SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY

Audit

Security Assessment
29. November, 2021

For

HODLME

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

Network
Binance Smart Chain (BEP20)

Website
https://hodlme.cash/

Telegram
https://t.me/hodlME_Official

Twitter
https://twitter.com/hodlme_Official

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

Description

TBA

Project Engagement

During the 24th of November 2021, **Hodlme 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://bscscan.com/address/0xda5c483be26be7236d4e1b88a17af1f4e6b8ef3a#code

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:

- interface
  - DividendPayingTokenInterface.sol
  - DividendPayingTokenOptionalInterface.sol
  - IERC20.sol
  - IERC20Metadata.sol
  - IPancakeFactory.sol
  - IPancakeRouter02.sol
- standard
  - ERC20.sol
  - Ownable.sol
  - SafeMath.sol
  - SafeMathInt.sol
  - SafeMathUint.sol
- Context.sol
- DividendPayingToken.sol
- HodlMe.sol
- HodlMeDividendTracker.sol
- HodlMeGame.sol
- IPancakeRouter01.sol
- IterableMapping.sol

Tested Contract Files

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

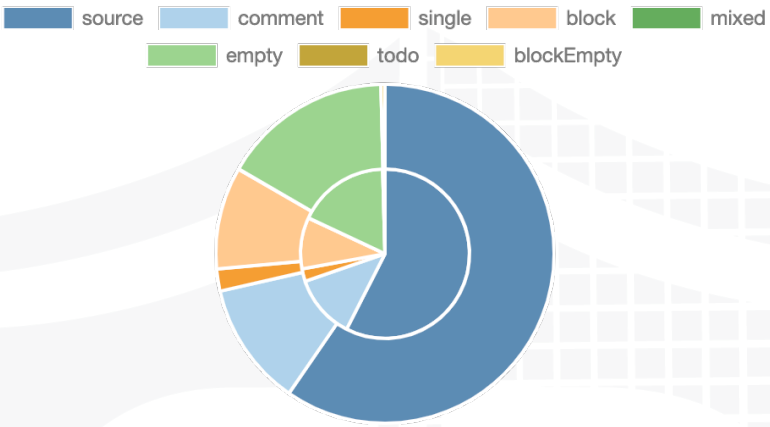
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

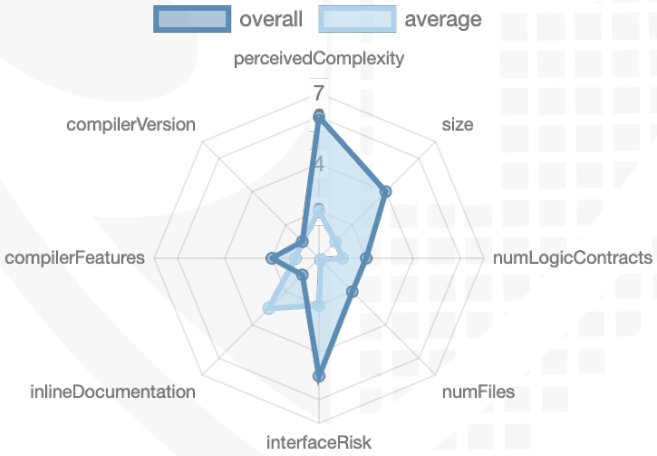
File Name	SHA-1 Hash
contracts/DividendPayingToken.sol	00086869aef0ebbb6165e89b5a6d4aa60771ba8b
contracts/Context.sol	5c7e18f8baa8f9d04e9a511433ab37ffb92a0a86
contracts/HodlMeDividendTracker.sol	77a6c2e92698a021105c075da6b30c30e9b1b0a4
contracts/hardhat/console.sol	b1e9d9fe3a5c1ce12f551fee5038b5ef3c499292
contracts/HodlMeGame.sol	fe8b926a994eaa2c473232cd2a4320b52946dcab
contracts/standard/SafeMathUint.sol	8841ac3e4d09303fa9309cf93f1ff738c84f1438
contracts/standard/SafeMathInt.sol	0af8ea2dabb6f95c5b27a3879fa4b83681ddac9b
contracts/standard/SafeMath.sol	7254371e32812f15a3101a69dc21f6c08c2de334
contracts/standard/Ownable.sol	623400fa363594c78727986465b735b8ddc3271f
contracts/standard/ERC20.sol	34c652915050345b07dea1e37a6b91d640dad0aa
contracts/IPancakeRouter01.sol	65211c75a915d8a44d55027d5e018e61f79d7e5b
contracts/HodlMe.sol	8594657479d4f3c0d518f4a4a0f353f2424c6a56
contracts/IterableMapping.sol	3f7629c62e8c40bf23eb1f66b854ecad694a4ad4
contracts/interface/DividendPayingTokenInterface.sol	7253a6f5738a904d2621d7293b02d0d942d84f35
contracts/interface/IERC20Metadata.sol	d5719172b635bbb3a8dad520f2ddbd768cccef8d
contracts/interface/IPancakeRouter02.sol	8ba6f39334175153abaf4db80726c2c26018f53f
contracts/interface/IPancakeFactory.sol	c64813d09a51191817a2269292825d88145d9106
contracts/interface/DividendPayingTokenOptionalInterface.sol	b2c1c5b448b0e7ff70c617915893e0de57fc14c6
contracts/interface/IERC20.sol	e854cccb4dcc1107b69032fae553c8ddd41d1625

Metrics

Source Lines
v1.0



Risk Level
v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	6	5	7	1

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	123	8

Version	External	Internal	Private	Pure	View
1.0	76	494	4	26	431

State Variables

Version	Total	Public
1.0	57	33

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	`^0.8.0` `>=0.4.22` `<0.9.0`		`yes`	`yes` (1 asm blocks)	

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

Scope of Work

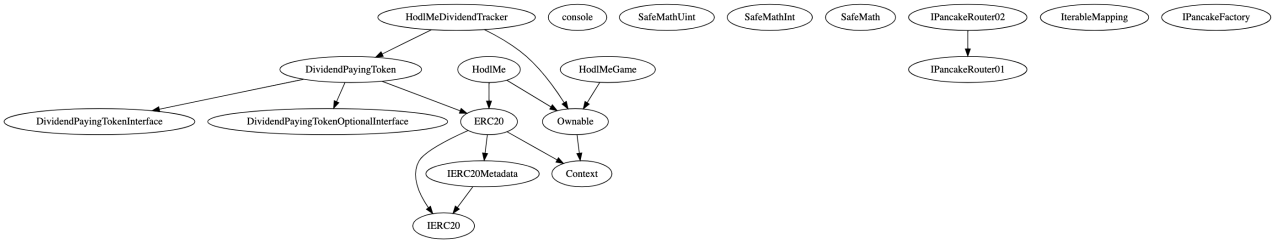
The above token Team provided us with the files that needs to be tested (Github, Bscscan, Etherscan, files, etc.). The scope of the audit is the main contract (usual the same name as team appended with .sol).

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

1. approve
2. claim
3. decreaseAllowance
4. excludeFromAllLimits
5. excludeFromDividends
6. excludeFromFees
7. excludeMultipleAccountsFromFees
8. increaseAllowance
9. processDividendTracker
10. renounceOwnership
11. setBuyFees
12. setCanTransferBeforeTradingIsEnabled
13. setGameIsEnabled
14. setHodlMeGameAddress
15. setMaxHoldingAmount
16. setSellFees
17. setSwapIsEnabled
18. setSwapTokensAtAmount
19. setTradingIsEnabled
20. setWallets
21. transfer
22. transferFrom
23. transferOwnership
24. updateClaimWait
25. updateDividendTracker
26. updateGasForProcessing
27. updateMinimumTokenBalanceForDividends
28. updatePancakeRouter

Deployer cannot mint any new tokens

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

Max / Total Supply: 100.000.000.000

Deployer cannot burn or lock user funds

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

Comments:

v1.0

- Deployer can lock user funds
  - If tradingEnabled is false
  - If maxHoldingAmount is set to 0
    - It can be set without any limitations

Deployer cannot pause the contract

Name	Exist	Tested	Verified
Deployer cannot pause	-	-	-

Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

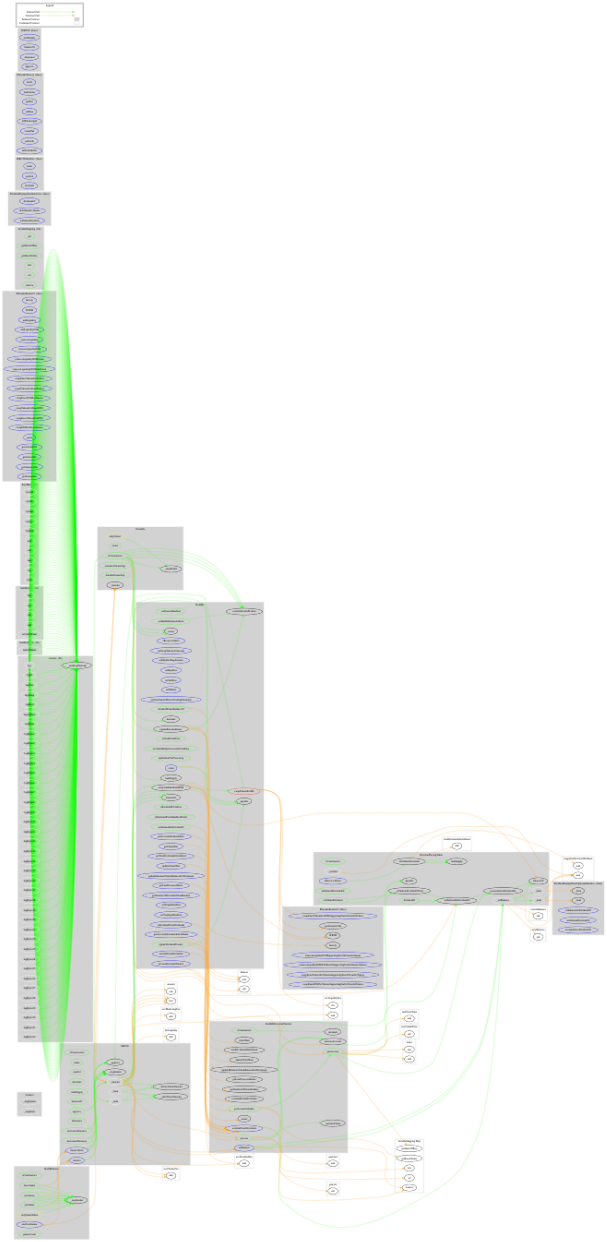
Attribute	Symbol
Verfified / Checked	✓
Partly Verified	🚩
Unverified / Not checked	✗
Not available	–

OnlyOwner functions

```
setGameIsEnabled
setHodlMeGameAddress
updatePancakeRouter
excludeFromAllLimits
setSwapTokensAtAmount
setMaxHoldingAmount
setBuyFees
setSellFees
setWallets
setCanTransferBeforeTradingIsEnabled
excludeFromDividends
updateDividendTracker
excludeFromFees
excludeMultipleAccountsFromFees
updateGasForProcessing
updateClaimWait
updateMinimumTokenBalanceForDividends
```

```
setSwapIsEnabled
setTradingIsEnabled
```

CallGraph



Source Units in Scope
v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
📝	contracts/DividendPayingToken.sol	1	—	229	207	113	65	79	💰
🎨	contracts/Context.sol	1	—	13	13	9	1	1	—
📝	contracts/HodlMeDividendTracker.sol	1	—	266	219	162	2	94	—
📚	contracts/hardhat/console.sol	1	—	1532	1532	1149	1	778	💻
📝	contracts/HodlMeGame.sol	1	—	173	173	152	1	64	🕹️
📚	contracts/standard/SafeMathUint.sol	1	—	11	11	8	1	3	—
📚	contracts/standard/SafeMathInt.sol	1	—	40	40	26	5	8	—
📚	contracts/standard/SafeMath.sol	1	—	236	204	69	121	10	🌟
📝	contracts/standard/Ownable.sol	1	—	64	64	34	21	24	—
📝	contracts/standard/ERC20.sol	1	—	373	323	116	168	80	—
🔍	contracts/IPancakeRouter01.sol	—	1	162	6	3	1	48	💰
📝	contracts/HodlMe.sol	1	—	592	534	358	85	331	💰🌀
📚	contracts/IterableMapping.sol	1	—	74	62	48	2	19	—
🔍	contracts/interface/DividendPayingTokenInterface.sol	—	1	14	6	3	1	10	💰
🔍	contracts/interface/IERC20Metadata.sol	—	1	22	11	4	10	9	🌟
🔍	contracts/interface/IPancakeRouter02.sol	—	1	52	8	4	1	16	💰
🔍	contracts/interface/IPancakeFactory.sol	—	1	33	13	9	1	17	—
🔍	contracts/interface/DividendPayingTokenOptionalInterface.sol	—	1	20	6	3	1	7	—
🔍	contracts/interface/IERC20.sol	—	1	87	27	21	54	13	🌟
📝📚🔍🎨	**Totals**	**12**	**7**	**3993**	**3459**	**2291**	**542**	**1611**	💻💰🕹️🌀🌟

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	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	HodlMe Game	A floating pragma is set	3	The current pragma Solidity directive is „"^0.8.0"".
#3	Hodlme	Missing Zero Address Validation (missing-zero-check)	149	Check that the address is not zero
#4	HodlMe Game	Missing Zero Address Validation (missing-zero-check)	66	Check that the address is not zero
#5	HodlMe Game	State variable visibility is not set	34, 35	It is best practice to set the visibility of state variables explicitly

Informational issues

Issue	File	Type	Line	Description

#1	DividendPayingToken	Error message is missing	72	Add error message in require statement
#2	HoldMeDividendTracker	Error message is missing	57	Add error message in require statement
#3	SafeMathInt	Error message is missing	9, 19, 25, 32, 37	Add error message in require statement
#4	SafeMathUInt	Error message is missing	8	Add error message in require statement

Audit Comments

29. November 2021:

- Deployer can lock user funds

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	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	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

Solid Proofed

Blockchain Security | Smart Contract Audits | KYC

MADE IN GERMANY