QuillAudits

Audit Report
May, 2021

incub8

Contents

Scope of Audit

The scope of this audit was to analyze and document Incubate smart contract codebase for quality, security, and correctness.

Checked Vulnerabilities

We have scanned the smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that we considered:

- Re-entrancy
- Timestamp Dependence
- Gas Limit and Loops
- DoS with Block Gas Limit
- Transaction-Ordering Dependence
- Use of tx.origin
- Exception disorder
- Gasless send
- Balance equality
- Byte array
- Transfer forwards all gas
- ERC20 API violation
- Malicious libraries
- Compiler version not fixed
- Redundant fallback function
- Send instead of transfer
- Style guide violation
- Unchecked external call
- Unchecked math
- Unsafe type inference
- Implicit visibility level

Techniques and Methods

Throughout the audit of smart contract, care was taken to ensure:

- The overall quality of code.
- Use of best practices.
- Code documentation and comments match logic and expected behaviour.
- Token distribution and calculations are as per the intended behaviour mentioned in the whitepaper.
- Implementation of ERC-20 token standards.
- Efficient use of gas.
- Code is safe from re-entrancy and other vulnerabilities.

The following techniques, methods and tools were used to review all the smart contracts.

Structural Analysis
In this step we have analyzed the design patterns and structure of smart contracts. A thorough check was done to ensure the smart contract is structured in a way that will not result in future problems.
SmartCheck.

Static Analysis
Static Analysis of Smart Contracts was done to identify contract vulnerabilities. In this step a series of automated tools are used to test security of smart contracts.

Code Review / Manual Analysis
Manual Analysis or review of code was done to identify new vulnerability or verify the vulnerabilities found during the static analysis. Contracts were completely manually analyzed, their logic was checked and compared with the one described in the whitepaper. Besides, the results of automated analysis were manually verified.

Gas Consumption
In this step we have checked the behaviour of smart contracts in production. Checks were done to know how much gas gets consumed and possibilities of optimization of code to reduce gas consumption.

Tools and Platforms used for Audit
Remix IDE, Truffle, Truffle Team, Ganache, Solhint, Mythril, Slither, SmartCheck.

Issue Categories

Every issue in this report has been assigned with a severity level. There are four levels of severity and each of them has been explained below.

High severity issues

A high severity issue or vulnerability means that your smart contract can be exploited. Issues on this level are critical to the smart contract's performance or functionality and we recommend these issues to be fixed before moving to a live environment.

Medium level severity issues

The issues marked as medium severity usually arise because of errors and deficiencies in the smart contract code. Issues on this level could potentially bring problems and they should still be fixed.

Low level severity issues

Low level severity issues can cause minor impact and or are just warnings that can remain unfixed for now. It would be better to fix these issues at some point in the future.

Informational

These are severity four issues which indicate an improvement request, a general question, a cosmetic or documentation error, or a request for information. There is low-to-no impact.

Number of issues per severity

Type	High	Medium	Low	Informational
Open	0	0	0	2
Closed	0	0	1	0

Introduction

During the period of **May 20th, 2021 to May 22th, 2021** - QuillAudits Team performed a security audit for Incubate smart contracts.

The code for the audit was taken from following the link:
Contract Address
0x329e937a6e3887f126c7c5d9cebc0b6db8c4bfe1 | BscScan

Issues Found – Code Review / Manual Testing

High severity issues

No issues were found.

Medium severity issues

No issues were found.

Low level severity issues

1. Block timestamp

Line	Function Names
1054-1067	unlockCurentTokens
1079-1113	stackToken
1120-1140	unStakeToken

Description:
Do not rely on block.timestamp, now and blockhash as a source of randomness, unless you know what you are doing.

Both the timestamp and the block hash can be influenced by miners to some degree. Bad actors in the mining community can for example run a casino payout function on a chosen hash and just retry a different hash if they did not receive any money.

The current block timestamp must be strictly larger than the timestamp of the last block, but the only guarantee is that it will be somewhere between the timestamps of two consecutive blocks in the canonical chain.

Remediation:
Avoid relying on block.timestamp. The miner could cheat in the timestamp by a tolerance of 900 seconds, therefore, if the contract checks outside this interval, the contract is safe.

References:
Block-Protocol-2.0
Block and Transaction Properties

Status: Closed

The Incubate team decided to continue utilizing the block.timestamp as discussed internally. Nevertheless, the team should be aware of using this low-level function when the tolerance is less than 900 seconds.

Informational

2. State variables that could be declared constant

Line	Code

874-884	uint256 public SEED_SALE_TOTAL = 4444e18;
	uint256 public PRIVATE_SALE_TOTAL = 8888e18;
	uint256 public LIQUIDITY = 13332e18;
	uint256 public STACKING = 13332e18;
	uint256 public REWARDS = 19998e18;
	uint256 public ECOSYSTEM = 4444e18;
	uint256 public FOUNDERS = 4444e18;
	uint256 public INCUBATION = 13332e18;
	uint256 public COMMUNITY = 4444e18;
	uint256 public TEAM = 1115e18;
	uint256 public ADVISOR = 1115e18;

Description:
Constant state variables should be declared constant to save gas.

Remediation:
Add the constant attributes to state variables that never change.

References:
State variables that could be declared constant

Status: Open

3. Conformance to Solidity naming conventions

Line	Code
1079	function stackToken()

Description:
A misunderstanding function name could influence code readability, in some cases, may lead to bugs in the future.

Remediation:
The name should clearly, without ambiguity indicate what the function does. In this case, the function name should be distinguished between stackToken and stakeToken.

Status: Open

Functional test

Function Names	Testing results
unlockCurentTokens	Passed
stackToken	Passed
unStakeToken	Passed
initialSetup	Passed
pendingStake	Passed
setDevFundReciever	Passed
setVault	Passed
unlockAllToken	Passed
transfer	Passed
transferFrom	Passed
approve	Passed
burn	Passed
burnFrom	Passed
decreaseAllowance	Passed
increaseAllowance	Passed

Automated Testing

Slither

```
INFO:Detectors:
Incubate.unlockAllToken() (Incubate.sol#1047-1052) passes array Incubate.foundersTokens (Incubate.sol#894)by reference to Incubate.unlockCurentTokens(Incubate.LockedToken[]) (Incubate.sol#1054-1067)which only takes arrays by value
Incubate.unlockAllToken() (Incubate.sol#1047-1052) passes array Incubate.seedTokens (Incubate.sol#895)by reference to Incubate.unlockCurentTokens(Incubate.LockedToken[]) (Incubate.sol#1054-1067)which only takes arrays by value
Incubate.unlockAllToken() (Incubate.sol#1047-1052) passes array Incubate.privateTokens (Incubate.sol#896)by reference to Incubate.unlockCurentTokens(Incubate.LockedToken[]) (Incubate.sol#1054-1067)which only takes arrays by value
Incubate.unlockAllToken() (Incubate.sol#1047-1052) passes array Incubate.incubateTokens (Incubate.sol#897)by reference to Incubate.unlockCurentTokens(Incubate.LockedToken[]) (Incubate.sol#1054-1067)which only takes arrays by value
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#modifying-storage-array-by-value
INFO:Detectors:
Incubate.stackToken(uint256,uint256) (Incubate.sol#1079-1113) performs a multiplication on the result of a division:
        -diff = ((block.timestamp.sub(user.lastReward)).div(60).div(60).div(24)).div(30) (Incubate.sol#1084)
        -pending = diff.mul(round.rewardPerMonth) (Incubate.sol#1085)
Incubate.unStakeToken(uint256) (Incubate.sol#1120-1140) performs a multiplication on the result of a division:
        -diff = ((block.timestamp.sub(user.lastReward)).div(60).div(60).div(24)).div(30) (Incubate.sol#1128)
        -pending = diff.mul(round.rewardPerMonth) (Incubate.sol#1129)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#divide-before-multiply
INFO:Detectors:
Incubate.stackToken(uint256,uint256) (Incubate.sol#1079-1113) contains a tautology or contradiction:
        - user.amount < 0 (Incubate.sol#1096)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#tautology-or-contradiction
INFO:Detectors:
Incubate.initialSetup(address) (Incubate.sol#925-1045) ignores return value by unlockNow.add(totalEcoIncuComTeamAdvisor) (Incubate.sol#937)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#unused-return
```

```
INFO:Detectors:
Incubate.setDevFundReciever(address)._devaddr (Incubate.sol#1071) lacks a zero-check on :
                - devFundAddress = _devaddr (Incubate.sol#1072)
Incubate.setVault(address)._v (Incubate.sol#1075) lacks a zero-check on :
                - vault = _v (Incubate.sol#1076)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#missing-zero-address-validation
INFO:Detectors:
Incubate.unlockCurentTokens(Incubate.LockedToken[]) (Incubate.sol#1054-1067) uses timestamp for comparisons
        Dangerous comparisons:
        - i < current.length (Incubate.sol#1055)
        - current[i].unlockedTime > block.timestamp (Incubate.sol#1056)
Incubate.stackToken(uint256,uint256) (Incubate.sol#1079-1113) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > user.from && block.timestamp <= user.to (Incubate.sol#1086)
        - pending > 0 && pending <= round.maxReward (Incubate.sol#1087)
Incubate.unStakeToken(uint256) (Incubate.sol#1120-1140) uses timestamp for comparisons
        Dangerous comparisons:
        - block.timestamp > user.end (Incubate.sol#1124)
        - block.timestamp > user.from && block.timestamp <= user.to (Incubate.sol#1130)
        - pending > 0 && pending <= round.maxReward (Incubate.sol#1131)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#block-timestamp
INFO:Detectors:
Address.isContract(address) (Incubate.sol#212-221) uses assembly
        - INLINE ASM (Incubate.sol#219)
Address._verifyCallResult(bool,bytes,string) (Incubate.sol#333-350) uses assembly
        - INLINE ASM (Incubate.sol#342-345)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#assembly-usage
```

```
INFO:Detectors:
Address._verifyCallResult(bool,bytes,string) (Incubate.sol#333-350) is never used and should be removed
Address.functionCall(address,bytes) (Incubate.sol#265-267) is never used and should be removed
Address.functionCall(address,bytes,string) (Incubate.sol#275-277) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256) (Incubate.sol#290-292) is never used and should be removed
Address.functionCallWithValue(address,bytes,uint256,string) (Incubate.sol#300-307) is never used and should be removed
Address.functionStaticCall(address,bytes) (Incubate.sol#315-317) is never used and should be removed
Address.functionStaticCall(address,bytes,string) (Incubate.sol#325-331) is never used and should be removed
Address.isContract(address) (Incubate.sol#212-221) is never used and should be removed
Address.sendValue(address,uint256) (Incubate.sol#239-245) is never used and should be removed
Context._msgData() (Incubate.sol#19-22) is never used and should be removed
ERC20._setupDecimals(uint8) (Incubate.sol#718-720) is never used and should be removed
SafeMath.mod(uint256,uint256) (Incubate.sol#164-166) is never used and should be removed
SafeMath.mod(uint256,uint256,string) (Incubate.sol#180-183) is never used and should be removed
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#dead-code
INFO:Detectors:
Low level call in Address.sendValue(address,uint256) (Incubate.sol#239-245):
        - (success) = recipient.call{value: amount}() (Incubate.sol#243)
Low level call in Address.functionCallWithValue(address,bytes,uint256,string) (Incubate.sol#300-307):
        - (success,returndata) = target.call{value: value}(data) (Incubate.sol#305)
Low level call in Address.functionStaticCall(address,bytes,string) (Incubate.sol#325-331):
        - (success,returndata) = target.staticcall(data) (Incubate.sol#329)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#low-level-calls
INFO:Detectors:
Parameter Incubate.initialSetup(address)._dev (Incubate.sol#925) is not in mixedCase
Parameter Incubate.setDevFundReciever(address)._devaddr (Incubate.sol#1071) is not in mixedCase
Parameter Incubate.setVault(address)._v (Incubate.sol#1075) is not in mixedCase
Parameter Incubate.stackToken(uint256,uint256)._amount (Incubate.sol#1079) is not in mixedCase
Parameter Incubate.stackToken(uint256,uint256)._round (Incubate.sol#1079) is not in mixedCase
Parameter Incubate.unStakeToken(uint256)._round (Incubate.sol#1120) is not in mixedCase
Variable Incubate.SEED_SALE_TOTAL (Incubate.sol#874) is not in mixedCase
Variable Incubate.PRIVATE_SALE_TOTAL (Incubate.sol#875) is not in mixedCase
Variable Incubate.LIQUIDITY (Incubate.sol#876) is not in mixedCase
Variable Incubate.STACKING (Incubate.sol#877) is not in mixedCase
Variable Incubate.REWARDS (Incubate.sol#878) is not in mixedCase
Variable Incubate.ECOSYSTEM (Incubate.sol#879) is not in mixedCase
Variable Incubate.FOUNDERS (Incubate.sol#880) is not in mixedCase
Variable Incubate.INCUBATION (Incubate.sol#881) is not in mixedCase
Variable Incubate.COMMUNITY (Incubate.sol#882) is not in mixedCase
Variable Incubate.TEAM (Incubate.sol#883) is not in mixedCase
Variable Incubate.ADVISOR (Incubate.sol#884) is not in mixedCase
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#conformance-to-solidity-naming-conventions
INFO:Detectors:
Redundant expression "this (Incubate.sol#20)" inContext (Incubate.sol#14-23)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#redundant-statements
INFO:Detectors:
Incubate.ADVISOR (Incubate.sol#884) should be constant
Incubate.COMMUNITY (Incubate.sol#882) should be constant
Incubate.ECOSYSTEM (Incubate.sol#879) should be constant
Incubate.FOUNDERS (Incubate.sol#880) should be constant
Incubate.INCUBATION (Incubate.sol#881) should be constant
Incubate.LIQUIDITY (Incubate.sol#876) should be constant
Incubate.PRIVATE_SALE_TOTAL (Incubate.sol#875) should be constant
Incubate.REWARDS (Incubate.sol#878) should be constant
Incubate.SEED_SALE_TOTAL (Incubate.sol#874) should be constant
Incubate.STACKING (Incubate.sol#877) should be constant
Incubate.TEAM (Incubate.sol#883) should be constant
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#state-variables-that-could-be-declared-constant
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (Incubate.sol#495-497)
symbol() should be declared external:
        - ERC20.symbol() (Incubate.sol#503-505)
decimals() should be declared external:
        - ERC20.decimals() (Incubate.sol#520-522)
totalSupply() should be declared external:
```

```
INFO:Detectors:
name() should be declared external:
        - ERC20.name() (Incubate.sol#495-497)
symbol() should be declared external:
        - ERC20.symbol() (Incubate.sol#503-505)
decimals() should be declared external:
        - ERC20.decimals() (Incubate.sol#520-522)
totalSupply() should be declared external:
        - ERC20.totalSupply() (Incubate.sol#527-529)
balanceOf(address) should be declared external:
        - ERC20.balanceOf(address) (Incubate.sol#534-536)
transfer(address,uint256) should be declared external:
        - ERC20.transfer(address,uint256) (Incubate.sol#546-549)
approve(address,uint256) should be declared external:
        - ERC20.approve(address,uint256) (Incubate.sol#565-568)
transferFrom(address,address,uint256) should be declared external:
        - ERC20.transferFrom(address,address,uint256) (Incubate.sol#583-587)
increaseAllowance(address,uint256) should be declared external:
        - ERC20.increaseAllowance(address,uint256) (Incubate.sol#601-604)
decreaseAllowance(address,uint256) should be declared external:
        - ERC20.decreaseAllowance(address,uint256) (Incubate.sol#620-623)
burn(uint256) should be declared external:
        - ERC20Burnable.burn(uint256) (Incubate.sol#759-761)
burnFrom(address,uint256) should be declared external:
        - ERC20Burnable.burnFrom(address,uint256) (Incubate.sol#774-779)
owner() should be declared external:
        - Ownable.owner() (Incubate.sol#816-818)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Incubate.sol#835-838)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Incubate.sol#844-848)
unlockAllToken() should be declared external:
        - Incubate.unlockAllToken() (Incubate.sol#1047-1052)
setDevFundReciever(address) should be declared external:
        - Incubate.setDevFundReciever(address) (Incubate.sol#1071-1073)
setVault(address) should be declared external:
        - Incubate.setVault(address) (Incubate.sol#1075-1077)
stackToken(uint256,uint256) should be declared external:
        - Incubate.stackToken(uint256,uint256) (Incubate.sol#1079-1113)
pendingStake() should be declared external:
```

```
burn(uint256) should be declared external:
        - ERC20Burnable.burn(uint256) (Incubate.sol#759-761)
burnFrom(address,uint256) should be declared external:
        - ERC20Burnable.burnFrom(address,uint256) (Incubate.sol#774-779)
owner() should be declared external:
        - Ownable.owner() (Incubate.sol#816-818)
renounceOwnership() should be declared external:
        - Ownable.renounceOwnership() (Incubate.sol#835-838)
transferOwnership(address) should be declared external:
        - Ownable.transferOwnership(address) (Incubate.sol#844-848)
unlockAllToken() should be declared external:
        - Incubate.unlockAllToken() (Incubate.sol#1047-1052)
setDevFundReciever(address) should be declared external:
        - Incubate.setDevFundReciever(address) (Incubate.sol#1071-1073)
setVault(address) should be declared external:
        - Incubate.setVault(address) (Incubate.sol#1075-1077)
stackToken(uint256,uint256) should be declared external:
        - Incubate.stackToken(uint256,uint256) (Incubate.sol#1079-1113)
pendingStake() should be declared external:
        - Incubate.pendingStake() (Incubate.sol#1115-1118)
Reference: https://github.com/crytic/slither/wiki/Detector-Documentation#public-function-that-could-be-declared-external
INFO:Slither:Incubate.sol analyzed (8 contracts with 75 detectors), 80 result(s) found
INFO:Slither:Use https://crytic.io/ to get access to additional detectors and Github integration
```

Theo

```
enderphan@enderphan Incubate % theo --rpc-http HTTP://127.0.0.1:7545
The account's private key (input hidden)
>
Contract to interact with
> 0xa3D2c14c7dE7b0242d9e11A3D9F6e83f73BA2A30
Scanning for exploits in contract: 0xa3D2c14c7dE7b0242d9e11A3D9F6e83f73BA2A30
Connecting to HTTP: HTTP://127.0.0.1:7545.
No exploits found. You're going to need to load some exploits.

Tools available in the console:
- `exploits` is an array of loaded exploits found by Mythril or read from a file
- `w3` an initialized instance of web3py for the provided HTTP RPC endpoint
- `dump()` writing a json representation of an object to a local file
```

Mythril

```
version: 0.8.12+commit.27d51765.Darwin.appleclang
enderphan@enderphan Incubate % myth a Incubate.sol
The analysis was completed successfully. No issues were detected.

enderphan@enderphan Incubate %
```

Manticore

```
enderphan@enderphan Incubate % manticore  Incubate.sol --contract Incubate
2021-05-22 11:46:30,215: [12438] m.c.manticore:WARNING: Manticore is only supported on Linux. Proceed at your own risk!
2021-05-22 11:46:30,340: [12438] m.main:INFO: Registered plugins: <class 'manticore.core.plugin.IntrospectionAPIPlugin'>, DetectInvalid, DetectIntegerOverfl
ow, DetectUninitializedStorage, DetectUninitializedMemory, DetectReentrancySimple, DetectReentrancyAdvanced, DetectUnusedRetVal, DetectSuicidal, DetectDeleg
atecall, DetectExternalCallAndLeak, DetectEnvInstruction, DetectManipulableBalance
2021-05-22 11:46:30,340: [12438] m.main:INFO: Beginning analysis
2021-05-22 11:46:30,346: [12438] m.e.manticore:INFO: Starting symbolic create contract


2021-05-22 18:55:31,678: [12438] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 833428678182770547088546190986472l
63367902633943, offset 128)
2021-05-22 18:55:32,071: [12438] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 833428678182770547088546190986472l
63367902633943, offset 129)
2021-05-22 18:55:32,380: [12438] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 833428678182770547088546190986472l
63367902633943, offset 130)
2021-05-22 18:55:32,685: [12438] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 833428678182770547088546190986472l
63367902633943, offset 131)
2021-05-22 18:55:33,085: [12438] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 833428678182770547088546190986472l
63367902633943, offset 132)
2021-05-22 18:55:33,413: [12438] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 833428678182770547088546190986472l
63367902633943, offset 133)
2021-05-22 18:55:33,925: [12438] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 833428678182770547088546190986472l
63367902633943, offset 134)
2021-05-22 18:55:34,239: [12438] m.e.detectors:WARNING: Potentially reading uninitialized memory at instruction (address: 833428678182770547088546190986472l
63367902633943, offset 135)
```

```
2021-05-22 18:58:04,294: [12438] m.e.manticore:INFO: Found a concrete globalsha3 b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x04' -> 62514009886607029107290561805838585334079798074568712924583230797734656856475
2021-05-22 18:58:54,579: [12438] m.e.manticore:INFO: Found a concrete globalsha3 b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x05' -> 15466780324412574526674567355828149599927827828167319226912722823335616997 60
2021-05-22 19:00:54,606: [12438] m.e.manticore:INFO: Found a concrete globalsha3 b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x91\xfc<\x05\x8a\xf7\x13
\xa0}\xbf\x97\xe7/\xffr\x9e\xaf\xb1\xdf\xd7\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x01' -> 54700993788853262460070280558070496612826684789945625998946216551350933238266
2021-05-22 19:04:41,149: [12438] m.e.detectors:WARNING: Warning TIMESTAMP instruction used
2021-05-22 19:08:25,096: [12438] m.e.detectors:WARNING: Potentially reading uninitialized storage
2021-05-22 19:31:33,771: [12438] m.e.detectors:WARNING: Potentially reading uninitialized storage
2021-05-22 19:31:44,596: [12438] m.e.manticore:INFO: Found a concrete globalsha3 b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x19' -> 67072331549493647622825787457569556318728415786901242217649037894484240406165
```

Solhint Linter

```
incubate.sol:1:1: Error: Compiler version 0.6.12
does not satisfy the r semver requirement
```

```
incubate.sol:736:94: Error: Code contains empty
blocks
```

```
incubate.sol:862:1: Error: Contract has 20 states
declarations but allowed no more than 15
```

```
incubate.sol:874:20: Error: Variable name must be in
mixedCase
```

```
incubate.sol:875:20: Error: Variable name must be in
mixedCase
```

```
incubate.sol:876:21: Error: Variable name must be in
mixedCase
```

```
incubate.sol:939:34: Error: Avoid to make time-based
decisions in your business logic
```

```
incubate.sol:973:39: Error: Avoid to make time-based
decisions in your business logic
```

```
incubate.sol:989:39: Error: Avoid to make time-based
decisions in your business logic
```

```
incubate.sol:999:39: Error: Avoid to make time-based
decisions in your business logic
```

```
incubate.sol:1018:39: Error: Avoid to make time-
based decisions in your business logic
```

```
incubate.sol:1034:39: Error: Avoid to make time-
based decisions in your business logic
```

```
incubate.sol:1056:43: Error: Avoid to make time-
based decisions in your business logic
```

Solidity Static Analysis

Security

Check-effects-interaction:

Potential violation of Checks-Effects-Interaction pattern in Address.functionCallWithValue(address,bytes,uint256,string): Could potentially lead to re-entrancy vulnerability. Note: Modifiers are currently not considered by this static analysis.

more

Pos: 300:4:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.

more

Pos: 219:8:

Inline assembly:

The Contract uses inline assembly, this is only advised in rare cases. Additionally static analysis modules do not parse inline Assembly, this can lead to wrong analysis results.

more

Pos: 342:16:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 939:33:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.

more

Pos: 973:38:

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.
more
Pos: 243:27:

Low level calls:

Use of "call": should be avoided whenever possible. It can lead to unexpected behavior if return value is not handled properly. Please use Direct Calls via specifying the called contract's interface.
more
Pos: 305:50:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 999:38:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 989:38:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 1018:38:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 1034:38:

Block timestamp:

Use of "block.timestamp": "block.timestamp" can be influenced by miners to a certain degree. That means that a miner can "choose" the block.timestamp, to a certain degree, to change the outcome of a transaction in the mined block.
more
Pos: 1056:42:

Disclaimer

Quillhash audit is not a security warranty, investment advice, or endorsement of the Incubate platform. This audit does not provide a security or correctness guarantee of the audited smart contracts. The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them. Securing smart contracts is a multistep process. One audit cannot be considered enough. We recommend that the Incubate Team put in place a bug bounty program to encourage further analysis of the smart contract by other third parties.

Closing Summary

Overall, smart contracts are very well written and adhere to guidelines.

No instances of Re-entrancy or Back-Door Entry were found in the contract.

During the process of the final audit, two informational issues were still found open. It is recommended to kindly go through the above-mentioned details and fix the code accordingly.

incub8

QuillAudits

Canada, India, Singapore and United Kingdom

audits.quillhash.com

audits@quillhash.com