SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

Pizon

Audit

Security Assessment
27. March, 2023

For

PIZON

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. March 2023	• Layout project • Automated- /Manual-Security Testing • Summary

Network
Arbitrum

Website
https://www.pizon.io/

Telegram
t.me/pizonannounce

Twitter
https://twitter.com/0xPizon

Description

At Pizon, we believe that the future of finance lies in decentralization. We envision a world where financial transactions are not controlled by a handful of centralized entities but rather by a network of participants who share in the decision-making process. Our protocol has been designed to make this vision a reality.

Pizon is built on the principles of transparency, security, and decentralization. With Pizon, users can rest assured that their transactions are safe and their assets are secure.

Project Engagement

During the Date of 20 March 2023, **Pizon 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://github.com/safudev/pizon-token
- Commit: ed74637

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:

```
@openzeppelin/contracts/token/ERC20/ERC20.sol
@openzeppelin/contracts/access/Ownable.sol
@openzeppelin/contracts/utils/math/SafeMath.sol
```

```
IRefKing
IReferral
ILockable
IFairLaunchToken
```

Tested Contract Files

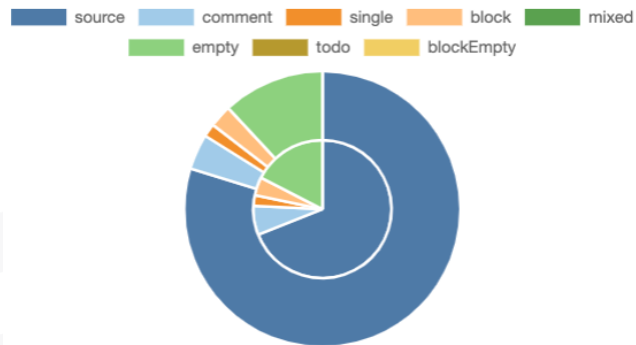This audit covered the following files listed below with a SHA-1 Hash.

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/ LoyaltyToken.sol	da84c0938770e89393203a382ae05667bce55220
contracts/Pause.sol	973cc1b9804fc6049bb13c9203ae515acd84555e
contracts/ FairLaunchToken.sol	e38d4648a5c943930d140b2e4b2d5c52c02ef506
contracts/Deploy.sol	c00d618a533ae0312b3babe0cfb152ffdc6aee7b
contracts/Information.sol	7fd5f8eb1f1eca7383dab1b393769ffbf24cfbef
contracts/ ReferralManager.sol	0e657acbdf0bdb1051f6df934a43bbe30f4bff2b

Metrics

Source Lines
v1.0

Risk Level
v1.0
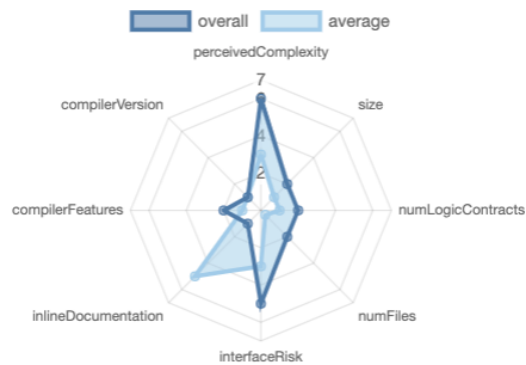
Capabilities

Components

📝Contracts	📚Libraries	🔍Interfaces	🎨Abstract
13	3	8	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
122	0

External	Internal	Private	Pure	View
72	103	0	5	94

StateVariables

Total	🌐Public
52	21

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	🌀 New/Create/Create2
———————	———————	———————	———————	———————	yes → NewContract:FairLaunchToken → NewContract:ReferralManager → NewContract:LoyaltyToken → NewContract:Pause

♻️ TryCatch	Σ Unchecked
yes	———————

Inheritance Graph
v1.0

LString   ILockableERC20   LoyaltyToken   Pause   FairLaunchToken   IRouter   LPercentage   LAmmDetector   ITokenCallback   Deploy   IToken   Information   ReferralManager

IERC20   ERC20   Ownable   Lockable   TransferTax   Referral   SecureApproval   CallbackableERC20   Importable   IFairLaunchToken

RefKing   IReferral   ILockable

IRefKing
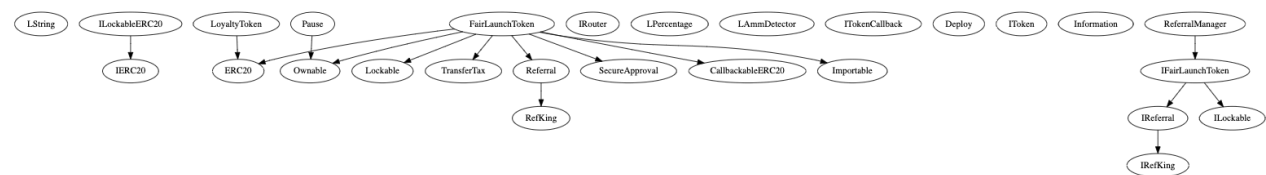
CallGraph
v1.0

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

- Owner cannot mint new tokens
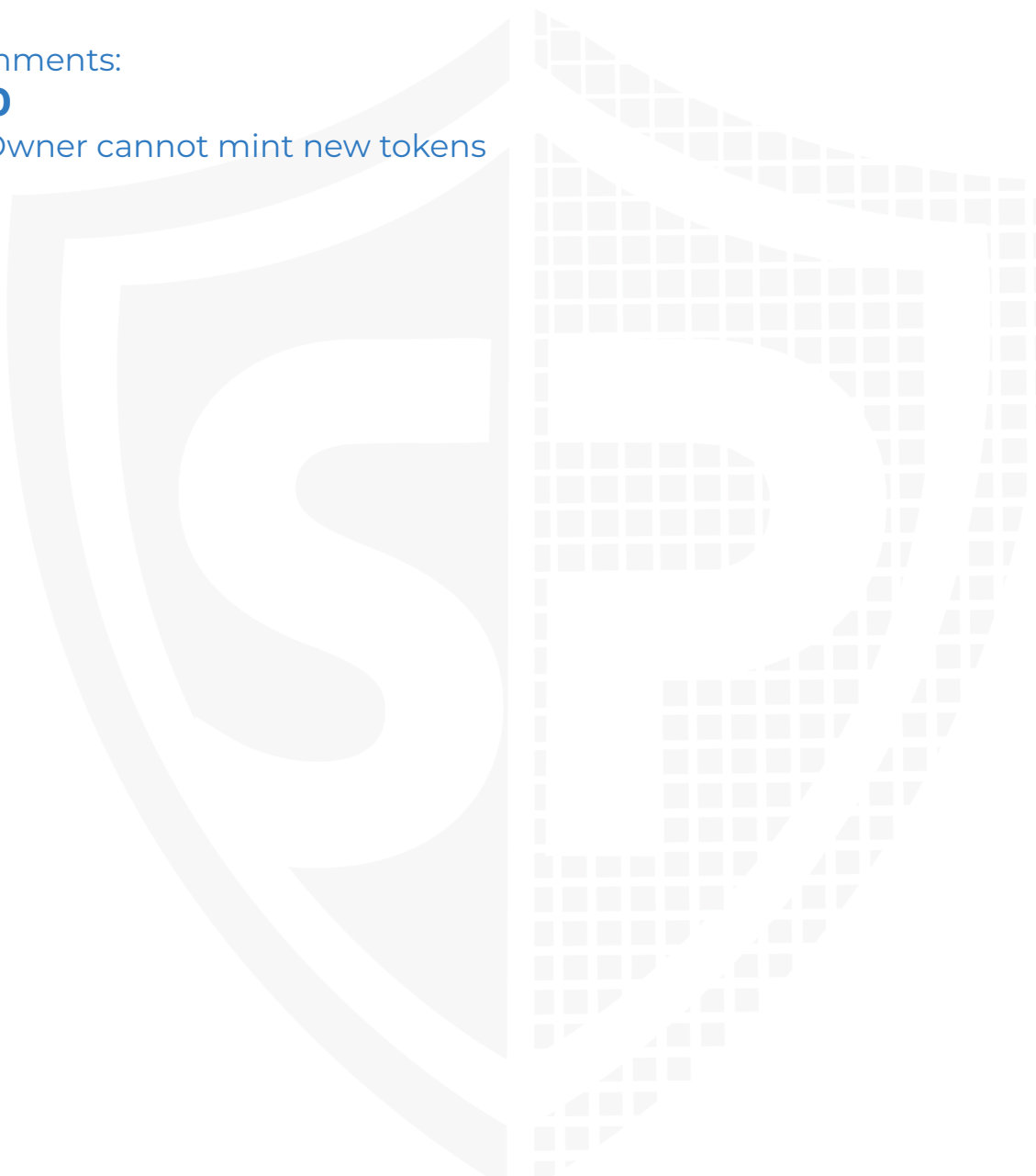
Deployer cannot burn or lock user funds

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

Comments:

v1.0

- Owner can lock user funds by
    - Calling the lock function, and it is possible to lock any number of tokens for any arbitrary time value

- Tokens
    - will be burned while tx
    - can be burned by the owner
    - can be burned by msg.sender

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 can set fees over 25%	✓	✓	✗
Deployer can set fees to nearly 100% or to 100%	✓	✓	✗

Comments:
v1.0

- Fees can be set upto a 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

FairLaunchToken.sol

- setDefaultLockPeriod
- Ⓜ onlyOwner
- setIsTaxerLocked
- Ⓜ onlyOwner
- setLockPeriodByFrom
- Ⓜ onlyOwner
- setTaxs
- Ⓜ onlyOwner
- taxIgnore
- Ⓜ onlyOwner
- setRefKingUnlockP
- Ⓜ onlyOwner
- setMaxSelfUnlockP
- Ⓜ onlyOwner
- configWar
- Ⓜ onlyOwner
- setReferralManager
- Ⓜ onlyOwner
- lock
- Ⓜ onlyOwner
- unlock
- Ⓜ onlyOwner
- afterTransferCallback
- setValidApprDuration
- Ⓜ onlyOwner
- setBeforeTransferTargets
- Ⓜ onlyOwner
- setAfterTransferTargets
- Ⓜ onlyOwner
- disableImport
- Ⓜ onlyOwner
- importAccount
- importAccounts

Ownership Privileges
- *FairLaunchToken.sol*
  - Set/Update default lock period to any arbitrary value
  - Lock the taxer which will result in lock of tax tokens for the tax address
  - Set the lock period for the "*from*" address to any arbitrary value at any time

23

- Set taxes up to a 100%
- Include/Exclude accounts from tax
- Set referral king unlock and max self unlock percentage to any arbitrary value including zero and 100%
- Set war duration, expInterval, and minRefInvValue to any number
- Set referral manager address which may lead to a smart contract with a different code and privileges.
- Lock/Unlock user funds at any moment in time
- Set app duration
- Set before and after transfer targets
- Disable account imports from the previous token address, but it is not possible to enable it again
- Users can set their or referrer percentage up to 100%

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/LoyaltyToken.sol	2	1	154	98	74	5	47
contracts/Pause.sol	1	——	21	15	7	5	7
contracts/FairLaunchToken.sol	10	2	1482	967	738	59	460
contracts/Deploy.sol	1	——	85	85	50	13	96
contracts/Information.sol	1	1	110	42	28	5	31
contracts/ReferralManager.sol	1	4	408	135	96	11	125
Totals	**16**	**8**	**2260**	**1342**	**993**	**98**	**766**

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

<div style="background-color: #7FE838; text-align: center;">**No critical issues**</div>

High issues

<div style="background-color: #7FE838; text-align: center;">**No high issues**</div>

Medium issues

Issue	File	Type	Line	Description
#1	FairLaunchToken.sol	Tax can be set upto 100%	450	Owner can set the tax percentage upto a 100% because the code allows it, but it is not recommended as it may lead to the loss of user funds.

Low issues

Issue	File	Type	Line	Description
#1	FairLaunchToken.sol	Missing Zero Address Validation (missing-zero-check)	327, 348, 479, 469,	Check that the address is not zero
#2	ReferralManager.sol	Redundant Code	319	The function is redundant as it only overrides a function and does not have any functionality.
#3	FairLaunchToken.sol	Local variables shadowing	1257, 1375, 1386	Rename the local variables that shadow another component
#4	FairLaunchToken.sol	Missing Events Arithmetic	348, 1434	Emit an event for critical parameter changes

Informational issues

Issue	File	Type	Line	Description
#1	Deploy.sol	State variables that could be declared immutable	17, 18, 19	Add the `constant` attributes to state variables that never change

#2	FairLau nchTok en.sol	Unused return values	54, 73, 93	Ensure that all the return values of the function calls are used and handle both success and failure cases if needed by the business logic
#3	Main	NatSpec documentation missing	-	If you started to comment your code, also comment 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/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.

27. March 2023:

- 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	**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**

[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**