SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC
Development | Marketing

MADE IN GERMANY

Iniverse

Audit

Security Assessment
16. June, 2023

For

SolidProof_io

@solidproof_io

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.

Version	Date	Description
1.0	14. June 2023	• Layout project • Automated- /Manual-Security Testing • Summary
	16. June 2023	• Reaudit

Note - This Audit report comprises a security analysis of the **Iniverse** smart contracts. This analysis did not include functional testing (or unit testing) of the contract’s logic.

Network
Binance Smart Chain

Website
http://iniverse.xyz/

Telegram
https://t.me/iniverse_xyz_real

Twitter
https://twitter.com/iniverse_xyz

Discord
https://discord.gg/DAe95GTq

Medium
https://medium.com/@iniverse

Description

TBA

Project Engagement

During the Date of 14 June 2023, **Iniverse Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical and focused on identifying security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

Logo

Contract Link

v1.0

- Provided as Files

Note for Investors: We only Audited a simple token contract and a Presale contract for **Iniverse**. However, If the project has other contracts (for example, a Staking contract etc), and they were not provided to us in the audit scope, then we cannot comment on its security, and we are not responsible for it in any way.

Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system. 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 executing the 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 evaluate the repository for security-related issues, code quality, and adherence to specification and best practices. To do so, reviewed line-by-line by our team of expert pentesters and smart contract developers, documenting any issues as there were discovered.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
    - i) Review of the specifications, sources, and instructions provided to SolidProof to make sure we understand the size, scope, and functionality of the smart contract.
    - ii) Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
    - iii) Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to SolidProof describe.

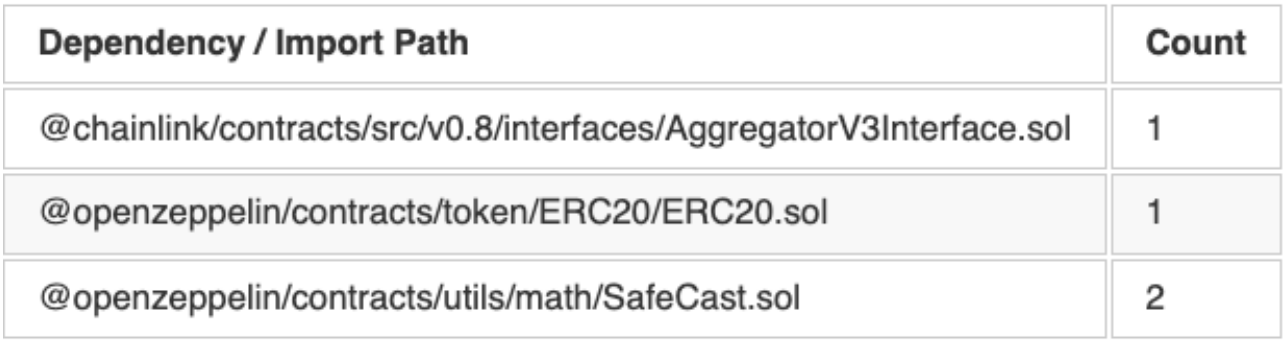
2. Testing and automated analysis that includes the following:
    - i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    - ii) Symbolic execution, which is analysing a program to determine what inputs causes each part of a program to execute.

3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

4. Specific, itemized, actionable recommendations to help you take steps to secure your smart contracts.

Used Code from other Frameworks/Smart Contracts (direct imports)

Imported packages:



Dependency / Import Path	Count
@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol	1
@openzeppelin/contracts/token/ERC20/ERC20.sol	1
@openzeppelin/contracts/utils/math/SafeCast.sol	2

Tested Contract Files

This audit covered the following files listed below with a SHA-1 Hash.

