Avacash Finance

Smart Contract Audit
Final Report

March 08, 2022

Introduction

1. About Avacash Finance

Avacash.Finance is a fully decentralized protocol that automatically invests your assets privately in different DeFi protocols on the Avalanche Blockchain.

And Beyond:
Avacash.Finance development team will be always developing new proposals and other DeFi protocols integrations. Using the $CASH governance token, the Avacash.Finance protocol will be alive forever as a Decentralized Autonomous Organization (DAO).

Visit https://avacash.finance/ to know more about it.

2. About ImmuneBytes

ImmuneBytes is a security start-up to provide professional services in the blockchain space. The team has hands-on experience in conducting smart contract audits, penetration testing, and security consulting. ImmuneBytes's security auditors have worked on various A-league projects and have a great understanding of DeFi projects like AAVE, Compound, 0x Protocol, Uniswap, dydx.

The team has been able to secure 105+ blockchain projects by providing security services on different frameworks. ImmuneBytes team helps start-ups with a detailed analysis of the system ensuring security and managing the overall project.

Visit http://immunebytes.com/ to know more about the services.

Documentation Details

The Avacash Finance team has provided the following doc for the purpose of audit:

1. https://avacash.medium.com/

Audit Process & Methodology

ImmuneBytes team has performed thorough testing of the project starting with analyzing the code design patterns in which we reviewed the smart contract architecture to ensure it is structured and safe use of third-party smart contracts and libraries.

Our team then performed a formal line-by-line inspection of the Smart Contract in order to find any potential issues like Signature Replay Attacks, Unchecked External Calls, External Contract Referencing, Variable Shadowing, Race conditions, Transaction-ordering dependence, timestamp dependence, DoS attacks, and others.

In the Unit testing phase, we run unit tests written by the developer in order to verify the functions work as intended. In Automated Testing, we tested the Smart Contract with our in-house developed tools to identify vulnerabilities and security flaws.

The code was audited by a team of independent auditors which includes -

1. Testing the functionality of the Smart Contract to determine proper logic has been followed throughout.
2. Analyzing the complexity of the code by thorough, manual review of the code, line-by-line.
3. Deploying the code on testnet using multiple clients to run live tests.
4. Analyzing failure preparations to check how the Smart Contract performs in case of bugs and vulnerabilities.
5. Checking whether all the libraries used in the code are on the latest version.
6. Analyzing the security of the on-chain data.

Audit Details

- Project Name: Avacash Finance
- Contracts Name: AvacashFinance_AVAX.sol, AvacashFlashLoanProvider.sol, ERC20Tornado.sol, ETHTornado.sol, MerkleTreeWithHistory.sol, Migrations.sol, Tornado.sol, Verifier.sol
- Languages: Solidity(Smart contract), Typescript (Unit Testing)
- Github commits for initial audit: 298d5784eff755fe58edb63e4477d555b351733b
- Github commits for final audit: 50e5adabdb2af29167ecd53da25a748986bd836e
- Platforms and Tools: Remix IDE, Truffle, Truffle Team, Ganache, Solhint, VScode, Contract Library, Slither, SmartCheck

Audit Goals

The focus of the audit was to verify that the smart contract system is secure, resilient, and working according to its specifications. The audit activities can be grouped into the following three categories:

1. Security: Identifying security-related issues within each contract and within the system of contracts.
2. Sound Architecture: Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices.
3. Code Correctness and Quality: A full review of the contract source code. The primary areas of focus include
    a. Correctness
    b. Readability
    c. Sections of code with high complexity
    d. Quantity and quality of test coverage

Security Level Reference

Every issue in this report were assigned a severity level from the following:

High severity issues will bring problems and should be fixed.

Medium severity issues could potentially bring problems and should eventually be fixed.

Low severity issues are minor details and warnings that can remain unfixed but would be better fixed at some point in the future.

Issues	High	Medium	Low
Open	-	-	-
Closed	-	-	-

High Severity Issues

No issues were found.

Medium Severity Issues

No issues were found.

Low Severity Issues

No issues were found.

Recommendation / Informational

1. **Optimize sload**

   We recommend using the optimized contracts of tornadoCash at the time of making an Avacash flash loan contract.

   Amended (March 08th, 2022): The issue was fixed by the **Avacash Finance** team and is no longer present.

- **unused variable declaration**

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

  Amended (March 08th, 2022): The issue was fixed by the **Avacash Finance** team and is no longer present.

- **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.

  Acknowledged (March 08th, 2022): The issue was acknowledged by the **Avacash Finance** team.

- **Missing comments and descriptions:**

  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 the context of the whole picture

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

  Amended (March 08th, 2022): The issue was fixed by the **Avacash Finance** team and is no longer present.

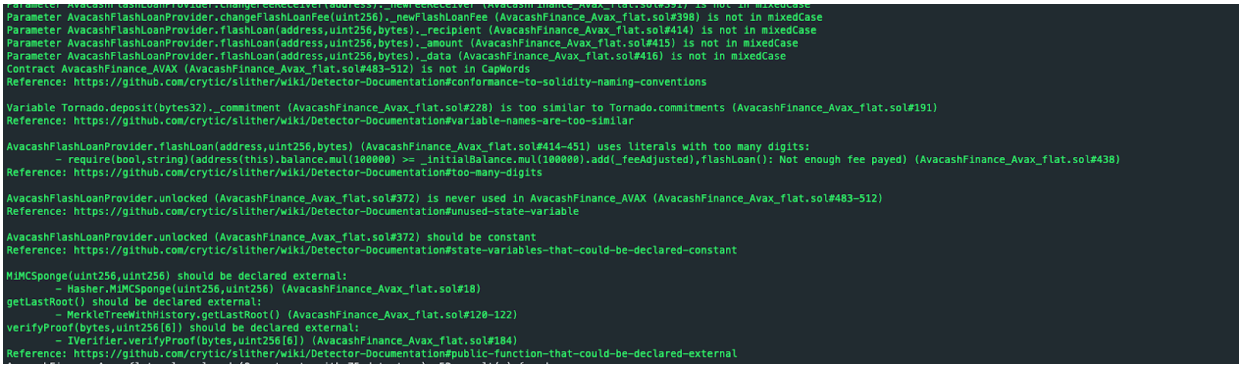
Automated Test Result

Slither

```
ethsec@032bde7bbaf7:/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#499-510) sends eth to arbitrary user
        Dangerous calls:
        - (success) = _recipient.call.value(denomination - _fee)() (AvacashFinance_Avax_flat.sol#504)
        - (success,None) = _relayer.call.value(_fee)() (AvacashFinance_Avax_flat.sol#507)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#functions-that-send-ether-to-arbitrary-destinations

MerkleTreeWithHistory (AvacashFinance_Avax_flat.sol#21-123) contract sets array length with a user-controlled value:
        - filledSubtrees.push(currentZero) (AvacashFinance_Avax_flat.sol#43)
MerkleTreeWithHistory (AvacashFinance_Avax_flat.sol#21-123) contract sets array length with a user-controlled value:
        - zeros.push(currentZero) (AvacashFinance_Avax_flat.sol#42)
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#249-258):
        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#287-289) should emit an event for:
        - operator = _newOperator (AvacashFinance_Avax_flat.sol#288)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-events-access-control

AvacashFlashLoanProvider.constructor(address)._flashLoanFeeReceiver (AvacashFinance_Avax_flat.sol#387) lacks a zero-check on :
                - flashLoanFeeReceiver = _flashLoanFeeReceiver (AvacashFinance_Avax_flat.sol#388)
Tornado.constructor(IVerifier,uint256,uint32,address)._operator (AvacashFinance_Avax_flat.sol#216) lacks a zero-check on :
                - operator = _operator (AvacashFinance_Avax_flat.sol#220)
Tornado.changeOperator(address)._newOperator (AvacashFinance_Avax_flat.sol#287) lacks a zero-check on :
                - operator = _newOperator (AvacashFinance_Avax_flat.sol#288)
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#449)
Reentrancy in Tornado.withdraw(bytes,bytes32,bytes32,address,address,uint256,uint256) (AvacashFinance_Avax_flat.sol#249-258):
        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#404-410) uses assembly
        - INLINE ASM (AvacashFinance_Avax_flat.sol#406-408)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage

Different versions of Solidity is used:
        - Version used: ['0.5.17', '^0.5.0']
        - 0.5.17 (AvacashFinance_Avax_flat.sol#15)
        - ^0.5.0 (AvacashFinance_Avax_flat.sol#127)
        - 0.5.17 (AvacashFinance_Avax_flat.sol#179)
```

```
        - 0.5.17 (AvacashFinance_Avax_flat.sol#298)
        - 0.5.17 (AvacashFinance_Avax_flat.sol#359)
        - 0.5.17 (AvacashFinance_Avax_flat.sol#479)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#different-pragma-directives-are-used

SafeMathUni.div(uint256,uint256) (AvacashFinance_Avax_flat.sol#315-317) is never used and should be removed
SafeMathUni.div(uint256,uint256,string) (AvacashFinance_Avax_flat.sol#330-336) is never used and should be removed
Tornado._processDeposit() (AvacashFinance_Avax_flat.sol#239) is never used and should be removed
Tornado._processWithdraw(address,address,uint256,uint256) (AvacashFinance_Avax_flat.sol#261) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code

Pragma version^0.5.0 (AvacashFinance_Avax_flat.sol#127) 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#499-510):
        - (success) = _recipient.call.value(denomination - _fee)() (AvacashFinance_Avax_flat.sol#504)
        - (success,None) = _relayer.call.value(_fee)() (AvacashFinance_Avax_flat.sol#507)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls

Function Hasher.MiMCSponge(uint256,uint256) (AvacashFinance_Avax_flat.sol#18) is not in mixedCase
Parameter Hasher.MiMCSponge(uint256,uint256).in_xL (AvacashFinance_Avax_flat.sol#18) is not in mixedCase
Parameter Hasher.MiMCSponge(uint256,uint256).in_xR (AvacashFinance_Avax_flat.sol#18) is not in mixedCase
Parameter MerkleTreeWithHistory.hashLeftRight(bytes32,bytes32)._left (AvacashFinance_Avax_flat.sol#57) is not in mixedCase
Parameter MerkleTreeWithHistory.hashLeftRight(bytes32,bytes32)._right (AvacashFinance_Avax_flat.sol#57) is not in mixedCase
Parameter MerkleTreeWithHistory.isKnownRoot(bytes32)._root (AvacashFinance_Avax_flat.sol#100) is not in mixedCase
Parameter Tornado.deposit(bytes32)._commitment (AvacashFinance_Avax_flat.sol#228) is not in mixedCase
Parameter Tornado.withdraw(bytes,bytes32,bytes32,address,address,uint256,uint256)._proof (AvacashFinance_Avax_flat.sol#249) is not in mixedCase
Parameter Tornado.withdraw(bytes,bytes32,bytes32,address,address,uint256,uint256)._root (AvacashFinance_Avax_flat.sol#249) is not in mixedCase
Parameter Tornado.withdraw(bytes,bytes32,bytes32,address,address,uint256,uint256)._nullifierHash (AvacashFinance_Avax_flat.sol#249) is not in mixedCase
Parameter Tornado.withdraw(bytes,bytes32,bytes32,address,address,uint256,uint256)._recipient (AvacashFinance_Avax_flat.sol#249) is not in mixedCase
Parameter Tornado.withdraw(bytes,bytes32,bytes32,address,address,uint256,uint256)._relayer (AvacashFinance_Avax_flat.sol#249) is not in mixedCase
Parameter Tornado.withdraw(bytes,bytes32,bytes32,address,address,uint256,uint256)._fee (AvacashFinance_Avax_flat.sol#249) is not in mixedCase
Parameter Tornado.withdraw(bytes,bytes32,bytes32,address,address,uint256,uint256)._refund (AvacashFinance_Avax_flat.sol#249) is not in mixedCase
Parameter Tornado.isSpent(bytes32)._nullifierHash (AvacashFinance_Avax_flat.sol#264) is not in mixedCase
Parameter Tornado.isSpentArray(bytes32[])._nullifierHashes (AvacashFinance_Avax_flat.sol#269) is not in mixedCase
Parameter Tornado.updateVerifier(address)._newVerifier (AvacashFinance_Avax_flat.sol#282) is not in mixedCase
Parameter Tornado.changeOperator(address)._newOperator (AvacashFinance_Avax_flat.sol#287) is not in mixedCase
Parameter AvacashFlashLoanProvider.changeFeeReceiver(address)._newFeeReceiver (AvacashFinance_Avax_flat.sol#391) is not in mixedCase
Parameter AvacashFlashLoanProvider.changeFlashLoanFee(uint256)._newFlashLoanFee (AvacashFinance_Avax_flat.sol#398) 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#414-451) 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#438)
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:
```

This audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.



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)

Concluding Remarks

While conducting the audits of the Avacash Finance smart contract, it was observed that the contracts contain only recommendations/informational pointers.

Disclaimer

ImmuneBytes's audit does not provide a security or correctness guarantee of the audited smart contract. Securing smart contracts is a multistep process, therefore running a bug bounty program as a complement to this audit is strongly recommended.

Our team does not endorse the Avacash Finance platform or its product nor this audit is investment advice.
Notes:

- Please make sure contracts deployed on the mainnet are the ones audited.
- Check for the code refactor by the team on critical issues.

ImmuneBytes