Secure Code Review

Findings and Recommendations Report Presented to:

Amulet Global (MTR Labs)

August 11, 2022
Version: 1.1

Presented by:

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

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

Overview

Amulet Global (MTR Labs) engaged Kudelski Security to perform a secure code assessment on the Amulet protocol smart contract system.

The assessment was conducted remotely by the Kudelski Security Team. The source code review took place from 7/14 – 8/11, and focused on the following objectives:

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

This report summarizes the engagement, tests performed, and findings. It also contains detailed descriptions of the discovered vulnerabilities, steps the Kudelski Security Team took to identify and validate each issue, as well as any applicable recommendations for remediation.

Key Findings

The issues found in the code were LOW or INFORMATIONAL findings. This shows that the overall risk profile of the application at the time of this assessment is low.

The following are the major 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.

- Safe math was used often, but should be used more consistently throughout the code to prevent potential vulnerabilities from being introduced in future updates.
- Insurance policy coverage duration, claim expiration dates and claim payout dates may be affected by a lack of precision when dev Epochs are used in place of UNIX time.
- Single administrator accounts have significant capabilities. These functions should be limited by requiring multiple signers to prevent collusion. We have been informed that this is currently occurring off-chain, but this should occur on-chain in the future for transparency.

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

- The code is well organized.
- Client contacts were very amenable to conducting joint secure code reviews with the Kudelski Security smart contract auditing team.
- Anchor framework usage is very consistent and follows the recommended syntax.
- Critical issues in architecture or code logic were discussed immediately via teleconference.

Scope and Rules of Engagement

Kudelski performed a Secure Code Review for Amulet Global on the Amulet protocol smart contract system. All rust files in the repo were the targets in scope for the engagement. Connections to Raydium and Saber liquidity pools were considered during the overall code review and will continue to be reviewed as Amulet service offerings grow.

The source code was supplied through a public repository:

In-scope source code, contracts, or programs	
https://github.com/Amulet-Protocol/meta-programs	Commit: a092a2b4565a4bda49a73bcc08e8601236a70eb5

Table 1: Scope

TECHNICAL ANALYSIS & FINDINGS

During the Secure Code Review, the Kudelksi Security team stayed in contact with the Amulet team and shared findings on a twice weekly basis so that questions could be answered, and critical issues could be remediated immediately. We discovered one (1) medium severity finding that was resolved and removed prior to draft publication, one (1) low severity finding that was accepted and closed and three (3) informational findings that have been resolved.

The following chart displays the findings by severity.

