Avacash Contract Initial Audit Report

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 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

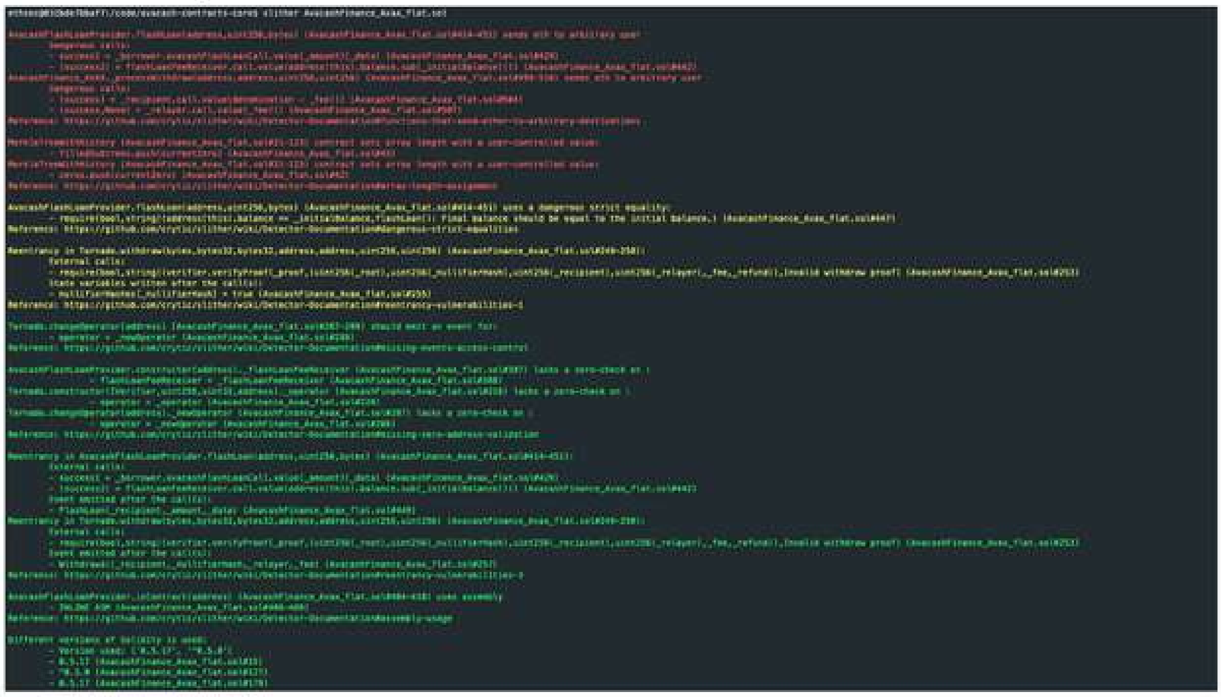
- **[INF4] Missing comments and description:**

  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

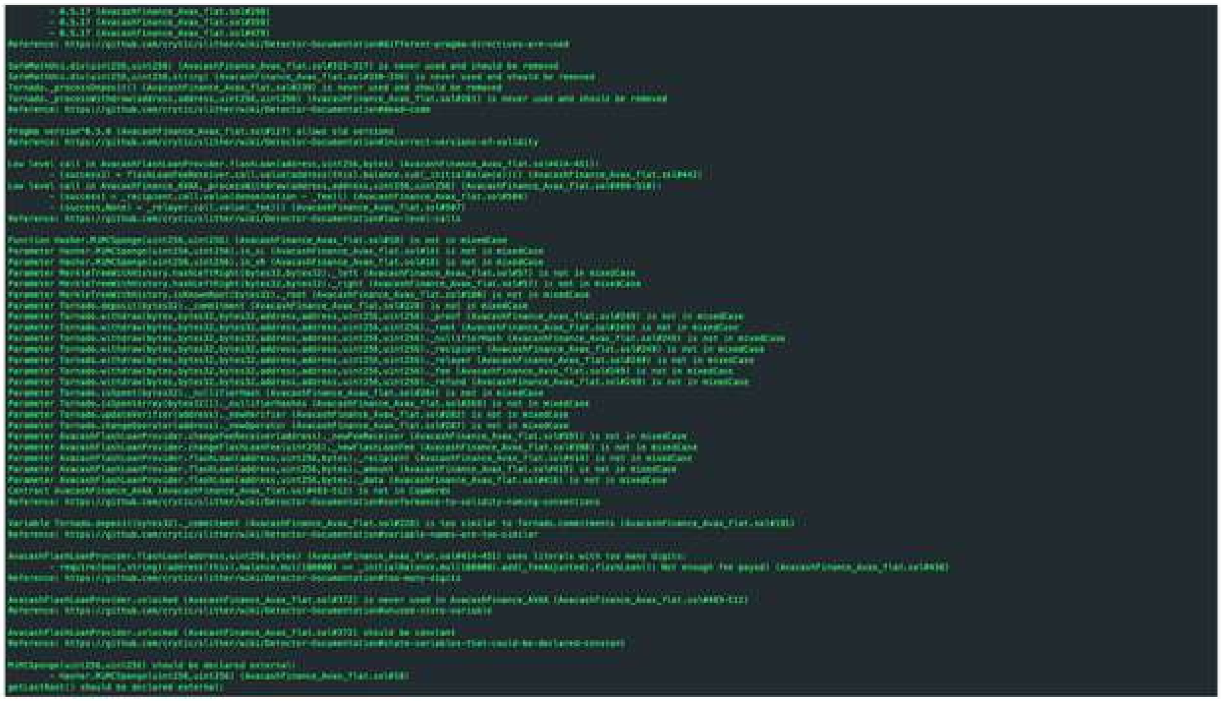
  Recommendation: Consider adding NatSpec format comments for the comments and state variables

  Status: Open

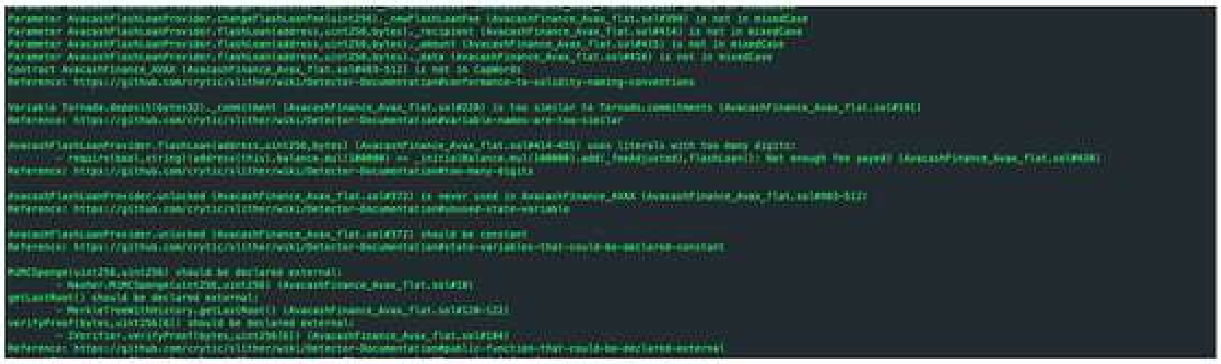
Slither Report



```
[illegible]:~/code/avacash-contracts-core$ slither AvacashFinance_Avax_flat.sol

AvacashFlashLoanProvider.flashLoan(address,uint256,bytes) (AvacashFinance_Avax_flat.sol#414-451) sends eth to arbitrary user
        Dangerous calls:
        - success1 = _borrower.avacashFlashLoanCall.value(_amount)(_data) (AvacashFinance_Avax_flat.sol#429)
        - (success2) = flashLoanFeeReceiver.call.value(address(this).balance.sub(_initialBalance))() (AvacashFinance_Avax_flat.sol#442)
AvacashFinance_AVAX._processWithdraw(address,address,uint256,uint256) (AvacashFinance_Avax_flat.sol#[illegible]) sends eth to arbitrary user
        Dangerous calls:
        - (success) = _recipient.call.value(denomination - _fee)() (AvacashFinance_Avax_flat.sol#[illegible])
        - (success,None) = _relayer.call.value(_fee)() (AvacashFinance_Avax_flat.sol#[illegible])
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations

MerkleTreeWithHistory (AvacashFinance_Avax_flat.sol#[illegible]) contract sets array length with a user-controlled value:
        - filledSubtrees.push(currentZero) (AvacashFinance_Avax_flat.sol#[illegible])
MerkleTreeWithHistory (AvacashFinance_Avax_flat.sol#[illegible]) contract sets array length with a user-controlled value:
        - zeros.push(currentZero) (AvacashFinance_Avax_flat.sol#[illegible])
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#array-length-assignment

AvacashFlashLoanProvider.flashLoan(address,uint256,bytes) (AvacashFinance_Avax_flat.sol#414-451) uses a dangerous strict equality:
        - require(bool,string)(address(this).balance == _initialBalance,flashLoan(): Final balance should be equal to the initial balance.) (AvacashFinance_Avax_flat.sol#447)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dangerous-strict-equalities

Reentrancy in Tornado.withdraw(bytes,bytes32,bytes32,address,address,uint256,uint256) (AvacashFinance_Avax_flat.sol#240-258):
        External calls:
        - require(bool,string)(verifier.verifyProof(_proof,[uint256(_root),uint256(_nullifierHash),uint256(_recipient),uint256(_relayer),_fee,_refund]),Invalid withdraw proof) (AvacashFinance_Avax_flat.sol#253)
        State variables written after the call(s):
        - nullifierHashes[_nullifierHash] = true (AvacashFinance_Avax_flat.sol#255)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-1

Tornado.changeOperator(address) (AvacashFinance_Avax_flat.sol#[illegible]) should emit an event for:
        - operator = _newOperator (AvacashFinance_Avax_flat.sol#[illegible])
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

AvacashFlashLoanProvider.constructor(address)._flashLoanFeeReceiver (AvacashFinance_Avax_flat.sol#[illegible]) lacks a zero-check on :
                - flashLoanFeeReceiver = _flashLoanFeeReceiver (AvacashFinance_Avax_flat.sol#[illegible])
Tornado.constructor(IVerifier,uint256,uint32,address)._operator (AvacashFinance_Avax_flat.sol#[illegible]) lacks a zero-check on :
                - operator = _operator (AvacashFinance_Avax_flat.sol#[illegible])
Tornado.changeOperator(address)._newOperator (AvacashFinance_Avax_flat.sol#[illegible]) lacks a zero-check on :
                - operator = _newOperator (AvacashFinance_Avax_flat.sol#[illegible])
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation

Reentrancy in AvacashFlashLoanProvider.flashLoan(address,uint256,bytes) (AvacashFinance_Avax_flat.sol#414-451):
        External calls:
        - success1 = _borrower.avacashFlashLoanCall.value(_amount)(_data) (AvacashFinance_Avax_flat.sol#429)
        - (success2) = flashLoanFeeReceiver.call.value(address(this).balance.sub(_initialBalance))() (AvacashFinance_Avax_flat.sol#442)
        Event emitted after the call(s):
        - FlashLoan(_recipient,_amount,_data) (AvacashFinance_Avax_flat.sol#[illegible])
Reentrancy in Tornado.withdraw(bytes,bytes32,bytes32,address,address,uint256,uint256) (AvacashFinance_Avax_flat.sol#240-258):
        External calls:
        - require(bool,string)(verifier.verifyProof(_proof,[uint256(_root),uint256(_nullifierHash),uint256(_recipient),uint256(_relayer),_fee,_refund]),Invalid withdraw proof) (AvacashFinance_Avax_flat.sol#253)
        Event emitted after the call(s):
        - Withdrawal(_recipient,_nullifierHash,_relayer,_fee) (AvacashFinance_Avax_flat.sol#257)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#reentrancy-vulnerabilities-3

AvacashFlashLoanProvider.isContract(address) (AvacashFinance_Avax_flat.sol#[illegible]) uses assembly
        - INLINE ASM (AvacashFinance_Avax_flat.sol#[illegible])
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.5.17', '^0.5.8']
        - 0.5.17 (AvacashFinance_Avax_flat.sol#15)
        - ^0.5.8 (AvacashFinance_Avax_flat.sol#[illegible])
        - 0.5.17 (AvacashFinance_Avax_flat.sol#[illegible])
```



```
        - 0.5.17 (AvacashFinance_Avax_flat.sol#[illegible])
        - 0.5.17 (AvacashFinance_Avax_flat.sol#[illegible])
        - 0.5.17 (AvacashFinance_Avax_flat.sol#[illegible])
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

SafeMath.div(uint256,uint256) (AvacashFinance_Avax_flat.sol#[illegible]) is never used and should be removed
SafeMath.div(uint256,uint256,string) (AvacashFinance_Avax_flat.sol#[illegible]) is never used and should be removed
Tornado._processDeposit() (AvacashFinance_Avax_flat.sol#[illegible]) is never used and should be removed
Tornado._processWithdraw(address,address,uint256,uint256) (AvacashFinance_Avax_flat.sol#[illegible]) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.5.8 (AvacashFinance_Avax_flat.sol#[illegible]) allows old versions
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#incorrect-versions-of-solidity

Low level call in AvacashFlashLoanProvider.flashLoan(address,uint256,bytes) (AvacashFinance_Avax_flat.sol#414-451):
        - (success2) = flashLoanFeeReceiver.call.value(address(this).balance.sub(_initialBalance))() (AvacashFinance_Avax_flat.sol#442)
Low level call in AvacashFinance_AVAX._processWithdraw(address,address,uint256,uint256) (AvacashFinance_Avax_flat.sol#[illegible]):
        - (success) = _recipient.call.value(denomination - _fee)() (AvacashFinance_Avax_flat.sol#[illegible])
        - (success,None) = _relayer.call.value(_fee)() (AvacashFinance_Avax_flat.sol#[illegible])
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function Hasher.MiMCSponge(uint256,uint256) (AvacashFinance_Avax_flat.sol#[illegible]) is not in mixedCase
Parameter Hasher.MiMCSponge(uint256,uint256).in_xL (AvacashFinance_Avax_flat.sol#[illegible]) is not in mixedCase
Parameter Hasher.MiMCSponge(uint256,uint256).in_xR (AvacashFinance_Avax_flat.sol#[illegible]) is not in mixedCase
Parameter MerkleTreeWithHistory.hashLeftRight(bytes32,bytes32)._left (AvacashFinance_Avax_flat.sol#[illegible]) is not in mixedCase
Parameter MerkleTreeWithHistory.hashLeftRight(bytes32,bytes32)._right (AvacashFinance_Avax_flat.sol#[illegible]) is not in mixedCase
Parameter MerkleTreeWithHistory.isKnownRoot(bytes32)._root (AvacashFinance_Avax_flat.sol#[illegible]) is not in mixedCase
Parameter Tornado.deposit(bytes32)._commitment (AvacashFinance_Avax_flat.sol#[illegible]) is not in mixedCase
Parameter Tornado.withdraw(bytes,bytes32,bytes32,address,address,uint256,uint256)._proof (AvacashFinance_Avax_flat.sol#[illegible]) is not in mixedCase
Parameter Tornado.withdraw(bytes,bytes32,bytes32,address,address,uint256,uint256)._root (AvacashFinance_Avax_flat.sol#[illegible]) is not in mixedCase
Parameter Tornado.withdraw(bytes,bytes32,bytes32,address,address,uint256,uint256)._nullifierHash (AvacashFinance_Avax_flat.sol#[illegible]) is not in mixedCase
Parameter Tornado.withdraw(bytes,bytes32,bytes32,address,address,uint256,uint256)._recipient (AvacashFinance_Avax_flat.sol#[illegible]) is not in mixedCase
Parameter Tornado.withdraw(bytes,bytes32,bytes32,address,address,uint256,uint256)._relayer (AvacashFinance_Avax_flat.sol#[illegible]) is not in mixedCase
Parameter Tornado.withdraw(bytes,bytes32,bytes32,address,address,uint256,uint256)._fee (AvacashFinance_Avax_flat.sol#[illegible]) is not in mixedCase
Parameter Tornado.withdraw(bytes,bytes32,bytes32,address,address,uint256,uint256)._refund (AvacashFinance_Avax_flat.sol#[illegible]) is not in mixedCase
Parameter Tornado.isSpent(bytes32)._nullifierHash (AvacashFinance_Avax_flat.sol#[illegible]) is not in mixedCase
Parameter Tornado.isSpentArray(bytes32[])._nullifierHashes (AvacashFinance_Avax_flat.sol#[illegible]) is not in mixedCase
Parameter Tornado.updateVerifier(address)._newVerifier (AvacashFinance_Avax_flat.sol#[illegible]) is not in mixedCase
Parameter Tornado.changeOperator(address)._newOperator (AvacashFinance_Avax_flat.sol#[illegible]) is not in mixedCase
Parameter AvacashFlashLoanProvider.changeFeeReceiver(address)._newFeeReceiver (AvacashFinance_Avax_flat.sol#[illegible]) is not in mixedCase
Parameter AvacashFlashLoanProvider.changeFlashLoanFee(uint256)._newFlashLoanFee (AvacashFinance_Avax_flat.sol#[illegible]) is not in mixedCase
Parameter AvacashFlashLoanProvider.flashLoan(address,uint256,bytes)._recipient (AvacashFinance_Avax_flat.sol#414) is not in mixedCase
Parameter AvacashFlashLoanProvider.flashLoan(address,uint256,bytes)._amount (AvacashFinance_Avax_flat.sol#415) is not in mixedCase
Parameter AvacashFlashLoanProvider.flashLoan(address,uint256,bytes)._data (AvacashFinance_Avax_flat.sol#416) is not in mixedCase
Contract AvacashFinance_AVAX (AvacashFinance_Avax_flat.sol#[illegible]) is not in CapWords
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable Tornado.deposit(bytes32)._commitment (AvacashFinance_Avax_flat.sol#[illegible]) is too similar to Tornado.commitments (AvacashFinance_Avax_flat.sol#[illegible])
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

AvacashFlashLoanProvider.flashLoan(address,uint256,bytes) (AvacashFinance_Avax_flat.sol#414-451) uses literals with too many digits:
        - require(bool,string)(address(this).balance.mul(100000) >= _initialBalance.mul(100000).add(_feeAdjusted),flashLoan(): Not enough fee payed) (AvacashFinance_Avax_flat.sol#[illegible])
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

AvacashFlashLoanProvider.unlocked (AvacashFinance_Avax_flat.sol#[illegible]) is never used in AvacashFinance_AVAX (AvacashFinance_Avax_flat.sol#[illegible])
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

AvacashFlashLoanProvider.unlocked (AvacashFinance_Avax_flat.sol#[illegible]) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

MiMCSponge(uint256,uint256) should be declared external:
        - Hasher.MiMCSponge(uint256,uint256) (AvacashFinance_Avax_flat.sol#[illegible])
getLastRoot() should be declared external:
```



```
Parameter AvacashFlashLoanProvider.changeFlashLoanFee(uint256)._newFlashLoanFee (AvacashFinance_Avax_flat.sol#[illegible]) is not in mixedCase
Parameter AvacashFlashLoanProvider.flashLoan(address,uint256,bytes)._recipient (AvacashFinance_Avax_flat.sol#414) is not in mixedCase
Parameter AvacashFlashLoanProvider.flashLoan(address,uint256,bytes)._amount (AvacashFinance_Avax_flat.sol#415) is not in mixedCase
Parameter AvacashFlashLoanProvider.flashLoan(address,uint256,bytes)._data (AvacashFinance_Avax_flat.sol#416) is not in mixedCase
Contract AvacashFinance_AVAX (AvacashFinance_Avax_flat.sol#483-512) is not in CapWords
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions

Variable Tornado.deposit(bytes32)._commitment (AvacashFinance_Avax_flat.sol#228) is too similar to Tornado.commitments (AvacashFinance_Avax_flat.sol#191)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#variable-names-are-too-similar

AvacashFlashLoanProvider.flashLoan(address,uint256,bytes) (AvacashFinance_Avax_flat.sol#414-435) uses literals with too many digits:
        - require(bool,string)(address(this).balance.mul(100000) >= _initialBalance.mul(100000).add(_feeAdjusted),FlashLoan: Not enough fee payed) (AvacashFinance_Avax_flat.sol#430)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#too-many-digits

AvacashFlashLoanProvider.unlocked (AvacashFinance_Avax_flat.sol#372) is never used in AvacashFinance_AVAX (AvacashFinance_Avax_flat.sol#483-512)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-state-variable

AvacashFlashLoanProvider.unlocked (AvacashFinance_Avax_flat.sol#372) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant

MiMCSponge(uint256,uint256) should be declared external:
        - Hasher.MiMCSponge(uint256,uint256) (AvacashFinance_Avax_flat.sol#18)
getLastRoot() should be declared external:
        - MerkleTreeWithHistory.getLastRoot() (AvacashFinance_Avax_flat.sol#120-122)
verifyProof(bytes,uint256[6]) should be declared external:
        - IVerifier.verifyProof(bytes,uint256[6]) (AvacashFinance_Avax_flat.sol#184)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
```

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

Goerli Test Contracts

Verifier:	0x6610849471166F11949069267cd46208dd61325C
AvacashFinance_AVAX:	0x3545C9Ee895010191dB5d7fc43EC669062CFE3b8
Borrower:	0xAFf7CB661b413a095Fffa393a2D4e47c44A533C6

Test Transactions

Payback to AvacashFinance_AVAX (send 1 eth) - PASS
0x768c3cbbd93a029d19eef191b1124d461a5011f1ed6f2818b7fcb3893d9f48c0

changeFlashLoanFee(4) - PASS
0x5fcbdf2a5062741792fce9c09632e25f38a0ddbc455ef4e47f7a3fccf87d1e07

Thief case - PASS
0x1db1293546e2f62d61a035c637748db351a070901c9b5c7d78dde31fd95284c3

Zero fee() - PASS
setLoanFeeToZero - true
0xb6198229de3f1f2e5965a39d33bd565153e8654ce2f01286cc41c02cc427090c
Successfully execute the flash loan
0x4a9d7c5afb89b1bcf10176a20c8a829a9268f26ffcffa349a403a818e701f961

ChangeFeeReceiver - PASS
Current Flash loan receiver can change * -
0xb8d86eadf24638a898c8b6cfd481f64e234944adf5559f499e55a7df5a543024

0x1fb515fac8e53f0181161fac31e54bde83753f589ff9bdd1c679d29dd0c54ff2

changeFlashLoanFee(10000) - PASS
0xfd1ec77c72ef979dbf34a714253106ebf1bef44349692060c03348fbc23c68c7

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.