SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

Base Club

AUDIT

SECURITY ASSESSMENT

23. August, 2023

FOR

BASECLUB

Introduction

SolidProof.io is a brand of the officially registered company MAKE Network GmbH, based in Germany. We're mainly focused on Blockchain Security such as Smart Contract Audits and KYC verification for project teams.
Solidproof.io assess potential security issues in the smart contracts implementations, review for potential inconsistencies between the code base and the whitepaper/documentation, and provide suggestions for improvement.

Disclaimer

SolidProof.io reports are not, nor should be considered, an "endorsement"or "disapproval" of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc'...)

SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

SolidProof.io Reports represent an extensive auditing 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. SolidProof's position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of the security or functionality of the technology we agree to analyze.

Project Overview

Summary

Project Name	Base Club
Website	https://baseclub.games/
About the project	BaseClub is a fully decentralized and autonomous liquidity on-chain casino on Base, a secure, low-cost, developer-friendly Ethereum L2 incubated by Coinbase.
Chain	Base Scan
Language	Solidity
Codebase	BaseClub: https://basescan.org/address/0xbf96f480bcceead4c86523a3c99df54d2c8c8222#code IDO: https://basescan.org/address/0x0dbadf73d11c141a1af22ae5cb911ad27d972d4e#code
Commit	N/A
Unit Tests	Not Provided

Social Medias

Telegram	https://t.me/BaseClubGames
Twitter	https://twitter.com/BaseClubGames
Facebook	N/A
Instagram	N/A
GitHub	N/A
Reddit	N/A
Medium	N/A
Discord	https://discord.com/invite/sWkeDnmcDH
YouTube	https://www.youtube.com/@BaseClubGames
TikTok	N/A
LinkedIn	N/A

Audit Summary

Version	Delivery Date	Change Log
v1.0	23. August 2023	· Layout Project · Automated/ Manual-Security Testing · Summary

Note - The following audit report presents a comprehensive security analysis of the smart contract utilized in the project. This analysis did not include functional testing (or unit testing) of the contract's logic. We cannot guarantee 100% logical correctness of the contract as it was not functionally tested by us.

File Overview

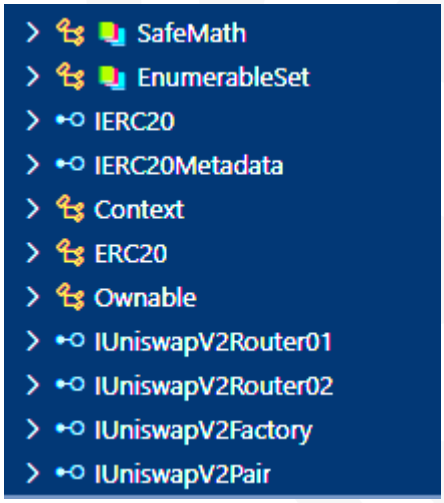
The Team provided us with the files that should be tested in the security assessment. This audit covered the following files listed below with an SHA-1 Hash.

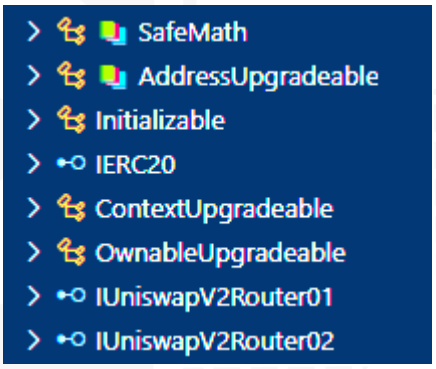
File Name	SHA-1 Hash
contracts/BaseClub.sol	24a276e9b48bf245ad01588e879affb17e80f84d
contracts/Ido.sol	55bea6a632ec4aef1b5dde385b7d33a1722319c5

Please note: Files with a different hash value than in this table have been modified after the security check, either intentionally or unintentionally. A different hash value may (but need not) be an indication of a changed state or potential vulnerability that was not the subject of this scan.

Imported packages

Used code from other Frameworks/Smart Contracts.





Note for Investors: We only audited contracts mentioned in the scope above. All contracts related to the project apart from that are not a part of the audit, and we cannot comment on its security and are not responsible for it in any way.

External/Public functions

External/public functions are functions that can be called from outside of a contract, i.e., they can be accessed by other contracts or external accounts on the blockchain. These functions are specified using the function declaration's external or public visibility modifier.

State variables

State variables are variables that are stored on the blockchain as part of the contract'sstate. They are declared at the contract level and can be accessed and modified by any function within the contract. State variables can be needed within visibility modifier, such as public, private or internal, which determines the access level of the variable.

Components

📝 Contracts	📚 Libraries	🔍 Interfaces	🎨 Abstract
3	4	9	5

Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.

🌐 Public	💰 Payable
143	13

External	Internal	Private	Pure	View
113	186	8	43	72

StateVariables

Total	🌐 Public
43	29

Capabilities

Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts

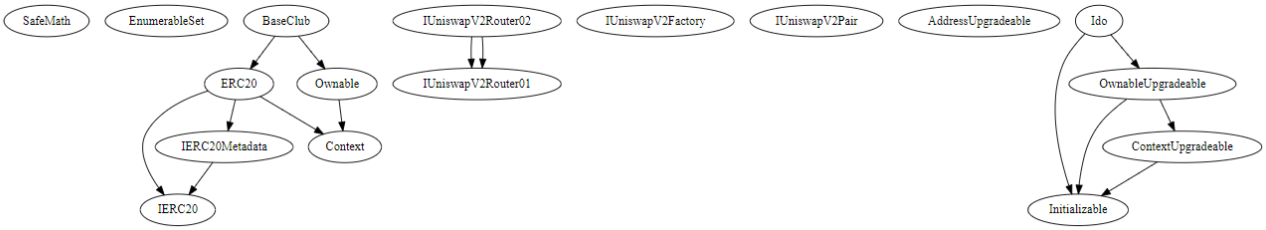
`^0.8.0`
`>=0.6.2`
`>=0.5.0`
`^0.8.1`
`^0.8.2`
	`----------`	`Yes`	`Yes`
			(4 asm blocks)
				`----------`

Transfers ETH	Low-Level Calls	Delegate Call	Uses Hash Functions	ECRecover	New/Create/Create2
`yes`					

TryCatch	Σ Unchecked
	`yes`

Inheritance Graph

An inheritance graph is a graphical representation of the inheritance hierarchy among contracts. In object-oriented programming, inheritance is a mechanism that allows one class (or contract, in the case of Solidity) to inherit properties and methodsfrom another class. It shows the relationships between different contracts and how they are related to each other through inheritance.



Audit Information

Vulnerability & Risk Level

Risk represents the probability that a certain source threat will exploit the vulnerability and the impact of that event on the organization or system.The risk level is computed based on CVSS version 3.0.

Level	Value	Vulnerability	Risk (Required Action)
Critical	9 - 10	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	7 – 8.9	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon aspossible.
Medium	4 – 6.9	A vulnerability that could affect the desired outcome of executingthe contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	2 – 3.9	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	0 – 1.9	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to check the repository for security-related issues, code quality, and compliance with specifications and best practices. To this end, our team of experienced pen-testers andsmart contract developers reviewed the code line by line and documented any issues discovered.

We check every file manually. We use automated tools only so that they help us achieve faster and better results.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
   a. Reviewing the specifications, sources, and instructions provided to SolidProof to ensure we understand the size, scope, and functionality of the smart contract.
   b. Manual review of the code, i.e., reading the source code line by line to identify potential vulnerabilities.
   c. Comparison to the specification, i.e., verifying that the code does what is described in the specifications, sources, and instructions provided to SolidProof.

2. Testing and automated analysis that includes the following:
   a. Test coverage analysis determines whether test cases cover code and how much code is executed when those test cases are executed.
   b. Symbolic execution, which is analysing a program to determine what inputs cause each part of a program to execute.

3. Review best practices, i.e., review smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on best practices, recommendations, and research from industry and academia.

4. Concrete, itemized and actionable recommendations to help you secure your smart contracts.

Overall Security

Upgradeability

Contract is an upgradable	✗ Deployer can update the contract with new functionalities.
Description	The deployer can replace the old contract with a new one with new features. Be aware of this, because the owner can add new features that may have a negative impact on your investments.
Example	We assume that you have funds in the contract, and it has been audited by any security audit firm. Now the audit has passed. After that, the deployer can upgrade the contract to allow him to transfer the funds you purchased without any approval from you. This has the consequence that your funds can be taken by the creator.
Comment	The Ido token can upgrade the ownable functionality of the contract after deployment. Also, the address library that is used in the contract is also upgradable.

File/Line(s): L982
Codebase: Ido.sol

```
pragma solidity ^0.8.0;
UnitTest stub | dependencies | uml | draw.io
contract Ido is Initializable, OwnableUpgradeable {

    using SafeMath for uint256;
```

Ownership

The ownership is not renounced	✗ The ownership is not renounced
Description	The owner has not renounced the ownership that means that the owner retains control over the contract's operations, including the ability to execute functions that may impact the contract's users or stakeholders. This can lead to several potential issues, including: • Centralizations • The owner has significant control over contract's operations.
Example	N/A
Comment	N/A

Note – *The contract cannot be considered as renounced till it is not deployed or having some functionality that can change the state of the contract.*

Ownership Privileges

These functions can be dangerous. Please note that abuse can lead to financial loss. We have a guide where you can learn more about these Functions.

Minting tokens

Minting tokens refer to the process of creating new tokens in a cryptocurrency or blockchain network. This process is typically performed by the project's owner or designated authority, who has the ability to add new tokens to the network's total supply.

Contract owner cannot mint new tokens	✅ **The owner cannot mint new tokens**
Description	The owner is not able to mint new tokens once the contract is deployed.
Comment	N/A

Burning tokens

Burning tokens is the process of permanently destroying a certain number of tokens, reducing the total supply of a cryptocurrency or token. This is usually done to increase the value of the remaining tokens, as the reduced supply can create scarcity and potentially drive up demand.

Contract owner cannot burn tokens	✅ **The owner cannot burn tokens**
Description	The owner is not able burn tokens without any allowances.
Comment	N/A

Blacklist addresses

Blacklisting addresses in smart contracts is the process of adding a certain address to a blacklist, effectively preventing them from accessing or participating in certain functionalities or transactions within the contract. This can be useful in preventing fraudulent or malicious activities, such as hacking attempts or money laundering.

Contract owner cannot blacklist addresses	✅ **The owner cannot blacklist addresses**
Description	The owner is not able blacklist addresses to lock funds.
Comment	N/A

Fees and Tax

In some smart contracts, the owner or creator of the contract can set fees for certain actions or operations within the contract. These fees can be used to cover the cost of running the contract, such as paying for gas fees or compensating the contract's owner for their time and effort indeveloping and maintaining the contract.

Contract owner cannot set fees more than 25%.	✅ **The owner cannot set fees more than 25%**

Description	The owner is not able to set the fees above 25%.
Comment	N/A

Lock User Funds

In a smart contract, locking refers to the process of restricting access to certain tokens or assets for a specified period of time. When token or assets are locked in a smart contract, they cannot be transferred or used until the lock-up period has expired or certain conditions have been met.

Contract owner cannot lock user funds.	✅ **The owner cannot lock user funds.**
Description	The owner is not able to lock the contract by any functions or updating any variables.
Comment	N/A

Centralization Privileges

Centralization can arise when one or more parties have privileged access or control over the contract's functionality, data, or decision-making. This can occur, for example,if the contract is controlled by a single entity or if certain participants have special permissions or abilities that others do not.

In the project, there are authorities that have access to the following functions:

File	Privileges
BaseClub.sol	➢ The owner can enable trading only once. ➢ The owner can whitelist addresses from fees. ➢ The owner can set the minimum tokens required for swapping. ➢ The owner can add and remove the liquidity pair address. ➢ The owner can enable and disable swapping. ➢ The marketing wallet address can change the marketing wallet address.
Ido.sol	➢ The owner can whitelist any amount for multiple wallets at once. ➢ The owner can start the Ido only once. ➢ The owner can finalize the Ido manually. ➢ The owner can refund the number of tokens to his wallet if the liquidity fails.

Recommendations

To avoid potential hacking risks, it is advisable for the client to manage the private key of the privileged account with care. Additionally, we recommend enhancing the security practices of centralized privileges or roles in the protocol through a decentralized mechanism or smart-contract-based accounts, such as multi-signature wallets.

Here are some suggestions of what the client can do:

- Consider using multi-signature wallets: Multi-signature wallets require multiple parties to sign off on a transaction before it can be executed, providing an extra layer of security e.g. Gnosis Safe
- Use of a timelock at least with a latency of e.g. 48-72 hours for awareness of privileged operations

- Introduce a DAO/Governance/Voting module to increase transparency and user involvement
- Consider Renouncing the ownership so that the owner cannot modify any state variables of the contract anymore. Make sure to set up everything before renouncing.

Audit Result

#1 | Transfer of tokens without enabling trade.

File	Severity	Location	Status
BaseClub.sol	Medium	L1506	Open

Description – The trading needs to be enabled by the owner in order for regular users to transfer tokens. On the contrary, the owner can authorize addresses manually and those addresses will be able to trade tokens. This functionality can be exploited in the following way, for example, there is a presale and the wallets used for the presale can be authorized by the owner. All the tokens obtained can be consolidated into a final wallet address and facilitate trading and selling of the acquired tokens, the last wallet address can be authorized.

Remediation – It is recommended that trading should be enabled at the time of deployment for every user to avoid these circumstances.

#2 | Liquidity is added to the external owner account.

File	Severity	Location	Status
Ido.sol	Medium	L1102-1113	ACK

Description – The contract's liquidity is automatically added to the 'owner' address, which is not recommended because, in an extreme scenario, this can be used to drain liquidity from the contract.

Alleviation – The contract owner is planning to use 'baseSwap' which will help them to lock the liquidity in their platform as of now they haven't found anything that can help them to maintain the liquidity on a safe platform. Hence, this is the reason for marking the issue as acknowledged.

#3 | Missing events arithmetic.

File	Severity	Location	Status
BaseClub.sol	Low	L1567, 1573, 1582, 1587, 1600	Open
Ido.sol	Low	L1069-1072, 1088-1092	Open

Description – Emit all the critical parameter changes.

#4 | Floating pragma solidity version.

File	Severity	Location	Status
BaseClub.sol	Low	L1444	Open
Ido.sol	Low	L981	Open

Description – Adding the constant version of solidity is recommended, as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.

#5 | Remove Safemath library.

File	Severity	Location	Status
BaseClub.sol	Low	L21-232	Open
Ido.sol	Low	L21-232	Open

Description – compiler version above 0.8.0 has the ability to control arithmetic overflow/underflow, it is recommended to remove the unwanted code in order to avoid high gas fees.

#6 | Missing Visibility.

File	Severity	Location	Status
BaseClub.sol	Low	L1458	Open

Description – There is no visibility present for the state variable which is not recommended.

Remediation – Add 'public' or 'private' while initializing any variable in the contract.

#7 | Missing dead address check.

File	Severity	Location	Status
BaseClub.sol	Low	L1557-1561, L1563-1567, L1580-1583	Open

Description – It is recommended to check that the address cannot be set to a dead address.

#8 | Missing 'require' check.

File	Severity	Location	Status
Ido.sol	Low	L1028-1056	Open

Description – Add a 'require' check that the end-time should always be greater than the start-time otherwise the functionality of the token will fail.

#9 | Unnecessary code.

File	Severity	Location	Status
Ido.sol	Low	L1241	Open

Description – This contract already contains the receive function which can handle all the ETH transactions in this contract.

Remediation – Remove the irrelevant 'fallback' function from the contract to avoid high gas fees and unnecessary code from the contract.

#10 | NatSpec Documentation missing.

File	Severity	Location	Status
IgniteToken.sol	Informational	--	Open

Description – If you started to comment on your code, also comment on all other functions, variables, etc.

Legend for the Issue Status

Attribute or Symbol	Meaning
Open	The issue is not fixed by the project team.
Fixed	The issue is fixed by the project team.
Acknowledged(ACK)	The issue has been acknowledged or declared as part of business logic.

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY