RD AUDITORS

YSL PROTOCOL SMART CONTRACT, CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: YSL Protocol
Prepared on: 07/10/2021
Platform: BSC
Language: Solidity

TABLE OF CONTENTS

THIS DOCUMENT MAY CONTAIN CONFIDENTIAL INFORMATION ABOUT ITS SYSTEMS AND INTELLECTUAL PROPERTY OF THE CUSTOMER AS WELL AS INFORMATION ABOUT POTENTIAL VULNERABILITIES AND METHODS OF THEIR EXPLOITATION.

THE REPORT CONTAINING CONFIDENTIAL INFORMATION CAN BE USED INTERNALLY BY THE CUSTOMER OR IT CAN BE DISCLOSED PUBLICLY AFTER ALL VULNERABILITIES ARE FIXED - UPON THE DECISION OF THE CUSTOMER.

Document

Name	Smart Contract Code Review and Security Analysis Report for YSL Protocol
Platform	BSC / Solidity
File 1	Airdrop.sol
MD5 hash	3C75545401581239EF7E632918466172
SHA256 hash	D88BA24BA8AABB062D51EF5754DF612106F2F9768C20820C51CB6F307F31DA2B
File 2	ApeAdapter.sol
MD5 hash	EC3ECAAAB474BCB4A59103C1AB18BD08
SHA256 hash	F8C9D41B16EE9FBE182C94E17448CE739C121636DEB77BBB2EB2C63250915AC4
File 3	aYSLToken.sol
MD5 hash	EE1D49BCD79BE295E85BB88DB6C838BC
SHA256 hash	A03EAFDFF2B1F7C5A4B54BE466440344C408E1FE9CFA22CFF566F2DD50CEA552
File 4	Referral.sol
MD5 hash	D54216433F4E0A62024C0AE4FBAB4336
SHA256 hash	AB88F9439136D300C863ACA237D21E47796B1D7FEAC481365C9FCB810999E5CC
File 5	StrategyLock.sol
MD5 hash	20C9894C6E5757CF007F7C3B41F4251B
SHA256 hash	18D1BD1DAE606570A75251F2BACFBCB64F3F46BE286091689795B7FA092C05F6
File 6	StrategySwap.sol

MD5 hash	BF0C1ED89E910CD67E64ACB5E4BA8CCB
SHA256 hash	BD57DE794D7884BA9D9B8F3473C9E85B466303DA1989A1963CA6850EFDEA42C5
File 7	sYSLShare.sol
MD5 hash	7BD39412C03B8F02A6DB08273E6218A1
SHA256 hash	DAE3D2CA541453A333ECAC857122E56C744848F36CA3410D5F5D8FA92997BB7D
File 8	sYSLToken.sol
MD5 hash	01053020E421F2C694C0424E070553B6
SHA256 hash	E6DCD096C053F101934938836D4DB56194F7D2500A5569A0E37B1AA7CF233462
File 9	xYSLAdapter.sol
MD5 hash	0F36645FF9D91791F5E9A9F6C1044E45
SHA256 hash	8C46D4A4344E562495B86F8FBA9198B0750A49D09C3184BAE8BD7A6EBC830932
File 10	xYSLToken.sol
MD5 hash	BE4E5A77F81BAE2415EC40C0FE2128DB
SHA256 hash	C1BCFDD50EE59EB432F890F2E03F2D037B0AC0C54065BC9C4C553244C79FB30A
File 11	YSLOpt.sol
MD5 hash	18802922232B80F31E67A85D5993B340
SHA256 hash	CA8F8852E9E72605B497AC0FFDB7A3D119FF15EB4709184E4EBEEE745D3B986D
File 12	YSLOptSingleApe.sol
MD5 hash	FE8ACAE71768402844106F3A11464E88
SHA256 hash	E601CDDEDDC490079D140D4C5344D952D1F216FDEDA0CB4E26BB80DF477A6842

File 13	YSLOptSinglePancake.sol
MD5 hash	AFA4E659443CA2FCAB6700AC64A9BFC0
SHA256 hash	0B5E2BA196A643A43A54219272FC45EC47C1F9144CD26C36B055940D053EE0C0
File 14	YSLProtocol.sol
MD5 hash	4CFA454205205F7A3AEA802EF2D1D7A1
SHA256 hash	3831A4678B9948A6E452894FDE6BCCC24DEA6DE72C3F2FF05573489AA5CB6A52
File 15	YSLSwapAdapter.sol
MD5 hash	B4BD138A1DFB6AF7A140DC4D8D3973CA
SHA256 hash	A31D606686C1871AD5951CE994C7C7BAB2346069AB59D64C23C636026D5EAB9A
File 16	YSLToken.sol
MD5 hash	CDCCC99D63AA18194F3F683E773D1986
SHA256 hash	42F67DA76E97891B6AB544A03E5DF9CBC37065BA1E7DC5661CE69A0A39D08263
File 17	YSLxBUSDShare.sol
MD5 hash	4CFB0F71DB5EF37EA533F2E2D367FBA1
SHA256 hash	0969A06B20591EA0A4CF955F11DDA3396D6B4474F5CC2EE3E2BA4075C7DD251C
Date	07/10/2021

Introduction

RD Auditors (Consultant) were contracted by YSL Protocol (Customer) to conduct a Smart Contracts Code Review and Security Analysis. This report represents the findings of the security assessment of the customer`s smart contracts and its code review conducted between 12 - 22 September and 24 September - 07 October 2021.

We have checked 17 files of this contract, which are in the above table.

Project Scope

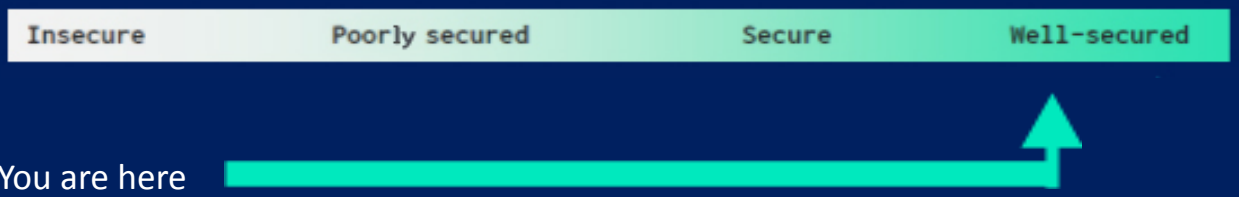
The scope of the project is a smart contract.

We have scanned this smart contract for commonly known and more specific vulnerabilities, below are those considered (the full list includes but is not limited to):

- Reentrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with (Unexpected) Throw
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Byte array vulnerabilities
- Style guide violation
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Unchecked external call - Unchecked math
- Unsafe type inference
- Implicit visibility level

Executive Summary

According to the assessment, the customer's solidity smart contract is well-secured.



Automated checks are with smartDec, Mythril, Slither and remix IDE. All issues were performed by our team, which included the analysis of code functionality, manual audit found during automated analysis were manually reviewed and applicable vulnerabilities are presented in the audit overview section. The general overview is presented in the AS-IS section and all issues found are located in the audit overview section.

We found 0 critical, 0 high, 0 medium, 0 low and 0 very low level issues.

Code Quality

The libraries within this smart contract are part of a logical algorithm. A library is a different type of smart contract that contains reusable code. Once deployed on the blockchain (only once), it is assigned to a specific address and its properties/methods can be reused many times by other contracts.

The YSL Protocol team has also conducted unit tests using scripts provided through the same github link which fortify functionality and security of the contract, which also helped us to determine the integrity of the code in an automated way.

Overall, the code is well commented. Commenting can provide rich documentation for functions, return variables and more. Use of Ethereum Natural Language Specification Format (NatSpec) for commenting is recommended.

Documentation

We were given the YSL Protocol contract as a github link:

https://github.com/ysl-io/ysl-protocol

The hash of that file is mentioned in the table. As mentioned above, It's well commented smart contract code, so anyone can quickly understand the programming flow as well as complex code logic.

Comments are very helpful in understanding the overall architecture of the protocol. It also provides a clear overview of the system components, including helpful details, like the lifetime of the background script.

Use of Dependencies

As per our observation, the libraries are used in this smart contract infrastructure. Those were based on well-known industry standard open source projects and even core code blocks that are written well and systematically.

AS-IS Overview

YSL Protocol

File And Function Level Report

File: Airdrop.sol

Contract: Airdrop
Import: Ownable
Inherit: Ownable
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	addRem	write	Passed	All Passed	No Issue	Passed
2	addNux	write	Passed	All Passed	No Issue	Passed
3	claim	write	Passed	All Passed	No Issue	Passed
4	getLockAmount	read	Passed	All Passed	No Issue	Passed
5	isAirdropped	read	Passed	All Passed	No Issue	Passed
6	addAirdropped	write	Passed	All Passed	No Issue	Passed
7	removeAirdropped	write	Passed	All Passed	No Issue	Passed

File: Referral.sol

Contract: Referral
Inherit: Ownable, AccessControl
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	adminAddReferal	write	Passed	All Passed	No Issue	Passed
2	adminDeleteReferral	write	Passed	All Passed	No Issue	Passed
3	setOperator	write	Passed	All Passed	No Issue	Passed
4	removeOperator	write	Passed	All Passed	No Issue	Passed
5	isOperator	read	Passed	All Passed	No Issue	Passed
6	proccessReferral	write	Passed	All Passed	No Issue	Passed
7	hasReferral	read	Passed	All Passed	No Issue	Passed

File: YSLOpt.sol

Contract: YSLOpt
Inherit: AccessControlUpgradeable, IStrategy
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	initialize	write	Passed	All Passed	No Issue	Passed
2	setCoefficients	write	Passed	All Passed	No Issue	Passed
3	setControllerFee	write	Passed	All Passed	No Issue	Passed
4	deposit	write	Passed	All Passed	No Issue	Passed
5	withdraw	write	Passed	All Passed	No Issue	Passed
6	addLiquidity	write	Passed	All Passed	No Issue	Passed
7	getPrice	read	Passed	All Passed	No Issue	Passed

File: YSLOptSingleApe.sol

Contract: YSLOptSingleApe
Inherit: YSLOpt
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	initialize	write	Passed	All Passed	No Issue	Passed
2	compound	write	Passed	All Passed	No Issue	Passed
3	earn	write	Passed	All Passed	No Issue	Passed
4	transferOut	write	Passed	All Passed	No Issue	Passed
5	transferIn	write	Passed	All Passed	No Issue	Passed
6	getUserDepositedUSD	read	Passed	All Passed	No Issue	Passed
7	getTotalDepositedUSD	read	Passed	All Passed	No Issue	Passed
8	mintBonus	write	Passed	All Passed	No Issue	Passed
9	sendReferralBonus	write	Passed	All Passed	No Issue	Passed
10	withdrawAcc	write	Passed	All Passed	No Issue	Passed
11	restorePool	write	Passed	All Passed	No Issue	Passed

File: YSLOptSinglePancake.sol

Contract: YSLOptSinglePancake
Inherit: YSLOpt
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	initialize	write	Passed	All Passed	No Issue	Passed
2	compound	write	Passed	All Passed	No Issue	Passed
3	earn	write	Passed	All Passed	No Issue	Passed
4	transferOut	write	Passed	All Passed	No Issue	Passed
5	transferIn	write	Passed	All Passed	No Issue	Passed
6	getUserDepositedUSD	read	Passed	All Passed	No Issue	Passed
7	getTotalDepositedUSD	read	Passed	All Passed	No Issue	Passed
8	mintBonus	write	Passed	All Passed	No Issue	Passed
9	sendReferralBonus	write	Passed	All Passed	No Issue	Passed
10	withdrawAcc	write	Passed	All Passed	No Issue	Passed
11	restorePool	write	Passed	All Passed	No Issue	Passed

File: YSLxBUSDShare.sol

Contract: YSLxBUSDShare
Inherit: YSLOpt
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	initialize	write	Passed	All Passed	No Issue	Passed
2	transferIn	write	Passed	All Passed	No Issue	Passed

File: sYSLShare.sol

Contract: sYSLShare
Inherit: YSLOpt
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	initialize	write	Passed	All Passed	No Issue	Passed
2	transferIn	write	Passed	All Passed	No Issue	Passed

File: ApeAdapter.sol

Contract: ApeAdapter
Inherit: Initializable, OwnableUpgradeable
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	initialize	write	Passed	All Passed	No Issue	Passed
2	setPair	write	Passed	All Passed	No Issue	Passed
3	withdrawTeam	write	Passed	All Passed	No Issue	Passed
4	recoverToken	write	Passed	All Passed	No Issue	Passed
5	setAirdrop	write	Passed	All Passed	No Issue	Passed
6	setSYSL	write	Passed	All Passed	No Issue	Passed
7	setDeadline	write	Passed	All Passed	No Issue	Passed
8	addLiquidityApe	write	Passed	All Passed	No Issue	Passed
9	neverWithdraw	write	Passed	All Passed	No Issue	Passed
10	sYSLwithdraw	write	Passed	All Passed	No Issue	Passed

File: YSLProtocol.sol

Contract: YSLProtocol
Inherit: OwnableUpgradeable
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	initialize	write	Passed	All Passed	No Issue	Passed
2	add	write	Passed	All Passed	No Issue	Passed
3	setAlloc	write	Passed	All Passed	No Issue	Passed
4	setStrat	write	Passed	All Passed	No Issue	Passed
5	setTreasury	write	Passed	All Passed	No Issue	Passed
6	setTeamRate	write	Passed	All Passed	No Issue	Passed
7	setReferral	write	Passed	All Passed	No Issue	Passed
8	updateEmissionRate	write	Passed	All Passed	No Issue	Passed
9	activatePool	write	Passed	All Passed	No Issue	Passed
10	deactivatePool	write	Passed	All Passed	No Issue	Passed
11	massUpdatePools	write	Passed	All Passed	No Issue	Passed
12	updatePool	write	Passed	All Passed	No Issue	Passed
13	deposit	write	Passed	All Passed	No Issue	Passed
14	depositFrom	write	Passed	All Passed	No Issue	Passed
15	withdraw	write	Passed	All Passed	No Issue	Passed
16	harvestAll	write	Passed	All Passed	No Issue	Passed
17	harvest	write	Passed	All Passed	No Issue	Passed
18	_harvest	write	Passed	All Passed	No Issue	Passed
19	pendingsYSL	read	Passed	All Passed	No Issue	Passed
20	isAmplified	read	Passed	All Passed	No Issue	Passed
21	safeSYSLtransfer	write	Passed	All Passed	No Issue	Passed

File: YSLSwapAdapter.sol

Contract: YSLSwapAdapter
Inherit: Ownable
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	set_sYSLPath	write	Passed	All Passed	No Issue	Passed
2	swap	write	Passed	All Passed	No Issue	Passed
3	swap	write	Passed	All Passed	No Issue	Passed
4	_swapBNBForTokens	write	Passed	All Passed	No Issue	Passed

5	_swapTokensForBNB	write	Passed	All Passed	No Issue	Passed
6	_swapTokensForTokens	write	Passed	All Passed	No Issue	Passed
7	_swapBNBFor_sYSL	write	Passed	All Passed	No Issue	Passed
8	_swap_sYSLForBNB	write	Passed	All Passed	No Issue	Passed
9	_swapTokensFor_sYSL	write	Passed	All Passed	No Issue	Passed
10	_swap_sYSLForTokens	write	Passed	All Passed	No Issue	Passed
11	getAmountOut	read	Passed	All Passed	No Issue	Passed
12	reversedPath	read	Passed	All Passed	No Issue	Passed

File: xYSLAdapter.sol

Contract: xYSLAdapter
Inherit: AccessControl
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	collectxYSLFee	write	Passed	All Passed	No Issue	Passed

File: StrategySwap.sol

Contract: StrategySwap
Inherit: Context, Initializable
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	initialize	write	Passed	All Passed	No Issue	Passed

2	swapLPToBusd	write	Passed	All Passed	No Issue	Passed
3	swapBusdToLP	write	Passed	All Passed	No Issue	Passed
4	swapLPToBusd	write	Passed	All Passed	No Issue	Passed
5	swapBusdToLP	write	Passed	All Passed	No Issue	Passed
6	getBusdAmount	read	Passed	All Passed	No Issue	Passed
7	getBusdAmount	read	Passed	All Passed	No Issue	Passed
8	reversedPath	read	Passed	All Passed	No Issue	Passed

File: StrategyLock.sol

Contract: StrategyLock
Inherit: Context, Initializable
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	initialize	write	Passed	All Passed	No Issue	Passed
2	lock	write	Passed	All Passed	No Issue	Passed
3	unlock	write	Passed	All Passed	No Issue	Passed
4	getUnlockedFunds	read	Passed	All Passed	No Issue	Passed
5	_vestedAmount	read	Passed	All Passed	No Issue	Passed

File: YSLToken.sol

Contract: YSLToken
Inherit: ERC20, Ownable, AccessControl
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	isMinter	write	Passed	All Passed	No Issue	Passed
2	mintFor	write	Passed	All Passed	No Issue	Passed
3	mint	write	Passed	All Passed	No Issue	Passed
4	burnFrom	write	Passed	All Passed	No Issue	Passed
5	burn	write	Passed	All Passed	No Issue	Passed
6	setMinter	write	Passed	All Passed	No Issue	Passed
7	removeMinter	write	Passed	All Passed	No Issue	Passed

File: sYSLToken.sol

Contract: sYSLToken
Inherit: ERC20, Ownable, AccessControl
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	setMinter	write	Passed	All Passed	No Issue	Passed
2	removeMinter	write	Passed	All Passed	No Issue	Passed
3	addToWhitelist	write	Passed	All Passed	No Issue	Passed
4	removeFromWhitelist	write	Passed	All Passed	No Issue	Passed
5	mint	write	Passed	All Passed	No Issue	Passed
6	mintFor	write	Passed	All Passed	No Issue	Passed
7	mintPurchased	write	Passed	All Passed	No Issue	Passed
8	mintAirdropped	write	Passed	All Passed	No Issue	Passed
9	burn	write	Passed	All Passed	No Issue	Passed
10	burnFrom	write	Passed	All Passed	No Issue	Passed
11	_beforeTokenTransfer	write	Passed	All Passed	No Issue	Passed
12	unlock	write	Passed	All Passed	No Issue	Passed
13	unlockAirdrop	write	Passed	All Passed	No Issue	Passed

File: xYSLToken.sol

Contract: xYSLToken
Inherit: ERC20, Ownable, AccessControl
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	isMinter	write	Passed	All Passed	No Issue	Passed
2	mintFor	write	Passed	All Passed	No Issue	Passed
3	mint	write	Passed	All Passed	No Issue	Passed
4	burnFrom	write	Passed	All Passed	No Issue	Passed
5	burn	write	Passed	All Passed	No Issue	Passed
6	setMinter	write	Passed	All Passed	No Issue	Passed
7	removeMinter	write	Passed	All Passed	No Issue	Passed

File: aYSLToken.sol

Contract: aYSLToken
Inherit: ERC20, Ownable, AccessControl
Observation: Passed
Test Report: Passed
Score: Passed
Conclusion: Passed

Sl.	Function	Type	Observation	Test Report	Conclusion	Score
1	isMinter	write	Passed	All Passed	No Issue	Passed
2	mintFor	write	Passed	All Passed	No Issue	Passed
3	mint	write	Passed	All Passed	No Issue	Passed
4	burnFrom	write	Passed	All Passed	No Issue	Passed
5	burn	write	Passed	All Passed	No Issue	Passed
6	setMinter	write	Passed	All Passed	No Issue	Passed
7	removeMinter	write	Passed	All Passed	No Issue	Passed

FILES TO BE IGNORED

The below files are available in the Github link provided by the client (https://github.com/ysl-io/ysl-protocol). They have not been audited by us, nor has it been used in this contract.

