SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

UNIQO

Audit

Security Assessment
18. November, 2022

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 security or functionality of the technology we agree to analyze.

Version	Date	Description
1.0	20. September 2022	• Layout project • Automated- /Manual-Security Testing • Summary
1.1	20. September 2022	• Reaudit
1.2	07. October 2022	• Reaudit
2.0	10. November 2022	• Audit new contract with changes
3.0	18. November 2022	• Audit new contract with changes

Network
Binance Smart Chain (BEP20)

Website
https://uniqo.finance/

Twitter
https://twitter.com/UniqoFinance

Discord
https://discord.gg/uniqo

Description

Uniqo is a new and innovative DeFi product that facilitates stable and secure investment. Powered by a unique protocol that strategically manages inflation and deflation, Uniqo helps to protect investors from traditional risks while providing a way to earn passive income.

Project Engagement

During the 17th of September 2022, **UNIQO 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

v2.0

- 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

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-upgradeable/access/OwnableUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/proxy/utils/Initializable.sol	1
@openzeppelin/contracts-upgradeable/proxy/utils/UUPSUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/security/PausableUpgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC20/IERC20Upgradeable.sol	1
@openzeppelin/contracts-upgradeable/token/ERC20/utils/SafeERC20Upgradeable.sol	1
@openzeppelin/contracts/token/ERC20/IERC20.sol	1
@openzeppelin/contracts/utils/math/SafeMath.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/interfaces/IDEXRouter.sol	8b9759ee87d1de52a9bcfa7776cb2679c3bfd2bb
contracts/interfaces/IDEXPair.sol	94cc8bc4830bdf66c45cbd6f70fb942d5d11ab27
contracts/interfaces/IDEXFactory.sol	10c578bfce2404cd27be712e03a5564e8178c310
contracts/UNIQO.sol	ef2a47b930c3ac9015c0cba31aa028c1912b7a1c

v2.0

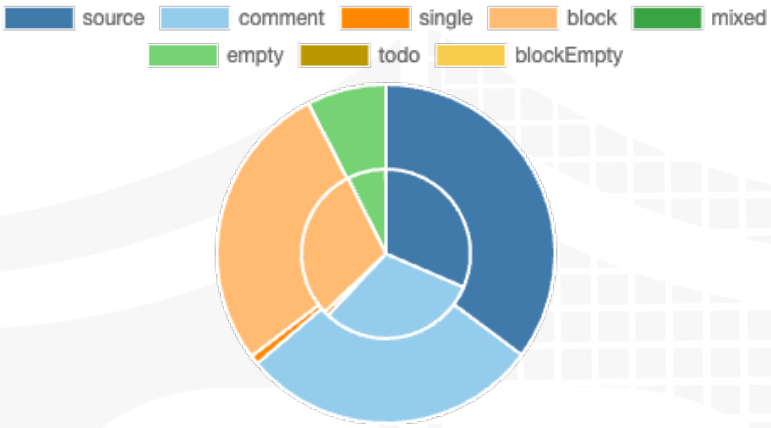
File Name	SHA-1 Hash
contracts/interfaces/IDEXRouter.sol	8b9759ee87d1de52a9bcfa7776cb2679c3bfd2bb
contracts/interfaces/IDEXPair.sol	94cc8bc4830bdf66c45cbd6f70fb942d5d11ab27
contracts/interfaces/IDEXFactory.sol	10c578bfce2404cd27be712e03a5564e8178c310
contracts/UNIQO.sol	8eee3fb6e84d30f911c951b65741464b78ec22a1

V3.0

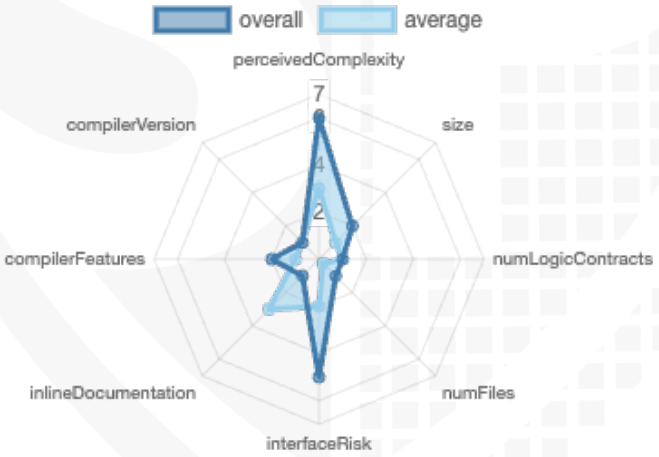
File Name	SHA-1 Hash
contracts/interfaces/IDEXRouter.sol	8b9759ee87d1de52a9bcfa7776cb2679c3bfd2bb
contracts/interfaces/IDEXPair.sol	94cc8bc4830bdf66c45cbd6f70fb942d5d11ab27
contracts/interfaces/IDEXFactory.sol	10c578bfce2404cd27be712e03a5564e8178c310
contracts/UNIQO.sol	4e38f5bb2664fe2ef751092edd855715bf0be453

Metrics

Source Lines
v3.0



Risk Level
v2.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
3.0	1	0	3	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.

Version	Public	Payable
3.0	68	2

Version	External	Internal	Private	Pure	View
3.0	50	55	14	6	28

State Variables

Version	Total	Public
3.0	58	43

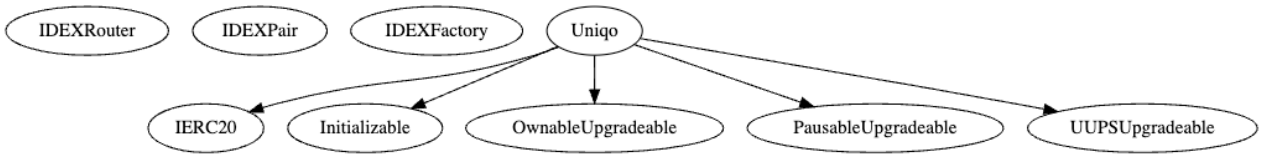
Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
3.0	`^0.8.13` `0.8.13`		`yes`		

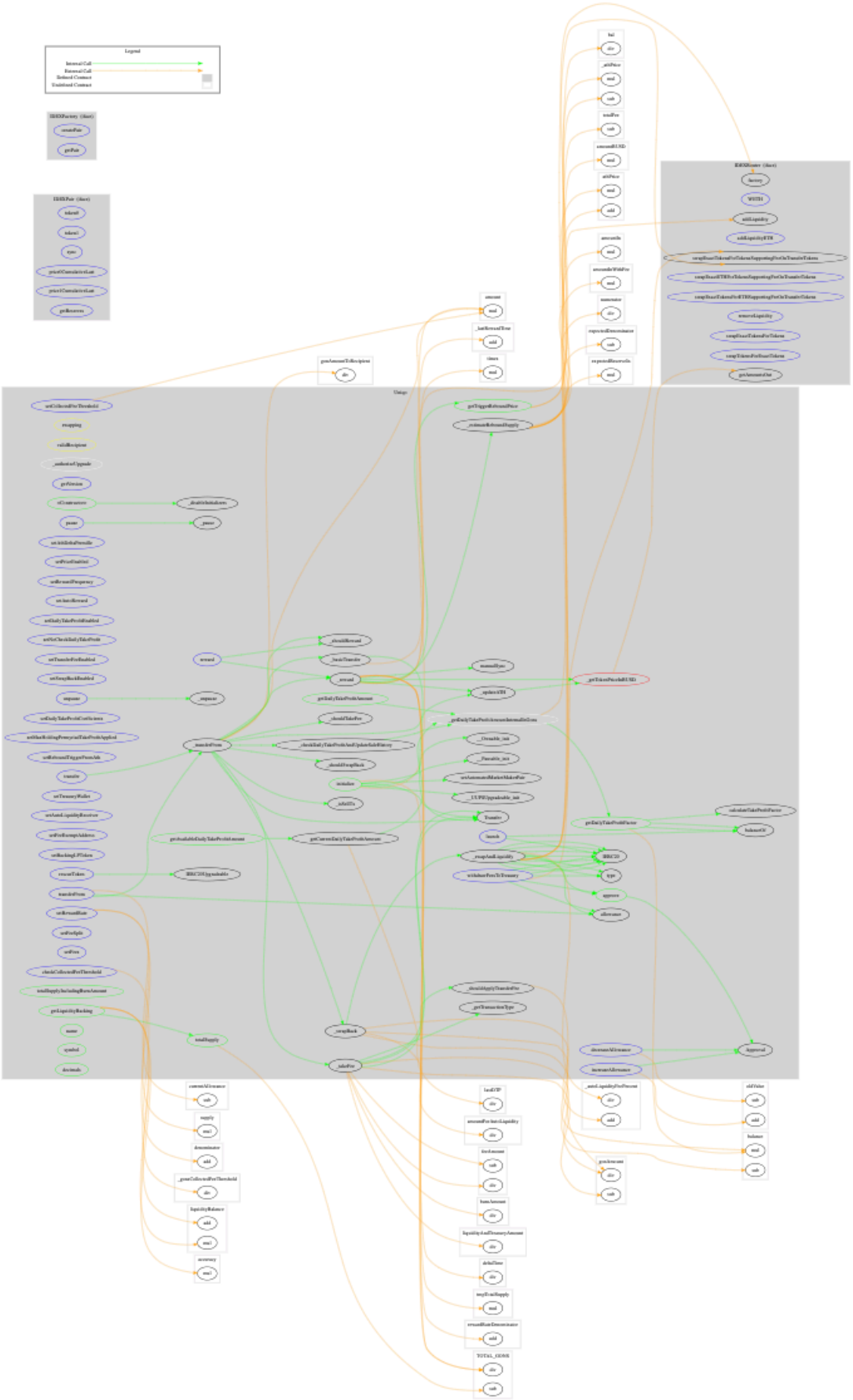
Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/ Create/ Create2
3.0	yes					

Inheritance Graph

v1.0



CallGraph
v3.0



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?	Yes

Comments:

v1.0

- Owner can deploy a new version of the contract which can change any limit and give owner new privileges
    - Be aware of this and do your own research for the contract which is the contract pointing to

Correct implementation of Token standard

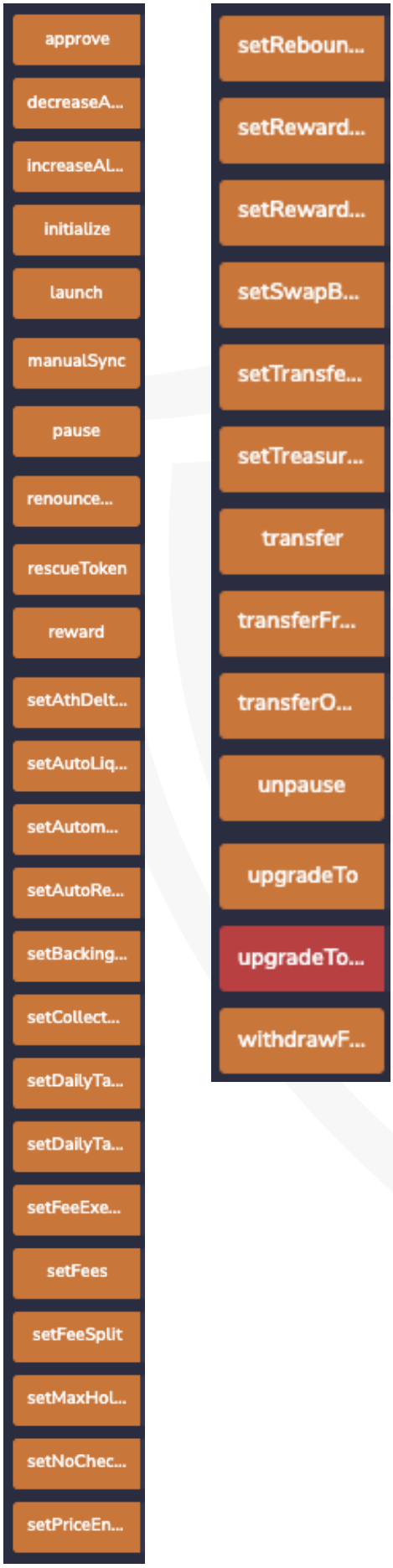
ERC20				
Function	**Description**	**Exist**	**Tested**	**Verified**
TotalSupply	Provides information about the total token supply	✓	✓	✓
BalanceOf	Provides account balance of the owner's account	✓	✓	✓
Transfer	Executes transfers of a specified number of tokens to a specified address	✓	✓	✓
TransferFrom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

Write functions of contract v1.0

v2.0


approve
decreaseA...
increaseAL...
initialize
launch
manualSync
pause
renounce...
rescueToken
reward
setAthDelt...
setAutoLiq...
setAutom...
setAutoRe...
setBacking...
setCollect...
setDailyTa...
setDailyTa...
setFeeExe...
setFees
setFeeSplit
setMaxHol...
setNoChec...
setPriceEn...
setReboun...
setReward...
setReward...
setSwapB...
setTransfe...
setTreasur...
transfer
transferFr...
transferO...
unpause
upgradeTo
upgradeTo...
withdrawF...

V3.0


approve
decreaseA...
increaseAL...
initialize
launch
manualSync
pause
renounce...
rescueToken
reward
setAthDelt...
setAutoLiq...
setAutom...
setAutoRe...
setBacking...
setCollect...
setDailyTa...
setDailyTa...
setFeeExe...
setFees
setFeeSplit
setMaxHoL...
setNoChec...
setPriceEn...
setReboun...
setReward...
setReward...
setSwapB...
setTransfe...
setTreasur...
transfer
transferFr...
transferO...
unpause
upgradeTo
upgradeTo...
withdrawF...

