“1Club” Security Review

Case study by:

0xThirdEye

Dated:

January 4th, 2023

0xThirdEye.com

Contents

Introduction

Thank you for trusting 0xThirdEye with this security audit. Our executive report presents the findings of our systematic evaluation of the related smart contract source code for the 1Club, conducted in accordance with the scope defined.

The purpose of our security review was to identify any potential security issues in the smart contract implementation, identify semantic inconsistencies between the code and design document, and provide recommendations for improvement.

Our team are experts in identifying and evaluating vulnerabilities, we provide recommendations for addressing these issues to ensure the overall security and integrity of the systems we audit.

For more information about us visit: 0xThirdEye

Executive Summary

During the audit of 1Club, we focused on several key areas to ensure the security of the smart contracts. These included testing against various attack vectors, evaluating the codebase for compliance with best practices and standards, verifying that the contract logic meets the client's specifications and intentions, comparing the contract structure and implementation to those of industry leaders, and conducting a thorough manual review of the entire codebase.

Our security assessment identified a range of findings, from critical to informational, and we recommend taking action to address these findings in order to maintain a high level of security and adhere to industry practices.

Please note that security audits are limited by the timeframe in which they are conducted and may not be able to identify all vulnerabilities. It is essential to incorporate other security measures in addition to an audit to ensure the overall security of a project.

Furthermore, this security review is not a guarantee against a hack or any other security incident. It is based on a snapshot of the 1Club code at a specific commit and does not take into account any changes that may have been made after the audit was conducted. Any modifications to the code after this audit will require a new security review to properly assess their potential impact on the security of the system.

Risk Classification

Severity level	Impact: High	Impact: Medium	Impact: Low
Likelihood: High	Critical	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

Vulnerabilities range from issues related to gas optimization to critical vulnerabilities that could affect the operation or financial stability of a protocol.

We also include Informational findings & Gas Findings

Critical/High

These vulnerabilities have the potential to result in significant financial losses for the protocol and its users, as well as disrupt the intended operation of the protocol. These vulnerabilities pose a significant risk to the stability and integrity of the protocol.

- Missing validation for malicious ERC20/ERC721 Tokens
- Missing validation for transfer success
- Protocol susceptible to front-run attacks
- Lack of re-entrancy protection
- Missing Oracle price validation
- Protocol susceptible to Flashloan manipulation
- Protocol missing sufficient access control
- Protocol susceptible to malicious/naive DOS

Medium

These vulnerabilities may result in relatively minor financial losses for the protocol and may only affect certain subsets of users. These vulnerabilities may be more difficult to exploit and may not offer a sufficient incentive for an attacker to attempt to exploit them. However, it is important to address these vulnerabilities to ensure the overall security and stability of the protocol.

- Incorrect implementation of Check-Effects-Interaction pattern.
- Missing validation for sender/recipient address
- Overflow/Underflow
- Insufficient error handling
- Signature replay attack
- Incorrect mathematical calculations

Low

These vulnerabilities may not provide a strong incentive for a malicious actor to exploit them and are unlikely to have a significant impact on the protocol. However, addressing these vulnerabilities can prevent minor complications from occurring.

Informational / Gas Optimisations

These vulnerabilities may improve the cost-efficiency of running the protocol for both the protocol operators and users. Gas optimization & informational measures are generally recommended as a way to enhance the user experience and are considered a "quality of life" improvement.

Protocol Overview

1Club is the world's first invite only digital members club. The 1Club ecosystem includes:

- $1CLB NFT contract with supply of 11,111
- $KEY - ERC-20 contract with 1,111,111 supply

Initial mint price of the $1CLB token will be at 0.25 ETH, after which the price increases at a linear rate of 0.002 ETH per member that joins.

Royalty fees are paid out via the smart-contract and sent to a charity contract and a multi-signatory wallet automatically.

The protocol has the following types of privileged actors:

- ADMIN_ROLE
- PAUSER_ROLE
- MANAGER_ROLE

A typical usage flow would be the following:

- Whitelisted user mints a $1CLB NFT by providing a signed 'Voucher'
- User holding sufficient $KEY ERC20s can invite other users giving them access to mint
- User can create a payment plan to mint $1CLB over a given period of time

Findings Summary

The following smart contracts were in scope of the audit:

- OneClub.sol
- The1ClubPaymentPlans.sol

ID	Title
[H-01]	Voucher NFT ID's can be stolen when minting may cause DOS
[H-02]	External mint functions missing success check
[M-01]	Missing events & timelocks for critical changes by admin
[M-02]	Nested loop may cause DOS when retrieving Stake Pool Balances
[L-01]	Unused receive() will lock ether in contract
[G-01]	State variables can be packed more efficiently
[G-02]	Use custom errors rather than require()/revert() strings
[G-03]	Don't initialise variables with their default value
[G-04]	Use bytes32 rather than strings where possible

[G-05]	I++ should be ++I
[G-06]	I++ can be unchecked{i++} when used in loops
[I-01]	Missing indexed fields
[I-02]	Check-Effect-Interaction pattern not followed
[I-03]	Open TODO/Comments should be resolved before deployment
[I-04]	Unnecessary Claim Function

Proof Of Concepts & Mitigations

ID - [H-01]
Title - Voucher NFT ID's can be stolen

Likelihood - Medium
Impact - High
Context - #L261-L294, #L185-L191

Description - A user who has completed their payment plan will pass the error checks here. The user can submit a malicious `Voucher` containing a `tokenId` owned by another user. The function `mintOneClubNFT` is missing checks to ensure the validity of the `voucher` parameter. The function is also missing checks to ensure `_client` == `msg.sender`. This function will then call the NFT contract and mint the NFT to the provided address `_client`. The user can then call claimNFT(), and receive the stolen NFT. This will cause DOS to the owner of the stolen NFT as they will not be able to mint with the NFT ID assigned to their Voucher.

Recommendation - Add a check to ensure `_client == msg.sender`

ID - [H-02]
Title - External mint functions missing success check

Likelihood - Medium
Impact - High
Context - #L277

Description - If the function `oneClub.safeMintToPaymentPlans()` reverts for any reason this will not be accounted for in the function `mintOneClubNFT()`. This will prevent the user from being able to access their NFT permanently as `paymentDetails[_client].nftMinted = true;`

Recommendation - Add a check to ensure the success of `oneClub.safeMintToPaymentPlans()`

ID - [M-01]
Title - Missing events & timelocks for critical changes by admin

Likelihood - Medium
Impact - Medium
Context - #L141-L146, #L150-L152, #L156-L158, #L160-L162, #L164-L169, #L210-L212, #L217-L219, #L224-L226, #L231-L233, #L238-L240, #L246-L255,#L260-L262, #L286-L288

Description: Owner/admin only functions that change critical parameters should emit events and have timelocks. Events allow capturing the changed parameters so that off-chain tools/interfaces can register such changes with timelocks that allow users to evaluate them and consider if they would like to engage/exit based on how they perceive the changes as affecting the trustworthiness of the protocol or profitability of the implemented financial services. The alternative of directly querying on-chain contract state for such changes is not considered practical for most users/usages.

Recommendation - Add events to all owner/admin functions that change critical parameters. Add timelocks to introduce time delays for critical parameter changes that significantly impact market/user incentives/security.

ID - [M-02]
Title - Nested loop may cause DOS when retrieving Stake Pool Balances

Likelihood - Low
Impact - High
Context - #L306-L326

Description - Function `getStakePoolBalances` specifies `uint256 upto` as a parameter however if this value becomes sufficiently large the for loop which is nested may be unable to complete due to it exceeding the gas limit. This will prevent the `onlyHolder` modifier from working as intended, preventing holders from calling `invite`.

Recommendation - Specify a limit for _stakingPoolCount

ID - [L-01]
Title - Unused receive() will lock ether in contract

Likelihood - Low
Impact - Low
Context - #L343

Description - If the intention is for the Ether to be used, the function should call another function, otherwise it should revert

Recommendation - The function should call another function, otherwise it should revert

ID - [G-01]
Title - State variables can be packed more efficiently, decreasing gas cost at deployment
Context - `OneClub.sol` `The1ClubPaymentPlans.sol`

ID - [G-02]
Title - Use custom errors rather than require()/revert() strings
Context - Throughout Codebase

ID - [G-03]
Title - Don't initialise variables with their default value
Context - #L91, #L94, #L106-L107, #L315, #L319, #L389, #L450, #L460, #L521, #L385, #L352

ID - [G-04]
Title - Use bytes32 rather than strings where possible
Context - #L80-L81, #L115

ID - [G-05]
Title - I++ cost more than ++I
Context - #L315, #L319, #L389, #L450, #L460, #L521

ID - [G-06]
Title - I++ can be unchecked{i++} when used in loops
Context - #L315, #L319, #L389, #L450, #L460, #L521

ID - [I-01]
Title - Missing indexed fields in events
Context - #L103-L107, #L109-L113, #L115, #L117, #L63-L67

Index event fields make the field more quickly accessible to off-chain tools that parse events. However, note that each index field costs extra gas during emission, so it's not necessarily best to index the maximum allowed per event (three fields).

Each event should use three indexed fields if there are three or more fields, and gas usage is not particularly of concern for the events in question. If there are fewer than three fields, all of the fields should be indexed.

ID - [I-02]
Title - Check-Effect-Interaction pattern not followed
Context - #L472-L575

ID - [I-03]
Title - Open TODO/Comments should be resolved before deployment
Context - #L71, #L81, #L90

ID - [I-04]
Title - Unnecessary Claim Function
Context - #L297-L299