慢雾科技
slow mist

Smart Contract Security Audit

1. Executive Summary

On Aug. 17, 2020, the SlowMist security team received the HAKKA team's security audit application for BlackHoleSwap system, 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 DApp 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 wether there are vulnerabilities in programs suck as nodes, SDK, etc.

SlowMist Smart Contract DApp project risk level:

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

3.1 Project Introduction

BlackHoleSwap is a decentralized AMM (Automatic Market Making) exchange designed for stablecoins. By integrating lending protocols to leverage the excess supply while borrowing on the inadequate side, it can therefore process transactions far exceeding its existing liquidity. Compared to other AMMs, BlackHoleSwap provides nearly infinite liquidity with the lowest price slippage, maximizing capital utilization..

Audit code file:

Hakka_Audit-master.zip

MD5: 3c5b180fb16d002abd0f2f82766a0009

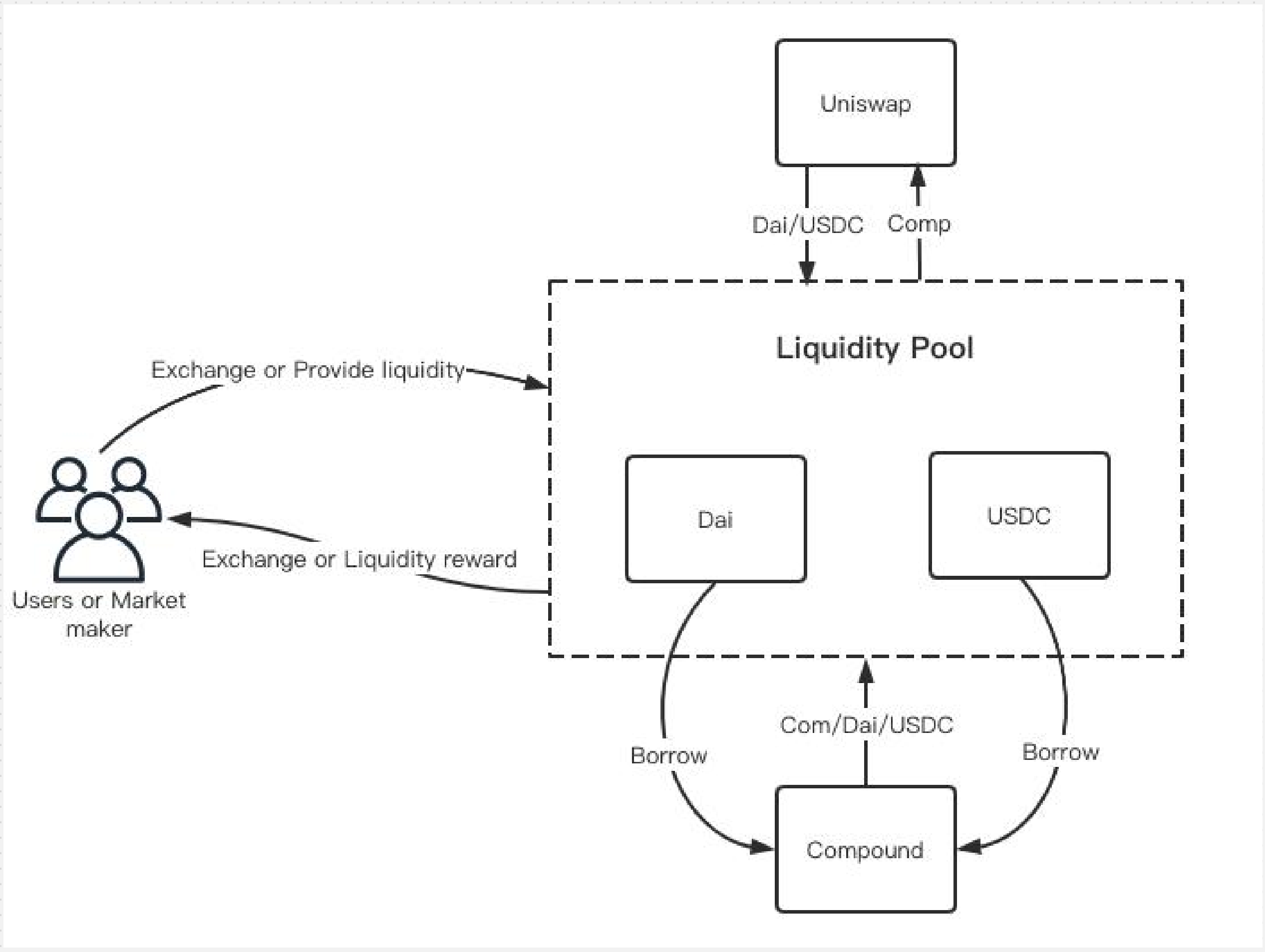
3.2 Project Structure

Hakka_Audit-master:

```
.
├── GuildBank.sol
├── Hakka.sol
├── README.md
└── blackholeswapV1.sol
```

3.3 Contract Structure

BlackHoleSwap builds a token exchange protocol by introducing a leverage multiple on the basis of a mathematical model of constant product. BlackHoleSwap integrates decentralized trading platforms such as Compound and Uniswap to achieve low slippage, unlimited liquidity and additional revenue. When the inventory of one token in the trading pair is exhausted, BlackHoleSwap will use another token as mortgage to lends the required currency from the integrated platform to complete the transaction, and BlackHoleSwap now chooses the lending protocol Compound as it integrates platform, which has a loan mining mechanism. BlackHoleSwap will swap the collected Comp tokens into stablecoins through Uniswap and put them into the BlackHoleSwap liquidity pool, increasing the revenue for liquidity providers. The overall structure of the contract is as follows:



4. Code Overview

4.1 Main File Hash

No	File Name	SHA-1 Hash
1	blackholeswapV1.sol	ffcc6076a70e6d8d9ad0025a3d729d467a1c16cd
2	GuildBank.sol	99c421ff39153596a1203f135a57283435a0e530
3	blackholeswapV1.sol	042da7ef4a5c69e6a58bbdb8a8ad6983a4cdee21

4.2 Main function visibility analysis

Contract Name	Function Name	Visibility
SafeMath	Library	——
	mul	Internal
	div	Internal
	divCeil	Internal
	sub	Internal
	add	Internal
	mod	Internal
	mul	Internal
	div	Internal
	sub	Internal
	add	Internal
	sqrt	Internal
ERC20	Implementation	——
	totalSupply	Public
	balanceOf	Public
	allowance	Public

	transfer	Public
	approve	Public
	transferFrom	Public
	_transfer	Internal
ERC20Mintable	Implementation	ERC20
	_mint	Internal
	_burn	Internal
CERC20	Implementation	ERC20
	borrow	External
	borrowBalanceCurrent	External
	repayBorrow	External
	mint	External
	redeemUnderlying	External
	balanceOfUnderlying	External
Comptroller	Interface	——
	enterMarkets	External
UniswapV2Router02	Implementation	——
	swapExactTokensForTokens	External
blackholeswap	Implementation	ERC20Mintable

	setAdmin	External
	setParams	External
	setVault	External
	getDaiBalance	Public
	getUSDCBalance	Public
	S	External
	F	Internal
	getInputPrice	Public
	getOutputPrice	Public
	rate	Public
	calcFee	Internal
	dai2usdcIn	External
	usdc2daiIn	External
	dai2usdcOut	External
	usdc2daiOut	External
	doTransferIn	Internal
	doTransferOut	Internal
	securityCheck	Internal
	addLiquidity	External

	removeLiquidity	External
	collectComp	Public
Ownable	Implementation	——
	transferOwnership	Public
ERC20NonStandard	Implementation	——
	transfer	Public
Burner	Implementation	——
	ragequit	External
GuildBank	Implementation	Ownable
	withdraw	External
	doTransferOut	Internal

4.3 Code Audit

4.3.1 Risk of over approve

When the contract is initialized, the contract approve amount to cDai, cUSDC, and Uniswap is uint256(-1). If the authorized external contract appears unknown risk or acts maliciously, all the authorized assets will be affected.

Code location: File blackholeswapV1.sol line 235 to 237

```
    constructor() public {
        symbol = "BHSc$";
        name = "BlackHoleSwap-Compound DAI/USDC v1";
        decimals = 18;

        Dai.approve(address(cDai), uint256(-1));
        USDC.approve(address(cUSDC), uint256(-1));
        Comp.approve(address(uniswap), uint256(-1));

        address[] memory cTokens = new address[](2);
        cTokens[0] = address(cDai);
        cTokens[1] = address(cUSDC);
        uint256[] memory errors = comptroller.enterMarkets(cTokens);
        require(errors[0] == 0 && errors[1] == 0, "Comptroller.enterMarkets failed.");

        admin = msg.sender;
    }
```

Fix status: After confirming with the project party, in order to save the gas usage, one-time approve should be taken and there is no countermeasure.

4.3.2 Excessive reliance on external contractual risk

BlackHoleSwap provides additional revenue and unlimited liquidity through the Compound project and the Uniswap project, but hardcodes the interaction logic directly into the contract. BlackHoleSwap will also be affected if the external project stops operating.

Code location: File blackholeswapV1.sol line 580.

```
function collectComp() public {
        uint256 _comp = Comp.balanceOf(address(this));
        if(_comp == 0) return;

        (uint256 a, uint256 b) = getDaiBalance();
        (uint256 c, uint256 d) = getUSDCBalance();

        bool isDai = a.add(d) > c.add(b);

        address[] memory path = new address[](3);
```

```
            path[0] = address(Comp);
            path[1] = 0xC02aaA39b223FE8D0A0e5C4F27eAD9083C756Cc2; //weth
            path[2] = isDai ? address(Dai) : address(USDC);
            uint256[] memory amounts = uniswap.swapExactTokensForTokens(_comp, 0, path, address(this), now);

            if(isDai)
                require(cDai.mint(amounts[2]) == 0, "ctoken.mint failed");
            else
                require(cUSDC.mint(amounts[2]) == 0, "ctoken.mint failed");

        }

}
```

Fix status: After confirming with the project party, no impact on the business, and the code will not be modified.

5. Audit Result

5.1 Enhancement Suggestions

- Part of the code is redundant

5.2 Conclusion

Audit Result : Passed

Audit Number : 0X002009020002

Audit Date : Sep. 02, 2020

Audit Team : SlowMist Security Team

Summary conclusion: The are 2 security issues found during the audit. After communication and feedback, with the Anyswap team, confirms that the risks found in the audit process are within the tolerable 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.

慢雾科技
slow mist

Official Website

www.slowmist.com

E-mail

team@slowmist.com

Twitter

@SlowMist_Team

WeChat Official Account