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	–	–	–
Max / Total Supply			1.000.000.000

Comments:

v1.0

- No mint function found in the contract but the owner can deploy a new version of the contract which could give the owner new privileges to mint new tokens
    - Be aware of this and do your own research for the contract which is the contract pointing to

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
  - Setting “takeProfitDenominator” to 0
  - Setting “rewardFrequency” to 0
  - Pausing the contract

- Tokens
  - will be burned while taking fee

v1.1

- Team has fixed takeProfitDenominator and rewardFrequency but team can still lock by pausing

V2.0

- Reward cannot be called when the contract is paused

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	✓	✓	✗

Comments:

v1.0

- Owner can pause contract

Deployer cannot set fees

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

Comments:

v1.0

- FeeDenominator can be set without any limitations
- transferFee can be set to 50%

v1.1

- Fees can be set to 51%

v1.2

- FeeDenominator is set to 100 and cannot be changed

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

- initialize
  - initializer
- transfer
  - validRecipient
  - whenNotPaused
- transferFrom
  - validRecipient
  - whenNotPaused
- decreaseAllowance
  - whenNotPaused
- increaseAllowance
  - whenNotPaused
- approve
  - whenNotPaused
- launch
  - onlyOwner
- setAthDeltaPermille
  - onlyOwner
- reward

```
withdrawFeesToTreasury
  swapping
  onlyOwner
setPriceEnabled
  onlyOwner
setRewardFrequency
  onlyOwner
setAutoReward
  onlyOwner
setDailyTakeProfitEnabled
  onlyOwner
setNoCheckDailyTakeProfit
  onlyOwner
setTransferFeeEnabled
  onlyOwner
setSwapBackEnabled
  onlyOwner
setCollectedFeeThreshold
  onlyOwner
setDailyTakeProfitCoefficients
  onlyOwner
setMaxHoldingPercentTakeProfitApplied
  onlyOwner
setReboundTriggerFromAth
  onlyOwner
setRewardRate
  onlyOwner
setTreasuryWallet
  onlyOwner
setAutoLiquidityReceiver
  onlyOwner
setFeeExemptAddress
  onlyOwner
setBackingLPToken
  onlyOwner
pause
  onlyOwner
unpause
  onlyOwner
rescueToken
  onlyOwner
setFeeSplit
  onlyOwner
setFees
  onlyOwner
setAutomatedMarketMakerPair
  onlyOwner
manualSync
```

Note: imported libraries was not listed down below

Comments

- *Deployer can set following state variables without any limitations*
  - autoLiquidityFeePercent
    - Can be set to 100 if “treasuryFeePercent” and “burnFeePercent” is set to 0
  - treasuryFeePercent
    - Can be set to 100 if “autoLiquidityFeePercent” and “burnFeePercent” is set to 0
  - burnFeePercent
    - Can be set to 100 if “autoLiquidityFeePercent” and “treasuryFeePercent” is set to 0
  - rewardRate

    - rewardRateDenominator
    - negativeFromAthPercentDenominator
    - negativeFromAthPercent
    - maxHoldingPercentTakeProfitApplied
    - coefficientA
    - coefficientB
    - takeProfitDenominator
    - _gonsCollectedFeeThreshold

- *Deployer can enable/disable following state variables*
    - automatedMarketMakerPairs
    - _paused
    - isFeeExempt
    - swapBackEnabled
    - transferFeeEnabled
    - _noCheckDailySellLimit
    - dailyTakeProfitEnabled
    - autoReward
    - priceEnabled

- *Deployer can set following addresses*
    - pair
    - autoLiquidityReceiver
    - treasury

- *Existing Modifiers*
    - Swapping
    - onlyOwner
    - validRecipient

- Anybody can call the initialize function. We recommend you to add a modifier that only the owner is able to call this function.
- Owner can pass own contract address to the “rescueToken” function with withdraw native tokens
- Liquidity goes to the “autoLiquidityReceiver” which is a private key wallet. Be aware of this because this wallet is able to drain the liquidity

