CERTIK

Security Assessment

FilDA

Jun 1st, 2021

Table of Contents

Summary

This report has been prepared for FilDa smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross-referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	FilDA
Description	Flash Loans are special uncollateralised loans that allow the borrowing of an asset, as long as the borrowed amount (and a fee) is returned before the end of the transaction.
Platform	Heco
Language	Solidity
Codebase	• https://github.com/fildaio/FlashLoan • https://github.com/fildaio/FlashLoanAdapter
Commits	• 47219fe5934393a0527b83b5be41f75d75e397b4 • 70a4b43a3ab76cfd5f76e52cbd163df76bbfcc0c • a418c518ba62997a71462d3c628c279b9c566f5b • ebd9052f7456535e90a704a4ccdb220a647fb9f6 • 1d8e0a7ef00b9c4280a95e2e8e91e6e834ae07fe

Audit Summary

Delivery Date	Jun 01, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	FlashLoan, FlashLoanAdapter

Vulnerability Summary

Total Issues	6
Critical	0
Major	0
Medium	0
Minor	1
Informational	5
Discussion	0

Audit Scope

ID	file	SHA256 Checksum
FLF	FlashLoan/FlashLoan.sol	f9a74a6483d32cf7e2935f3deb24f7f0c5fe1590e2d04f8d0be21a8f9793568e
FLR	FlashLoan/FlashLoanReceiverBase.sol	fe9d80e80a3e1c46d2affdab564eea3daebb536a83321357579eb3a2da61f837
FLS	FlashLoan/FlashLoanStorage.sol	daa1557ac8a36f95470b852b09ea5a21525baf7c0f52117a6b8dfc25c3bebfcc
GFL	FlashLoan/Governable.sol	3a91976f71b84f54ff43856b0833191d5dc561fa14af5cd8fdc71a28654b10ef
IFL	FlashLoan/IFlashLoan.sol	c9610743b9458f704179ad296ed7f76db5fb118ea8e4bbd03e8bfb6af2441d60
IFR	FlashLoan/IFlashLoanReceiver.sol	bec9be3383aa599682aba3a26d143e966ed95f0033d03c9730656b00a72e0c7e
MFL	FlashLoan/Migrations.sol	8a6b38936c738a0e612391ee231f39352cc8878f4a5b41c05f0895fb662b3fd6
FLD	FlashLoan/dependency.sol	79087b32295fae36bc2aad879e211e233e65513d61204ccbb48ac48e88c23f8d
BAF	FlashLoanAdapter/BaseAdapter.sol	750afaf87b16c4b12b057a382ac0ecbd35bfd6121fda2dc25dbbf9ee880fcbc4
FMF	FlashLoanAdapter/FeeManager.sol	5f60c64994d10f1b1a7b254ac3bf1bfd0313f8a9e755ab587770028d3fcafaa1
GFA	FlashLoanAdapter/Governable.sol	3a91976f71b84f54ff43856b0833191d5dc561fa14af5cd8fdc71a28654b10ef
LSF	FlashLoanAdapter/LiquiditySwap.sol	02949c112add72bf1d4b0f1d7cfca1b1a09336b1883ce4a1f96ffbdfd182a224
MFA	FlashLoanAdapter/Migrations.sol	8a6b38936c738a0e612391ee231f39352cc8878f4a5b41c05f0895fb662b3fd6
RLF	FlashLoanAdapter/RepayLoan.sol	30fc679d6b47eb86d5ce4c60fe251d8ea9fb0e3e6beea11d97c6e9dc438b26a8
RPF	FlashLoanAdapter/RewardPool.sol	b6fb626c7d4c94343578328e44949efc4bd0d566cff6cccf73b62c22800ef924
WET	FlashLoanAdapter/WETH.sol	16308e34952c4385bdcd86fada6621b8e0a7d89d0894cce769fcb19fa388f26a

ID	file	SHA256 Checksum
FLA	FlashLoanAdapter/dependency.sol	79087b32295fae36bc2aad879e211e233e65513d61204ccbb48ac48e88c23f8d
FLB	FlashLoanAdapter/flashloan/FlashLoanReceiverBase.sol	8687c06b07452646de67897dfd2d1ccb357aae8078989860dab4d9e4b263892e
IFF	FlashLoanAdapter/flashloan/IFlashLoan.sol	2c81ab0585fcbccece02520ce9b7ffea0dd5417097d0146755a622794d8fa602
IFA	FlashLoanAdapter/flashloan/IFlashLoanReceiver.sol	0e46d513af7bed5bb0885fe323ef1126fc631a47424b76de5d0772badf1d596f

System Overview

FilDA is a highly secure decentralized banking platform containing two fundamental protocols.

- Banking - Lending and Borrowing assets (based on Compound)
- Staking - Locking of assets to earn rewards (based on Harvest)

These two protocols allow users to:

- Deposit - crypto-assets to earn interest (dynamic rates
- Borrow - a variety of crypto assets with no fixed terms
- Stake - crypto pairs in liquidity pools to earn rewards (in FilDA)

Running on the Huobi ECO Chain (HECO) provides a safe and secure environment, with fast transactions and low fees. HECO is a space for users to participate in the DeFi experience, while at the same time, combats many of the performance and cost issues faced by competing platforms.

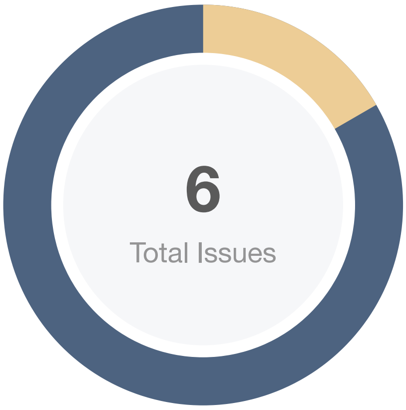
Audit Overview

The scope of the current audit is `FlashLoan` and `FlashLoanAdapter`. And this part has external dependencies (like Chainlink, and Compound). And these external dependencies protocols are not in the scope of this audit.

- FlashLoan - uncollateralized loans that allow borrowing an asset, as long as the borrowed amount is returned before the end of the transaction.

- FlashLoanAdapter - repay the loan by FlashLoan.

Findings



Critical	**0** (0.00%)
Major	**0** (0.00%)
Medium	**0** (0.00%)
Minor	**1** (16.67%)
Informational	**5** (83.33%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
BAF-01	Missing emit event	Coding Style	Informational	Resolved
FLF-01	Pragma version not locked	Coding Style	Informational	Resolved
FLF-02	Discussion on `sub(1e8)`	Logical Issue	Informational	Resolved
FLF-03	Potentially excessive permissions	**Centralization / Privilege**	**Minor**	**Resolved**
FMF-01	State variables that could be declared constant	Coding Style	Informational	Resolved
FMF-02	Divide before multiple	Mathematical Operations	Informational	Resolved

BAF-01 | Missing emit event

Category	Severity	Location	Status
Coding Style	● Informational	FlashLoanAdapter/BaseAdapter.sol: 71(BaseAdapter)	Resolved

Description

Function `setFeeManager` is only called by governance, it allows the caller to change the `feeManager` address. And the state variable `feeManager` is used to calculate the flash loan fee. It is better to add `emit event` to track the changes on variable value.

Recommendation

We recommend adding event and emit it in the function `setFeeManager`.

Alleviation

FilDA team heeded the advice. Added an event in function `setFeeManager` and applied in commited `1d8e0a7ef00b9c4280a95e2e8e91e6e834ae07fe`.

FLF-01 | Pragma version not locked

Category	Severity	Location	Status
Coding Style	● Informational	FlashLoan/FlashLoan.sol: 2	Resolved

Description

solc frequently releases new compiler versions. Using an old version prevents access to new Solidity security checks.

The contract uses some different versions, such as `pragma solidity >=0.4.22 <0.8.0;` , `pragma solidity ^0.5.0;` and `pragma solidity ^0.5.16;`, and all of these are not locked. This is not recommended. Pragmas should be locked to specific compiler versions and flags that they have been tested the most with. Locking the pragma helps ensure that contracts do not accidentally get deployed using, for example, the latest compiler, which may have higher risks of undiscovered bugs.

Recommendation

Avoid a floating pragma version instead specify pragma version without using the caret symbol, i.e. `pragma solidity 0.6.11;`

Deploy with any of the following solidity versions:

- 0.5.11 - 0.5.13
- 0.5.15 - 0.5.17
- 0.6.8
- 0.6.10 - 0.6.11

Use a simple pragma version that allows any of these versions.

We recommend using latest version of solidity for testing.

Alleviation

FilDA team heeded the advice and used 0.5.16 version in the truffle-config.js file.

FLF-02 | Discussion on `sub(1e8)`

Category	Severity	Location	Status
Logical Issue	● Informational	FlashLoan/FlashLoan.sol: 300	Resolved

Description

Why use `sub(1e8)` in line 300?

```
300    return liquidity.sub(1e8).mul(10**decimals).div(tokenPrice);
```

Alleviation

FilDA team removed the `sub(1e8)` code and it was applied in commit `a418c518ba62997a71462d3c628c279b9c566f5b`.

FLF-03 | Potentially excessive permissions

Category	Severity	Location	Status
Centralization / Privilege	Minor	FlashLoan/FlashLoan.sol: 280	Resolved

Description

Function `setOracle` is only called by the governance, and it allows the caller to set `_oracle` address. This oracle address is used to get token price. To improve the trustworthiness of this protocol, any plan to set the `_oracle` address should move to the execution queue of the Timelock, and also add an `emit event`, and make the governance Multi-sig.

Recommendation

We recommend adding an `emit event` at the `setOracle` function. And then transfer the governance of this contract to Timelock, it is better to make the governance Multi-sig, or implement DAO.

Alleviation

FilDA team added an event in function `setOracle` and would transfer the governance to a Multi-sig contract. The change was applied in commit `ebd9052f7456535e90a704a4ccdb220a647fb9f6`.

FMF-01 | State variables that could be declared constant

Category	Severity	Location	Status
Coding Style	Informational	FlashLoanAdapter/FeeManager.sol: 14~16	Resolved

Description

Constant state variables could be declared constant to save gas. And constant variable should be named UPPER_CASE_WITH_UNDERSCORES.

Recommendation

We recommend declaring the state variables as constant variables. And constant variables should be named UPPER_CASE_WITH_UNDERSCORES.

Alleviation

FilDA team heeded our advice and renamed the constant state variables UPPER_CASE_WITH_UNDERSCORES.

The code was applied in commit `1d8e0a7ef00b9c4280a95e2e8e91e6e834ae07fe`.

FMF-02 | Divide before multiple

Category	Severity	Location	Status
Mathematical Operations	● Informational	FlashLoanAdapter/FeeManager.sol: 35(FeeManager)	Resolved

Description

Solidity integer division might truncate. As a result, performing multiplication before division can sometimes avoid loss of precision.

```
100
amount.mul(freeQuota.sub(balance)).div(freeQuota).mul(feeMolecular).div(feeDenominator)
```

Alleviation

FilDA team heeded our advice and performed multiplication before division.

The code was applied in commit `1d8e0a7ef00b9c4280a95e2e8e91e6e834ae07fe`.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without CertiK's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we’re able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

CERTIK