Audit

Security Assessment
29. November, 2021

For

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	29. November 2021	• Layout project • Automated- /Manual-Security Testing • Summary

Network
Binance Smart Chain (BEP20)

Website
https://www.wizardium.io/

Telegram
https://t.me/wizardium_nft

Twitter
https://twitter.com/wizardium_

Instagram
https://instagram.com/wizardium_nft

Discord
https://discord.com/invite/3QDpPhbA24

Reddit
https://www.reddit.com/r/wizardium_nft

Medium
https://wizardium.medium.com/

Description

Welcome to the crypto magic Metaverse full of mystical items and rare elements. Choose your Hero and have
a chance to dive into the world of magic and alchemy. Explore lands to find the elements and minerals, drop
them to the mighty cauldron and get the next Binance Smart Chain gems - Wizardium $WIZZY tokens, that can
get you insane ROI. Play to Earn, to make magic, to make money, to mine $WIZZY tokens, and to enjoy this
amazing crypto journey.

Project Engagement

During the 23rd of November 2021, **Wizardium Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical in nature 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
TBA

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

<u>Auditing Strategy and Techniques Applied</u>

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:
wizFarms

```
VRFRequestIDBase
LinkTokenInterface
VRFConsumerBase
WizzERC20
WizzERC721
📚 WizHeros
Ownable
📚 Nums
```

Wizz20

```
Context
WIZZ20
WIZZ20Metadata
```

WIZZ20Vesting

```
Context
Ownable
WIZZ20
```

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

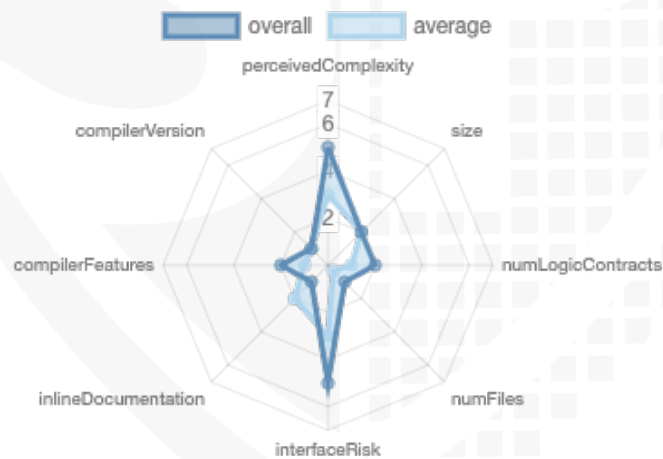| File Name	SHA-1 Hash
contracts/Wizz20Vesting.sol	9b0c697aafe9e9bbec250ee8b316b92ddae5075f
contracts/wizFarms.sol	78406443c33927f0442b19cd7f4fb9d9aaeb32f5
contracts/Wizz20.sol	7797600fe56b3052c807fa3ecf208fd4c216412a

Metrics

Source Lines
v1.0
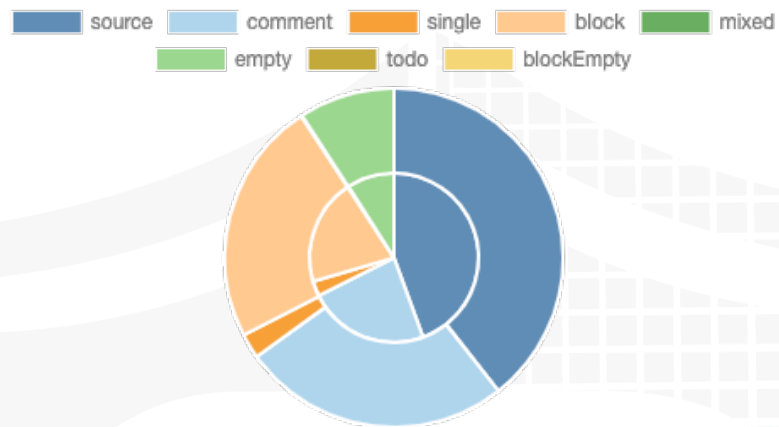
Risk Level
v1.0

Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	4	2	6	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.

Version	Public	Payable
1.0	82	6

Version	External	Internal	Private	Pure	View
1.0	28	101	0	10	41

State Variables

Version	Total	Public
1.0	33	8

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	>=0.7.0 <0.9.0		yes	**** (0 asm blocks)	

| Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/ Create/ Create 2
1.0	yes			yes		

Scope of Work

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. Correct implementation of Token standard
2. Deployer cannot mint any new tokens
3. Deployer cannot burn or lock user funds
4. Deployer cannot pause the contract
5. Overall checkup (Smart Contract Security)

Inheritance Graph
v1.0
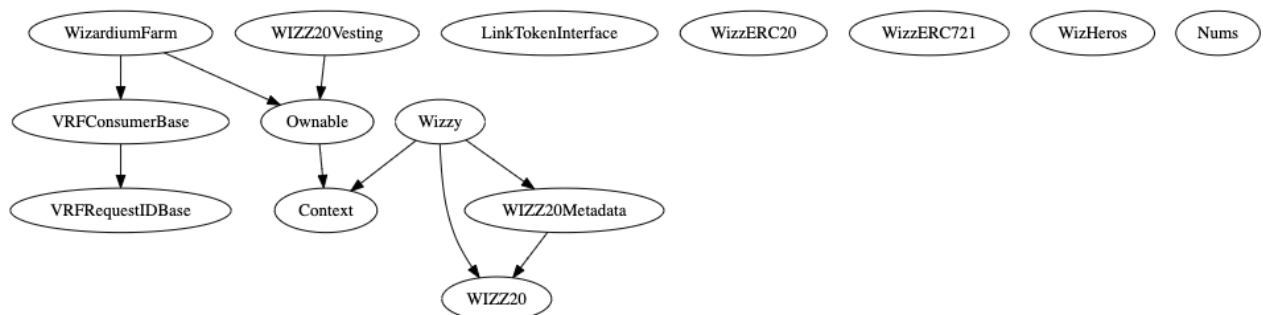
Verify Claims
Correct implementation of Token standard

