BLOCK AUDIT REPORT

Smart Contract Security Audit Report

VanSwap

Block Audit Report Team received the VanFarm.sol, VanPool.sol, VanSwapFactory.sol, and VanSwapRouter.sol files for smart contract security audit of the on July 26, 2022. The following are the details and results of this smart contract security audit:

Project Name: VanSwap

Link Address:
https://www.visionscan.org/contract/VVyeB9gSWvhgyv9ewYpJ6mbdB8sNWpxEJm/code

https://www.visionscan.org/contract/VKV66XJrAqyGFRURaeR6M7W4isZrrhNjPR/code

https://github.com/InfoVanswap/VanSwap-contratcts.git

The audit items and results:
(Other undiscovered security vulnerabilities are not included in the audit responsibility scope)

Audit Result: Passed

Audit Number: BAR0027626072022
Audit Date: July 26, 2022
Audit Team: Block Audit Report Team

Table of Content

Introduction

This Audit Report mainly focuses on the extensive security of VanSwap Smart Contracts. With this report, we attempt to ensure the reliability and correctness of the smart contract by complete and rigorous assessment of the system's architecture and the smart contract codebase.

Auditing Approach and Methodologies applied

The Block Audit Report team has performed rigorous testing of the project including the analysis of the code design patterns where we reviewed the smart contract architecture to ensure it is structured along with the safe use of standard inherited contracts and libraries. Our team also conducted a formal line by line inspection of the Smart Contract i.e., a manual review, to find potential issues including but not limited to;

- Race conditions
- Zero race conditions approval attacks
- Re-entrancy
- Transaction-ordering dependence
- Timestamp dependence
- Check-effects-interaction pattern (optimistic accounting)
- Decentralized denial-of-service attacks
- Secure ether transfer pattern
- Guard check pattern
- Fail-safe mode
- Gas-limits and infinite loops
- Call Stack depth

In the Unit testing Phase, we coded/conducted custom unit tests written against each function in the contract to verify the claimed functionality from our client.
In Automated Testing, we tested the Smart Contract with our standard set of multifunctional tools to identify vulnerabilities and security flaws.
The code was tested in collaboration of our multiple team members and this included but not limited to;

- Testing the functionality of the Smart Contract to determine proper logic has been followed throughout the whole process.
- Analyzing the complexity of the code in depth and detailed, manual review of the code, line-by-line.
- Deploying the code on testnet using multiple clients to run live tests.
- Analyzing failure preparations to check how the Smart Contract performs in case of any bugs and vulnerabilities.
- Checking whether all the libraries used in the code are on the latest version.
- Analyzing the security of the on-chain data.

Audit Details

Project Name: **VanSwap**
Website/Etherscan Code (**Testnet**):

https://github.com/InfoVanswap/VanSwap-contratcts.git

Languages: Solidity (Smart contract)
Platforms and Tools: Remix IDE, Truffle, Ganache, Mythril, Contract Library, Slither, Dapp.Tools, Echidna, Etheno

Audit Goals

The focus of the audit was to verify that the Smart Contract System is secure, resilient and working according to the specifications. The audit activities can be grouped in the following three categories:

Security Sight

Identifying security related issues within each contract and the system of contract.

Sound Architecture

Evaluation of the architecture of this system through the lens of established smart contract best practices and general software best practices, standard software design principle, design patterns and practices.

Code Correctness and Quality

A full review of the contract source code. The primary areas of focus include:

- Accuracy
- Readability
- Usability vs Security
- Sections of code with high complexity
- Quantity and quality of test coverage

Issue Categories

Every issue in this report was assigned a severity level from the following:

Critical Severity Issues

Issues of this level are critical to the smart contract's performance/functionality and should be fixed before moving to a production environment.

High level severity issues

Issues on this level are strongly suggested by the team to be fixed before moving to the production environment.

Medium level severity issues

Issues on this level could potentially bring problems and should eventually be fixed.

Low level severity issues

Issues on this level are minor details and warning's that can remain unfixed but would be better fixed at some point in the future.

Issues Checking Status

№	Issue description	Checking status
1	Compiler warnings.	Failed
2	Race conditions and Reentrancy. Cross-function race conditions.	Passed
3	Oracle calls.	Passed
4	Timestamp dependence.	Failed
5	DoS with Revert.	Passed
6	DoS with block gas limit.	Passed
7	Methods execution permissions.	Passed
8	Economy model.	Passed
9	The impact of the exchange rate on the logic.	Passed
10	Malicious Event log.	Passed
11	Scoping and Declarations.	Passed
12	Uninitialized storage pointers.	Passed
13	Arithmetic Operations accuracy.	Passed
14	Design Logic.	Passed
15	Cross-function race conditions.	Passed
16	Safe usage for Open Zeppelin module.	Failed
17	Fallback function security.	Passed
18	Send & receive ether.	Passed
19	Zero race condition approval attacks.	Passed
20	Short address attack.	Passed
21	Owner’s authority to freeze.	Failed
22	Attempt to block ether flows.	Passed
23	Redundant inheritance check.	Passed
24	Silent overrides of mapping structs.	Passed
25	Function state mutability.	Passed
26	Unnecessary conversion of type.	Passed

Manual Audit:

For this section the code was tested/read line by line by our auditors. We used Remix IDE's JavaScript VM, Truffle and testnet Kovan to test the contract functionality in a simulated environment.

Files in scope

All solidity files in:

https://github.com/InfoVanswap/VanSwap-contratcts.git

File 1 of 4 : VanFarm.sol	Passed
File 2 of 4 : VanSwapFactory.sol	Passed
File 3 of 4 : VanPool.sol	Passed
File 4 of 4 : VanSwapRouter.sol	Passed

Issue Categories

Critical Severity Issues

No critical severity issues found.

High Severity Issues

No high severity issues found.

Medium Severity Issues

Issue	File	Type	Line	Description
#1	VanSwapRouter.sol	Access Control	320, 337	There are insufficient access controls in the remove liquidity function that may lead to burning of tokens by anyone because the burn function itself doesn't check for allowance.

Low Severity Issues

Issue	File	Type	Line	Description
#1	VanSwapRouter.sol	Missing zero check	All the functions that requires an address.	Add a zero address check
#2	VanSwapRouter.sol	Missing events	All the functions in VanswapRouter02 contract (229-643)	Emit events for critical parameter changes
#3	VanSwapRouter.sol	Floating Pragma	1	The pragma is not locked The current pragma Solidity directive is ,">=0.5.0"
#4	VanSwapRouter.sol	Old compiler version	1	Old compiler versions are deprecated and known to have bugs in them and are not recommended for deployment
#5	VanPool.sol/VanFarm.sol	Old libraries used (0.6.0)	563/793	Old versions of libraries are prone to known bugs and it is recommended to use the

				latest version of such libraries
#6	VanPool.sol/VanFarm.sol	Floating Pragma	N/A	The pragma is not locked The current pragma Solidity directive is,"^0.8.0". There are multiple pragmas in the contracts which is not recommended.
#7	VanSwapFactory.sol	Missing safety checks	313, 337, 362	There are no safety checks in these functions. We understand that this contract is used to be used as import but we still recommend to put some safety checks in place in these functions. For example, zero address checks, approval check, access control, etc.
#8	VanSwapFactory.sol	Missing zero check	165, 176, 441, 446	Check that the address is not zero, otherwise the tokens may be lost.
#9	VanSwapFactory.sol	Missing Events	259, 441, 446	Emit events for critical parameter changes
#10	VanSwapFactory.sol	Commented Code	134	There is some commented code exists in the contract and we recommend it to be removed.
#11	All Files	Contract doesn't import npm packages from source (like OpenZeppelin etc.)	N/A	We recommend to import all packages from npm directly without flatten the contract. Functions could be modified or can be susceptible to vulnerabilities
#12	VanFarm.sol	Check for existing address	1414	While adding an address in the whitelist, make sure that it doesn't exist beforehand in order to avoid double entries and unnecessary gas consumptions
#13	VanFarm.sol	Missing zero check	1525, 1530, 1535, 1540	Check that the address is not zero

Owner privileges

Issue	File	Description
#1	VanPool.sol and VanFarm.sol	• Contracts are pausable by the owner/operators • The owner can halt the transactions and lock user funds whenever they want. • Moreover there are multiple authorities controlling the contract. For example, it's also possible to call certain ownership functions even after renouncing ownership of the contract. • The owner can blacklist whoever they want

Auditor's Comments:

- There are some compilation errors with the files provided and it was conveyed to us that we should use compiler version "0.5.10" but the contracts explicitly uses 0.8.0 version as well which means there are multiple compiler versions used in the contracts. Thus, it is not recommended.
- There are some bugs that will exist in the contracts if they are not used as an import.
- We also recommend to extensively unit test these contracts before deployment and provide us with test cases during the reaudit.

Automated Audit

No major issues were found. Some false positive errors were reported by the tools. All the other issues have been categorized above according to their level of severity.

Disclaimer

This is a limited report on our findings based on our analysis, in accordance with good industry practice as at the date of this report, in relation to cybersecurity vulnerabilities and issues in the framework and algorithms based on smart contracts, the details of which are set out in this report. In order to get a full view of our analysis, it is crucial for the client to read the full report. While we have done our best in conducting our analysis and producing this report, it is important to note that the client should not rely on this report and cannot claim against us on the basis of what it says or doesn't say, or how we produced it, and it is important for the client to conduct the client's own independent investigations before making any decisions. We go into more detail on this in the below disclaimer below – please make sure to read it in full.

By reading this report or any part of it, the client agrees to the terms of this disclaimer. If the client does not agree to the terms, then please immediately cease reading this report, and delete and destroy any/all copies of this report downloaded and/or printed by the client. This report is provided for information purposes only and stays on a non-reliance basis, and does not constitute investment advice. No one/ NONE shall have any rights to rely on the report or its contents, and BlockAudit and its affiliates (including holding companies, shareholders, subsidiaries, employees, directors, officers and other representatives).
(BlockAudit) owes no duty of care towards the client or any other person, nor does BlockAudit claim any warranty or representation to any person on the accuracy or completeness of the report. The report is provided "as is", without any conditions, warranties or other terms of any kind except as set out in this disclaimer, and BlockAudit hereby excludes all representations, warranties, conditions and other terms (including, without limitation, the warranties implied by law of satisfactory quality, fitness for purpose and the use of reasonable care and skill) which, but for this clause, might have effects in relation to the report. Except and only to the extent that it is prohibited by law, BlockAudit hereby excludes all liability and responsibility, and neither the client nor any other person shall have any claim against BlockAudit, for any amount or kind of loss or damage that may result to the client or any other person (including without limitation, any direct, indirect, special, punitive, consequential or pure economic loss or damages, or any loss of income, profits, goodwill, data, contracts, use of money, or business interruption, and whether in delict, tort (including without limitation negligence), contract, breach of statutory duty, misrepresentation (whether innocent or negligent) or otherwise under any claim of any nature whatsoever in any jurisdiction) in any way arising from or connected with this report and the use, inability to use or the results of use of this report, and any reliance on this report.

The analysis of the security is purely based on the received smart contracts alone. No related/third-party smart contracts, applications or operations were reviewed for security. No product code has been reviewed.
Note: The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the VanSwap team put a bug bounty program in place to encourage further analysis of the smart contracts by other third parties.

Summary

Smart contracts received in file named: “VanSwap” does not contain any high severity issues!

Note:
Please read the disclaimer above and note, the audit claims NO statements or warranties on business model, investment advice/ attractiveness or code sustainability. This report is provided for the only set of contracts mentioned in the report and does not claim responsibility to include security audits for any other contracts deployed by Owner.

BLOCK AUDIT REPORT

Official Website
www.blockaudit.report

E-Mail
team@blockaudit.report

Twitter
https://twitter.com/BlockAudit

Github
https://github.com/blockauditreport