V1.1

- The team fixed the “initialize” call from anybody

V2.0

- initialize
  - initializer
- transfer
  - validRecipient
  - whenNotPaused
- transferFrom
  - validRecipient
  - whenNotPaused
- decreaseAllowance
  - whenNotPaused
- increaseAllowance
  - whenNotPaused
- approve
  - whenNotPaused
- launch
  - onlyOwner
- setAthDeltaPermille
  - onlyOwner
- reward
  - whenNotPaused
- withdrawFeesToTreasury
  - swapping
  - onlyOwner
- setPriceEnabled
  - onlyOwner
- setRewardFrequency
  - onlyOwner
- setAutoReward
  - onlyOwner
- setDailyTakeProfitEnabled
  - onlyOwner
- setNoCheckDailyTakeProfit
  - onlyOwner
- setTransferFeeEnabled
  - onlyOwner
- setSwapBackEnabled
  - onlyOwner
- setCollectedFeeThreshold
  - onlyOwner
- setDailyTakeProfitCoefficients
  - onlyOwner
- setMaxHoldingPermyriadTakeProfitApplied
  - onlyOwner
- setReboundTriggerFromAth
  - onlyOwner
- setRewardRate
  - onlyOwner
- setTreasuryWallet
  - onlyOwner
- setAutoLiquidityReceiver
  - onlyOwner
- setFeeExemptAddress
  - onlyOwner
- setBackingLPToken
  - onlyOwner
- pause
  - onlyOwner
- unpause
  - onlyOwner
- rescueToken
  - onlyOwner
- setFeeSplit
  - onlyOwner
- setFees
  - onlyOwner
- setAutomatedMarketMakerPair
  - onlyOwner
- manualSync

V3.0

```
initialize
  initializer
transfer
  validRecipient
  whenNotPaused
transferFrom
  validRecipient
  whenNotPaused
decreaseAllowance
  whenNotPaused
increaseAllowance
  whenNotPaused
approve
  whenNotPaused
launch
  onlyOwner
setAthDeltaPermille
  onlyOwner
reward
  whenNotPaused
withdrawFeesToTreasury
  swapping
  onlyOwner
setPriceEnabled
  onlyOwner
setRewardFrequency
  onlyOwner
setAutoReward
  onlyOwner
setDailyTakeProfitEnabled
  onlyOwner
setNoCheckDailyTakeProfit
  onlyOwner
setTransferFeeEnabled
  onlyOwner
setSwapBackEnabled
  onlyOwner
setCollectedFeeThreshold
  onlyOwner
setDailyTakeProfitCoefficients
  onlyOwner
setMaxHoldingPermyriadTakeProfitApplied
  onlyOwner
setReboundTriggerFromAth
  onlyOwner
setRewardRate
  onlyOwner
```

```
setTreasuryWallet
  onlyOwner
setAutoLiquidityReceiver
  onlyOwner
setFeeExemptAddress
  onlyOwner
setBackingLPToken
  onlyOwner
pause
  onlyOwner
unpause
  onlyOwner
rescueToken
  onlyOwner
setFeeSplit
  onlyOwner
setFees
  onlyOwner
setAutomatedMarketMakerPair
  onlyOwner
manualSync
```

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

Source Units in Scope

v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
🔍	contracts/interfaces/IDEXRouter.sol	————	1	92	5	3	1	29	💰
🔍	contracts/interfaces/IDEXPair.sol	————	1	23	5	3	1	13	————
🔍	contracts/interfaces/IDEXFactory.sol	————	1	8	5	3	1	5	————
📝	contracts/UNIQO.sol	1	————	1462	1418	623	647	542	📩
📝🔍	**Totals**	**1**	**3**	**1585**	**1433**	**632**	**650**	**589**	💰📩

v2.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
🔍	contracts/interfaces/IDEXRouter.sol	————	1	92	5	3	1	29	💰
🔍	contracts/interfaces/IDEXPair.sol	————	1	23	5	3	1	13	————
🔍	contracts/interfaces/IDEXFactory.sol	————	1	8	5	3	1	5	————
📝	contracts/UNIQO.sol	1	————	1481	1439	643	647	551	📩
📝🔍	**Totals**	**1**	**3**	**1604**	**1454**	**652**	**650**	**598**	💰📩

v3.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
🔍	contracts/interfaces/IDEXRouter.sol	————	1	92	5	3	1	29	💰
🔍	contracts/interfaces/IDEXPair.sol	————	1	23	5	3	1	13	————
🔍	contracts/interfaces/IDEXFactory.sol	————	1	8	5	3	1	5	————
📝	contracts/UNIQO.sol	1	————	1513	1471	666	653	569	📩
📝🔍	**Totals**	**1**	**3**	**1636**	**1486**	**675**	**656**	**616**	💰📩

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

No medium issues

Low issues

No low issues

Informational issues

No informational issues

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.

18. November 2022:

- 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	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

SolidProofed

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY