SPIRITSWAP CORE SMART CONTRACT AUDIT

June 08, 2021

MixBytes()

CONTENTS

1. INTRODUCTION

1.1 DISCLAIMER

The audit makes no statements or warranties about utility of the code, safety of the code, suitability of the business model, investment advice, endorsement of the platform or its products, regulatory regime for the business model, or any other statements about fitness of the contracts to purpose, or their bug free status. The audit documentation is for discussion purposes only. The information presented in this report is confidential and privileged. If you are reading this report, you agree to keep it confidential, not to copy, disclose or disseminate without the agreement of SpiritSwap Finance. If you are not the intended recipient(s) of this document, please note that any disclosure, copying or dissemination of its content is strictly forbidden.

1.2 PROJECT OVERVIEW

SpiritSwap is a decentralized exchange (DEX) on the Fantom Opera Chain. SpiritSwap's design is based on the Uniswap constant-product automated market maker (AMM). In an AMM, liquidity providers simply deposit a pair of tokens and an algorithm automatically makes markets for the token pair. Traders can easily swap between tokens in the AMM and get guaranteed rates for the swaps. Each swap on SpiritSwap incurs a fee, which gets distributed to liquidity providers.

1.3 SECURITY ASSESSMENT METHODOLOGY

At least 2 auditors are involved in the work on the audit who check the provided source code independently of each other in accordance with the methodology described below:

01 "Blind" audit includes:
> Manual code study
> "Reverse" research and study of the architecture of the code based on the source code only

Stage goal:
Building an independent view of the project's architecture
Finding logical flaws

02 Checking the code against the checklist of known vulnerabilities includes:
> Manual code check for vulnerabilities from the company's internal checklist
> The company's checklist is constantly updated based on the analysis of hacks, research and audit of the clients' code

Stage goal:
Eliminate typical vulnerabilities (e.g. reentrancy, gas limit, flashloan attacks, etc.)

03 Checking the logic, architecture of the security model for compliance with the desired model, which includes:
> Detailed study of the project documentation
> Examining contracts tests
> Examining comments in code
> Comparison of the desired model obtained during the study with the reversed view obtained during the blind audit

Stage goal:
Detection of inconsistencies with the desired model

04 Consolidation of the reports from all auditors into one common interim report document
> Cross check: each auditor reviews the reports of the others
> Discussion of the found issues by the auditors
> Formation of a general (merged) report

Stage goal:
Re-check all the problems for relevance and correctness of the threat level
Provide the client with an interim report

05 Bug fixing & re-check.
> Client fixes or comments on every issue
> Upon completion of the bug fixing, the auditors double-check each fix and set the statuses with a link to the fix

Stage goal:
Preparation of the final code version with all the fixes

06 Preparation of the final audit report and delivery to the customer.

Findings discovered during the audit are classified as follows:

FINDINGS SEVERITY BREAKDOWN

Level	Description	Required action
Critical	Bugs leading to assets theft, fund access locking, or any other loss funds to be transferred to any party	Immediate action to fix issue
Major	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.	Implement fix as soon as possible
Warning	Bugs that can break the intended contract logic or expose it to DoS attacks	Take into consideration and implement fix in certain period
Comment	Other issues and recommendations reported to/acknowledged by the team	Take into consideration

Based on the feedback received from the Customer's team regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The project team is aware of this finding. Recommendations for this finding are planned to be resolved in the future. This finding does not affect the overall safety of the project.
No issue	Finding does not affect the overall safety of the project and does not violate the logic of its work.

1.4 EXECUTIVE SUMMARY

In the scope of the audit there are several smart-contracts intended to add, remove and provide liquidity, implement swap of tokens and automatically issue of project token SPIRIT as a fee for liquidity participated in the swap operations.

