Coinsult

Request your audit at coinsult.net

Advanced Manual
Smart Contract Audit

November 3, 2022

CoinsultAudits

info@coinsult.net

coinsult.net

Audit requested by

NEB Storage

0x0194a9e46fee48b85e1cfafdb9d3ff9c4f2548ec

Table of Contents

Audit Summary

Project Name	NEB Storage
Website	https://nebank.io/#/home
Blockchain	Binance Smart Chain
Smart Contract Language	Solidity
Contract Address	0x0194a9e46fee48b85e1cfafdb9d3ff9c4f2548ec
Audit Method	Static Analysis, Manual Review
Date of Audit	3 November 2022

This audit report has been prepared by Coinsult's experts at the request of the client. In this audit, the results of the static analysis and the manual code review will be presented. The purpose of the audit is to see if the functions work as intended, and to identify potential security issues within the smart contract.

The information in this report should be used to understand the risks associated with the smart contract. This report can be used as a guide for the development team on how the contract could possibly be improved by remediating the issues that were identified.

Audit Scope

Source Code

Coinsult was comissioned by NEB Storage to perform an audit based on the following code:

https://bscscan.com/address/0x0194a9e46fee48b85e1cfafdb9d3ff9c4f2548ec#code

Note that we only audited the code available to us on this URL at the time of the audit. If the URL is not from any block explorer (main net), it may be subject to change. Always check the contract address on this audit report and compare it to the token you are doing research for.

Tokenomics

Not available

Audit Method

Coinsult’s manual smart contract audit is an extensive methodical examination and analysis of the smart contract’s code that is used to interact with the blockchain. This process is conducted to discover errors, issues and security vulnerabilities in the code in order to suggest improvements and ways to fix them.

Automated Vulnerability Check

Coinsult uses software that checks for common vulnerability issues within smart contracts. We use automated tools that scan the contract for security vulnerabilities such as integer-overflow, integer-underflow, out-of-gas-situations, unchecked transfers, etc.

Manual Code Review

Coinsult’s manual code review involves a human looking at source code, line by line, to find vulnerabilities. Manual code review helps to clarify the context of coding decisions. Automated tools are faster but they cannot take the developer’s intentions and general business logic into consideration.

Used Tools

- Slither: Solidity static analysis framework
- Remix: IDE Developer Tool
- CWE: Common Weakness Enumeration
- SWC: Smart Contract Weakness Classification and Test Cases
- DEX: Testnet Blockchains

Risk Classification

Coinsult uses certain vulnerability levels, these indicate how bad a certain issue is. The higher the risk, the more strictly it is recommended to correct the error before using the contract.

Vulnerability Level	Description
● Informational	Does not compromise the functionality of the contract in any way
● Low-Risk	Won't cause any problems, but can be adjusted for improvement
● Medium-Risk	Will likely cause problems and it is recommended to adjust
● High-Risk	Will definitely cause problems, this needs to be adjusted

Coinsult has four statuses that are used for each risk level. Below we explain them briefly.

Risk Status	Description
Total	Total amount of issues within this category
Pending	Risks that have yet to be addressed by the team
Acknowledged	The team is aware of the risks but does not resolve them
Resolved	The team has resolved and remedied the risk

Disclaimer

This audit report has been prepared by Coinsult's experts at the request of the client. In this audit, the results of the static analysis and the manual code review will be presented. The purpose of the audit is to see if the functions work as intended, and to identify potential security issues within the smart contract.

The information in this report should be used to understand the risks associated with the smart contract. This report can be used as a guide for the development team on how the contract could possibly be improved by remediating the issues that were identified.

Coinsult is not responsible if a project turns out to be a scam, rug-pull or honeypot. We only provide a detailed analysis for your own research.

Coinsult is not responsible for any financial losses. Nothing in this contract audit is financial advice, please do your own research.

The information provided in this audit is for informational purposes only and should not be considered investment advice. Coinsult does not endorse, recommend, support or suggest to invest in any project.

Coinsult can not be held responsible for when a project turns out to be a rug-pull, honeypot or scam.

Global Overview

Manual Code Review

In this audit report we will highlight the following issues:

Vulnerability Level	Total	Pending	Acknowledged	Resolved
🔵 Informational	0	0	0	0
🟢 Low-Risk	3	0	3	0
🟡 Medium-Risk	2	0	2	0
🔴 High-Risk	0	0	0	0

Error Code	Description
SLT: 056	Missing Zero Address Validation

● **Low-Risk:** Could be fixed, will not bring problems.

No zero address validation for some functions

Detect missing zero address validation.

```
function addDev(address newDev) public onlyOwner {
    (bool flag, ) = isDev(newDev);
    require(!flag, "GValidator: already dev");
    _devs.push(newDev);
}
```

Recommendation

Check that the new address is not zero.

Exploit scenario

```
contract C {

  modifier onlyAdmin {
    if (msg.sender != owner) throw;
    _;
  }

  function updateOwner(address newOwner) onlyAdmin external {
    owner = newOwner;
  }
}
```

Bob calls `updateOwner` without specifying the `newOwner`, soBob loses ownership of the contract.

Error Code	Description
SWC-104	CWE-252: Unchecked Return Value

● **Low-Risk:** Could be fixed, will not bring problems.

Unchecked transfer

The return value of an external transfer/transferFrom call is not checked.

```
function withdrawByToken(
    address to,
    address token,
    uint256 amount
) public onlyOwner {
    if (token == address(0)) {
        payable(to).transfer(amount);
    } else {
        IERC20(token).safeTransfer(to, amount);
    }
}
```

Recommendation

Use `SafeERC20`, or ensure that the transfer/transferFrom return value is checked.

Exploit scenario

```
contract Token {
    function transferFrom(address _from, address _to, uint256 _value) public returns (bool success);
}
contract MyBank{
    mapping(address => uint) balances;
    Token token;
    function deposit(uint amount) public{
        token.transferFrom(msg.sender, address(this), amount);
        balances[msg.sender] += amount;
    }
}
```

Several tokens do not revert in case of failure and return false. If one of these tokens is used in `MyBank`, `deposit` will not revert if the transfer fails, and an attacker can call `deposit` for free..

Error Code	Description
SLT: 054	Missing Events Arithmetic

● **Low-Risk:** Could be fixed, will not bring problems.

Missing events arithmetic

Detect missing events for critical arithmetic parameters.

```
function setToken(uint256 tid, address token) public onlyOwner {
    require(validTid(tid), "tid invalid");
    _tokenList[tid] = token;
}
```

Recommendation

Emit an event for critical parameter changes.

Exploit scenario

```
contract C {

  modifier onlyAdmin {
    if (msg.sender != owner) throw;
    _;
  }

  function updateOwner(address newOwner) onlyAdmin external {
    owner = newOwner;
  }
}
```

`updateOwner()` has no event, so it is difficult to track off-chain changes in the buy price.

Error Code	Description
CSM-01	Potential OoG (Out-of-Gas) situation in addTokens()

● **Medium-Risk:** Should be fixed, could bring problems.

Potential OoG (Out-of-Gas) situation in addTokens()

```
function addTokens(address[] memory tokens) public onlyOwner {
    for (uint256 i = 0; i &lt; tokens.length; i++) {
        _tokenList.push(tokens[i]);
    }
}
```

Recommendation

Caution is advised when you expect to have large arrays that grow over time. Actions that require looping across the entire data structure should be avoided. If you absolutely must loop over an array of unknown size, then you should plan for it to potentially take multiple blocks, and therefore require multiple transactions.

Error Code	Description
CSM-02	Centralization Risk

● **Medium-Risk:** Should be fixed, could bring problems.

Centralization Risk

```
function deposit(uint256 amount, uint256 tid) public {

function withdraw(uint256 amount, uint256 tid, address to) public onlyDevOrOwner {
```

Recommendation

Everyone can deposit, but only dev or owner address can withdraw funds

Other Owner Privileges Check

Error Code	Description
CEN-100	Centralization: Operator Priviliges

Coinsult lists all important contract methods which the owner can interact with.

Owner can set cold wallet

Owner can withdraw tokens from contract

Notes

Notes by NEB Storage

No notes provided by the team.

Notes by Coinsult

No notes provided by Coinsult

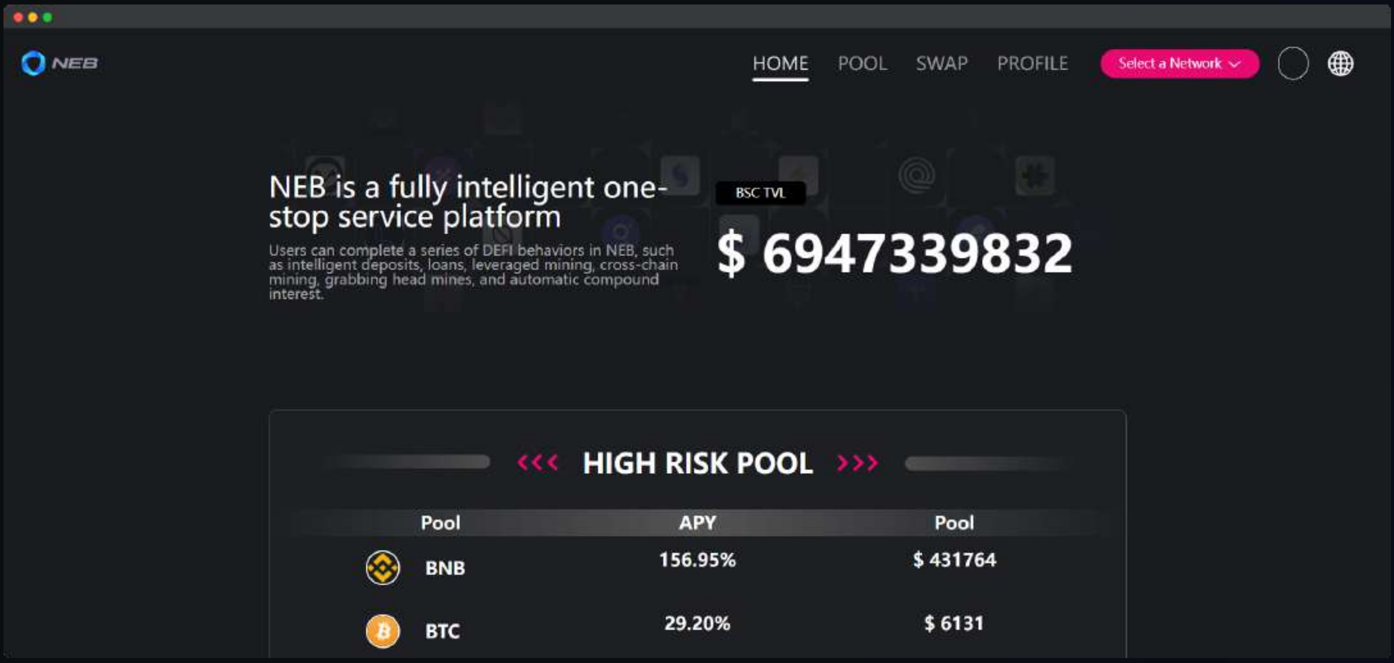
Contract Snapshot

This is how the constructor of the contract looked at the time of auditing the smart contract.

```
pragma solidity ^0.8.0;

contract Storage is Validator {
    using SafeERC20 for IERC20;

    address public coldWallet;
    address[] private _tokenList;

    event DepositEvent(address indexed account, uint256 tid, uint256 amount);
    event WithdrawEvent(address indexed account, uint256 tid, uint256 amount);

    constructor(address coldWallet_) {
        coldWallet = coldWallet_;
        _tokenList.push(address(0));
    }

    receive() external payable {
        if (msg.sender == coldWallet) {
            return;
        }
        uint256 amount = msg.value;
        require(amount &gt; 0, "amount invalid");
        payable(coldWallet).transfer(amount);
        emit DepositEvent(_msgSender(), 0, amount);
    }

    function validTid(uint256 tid) public view returns (bool) {
        return tid  0, "erc20 must use tid &gt; 0");
        IERC20(_tokenList[tid]).safeTransferFrom(
            _msgSender(),
            address(this),
            amount
        );
        IERC20(_tokenList[tid]).safeTransfer(coldWallet, amount);
        emit DepositEvent(_msgSender(), tid, amount);
    }
```

Website Review

Coinsult checks the website completely manually and looks for visual, technical and textual errors. We also look at the security, speed and accessibility of the website. In short, a complete check to see if the website meets the current standard of the web development industry.



NEB is a fully intelligent one-stop service platform

Users can complete a series of DEFI behaviors in NEB, such as intelligent deposits, loans, leveraged mining, cross-chain mining, grabbing head mines, and automatic compound interest.

$ 6947339832

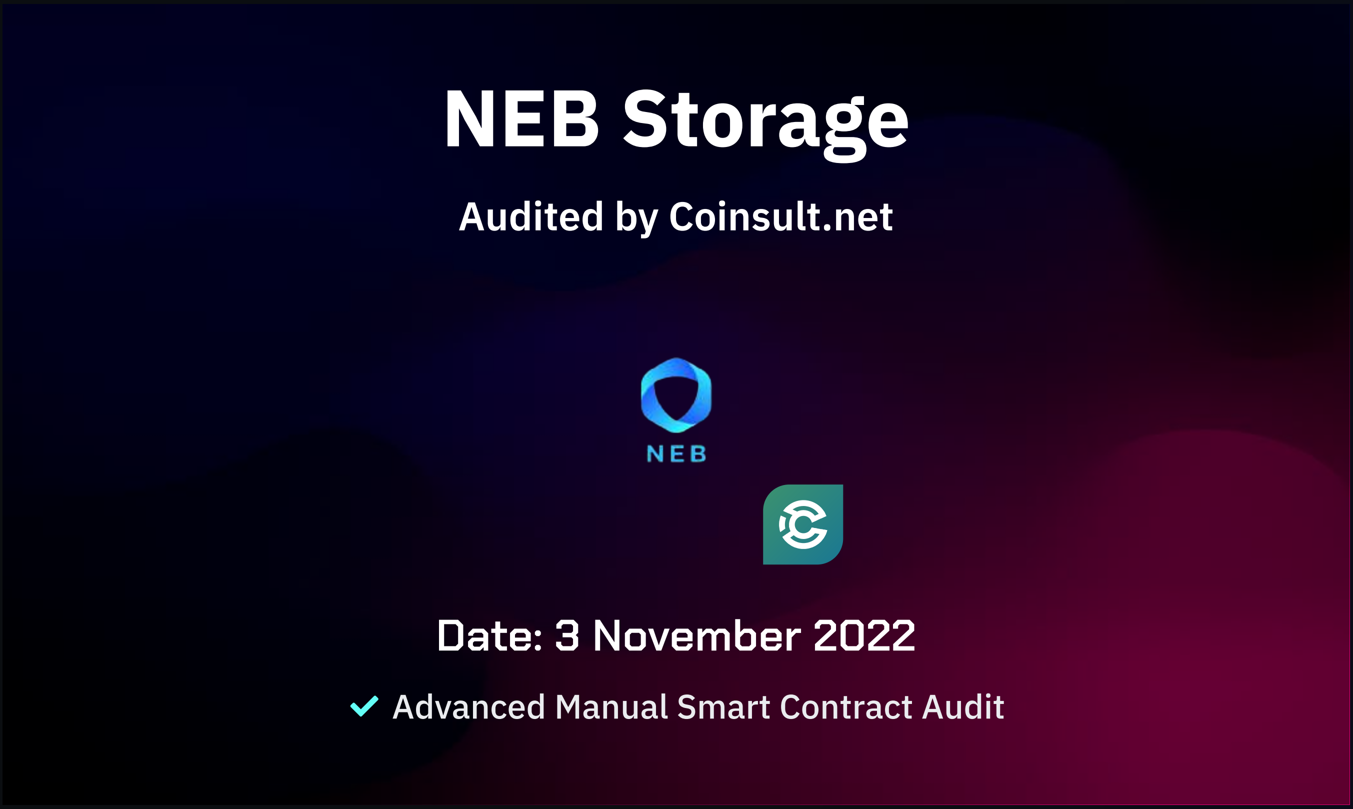
<<< HIGH RISK POOL >>>

Pool	APY	Pool
BNB	156.95%	$ 431764
BTC	29.20%	$ 6131

Type of check	Description
Mobile friendly?	● The website is mobile friendly
Contains jQuery errors?	● The website does not contain jQuery errors
Is SSL secured?	● The website is SSL secured
Contains spelling errors?	● The website does not contain spelling errors

Certificate of Proof

● Not KYC verified by Coinsult



NEB Storage

Audited by Coinsult.net

Date: 3 November 2022

✓ Advanced Manual Smart Contract Audit

Coinsult

coinsult.net

End of report
Smart Contract Audit

CoinsultAudits

info@coinsult.net

coinsult.net

Request your smart contract audit / KYC

t.me/coinsult_tg