Secure Code Review

Findings and Recommendations Report Presented to:

PsyOptions

July 18, 2022
Version: 4.0

Presented by:

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

PUBLIC

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

EXECUTIVE SUMMARY

Overview

PsyOptions engaged Kudelski Security to perform a secure code assessment of the PsyStake and Psyfi-Euros repositories.

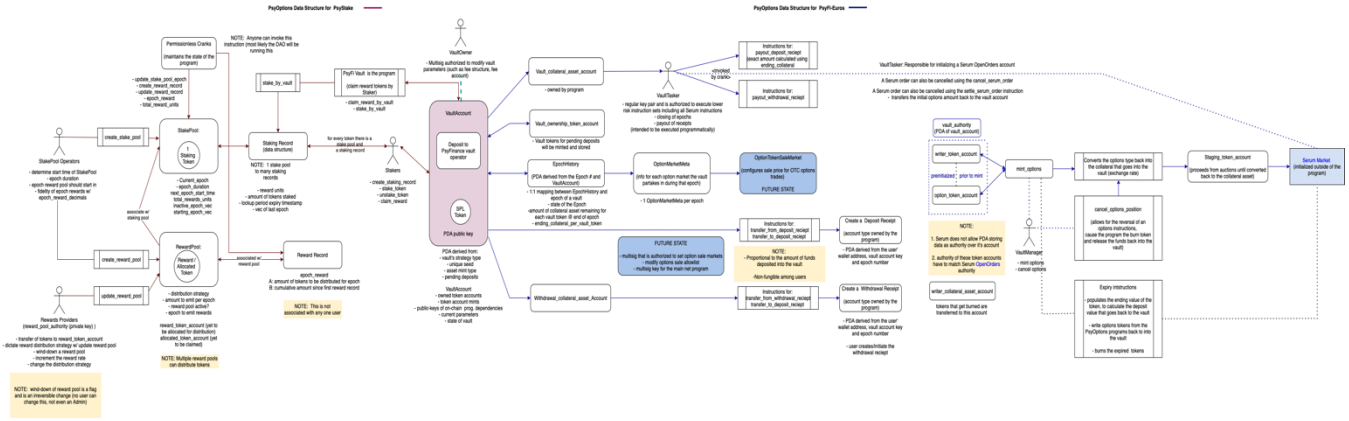
The assessment was conducted remotely by the Kudelski Security Team. Testing took place on April 18, 2022 - June 10, 2022, and focused on the following objectives:

- Provide the PsyOptions with an assessment of the security of recently added functionality
- To provide a professional opinion on the maturity, efficiency, and coding practices
- To identify potential issues and include improvement recommendations based on the result of our 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.

Architecture Review

The following set of diagrams reflect the architecture and data structure for both ***PsyStake*** and ***Psyfi-Euros***; the diagrams were combined since both PsyStake and Psyfi-Euros utilize the ***PsyOptions*** vault design. **(**See attached draw.io file with deliverable)



Key Findings

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 to the risk they pose.

- VaultTasker is a single point of failure

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

- Great usage of anchor and Solana security best practices
- Consistent with checking for safe math operations
- Used integration test in both rust and typescript
- The code was well organized and contained comments that provided insight into functionality

Scope and Rules of Engagement

Kudelski performed a Secure Code Review for PsyOptions. The following table documents the targets in scope for the engagement. No additional systems or resources were in scope for this assessment.

PsyStake Commit Hash	
Commit Hash	721be1e546a092971080a263c1ade8231adc7e24

Table 1: PsyStake Commit Hash

Psyfi-Euros Commit Hash	
Commit Hash	6094109d094e7dfcaeed7defce6908fc94b1be4a

Table 2: Psyfi-Euros Commit Hash

TECHNICAL ANALYSIS & FINDINGS

The following chart displays the findings by severity.

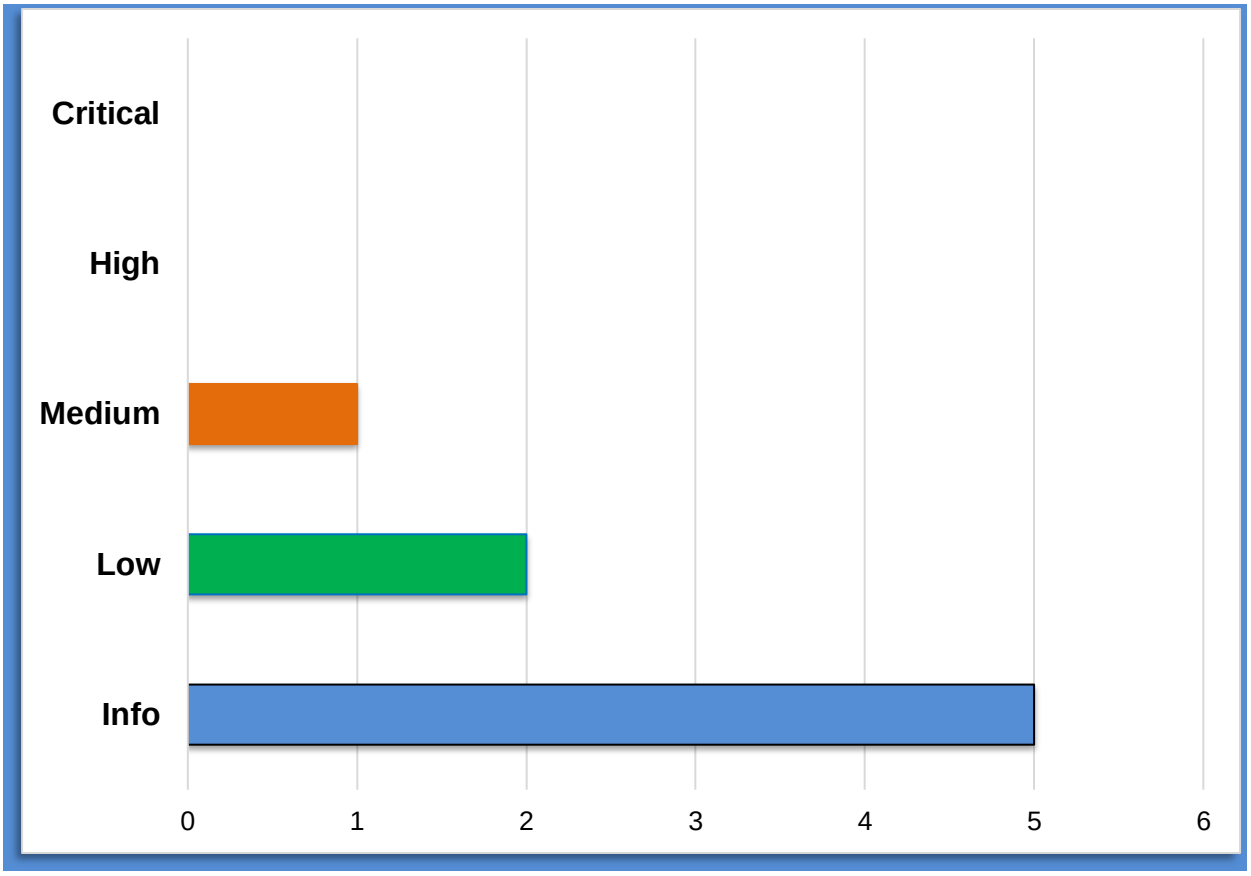


Figure 1: Findings by Severity

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.

#	Status	Severity	Description
1	Resolved	Medium	VaultTasker is a single point of failure
2	Resolved	Low	Unsafe Math
3	Resolved	Low	Use of Deprecated Pyth Client
4	Resolved	Informational	Incorrect Arithmetic
5	Resolved	Informational	Unchecked Account Type
6	Resolved	Informational	Mismatch of Comment and Intended Functionality
7	Accepted	Informational	Multiple references to epoch number
8	Resolved	Informational	Vault token supply or vault collateral asset account amount should not be equal to zero

Table 2: Findings Overview

KSI-01 – Overly restricted authority mechanism that can result in potential DoS

Severity	Medium

Impact	Likelihood	Difficulty
High	Low	Low

Description

The Kudelski Security Team noticed that the VaultTasker service represented a potential risk of denial of service for PsyOptions platform's users. VaultTasker was observed to be an external service, maintained, and operated by the PsyOptions team in Google Cloud functions, that managed the epoch transitions. The team verified that the VaultTasker service has special permissions through checking the signature of the instruction shown on evidence. After carefully reviewing the architecture of the PsyOptions service, we concluded that the function close_post_expiry was overly restricted.

KSI-02 – Unsafe Math

Severity	Low

Impact	Likelihood	Difficulty
Low	Low	High

Description

The Kudelski Security Team noticed that on line 56:`one_in_aligned_decimals` and line 57: `one_in_oracle_decimals` the program performs exponential calculations that could potentially cause an overflow to occur. While the place of origin for `aligned_to_decimals` was not observed in the scope of Psyfi-Euros, there were no identified validation checks used for this value to prevent an overflow from occurring. The program consumed `exp` from Pyth without any validation checks on its value as well.

KSI-03 – Use of Deprecated Pyth Client

Severity	Low

Impact	Likelihood	Difficulty
Low	Low	High

Description

The Kudelski Security Team noticed that the cargo.toml:line *27* revealed the use of pyth-client version 0.2.2 which is a deprecated crate for the Pyth oracle. This means that the observed dependency for Pyth that was used for Psyfi-Euros during the code review is not supported by the original developers of that crate. The new Pyth sdk/crate implements additional functionality along with slight fixes.

KSI-04 – Incorrect Arithmetic

Severity	Informational

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

Description

The Kudelski Security Team noticed that the arithmetic displayed in the code comments is incorrect and the code comments may reflect the total bytes used for token sales on the OptionMarketMeta. The arithmetic in the code below should equal the following: ((32*3) + 9) = 105 bytes

Additionally, the same observation was made for the DepositReceipt struct. The arithmetic in the code below should equal the following: ((32*3) + 8 + 3) = 107 bytes

KSI-05 – Unchecked Account Type

Severity	Informational

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

Description

The Kudelski Security Team noticed the unchecked account type `AccountInfo` was used within the reviewed codebase. When using the `AccountInfo` type, owner check bugs and account type substitution bugs become more prevalent.

KSI-06 – Mismatch of Comment and Intended Functionality

Severity	Informational

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

Description

The Kudelski Security Team noticed a mismatch of the design in the comments and what was implemented in the code was identified. On line 46 in utils.rs the comment mentioned the reward_multiplier being divided by 10 instead of the performed 100. Although this mismatch occurred, the intended reward multiplier calculations appeared to be performed correctly.

KSI-07 – Multiple references to epoch number

Severity	Informational

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

Description

The Kudelski Security Team noticed when the next epoch number is calculated, it is stored in multiple places, which creates a duplication of the same information.

KSI-08 – Vault token supply or vault collateral asset account amount should not be equal to zero

Severity	Informational

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

Description

The Kudelski Security Team noticed the code comments and the following calculation:
Collateral to withdraw = withdrawal_token_amount / vault token supply *
vault_collateral_asset_account.amount

There are two methods to perform this calculation:
Method 1: Using PEMDAS order of mathematical operations (multiply before dividing)
Method 2: Using ISO standards for mathematical operations (run computation from left to right)

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

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 2019
- 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 affect 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 affect 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

KUDELSKI SECURITY CONTACTS

NAME	POSITION	CONTACT INFORMATION
Kelly Ryver	**Blockchain Security Expert**	**Kelly.Ryver@kudelskisecurity.com**
Miles Nolan	**Blockchain Security Analyst**	**Miles.Nolan@kudelskisecurity.com**
Matias Barrios	**Blockchain Security Expert**	**Matias.Barrios@Skidata.com**