**Avacash Contract Initial Audit Report**

[**Overview**](#_a3yxug87siiq) **2**

[Scope of Audit](#_la4tz2m9f16g) 2

[Check Vulnerabilities](#_fk11hakqdckb) 2

[**Techniques and Methods**](#_db8tp6ec385z) **3**

[Issue Categories](#_2p6a8gpqk15r) 4

[Number of security issues per severity.](#_2idbboidjlj7) 5

[**Introduction**](#_8d6evnfdp3d4) **5**

[**Issues Found – Code Review / Manual Testing**](#_jnyy7e310pds) **5**

[High Severity Issues](#_fpa7vs16wijg) 5

[Medium Severity Issues](#_ihaplnhuvava) 5

[Low Severity Issues](#_3xubtpht259r) 5

[Informational Issues](#_ndb9xm21sigw) 6

[**Slither Report**](#_l7nivtfmfq2q) **7**  
**Goerli Test Contracts 8**[**Test Transactions**](#_k1d9rfg29kl2) **8**  
[**Closing Summary**](#_dgf3621lwhzv) **11**

[**Disclaimer**](#_s8o85hoasaz2) **11**

## **Overview**

## **Scope of Audit**

The scope of this audit was to analyze and document the staking smart contract codebase for quality, security, and correctness.

## **Check Vulnerabilities**

* Re-entrancy
* Timestamp Dependence
* Gas Limit and Loops
* DoS with Block Gas Limit
* Transaction-Ordering Dependence
* Use of tx.origin
* Exception disorder
* Gasless send
* Balance equality
* Byte array
* Transfer forwards all gas
* ERC20 API violation
* Malicious libraries
* Compiler version not fixed
* Redundant fallback function
* Send instead of transfer
* Style guide violation
* Unchecked external call
* Unchecked math
* Unsafe type inference
* Implicit visibility level

## **Techniques and Methods**

Throughout the audit of smart contracts, care was taken to ensure:

* The overall quality of code.
* Use of best practices.
* Code documentation and comments match logic and expected behavior.
* Token distribution and calculations are as per the intended behavior mentioned in the whitepaper.
* Implementation of ERC-20 token standards.
* Efficient use of gas.
* Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods, and tools were used to review all the smart contracts.

## **Structural Analysis**

In this step, we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.

## **Static Analysis**

A static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step, a series of automated tools are used to test the security of smart contracts.

## **Code Review / Manual Analysis**

Manual Analysis or review of code was done to identify new vulnerabilities or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of the automated analysis were manually verified.

## **Gas Consumption**

In this step, we have checked the behavior of smart contracts in production. Checks were done to know how much gas gets consumed and the possibilities of optimization of code to reduce gas consumption.

## **Tools and Platforms used for Audit**

Remix IDE, Truffle, Truffle Team, Solhint, Mythril, Slither, Solidity statistic analysis.

## **Issue Categories**

Every issue in this report has been assigned to a severity level. There are four levels of severity, and each of them has been explained below.

**High Severity Issues**

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract’s performance or functionality, and we recommend these issues be fixed before moving to a live environment.

**Medium Severity Issues**

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems, and they should still be fixed.

**Low Severity Issues**

Low-level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

**Informational** **Issues**

These are four severity issues that indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

## 

## **Number of security issues per severity.**

| **TYPE** | **HIGH** | **MEDIUM** | **LOW** | **INFORMATIONAL** |
| --- | --- | --- | --- | --- |
| **Open** | **0** | **0** | **0** | **4** |
| **Acknowledged** | **0** | **0** | **0** | **0** |
| **Closed** | **0** | **0** | **0** | **0** |

## **Introduction**

During the period of **Feb 7, 2022, to Feb 13, 2022** - ImmuneBytes Team performed a security audit for **Avacash** **Finance** smart contract.

# **Issues Found – Code Review / Manual Testing**

## **High Severity Issues**

**None**

## **Medium Severity Issues**

**None**

## **Low Severity Issues**

**None**

## **Informational Issues**

### [INF1] Optimize sload

### We recommend to use the optimize contracts of [tornadoCash](https://github.com/tornadocash/tornado-core/commit/3ad634594eef7fb9e894033492110633614778ad) at the time of making an Avacash flash loan contract.

**Status**: **Open**

### [INF2] unused variable declaration

In AvacashFlashLoanProvider the unlocked variable is defined and not used  
 anywhere.We recommend removing the variable and improvising the  
 deployment cost.

**Status**: **Open**

### [INF3] unnecessary use of reentrancy wrapper

In changeFeeReceiver and changeFlashLoanFee has no external call and can’t  
 have reentrancy ever.

We recommend removing the wrapper and saving the transaction gas.

**Status**: **Open**

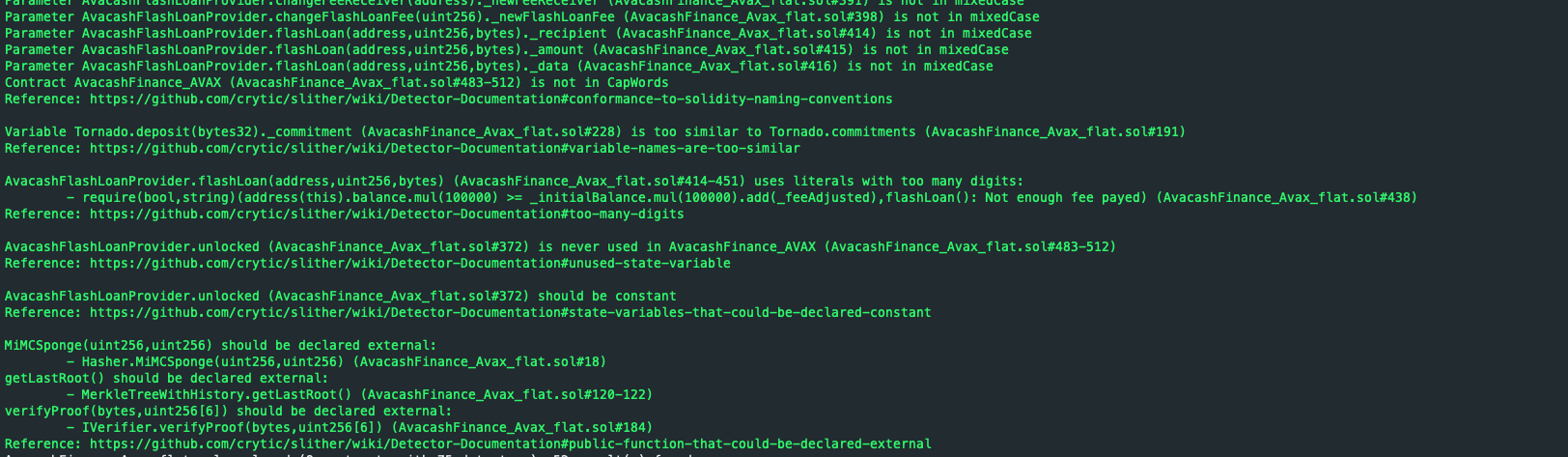
### [INF4] Missing comments and description:

Comments and Description of the methods and the variables are missing, it's  
 hard to read and understand the purpose of the variables and the methods in  
 context of the whole picture

**Recommendation**: Consider adding NatSpec format comments for the  
 comments and state variables

**Status**: **Open**

## **Slither Report**



All issues raised by slither are covered in the manual audit or are not relevant.

## **Goerli Test Contracts**

**Verifier:** [**0x6610849471166F11949069267cd46208dd61325C**](https://goerli.etherscan.io/address/0x6610849471166f11949069267cd46208dd61325c)

**AvacashFinance\_AVAX:** [**0x3545C9Ee895010191dB5d7fc43EC669062CFE3b8**](https://goerli.etherscan.io/address/0x3545C9Ee895010191dB5d7fc43EC669062CFE3b8)

**Borrower:** [**0xAFf7CB661b413a095Fffa393a2D4e47c44A533C6**](https://goerli.etherscan.io/address/0xAFf7CB661b413a095Fffa393a2D4e47c44A533C6)

## Test Transactions

**Payback to AvacashFinance\_AVAX (send 1 eth) - PASS**[**0x768c3cbbd93a029d19eef191b1124d461a5011f1ed6f2818b7fcb3893d9f48c0**](https://goerli.etherscan.io/tx/0x768c3cbbd93a029d19eef191b1124d461a5011f1ed6f2818b7fcb3893d9f48c0)

**changeFlashLoanFee(4) - PASS**

[**0x5fcbdf2a5062741792fce9c09632e25f38a0ddbc455ef4e47f7a3fccf87d1e07**](https://goerli.etherscan.io/tx/0x5fcbdf2a5062741792fce9c09632e25f38a0ddbc455ef4e47f7a3fccf87d1e07)

***Thief case -* PASS**

[**0x1db1293546e2f62d61a035c637748db351a070901c9b5c7d78dde31fd95284c3**](https://goerli.etherscan.io/tx/0x1db1293546e2f62d61a035c637748db351a070901c9b5c7d78dde31fd95284c3)

**Zero fee() - PASS**

**setLoanFeeToZero - true**[**0xb6198229de3f1f2e5965a39d33bd565153e8654ce2f01286cc41c02cc427090c**](https://goerli.etherscan.io/tx/0xb6198229de3f1f2e5965a39d33bd565153e8654ce2f01286cc41c02cc427090c)

**Successfully execute the flash loan**

[**0x4a9d7c5afb89b1bcf10176a20c8a829a9268f26ffcffa349a403a818e701f961**](https://goerli.etherscan.io/tx/0x4a9d7c5afb89b1bcf10176a20c8a829a9268f26ffcffa349a403a818e701f961)

**ChangeFeeReceiver - PASS**

**Current Flash loan receiver can change \* -** [**0xb8d86eadf24638a898c8b6cfd481f64e234944adf5559f499e55a7df5a543024**](https://goerli.etherscan.io/tx/0xb8d86eadf24638a898c8b6cfd481f64e234944adf5559f499e55a7df5a543024)

[**0x1fb515fac8e53f0181161fac31e54bde83753f589ff9bdd1c679d29dd0c54ff2**](https://goerli.etherscan.io/tx/0x1fb515fac8e53f0181161fac31e54bde83753f589ff9bdd1c679d29dd0c54ff2)

**changeFlashLoanFee(10000) - PASS**

[**0xfd1ec77c72ef979dbf34a714253106ebf1bef44349692060c03348fbc23c68c7**](https://goerli.etherscan.io/tx/0xfd1ec77c72ef979dbf34a714253106ebf1bef44349692060c03348fbc23c68c7)

[**0x782cafe19c322bfd37ecde62e5b8a54080f73133317180b6eef43da3b5655c0c**](https://goerli.etherscan.io/tx/0x782cafe19c322bfd37ecde62e5b8a54080f73133317180b6eef43da3b5655c0c)

**Borrower balance decreased by 10 wei (fee transferred)**

## **Closing Summary**

Overall, smart contracts are well written and adhere to guidelines.  
No major severity is detected. We have listed some informative issues in the contracts to make it more optimized.  
We recommend that resolving the informative issues helps to decrease the gas cost and make the contract lighter.

Apart from the Audit we suggest the **Avacash Finance** team to write more Unit Test cases and maintain the coverage for around 100%.

## **Disclaimer**

ImmuneBytes audit is not a security warranty, investment advice, or an endorsement of **Avacash Finance**. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.