CRACKEN

Smart Contract Security Audit

Project: Narwhal

Jan 30, 2024

Contract Address

0xa299CBE101aAFb46362F78F3d2C5BB9E3B32C1aC

Table of Contents

Disclaimer

The contents of this report reflect only the CRACKEN TECH audit team's understanding of the current progress and status of the security of the code audited, to verify the integrity of the code provided for the scope of this audit. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Given the size of the project, the findings detailed here are not to be considered exhaustive, and further testing and audit are recommended after the issues covered are fixed. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report.

The review does not address the compiler layer, any other areas beyond the programming language, or other programming aspects that could present security risks. If the audited source files are smart contract files, risks or issues introduced by using data feeds from off-chain sources are not extended by this review either.

Audit Review

The source code of the Narwhal Coin was audited in order to acquire a clear impression of how the project was implemented. The Cracken Tech audit team conducted in-depth research, analysis, and scrutiny, resulting in a series of observations. A detailed list of each issue found, and vulnerabilities in the source code will be included in the audit report. The problems and potential solutions are given in this report, we will identify common sources for such problems and comments for improvement.

The auditing process will follow a routine as special considerations by Cracken:

- Review of the specifications, sources, and instructions provided to Cracken to make sure the contract logic meets the intentions of the client without exposing the user's funds to risk.
- Manual review of the entire codebase by our experts, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
- Specification comparison is the process of checking whether the code does what the specifications, sources, and instructions provided to Cracken describe.
- Test coverage analysis determines whether the test cases are covering the code and how much code is exercised when we run the test cases.
- Symbolic execution is analyzing a program to determine what inputs cause each part of a program to execute.
- Reviewing the codebase to improve maintainability, security, and control based on the established industry and academic practices.

Project Review

Token Summary

Parameter	Result
Token Name	NRW
Token Symbol	NRW
Token Decimal	18
Total Supply	1,000,000,000
Platform	BSC
Buy Tax Fee	3%
Sell Tax Fee	3%
Contract Creation Date	Jan 10, 2024
Liquidity Status	$532,154 USD
Liquidity Lockup Time	Unknown Lock
Compiler Version	v0.8.18+commit.87f61d96
Optimization	No with 200 runs
Contract Address	0xa299CBE101aAFb46362F78F3d2C5BB9E3B32C1aC
Deployer Address	0x1A09C98E8574b39484e093c3A9035506C2B99261
Owner Address	0xA1A418760E5427BeF0e19e77933C57C61E386368

Source Code

CRACKEN was commissioned by Narwhal Coin to perform an audit based on the following smart contract:

https://bscscan.com/token/0xa299cbe101aafb46362f78f3d2c5bb9e3b32c1ac#code

Smart Contract Vulnerability Checks

Vulnerability	Auto-Scan	Manual-Scan	Result
Unencrypted Private Data On-Chain	Complete	Complete	Low / No Risk
Code With No Effects	Complete	Complete	Low / No Risk
Message call with hardcoded gas amount	Complete	Complete	Low / No Risk
Hash Collisions with Multiple Variable Length Arguments	Complete	Complete	Low / No Risk
Unexpected Ether balance	Complete	Complete	Low / No Risk
Presence of unused variables	Complete	Complete	Low / No Risk
Right-To-Left-Override control character (U+202E)	Complete	Complete	Low / No Risk
Typographical Error	Complete	Complete	Low / No Risk
DoS With Block Gas Limit	Complete	Complete	Low / No Risk
Arbitrary Jump with Function Type Variable	Complete	Complete	Low / No Risk
Insufficient Gas Grieving	Complete	Complete	Low / No Risk
Incorrect Inheritance Order	Complete	Complete	Low / No Risk
Write to Arbitrary Storage Location	Complete	Complete	Low / No Risk
Requirement Violation	Complete	Complete	Low / No Risk
Missing Protection against Signature Replay Attacks	Complete	Complete	Low / No Risk
Weak Sources of Randomness from Chain Attributes	Complete	Complete	Low / No Risk
Authorization through tx. origin	Complete	Complete	Low / No Risk
Delegate call to Untrusted Callee	Complete	Complete	Low / No Risk

Vulnerability	Auto-Scan	Manual-Scan	Result
Use of Deprecated Solidity Functions	Complete	Complete	Low / No Risk
Assert Violation	Complete	Complete	Low / No Risk
Reentrancy	Complete	Complete	Low / No Risk
Unprotected SELF-DESTRUCT Instruction	Complete	Complete	Low / No Risk
Unprotected Ether Withdrawal	Complete	Complete	Low / No Risk
Outdated Compiler Version	Complete	Complete	Low / No Risk
Integer Overflow and Underflow	Complete	Complete	Low / No Risk
Function Default Visibility	Complete	Complete	Low / No Risk

Manual Code Review

Classification of Issues

Severity	Description
High-Risk	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.
Medium-Risk	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.
Low-Risk	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.
Informational	A vulnerability that has an informational character but is not affecting any of the code.

Findings

Severity	Found
High-Risk	1
Medium-Risk	0
Low-Risk	0
Informational	0
Total	1

High-Risk: functions make cause the rug or scam project. Must be fixed.

Set max buy / sell tax fee

Description:

The owner can change the buy & sell fees up to 100%.

[HIGH-RISK]

```
function _transfer(address sender, address recipient, uint256 amount) internal {
    // require(sender != address(0), "BEP20: transfer from the zero address");
    // require(recipient != address(0), "BEP20: transfer to the zero address");
    if (sender == _pairAddress || recipient == _pairAddress) {
        (bool isAdd, bool isDel) = _isLiquidity(sender, recipient);
        if (isAdd || isDel) {
            _balances[sender] = _balances[sender].sub(amount, "BEP20: transfer amount
exceeds balance");
            _balances[recipient] = _balances[recipient].add(amount);
            emit Transfer(sender, recipient, amount);
            return;
        }
    }
    if (sender == _pairAddress && !_isExcluded[recipient]) {
        uint256 ratio = _buyFeeRatio;
        uint256 fee = amount.mul(ratio).div(100);
        uint256 subAmount = amount.sub(fee);
        _balances[sender] = _balances[sender].sub(amount, "BEP20: transfer amount
exceeds balance");
        if (subAmount > 0) {
            _balances[recipient] = _balances[recipient].add(subAmount);
            emit Transfer(sender, recipient, subAmount);
        }
        uint256 subFeeAmount = fee;
```

```
        if (fee > 0) {
            if (_feeBurnRatio > 0) {
                uint256 burnFee = amount.mul(_feeBurnRatio).div(100);
                _balances[deadAddress] = _balances[deadAddress].add(burnFee);
                emit Transfer(sender, deadAddress, burnFee);
                subFeeAmount = subFeeAmount.sub(burnFee);
            }
            if (subFeeAmount > 0) {
                _balances[_feeNodeReceiver] =
_balances[_feeNodeReceiver].add(subFeeAmount);
                emit Transfer(sender, _feeNodeReceiver, subFeeAmount);
            }
        }
        emit Swap(sender, recipient, amount, fee, subFeeAmount);
    } else if (recipient == _pairAddress && !_isExcluded[sender]) {
        uint256 ratio = _sellFeeRatio;
        uint256 fee = amount.mul(ratio).div(100);
        uint256 subAmount = amount.sub(fee);
        _balances[sender] = _balances[sender].sub(amount, "BEP20: transfer amount
exceeds balance");
        if (subAmount > 0) {
            _balances[recipient] = _balances[recipient].add(subAmount);
            emit Transfer(sender, recipient, subAmount);
        }
        uint256 subFeeAmount = fee;
        if (fee > 0) {
            if (_feeBurnRatio > 0) {
                uint256 burnFee = amount.mul(_feeBurnRatio).div(100);
                _balances[deadAddress] = _balances[deadAddress].add(burnFee);
                emit Transfer(sender, deadAddress, burnFee);
```

```
                subFeeAmount = subFeeAmount.sub(burnFee);
            }
            if (subFeeAmount > 0) {
                _balances[_feeNodeReceiver] =
_balances[_feeNodeReceiver].add(subFeeAmount);
                emit Transfer(sender, _feeNodeReceiver, subFeeAmount);
            }
        }
        emit Swap(sender, recipient, amount, fee, subFeeAmount);
    } else {
        _balances[sender] = _balances[sender].sub(amount, "BEP20: transfer amount
exceeds balance");
        _balances[recipient] = _balances[recipient].add(amount);
        emit Transfer(sender, recipient, amount);
    }
  }
```

Recommendation:

We recommend adding a requirement to limit the max fee amount.

Privileged Functions

onlyOwner

Function Name	Parameters	Visibility
approve	address spender, uint256 amount	External
burn	uint256 amount	Public
decreaseAllowance	address spender, uint256 subtractedValue	Public
increaseAllowance	address spender, uint256 addedValue	Public
renounceOwnership	none	Public
setBuyFeeRatio	uint256 ratio	Public
setDeadAddress	Address addr	External
setExcluded	address[] calldata accounts, bool flag	External
setFeeBurnRatio	uint256 ratio	Public
setFeeNodeReceiver	Address addr	Public
setPairAddress	address newAddress	External
setSellFeeRatio	uint256 ratio	Public
setTxAmount	uint256 apta	External
transferByLimitOnly	bool newValue	Public
transferFrom	bool _enabled	Public
transferOwnership	uint256 newMarketingFee	External

Contract Ownership

The contract ownership of Narwhal Coin is not currently being renounced. The ownership of the contract grants special powers to the protocol creators, making them the sole addresses that can call sensible ownable functions that may alter the state of the protocol.

The current owner is the address 0xA1A418760E5427BeF0e19e77933C57C61E386368 which can be viewed: HERE

The owner wallet has the power to call the functions displayed on the privileged functions list above, if the owner wallet is compromised these privileges could be exploited.

We recommend the team renounce ownership at the right timing if possible, or gradually migrate to a time lock with governing functionalities in respect of transparency and safety considerations.

Liquidity Overview

Liquidity Information

Parameter	Result
Pair Address	0xc8d3fd2f6e87a46f4fa1c8e11219850adab2b13d
Narwhal Reserves	243.89K Narwhal
USDT Reserves	528.40 USDT
Liquidity Value	$1.05M USD
Liquidity Ownership	Unknown Locked

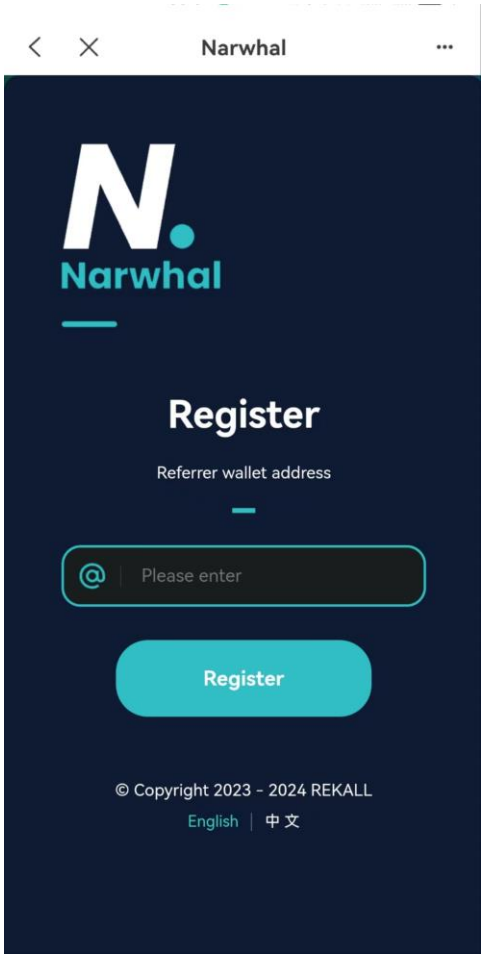
Tokenomics

Rank	Address	Quantity (Token)	Percentage
1	0xED342fE84528cFDB7ab82707FC45D2efa5384c08	999,700,000	99.9700%
2	PancakeSwap V2: BSC-USD-Narwhal	243,624.3980	0.0244%
3	0x2C5C8B485c8D2DbBCC81e2596321720a906a9633	50,509.7896	0.0051%
4	0x706d7Ec3ffEA75D4ed9C797369753298C8C68280	3,292.0257	0.0003%
5	0x9a7F1c1aDe1303101073D506427C298137BeaAf1	915.3712	0.0001%

Social Media Check

Social Media Type	Link	Result
Website	https://www.Narwhal.ac/	Checked
Twitter	https://twitter.com/narwhal_fyi/	Checked

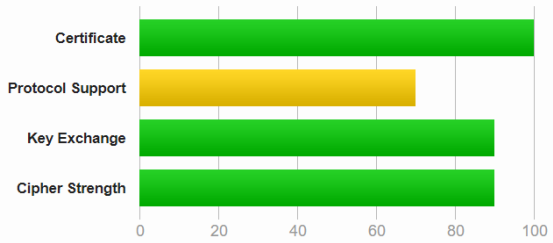
Website Review



Summary

Overall Rating

B



Visit our documentation page for more information, configuration guides, and books. Known issues are documented here.

This server supports TLS 1.1. Grade capped to B. MORE INFO »

This site works only in browsers with SNI support.

This server supports TLS 1.3.

HTTP Strict Transport Security (HSTS) with long duration deployed on this server. MORE INFO »

Certificate #1: RSA 2048 bits (SHA256withRSA)

Server Key and Certificate #1

Subject	www.nrw.ac Fingerprint SHA256: 644132664f0ba7b86f46ff966b80da6e6e3887d66ef367609cb03c77a53f2b23 Pin SHA256: tE7CN0VZw6/ixxtDzzQIlfF+2CM1R5dw4WgYOwnND1M=
Common names	www.nrw.ac
Alternative names	www.nrw.ac
Serial Number	04e0087297a87dbf9db6a918873217e2dd3e
Valid from	Fri, 26 Jan 2024 11:15:05 UTC
Valid until	Thu, 25 Apr 2024 11:15:04 UTC (expires in 2 months and 25 days)
Key	RSA 2048 bits (e 65537)
Weak key (Debian)	No
Issuer	R3 AIA: http://r3.i.lencr.org/
Signature algorithm	SHA256withRSA
Extended Validation	No
Certificate Transparency	Yes (certificate)
OCSP Must Staple	No
Revocation information	OCSP OCSP: http://r3.o.lencr.org
Revocation status	Good (not revoked)
DNS CAA	No (more info)
Trusted	Yes Mozilla Apple Android Java Windows

- Mobile Friendly
- Contains no code errors
- SSL is not secured
- No spelling errors
- Can not connect the web3 wallet

Audit Conclusion

- The owner cannot pause trading
- The owner cannot mint new tokens
- The owner can set the max transaction amount
- **The owner can change the buy/sell fee up to 100% [High-Risk]**
- The owner cannot set wallet max limit
- The owner cannot blacklist wallets

(All functions cannot be used if the ownership is renounced)

AUDIT IS PASSED