SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

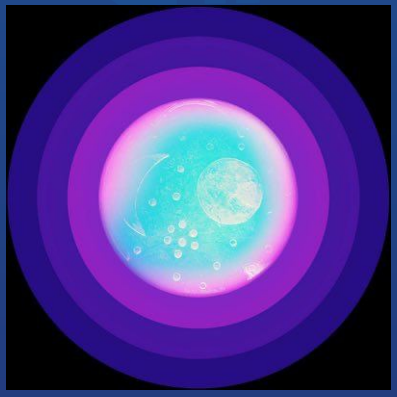
Dione Governance

Audit

Security Assessment

17. May, 2023

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	11. March 2023	• Layout project • Automated- /Manual-Security Testing • Summary
1.1	17. May 2023	• Reaudit

Note - This Audit report consists of a security analysis of the **Dione Protocol Governance** smart contracts. This analysis did not include functional testing (or unit testing) of the contract’s logic.

Network
Ethereum

Website
https://dioneprotocol.com

Telegram
t.me/DioneProtocol

Twitter
twitter.com/DioneProtocol

Instagram
instagram.com/DioneProtocol

YouTube
youtube.com/DioneProtocol

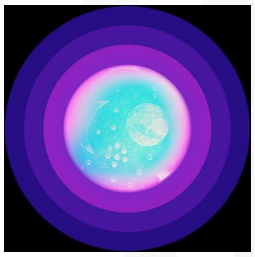
Description

This document, the Whitepaper, is the only source of truth regarding Dione. The technologies and products introduced in this document are currently in development and this document will continue to evolve. Therefore, this document does not aim to provide definite and absolute answers.

Project Engagement

During the Date of 10 May 2023, **Dione Protocol 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/DioneProtocol/Governance-sc
- Commit: ad2bf9f

v1.1

- https://github.com/DioneProtocol/Governance-sc
- Commit: 126b537

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.

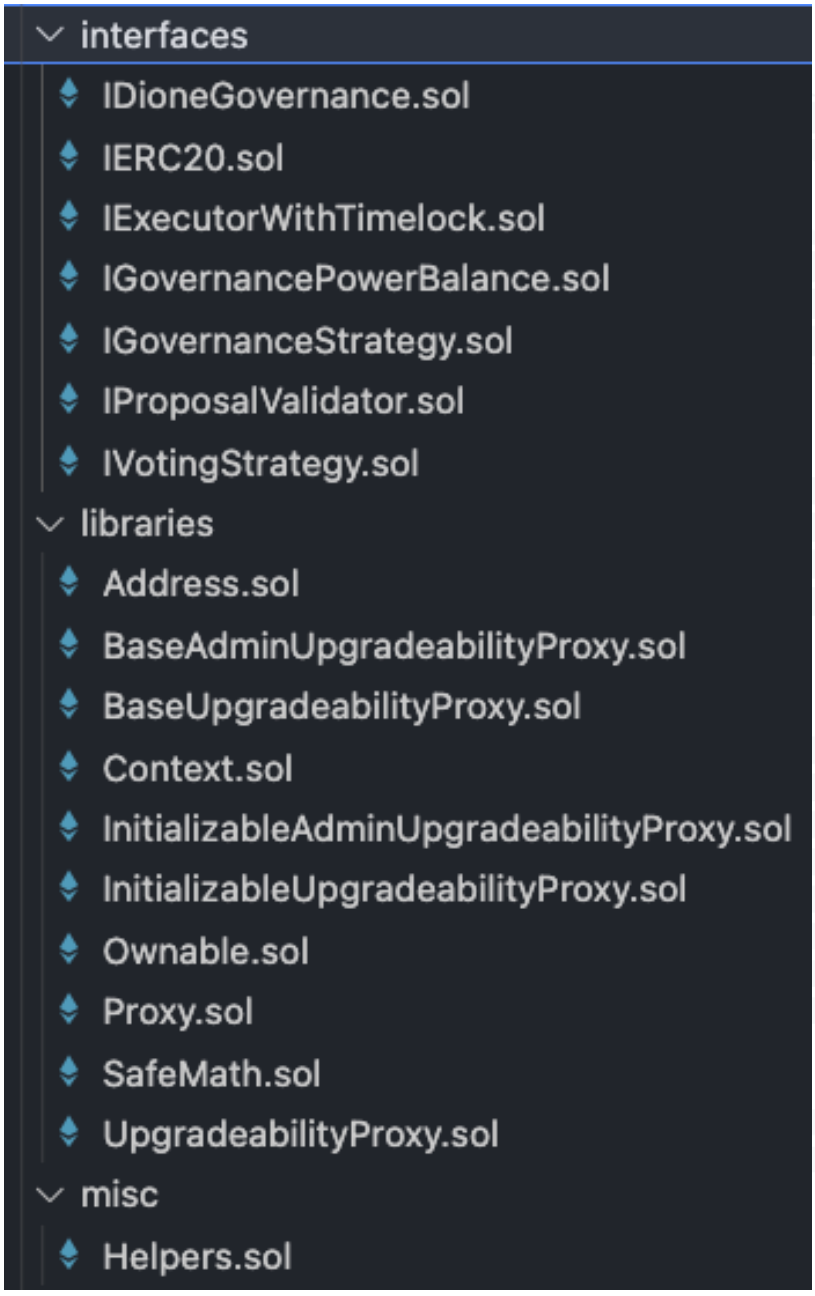
2. Testing and automated analysis that includes the following:
    i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    ii) Symbolic execution, which is analysing a program 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:
v1.0



