



Smart Contract Audit Report

Knoknok Smart Contract

20 Dec 2022



Table of Content

1 Executive Summary	2
Methodology	2
2 Findings Overview	5
2.1 Project info and Contract address	6
2.2 Summary	6
2.3 Key Findings	7
3 Detailed Description of Findings	7
3.1 Token burn function	8
3.2 Mint unlimited	9
3.3 Repeatedly set the vault function	10
3.4 RevokeMinter function	11
3.5 Deposit and withdraw function	12
4 Conclusion	15
5 Appendix	16
5.1 Basic Coding Assessment	16
5.2 Advanced Code Scrutiny	17
6 Disclaimer	19
References	20



1 EXECUTIVE SUMMARY

Numen Cyber Technology was engaged by Knoknok to review smart contract implementation. The assessment was conducted in accordance with our systematic approach to evaluate potential security issues based upon customer requirement. The report provides detailed recommendations to resolve the issue and provide additional suggestions or recommendations for improvement.

Five severity findings are related to business logistics, where are one medium severity, two low severity and two Informational severity.

The outcome of the assessment outlined in chapter 3 provides the system's owners a full description of the vulnerabilities identified, the associated risk rating for each vulnerability, and detailed recommendations that will resolve the underlying technical issue.

METHODOLOGY

To standardize the evaluation, we define the following terminology based on OWASP Risk Rating Methodology [10] which is the gold standard in risk assessment using the following risk models:

- Likelihood: represents how likely a particular vulnerability is to be uncovered and exploited in the wild.
- Impact: measures the technical loss and business damage of a successful attack.
- Severity: determine the overall criticality of the risk.

Likelihood and impact are categorized into three ratings: High, Medium and Low. Severity is determined by likelihood and impact and can be classified into four categories accordingly, Critical, High, Medium, Low shown in table 1.1.



Table 1.1: Overall Risk Severity

To evaluate the risk, we will be going through a list of items, and each would be labelled with a severity category. The audit was performed with a systematic approach guided by a comprehensive assessment list carefully designed to identify known and impactful security issues. If our tool or analysis does not identify any issue, the contract can be considered safe regarding the assessed item. For any discovered issue, we might further deploy contracts on our private test environment and run tests to confirm the findings. If necessary, we would additionally build a PoC to demonstrate the possibility of exploitation. The concrete list of check items is shown in Table 1.2.

- **Basic Coding Bugs:** We first statically analyze given smart contracts with our proprietary static code analyzer for known coding bugs, and then manually verify (reject or confirm) all the issues found by our tool.
- **Code and business security testing:** We further review business logics, examine system operations, and place DeFi-related aspects under scrutiny to uncover possible pitfalls and/or bugs.
- **Additional Recommendations:** We also provide additional suggestions regarding the coding and development of smart contracts from the perspective of proven programming practices.

Category	Assessment Item
-----------------	------------------------



Basic Coding Assessment	Apply Verification Control
	Authorization Access Control
	Forged Transfer Vulnerability
	Forged Transfer Notification
	Numeric Overflow
	Transaction Rollback Attack
	Transaction Block Stuffing Attack
	Soft fail Attack
	Hard fail Attack
	Abnormal Memo
	Abnormal Resource Consumption
	Secure Random Number
Advanced Source Code Scrutiny	Asset Security
	Cryptography Security
	Business Logic Review
	Source Code Functional Verification
	Account Authorization Control
	Sensitive Information Disclosure
	Circuit Breaker



	Blacklist Control
	System API Call Analysis
	Contract Deployment Consistency Check
Additional Recommendations	Semantic Consistency Checks
	Following Other Best Practices

Table 1.2: The Full List of Assessment Items

To better describe each issue we identified, we categorize the findings with Common Weakness Enumeration (CWE-699) [14], which is a community-developed list of software weakness types to better delineate and organize weaknesses around concepts frequently encountered in software development.

2 FINDINGS OVERVIEW



2.1 PROJECT INFO AND CONTRACT ADDRESS

Project Name: Knoknok

Project URL: NULL

Audit Time: 2022/12.15 - 2022/12.20


Language: solidity

Source Code Link	Commit Hash
https://github.com/Knoknok/chaindata/blob/main/AnyswapV6ERC20.sol	489daf31ce76db8ff59f9053db0397c11abcbf0b



Token Info:

Token Name	Fill in after deploy
Token Symbol	Fill in after deploy
Decimals	Fill in after deploy
TotalSupply	Fill in after deploy
Token Type	BEP20

2.2 SUMMARY

Severity	Found	
Critical	0	
High	0	
Medium	1	



Low	2	
Informational	2	

2.3 KEY FINDINGS

Five severity findings are related to business logistics, where are one medium severity, two low severity and two Informational severity.

ID	Severity	Findings Title	Status	Confirm
NVE-001	Medium	Token burn function	Ignore	Confirmed
NVE-002	Low	Mint unlimited	Ignore	Confirmed
NVE-003	Low	Similar Mechanisms on set the vault function	Ignore	Confirmed
NVE-004	Informational	revokeMinter function	Ignore	Confirmed
NVE-005	Informational	Deposit and withdraw function	Ignore	Confirmed

Table 2.1: Key Audit Findings

3 DETAILED DESCRIPTION OF FINDINGS



3.1 TOKEN BURN FUNCTION

ID: NVE-001

Location: AnyswapV6ERC20.sol

Severity: Medium

Category: Business Issues

Likelihood: Medium

Impact: Medium

Description:

As shown in figure 1 ,2 below, the addresses with minter permission can call the “burn” function to burn the tokens of the specified amount "amount" in the specified address "from", it may affect the security of user assets.

```
function burn(address from, uint256 amount) external onlyAuth returns (bool) {  
    _burn(from, amount);  
    return true;  
}
```

Figure 1 burn function

```
function _burn(address account, uint256 amount) internal {  
    require(account != address(0), "ERC20: burn from the zero address");  
  
    uint256 balance = balanceOf[account];  
    require(balance >= amount, "ERC20: burn amount exceeds balance");  
  
    balanceOf[account] = balance - amount;  
    _totalSupply -= amount;  
    emit Transfer(account, address(0), amount);  
}
```

Figure 2 _burn function

Recommendations:



Numen Cyber Lab recommends to modify code logic.

Result: Pass

Fix Result:

Ignore (After discussion with the project party, who has confirmed that this function is necessary according to project design and this function will be used reasonably).

3.2 MINT UNLIMITED

ID: NVE-002

Location: AnyswapV6ERC20.sol

Severity: Low

Category: Business Issues

Likelihood: Low

Impact: Low

Description:

As shown in Figure 3,4 below, the address with minter permissions can call the “*mint*” function to mint the specified amount of tokens “amount” to the specified address “to”, and there is no limitation.

```
function mint(address to, uint256 amount) external onlyAuth returns (bool) {  
    | _mint(to, amount);  
    | return true;  
    | }  
}
```

Figure 3 mint function



```
function _mint(address account, uint256 amount) internal {  
    require(account != address(0), "ERC20: mint to the zero address");  
  
    _totalSupply += amount;  
    balanceOf[account] += amount;  
    emit Transfer(address(0), account, amount);  
}
```

Figure 4 _mint function

Recommendations:

Numen Cyber Lab recommends to set token cap.

Result: Pass

Fix Result:

Ignore (After discussion with the project party, who has confirmed that this function is necessary according to project design and this function will be used reasonably).