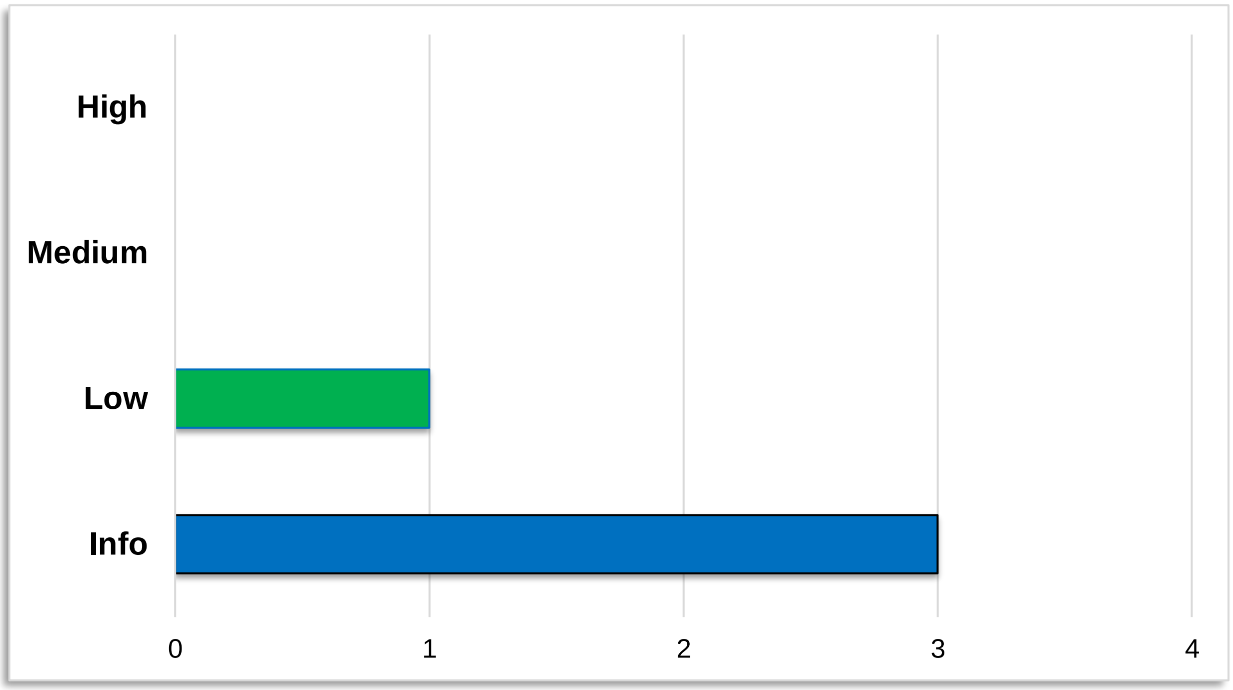

Figure 1: Findings by Severity

Findings

The *Findings* section provides detailed information on each of the findings, including methods of discovery, explanation of severity determination, recommendations, and applicable references.

The following table provides an overview of the findings.

#	Severity	Description	Resolution Status
01	**Low**	Utilize UNIX time when more precision is required for insurance policies, coverage duration and insurance claims	Accepted – Closed
02	**Informational**	Unchecked deserialization function is utilized	Resolved
03	**Informational**	Unsafe math is used in multiple places	Resolved
04	**Informational**	No constraint in place when closing an account and initiating a transfer of SOL	Resolved

Table 2: Findings Overview

KSI-AM-01 – Utilize UNIX Time When More Precision is Required for Insurance Policies, Coverage Duration and Insurance Claims

Severity	LOW

Impact	Likelihood	Difficulty
Medium	Low	High

Description

The Kudelski Security team noticed and followed up with developers via conference call; that an epoch is used in several places within the code. The Solana SDK uses the concept of a dev epoch, while Anchor and Serum will utilize and subsequently have documentation around the UNIX epoch – which functions like a system clock, as well as the dev epoch.

KSI-AM-02 – Unchecked Deserialization Function is Utilized

Severity	Informational

Impact	Likelihood	Difficulty
N/a	N/a	N/a

Description

The Kudelski Security team noticed that stake_wrapper.rs utilizes an unchecked deserialization function.

KSI-AM-03 – Unsafe Math is used in Multiple Places

Severity	Informational

Impact	Likelihood	Difficulty
N/a	N/a	N/a

Description

The Kudelski Security team noticed that unsafe math is used in several places within the code.

KSI-AM-04 – No Constraint in Place when Closing an Account and Initiating a Transfer of SOL

Severity	Informational

Impact	Likelihood	Difficulty
N/a	N/a	N/a

Description

The Kudelski Security team noticed that there did not seem to be an owner or authority assigned when closing or transferring SOL to a program PDA account.

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

These categories incorporate common Solana vulnerabilities such as:

- Missing signer checks
- Missing ownership checks
- Missing rent exemption checks
- Signed invocation of unverified programs
- Solana account confusions
- Re-initiation with cross-instance confusion
- Arithmetic overflow/underflows
- Numerical precision errors
- Incorrect calculation
- Casting truncation
- Exponential complexity in calculation
- Insufficient SPL-Token account verification
- Unsafe design
- Any hidden backdoors
- Call stack depth
- Contract logic correctly implements the project specifications
- Flaws in design, logic, or access control
- Compiler warnings

These categories incorporate common Rust vulnerabilities such as:

- Unsafe Rust code: If a smart contract contains any unsafe Rust code, it may still suffer from memory corruption such as buffer overflows, use after frees, uninitialized memory, etc.
- Outdated dependencies
- Redundant code
- Do not follow security best practices

Tools

The following tools were used during this portion of the secure code audit.

- Visual Studio 2022
- Visual Studio Code
- Semgrep

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 effect 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

KUDELSKI SECURITY CONTACTS

NAME	POSITION	CONTACT INFORMATION
Ramsey El-Khazen	Head of Blockchain Security EMEA	Ramsey.El-Khazen@KudelskiSecurity.com
Kelly Ryver	Blockchain Security Expert	Kelly.Ryver@KudelskiSecurity.com
Rez Khan	Blockchain Security Expert	Rez.Khan@KudelskiSecurity.com