1.5 PROJECT DASHBOARD

Client	SpiritSwap Finance
Audit name	SpiritSwap-Core
Initial version	a23463f87fd3c7633e97fab5e8124b4499e1519e
Final version	-
SLOC	2095
Date	2021-05-14 - 2021-06-08
Auditors engaged	2 auditors

FILES LISTING

SPIRITMasterChef.sol	SPIRITMasterChef.sol
SPIRITToken.sol	SPIRITToken.sol
SpiritFactory.sol	SpiritFactory.sol
SpiritMultiCall.sol	SpiritMultiCall.sol
SpiritRouter.sol	SpiritRouter.sol

FINDINGS SUMMARY

Level	Amount
Critical	0
Major	0
Warning	1
Comment	4

CONCLUSION

Smart contracts have been audited and several suspicious places have been spotted. During the audit no critical or major issues were found. One issue was marked as warning and several comments were found and discussed with the client. After working on the reported findings all of them were acknowledged (as the problems were not critical). So, the contracts are assumed as secure to use according to our security criteria.

2. FINDINGS REPORT

2.1 CRITICAL

Not Found

2.2 MAJOR

Not Found

2.3 WARNING

WRN-1	Useless/insecure pause functionality into SpiritFactory
File	SpiritFactory.sol
Severity	Warning
Status	Acknowledged

DESCRIPTION

The SpiritFactory contains implementation to pause / to lock swap operations. SpiritFactory.sol#L499 However, that lock status is ignored inside PancakePair and can be easyly avoided by an attacker.

RECOMMENDATION

Please note that PancakePair `swap()` function has unrestricted access. Any swap-related restrictions should be implemented inside the PancakePair instead of the other contracts. Depending on what is actually required, we recommend to move implementation of pause/lock into PancakePair or to remove it and save the gas.

CLIENT'S COMMENTARY

This was added as a feature to lock the swap function. We made this because there was a scammer who was scanning the transactions and doing swap against the pancake LPs before we launched. This will be removed as part of the AMM upgrade in future.

2.4 COMMENTS

CMT-1	Function getMultiplier() can be restricted to "pure"
File	SPIRITMasterChef.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

At the line
SPIRITMasterChef.sol#L1251
the "view" modifier is used for function restriction, but when a function does not read any storage data, it's recommended to use "pure" modificator instead of "view".

RECOMMENDATION

Use "pure" modificator or inline function.

CLIENT'S COMMENTARY

This will be fixed as part of the AMM upgrade in future

CMT-2	getReserves() contains unused function call
File	SpiritRouter.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

This function
SpiritRouter.sol#L274
has unused pairFor() call

RECOMMENDATION

We recommend to remove line 274

CLIENT'S COMMENTARY

This will be removed as part of the AMM upgrade in future

CMT-3	Unsafe fee recipient addresses transition
File	SPIRITMasterChef.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

The setter functions of fee and developer fee address SPIRITMasterChef.sol#L1360 does not do any check on new address. If a wrong address is accidentally used, there is no way to fix it anymore.

RECOMMENDATION

We recommend to transfer ownership and fee addresses into two steps: approve by the old address and accept by a new address.

CLIENT'S COMMENTARY

This will be fixed as part of the AMM upgrade in future

CMT-4	YAM protocol incompliance/useless operation for YAM protocol compliance
File	SPIRITToken.sol
Severity	Comment
Status	Acknowledged

DESCRIPTION

The SPIRIT token contains some code pieces for YAM protocol implementation. However, implementation is incomplete, e.g. withdrawal doesn't move delegation to the recipient.

RECOMMENDATION

We recommend to complete or to remove YAM compliance implementation depending on whether is YAM implementation actually planned or not.

CLIENT'S COMMENTARY

This will be fixed as part of the AMM upgrade in future

3. ABOUT MIXBYTES

MixBytes is a team of blockchain developers, auditors and analysts keen on decentralized systems. We build open-source solutions, smart contracts and blockchain protocols, perform security audits, work on benchmarking and software testing solutions, do research and tech consultancy.

BLOCKCHAINS

 Ethereum

 Cosmos

 EOS

 Substrate

TECH STACK

 Python

 Solidity

 Rust

 C++

CONTACTS

 https://github.com/mixbytes/audits_public

 https://mixbytes.io/

 hello@mixbytes.io

 https://t.me/MixBytes

 https://twitter.com/mixbytes