3.3 SIMILAR MECHANISMS ON SET THE VAULT FUNCTION

ID: NVE-003

Location: AnyswapV6ERC20.sol

Severity: Low

Category: Business Issues

Likelihood: Low

Impact: Low

Description:

As shown in Figure 5, 6 below, there are two similar mechanisms when modify the "vault" address which is not necessary. One mechanism is to call the "setVault" function to delay modification for two days, and the other one is to call the "changeVault" function to modify immediately.

```
function setVault(address _vault) external onlyVault {
    require(_vault != address(0), "AnyswapV6ERC20: address(0)");
    pendingVault = _vault;
    delayVault = block.timestamp + DELAY;
}

function applyVault() external onlyVault {
    require(pendingVault != address(0) && block.timestamp >= delayVault);
    vault = pendingVault;

    pendingVault = address(0);
    delayVault = 0;
}
```

Figure 5 the part of code

```
function changeVault(address newVault) external onlyVault returns (bool) {
    require(newVault != address(0), "AnyswapV6ERC20: address(0)");
    emit LogChangeVault(vault, newVault, block.timestamp);
    vault = newVault;
    pendingVault = address(0);
    delayVault = 0;
    return true;
}
```

Figure 6 changeVault function

Recommendations:

Numen Cyber Lab recommends to keep one mechanism only.

Result: Pass

Fix Result:

Ignore (After discussion with the project party, who has confirmed that this function is necessary according to project design and this function will be used reasonably).

3.4 REVOKE MINTER FUNCTION



ID: NVE-004

Location: AnyswapV6ERC20.sol

Severity: Informational

Category: Business Issues

Likelihood: Informational

Impact: Informational

Description:

As shown in Figure 7 below, When the “vault” address calls the “*revokeMinter*” function to remove the minter permission of the specified address “_auth”, only the status is changed to false, and “_auth” is not removed from the minters array.

```
function revokeMinter(address _auth) external onlyVault {
    | isMinter[_auth] = false;
}

function getAllMinters() external view returns (address[] memory) {
    | return minters;
}
```

Figure 7 *revokeMinter* function

Result: Pass

Fix Result:

Ignore (After discussion with the project party, the minters array stores all authorized minter addresses, so there is security concern).

3.5 DEPOSIT AND WITHDRAW FUNCTION

ID: NVE-005

Location: AnyswapV6ERC20.sol

Severity: Informational

Category: Business Issues

Likelihood: Informational



Impact: Informational

Description:

As shown in Figure 8 below, according to the design of the project party, as long as the underlying is 0 address, the deposit and withdraw functions cannot be used, and there is no security risk.

```
function deposit() external returns (uint) {
    uint _amount = IERC20(underlying).balanceOf(msg.sender);
    IERC20(underlying).safeTransferFrom(msg.sender, address(this), _amount);
    return _deposit(_amount, msg.sender);
}

function deposit(uint amount) external returns (uint) {
    IERC20(underlying).safeTransferFrom(msg.sender, address(this), amount);
    return _deposit(amount, msg.sender);
}

function deposit(uint amount, address to) external returns (uint) {
    IERC20(underlying).safeTransferFrom(msg.sender, address(this), amount);
    return _deposit(amount, to);
}

function depositVault(uint amount, address to) external onlyVault returns (uint) {
    return _deposit(amount, to);
}

function _deposit(uint amount, address to) internal returns (uint) {
    require(!underlyingIsMinted);
    require(underlying != address(0) && underlying != address(this));
    _mint(to, amount);
    return amount;
}

function withdraw() external returns (uint) {
    return _withdraw(msg.sender, balanceOf[msg.sender], msg.sender);
}

function withdraw(uint amount) external returns (uint) {
    return _withdraw(msg.sender, amount, msg.sender);
}
```



Figure 8 the part of code

Result: Pass

Fix Result:

Ignore(After discussion with the project party, as long as the underlying is 0 address, the deposit and withdraw functions cannot be used, and there is no security risk).



4 CONCLUSION

In this audit, we thoroughly analyzed Knoknok smart contract implementation. The problems found are described and explained in detail in Section 3. The problems found in the audit have been brought up to the project party, ignored issues are in line with the project design, and permissions are only used for the project to properly function. We therefore deem the audit result to be a **PASS**. To improve this report, we greatly appreciate any constructive feedbacks or suggestions, on our methodology, audit findings, or potential gaps in scope/coverage.

5 APPENDIX

5.1 BASIC CODING ASSESSMENT

5.1.1 Apply Verification Control

- Description: The security of apply verification
- Result: Not found
- Severity: **Critical**

5.1.2 Authorization Access Control

- Description: Permission checks for external integral functions
- Result: Not found
- Severity: **Critical**

5.1.3 Forged Transfer Vulnerability

- Description: Assess whether there is a forged transfer notification vulnerability in the contract
- Result: Not found
- Severity: **Critical**

5.1.4 Transaction Rollback Attack

- Description: Assess whether there is transaction rollback attack vulnerability in the contract.
- Result: Not found
- Severity: **Critical**

5.1.5 Transaction Block Stuffing Attack

- Description: Assess whether there is transaction blocking attack vulnerability.
- Result: Not found
- Severity: **Critical**

5.1.6 soft fail Attack Assessment

- Description: Assess whether there is soft fail attack vulnerability.
- Result: Not found
- Severity: **Critical**

5.1.7 hard fail Attack Assessment

- Description: Examine for hard fail attack vulnerability
- Result: Not found
- Severity: **Critical**

5.1.8 Abnormal Memo Assessment



- Description: Assess whether there is abnormal memo vulnerability in the contract.
- Result: Not found
- Severity: **Critical**

5.1.9 Abnormal Resource Consumption

- Description: Examine whether abnormal resource consumption in contract processing.
- Result: Not found
- Severity: **Critical**

5.1.10 Random Number Security

- Description: Examine whether the code uses insecure random number.
- Result: Not found
- Severity: **Critical**

5.2 ADVANCED CODE SCRUTINY

5.2.1 Cryptography Security

- Description: Examine for weakness in cryptograph implementation.
- Results: Not Found
- Severity: **High**

5.2.2 Account Permission Control

- Description: Examine permission control issue in the contract
- Results: Not Found
- Severity: **Medium**

5.2.3 Malicious Code Behaviour

- Description: Examine whether sensitive behaviour present in the code
- Results: Not found
- Severity: **Medium**

5.2.4 Sensitive Information Disclosure



- Description: Examine whether sensitive information disclosure issue present in the code.
- Result: Not found
- Severity: **Medium**

5.2.5 System API

- Description: Examine whether system API application issue present in the code
- Results: Not found
- Severity: **Low**



6 DISCLAIMER

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to the Company in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes without Numen's prior written consent.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts Numen to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Numen's position is that each company and individual are responsible for their own due diligence and continuous security. Numen's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.



REFERENCES

[1] MITRE. CWE- 191: Integer Underflow (Wrap or Wraparound).

<https://cwe.mitre.org/data/definitions/191.html>.

[2] MITRE. CWE- 197: Numeric Truncation Error.

<https://cwe.mitre.org/data/definitions/197.html>.

[3] MITRE. CWE-400: Uncontrolled Resource Consumption.

<https://cwe.mitre.org/data/definitions/400.html>.

[4] MITRE. CWE-440: Expected Behavior Violation.

<https://cwe.mitre.org/data/definitions/440.html>.

[5] MITRE. CWE-684: Protection Mechanism Failure.

<https://cwe.mitre.org/data/definitions/693.html>.

[6] MITRE. CWE CATEGORY: 7PK - Security Features.

<https://cwe.mitre.org/data/definitions/254.html>.

[7] MITRE. CWE CATEGORY: Behavioral Problems.

<https://cwe.mitre.org/data/definitions/438.html>.

[8] MITRE. CWE CATEGORY: Numeric Errors.

<https://cwe.mitre.org/data/definitions/189.html>.

[9] MITRE. CWE CATEGORY: Resource Management Errors.

<https://cwe.mitre.org/data/definitions/399.html>.

[10] OWASP. Risk Rating Methodology.

https://www.owasp.org/index.php/OWASP_Risk_Rating_Methodology



Numen Cyber Technology Pte. Ltd.

11 North Buona Vista Drive, #04-09,
The Metropolis, Singapore 138589

Tel: 65-63555555

Fax: 65-63666666

Email: sales@numencyber.com

Web: <https://numencyber.com>