SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC
Development | Marketing

MADE IN GERMANY

Wonka Capital

Audit

Security Assessment
09. May, 2023

For

WONKA
CAPITAL

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 security or functionality of the technology we agree to analyze.

Version	Date	Description
1.0	02. May 2023	• Layout project • Automated- /Manual-Security Testing • Summary

Note - This Audit report consists of a security analysis of the Wonka Capital smart contracts. This analysis did not include functional testing (or unit testing) of the contract's logic.

Network
Binance Smart Chain

Website
https://www.wonka.capital/

Telegram
https://t.me/wonkacapital

Twitter
https://twitter.com/wonkacap

TikTok
https://www.tiktok.com/@wonkacapital

Description

Wonka Capital is a DeFi token on Binance Smart Chain that utilizes a DAO model to solve the limitations of traditional investment funds. Our platform enables investors to have control and transparency over their investments through community-driven voting and transparent reporting.

Project Engagement

During the 29th Date of April 2023, **Wonka Capital 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

https://testnet.bscscan.com/address/0x80c8740732143ca5851fe6489839e73c428d608b#code

https://testnet.bscscan.com/address/0xbCF4E67C85BC58Be93DF315cA81f74223748f8c8#code

https://testnet.bscscan.com/address/0x47bda035d4bC2966d00b6C12c4221df2e7f352Fb#code

https://testnet.bscscan.com/address/0x3C313e9d8F44802fd244dF6E9379D3b9244b3453#code

https://testnet.bscscan.com/address/0xA25dCC4683Ebc738860988241164DA5e97d1E02a#code

https://testnet.bscscan.com/address/0x23ACf8f9295950DeDf4C1232c0954bd2d60c1f89#code

v1.1

WONKACAPITAL: 0xafD3bd098599C2ACC0D43eA99421FD92e6688564

WONKABRONZE: 0xeC0f1541f1E20bB2470aF743C9F7F6a779C05EB7

WONKASILVER: 0xE6a5e8a8b53Aaf7FFBfF3AB17737907457E45bF5

WONKAGOLD: 0x7f44c416313e9F9B59a3DA4Fb3DA4d478C47eE93

WONKASTAKE: 0x4C7aa7E331eA9b719B16a6118880A049119Bc8A9

WONKAEXCHANGE: 0xD394C1558c46bf90dbf9622C70c9E0C46e22132C

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.

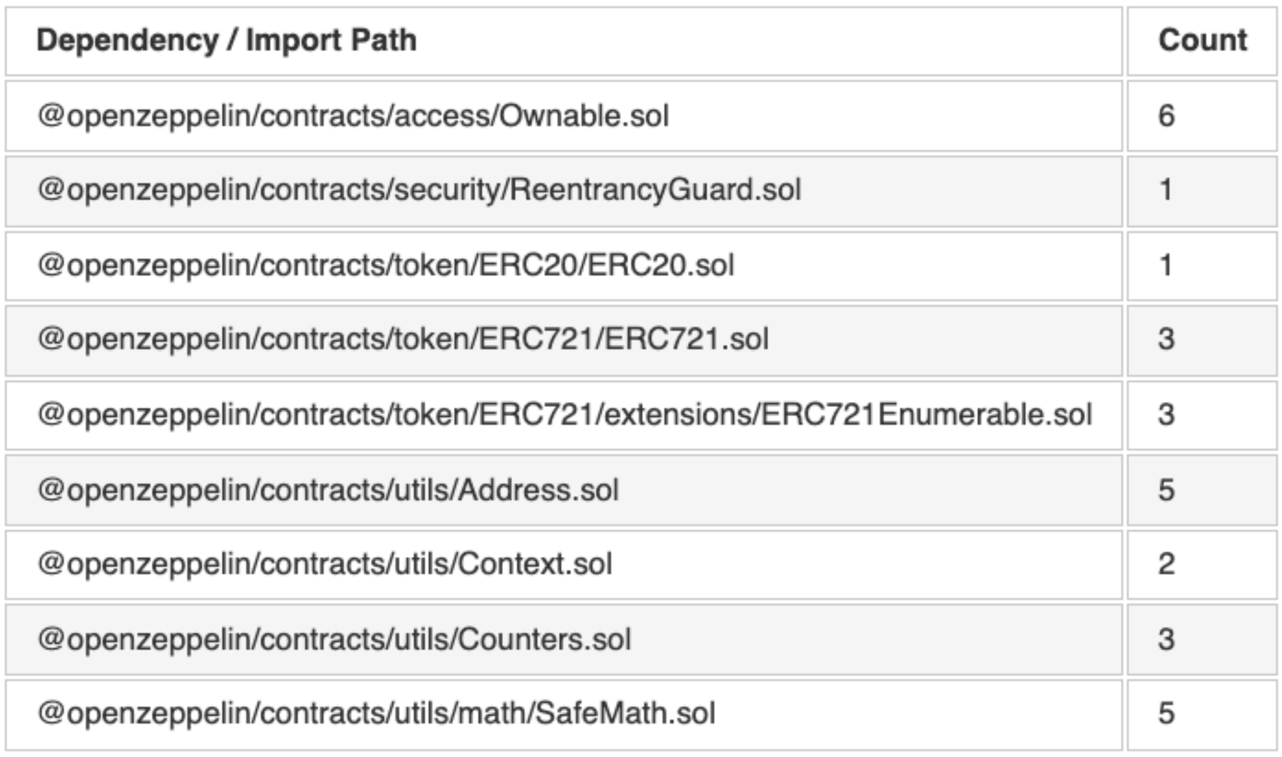
2. Testing and automated analysis that includes the following:
    - i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    - ii) Symbolic execution, which is analysing a program 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
@openzeppelin/contracts/access/Ownable.sol	6
@openzeppelin/contracts/security/ReentrancyGuard.sol	1
@openzeppelin/contracts/token/ERC20/ERC20.sol	1
@openzeppelin/contracts/token/ERC721/ERC721.sol	3
@openzeppelin/contracts/token/ERC721/extensions/ERC721Enumerable.sol	3
@openzeppelin/contracts/utils/Address.sol	5
@openzeppelin/contracts/utils/Context.sol	2
@openzeppelin/contracts/utils/Counters.sol	3
@openzeppelin/contracts/utils/math/SafeMath.sol	5

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.

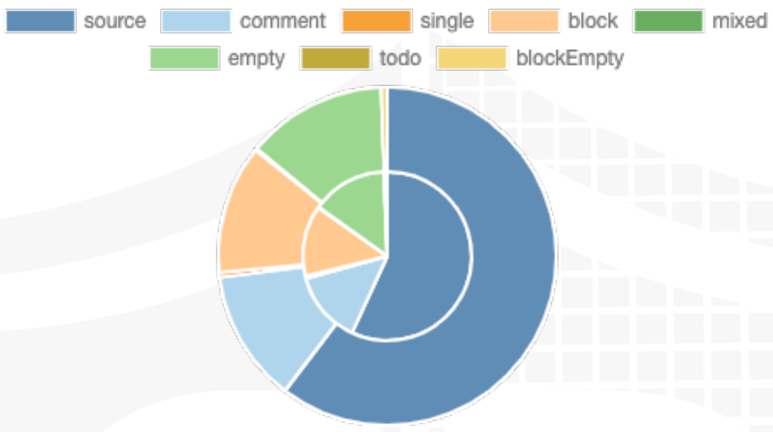
v1.0

File Name	SHA-1 Hash
contracts/nft/WonkaBronze.sol	6945e50c9b2f38855f8687f43aa97e923e2cc328
contracts/nft/WonkaSilver.sol	dc1b4cdc84cd0aa5c13c13e264c052e812f614dc
contracts/nft/WonkaGold.sol	5e5641ea7eb6a18f0501082c6ed3750bb1561cfe
contracts/stake/WonkaStake.sol	a1070d1940342c879d27b1a8ded7e35a3f6a083f
contracts/token/WonkaCapital.sol	6b5d1b53a68fe4ccc32b8fee5dacd30f7e744742
contracts/otc/WonkaExchange.sol	c0bf16c842052355a2109f5b33851e57b6c1908b

Metrics

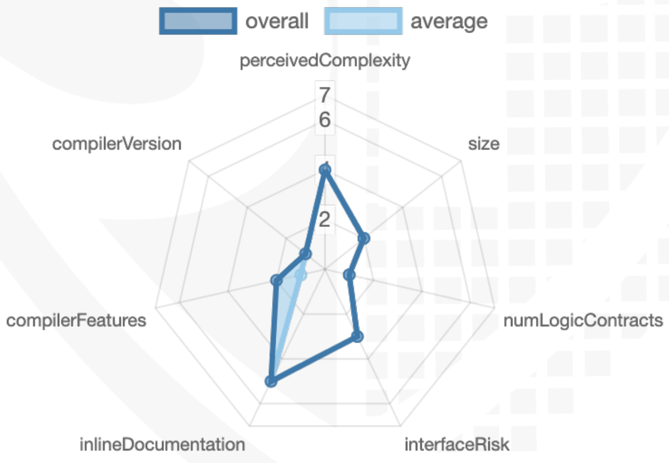
Source Lines

v1.0



Risk Level

v1.0



Capabilities

Components

📝Contracts	📚Libraries	🔍Interfaces	🎨Abstract
6	0	5	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
92	6

External	Internal	Private	Pure	View
68	103	2	5	52

StateVariables

Total	🌐Public
75	54

Capabilities

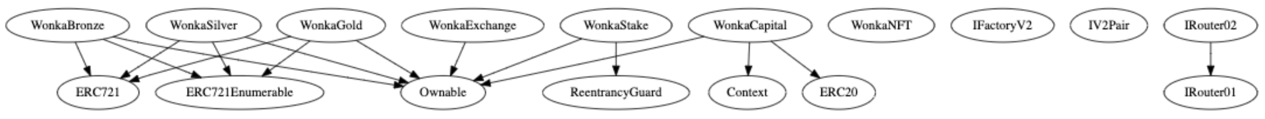
Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts
`^0.8.11` `>=0.8.11 <0.9.0`		`yes`	----	----

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

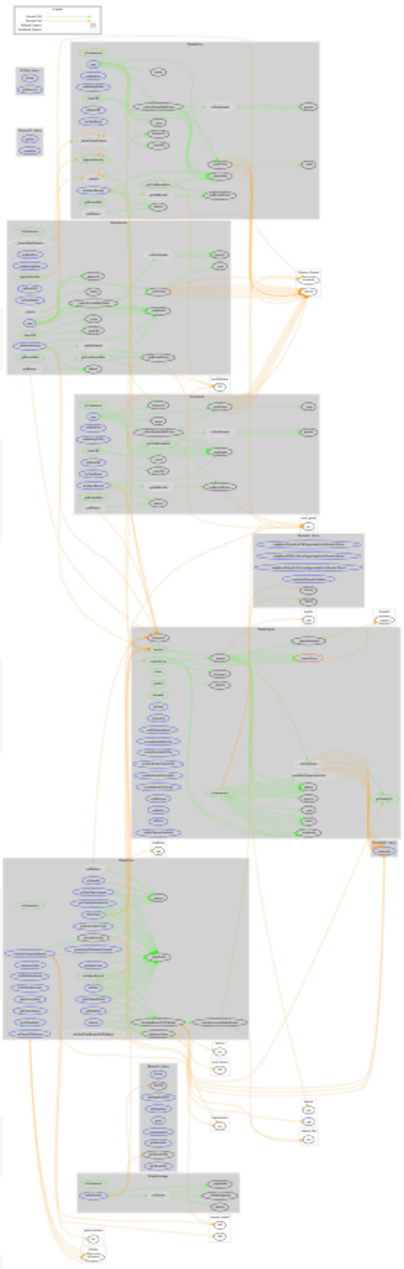
♻️ TryCatch	Σ Unchecked
----	----

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

ERC721				
Function	**Description**	**Exist**	**Tested**	**Verified**
BalanceOf	Count all NFTs assigned to an owner	✓	✓	✓
OwnerOf	Find the owner of an NFT	✓	✓	✓
SafeTransferFrom	Transfers the ownership of an NFT from one address to another address	✓	✓	✓
SafeTransferFrom	See above - Difference is that this function has an extra data parameter	✓	✓	✓
TransferFrom	Transfer ownership of an NFT	✓	✓	✓
Approve	Change or reaffirm the approved address for an NFT	✓	✓	✓
SetApprovalForAll	Enable or disable approval for a third party ("operator") to manage all of \`msg.sender\`'s assets	✓	✓	✓
GetApproved	Get the approved address for a single NFT	✓	✓	✓
IsApprovedForAll	Query if an address is an authorized operator for another address	✓	✓	✓
SupportsInterface	Query if a contract implements an interface	✓	✓	✓
Name	Provides information about the name	✓	✓	✓
Symbol	Provides information about the symbol	✓	✓	✓
TokenURI	Provides information about the TokenUri	✓	✓	✓

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✓
Max / Total Supply	1_000_000_000_000		

Comments:

v1.0

- Owner cannot mint new tokens

Deployer cannot burn or lock user funds

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

Comments:

v1.0

- Owner can lock user funds by
    - blacklisting addresses
    - Setting max wallet amount to 0

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	✓	✓	✗

Comments:

v1.0

- Owner can pause the staking contract

Deployer cannot set fees

Name	Exist	Tested	Status
Deployer cannot set fees over 20%	✓	✓	✓
Deployer cannot set fees to nearly 100% or to 100%	✓	✓	✓

Comments:

v1.1

- Fees cannot be set without any limitations

Deployer can blacklist/antisnipe addresses

Name	Exist	Tested	Status
Deployer cannot blacklist/antisnipe addresses	✓	✓	✗

Comments:

v1.0

- Owner is able to blacklist addresses

Overall checkup (Smart Contract Security)

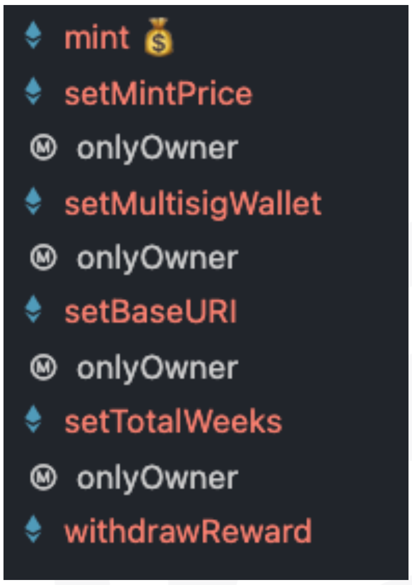
Tested	Verified
✓	✓

Legend

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

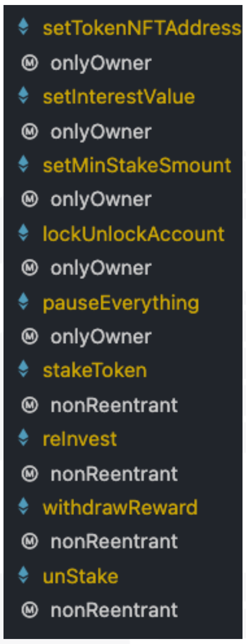
Modifiers and public functions
v1.0

WonkaGold



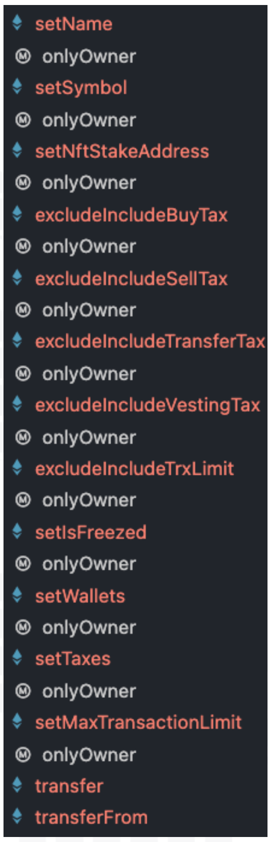
```
mint 💰
setMintPrice
onlyOwner
setMultisigWallet
onlyOwner
setBaseURI
onlyOwner
setTotalWeeks
onlyOwner
withdrawReward
```

WonkaStake



```
setTokenNFTAddress
onlyOwner
setInterestValue
onlyOwner
setMinStakeSmount
onlyOwner
lockUnlockAccount
onlyOwner
pauseEverything
onlyOwner
stakeToken
nonReentrant
reInvest
nonReentrant
withdrawReward
nonReentrant
unStake
nonReentrant
```

WonkaCapital



```
setName
onlyOwner
setSymbol
onlyOwner
setNftStakeAddress
onlyOwner
excludeIncludeBuyTax
onlyOwner
excludeIncludeSellTax
onlyOwner
excludeIncludeTransferTax
onlyOwner
excludeIncludeVestingTax
onlyOwner
excludeIncludeTrxLimit
onlyOwner
setIsFreezed
onlyOwner
setWallets
onlyOwner
setTaxes
onlyOwner
setMaxTransactionLimit
onlyOwner
transfer
transferFrom
```

Ownership/Authorized Privileges

❖ *WonkaGold.sol*

- Set mint price of token to any arbitrary value.
- Set multisig wallet address
- Set Total Weeks for rewards
- Set Base URI
- Owner can mint tokens without paying any fee but only once. Moreover if the owner wants then they can mint more than 1 token by transferring the ownership and then mint an NFT with the new wallet

Note- The privileges in the "*WonkaSilver*", and "*WonkaBronze*" are as same as WonkaGold

❖ *WonkaStake.sol*

- Set Token addresses
- Set Minimum stake amount to any arbitrary value

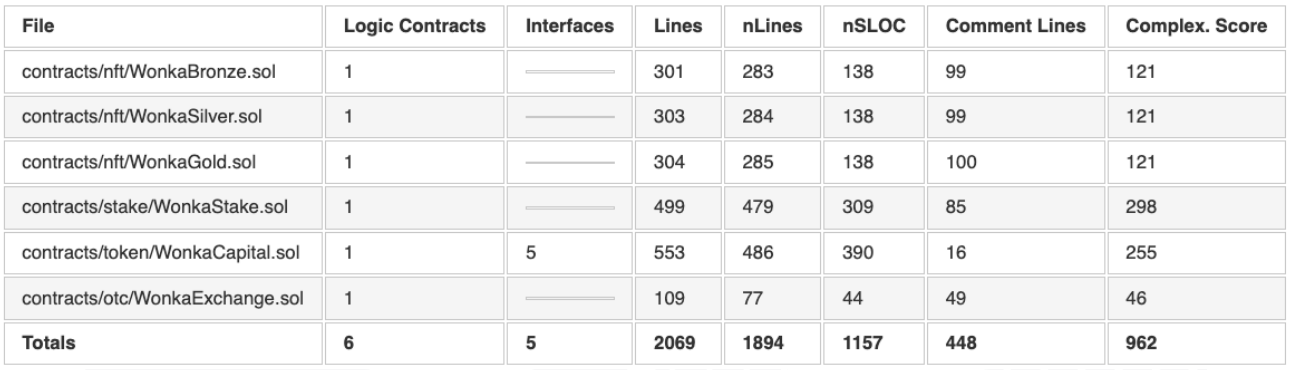
- Set interest values for different tiers to any arbitrary value including zero.
- Lock and Unlock accounts. Thus, locked accounts won't be able to stake and unstake tokens. Beware of it as it is a blacklisting mechanism
- Pause all public functionality in the contract

