Smart Contract Security Audit Report

Contents

1. Executive Summary

On Jan. 18th, 2021, the SlowMist security team received the Deerfi team's security audit application for the FlashLoan V1, developed the audit plan according to the agreement of both parties and the characteristics of the project, and finally issued the security audit report.

The SlowMist security team adopts the strategy of "white box lead, black, grey box assists" to conduct a complete security test on the project in the way closest to the real attack.

SlowMist Smart Contract project test method:

Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code module through the scripting tool, observing the internal running status, mining weaknesses.
White box testing	Based on the open source code, non-open source code, to detect whether there are vulnerabilities in programs such as nodes, SDK, etc.

SlowMist Smart Contract project risk level:

Critical vulnerabilities	Critical vulnerabilities will have a significant impact on the security of the project, and it is strongly recommended to fix the critical vulnerabilities.
High-risk vulnerabilities	High-risk vulnerabilities will affect the normal operation of project. It is strongly recommended to fix high-risk vulnerabilities.
Medium-risk vulnerabilities	Medium vulnerability will affect the operation of project. It is recommended to fix medium-risk vulnerabilities.

Low-risk vulnerabilities	Low-risk vulnerabilities may affect the operation of project in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed.
Weaknesses	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.
Enhancement Suggestions	There are better practices for coding or architecture.

2. Audit Methodology

Our security audit process for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using public and in-house automated analysis tools.
- Manual audit of the codes for security issues. The contracts are manually analyzed to look for any potential problems.

Following is the list of commonly known vulnerabilities that was considered during the audit of the smart contract:

- Reentrancy attack and other Race Conditions
- Replay attack
- Reordering attack
- Short address attack
- Denial of service attack
- Transaction Ordering Dependence attack
- Conditional Completion attack
- Authority Control attack
- Integer Overflow and Underflow attack

- TimeStamp Dependence attack
- Gas Usage, Gas Limit and Loops
- Redundant fallback function
- Unsafe type Inference
- Explicit visibility of functions state variables
- Logic Flaws
- Uninitialized Storage Pointers
- Floating Points and Numerical Precision
- tx.origin Authentication
- "False top-up" Vulnerability
- Scoping and Declarations

3. Project Background

3.1 Project Introduction

Deer FlashLoan V1 is the world's first decentralized flash loan protocol. Like Uniswap, users can freely create any ERC20 token-based flash loan pool. Developers could borrow the assetS from the Deer FlashLoan V1 poolS by paying a small fee governed by DEER token holders. All liquidity providers who deposit tokens into the flash loan pool will share the fee income pro-rata.

Audit version file information

Initial audit files:

https://github.com/Deerfi/flashloan-v1-core

commit: ee75fb389f77a5e0fde55bfb13d276dd0d3e73aa

Final audit files:

https://github.com/Deerfi/flashloan-v1-core

commit: e60c5b778604317687ba4d7f26b73db0bb098ca7

3.2 Project Structure

```
├── FlashLoanV1ERC20.sol
├── FlashLoanV1Factory.sol
├── FlashLoanV1Pool.sol
├── interfaces
│   ├── IERC20.sol
│   ├── IFlashLoanReceiver.sol
│   ├── IFlashLoanV1ERC20.sol
│   ├── IFlashLoanV1Factory.sol
│   └── IFlashLoanV1Pool.sol
└── libraries
    └── SafeMath.sol
```

4. Code Overview

4.1 Main Contract address

Contract Name	Contract Address
FlashLoanV1Factory	0xa22F8cf50D9827Daef24dCb5BAC92C147a9D342e

4.2 Contracts Description

The SlowMist Security team analyzed the visibility of major contracts during the audit, the result as follows:

FlashLoanV1ERC20			
Function Name	Visibility	Mutability	Modifiers
_mint	Internal	Can modify state	-
_burn	Internal	Can modify state	-
_approve	Private	Can modify state	-
_transfer	Private	Can modify state	-
approve	External	Can modify state	-
transfer	External	Can modify state	-

transferFrom	External	Can modify state	-
permit	External	Can modify state	-

FlashLoanV1Factory			
Function Name	Visibility	Mutability	Modifiers
allPoolsLength	External	-	-
createPool	External	Can modify state	-
setFeeInBips	External	Can modify state	-
setFeeTo	External	Can modify state	-
setFeeToSetter	External	Can modify state	-

FlashLoanV1Pool			
Function Name	Visibility	Mutability	Modifiers
_safeTransfer	Private	Can modify state	-
initialize	External	Can modify state	-
_update	Private	Can modify state	-
_mintFee	Private	Can modify state	-
mint	External	Can modify state	lock
burn	External	Can modify state	lock
flashLoan	External	Can modify state	lock
skim	External	Can modify state	lock
sync	External	Can modify state	lock

4.3 Code Audit

4.3.1 Medium-risk vulnerabilities

4.3.1.1 Risk of excessive authority

The feeToSetter role can arbitrarily set the value of feeInBips through the setFeeInBips function.

Because feeInBips will affect the fee in flashloan, the feeToSetter role will have the risk of excessive

authority.

Fix suggestions: It is suggested to set a maximum limit for feeInBips.

Code location: FlashLoanV1Factory.sol

```
function setFeeInBips(uint _feeInBips) external {
    require(msg.sender == feeToSetter, 'FlashLoanV1: FORBIDDEN');
    feeInBips = _feeInBips;
}
```

Fix status: Fixed.

4.3.2 Enhancement Suggestions

4.3.2.1 Compatibility issues

Users can transfer excess tokens from the pool through the skim function.

If the token of this Pool contract is a rebase token, such as AMPL, then when the token is issued, the tokens in the pool contract will increase accordingly. The user can directly take it out through the skim function. This will damage the liquidity provider.

Fix suggestions: There is no better solution yet, and it is suggested not to create a pool of rebase tokens.

Code location: FlashLoanV1Pool.sol

```
function skim(address to) external lock {
    address _token = token; // gas savings
    _safeTransfer(_token, to, IERC20(_token).balanceOf(address(this)).sub(reserve));
}
```

Fix status: There is no better solution yet, and it is suggested not to create a pool of rebase tokens.

5. Audit Result

5.1 Conclusion

Audit Result : Passed

Audit Number : 0X002101210001

Audit Date : Jan. 21, 2021

Audit Team : SlowMist Security Team

Summary conclusion: The SlowMist security team use a manual and SlowMist Team analysis tool audit of the codes for security issues. There are two security issues found during the audit. There are one low-risk vulnerabilities. We also provide one enhancement suggestions. After communication with the project party, all issues have been fixed or the risks are within the acceptable range.

6. Statement

SlowMist issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility base on these.

For the facts that occurred or existed after the issuance, SlowMist is not able to judge the security status of this project, and is not responsible for them. The security audit analysis and other contents of this report are based on the documents and materials provided to SlowMist by the information provider till the date of the insurance this report (referred to as "provided information"). SlowMist assumes: The information provided is not missing, tampered with, deleted or concealed. If the information provided is missing, tampered with, deleted, concealed, or inconsistent with the actual situation, the SlowMist shall not be liable for any loss or adverse effect resulting therefrom. SlowMist only conducts the agreed security audit on the security situation of the project and issues this report. SlowMist is not responsible for the background and other conditions of the project.

Official Website
www.slowmist.com

E-mail
team@slowmist.com

Twitter
@SlowMist_Team

Github
https://github.com/slowmist