SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Audit

Security Assessment

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

Network
Binance Smart Chain (BEP20)

Website
https://eterland.app/en/index.html

Telegram
https://t.me/joinchat/LWNIlZpeQtg5MWM5

Twitter
https://twitter.com/Eterlandnft

Facebook
https://www.facebook.com/Eterlandnft

Github
https://github.com/Eterland

Discord
https://discord.gg/3eyHQWFZ4b

Description

Under a dark and silent night an asteroid erupts with a terrifying hiss, hissing that heralded the end of an era, an era of gigantic dinosaurs where human beings still had no part.

That asteroid seemed that it was not an ordinary piece of rock, apparently it carried secrets that had to be revealed ...

All over the earth these mysterious fragments of ether were scattered, now this would become a field of investigation to reveal what this new world called Eterland has in store for us.

After the scattering of these fragments you will have to start investigating and help us reveal what is behind these mysterious fragments of ether.

With everyone's help we will be able to build a new world that hides behind this mysterious energy that emanates from the aforementioned fragments of ether, **but what could it be?** Join the adventure and help us reveal the multiple enigmas that have remained for years. .

Project Engagement

During the 4th of November 2021, **Eterland 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

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/AccessControl.sol	1
@openzeppelin/contracts/token/ERC20/ERC20.sol	1
@openzeppelin/contracts/token/ERC20/IERC20.sol	1
@openzeppelin/contracts/token/ERC20/utils/SafeERC20.sol	1
@openzeppelin/contracts/token/ERC721/ERC721.sol	1
@openzeppelin/contracts/utils/Context.sol	1
@openzeppelin/contracts/utils/Counters.sol	1

Tested Contract Files

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

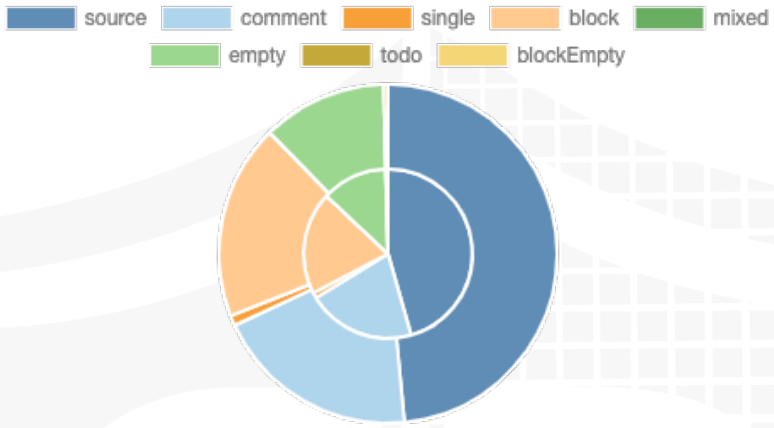
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

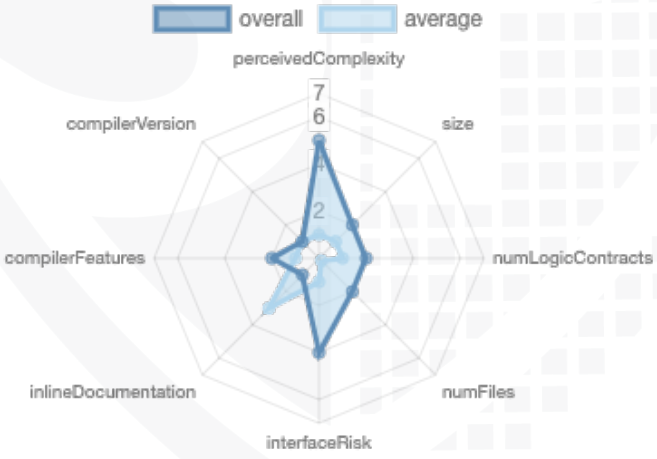
File Name	SHA-1 Hash
contracts/TeamVesting.sol	739a5811c6f586e43ddafb2226a9c32bd815c7ef
contracts/EterlandWorld.sol	9eea9e48740584bacf27e3af14e0cc724b5f7929
contracts/EterlandPresale.sol	ed06284ce12541ad6c256547ccdca6ab833ee857
contracts/Token.sol	da7f5e26a610a1ed9c560e37420b104774061757
contracts/abstract/Staking.sol	36ca5fb6a2da2c1ccf2d3b9dd5e2ecde3ef5bb3e
contracts/abstract/ExtendedAccessControl.sol	0da75e0f00f3b0e62c4da66b0b15f63517e8dede
contracts/abstract/EcoSystem.sol	c6fa4ed9367ac1fef07c8e471425cc75837b989e
contracts/abstract/Presale.sol	b194d3a2633a91b42752737f10ebfe8664f915c7
contracts/abstract/Moderator.sol	045d6ddc20d4fa0b88e29595459650af48c6ec1e
contracts/abstract/Liquidity.sol	6533c368951dcd6a9fd2c0ba68cdc14080d7feb9
contracts/abstract/Event.sol	75f719afa0d1a60a81511403f797d5d68d150c72
contracts/abstract/ERC20VestingWallet.sol	cf9791718ed59aa759d6219507db593aff352422
contracts/abstract/ContentCreator.sol	dc7526686d66f302b0174d798500836c8dc68a68
contracts/abstract/EterAccessControl.sol	100c2af72a0e26952974ed56382b8c68e2132cd3

Metrics

Source Lines
v1.0



Risk Level
v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	4	0	0	10

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	43	1

Version	External	Internal	Private	Pure	View
1.0	13	44	6	3	31

State Variables

Version	Total	Public
1.0	71	11

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`^0.8.0`		`yes`	**** (0 asm blocks)	

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

Scope of Work

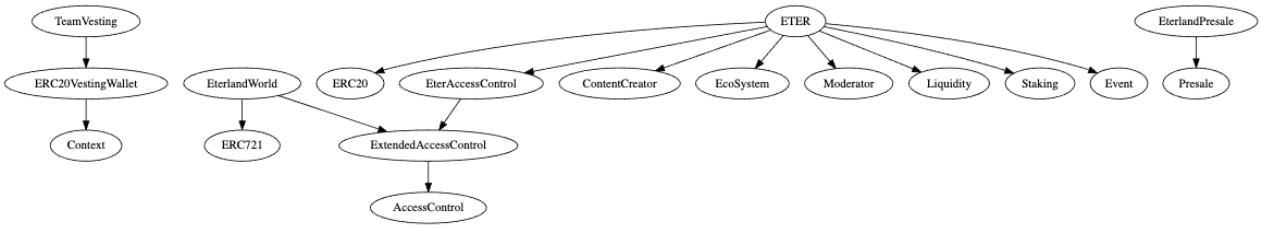
The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

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

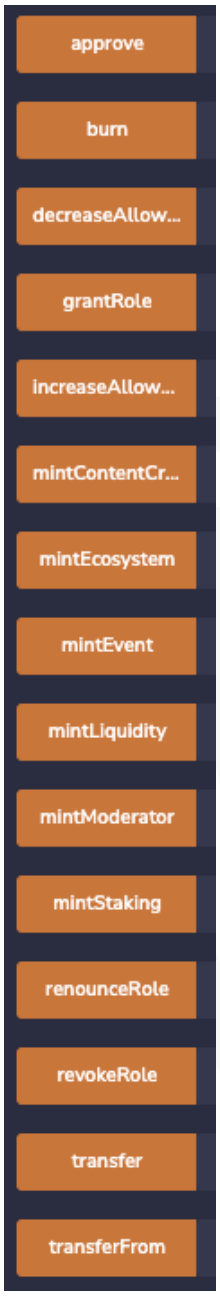
Correct implementation of Token standard

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

Token:



TeamVesting:

EterWorld:

EterPresale:

Deployer cannot mint any new tokens

Name	Exist	Tested	Verified	File
Deployer cannot mint	✓	✓	✗	Main
Comment	Line: -			

Max / Total Supply: 300.000.000

Comments:

v1.0

- Address with MinterRole can mint token
- Address with ECOSYSTEM_MINTER_ROLE can mint token with mintEcosystem function
    - Transaction will be reverted if totalMinted + amount is higher than MAX_SUPPLY

```
modifier canIncrementEcosystemSupply(uint256 _amount⬆){
    uint256 nextEcosystemSupply = _currentEcosystemSupply + _amount⬆;
    require(nextEcosystemSupply <= MAX_ECOSYSTEM_SUPPLY,string(
    abi.encodePacked("The max ecosystem supply is: ",MAX_ECOSYSTEM_SUPPLY)));
    require(nextEcosystemSupply <= getTotalEcosystemMint(),
    "can't mint that value yet");
    _;
}
```

Modifier before minting

- Address with LIQUIDITY_MINTER_ROLE can mint token with mintLiquidity function
    - Transaction will be reverted if totalLiquidityMinted + _amount is higher than MAX_MAC_LIQUIDITY_SELL, after that totalMinted + amount is higher than MAX_SUPPLY
- Address with MOD_MINTER_ROLE can mint token with mintModerator function
    - Transaction will be reverted if totalMinted + amount is higher than MAX_SUPPLY

```
modifier mintDailyLimitModerator(uint256 _amount⬆) {
    bool canMint = _lastMintTime == 0 ||
        block.timestamp > _lastMintTime + 1 days;

    require(canMint, "You can mint only once every day");
    require(_amount⬆ <= MAX_DAILY_MINT, "The amount exceed max daily mint");
    _;
}
```


Modifier before minting

- Address with STAKING_MINTER_ROLE can mint token with mintStaking function
    - Transaction will be reverted if totalMinted + amount is higher than MAX_SUPPLY

```
modifier mintDailyLimitStaking(uint256 _amount) {
    bool canMint = _lastMintTime == 0 ||
        block.timestamp > _lastMintTime + 1 days;

    require(canMint, "You can mint only once every day");
    require(_amount <= MAX_DAILY_MINT, "The amount exceed max daily mint");
    _;
}
```

Modifier before minting

- Address with EVENTS_MINTER_ROLE can mint token with mintEvent function
    - Transaction will be reverted if totalMinted + amount is higher than MAX_SUPPLY

```
modifier mintDailyLimitEvent(uint256 _amount) {
    bool canMint = _lastMintTime == 0 ||
        block.timestamp > _lastMintTime + 1 days;

    require(canMint, "You can mint only once every day");
    require(_amount <= MAX_DAILY_MINT, "The amount exceed max daily mint");
    _;
}
```

Modifier before minting

- Address with CONTENT_CREATOR_MINTER_ROLE can mint token with mintContentCreator function
    - Transaction will be reverted if totalMinted + amount is higher than MAX_SUPPLY

```
modifier mintDailyLimitContentCreators(uint256 _amount) {
    bool canMint = _lastMintTime == 0 ||
        block.timestamp > _lastMintTime + 1 days;

    require(canMint, "You can mint only once every day");
    require(_amount <= MAX_DAILY_MINT, "The amount exceed max daily mint");
    _;
}
```

Modifier before minting

Deployer cannot burn or lock user funds

Name	Exist	Tested	Verified
Deployer cannot lock	✓	✓	✗
Deployer cannot burn	✓	✓	✗

Comments:

v1.0

```
function burn(uint256 id) public{
    require(_isApprovedOrOwner(_msgSender(), id), "ERC721: transfer caller is not owner nor approved");
    _burn(id);
}
```

```
function _isApprovedOrOwner(address spender, uint256 tokenId) internal view virtual returns (bool) {
    require(_exists(tokenId), "ERC721: operator query for nonexistent token");
    address owner = ERC721.ownerOf(tokenId);
    return (spender == owner || getApproved(tokenId) == spender || isApprovedForAll(owner, spender));
}
```

-

```
function burn(uint256 _amount) public hasBurnRole {
    _burn(msg.sender, _amount);
    totalBurn += _amount;
}
```

Burn only with BurnRole

- Address with NFT_ADMINISTRATOR_ROLE can enable/disable tokens
  - This leads to the fact that NFT_ADMINISTRATOR_ROLE address can lock tokens

```
function changeStatus(uint256 id,bool status) hasNftAdministratorRole public {
    require(_exists(id),"didn't exist");
    _isDisabled[id] = status;
    address owner = ownerOf(id);

    emit StatusChanged(owner, id, status);
}
```


function _beforeTokenTransfer(
address from,
address to,
uint256 tokenId
) internal virtual override(ERC721) {
require(
_isDisabled[tokenId]==false,
"This nft is disabled"
);
}

Deployer cannot pause the contract

Name	Exist	Tested	Verified
Deployer cannot pause	✓	✓	✓

Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

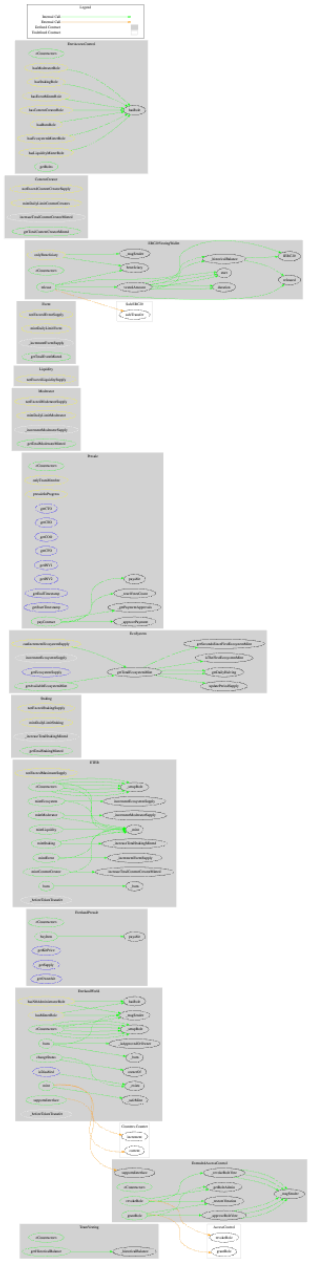
Legend

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

Role functions

- hasNftAdministratorRole
  - changeStatus
- hasMinterRole
  - Mint
- hasEcosystemMinterRole
  - mintEcosystem
- hasBurnRole
  - burn
- hasLiquidityMinterRole
  - mintLiquidity
- hasModeratorRole
  - mintModerator
- hasStakingRole
  - mintStaking
- hasEventMinterRole
  - mintEvent
- hasContentCreatorRole
  - mintContentCreator

CallGraph



Source Units in Scope
v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/TeamVesting.sol	1	———	22	22	10	8	9	———
📝	contracts/EterlandWorld.sol	1	———	130	116	63	31	56	🧱
📝	contracts/EterlandPresale.sol	1	———	120	116	59	36	44	💰💠
📝	contracts/Token.sol	1	———	205	179	90	68	90	———
🎨	contracts/abstract/Staking.sol	1	———	42	38	26	8	16	———
🎨	contracts/abstract/ExtendedAccessControl.sol	1	———	131	117	59	35	48	———
🎨	contracts/abstract/EcoSystem.sol	1	———	143	142	71	42	41	———
🎨	contracts/abstract/Presale.sol	1	———	206	202	141	32	53	💠
🎨	contracts/abstract/Moderator.sol	1	———	45	41	28	8	16	———
🎨	contracts/abstract/Liquidity.sol	1	———	18	18	10	5	6	———
🎨	contracts/abstract/Event.sol	1	———	47	43	26	11	16	———
🎨	contracts/abstract/ERC20VestingWallet.sol	1	———	111	111	56	39	43	———
🎨	contracts/abstract/ContentCreator.sol	1	———	46	42	28	8	16	———
🎨	contracts/abstract/EterAccessControl.sol	1	———	109	97	79	8	41	🧱
📝🎨	**Totals**	**14**	———	**1375**	**1284**	**746**	**339**	**495**	💰💠🧱

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

AUDIT PASSED

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	TeamVersting	A floating pragma is set	2	The current pragma Solidity directive is „“^0.8.0””.
#2	EterlandPresale	A floating pragma is set	2	The current pragma Solidity directive is „“^0.8.0””.
#3	EterlandWorld	A floating pragma is set	2	The current pragma Solidity directive is „“^0.8.0””.
#4	Token	A floating pragma is set	2	The current pragma Solidity directive is „“^0.8.0””.

Informational issues

Issue	File	Type	Line	Description
#1	EcoSystem	State variables that could be declared constant (constable-states)	20, 25	Add the `constant` attributes to state variables that never change

Audit Comments

09. November 2021:

- For more informations please read sections above

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	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

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY