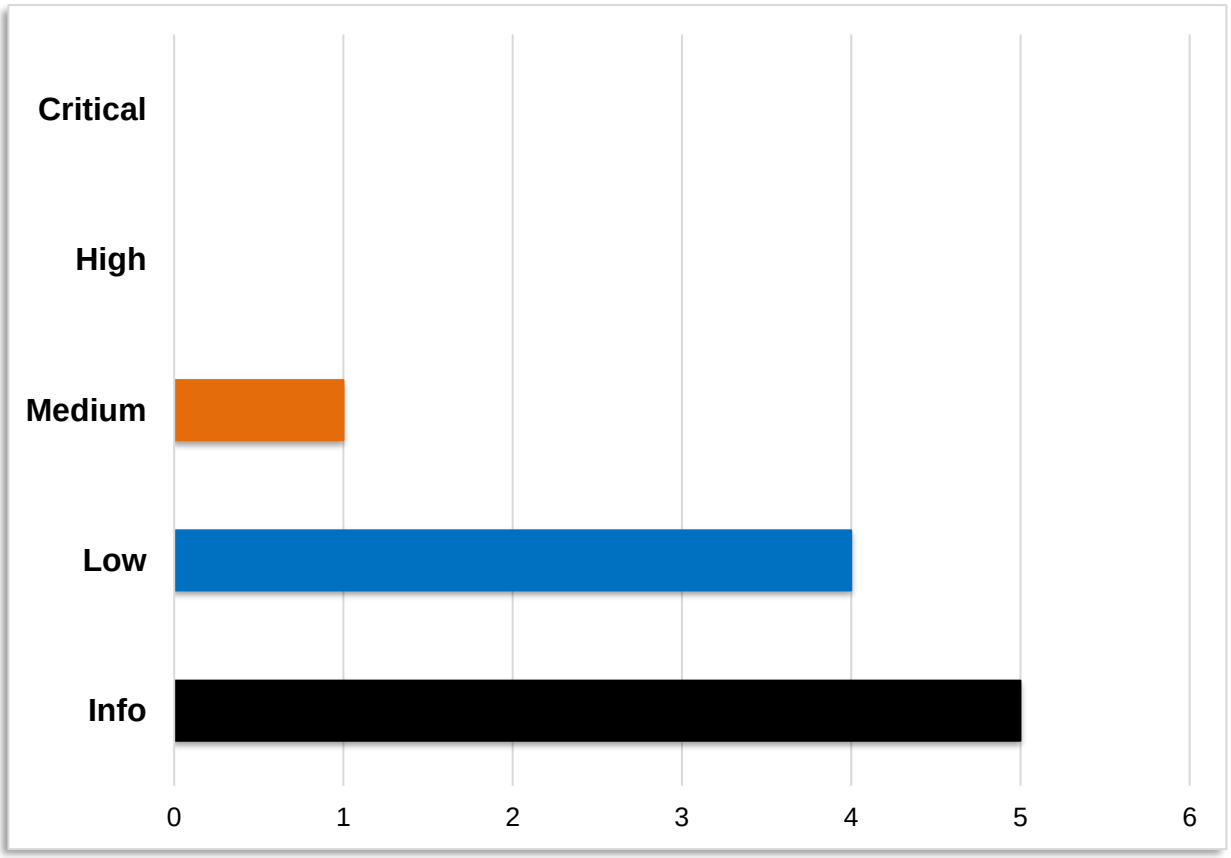

Figure 1: Findings by Severity

Threat Analysis

This threat analysis section summarizes the threat scope and key threats identified during the code review, which informed the secure code review analysis. It also contains descriptions of the threats discovered and potential vulnerabilities as well as any applicable recommendations for remediation.

Threat Scope

Kudelski utilized the Defly-provided architecture diagrams to identify threat boundaries, threat actors, and quantify possible threats to the provided application source, its infrastructure, and supporting processes. To further refine this activity, threat analysis to these components was scoped to actors targeting the provided codebases and supporting libraries.

Threat Actors

During the secure code review, Kudelski considered several different threat actors that could target various Defly components. Of the identified threat actors, malicious external attackers were considered most likely to target various Defly processes, users, and personnel. This is followed by malicious insiders such as administrators and developers and anonymous internal and external users.

Key Threats

- In the event that a malicious user or attacker can manipulate the Defly API or conduct man-in-the-middle attacks to return a maliciously crafted smart contract pool, then the attacker might be able to keep 1% of the swap for themself, or, in a worst-case scenario, an attacker could empty the pool, or manipulate prices to his or her own advantage. In these cases, trader or liquidity provider funds would be in peril.
- Defly security depends on secure network communications. Attacks targeting network weaknesses (e.g., man-in-the-middle, denial-of-service, broken cryptography) must be protected against.
- Since the Defly mobile applications only execute transactions that utilize trusted pools and assets signed by Defly utilizing a custom library, this process must be protected to prevent unauthorized tampering or inclusion of malicious assets or pools within the Defly API responses.
- The mobile Defly application executes transactions utilizing keys tied to the device and associated user wallet. Any sensitive tokens used in the process must be stored in a secure manner to prevent unauthorized access to user.
- The build process pulls in libraries from public repositories, exposing the various application components to supply-chain attacks by malicious attackers.

Findings Summary

The following table provides a summary of the identified findings and associated risks.

#	Severity	Description	Status
1	Medium	Weakness in Metadata Verification Workflow	Mitigated
~~2~~	Low	Denial of Service	Mitigated
3	Low	Insufficient Logging - Shared Token Usage	Mitigated
4	Low	Insufficient Logging	Mitigated
5	Low	Insecure String Comparison	Mitigated
6	Informational	Lack of Certificate Pinning	Acknowledged
7	Informational	Platform Disclosure	Acknowledged
8	Informational	Explicit Input Validation	Mitigated
9	Informational	Unnecessary Content	Mitigated
10	Informational	External Call from an Internal Resource	Acknowledged

Table 2: Findings Overview

METHODOLOGY

During this source code review, the Kudelski Security Services team reviewed code within the project within an appropriate IDE. During every review, the team spends considerable time working with the client to determine correct and expected functionality, business logic, and content to ensure that findings incorporate this business logic into each description and impact. Following this discovery phase the team works through the following categories:

- Authentication
- Authorization and Access Control
- Auditing and Logging
- Injection and Tampering
- Configuration Issues
- Logic Flaws
- Cryptography

Approach

Kudelski utilizes a standard methodology for assessments that is comprised of three phases: information gathering, vulnerability identification, and reporting. Each phase feeds the next, but any activity in later phases may inform additional research and testing. The activities are cyclical to provide the analyst with working knowledge of the targeted properties for additional threat vectors.

Security methods in Cryptocurrency and Cryptocurrency Exchanges

In analyses of the threat vectors facing Cryptocurrency applications, source code, and exchanges, Kudelski uses a testing regimen that follows a best-practices heuristic recognizing five likely areas of security weaknesses specific to cryptocurrency: 1) Susceptibility to phishing; 2) Weak hot wallet protections; 3) Broken Authorization class vulnerabilities related to login credentials of individuals with privileged roles; 4) Software vulnerabilities; and 5) Transaction malleabilities.

Information Gathering

Kudelski starts by reviewing application endpoints based on availability, application use-cases, developer documentation, and application source code. These endpoints are analyzed for use, potential parameters, additional attack surface, and possible threats. Applications are reviewed during this phase from multiple points of view, including an anonymous, un-authenticated user, an authenticated user, and an authenticated partner.

Kudelski analyzes available endpoints and source code during this phase for controls that affect security posture, including authentication and authorization controls, logging behavior, communication protocols, input handling, encryption settings, and other application behavior.

Vulnerability Identification

Kudelski uses the identified endpoints and controls of the identified assets to identify and explore possible security vulnerabilities across applications based on our expertise in assessing application flaws. Special attention will be paid to possible fraud and business logic flaws that could affect the Client, its partners, or its customers.

Kudelski utilizes industry-standard vulnerability lists for assessment purposes, including OWASP's Application Security Verification Standard, the OWASP Top 10 Security Risks, and the SANS CWE Top 25 Software Errors. These vulnerabilities are assessed across various security domains as they apply to

the targeted application. Additional attack surfaces and weaknesses may be noted during this portion of the assessment for further research.

Reporting

To finalize the assessment activity, Kudelski documents the assessment vulnerabilities, endpoints, and findings in a report that summarizes the results into actionable items for remediation by the Client. Each finding documents the steps required to reproduce identified vulnerabilities and includes recommendations for remediating or mitigating the threat.

Tools

The following tools were used during this portion of the test. A link for more information about the tool is provided as well.

- Visual Studio Code - https://code.visualstudio.com/
- Semgrep - https://semgrep.dev/
- Dependency Check - https://owasp.org/www-project-dependency-check/
- Burp Suite Professional - https://portswigger.net/burp

Vulnerability Scoring Systems

Kudelski Security utilizes a vulnerability scoring system based on impact of the vulnerability, likelihood of an attack against the vulnerability, and the difficulty of executing an attack against the vulnerability based on a high, medium, and low rating system

Impact
The overall effect of the vulnerability against the system or organization based on the areas of concern or affected components discussed with the client during the scoping of the engagement.

> **High:**
> The vulnerability has a severe effect on the company and systems or has an affect within one of the primary areas of concern noted by the client
>
> **Medium:**
> It is reasonable to assume that the vulnerability would have a measurable effect on the company and systems that may cause minor financial or reputational damage.
>
> **Low:**
> There is little to no affect from the vulnerability being compromised. These vulnerabilities could lead to complex attacks or create footholds used in more severe attacks.

Likelihood
The likelihood of an attacker discovering a vulnerability, exploiting it, and obtaining a foothold varies based on a variety of factors including compensating controls, location of the application, availability of commonly used exploits, and institutional knowledge

> **High:**
> It is extremely likely that this vulnerability will be discovered and abused
>
> **Medium:**
> It is likely that this vulnerability will be discovered and abused by a skilled attacker
>
> **Low:**
> It is unlikely that this vulnerability will be discovered or abused when discovered.

Difficulty
Difficulty is measured according to the ease of exploit by an attacker based on availability of readily available exploits, knowledge of the system, and complexity of attack. It should be noted that a LOW difficulty results in a HIGHER severity.

> **High:**
> The vulnerability is difficult to exploit and requires advanced knowledge from a skilled attacker to write an exploit
>
> **Medium:**
> The vulnerability is partially defended against, difficult to exploit, or requires a skilled attacker to exploit.
>
> **Low:**
> The vulnerability is easy to exploit or has readily available techniques for exploit

Severity
Severity is the overall score of the weakness or vulnerability as it is measured from Impact, Likelihood, and Difficulty

CWE

The CWE system is a community-developed list of common software security weaknesses. It serves as a common language, a measuring stick for software security tools, and as a baseline for weakness identification, mitigation, and prevention efforts. Some common types of software weaknesses classified by the CWE are:

- Buffer Overflows, Format Strings, etc.
- Structure and Validity Problems
- Common Special Element Manipulations
- Channel and Path Errors
- Handler Errors
- User Interface Errors
- Pathname Traversal and Equivalence Errors
- Authentication Errors
- Resource Management Errors
- Insufficient Verification of Data
- Code Evaluation and Injection
- Randomness and Predictability