Smart Contract Audit Report

Hung Yuan KYC System - Security Assessment

Conducted by Buidler House

Date: February 10, 2025
Audit Period: February 15 - February 20, 2025

Table of Contents

1. Introduction

The Builder House team received a request to conduct a security audit on the **Hung Yuan KYC System** smart contract. The goal of this audit is to identify potential security vulnerabilities, assess the contract's security, and provide recommendations to enhance the security posture of the system.

Hunyuan is an on-chain KYC platform that aims to provide efficient, secure and compliant customer identity authentication services for enterprises and blockchain projects. As the global regulatory environment becomes increasingly stringent, many enterprises and project parties need to verify the identity of users before providing services to ensure that their customers are authentic and trustworthy individuals, thereby fulfilling legal and anti-money laundering regulations.

Our team follows a **white-box, black-box, and gray-box** testing strategy, closely simulating real-world attack scenarios to comprehensively evaluate the contract's security.

Audit Methodology

Method	Description
Black-Box Testing	Security testing from an external attacker's perspective without access to source code.
Gray-Box Testing	Security analysis using partial knowledge of the internal structure to find vulnerabilities.
White-Box Testing	A full-code review to identify security flaws, logic errors, and vulnerabilities.

Risk Classification

Risk Level	Impact
Critical	Can lead to a complete system compromise and major financial loss.
High	Affects system integrity, allowing unauthorized access or major exploitation.
Medium	Can cause partial security failures but does not immediately compromise the entire system.
Low	May result in limited impact but should still be addressed.
Minor/Recommendation	Not a security risk but recommended for optimization.

2. Audit Methodology

Our auditing process consists of two key phases:

1. **Automated Vulnerability Scanning**
    - Running security analysis tools to identify common smart contract vulnerabilities.
2. **Manual Code Review**
    - A detailed review of the source code to detect security weaknesses and logical flaws.

During this audit, we focused on vulnerabilities including but not limited to:

- **Reentrancy Attacks**
- **Replay Attacks**
- **Short Address Attack**
- **Denial-of-Service (DoS)**
- **Transaction Order Dependency**
- **Race Conditions**
- **Access Control Flaws**
- **Integer Overflow/Underflow**
- **Timestamp Dependency**
- **Unchecked Storage Pointers**
- **Arithmetic Precision Issues**
- **tx.origin Authentication Flaws**
- **Fake Deposit Exploits**
- **State Variable Visibility Issues**
- **Gas Optimization Concerns**
- **Malicious Event Emission**
- **Insecure External Calls**
- **Unprotected Function Modifiers**
- **Business Logic Flaws**
- **Unvalidated Input Handling**

Audit Schedule

Date	Task
Day 1	Environment setup & automated scanning
Day 2	Manual review of core business logic
Day 3	Access control & security mechanism validation
Day 4	Exploit reproduction & Proof of Concept (PoC) development
Day 5	Report documentation & fix verification

3. Project Overview

3.1 Project Description

The **Hung Yuan KYC System** is a decentralized identity verification solution built on the **HashKey Network**. It utilizes **Ethereum Name Service (ENS)** and **Soulbound Token (SBT)** technology to provide secure, non-transferable identity verification.

Key Features:

- **User Verification:**
  - Users initiate KYC verification using ENS names (e.g., `alice.hsk`).
  - Registration requires a fee to support system sustainability.
- **Admin Controls:**
  - Administrators can approve or revoke KYC status.
  - Supports multi-admin operations for better security.
- **KYC Management:**
  - KYC status is linked to ENS names using **Soulbound tokens**.
  - Supports multiple KYC levels (Basic, Advanced, Premium).
- **Security Measures:**
  - **Role-Based Access Control (RBAC)**
  - **Emergency Pause Mechanism** for security incidents.

Technical Implementation:

- **ENS Integration:**
  - Uses a custom `.hsk` TLD for ENS ownership verification.
  - Queries KYC status via ENS resolvers.
- **Smart Contract Architecture:**
  - **Primary Contract** manages KYC status and verification.
  - **Storage Contract** ensures data security and immutability.

3.2 Audited Smart Contract

- **Project Website:** kyc-testnet.hunyuankyc.com
- **Audited Code SHA-256:** `79ec15cc07bf4d699f94bad9c7462ebcd2736e282dd39799255370097f6a4844`
- **Github:** https://github.com/hunyuan-kyc/kyc-sbt-contract/commit/8a1f7696706a519d7445828d4887de9cd25bf2d4
-

4. Audit Findings

ID	Issue	Type	Severity	Status
N1	Timestamp Manipulation	Timestamp Dependency	Low	Fixed
N2	Missing Event Logs	Logging Issue	Recommendation	Ignored
N3	Excessive Admin Privileges	Access Control	Low	Confirmed
N4	Lack of Event Logging	Logging Issue	Medium	Pending Fix
N5	Timestamp Dependency	Timestamp Vulnerability	Low	Pending Fix
N6	Poor Access Control	Access Control Flaw	High	Pending Fix
N7	Missing Input Validation	Input Validation	Recommendation	Pending Fix
N8	Reentrancy Risk	Reentrancy	Medium	Pending Fix
N9	Variable Visibility Issues	Data Exposure	Low	Pending Fix
N10	Missing Error Handling	Error Handling	Recommendation	Pending Fix
N11	Business Logic Flaw	Logical Issue	Medium	Pending Fix
N12	Redundant Code	Code Optimization	Low	Pending Fix

4.1 Detailed Vulnerability Report

[N1] Timestamp Manipulation

- **Severity:** Low
- **Issue:** The contract uses `block.timestamp` for decision-making, which miners can manipulate slightly.
- **Code:**

```
solidity
CopyEdit
function requestKyc(string calldata ensName) external payable
whenNotPaused {
    require(ensName.length >= minNameLength, "KycSBT: Name too short");
    require(ensNameToAddress[ensName] == address(0), "KycSBT: Name
already registered");
    require(ensNameToAddress[ensName] == msg.sender, "KycSBT: Name not
owned by sender");
}
```

- **Fix:** Use an **oracle service** instead of `block.timestamp`.

[N2] Missing Event Logs

- **Severity:** Recommendation
- **Issue:** Functions like `setRegistrationFee()` lack event logging.
- **Code:**

```
solidity
CopyEdit
function setRegistrationFee(uint256 newFee) external onlyOwner {
    registrationFee = newFee;
}
```

- **Fix:** Emit an event after state changes.

```
solidity
CopyEdit
event RegistrationFeeUpdated(uint256 newFee);

function setRegistrationFee(uint256 newFee) external onlyOwner {
    registrationFee = newFee;
    emit RegistrationFeeUpdated(newFee);
}
```

[N3] Excessive Admin Privileges

- **Severity:** Low

- **Issue:** Certain functions can only be executed by the contract owner, increasing centralization risks.
- **Code:**

```
solidity
CopyEdit
function setENSAndResolver(address resolver) external onlyOwner {
    ensResolver = resolver;
}
```

- **Fix:** Implement **multi-signature (multisig) access control**.

[N4] Lack of Event Logging

- **Severity:** Medium
- **Issue:** `setEnsFee()` and `setRegistrationFee()` lack event logs.
- **Fix:** Add event logs for each function.

[N5] Timestamp Dependency

- **Severity:** Low
- **Issue:** `isValid()` relies on `block.timestamp`, which can be manipulated.
- **Fix:** Use **an oracle-based time source**.

[N6] Poor Access Control

- **Severity:** High
- **Issue:** `approveKyc()` is too centralized.
- **Code:**

```
solidity
CopyEdit
function approveKyc(address user) external onlyOwner {
    kycApproved[user] = true;
}
```

- **Fix:** Implement **RBAC (Role-Based Access Control)**.

[N7] Missing Input Validation

- **Severity:** Recommendation
- **Issue:** The `setSuffix()` function does not validate inputs.
- **Code:**

```
solidity
CopyEdit
function setSuffix(string memory newSuffix) external onlyOwner {
    suffix = newSuffix;
}
```

- **Fix:** Add **format validation**.

[N8] Reentrancy Risk

- **Severity:** Medium
- **Issue:** `withdrawFees()` can be **exploited** by reentrancy attacks.
- **Code:**

```
solidity
CopyEdit
function withdrawFees() external onlyOwner {
    (bool success, ) = owner().call{value: address(this).balance}("");
    require(success, "Transfer failed");
}
```

- **Fix:**
  - Follow **Checks-Effects-Interactions** pattern.
  - Use **ReentrancyGuard**.

[N9] Variable Visibility Issues

- **Severity:** Low
- **Issue:** Variables are public but should be **private**.
- **Code:**

```
solidity
CopyEdit
mapping(string => address) public ensNameToAddress;
```

- **Fix:** Change to **private**.

[N10] Missing Error Handling

- **Severity:** Recommendation
- **Issue:** Some function calls do not handle failures.
- **Code:**

```
solidity
CopyEdit
```

```
(bool sent, ) = feeCollector.call{value: msg.value}("");
```

- **Fix:** Use **require**.

```
solidity
CopyEdit
require(sent, "Transfer failed");
```

[N11] Business Logic Flaw

- **Severity:** Medium
- **Issue:** `approveEnsName()` does not check for duplicates.
- **Code:**

```
solidity
CopyEdit
function approveEnsName(string memory ensName) external onlyOwner {
    approvedEnsNames[ensName] = true;
}
```

- **Fix:** Add a **duplicate check**.

[N12] Redundant Code

- **Severity:** Low
- **Issue:** The `setAddr()` and `addr()` functions contain duplicated logic.
- **Fix:** Refactor code using **modifiers**.

4.2 Security Summary

- **High Risk:** 1 issue (N6)
- **Medium Risk:** 3 issues (N4, N8, N11)
- **Low Risk:** 4 issues (N1, N3, N5, N9)
- **Recommendations:** 5 issues (N2, N7, N10, N12, N13)

5. Detailed Vulnerability Report

[N6] Improper Access Control

- **Risk Level:** ⚠️ **High**
- **Issue:** The `approveKyc` function is only accessible by the contract owner, creating a **centralized risk**.
- **Recommendation:**
  - Implement **Role-Based Access Control (RBAC)**
  - Introduce **multi-signature approval**
  - Add **time delay mechanisms** for high-risk operations.

[N8] Reentrancy Vulnerability

- **Risk Level:** ⚠️ **Medium**
- **Issue:** The `withdrawFees()` function is vulnerable to **reentrancy attacks**.
- **Recommendation:**
  - Follow the **Checks-Effects-Interactions** pattern.
  - Use **Reentrancy Guard**.

[N11] Business Logic Flaw

- **Risk Level:** ⚠️ **Medium**
- **Issue:** The `approveEnsName()` function does not enforce **unique ENS name approvals**.
- **Recommendation:**
  - Add a **duplicate check** before assigning values.

6. Conclusion

6.1 Overall Security Score

Security Score: 7.2/10
Major deductions:

- **Access control flaws (-1.5)**
- **Error handling gaps (-0.8)**
- **Incomplete logging (-0.5)**

6.2 Recommended Fix Timeline

Priority	Action	Deadline
Immediate	Fix High-Risk Issues (N6)	ASAP
Short-Term	Fix Medium-Risk Issues (N4, N8, N11)	3 Days
Long-Term	Fix Low-Risk & Code Quality Issues	Next Release

6.3 Disclaimer

This report only applies to the audited version. It does not cover:

- Future modifications.
- Third-party contract risks.
- Economic design flaws.