A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

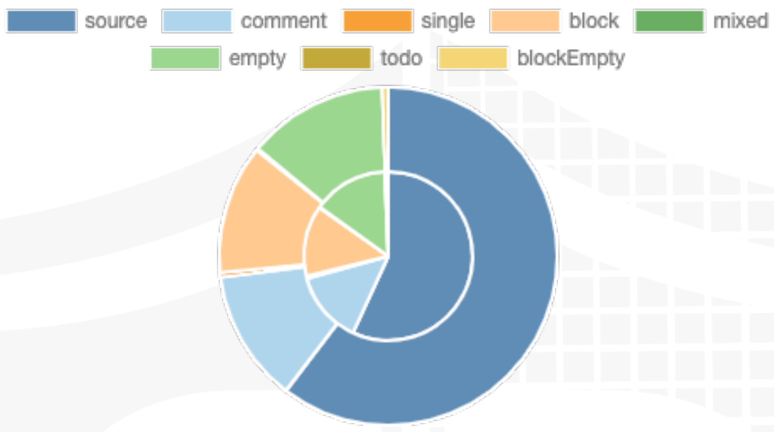
v1.0

File Name	SHA-1 Hash
contracts/ICOPrimary.sol	49680417573fcb5c6b2a7b3adc63181 17f21a987
contracts/ICOSecondary.sol	b33f61d1c5cfeb0165f7a1e99d77f5aad 8c4ebf8
contracts/Context.sol	2af917a1a1c7478a7643affb76178110 7e385688
contracts/ TokenOnPrimaryNetwork.sol	2bc2b75cc9b0540222a1c292ecb33da 1467a74cc
contracts/Ownable.sol	0967954b34fdbf61c8bb160af9787b81 922b915d
contracts/ ReentrancyGuard.sol	24337b2560f10ea31e87bb01335a713 4641f8cf6

Metrics

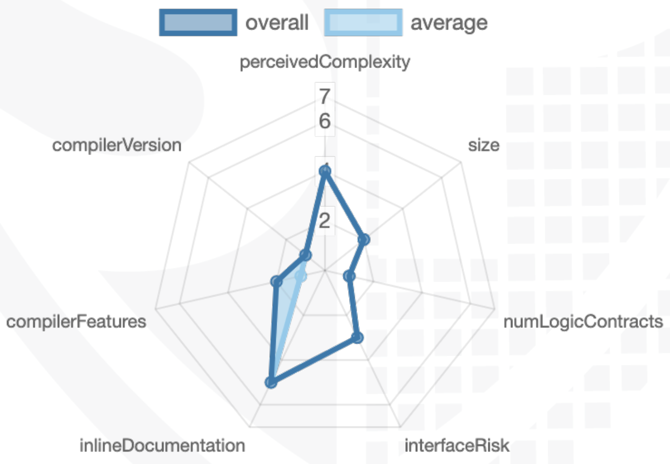
Source Lines

v1.0



Risk Level

v1.0



Capabilities

Components

📝Contracts	📚Libraries	🔍Interfaces	🎨Abstract
4	0	2	2

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
54	5

External	Internal	Private	Pure	View
36	42	0	0	28

StateVariables

Total	🌐Public
34	19

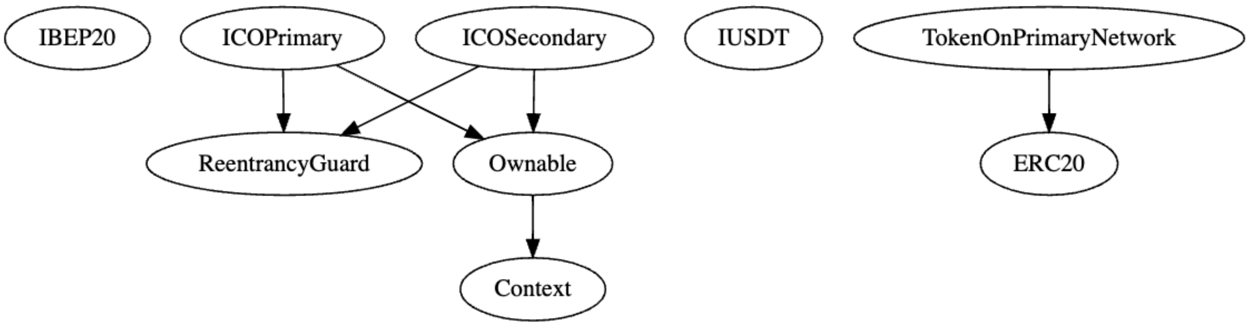
Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts
`^0.8.4` `^0.8.0`		`yes`		

📤 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧮 Uses Hash Functions	🔖 ECRecover	🌀 New/Create/Create2
`yes`					

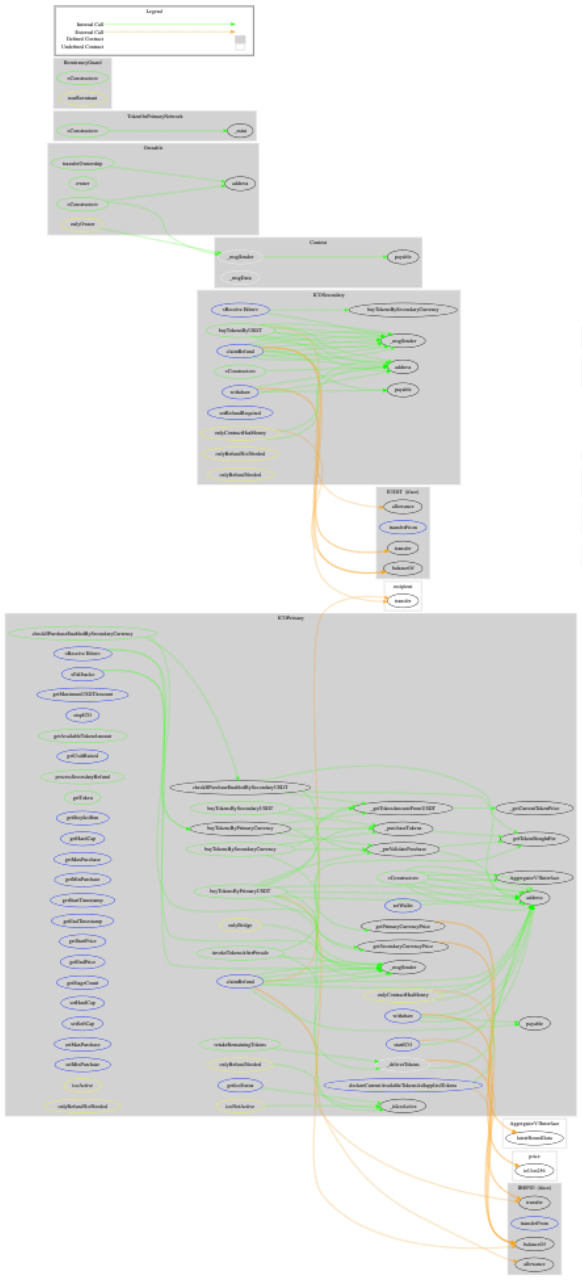
♻️ TryCatch	Σ Unchecked

Inheritance Graph
v1.0



CallGraph

v1.0



Scope of Work/Verify Claims

The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

We will verify the following claims:

1. Is contract an upgradeable
2. Correct implementation of Token standard
3. Deployer cannot mint any new tokens
4. Deployer cannot burn or lock user funds
5. Deployer cannot pause the contract
6. Deployer cannot set fees
7. Deployer cannot blacklist/antisnipe addresses
8. Overall checkup (Smart Contract Security)

Is contract an upgradeable

Name	
Is contract an upgradeable?	No

Correct implementation of Token standard

ERC20				
Function	**Description**	**Exist**	**Tested**	**Verified**
TotalSupply	Provides information about the total token supply	✓	✓	✓
BalanceOf	Provides account balance of the owner's account	✓	✓	✓
Transfer	Executes transfers of a specified number of tokens to a specified address	✓	✓	✓
TransferFrom	Executes transfers of a specified number of tokens from a specified address	✓	✓	✓
Approve	Allow a spender to withdraw a set number of tokens from a specified account	✓	✓	✓
Allowance	Returns a set number of tokens from a spender to the owner	✓	✓	✓

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✓
Max / Total Supply	N/A		

Comments:

v1.0

- The total supply will be decided at the time of deployment

Deployer cannot burn or lock user funds

Name	Exist	Tested	Status
Deployer cannot lock	–	–	–
Deployer cannot burn	–	–	–

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer can pause	✓	✓	✗

Comments:

v1.0

- Owner can stop the ICO at anytime

Deployer cannot set fees

Name	Exist	Tested	Status
Deployer cannot set fees over 25%	–	–	–
Deployer cannot set fees to nearly 100% or to 100%	–	–	–

Deployer can blacklist/antisnipe addresses

Name	Exist	Tested	Status
Deployer cannot blacklist/antisnipe addresses	–	–	–

Overall checkup (Smart Contract Security)

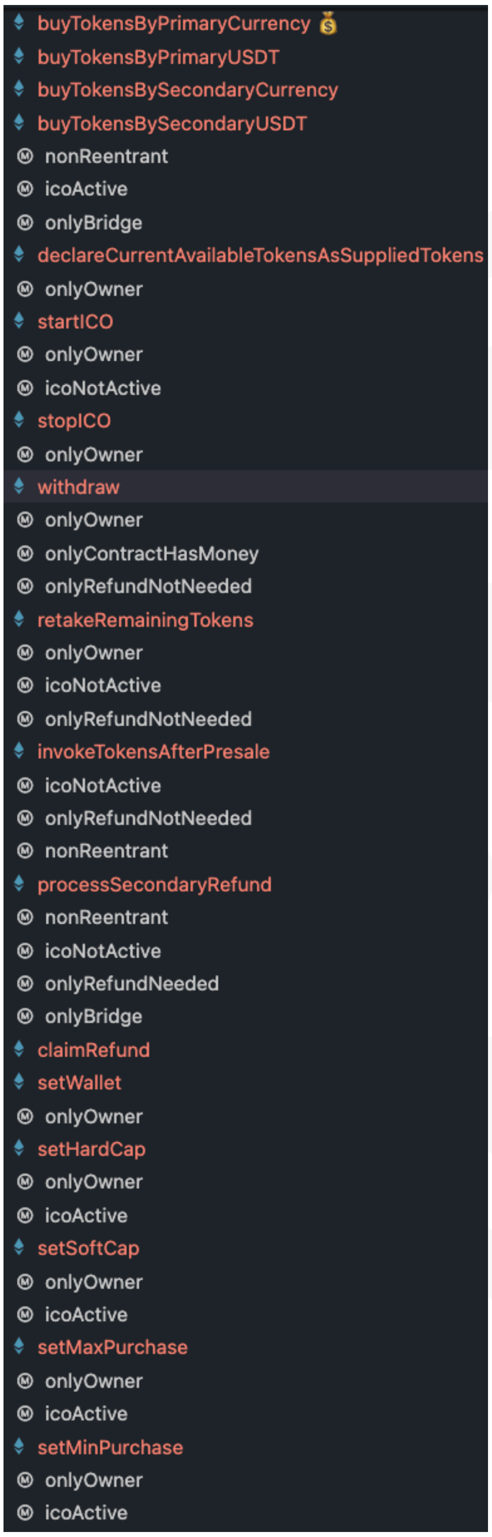
Tested	Verified
✓	✓

Legend

Attribute	Symbol
Verified / Checked	✓
Partly Verified	⚑
Unverified / Not checked	✗
Not available	–

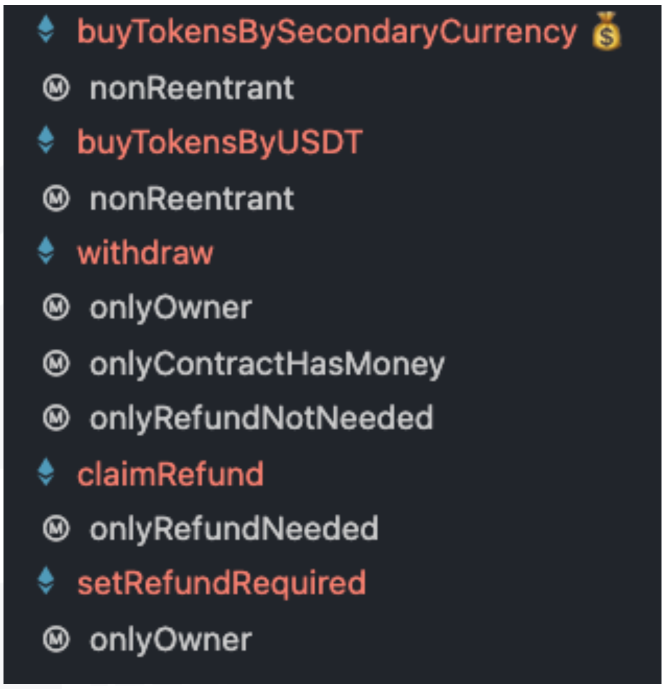
Modifiers and public functions
v1.0

ICOPrimary



```
buyTokensByPrimaryCurrency 💰
buyTokensByPrimaryUSDT
buyTokensBySecondaryCurrency
buyTokensBySecondaryUSDT
  nonReentrant
  icoActive
  onlyBridge
declareCurrentAvailableTokensAsSuppliedTokens
  onlyOwner
startICO
  onlyOwner
  icoNotActive
stopICO
  onlyOwner
withdraw
  onlyOwner
  onlyContractHasMoney
  onlyRefundNotNeeded
retakeRemainingTokens
  onlyOwner
  icoNotActive
  onlyRefundNotNeeded
invokeTokensAfterPresale
  icoNotActive
  onlyRefundNotNeeded
  nonReentrant
processSecondaryRefund
  nonReentrant
  icoNotActive
  onlyRefundNeeded
  onlyBridge
claimRefund
setWallet
  onlyOwner
setHardCap
  onlyOwner
  icoActive
setSoftCap
  onlyOwner
  icoActive
setMaxPurchase
  onlyOwner
  icoActive
setMinPurchase
  onlyOwner
  icoActive
```

ICOSecondary



```
buyTokensBySecondaryCurrency 💰
  nonReentrant
buyTokensByUSDT
  nonReentrant
withdraw
  onlyOwner
  onlyContractHasMoney
  onlyRefundNotNeeded
claimRefund
  onlyRefundNeeded
setRefundRequired
  onlyOwner
```

Ownership Privileges

❖ *ICOPrimary.sol* -

The owner can only set these parameters when the ICO is active:

- Set hard cap and soft cap
- Set maximum and minimum purchase value to any arbitrary number

- The owner can withdraw the contract balance only when the soft cap is reached, but it is not strict because the owner also controls the soft cap
- The owner can retake the remaining tokens from the contract when the ico is not active
- The owner can start and stop the ICO at any time but once stopped, a new ICO needs to be created which means the same ICO cannot be resumed once it is stopped.
- Set the withdrawal wallet

❖ *ICOSecondary.sol* -

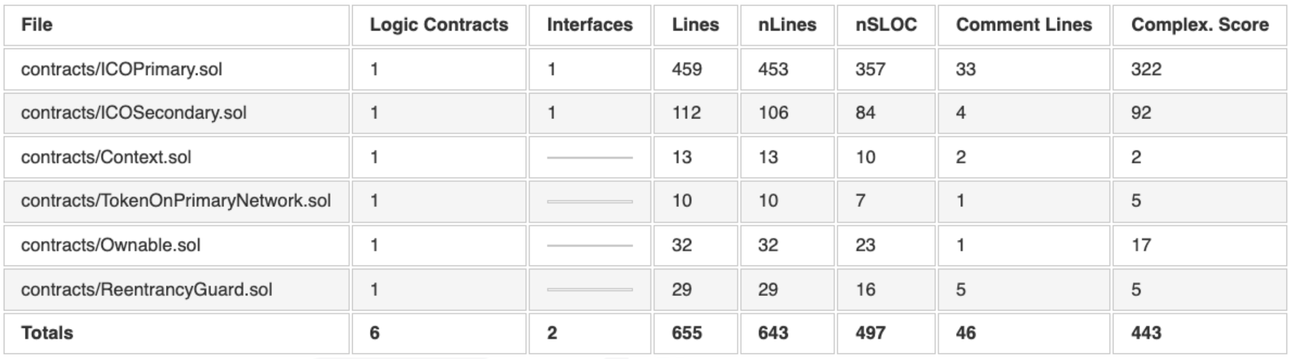
- The owner can withdraw tokens from the contract's balance at any time because the owner also controls the refundable status.

- The ownership of the contracts can never be renounced because of the custom-made ownable contract.
- The bridge address in the ICOPrimary contract must not be set as an EOA controlled by the deployer or owner because if done so then the owner can trigger secondary refunds to any arbitrary accounts manually.
    - Be aware of this

Please check if an OnlyOwner or similar restrictive modifier has been forgotten.

Source Units in Scope

v1.0



File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score
contracts/ICOPrimary.sol	1	1	459	453	357	33	322
contracts/ICOSecondary.sol	1	1	112	106	84	4	92
contracts/Context.sol	1	—	13	13	10	2	2
contracts/TokenOnPrimaryNetwork.sol	1	—	10	10	7	1	5
contracts/Ownable.sol	1	—	32	32	23	1	17
contracts/ReentrancyGuard.sol	1	—	29	29	16	5	5
Totals	**6**	**2**	**655**	**643**	**497**	**46**	**443**

Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalised lines of the source unit (e.g. normalises functions spanning multiple lines)
nSLOC	normalised source lines of code (only source-code lines; no comments, no blank lines)
Comment Lines	lines containing single or block comments
Complexity Score	a custom complexity score derived from code statements that are known to introduce code complexity (branches, loops, calls, external interfaces, ...)

Audit Results

Critical issues

No critical issues

High issues

No high issues

Medium issues

Medium Issues Fixed

Issue	File	Type	Line	Description	Status
#1	ICOSecondary.sol	Centralized Refund Function	74	The owner has complete control over the claimRefund function which means that the owner is able to stop that function and then no user will be able to claim the refund and their tokens will be locked. Moreover, because the owner is able to withdraw contract balance at any point, the refund amoutn won't be claimable.	Fixed

Low issues

Issue	File	Type	Line	Description	Status
#1	ICOPrimary and ICOSecondary	Contract doesn't import npm packages from source (like OpenZeppelin etc.)	—	We recommend to import all packages from npm directly without flatten the contract. Functions could be modified or can be susceptible to vulnerabilities	Fixed
#2	All	A floating pragma is set	—	The current pragma Solidity directive is „"^0.8.4", and "^0.8.0".	Fixed
#3	ICOSecondary.sol	State variable visibility is not set	27, 28	It is best practice to set the visibility of state variables explicitly. Moreover, due to this no one is able to see thier invested amount.	Fixed

#4	ICOPrimary.sol	State variable visibility is not set	21-24	It is best practice to set the visibility of state variables explicitly. Moreover, due to this no one is able to see thier invested amount.	Fixed
#5	ICOPrimary.sol	Missing Events Arithmetic	411-423	Emit an event for critical parameter changes	Fixed

Informational issues

Issue	File	Type	Line	Description	Status
#1	ICOSecondary.sol	State variables that could be declared constant (constable-states)	20	Add the `constant` attributes to state variables that never change	Fixed
#2	ICOSecondary.sol	Bad Modifier Usage	98	onlyContractHasMoney => Here of one of them is true, it will still call the other action which is not true. Instead, Seperate the functionality of withdraw to withrawCurrency and withdrawUsdt	Fixed
#3	ICOSecondary.sol	Wrong data type	94	Change the uint8 value to a bool to set the status directly instead of 1 for true and other numbers between the range of 0 - 2^8-1	Fixed
#4	ICOSecondary.sol	Incomplete Comment	67	Make sure to finish the comment to improve readability.	Fixed
#5	All	NatSpec documentation missing	—	If you started to comment your code, also comment all other functions, variables etc.	Acknowledged

Audit Comments

We recommend you use the particular form of comments (NatSpec Format, Follow the link for more information https://docs.soliditylang.org/en/latest/natspec-format.html) for your contracts to provide rich documentation for functions, return variables and more. This helps investors to make clear what that variable, functions etc., do.

14. June 2023:

- Unit tests with 95% code coverage were not provided to SolidProof so we cannot ensure complete functional correctness of the code's logic.
- We recommend **Iniverse** team conduct unit and fuzz tests thoroughly to rule out the possibility of unwanted logical and calculation errors.
- The refund can only be claimed by the users when the ICO is not active and raised USDT is less Than the soft cap.
- Make sure to set the withdraw wallet in the ICOSecondary contract carefully because if set to zero then it can never be changed.
- There is still an owner (The owner still has not renounced ownership)
- Read the whole report and modifiers section for more information

16. June 2023:

- ICOSecondary
    - Following functions
        - have been removed
            - processSecondaryRefund
            - claimRefund
        - Have been added
            - withdrawCurrency
    - Following modifiers have been removed
        - onlyRefundNotNeeded
    - Softcap has been removed from the contract
- During the audit, the website was not ready at this time.

SWC Attacks

ID	Title	Relationships	Status
SWC-136	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	**PASSED**
SWC-135	Code With No Effects	CWE-1164: Irrelevant Code	**PASSED**
SWC-134	Message call with hardcoded gas amount	CWE-655: Improper Initialization	**PASSED**
SWC-133	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	**PASSED**
SWC-132	Unexpected Ether balance	CWE-667: Improper Locking	**PASSED**
SWC-131	Presence of unused variables	CWE-1164: Irrelevant Code	**PASSED**
SWC-130	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	**PASSED**
SWC-129	Typographical Error	CWE-480: Use of Incorrect Operator	**PASSED**
SWC-128	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	**PASSED**

SWC-127	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
SWC-125	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SWC-124	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
SWC-123	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SWC-122	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SWC-121	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SWC-120	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	PASSED
SWC-119	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	PASSED
SWC-118	Incorrect Constructor Name	CWE-665: Improper Initialization	PASSED
SWC-117	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	PASSED

SWC-116	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-115	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SWC-114	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SWC-113	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SWC-112	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SWC-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SWC-110	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SWC-109	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SWC-108	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED
SWC-107	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SWC-106	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SWC-105	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	**PASSED**
SWC-104	Unchecked Call Return Value	CWE-252: Unchecked Return Value	**PASSED**
SWC-103	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	**PASSED**
SWC-102	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	**PASSED**
SWC-101	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	**PASSED**
SWC-100	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	**PASSED**

SolidProof_io

@solidproof_io

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC
Development | Marketing

MADE IN GERMANY