verichains

SECURITY AUDIT OF

VNDC 2.0

Public Report

Aug 06, 2024

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ABBREVIATIONS

Name	Description
Ethereum	An open source platform based on blockchain technology to create and distribute smart contracts and decentralized applications.
Ether (ETH)	A cryptocurrency whose blockchain is generated by the Ethereum platform. Ether is used for payment of transactions and computing services in the Ethereum network.
Smart contract	A computer protocol intended to digitally facilitate, verify or enforce the negotiation or performance of a contract.
Solidity	A contract-oriented, high-level language for implementing smart contracts for the Ethereum platform.
Solc	A compiler for Solidity.
ERC20	ERC20 (BEP20 in Binance Smart Chain or *x*RP20 in other chains) tokens are blockchain-based assets that have value and can be sent and received. The primary difference with the primary coin is that instead of running on their own blockchain, ERC20 tokens are issued on a network that supports smart contracts such as Ethereum or Binance Smart Chain.

EXECUTIVE SUMMARY

This Security Audit Report was prepared by Verichains Lab on Aug 06, 2024. We would like to thank the ONUS for trusting Verichains Lab in auditing smart contracts. Delivering high-quality audits is always our top priority.

This audit focused on identifying security flaws in code and the design of the VNDC 2.0. The scope of the audit is limited to the source code files provided to Verichains. Verichains Lab completed the assessment using manual, static, and dynamic analysis techniques.

During the audit process, the audit team had identified some vulnerable issue in the smart contracts code. The VNDC 2.0 development team has acknowledged the issues identified through the previously sent draft.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About VNDC 2.0

VNDC 2.0 is a protocol providing on-chain liquidity. By ensuring that each VNDC in circulation is collateralized by a corresponding amount of USDT according to the USDT/VNDC exchange rate, VNDC 2.0 brings transparency, openness, and convenience to VNDC users. Users can convert VNDC tokens into USDT or other assets without going through intermediaries.

At the Beta version launch, VNDC 2.0 is being deployed on the ONUSChain network. The protocol will expand to other networks such as Ethereum, Binance SmartChain, and Arbitrum… in the near future.

1.2. Audit Scope

This audit focused on identifying security flaws in code and the design of the VNDC 2.0.

It was conduct on the commit `b4a5f3708ac979d1341920ef0342fd3246324336` from git repository `https://github.com/ONUS-APP/vndc-2.0/blob/main/contracts/VNDC2.0.sol`.

The latest version of the following files were made available in the course of the review:

SHA256 Sum	File
`5702588a9d4f869277e46c3fa05d5d6b6434ffcdc70b5732b913cffe1be4dc81`	`contracts/VNDC2.0.sol`

Table 1. The audited files

1.3. 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 RK87, our in-house smart contract security analysis tool.
- 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 were considered during the audit of the smart contract:

- Integer Overflow and Underflow
- Timestamp Dependence
- Race Conditions
- Transaction-Ordering Dependence

- DoS with (Unexpected) revert
- DoS with Block Gas Limit
- Gas Usage, Gas Limit and Loops
- Redundant fallback function
- Unsafe type Inference
- Reentrancy
- Explicit visibility of functions state variables (external, internal, private and public)
- Logic Flaws

For vulnerabilities, we categorize the findings into categories as listed in table below, depending on their severity level:

SEVERITY LEVEL	DESCRIPTION
CRITICAL	A vulnerability that can disrupt the contract functioning; creates a critical risk to the contract; required to be fixed immediately.
HIGH	A vulnerability that could affect the desired outcome of executing the contract with high impact; needs to be fixed with high priority.
MEDIUM	A vulnerability that could affect the desired outcome of executing the contract with medium impact in a specific scenario; needs to be fixed.
LOW	An issue that does not have a significant impact, can be considered as less important.

Table 2. Severity levels

1.4. Disclaimer

ONUS acknowledges that the security services provided by Verichains, are conducted to the best of their professional abilities but cannot guarantee 100% coverage of all security vulnerabilities. ONUS understands and accepts that despite rigorous auditing, certain vulnerabilities may remain undetected. Therefore, ONUS agrees that Verichains shall not be held responsible or liable, and shall not be charged for any hacking incidents that occur due to security vulnerabilities not identified during the audit process.

1.5. Acceptance Minute

This final report served by Verichains to the ONUS will be considered an Acceptance Minute. Within 7 days, if no any further responses or reports is received from the ONUS, the final report will be considered fully accepted by the ONUS without the signature.

2. AUDIT RESULT

2.1. Overview

The VNDC 2.0 is a smart contract written in Solidity with a required version of `0.8.18`. It extends the `ERC20`, `ERC20Burnable`, `Pausable`, and `Ownable` contracts and uses `SafeMath` and `SafeERC20` for secure arithmetic and token operations. Below are some key properties and audit findings of the VNDC 2.0:

PROPERTY	VALUE
Name	VNDC
Symbol	VNDC
Decimals	0
Initial Supply	26,000,000,000 (26 billion)

Table 3. The VNDC 2.0 properties

The audit team examined the `ERC20` token against standard protocols, with the following results:

Title	Status
Total Supply Consistency	Passed
Approval	Passed
Self Transfer	Passed
Transfer from/to Zero	Passed
Transfer Effective	Passed

Table 4. The standard testing for ERC20

The VNDC 2.0 contract includes additional functionalities such as minting and withdrawing through a whitelist system, interaction with liquidity pools, and setting various parameters like fees and bonuses.

2.2. Findings

During the audit process, the audit team identified some vulnerabilities in the VNDC 2.0.

#	Title	Severity	Status
1	Price Manipulation	MEDIUM	ACKNOWLEDGED
2	Sandwich Attack Due to On-Chain Slippage Calculation	MEDIUM	ACKNOWLEDGED

Table 5. The issues list

2.2.1. MEDIUM - Price Manipulation

Description

The `mintWhitelist` function calculates the amount of VNDC to be minted based on the reserves of VNDC and the specified currency in a liquidity pool. An attacker can exploit this by manipulating the reserves to mint an artificially inflated amount of VNDC.

By manipulating the reserves in the liquidity pool, an attacker can mint a significantly higher amount of VNDC than intended.

```
function mintWhitelist(uint256 _amount, address _currency, uint256 slippage) public {
    //...
    (uint256 VNDCReserve, uint256 currencyReserve) = getReserve(_currency);
    uint256 amountsOut = router.quote(_amount, currencyReserve, VNDCReserve);
    require(getPriceInputToken(_amount, _currency) > minMaxSetting.minPriceMint, "The price
must be more than minPriceMint");

    IERC20(_currency).safeTransferFrom(msg.sender, address(this), _amount);
    //...
    _mint(address(this), amountsOut);
    //...

    emit EventMintWhitelist(msg.sender, _amount, _currency, amountsOut + bonusVNDC,
liquidity);
}
```

Exploit Scenario:

- **Initial Reserves**: Assume initial reserves of VNDC and another currency (e.g., USDT) in the liquidity pool are stable.
- **Price Manipulation**: An attacker swaps a large amount of VNDC to USDT, significantly decreasing the VNDC price in the pool.
- **Minting VNDC**: Using the manipulated reserves, the attacker calls the `mintWhitelist` function to mint VNDC at an inflated rate, receiving more VNDC than intended.

- **Reverting Price**: The attacker can perform a swap to revert the price to their original.
- **Profit**: The attacker profits from sell more VNDC than they should have received.

RECOMMENDATION

To mitigate this issue, consider implementing the following strategies:

- **Time-Weighted Average Prices (TWAP)**.
- **External Price Oracles**.

By implementing these measures, the contract can significantly reduce the risk of price manipulation and ensure a fair and stable minting process for VNDC tokens.

UPDATES

- ***Aug 06, 2024***: The issue is acknowledged by the VNDC 2.0 team.

2.2.2. MEDIUM - Sandwich Attack Due to On-Chain Slippage Calculation

Description

The `mintWhitelist` and `withdrawWhitelist` functions calculate slippage directly on-chain, which exposes them to sandwich attacks. In these attacks, bad actors use the lack of minimum requirements to inset transactions before and after, influencing the token's price and benefiting at the expense of other traders.

```
function mintWhitelist(uint256 _amount, address _currency, uint256 slippage) public {
    // ... other checks and logic ...

    // get reserve
    (uint256 VNDCReserve, uint256 currencyReserve) = getReserve(_currency);
    // Get amount out
    uint256 amountsOut = router.quote(_amount, currencyReserve, VNDCReserve);
    //...
    uint256 amountAMin = amountsOut.sub(amountsOut.mul(slippage).div(BASE_RATE));
    uint256 amountBMin = _amount.sub(_amount.mul(slippage).div(BASE_RATE));

    // ... further logic and minting process ...
}

function withdrawWhitelist(uint256 _amountVNDC, address _currency, uint256 _slippage)
public {
    // ... other checks and logic ...

    uint256 amountLiquidity = getAmountLiquidity(_amountVNDC, msg.sender, _currency);
    (uint256 VNDCAmount, uint256 currencyAmount) = estimateAmountByLp(_currency,
amountLiquidity);
    uint256 amountAMin = VNDCAmount.sub(VNDCAmount.mul(_slippage).div(BASE_RATE)); //
```

```
slippage
    uint256 amountBMin = currencyAmount.sub(currencyAmount.mul(_slippage).div(BASE_RATE));
// slippage

    // ... further logic and withdrawal process ...
}
```

Exploit Scenario:

- **Front-Running**: The attacker submits a transaction to manipulate the reserves by trading the token pairs involved.
- **Victim's Transaction**: The victim's transaction executes with the manipulated price, resulting in a less favorable exchange rate due to on-chain slippage calculation.
- **Back-Running**: The attacker submits another transaction to revert the price manipulation, profiting from the price difference.

RECOMMENDATION

The minimum amount should be calculated off-chain and passed as a parameter to the `mintWhitelist` and `withdrawWhitelist` functions.

```
function mintWhitelist(uint256 _amount, address _currency, uint256 amountAMin, uint256
amountBMin) public {}
function withdrawWhitelist(uint256 _amountVNDC, address _currency, uint256 amountAMin,
uint256 amountBMin) public {}
```

UPDATES

- ***Aug 06, 2024***: The issue is acknowledged by the VNDC 2.0 team.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Aug 06, 2024*	Public Report	Verichains Lab

Table 6. Report versions history