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.

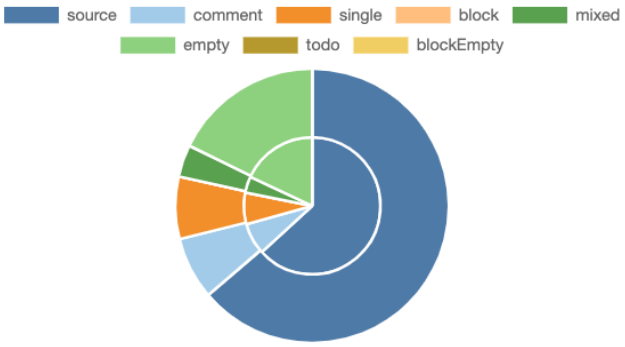
v1.0

File Name	SHA-1 Hash
DioneGovernance/contracts/governance/interfaces/IDioneGovernance.sol	b265a68da47e7c3e40f178788a125ccb03be03bc
DioneGovernance/contracts/governance/interfaces/IExecutorWithTimelock.sol	9684b79285a9bbf7ef2a7b822d27a5f904740742
DioneGovernance/contracts/governance/interfaces/IGovernancePowerBalance.sol	1dcd53a6b92a0915113bff5d13e5be79c85fe96f
DioneGovernance/contracts/governance/interfaces/IVotingStrategy.sol	dcd4fe266d86ba40d71f1130a4bca1d3910e54be
DioneGovernance/contracts/governance/interfaces/IGovernanceStrategy.sol	ce78c1afb0e78a37d1af99c3616a05fc0b3ebc78
DioneGovernance/contracts/governance/interfaces/IProposalValidator.sol	ef146c0fbf52140fd0d968f03034a89d35a82594
DioneGovernance/contracts/governance/interfaces/IERC20.sol	06f47a39b58cb970752cb480e84e50fee11480c4
DioneGovernance/contracts/governance/Executor.sol	01eddefea92d4ad7ceb1dcb4e94a5a5dc5881ef0
DioneGovernance/contracts/governance/ProposalValidator.sol	e1e7c6a2db5ec0fea501e838eff373a93360b143
DioneGovernance/contracts/governance/GovernanceStrategy.sol	49157b86533d26c9081869155556743f449d77fc
DioneGovernance/contracts/governance/DioneGovernance.sol	e2e84ede0948d4ba3b3c68c00baee5d11ba3afe7

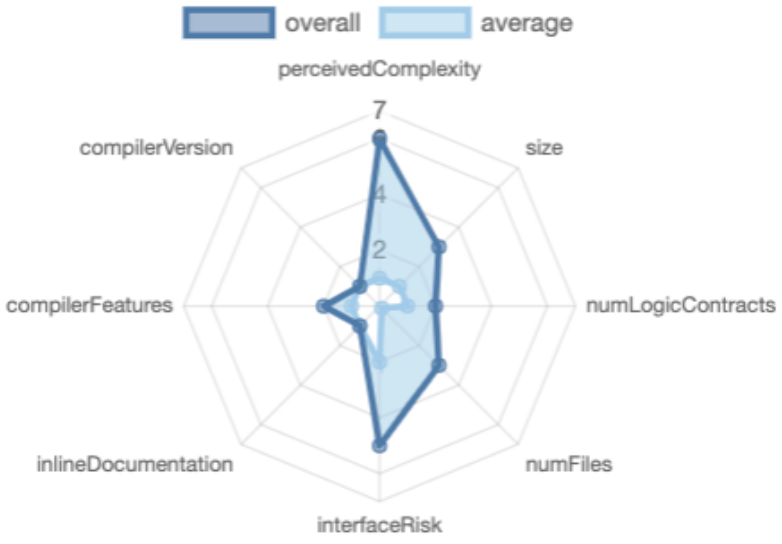
DioneGovernance/contracts/governance/ExecutorWithTimelock.sol	7050bb87dc9e85ae38059c1d0aa88e634c00c44b
DioneGovernance/contracts/governance/misc/Helpers.sol	479c8e51ec1bff097ef21869bb81678e0ff9dbd1
DioneGovernance/contracts/snapshot/DioneNativeBalance.sol	708bf6463e6f178d3304a6dd1e15f168dca28666
DioneGovernance/contracts/snapshot/DioneStakingBalance.sol	672c8279f430efb326286413f7b939176d5986f7
DioneGovernance/contracts/snapshot/DioneERC20Balance.sol	f9ba608897505c9f9a4b1e27ed2844a7e23deabc

Metrics

Source Lines
v1.0



Risk Level
v1.0



Capabilities

Components

📝Contracts	📚Libraries	🔍Interfaces	💊Abstract
15	3	9	2

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
117	10

External	Internal	Private	Pure	View
92	106	0	21	81

StateVariables

Total	🌐Public
32	17

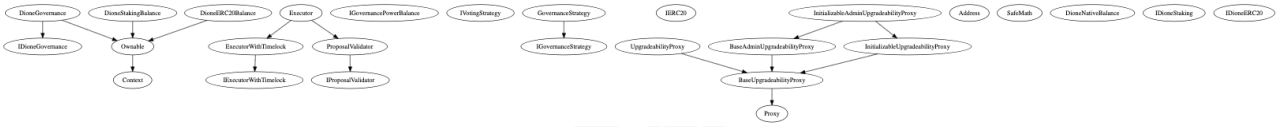
Capabilities

Solidity Versions observed	🧪 Experimental Features	💰 Can Receive Funds	🖥 Uses Assembly	💣 Has Destroyable Contracts
`^0.8.10` `^0.8.0` `0.8.12`		`yes`	`yes` (8 asm blocks)	

📩 Transfers ETH	⚡ Low-Level Calls	👥 DelegateCall	🧱 Uses Hash Functions	🦑 ECRecover	🌀 New/Create/Create2
		`yes`	`yes`	`yes`	

♻️ TryCatch	Σ Unchecked
	`yes`

Inheritance 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. Overall checkup (Smart Contract Security)

Is contract an upgradeable

Name	
Is contract an upgradeable?	No

Overall checkup (Smart Contract Security)

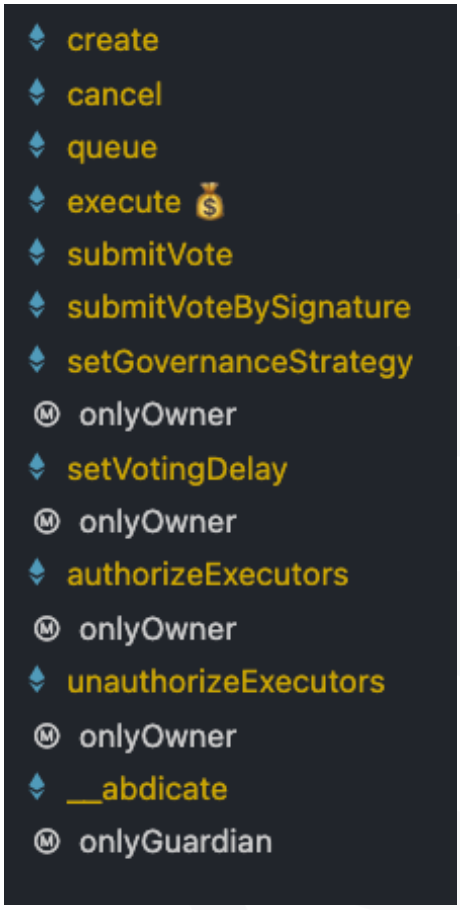
Tested	Verified
✓	✓

Legend

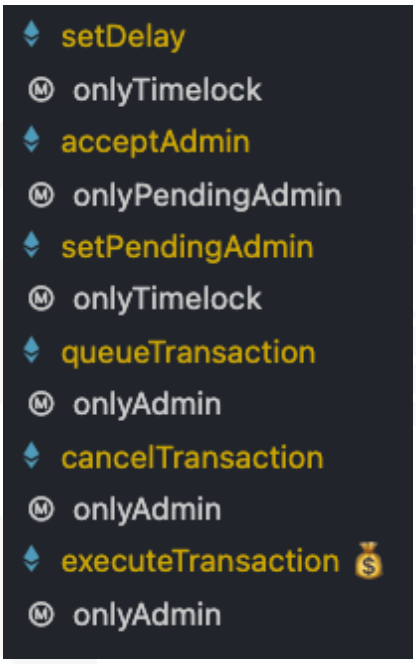
Attribute	Symbol
Verified / Checked	✓
Partly Verified	⚑
Unverified / Not checked	✗
Not available	-

Modifiers and public functions
v1.1

DioneGovernance



ExecutorWithTimelock



Ownership/Authority Privileges

❖ *DioneGovernance.sol*

- Set Governance Strategy
- Set voting delay
- Authorise/Unauthorize Executors
- Only Guardian Address can renounce the guardianship

❖ *ExecutorWithTimelock.sol*

- OnlyAdmin address can queue and cancel Transactions for a target address
- Only the contract address can Set pending admin address, and then the pending admin address can accept being an admin
- Only Admin address can manually execute the transaction

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

Source Units in Scope
v1.1

File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score
DioneGovernance/contracts/governance/interfaces/IDioneGovernance.sol		1	269	67	49	134	42
DioneGovernance/contracts/governance/interfaces/IExecutorWithTimelock.sol		1	184	90	36	106	26
DioneGovernance/contracts/governance/interfaces/IGovernancePowerBalance.sol		1	6	5	4		3
DioneGovernance/contracts/governance/interfaces/IVotingStrategy.sol		1	6	5	4		3
DioneGovernance/contracts/governance/interfaces/IGovernanceStrategy.sol		1	27	10	4	18	9
DioneGovernance/contracts/governance/interfaces/IProposalValidator.sol		1	124	14	5	70	27
DioneGovernance/contracts/governance/interfaces/IERC20.sol		1	7	5	4		5
DioneGovernance/contracts/governance/Executor.sol	1		29	29	20	7	7
DioneGovernance/contracts/governance/ProposalValidator.sol	1		174	140	69	58	51
DioneGovernance/contracts/governance/GovernanceStrategy.sol	1		81	69	33	27	23
DioneGovernance/contracts/governance/DioneGovernance.sol	1		489	448	248	100	182
DioneGovernance/contracts/governance/ExecutorWithTimelock.sol	1		280	254	140	75	91
DioneGovernance/contracts/governance/misc/Helpers.sol			23	23	17	4	10
DioneGovernance/contracts/snapshot/DioneNativeBalance.sol	1		16	16	6	9	3
DioneGovernance/contracts/snapshot/DioneStakingBalance.sol	1	1	60	51	34	13	29
DioneGovernance/contracts/snapshot/DioneERC20Balance.sol	1	1	41	34	19	13	20
Totals	**8**	**9**	**1816**	**1260**	**692**	**634**	**531**

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

Issue	File	Type	Line	Description	Status
#1	All	Contract doesn't import npm packages from source (like OpenZeppelin etc.)	—	We recommend to import all packages from npm directly without flatten the contract. Functions could be modified or can be susceptible to vulnerabilities	Fixed
#2	All	A floating pragma is set	—	The current pragma Solidity directive is „"^0.8.10" ".	Fixed
#3	DionGovernance.sol	Missing zero address validation	216, 224	Check that the address is not zero	Fixed
#4	ExecutorWithTimelock.sol	Missing zero address validation	76	Check that the address is not zero	Fixed

Informational issues

Issue	File	Type	Line	Description	Status

#1	All	NatSpec documentation missing	—	If you started to comment your code, also comment all other functions, variables etc.	Fixed

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.

17. May 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	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