Smart Contract
Security Audit Report

Table Of Contents

1 Executive Summary

On 2023.11.25, the SlowMist security team received the DeSyn Protocol team's security audit application for DeSyn ETH Flashloan Leverage Staking, 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.

The test method information:

Test method	Description
Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	Conduct security testing on code modules 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.

The vulnerability severity level information:

Level	Description
Critical	Critical severity vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High	High severity vulnerabilities will affect the normal operation of the DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium	Medium severity vulnerability will affect the operation of the DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low	Low severity vulnerabilities may affect the operation of the DeFi project in certain scenarios. It is suggested that the project team should evaluate and consider whether these vulnerabilities need to be fixed.
Weakness	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.
Suggestion	There are better practices for coding or architecture.

2 Audit Methodology

The security audit process of SlowMist security team for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using 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:

Serial Number	Audit Class	Audit Subclass
1	Overflow Audit	-
2	Reentrancy Attack Audit	-
3	Replay Attack Audit	-
4	Flashloan Attack Audit	-
5	Race Conditions Audit	Reordering Attack Audit
6	Permission Vulnerability Audit	Access Control Audit
		Excessive Authority Audit
7	Security Design Audit	External Module Safe Use Audit
		Compiler Version Security Audit
		Hard-coded Address Security Audit
		Fallback Function Safe Use Audit
		Show Coding Security Audit
		Function Return Value Security Audit
		External Call Function Security Audit

Serial Number	Audit Class	Audit Subclass
7	Security Design Audit	Block data Dependence Security Audit
		tx.origin Authentication Security Audit
8	Denial of Service Audit	-
9	Gas Optimization Audit	-
10	Design Logic Audit	-
11	Variable Coverage Vulnerability Audit	-
12	"False Top-up" Vulnerability Audit	-
13	Scoping and Declarations Audit	-
14	Malicious Event Log Audit	-
15	Arithmetic Accuracy Deviation Audit	-
16	Uninitialized Storage Pointer Audit	-

3 Project Overview

3.1 Project Introduction

Desyn is a web3 asset management platform that provides a decentralized asset management infrastructure for everyone around the world. This audit is for the new flashloan leverage feature added to the LeverageStake module. The Admin role can use the Uniswap v3 flash function to borrow WETH tokens and convert them to stETH tokens to be deposited into AAVE, and then borrow WETH from AAVE to return the flashloan.

3.2 Vulnerability Information

The following is the status of the vulnerabilities found in this audit:

NO	Title	Category	Level	Status
N1	Unable to perform uniswapV3FlashCallback operation	Design Logic Audit	Critical	Fixed
N2	Potential liquidation risk caused by unrestricted flash loan leverage	Design Logic Audit	Medium	Acknowledged
N3	Flashloan function missing privilege control	Authority Control Vulnerability Audit	High	Fixed
N4	Allowing the free choice of isTrade leads to a potential risk of arbitrage	Reordering Vulnerability	Suggestion	Acknowledged
N5	Allow any type of ETF to interact with AAVE	Others	Information	Acknowledged

4 Code Overview

4.1 Contracts Description

Audit Version:

https://github.com/Meta-DesynLab/desyn-modules/tree/leverageStakingEnhanced

commit: 5cd6f0aa0bc768c24b0fcc17c52a05608364dac5

Fixed Version:

https://github.com/Meta-DesynLab/desyn-modules/tree/leverageStakingEnhanced

commit: 38279a6ade4cb4f6cfc0c600f46944f4369d999f

The main network address of the contract is as follows:

The code was not deployed to the mainnet.

