Vesper Pools Security Audit

coinspect

Smart Contract Audit Vesper Pools - PaymentSplitter

Prepared for Bloq • February 2021

v210210

1. Executive Summary

In February 2021, Bloq engaged Coinspect to perform a source code review of a PaymentSplitter contract to be used in the Vesper platform. The objective of the audit was to evaluate the security of this smart contract.

The assessment was conducted on contracts from the Git repository at https://github.com/bloqpriv/bpools obtained on **Feb 9th**.

The following issues were identified during the assessment:

High Risk	Medium Risk	Low Risk
0	0	3

No high or medium risk vulnerabilities were found in the new smart contract. Three minor issues related to smart contracts development best practices are included in this report.

The following report details the tasks performed and the vulnerabilities found during this audit as well as several suggestions aimed at improving the overall code quality and warnings regarding potential issues.

2. Assessment

Vesper is a growing ecosystem of decentralized financial products, centered on ease-of-use.

The focus of this audit was the new `PaymentSplitter` smart contract.

The scope of the audit was limited to the latest version of the `strategies/PaymentSplitter.sol` Solidity source file with the following sha256hash:

`7660ab7cd23a89676fa14c7b9b03e8eaff820a70df753ac90eae820bfa771fab`

The goal of this new contract is to enable the distribution of funds (tokens and/or Ether) between a group of accounts, in a transparent way to the sender of those funds.

The split proportion is specified as a number of shares for each payee: both number of shares and payees are provided as parameters to the contract constructor. Neither the number of shares or the payees can be modified after the contract is created.

All payments received are stored in the `PaymentSplitter` contract, until the payment is requested for each of the payees. When this happens, the proportion of funds corresponding to the payee share is transferred to him. Separate functions need to be called in order to initiate the transfer of tokens (different calls for each token address are required) and also for Ether ones.

A new set of tests for the new contract is included in the repository, all of these 47 tests pass.

3. Summary of Findings

ID	Description	Risk	Fixed
VSP-021	Events not indexed in the PaymentSplitter contract	Low	✘
VSP-022	Outdated Solidity version	Low	✘
VSP-023	No test for PayeeAdded event	Low	✘

4. Findings

VSP-021	Events not indexed in the PaymentSplitter contract

Total Risk	Impact	Location
Low	Low	strategies/PaymentSplitter.sol
Fixed	Likelihood	
✘	Low	

Description

The following two events are emitted by the `PaymentSplitter` contract:

```
event PayeeAdded(address payee, uint256 share);
event PaymentReleased(address payee, address asset, uint256 tokens);
```

However, none of the event's parameters are declared indexed. Declaring a parameter as indexed causes the arguments to be treated as topics and facilitates searching and filtering events in off-chain components.

Recommendations

Declare up to three parameters as indexed in order to enable off-chain components to search and filter events efficiently.

VSP-022	Outdated Solidity version

Total Risk	Impact	Location
Low	Low	strategies/PaymentSplitter.sol
Fixed	Likelihood	
✘	Low	

Description

Currently the contract code specifies with the pragma statement below

```
pragma solidity ^0.6.0;
```

that it should be built with an outdated version of the Solidity compiler, which is not the latest production release. The latest versions have added additional warnings that can help to detect problems, solve bugs and enforce new rules to enhance security.

The `PaymentSplitter` contract pragma indicates that compiler version at least 0.6.0 can be used, but Solidity releases 0.7.x and 0.8.x are already available and should be preferred. A small risk exists that undocumented vulnerabilities are present in old releases of the compiler.

Recommendations

If possible, bump the compiler version to a newer version.

For more information on the use of the version pragma and how to handle the different versions accepted, see the following reference links:

- https://solidity.readthedocs.io/en/develop/layout-of-source-files.html#version-pragma
- https://docs.npmjs.com/misc/semver#versions
- http://solidity.readthedocs.io/en/develop/bugs.html

VSP-023	No test for PayeeAdded event

Total Risk	Impact	Location
Low	Low	test/payment-splitter.js
Fixed	**Likelihood**	
✘	Low	

Description

The `PayeeAdded` event emitted by the `PaymentSplitter` contract is never tested for in the project's test suite.

Recommendation

Although test coverage for this component is already very good, it is recommended to improve coverage by including a test for the missing event. The tests should verify that the event is emitted when expected and the event parameters are correct.

5. Conclusion

Overall, Coinspect did not find any high risk security vulnerabilities in the `PaymentSplitter` smart contract that would result in stolen or lost user funds. The code was properly documented and implemented according to the industry best practices. A few minor code quality issues were observed during this review and Coinspect recommends these are fixed in order to further improve the smart contract code.

6. Disclaimer

The information presented in this document is provided as is and without warranty. Vulnerability assessments are a “point in time” analysis and as such it is possible that something in the environment could have changed since the tests reflected in this report were run. This report should not be considered a perfect representation of the risks threatening the analysed system, networks, and applications.