1. PriceDiscovery.sol
2. YSLOptApe.sol
3. YSLOptPancake.sol
4. sYSLxBUSDShare.sol
5. All files inside ysl-protocol/contracts/mock/ are for testing purposes.

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to lost tokens etc.
High	High level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g. public access to crucial functions.
Medium	Medium level vulnerabilities are important to fix; however, they cannot lead to lost tokens.
Low	Low level vulnerabilities are most related to outdated, unused etc. These code snippets cannot have a significant impact on execution.
Lowest Code Style/ Best Practice	Lowest level vulnerabilities, code style violations and information statements cannot affect smart contract execution and can be ignored.

Audit Findings

Critical

No critical severity vulnerabilities were found.

High

No high severity vulnerabilities were found.

Medium

No medium severity vulnerabilities were found.

Low

No low severity vulnerabilities were found.

No Very Low

No very low severity vulnerabilities were found.

Discussion

All loops need to be checked and set to some limit otherwise it may fail.

Conclusion

We were given a contract file and have used all possible tests based on the given object. This contract is now ready to go for production.

Since possible test cases can be unlimited and developer level documentation (code flow diagram with function level description) not provided, for such an extensive smart contract protocol, we provide no such guarantee of future outcomes. We have used all the latest static tools and manual observations to cover maximum possible test cases to scan everything.

The security state of the reviewed contract is now “well secured”

Note For Contract user

There might be other contracts in their platform which are unaudited.. Owner has full control over the smart contract. Thus, technical auditing does not guarantee the project's ethical side.

Please do your due diligence before investing. Our audit report is never an investment advice.

Please also note that the below areas are safety functions, however it can be used maliciously.

Referral.sol

```
function adminAddReferal(address sender, address creator) external operatorOnly {
    require(creator != address(0), "Use delete referral");
    referrals[sender] = creator;
}
```

The owner can manipulate the user's referral address.

```
function adminDeleteReferral(address sender) external operatorOnly {
    referrals[sender] = address(0);
}
```

The owner can remove the user's referral address.

YSLProtocol.sol

```
function deactivatePool(uint256 _pid) external onlyOwner {
    PoolInfo storage pool = poolInfo[_pid];
    require(pool.isActive, "The pool is already deactivated");
    pool.isActive = false;
}
```

The owner could potentially deactivate the pool, stopping deposits.

Our Methodology

We like to work with a transparent process and make our reviews a collaborative effort. The goals of our security audits are to improve the quality of systems we review and aim for sufficient remediation to help protect users. The following is the methodology we use in our security audit process.

Manual Code Review:

In manually reviewing all of the code, we look for any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behavior when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our audit techniques included manual code analysis, user interface interaction, and whitebox penetration testing. We look at the project's web site to get a high level understanding of what functionality the software under review provides. We then meet with the developers to gain an appreciation of their vision of the software. We install and use the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We follow a conservative, transparent process for analyzing potential security vulnerabilities and seeing them through successful remediation. Whenever a potential issue is discovered, we immediately create an Issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is conservative because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this we analyse the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take, and finally we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinised by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the details are made public.

Disclaimers

The smart contracts given for audit have been analysed in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment and functionality (performing the intended functions).

Because the total number of test cases are unlimited, the audit makes no statements or warranties on the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bugfree status or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only - we recommend proceeding with several independent audits and a public bug bounty program to ensure security of smart contracts.

Technical Disclaimer

Smart contracts are deployed and executed on the blockchain. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to hacks. Thus, the audit can’t guarantee explicit security of the audited smart contracts.

RD
AUDITORS

Email: info@rdauditors.com

Website: www.rdauditors.com