4.2 Visibility Description

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

LeverageStake			
Function Name	Visibility	Mutability	Modifiers
<Constructor>	Public	Can Modify State	-
getAstETHBalance	Public	-	-
getBalanceSheet	Public	-	-
getStethBalance	Public	-	-
getLeverageInfo	Public	-	-
manualUpdatePosition	External	Can Modify State	-
deposit	Internal	Can Modify State	-
borrow	Internal	Can Modify State	-
withdraw	Internal	Can Modify State	-
repayBorrow	Public	Can Modify State	-
setUserUseReserveAsCollateral	External	Can Modify State	-
batchIncreaseLever	External	Can Modify State	-
batchDecreaseLever	External	Can Modify State	-
increaseLever	Public	Can Modify State	-
increaseLeverByFlashloan	External	Can Modify State	-
decreaseLeverByFlashloan	External	Can Modify State	-
uniswapV3FlashCallback	External	Can Modify State	-
decreaseLever	Public	Can Modify State	-
exchange	Internal	Can Modify State	-
convertToAstEth	External	Can Modify State	-

LeverageStake			
convertToWeth	External	Can Modify State	-
setFactory	External	Can Modify State	onlyOwner
setBorrowRate	External	Can Modify State	onlyOwner
setDefaultSlippage	External	Can Modify State	onlyOwner
updateLendingPoolInfo	External	Can Modify State	onlyOwner
_updatePosition	Internal	Can Modify State	-
_checkTx	Internal	-	-

4.3 Vulnerability Summary

[N1] [Critical] Unable to perform uniswapV3FlashCallback operation

Category: Design Logic Audit

Content

In LeverageStake contracts, the uniswapV3FlashCallback function is called by the Uniswap v3 Pool flash function. It will call the repayBorrow/withdraw/deposit function within the LeverageStake contract to interact with the AAVE. However, the repayBorrow/withdraw/deposit function uses `_checkTx` for permission checking and can only be called by the admin role. This will result in the Uniswap v3 Pool not being able to call back to the uniswapV3FlashCallback function properly.

Code location:

contracts/staking/LeverageStake.sol#L345

contracts/staking/LeverageStake.sol#L349

contracts/staking/LeverageStake.sol#L375

contracts/staking/LeverageStake.sol#L177

```
  function uniswapV3FlashCallback(uint256 _fee0, uint256 fee1, bytes calldata data)
external {
    ...
    if (decoded.isDecrease) {
```

```
      if (fee1 > 0) {
        uint256 repayAmount = repayBorrow(decoded.amount1);
        ...
        withdraw(astETHBal);
        ...
      }
      ...
    } else {
      ...
      deposit(receivedStETH);
      ...
  }
```

Solution

It is recommended to implement a repayBorrow/withdraw/deposit function for uniswapV3FlashCallback without `_checkTx` check. Or check whether the caller parameter of FlashCallbackData is admin in the uniswapV3FlashCallback function.

Status

Fixed; After communicating with the project team, the project team stated that they have give the flash loan pool a admin role so that it can perform series of actions

[N2] [Medium] Potential liquidation risk caused by unrestricted flash loan leverage

Category: Design Logic Audit

Content

In LeverageStake contracts, the user can flashloan from the Uniswap v3 Pool with the function createLeverByFlashloan in order to make deposits in AAVE for higher profits. Unfortunately the increaseLeverByFlashloan function does not check the amount of leverage on the current bPool debt. This allows a malicious caller to increase the leverage of the bPool with the increaseLeverByFlashloan function to bring the user's funds closer to the liquidation line. This puts the user's funds at risk of liquidation when the stETH price fluctuates.

Code location: contracts/staking/LeverageStake.sol#L300

```
  function increaseLeverByFlashloan(uint256 lever, bool isTrade) external {
    uint256 curBal = WETH.balanceOf(etf.bPool());
    uint256 amount1 = curBal.mul(lever).div(1000).sub(curBal);
```

```
    ...
  }
```

Solution

It is recommended that the maximum leverage check be performed in the increaseLeverByFlashloan function, as is done for debt in the increaseLever function, rather than accepting an arbitrary amount of leverage passed in by the user.

Status

Acknowledged; After communicating with the project team, the project team stated that the operator of flash loan leverage is the admin role, and that it will monitor the health status of the protocol position off-chain and adjust the leverage in a timely manner.

[N3] [High] Flashloan function missing privilege control

Category: Authority Control Vulnerability Audit

Content

In the LeverageStake contract, any user can call the increaseLeverByFlashloan/decreaseLeverByFlashloan function. Malicious users can consume bPool funds through frequent calls, for example: exchange slippage leads to capital damage, frequent entry/exit of the AAVE pool leads to losses in fees, and frequent flash loans lead to losses in flash loan fees. Although this consumes gas for the malicious caller, it can reduce losses or even make a profit by arbitraging in the Curve Pool or providing liquidity in the Uniswap Pool.

Code location: contracts/staking/LeverageStake.sol#L298-L328

```
  function increaseLeverByFlashloan(uint256 lever, bool isTrade) external {
    ...
  }

  function decreaseLeverByFlashloan() external {
    ...
  }
```

Solution

It is recommended to add `_checkTx` check in the increaseLeverByFlashloan and decreaseLeverByFlashloan functions.

Status

Fixed

[N4] [Suggestion] Allowing the free choice of isTrade leads to a potential risk of arbitrage

Category: Reordering Vulnerability

Content

The increaseLever/increaseLeverByFlashloan/convertToAstEth functions of the LeverageStake contract allow the user to freely choose whether or not to exchange ETH-stETH through the Curve Pool by passing in the isTrade parameter. Even though the contract has a slippage check via the defaultSlippage parameter, the user still has an arbitrage profit.

Code location:

contracts/staking/LeverageStake.sol#L298

contracts/staking/LeverageStake.sol#L257

contracts/staking/LeverageStake.sol#L462

```
function increaseLever(
  uint256 amount,
  uint16 referralCode,
  bool isTrade
) public override returns (uint256) {
  ...
}

function increaseLeverByFlashloan(uint256 lever, bool isTrade) external {
  ...
}

function convertToAstEth(bool isTrade) external override {
  ...
}
```

Solution

It is recommended to flexibly adjust slippage parameters to deal with this risk.

Status

Acknowledged

[N5] [Information] Allow any type of ETF to interact with AAVE

Category: Others

Content

In the `_checkTx` function of the LeverageStake contract, it uses if conditions to check the status of the ETF, which means that both open and closed ETFs can interact with AAVE. And for closed ETFs, it will no longer check whether the closed period of the pool has ended. This is different from the previous version's implementation.

Code location: contracts/staking/LeverageStake.sol#L558-L569

```
function _checkTx() internal view {
  require(!IFactory(factory).isPaused(), 'PAUSED');

  require(etf.adminList(msg.sender), 'NOT_ADMIN');

  (, uint256 collectEndTime, , uint256 closureEndTime, , , , , , , ) =
etf.etfStatus();

  if (etf.etype() == 1) {
    require(etf.isCompletedCollect(), 'COLLECTION_FAILED');
    require(block.timestamp > collectEndTime, 'NOT_REBALANCE_PERIOD');
  }
}
```

Solution

This is a departure from the previous version of the design and it is recommended that the type of ETF and its status be reviewed if it is different from the intended design.

Status

Acknowledged; After communicating with the project team, the project team stated that this was the expected design.

5 Audit Result

Audit Number	Audit Team	Audit Date	Audit Result
0X002311260001	SlowMist Security Team	2023.11.25 - 2023.11.26	Low Risk

Summary conclusion: The SlowMist security team uses a manual and SlowMist team's analysis tool to audit the project, during the audit work we found 1 critical risk, 1 high risk, 1 medium risk, and 1 suggestion. All the findings were fixed or acknowledged. The code was not deployed to the mainnet. The use of the high leverage feature can still result in the liquidation of the bPool's funds in the event of high market volatility, so the protocol remains partially risky.

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 based 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 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