❖ *WonkaCapital.sol*
- Set name and symbol
- Set NFT stake addresses
- Include/Exclude accounts from tax, and transaction limit
- Blacklist wallets from transferring tokens
- Set taxes to a maximum of 20%
- Set max transaction limit to any arbitrary value including zero and may lock user funds

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/nft/WonkaBronze.sol	1	—	301	283	138	99	121
contracts/nft/WonkaSilver.sol	1	—	303	284	138	99	121
contracts/nft/WonkaGold.sol	1	—	304	285	138	100	121
contracts/stake/WonkaStake.sol	1	—	499	479	309	85	298
contracts/token/WonkaCapital.sol	1	5	553	486	390	16	255
contracts/otc/WonkaExchange.sol	1	—	109	77	44	49	46
Totals	**6**	**5**	**2069**	**1894**	**1157**	**448**	**962**

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

Issue	File	Type	Line	Description	Status
#1	WonkaStake.sol	Owner can lock funds	132	Owner can blacklist accounts and if it is done for an account that has already deposited tokens then the users' of those accounts won't be able to withdraw their tokens. We recommend removing the lock functionality from rewards and staked amount	Fixed
#2	WonkaCapital.sol	Fees can be 100% or more	373	Owner can set buy, sell, vesting and transfer fee to 100% or more which may lead to loss of user funds. We recommend putting a maximum limit on the taxes.	Fixed

Low issues

Issue	File	Type	Line	Description	Status
#1	All	A floating pragma is set	—	The current pragma Solidity directive is „“^0.8.11”.	Open
#2	WonkaCapital.sol	Missing Zero Address Validation (missing-zero-check)	280, 316	Check that the address is not zero	Open
#3	WonkaCapital.sol	Local variables shadowing	98, 99	Rename the local variables that shadow another component	Open

#4	WonkaCapital.sol	Missing Events Arithmetic	225-384	Emit an event for critical parameter changes	Open

Informational issues

Issue	File	Type	Line	Description	Status
#1	WonkaBronze.sol	State variables that could be declared constant (constable-states)	42, 44	Add the `constant` attributes to state variables that never change	Fixed
#2	WonkaStake.sol	Functions that are not used	412, 496	Remove unused functions. Before removing check the function, it could be possible, that you forget to implement it into the contract	Fixed
#3	WonkaCapital.sol	State variables that could be declared constant (constable-states)	100-104	Add the `constant` attributes to state variables that never change	Fixed
#4	WonkaGold.sol	State variables that could be declared constant (constable-states)	45, 47	Add the `constant` attributes to state variables that never change	Fixed
#5	All	NatSpec documentation missing	—	If you started to comment your code, also comment all other functions, variables etc.	Open

Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow 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 variables, functions etc. do.

09. May 2023:

- Unit tests with 100% code coverage was not provided to SolidProof so we cannot ensure complete functional correctness of the code's logic.
- We recommend Wonka Capital team to conduct unit and fuzz tests thoroughly to rule out possibilities of an unwanted logical and calculation errors.
- There is still an owner (Owner still has not renounced ownership)
- Owner can deploy a new version of the contract which can change any limit and give owner new privileges
- Read whole report and modifiers section 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	NOT 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	**NOT 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**

SolidProof_io @solidproof_io

Blockchain Security | Smart Contract Audits | KYC
Development | Marketing

MADE IN GERMANY

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