SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

MDB

Audit

Security Assessment
28. April, 2022

For

MILLION DOLLAR BABY

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	. April 2022	• Layout project • Automated- /Manual-Security Testing • Summary

Network
Binance Smart Chain (BEP20)

Website
https://www.mdb.fund/

Telegram
https://t.me/mdbtoken

Twitter
https://twitter.com/MDB_DeFi

Discord
https://discord.com/invite/milliondollarbaby

Description

TBA

Project Engagement

During the 26th of April 2022, **MDB 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

MILLION DOLLAR BABY

Contract Link

v1.0

- Github
  - https://github.com/markynap/MDB
  - Commit: 63034f90993b6a4b65aa6cb1fe71707dcb3f8ced

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.

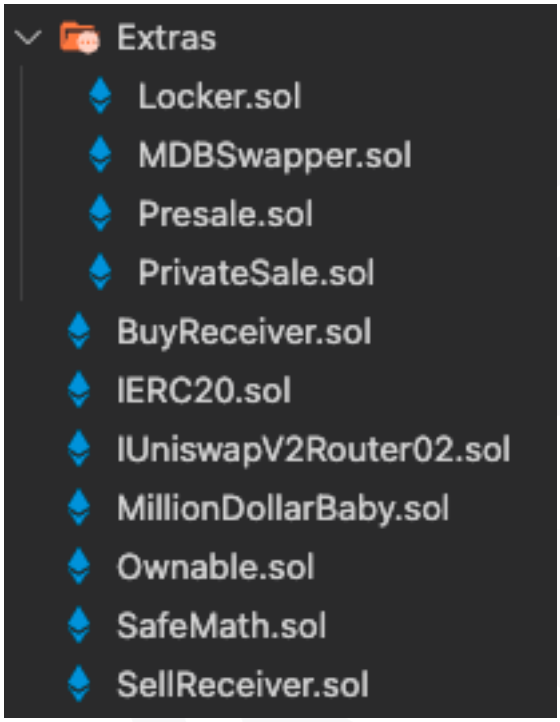
2. Testing and automated analysis that includes the following:
    i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    ii) Symbolic execution, which is analysing a program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



Tested Contract Files

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

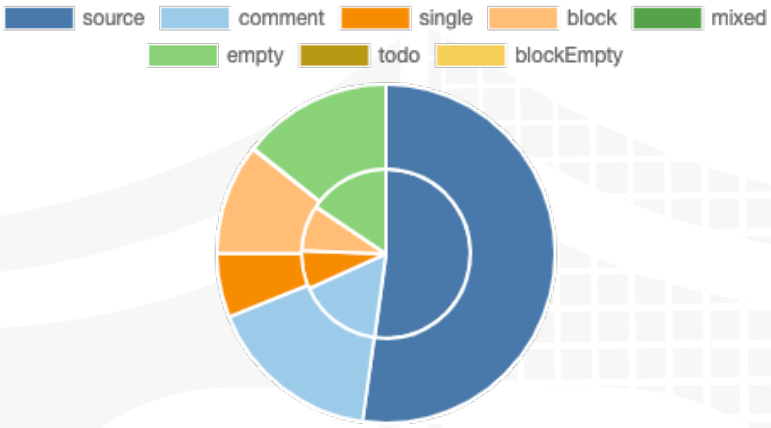
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

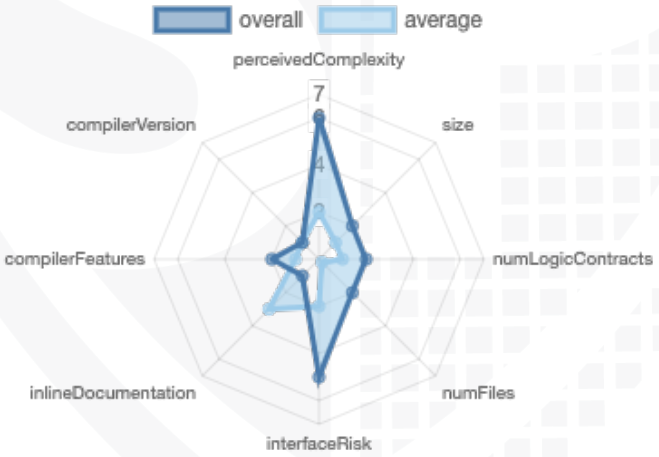
File Name	SHA-1 Hash
contracts/SellReceiver.sol	93cc154f3b62a13932185b0fc208a979333ceaed
contracts/Extras/MDBSwapper.sol	8265fd0f280c18e423d2ef82249ce75919ae931f
contracts/Extras/Locker.sol	ea3eeb4f0a39ba148075004feab36ee196ef8524
contracts/Extras/PrivateSale.sol	b84a90eb8f1419cdd464941c8f18af547ef06251
contracts/Extras/Presale.sol	a9bb3cd813e4359d1e350b3b2405a5825cb40db2
contracts/SafeMath.sol	693e892c17dcef0f372b65b75c37ebbf09a6fdb2
contracts/Ownable.sol	581230b27b5e200cae1b3a751fd677a4a416cd85
contracts/IUniswapV2Router02.sol	4c60f74df4d78102163d39d2ec0523791cf82417
contracts/MillionDollarBaby.sol	f9961a6972a7d25759fdadfabcdfb567070e4c82
contracts/BuyReceiver.sol	b509f96dc72ac0d2a4331587e3a26f7dd38e431a
contracts/IERC20.sol	306e7bb39d2abc537d0776702258ed1b5fc523e8

Metrics

Source Lines
v1.0



Risk Level
v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	8	1	7	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
1.0	131	15

Version	External	Internal	Private	Pure	View
1.0	118	100	0	18	38

State Variables

Version	Total	Public
1.0	59	33

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`0.8.4` `0.8.7`		`yes`		

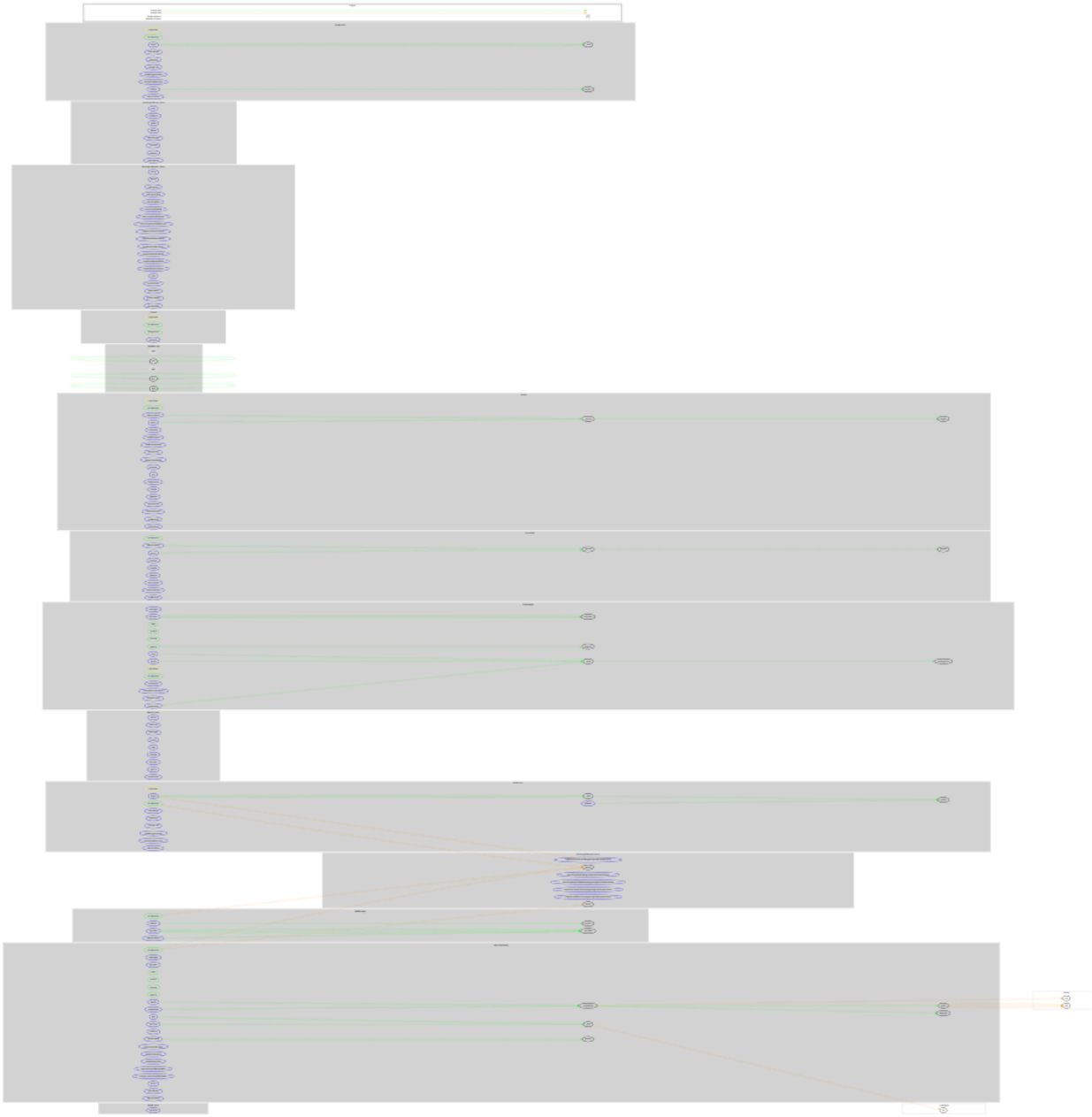
Version	Transfers ETH	Low-Level Calls	DelegateCall	Uses Hash Functions	ECRecover	New/ Create/ Create2

1.0	yes					

Inheritance Graph

v1.0

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. 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)

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

MillionDollarBaby

```
approve
transfer
transferFrom
burn
burnFrom
withdraw
withdrawBNB
setTransferFeeRecipient
setBuyFeeRecipient
setSellFeeRecipient
registerAutomatedMarketMaker
unRegisterAutomatedMarketMaker
setFees
setFeeExempt
changeOwner
```

SellReceiver/BuyReceiver

```
trigger
setTrustFund
setMultisig
setApproved
setMinTriggerAmount
setTrustFundPercentage
withdraw
```

Locker

```
transfer
transferFrom
<Constructor>
setCanClaim
reduceReleasedPerBlock
changeRecipient
claim
```

MDBSwapper

```
withdraw
buyToken 💰
```

Presale

```
donate 💰
setHardCap
setMinDonation
setMaxContribution
whitelistUsers
increaseContribution
terminate
start
changeCreator
```

PrivateSale

```
donate 💰
terminate
```

Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	–	–	–
Max / Total Supply			1000000000

Deployer cannot burn or lock user funds

Name	Exist	Tested	Status
Deployer cannot lock	✓	✓	🚩
Deployer cannot burn	✓	✓	✓

Comments:

v1.0

- Tokens
  - can be burned by msg.sender

- Claim function in locker will transfer tokens of “token” to “recipient” which will be set by the deployer in the constructor while deploying. Nobody can call claim function because nobody is set as claimer, also the recipient is not a claimer.

Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	–	–	–

Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

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

Modifiers and public functions
v1.0

Locker

```
transferFrom
<Constructor>
setCanClaim
  onlyClaimer
reduceReleasedPerBlock
changeRecipient
claim
```

PrivateSale

```
donate
terminate
```

Presale

```
trigger
setTrustFund
  onlyOwner
setMultisig
  onlyOwner
setApproved
  onlyOwner
setMinTriggerAmount
  onlyOwner
setTrustFundPercentage
  onlyOwner
withdraw
  onlyOwner
```

Presale

```
donate
setHardCap
  onlyCreator
setMinDonation
  onlyCreator
setMaxContribution
  onlyCreator
whitelistUsers
  onlyCreator
increaseContribution
  onlyCreator
terminate
  onlyCreator
start
  onlyCreator
changeCreator
  onlyCreator
```

MDBSwapper

```
withdraw
buyToken
```

Comments

- *Deployer can set following state variables without any limitations*
  - SellReceiver / BuyReceiver
    - trustFundPercentage
    - minimumTokensRequiredToTrigger
  - PrivateSale
    - isTerminated
  - Presale
    - max_contribution
    - min_donation
    - hardCap
- *Deployer can enable/disable following state variables*
  - SellReceiver / BuyReceiver
    - approved
  - MillionDollarBaby
    - permissions[account].isLiquidityPool
    - permissions[account].isFeeExempt
  - Presale
    - _whitelisted
- *Deployer can set following addresses*
  - SellReceiver / BuyReceiver
    - multisig
    - trustFund
  - MillionDollarBaby
    - sellFeeRecipient
    - buyFeeRecipient
    - transferFeeRecipient
  - Presale
    - creator
- *Existing Modifiers*
  - onlyClaimer
  - onlyOwner
  - onlyCreator

- There are several authorities which are authorized to call some functions, that means, if the owner is renounced, another address is still authorized to call functions
    - Be aware of this

- SellReceiver / BuyReceiver
    - Owner can withdraw specific token with withdraw function
    - Owner can withdraw complete address balance
- MillionDollarBaby
    - Don't forget to set permissions[recipient].isFeeExempt back to false after changing
        - sellFeeRecipient
        - buyFeeRecipient
        - transferFeeRecipient
    - Owner can withdraw complete address balance
    - Owner an withdraw tokens from contract
        - We recommend you to prevent to pass contract address as token address to the withdraw function

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/SellReceiver.sol	1	1	123	120	77	20	91	💰📩🔅
📝🔍	contracts/Extras/MDBSwapper.sol	1	1	68	64	48	4	64	💰📩
📝	contracts/Extras/Locker.sol	1	———	131	131	87	20	76	📩
📝	contracts/Extras/PrivateSale.sol	1	———	95	95	62	9	42	💰
📝	contracts/Extras/Presale.sol	1	———	158	158	109	13	79	💰
📚	contracts/SafeMath.sol	1	———	145	145	39	93	10	🔅
📝	contracts/Ownable.sol	1	———	51	51	20	24	11	———
🔍	contracts/IUniswapV2Router02.sol	———	3	155	11	7	1	81	💰
📝	contracts/MillionDollarBaby.sol	1	———	252	252	187	21	161	💰📩
📝🔍	contracts/BuyReceiver.sol	1	1	112	109	67	16	66	💰📩🔅
🔍	contracts/IERC20.sol	———	1	80	20	17	54	19	———
📝📚🔍	**Totals**	**9**	**7**	**1370**	**1156**	**720**	**275**	**700**	💰📩🔅

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

High issues

No high issues

Medium issues

No medium issues

Low issues

Issue	File	Type	Line	Description
#1	Main	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
#2	MDBSwapper	Missing Zero Address Validation (missing-zero-check)	20	Check that the address is not zero
#3	Ownable	Missing Zero Address Validation (missing-zero-check)	39	Check that the address is not zero
#4	Presale	Missing Zero Address Validation (missing-zero-check)	91, 46	Check that the address is not zero
#5	PrivateSale	Missing Zero Address Validation (missing-zero-check)	30	Check that the address is not zero

#6	Locker	Missing Zero Address Validation (missing-zero-check)	95, 70, 69	Check that the address is not zero
#7	Locker	State variable visibility is not set	40	It is best practice to set the visibility of state variables explicitly
#8	MDBSwapper	State variable visibility is not set	15, 18	It is best practice to set the visibility of state variables explicitly
#9	Presale	State variable visibility is not set	37	It is best practice to set the visibility of state variables explicitly
#10	PrivateSale	State variable visibility is not set	25	It is best practice to set the visibility of state variables explicitly
#11	SellReceiver	State variable visibility is not set	24	It is best practice to set the visibility of state variables explicitly
#12	MDBSwapper	Local variables shadowing	27	Rename the local variables that shadow another component From “address token” to “address token_”
#13	BuyReicer/ SellReceiver	Missing Events Arithmetic	setTrustFundPercentage function setMinTriggerAmount function	Emit an event for critical parameter changes - trustFundPercentage - minimumTokensRequiredToTrigger
#14	Presale	Missing Events Arithmetic	60, 68, 64, 92	Emit an event for critical parameter changes
#15	Locker	Missing Events Arithmetic	92	Emit an event for critical parameter changes
#16	Locker	No one is able to claim	See description	Nobody is able to call claim function because nobody is set to canClaim as true
#17	MDBSwapper	Undeclared	15	IUniswapV2Router02 is not imported
#18	Locker	Misuse of Boolean constant	32, 36	Function will always return true. Verify and simplify the condition

Informational issues

Issue	File	Type	Line	Description
#1	PrivateSale	State variables that could be declared constant (constable-states)	22	Add the `constant` attributes to state variables that never change
#2	Locker	Wrong import path	4	Please fix the import path of IERC20 file
#3	Locker	Wrong event emitting	26	There is no Approve event existing, change it to Approval
#4	Locker	Functions cannot be pure	See description	Following functions cannot be declared as "pure" unless the state variables are set as constants: - name() L16 - symbol() L19
#5	Locker	Layout ordering	See description	Actually the layout is not ordered like it should. For more information read the documentation at https://docs.soliditylang.org/en/v0.8.13/style-guide.html

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.

28. April 2022:

- DYOR for the TokenDripper contract
    - It used ordinary functions which are not the same as you know
- Read whole report 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	NOT 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**

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY