nimbus

SMART CONTRACT AUDIT

ZOKYO.

Dec 21, 2020 | v. 1.0

PASS

Zokyo's Security Team has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges.

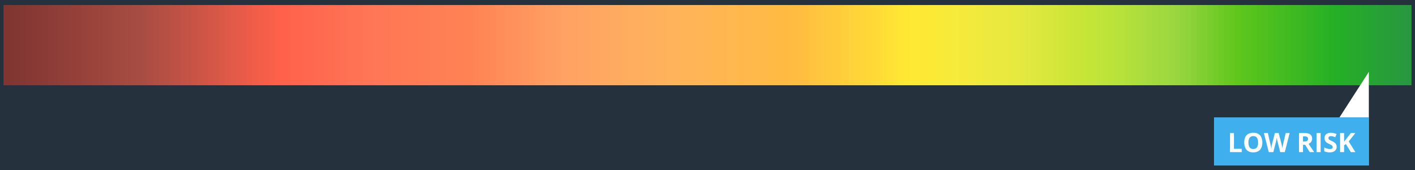
GOOD

TECHNICAL SUMMARY

This document outlines the overall security of the Nimbus smart contracts, evaluated by Zokyo's Blockchain Security team.

The scope of this audit was to analyze and document the Nimbus smart contract codebase for quality, security, and correctness.

Contract Status



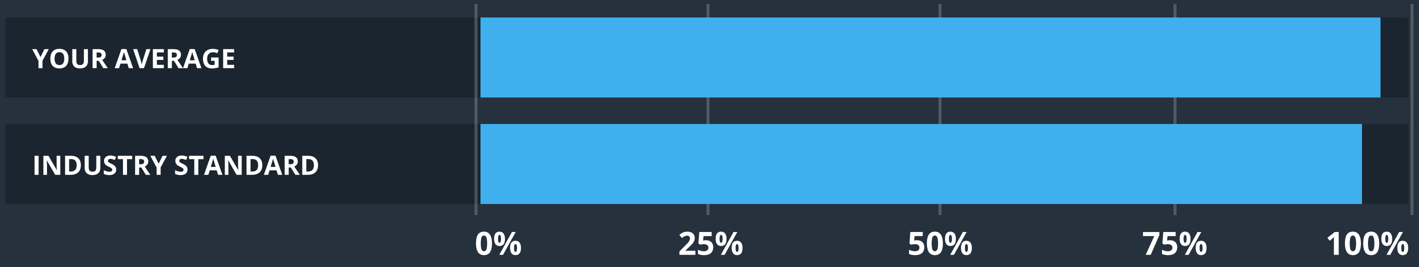
There were no critical issues found during the audit. Contracts suite is based on Uniswap projects:

- Uniswap Core (https://github.com/Uniswap/uniswap-v2-core);
- Uniswap Peripheral (https://github.com/Uniswap/uniswap-v2-periphery);
- Uniswap Governance (https://github.com/Uniswap/uniswap-v2-periphery);
- Uniswap Liquidity Staking (https://github.com/Uniswap/liquidity-staker).

Customizations list:

- Uniswap text occurrences were replaced with Nimbus substring;
- Uniswap Governance Uni token contract lost the minting feature;
- Uniswap Governance Uni token contract acquired the burning feature.

Testable Code



Testable code is 97% which is above the industry standard of 95%.

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the Ethereum network's fast-paced and rapidly changing environment, we at Zokyo recommend that the Nimbus team put in place a bug bounty program to encourage further and active analysis of the smart contract.

TABLE OF CONTENTS

AUDITING STRATEGY AND TECHNIQUES APPLIED

The Smart contract's source code was taken from the following repository - https://github.com/nimbusplatformorg/nim-smartcontract.
Commit id – 54bfea423fdb006a590635dfd2b0de1e3c6ced27.

Throughout the review process, care was taken to ensure that the token contract:

- Implements and adheres to existing Token standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices in efficient use of gas, without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the latest vulnerabilities;
- Whether the code meets best practices in code readability, etc.

Zokyo's Security Team has followed best practices and industry-standard techniques to verify the implementation of Nimbus smart contracts. To do so, the code is reviewed line-by-line by our smart contract developers, documenting any issues as they are discovered. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

1	Due diligence in assessing the overall code quality of the codebase.	3	Testing contract logic against common and uncommon attack vectors.
2	Cross-comparison with other, similar smart contracts by industry leaders.	4	Thorough, manual review of the codebase, line-by-line.

EXECUTIVE SUMMARY

Based on the code analyzed, we can give a Good score to the codebase provided. The issues found and described below are stopping us from giving the Excellent score.

STRUCTURE AND ORGANIZATION OF DOCUMENT

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged "Resolved" or "Unresolved" depending on whether they have been fixed or addressed. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

● Critical

The issue affects the ability of the contract to compile or operate in a significant way.

● High

The issue affects the ability of the contract to compile or operate in a significant way.

● Medium

The issue affects the ability of the contract to operate in a way that doesn't significantly hinder its behavior.

● Low

The issue has minimal impact on the contract's ability to operate.

● Informational

The issue has no impact on the contract's ability to operate.

MANUAL REVIEW

MEDIUM | URESOLVED

Smart contracts are using outdated Solidity compiler versions (0.5.16, 0.6.6).

Recommendation:
Use up-to-date Solidity compiler version for all contracts.

MEDIUM | RESOLVED

NUS.sol, NUS_WETH.sol, REWARDSFACTORY.sol are not referring to specific Solidity compiler version.

Recommendation:
Stick to the specific Solidity compiler version.

INFORMATIONAL | RESOLVED

File NimbisFactory.sol has an incorrect name.

Recommendation:
Rename file to NimbusFactory.sol

INFORMATIONAL | UNRESOLVED

Complicated dependency management, Solidity files contain all dependencies required for specific Contract.

Recommendation:
Split FACTORY.sol, ROUTER.sol, NUS.sol, NUS_WETH.sol, REWARDSFACTORY.sol to have one contract per Solidity file.

MEDIUM | RESOLVED

StakingRewards.sol is a complete copy of the StakingRewards contract located at StakingRewardFactory.sol, and is not utilized by end product as StakingRewards has to be created by StakingRewardsFactory.

Recommendation:
Remove StakingRewards.sol

INFORMATIONAL | RESOLVED

NUS.sol, Code-style issue, method tokensBurner has incorrect name as it is not clearly states the goal of method.

Recommendation:
Rename method to reflect behaviour, like burnTokensByBurner or burnOwnerTokensByBurner

INFORMATIONAL | RESOLVED

NUS_WETH.sol, Code-style issue, Owned & NUS_WETH contracts are having unexpected property names: NUSITERATOR, SWAPROUTER, NUSTOKEN, NUSWETH.

Recommendation:
Use camel case for all properties that are not constants.

INFORMATIONAL | UNRESOLVED

NUS_WETH.sol, Code-style issue, Nus_Weth contract name should be in Pascal case instead of Pascal Snake Case.

Recommendation:
Use pascal case for contract names.

	NimbusFactory	NimbusRouter	NimbusPair
Re-entrancy	Not affected	Not affected	Not affected
Access Management Hierarchy	Not affected	Not affected	Not affected
Arithmetic Over/Under Flows	Not affected	Not affected	Not affected
Unexpected Ether	Not affected	Not affected	Not affected
Delegatecall	Not affected	Not affected	Not affected
Default Public Visibility	Not affected	Not affected	Not affected
Hidden Malicious Code	Not affected	Not affected	Not affected
Entropy Illusion (Lack of Randomness)	Not affected	Not affected	Not affected
External Contract Referencing	Not affected	Not affected	Not affected
Short Address/ Parameter Attack	Not affected	Not affected	Not affected
Unchecked CALL Return Values	Not affected	Not affected	Not affected
Race Conditions / Front Running	Not affected	Not affected	Not affected
General Denial Of Service (DOS)	Not affected	Not affected	Not affected

Uninitialized Storage Pointers	Not affected	Not affected	Not affected
Floating Points and Precision	Not affected	Not affected	Not affected
Tx.Origin Authentication	Not affected	Not affected	Not affected
Signatures Replay	Not affected	Not affected	Not affected
Pool Asset Security (backdoors in the underlying ERC-20)	Not affected	Not affected	Not affected

	WETH	**StakingRewardFactory**	**StakingReward**
Re-entrancy	Not affected	Not affected	Not affected
Access Management Hierarchy	Not affected	Not affected	Not affected
Arithmetic Over/Under Flows	Not affected	Not affected	Not affected
Unexpected Ether	Not affected	Not affected	Not affected
Delegatecall	Not affected	Not affected	Not affected
Default Public Visibility	Not affected	Not affected	Not affected
Hidden Malicious Code	Not affected	Not affected	Not affected
Entropy Illusion (Lack of Randomness)	Not affected	Not affected	Not affected
External Contract Referencing	Not affected	Not affected	Not affected
Short Address/ Parameter Attack	Not affected	Not affected	Not affected
Unchecked CALL Return Values	Not affected	Not affected	Not affected

Race Conditions / Front Running	Not affected	Not affected	Not affected
General Denial Of Service (DOS)	Not affected	Not affected	Not affected
Uninitialized Storage Pointers	Not affected	Not affected	Not affected
Floating Points and Precision	Not affected	Not affected	Not affected
Tx.Origin Authentication	Not affected	Not affected	Not affected
Signatures Replay	Not affected	Not affected	Not affected
Pool Asset Security (backdoors in the underlying ERC-20)	Not affected	Not affected	Not affected

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Nimbus team (original test coverage)

The Nimbus team did not write or provide any integration/e2e tests.

Automation testing by Zokyo Secured team

As part of our work assisting Nimbus in verifying the correctness of their contract code, our team was responsible for supplying additional tests by porting tests from the Uniswap project using the Waffle testing framework.

```
NimbusERC20
  ✓ name, symbol, decimals, totalSupply, balanceOf, DOMAIN_SEPARATOR, PERMIT_TYPEHASH (132ms)
  ✓ approve (120ms)
  ✓ transfer (136ms)
  ✓ transfer:fail (50ms)
  ✓ transferFrom (200ms)
  ✓ transferFrom:max (176ms)
  ✓ permit (44ms)

NimbusFactory
  ✓ feeTo, feeToSetter, allPairsLength (41ms)
  ✓ createPair (304ms)
  ✓ createPair:reverse (318ms)
  ✓ createPair:gas (124ms)
  ✓ setFeeTo (81ms)
  ✓ setFeeToSetter (99ms)

NimbusPair
  ✓ mint (218ms)
  ✓ getInputPrice:0 (321ms)
  ✓ getInputPrice:1 (289ms)
  ✓ getInputPrice:2 (317ms)
  ✓ getInputPrice:3 (316ms)
```

```
    ✓ getInputPrice:4 (284ms)
    ✓ getInputPrice:5 (271ms)
    ✓ getInputPrice:6 (275ms)
    ✓ optimistic:0 (305ms)
    ✓ optimistic:1 (255ms)
    ✓ optimistic:2 (253ms)
    ✓ optimistic:3 (254ms)
    ✓ swap:token0 (328ms)
    ✓ swap:token1 (287ms)
    ✓ swap:gas (286ms)
    ✓ burn (322ms)
    ✓ price{0,1}CumulativeLast (373ms)
    ✓ feeTo:off (319ms)
    ✓ feeTo:on (421ms)

  32 passing (9s)

  NUS
    ✓ permit (405ms)
    ✓ nested delegation (690ms)

  2 passing (1s)

  UniswapV2Router02
  WARNING: unsupported ABI type - receive
  WARNING: unsupported ABI type - receive
    ✓ quote (109ms)
    ✓ getAmountOut (159ms)
    ✓ getAmountIn (177ms)
    ✓ getAmountsOut (605ms)
    ✓ getAmountsIn (856ms)

  fee-on-transfer tokens
  WARNING: unsupported ABI type - receive
  WARNING: unsupported ABI type - receive
    ✓ removeLiquidityETHSupportingFeeOnTransferTokens (473ms)
    ✓ removeLiquidityETHWithPermitSupportingFeeOnTransferTokens (479ms)
    ✓ swapExactETHForTokensSupportingFeeOnTransferTokens (217ms)
    ✓ swapExactTokensForETHSupportingFeeOnTransferTokens (324ms)
    swapExactTokensForTokensSupportingFeeOnTransferTokens
      ✓ DTT -> WETH (117ms)
      ✓ WETH -> DTT (206ms)

  fee-on-transfer tokens: reloaded
    swapExactTokensForTokensSupportingFeeOnTransferTokens
  WARNING: unsupported ABI type - receive
  WARNING: unsupported ABI type - receive
    ✓ DTT -> DTT2 (162ms)


  12 passing (10s)


  StakingRewardsFactory
    ✓ deployment gas
    # deploy
      ✓ pushes the token into the list (239ms)
      ✓ fails if called twice for same token (258ms)
      ✓ can only be called by the owner (41ms)
      ✓ stores the address of stakingRewards and reward amount (229ms)
      ✓ deployed staking rewards has correct parameters (290ms)
    # notifyRewardsAmounts
      ✓ called before any deploys
      after deploying all staking reward contracts
        ✓ gas (574ms)
        ✓ no op if called twice (621ms)
        ✓ fails if called without sufficient balance
        ✓ calls notifyRewards on each contract (426ms)
        ✓ transfers the reward tokens to the individual contracts (440ms)
        ✓ sets rewardAmount to 0 (540ms)
        ✓ succeeds when has sufficient balance and after genesis time (363ms)


  14 passing (9s)


  StakingRewards
    ✓ deploy cost (165ms)
    ✓ rewardsDuration
    ✓ notifyRewardAmount: full (1071ms)
    ✓ stakeWithPermit (840ms)
    ✓ notifyRewardAmount: ~half (769ms)
    ✓ notifyRewardAmount: two stakers (1266ms)

  6 passing (5s)

  Overall Tests: 66
  Passed: 66
  Failed: 0
```

We are grateful to have been given the opportunity to work with the Nimbus team.

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.

Zokyo's Security Team recommends that the Nimbus team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.