



SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

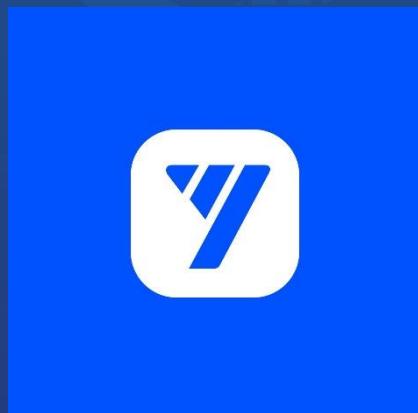
YFiONE

Audit

Security Assessment

01.March,2024

For



SolidProof.io



@solidproof_io

Disclaimer	3
Description	5
Project Engagement	5
Logo	5
Contract Link	5
Methodology	7
Used Code from other Frameworks/Smart Contracts (direct imports)	8
Tested Contract Files	9
Source Lines	10
Risk Level	10
Capabilities	11
Inheritance Graph	12
CallGraph	12
Scope of Work/Verify Claims	13
Modifiers and public functions	20
Source Units in Scope	21
Critical issues	22
High issues	22
Medium issues	22
Low issues	22
Informational issues	22
Audit Comments	23
SWC Attacks	23

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 security or functionality of the technology we agree to analyze.

Version	Date	Description
1.0	20. June,2023	<ul style="list-style-type: none">• Layout project• Automated- /Manual-Security Testing• Summary
1.1	01. March, 2024	<ul style="list-style-type: none">• Reaudit

Network
Ethereum Network(eth)

Website
<https://yfione.org/#/>

Twitter
<https://twitter.com/yfione>

Telegram
https://t.me/YFI_ONE

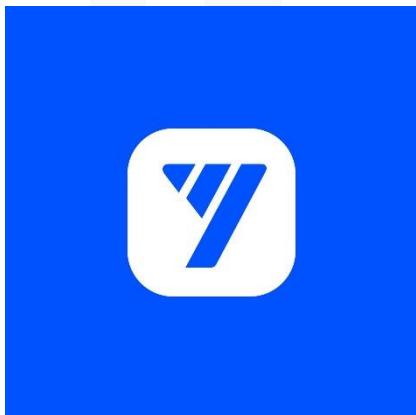
Description

YFiONE is committed to systematically producing new Defi products and will help develop, market and launch these Defi products to help develop its biosphere while providing ideal benefits to communities, projects and users. YFiONE can also be described as a decentralized venture capital (DVC), which empowers a decentralized interest-free financial ecosystem by building its community-invested Defi products, with the purpose of distributing revenue as a profit share (reward) To stakeholders.

Project Engagement

During the 20th of June 2023, **YFiONE** team engaged Solidproof.io to audit the smart contracts that they created. The engagement was technical in nature and focused on identifying the security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

Logo



Contract Links

v1.0

<https://etherscan.io/address/0xb518750b38b7b7795107b1912a5ef50db8566c76>

v1.1

<https://bscscan.com/token/0xB653e9Da791DD33e24CD687260C7C281928411Ba#code>

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 as possible.
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 they 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 analyzing 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
@openzeppelin/contracts/token/ERC20/ERC20.sol	1

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/yfo.sol	1fe19c17349f0f3a6f681fcc0ccb2760b0bca6c5

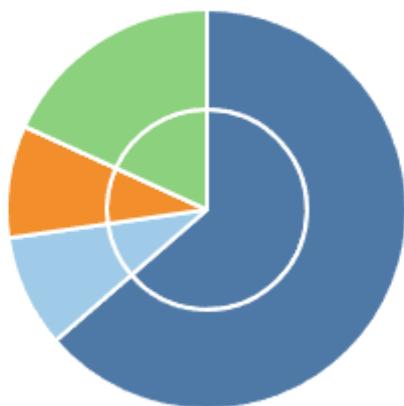
v1.1

File Name	SHA-1 Hash
Contracts/YFOToken.sol	c663440986773cad008bd03d20d082c3b42cc1a6

Metrics

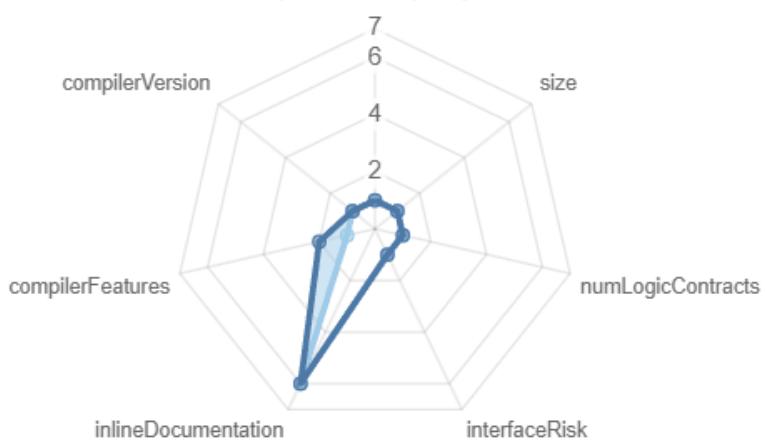
Source Lines v1.0

source comment single block mixed
empty todo blockEmpty



Risk Level v1.0

overall average
perceivedComplexity



Capabilities

Components

Contracts	Libraries	Interfaces	Abstract
1	0	0	0

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
0	0

External	Internal	Private	Pure	View
0	1	0	0	0

StateVariables

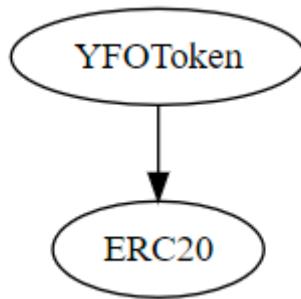
Total	Public
0	0

Capabilities

Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
^0.8.0				
Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover
TryCatch	Unchecked			

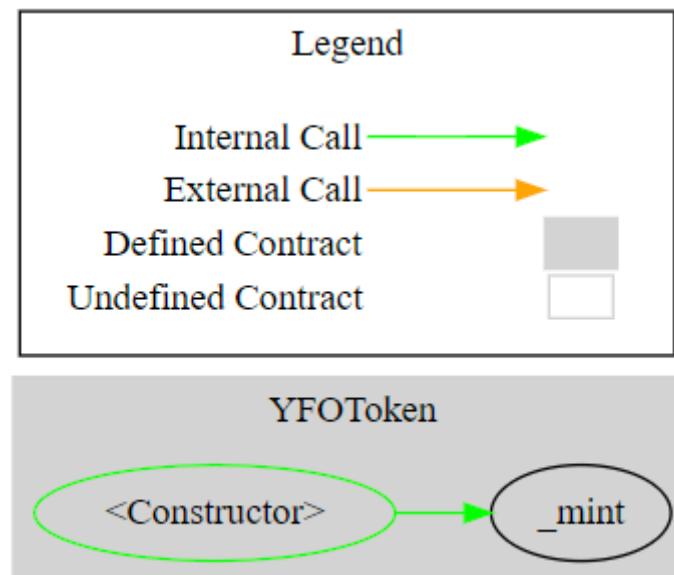
Inheritance Graph

v1.0



Call Graph

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. 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	Yes	Yes	Yes
balanceOf	Provides account balance of the owner's account	Yes	Yes	Yes
transfer	Executes transfers of a specified number of tokens to a specified address	Yes	Yes	Yes
transferFrom	Executes transfers of a specified number of tokens from a specified address	Yes	Yes	Yes
approve	Allow a spender to withdraw a set number of tokens from a specified account	Yes	Yes	Yes
allowance	Returns a set number of tokens from a spender to the owner	Yes	Yes	Yes

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	N/A	N/A	N/A
Max / Total Supply	50000000000000000000 0000		

Deployer cannot burn or lock user funds

Name	Exist	Tested	Status
Deployer cannot lock	N/A	N/A	N/A
Deployer cannot burn	N/A	N/A	N/A

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	N/A	N/A	N/A

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

N/A

Ownership Privileges:

- There are no ownership privileges in the contract.

Source Units in Scope

v1.0

Type	File	Logical Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complexity Score	Capabilities
	contracts\YFOToken.sol	1		10	10	7	1	5	
	Totals	1		10	10	7	1	5	

Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalized lines of the source unit (e.g. normalizes functions spanning multiple lines)
nSLOC	normalized 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

AUDIT PASSED

Critical issues

No critical issues

High issues

No high issues

Medium issues

No medium issues

Low issues

Issue	File	Type	Line	Description
#1	YFOToken.sol	Floating pragma solidity version	L2	It is recommended to add the constant version of solidity as this prevents the unintentional deployment of a contract with an outdated compiler that contains unresolved bugs.

Informational issues

Issue	File	Type	Line	Description
#1	YFOToken.sol	Natspec documentation missing	-	If you started to comment on your code, also comment on all other functions, variables, etc.

Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information <https://docs.soliditylang.org/en/v0.5.10/natspec-format.html>) for your contracts to provide rich documentation for functions, return variables and more. This helps investors to make clear what that variables, functions etc. do.

20. June, 2023:

- Read the report and modifier section to get more details.

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

<u>SWC-132</u>	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
<u>SWC-131</u>	Presence of unused variables	CWE-1164: Irrelevant Code	PASSED
<u>SWC-130</u>	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
<u>SWC-129</u>	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
<u>SWC-128</u>	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED
<u>SWC-127</u>	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
<u>SWC-125</u>	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
<u>SWC-124</u>	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED

<u>SWC -123</u>	Requirement Violation	<u>CWE-573: Improper Following of Specification by Caller</u>	PASSED
<u>SWC -122</u>	Lack of Proper Signature Verification	<u>CWE-345: Insufficient Verification of Data Authenticity</u>	PASSED
<u>SWC -121</u>	Missing Protection against Signature Replay Attacks	<u>CWE-347: Improper Verification of Cryptographic Signature</u>	PASSED
<u>SWC -120</u>	Weak Sources of Randomness from Chain Attributes	<u>CWE-330: Use of Insufficiently Random Values</u>	PASSED
<u>SWC -119</u>	Shadowing State Variables	<u>CWE-710: Improper Adherence to Coding Standards</u>	PASSED
<u>SWC -118</u>	Incorrect Constructor Name	<u>CWE-665: Improper Initialization</u>	PASSED
<u>SWC -117</u>	Signature Malleability	<u>CWE-347: Improper Verification of Cryptographic Signature</u>	PASSED
<u>SWC -116</u>	Timestamp	<u>CWE-829: Inclusion of Functionality from Untrusted Control Sphere</u>	PASSED

	Dependence		
<u>SWC -115</u>	Authorization through tx.origin	<u>CWE-477: Use of Obsolete Function</u>	PASSED
<u>SWC -114</u>	Transaction Order Dependence	<u>CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')</u>	PASSED
<u>SWC -113</u>	DoS with Failed Call	<u>CWE-703: Improper Check or Handling of Exceptional Conditions</u>	PASSED
<u>SWC -112</u>	Delegate call to Untrusted Callee	<u>CWE-829: Inclusion of Functionality from Untrusted Control Sphere</u>	PASSED
<u>SWC -111</u>	Use of Deprecated Solidity Functions	<u>CWE-477: Use of Obsolete Function</u>	PASSED
<u>SWC -110</u>	Assert Violation	<u>CWE-670: Always-Incorrect Control Flow Implementation</u>	PASSED
<u>SWC -109</u>	Uninitialized Storage Pointer	<u>CWE-824: Access of Uninitialized Pointer</u>	PASSED
<u>SWC -108</u>	State Variable	<u>CWE-710: Improper Adherence to Coding Standards</u>	PASSED

	Default Visibility		
<u>SWC -107</u>	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
<u>SWC -106</u>	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED
<u>SWC -105</u>	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
<u>SWC -104</u>	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
<u>SWC -103</u>	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	PASSED
<u>SWC -102</u>	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	Failed
<u>SWC -101</u>	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED

<u>SWC -100</u>	Function Default Visibility	<u>CWE-710: Improper Adherence to Coding Standards</u>	PASSED
-------------------------------------	-----------------------------------	--	---------------





[@SolidProof_io](https://SolidProof.io)



[@solidproof_io](https://t.me/Solidproof)

Solid
Proofed

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY