STRONGHOLD
SECURITY

Papaya Finance Security Audit Report

Apr 16, 2024

Contents

Executive Summary

Title	Description
Client	Papaya Finance
Project	Papaya Protocol
Platform	Ethereum
Language	Solidity
Repository	https://github.com/papaya-metaverse/Payout
Initial commit	c03732d4471894a630f4b709db31739ae5ee71d3, 916cfb5231960a7ecf863e97dab78f6705a3a62b, 452be1c32faccd6e4bcd0b79e792051704888a09, 70fc169ba958e4707f0fba6fc0dd30013003ea1e
Final commit	2848ebe66900134b73336bc6b1ddf60566d256bb
Timeline	March 22 2023 - April 16 2024

Project Overview

The Papaya Finance platform offers subscription services with financial transactions that are conducted using cryptocurrencies.

Audit Scope

File	Link
UserLib.sol	UserLib.sol
Papaya.sol	Papaya.sol
BySig.sol	BySig.sol

Audit Methodology

General Code Assessment

The code is reviewed for clarity, consistency, style, and whether it follows code best practices applicable to the particular programming language used, such as indentation, naming convention, commented code blocks, code duplication, confusing names, irrelevant
or missing comments, etc. This part is aimed at understanding the overall code structure and protocol architecture. Also, it seeks to learn overall system architecture and business logic and how different parts of the code are related to each other.

Code Logic Analysis

The code logic of particular functions is analyzed for correctness and efficiency. The code is checked for what it is intended for, the algorithms are optimal and valid, and the correct data types are used. The external libraries are checked for relevance and correspond to the tasks they solve in the code. This part is needed to understand the data structures used and the purposes for which they are used. At this stage, various public checklists are applied in order to ensure that logical flaws are detected.

Entities and Dependencies Usage Analysis

The usages of various entities defined in the code are analyzed. This includes both: internal usage from other parts of the code as well as possible dependencies and integration usage. This part aims to understand and spot overall system architecture flaws and bugs in integrations with other protocols.

Access Control Analysis

Access control measures are analyzed for those entities that can be accessed from outside. This part focuses on understanding user roles and permissions, as well as which assets should be protected and how.

Use of checklists and auditor tools

Auditors can perform a more thorough check by using multiple public checklists to look at the code from different angles. Static analysis tools (Slither) help identify simple errors and highlight potentially hazardous areas. While using Echidna for fuzz testing will speed up the testing of many invariants, if necessary.

Vulnerabilities

The audit is directed at identifying possible vulnerabilities in the project's code. The result of the audit is a report with a list of detected vulnerabilities ranked by severity level:

Severity	Description
● Critical	Vulnerabilities leading to the theft of assets, blocking access to funds, or any other loss of funds.
● High	Vulnerabilities that cause the contract to fail and that can only be fixed b y modifying or completely replacing the contract code.
● Medium	Vulnerabilities breaking the intended contract logic but without loss of fun ds and need for contract replacement.
● Low	Minor bugs that can be taken into account in order to improve the overall qu ality of the code

After the stage of bug fixing by the Customer, the findings can be assigned t he following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect it s security.
Acknowledged	The Customer took into account the finding. However, the recommendations wer e not implemented since they did not affect the project's safety.

Findings Summary

Severity	# of Findings
● Critical	1
● High	0
● Medium	8
● Low	7

ID	Severity	Title	Status
C-1	● Critical	A user can drain the protocol balance	Fixed
M-1	● Medium	The liquidator may lose funds	Acknowledged
M-2	● Medium	Authors may lose their income funds	Acknowledged
M-3	● Medium	Mismatch of donations and awards	Acknowledged
M-4	● Medium	The reward for the liquidator may be lower than the cost of gas for liquidation	Acknowledged
M-5	● Medium	A user can liquidate themselves and lose the assets	Fixed
M-6	● Medium	The user balance may change due to unchecked casts	Fixed
M-7	● Medium	User can't Unsubscribe or Liquidate	Acknowledged
M-8	● Medium	The `rescueFunds` function compares balances incorrectly	Fixed
L-1	● Low	Subscribe to themselves	Fixed
L-2	● Low	The Chainlink's `latestRoundData` can return stale results	Acknowledged
L-3	● Low	The `_encodeRates` function may return the same result with different inputs	Acknowledged
L-4	● Low	Denial of the ChainLink Oracle service	Acknowledged
L-5	● Low	Fee Tokens	Acknowledged
L-6	● Low	Use `Ownable2Step` instead of `Ownable`	Acknowledged

L-7	● Low	Null address checks are missing	Acknowledged

Findings

Critical

C-1 ● Critical A user can drain the protocol balance **Fixed**

Description

Papaya.sol#L180-L199

A user can drain the protocol balance by setting `outgoingRate` and `incomeRate` to a negative number.
A user can subscribe it to another user with `type(uint96).max`.
So, the rates are -1, and the rate can be lower than -1 if an attacker uses a value smaller than `type(uint96).max`

Recommendation

We recommend checking the `subscriptionRate` parameter in the `subscribe` function, which is not higher than `type(int96).max`.

Client's commentary

Fixed in 916cfb52

High

Not Found

Medium

M-1 ● Medium The liquidator may lose funds Acknowledged

Description

Papaya.sol#L217

UserLib.sol#L73-L77

When the user's balance decreases significantly and becomes negative, the liquidator may lose their funds due to the `drainBalance` implementation.

Because of:

```
        balance = user.balance;
        liquidator.balance += balance;
```

Here, the liquidator receives the user's balance, but if the user's balance is negative, it decreases the liquidator's balance.

Recommendation

We recommend adding only a positive balance to the liquidator.

Client's commentary

If the liquidator loses his funds during the liquidation process, he can request a refund.

M-2 ● Medium **Authors may lose their income funds** Acknowledged

Description

With the current implementation, the following case is possible:

One author has wealthy users and another one has poor subscribers.

Poor subscribers don't transfer funds to their balance; their balances decrease, but no one liquidates them.

Next, wealthy subscribers donate to the first author, but since no one liquidates subscribers of the second author, the second author can receive funds from the first author.

Let's consider the next steps:

1. Two authors: `a1`, `a2`;
2. Two subscribers: `s1`, `s2`, both have 100 coins;
3. `s1` subscribes to `a1` with rate=10;
4. `s2` subscribes to `a2` with rate=10;
5. after 10 seconds, both of them donate 100 coins (10 * 10) to `a1` and `a2`;
6. next, `s2` transfers the next 100 coins to the protocol;
7. `s1` balance = 0, `s2` balance = 100;
8. after 10 seconds, both of them donate another 100 coins (10 * 10) to `a1` and `a2`;
9. `s1` balance = -100, s2 balance = 0, `a1` balance = 200, `a2` balance = 200, contact balance = 300;
10. Next, `a1` withdraws all their funds (200), and from this moment, `a2` can withdraw only 100 instead of 200;

Recommendation

We recommend having a separate storage for each author.

Client's commentary

In the described case, the user will be liquidated. A negative balance is required for the protocol to function properly. This ensures that if the user was not liquidated, it can be done later.

M-3 ● Medium Mismatch of donations and awards Acknowledged

Description

When a user's balance becomes negative, the author's balance grows. After the user is liquidated, the author's balance does not change.

For example:

1. User balance = -10, rating = 1, author balance = 0;
2. After 10 seconds, balance = -20, rate = 1, author balance = 10;
3. If someone now eliminates this user, the author's balance will still be 10;

Recommendation

We recommend having a separate repository for each author.

Client's commentary

In the described case, the user will be liquidated.

M-4 ● Medium **The reward for the liquidator may be lower than the cost of gas for liquidation** Acknowledged

Description

[Papaya.sol#L210]

`_liquidationThreshold(account)` will return the amount of native tokens spent as gas to "unfollow" from authors that the `account` follows.

But there is a chance that the reward that the `liquidator` will receive will be less than what the `liquidator` spent on the TX execution.

Recommendation

We recommend adding setter functions for the `APPROX_LIQUIDATE_GAS` and `APPROX_SUBSCRIPTION_GAS` params for the owner only.

Client's commentary

If the liquidator loses his funds during the liquidation process, he can request a refund.

M-5 ● Medium A user can liquidate themselves and lose the assets Fixed

Description

Papaya.sol#L208

A user can liquidate themselves and lose the assets (UserLib.sol#L73):

```
liquidator.balance += balance;
user.balance = 0;
```

Recommendation

We recommend disallowing calling `liquidate` on the same address as `_msgSender()`.

Client's commentary

Fixed in 916cfb52

M-6 ● Medium **The user balance may change due to unchecked casts** Fixed

Description

Since all transfer-related functions accept `uint256` as the amount, but the user balance is stored as `int256` (UserLib.sol#L15),
the `decreaseBalance` function (UserLib.sol#L66)
and the `increaseBalance` function (UserLib.sol#L62)
perform casts of the amount from `uint256` to `int256` to manipulate the user balances. A hacker can call the `pay` function (Papaya.sol#L176)
with `amount` set to `2 ** 256 - X` to move the `X` token amount from any user participating in the contract.

Recommendation

We recommend checking values after the cast.

Client's commentary

Fixed in 916cfb52

M-7 Medium User can't Unsubscribe or Liquidate Acknowledged

Description

Papaya.sol#L249-L258

Papaya.sol#L215

Papaya.sol#L205

The `unsubscribe` and `liquidate` functions might fail due to the author or the admin Threshold level break.

This might happen in the `_unsubscribeEffects` function.
When the income rate decreases, it's additionally checks the Threshold level:

```
users[author].decreaseIncomeRate(..., _liquidationThreshold(author));
users[admin].decreaseIncomeRate(..., _liquidationThreshold(admin));
```

That transaction is reverted if the `author`' or `admin`' balance is less than `_liquidationThreshold`.

A possible case:

1. `user1` subscribes to `user2` to pay them `50 tokens/second`;
2. `user2` subscribes to `user1` to pay them `500 tokens/second`;
3. Wait a bit;
4. `user1` wants to unsubscribe, so they call the `unsubscribe` function, which calls the `decreaseIncomeRate` function;
5. In `decreaseIncomeRate` for `user2`, the income rate decreases to 0, but their outgoing rate is still 500;
6. Therefore, the transaction will be reverted.

Also, if any of the subscriptions cannot be canceled, the liquidation function will not be fully executed.

Recommendation

We recommend changing the synchronization mechanism for income rates.

Client's commentary

In the current implementation, a transfer will be made to one of the blocking accounts, and all participants will be liquidated.

M-8 ● Medium The `rescueFunds` function compares balances incorrectly Fixed

Description

Papaya.sol#L99

The `totalSupply` variable has 18 decimals, but the `balanceOf` function call may return a value with a different number of decimals.

Recommendation

We recommend fixing the `rescueFunds` function.

Client's commentary

Fixed in 2848ebe6

Low

L-1 ● Low **Subscribe to themselves** Fixed

Description

A user can subscribe to themselves.

Recommendation

We recommend disallowing users from subscribing to themselves.

Client's commentary

Fixed in [2848ebe6]

L-2 ● Low The Chainlink's `latestRoundData` can return stale results Acknowledged

Description

Papaya.sol#L224-L225

The `ChainlinkAdapter` accesses the Chainlink oracle, receiving `lateRoundData()`. Suppose Chainlink has a problem launching a new round and moving to consensus on a new scenario for an Oracle user. In that case, this contract may continue to use virtual or uncorrected data.

Recommendation

We recommend adding checks for the Chainlink (for `TOKEN_PRICE_FEED` and `COIN_PRICE_FEED`):

L-3	● Low	The `_encodeRates` function may return the same result with different inputs	Acknowledged

Description

The `projectId` is uint256 as an input, but in the function this `uint256` should fit in `uint64` slot(256-96-96 = 64)

So, if `projectId` is `type(uint64).max+1`or `projectId` == 0 and other values are 1, the `_encodeRates` function will return the same result.

```
// (1,1,type(uint64).max+1)
_encodeRates(1,1,18446744073709551616) = _encodeRates(1,1,0)
```

Recommendation

We recommend adding additional checks for inputs or setting `projectId` as a `uint64` value.

L-4 ● Low **Denial of the ChainLink Oracle service** Acknowledged

Description

ChainLink oracles can block access to price feeds, which will render the contract functionality unavailable since the `_liquidationThreshold` method directly uses these feeds ([Papaya.sol#L224]). Thus, any method which calls `_liquidationThreshold` will also become unavailable (`withdraw`, `withdrawTo`, `pay`, `subscribe`, `unsubscribe`, `liquidate`)

Recommendation

We recommend using a `try {} catch {}` structure when working with the oracles to prevent denial of service.

L-5 ● Low Fee Tokens Acknowledged

Description

Some ERC20 tokens charge a transaction fee for every transfer (for example, USDT is a fee token with a null commission now). Thus, the amount of the tokens received using `transferFrom` may differ from the transfer amount.

[Papaya.sol#L157]()
[Papaya.sol#L159]()
[Papaya.sol#L173]()

Recommendation

We recommend checking token balances before and after `transferFrom`.

L-6 ● Low Use `Ownable2Step` instead of `Ownable` Acknowledged

Description

Papaya.sol#L8

The `transferOwnership` function is used to change ownership from `Ownable.sol`. The owner may accidentally specify a non-active address and lose access. `Ownable2Step.sol` is more secure due to a 2-stage ownership transfer.

Recommendation

We recommend using the `Ownable2Step` contract from OZ (Ownable2Step.sol) instead.

Client's commentary

The multisig contract will be the owner of the protocol.

L-7 ● Low **Null address checks are missing** Acknowledged

Description

Adding null address checks can prevent users from accidentally losing funds.

Papaya.sol#L149

Papaya.sol#L167

Papaya.sol#L176

Papaya.sol#L180

Recommendation

We recommend adding null address checks for addresses.

Conclusion

During the audit process 1 CRITICAL, 8 MEDIUM and 7 LOW severity findings have been spotted.

Disclaimer

The Stronghold audit makes no statements or warranties about the utility of the code, the safety of the code, the suitability of the business model, investment advice, endorsement of the platform or its products, the regulatory regime for the business model, or any other statements about the fitness of the contracts to purpose, or their bug-free status. The audit documentation is for discussion purposes only.