Tested	Verified
✓	✓

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	✓	✓	✓

Write functions of contract

WizardiumFarm

- buyPotion
- changePotion...
- checkHeroSen...
- editFarm
- editPotions
- getRandomNu...
- initFarms
- initPotions
- processHero
- rawFulfillRand...
- retreive4rmW...
- renounceOwn...
- send2Worlds
- transferOwner...
- withdraw
- withdrawLink
- withdrawWiz

Wizzy

- approve
- burn
- decreaseAllow...
- increaseAllow...
- transfer
- transferFrom

Wizz20Vesting

- addTeamVesti...
- addVestor
- getReturnFro...
- renounceOwn...
- transferOwner...
- vestSome
- withdraw
- withdrawWizzy

15

Deployer cannot mint any new tokens

File	Name	Exist	Tested	Verified
WizardiumFarm	cannot mint	–	–	–
Wizzy	cannot mint	✓	✓	✓
Wizz20Vesting	cannot mint	–	–	–

Max / Total Supply: can be set while deploying

Deployer cannot burn or lock user funds

	Name	Exist	Tested	Verified
WizardiumFarm	cannot lock	✓	✓	✓
	cannot burn	–	–	–
Wizzy	cannot lock	✓	✓	✓
	cannot burn	✓	✓	✗
Wizz20Vesting	cannot lock	✓	✓	✓
	cannot burn	–	–	–

Deployer cannot pause the contract

File	Name	Exist	Tested	Verified
WizardiumFarm	cannot pause	✓	✓	✗
Wizzy	cannot pause	–	–	–
Wizz20Vesting	cannot pause	–	–	–

Comments:
v1.0
- OnlyOwner can pause farms

Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

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

Modifiers

WizFarms

onlyOwner

> changePotionProps

> initPotions
> editPotions
> initFarms
> editFarm

> withdrawWiz 💰
> withdrawLink 💰
> withdraw 💰

onlyVRFCoordinator
- rawFulfillRandomness
-

Wizz20Vesting

> addVestor

> addTeamVesting
> withdraw
> withdrawWizzy

CallGraph

Source Units in Scope
v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝🔍🎨	contracts/Wizz20Vesting.sol	3	1	222	219	152	36	135	💰🔄
📝📊🔍🎨	contracts/wizFarms.sol	6	3	592	509	342	146	297	💰🔄🏬☀️
📝🔍🎨	contracts/Wizz20.sol	2	2	502	406	165	274	109	☀️
📝📊🔍🎨	**Totals**	**11**	**6**	**1316**	**1134**	**659**	**456**	**541**	💰🔄🏬☀️

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

<div style="background-color:#7FEB5C; text-align:center">

AUDIT PASSED

</div>

Critical issues
- no critical issues found -

High issues
- no high issues found -

Medium issues
- no medium issues found -

Low issues

Issue	File	Type	Line	Description
#1	WizFarms	Contract doesn't import npm packages from source (like OpenZeppelin etc.)	392, 394, 403, 402, 404, 339, 341, 400	We recommend to import all packages from npm directly without flatten the contract. Functions could be modified or can be susceptible to vulnerabilities
#2	WizFarms	A floating pragma is set	3	The current pragma Solidity directive is „">=0.7.0 <0.9.0 "".
#3	Wizz20 Vesting	Missing Zero Address Validation (missing-zero-check)	100	Check that the address is not zero
#4	WizFarms	State variable visibility is not set	349, 378, 389	It is best practice to set the visibility of state variables explicitly
#5	Wizz20	A floating pragma is set	4	The current pragma Solidity directive is „">=0.7.0 <0.9.0 "".
#6	Wizz20	State variable visibility is not set	175	It is best practice to set the visibility of state variables explicitly

#7	Wizz20 Vesting	A floating pragma is set	3	The current pragma Solidity directive is „">=0.7.0 <0.9.0 "".

Informational issues

Issue	File	Type	Line	Description
#1	WizFarms	Misspelling word	355	_pasued should be _paused
#2	WizFarms	Missing require error message	582, 586, 590	Add an error message to describe the error
#3	WizFarms	Function has not effect if _tokenID is owner of _from	449	It can be replaced with modifier
#4	WizFarms	No return statement	507	There is no uint256 typed variable returned
#5	WizFarms	Functions that are not used	311, 321, 315246, 93,	Remove unused functions
#6	Wizz20	Functions that are not used	490	Remove unused functions
#7	Wizz20	Unimplemented functions	-	Implement all unimplemented functions in any contract you intend to use directly (not simply inherit from) Wizzy._afterTokenTransfer(address,address,uint256) (Wizz20.sol#490-494) Wizzy._approve(address,address,uint256) (Wizz20.sol#444-454) Wizzy._beforeTokenTransfer(address,address,uint256) (Wizz20.sol#470-474) Wizzy._burn(address,uint256) (Wizz20.sol#414-429) Wizzy._mint(address,uint256) (Wizz20.sol#390-401) Context._msgData() (Wizz20Vesting.sol#20-22) Context._msgSender() (Wizz20Vesting.sol#16-18)
#8	WizFarms	Unused state variables	400	Remove unused state variables

Audit Comments
29. November 2021:
- Read whole report for more information

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	**NOT 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	NOT 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

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY