



**SOLIDProof**  
*Bring trust into your projects*

**Blockchain Security | Smart Contract Audits | KYC**

MADE IN GERMANY

# Ubuntu Tribe Audit

**Security Assessment**  
**26. February, 2022**

**For**



# Ubuntu Tribe

Disclaimer	3
Description	5
Project Engagement	5
Logo	5
Contract Link	5
Methodology	7
Used Code from other Frameworks/Smart Contracts (direct imports)	8
Tested Contract Files	9
Source Lines	10
Risk Level	10
Capabilities	11
Inheritance Graph	12
CallGraph	13
Scope of Work/Verify Claims	14
Modifiers and public functions	20
Source Units in Scope	22
Critical issues	23
High issues	23
Medium issues	23
Low issues	23
Informational issues	23
Commented Code exist	24
Audit Comments	24
SWC Attacks	26

## 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	25. February 2022	<ul style="list-style-type: none"><li>• Layout project</li><li>• Automated- /Manual-Security Testing</li><li>• Summary</li></ul>
1.1	26. February 2022	Reaudit

## **Network**

Ethereum (ERC20)

## **Website**

<https://www.utribe.one/>

## **Twitter**

<https://twitter.com/utribeone>

## **Facebook**

<https://www.utribe.one/>

## **Instagram**

<https://instagram.com/ubuntu.coin>

## **LinkedIn**

<https://linkedin.com/company/utribeone>

## **Youtube**

[https://www.youtube.com/channel/UCZp1v9UrM\\_wgDirC\\_9bjLbA](https://www.youtube.com/channel/UCZp1v9UrM_wgDirC_9bjLbA)

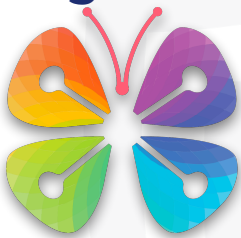
## Description

With Ubuntu Tribe, people can reclaim control over their wealth and multiply it, contributing to the wellbeing of all

## Project Engagement

During the 21st of February 2022, **Ubuntu Tribe 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



# Ubuntu Tribe

## Contract Link

### v1.0

- Gitlab
  - <https://gitlab.com/fluidefi1/utribе/-/tree/gift-erc20-token/contracts/gift-erc20-token>
  - Commit: 1ca3325e065a3230f3697d4f1af38ee496f259f9

### v1.1

- Gitlab
  - <https://gitlab.com/fluidefi1/utribе/-/tree/gift-erc20-token/contracts/gift-erc20-token>
  - Commit: b7560815c7c8915674a1637caded5c2266c4835f

# 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)
<b>Critical</b>	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.
<b>High</b>	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 as possible.
<b>Medium</b>	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.
<b>Low</b>	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.
<b>Informational</b>	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:

```
Arrays.sol  
Context.sol  
Counters.sol  
ERC20.sol  
ERC20Pausable.sol  
ERC20Snapshot.sol  
IERC20.sol  
IERC20Metadata.sol  
Math.sol  
Ownable.sol  
Pausable.sol  
SafeMath.sol
```



## Tested Contract Files

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

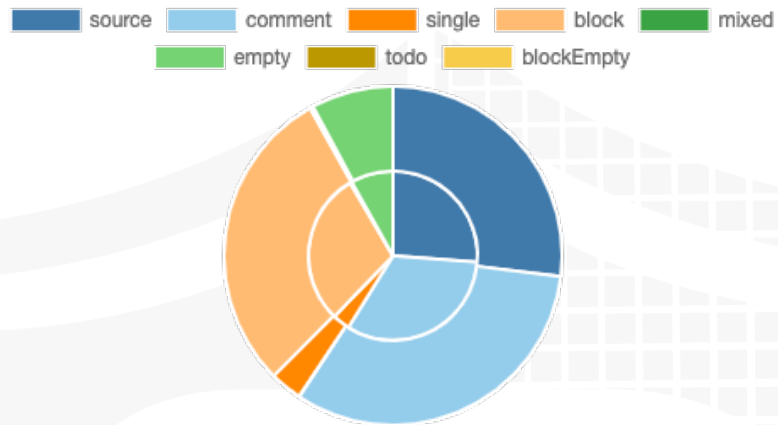
*A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.*

### v1.0

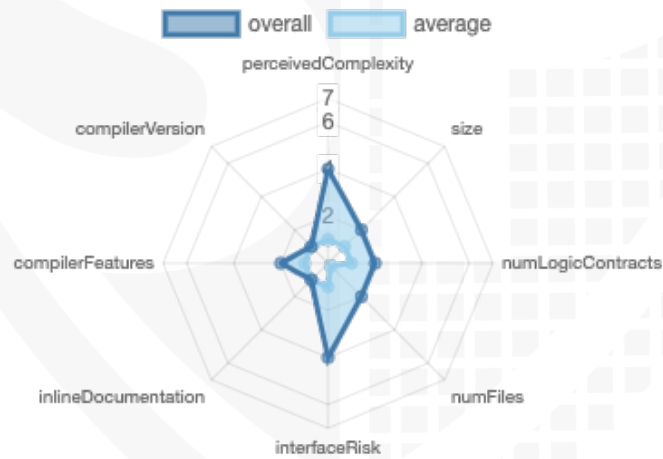
File Name	SHA-1 Hash
contracts/ERC20Snapshot.sol	695371769e0213ae48c1eb35ca5ed34e6d689092
contracts/Math.sol	883142f8542c55a0ae5ded5ab19a6c0fea91472c
contracts/Context.sol	2da7a4b124d7080a2a0182aecee63aa9bd0d1fb6
contracts/IERC20Metadata.sol	67cec1b0ea0da837602e1a674f6cb8a5d689bee5
contracts/GIFT.sol	c36a1212ba69130cfabb21eed266585cddcb041e
contracts/ERC20Pausable.sol	c5a5ad7bdfcd76517cd3fa0b3b9060b342eb0ea1
contracts/Arrays.sol	7f4d5417e8eeb2e323b4b18d0b85a65c7f08fab4
contracts/SafeMath.sol	d6d2bea2b925e4f6ac46bc16dde479979ce1f773
contracts/Ownable.sol	322915f34f844670c2b4065df9988374468a2c29
contracts/Counters.sol	cc91ca5dd4105db3ae0641855a31fd934e9c9f4a
contracts/Pausable.sol	f40f561c0eb026c588a0c4fcb8cbe2437a6be295
contracts/ERC20.sol	9e1f17c88615d137418409653b0c638ee6d4bc91
contracts/IERC20.sol	c6244bea30e3053e1a4bcdb40198dc5b3e15cd29

# Metrics

## Source Lines v1.0



## Risk Level v1.0



## Capabilities

### Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	2	4	2	5

### 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	40	0

Version	External	Internal	Private	Pure	View
1.0	9	86	5	17	24

### State Variables

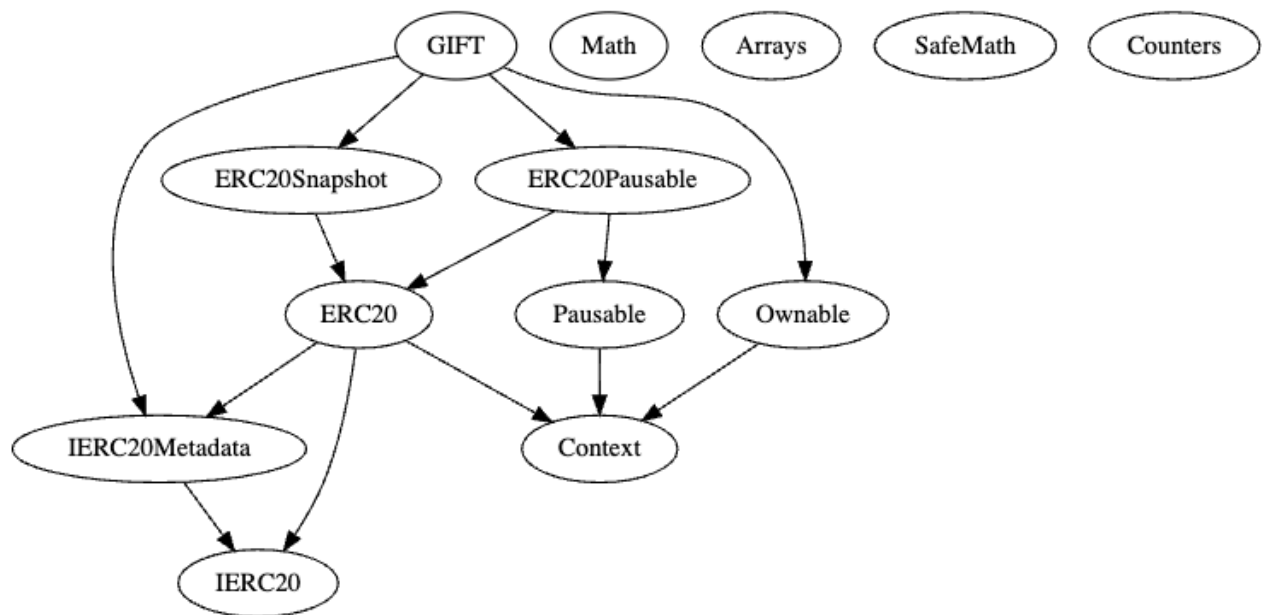
Version	Total	Public
1.0	23	13

### Capabilities

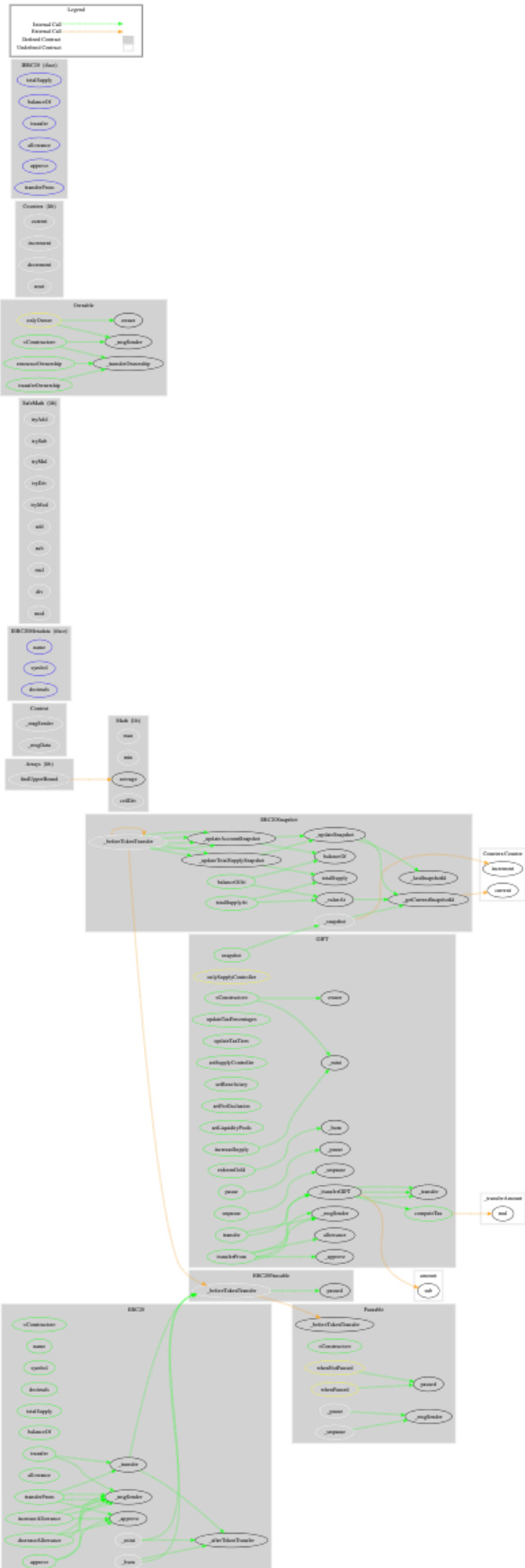
Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	^0.8.4				

# 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

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

snapshot  
updateTaxPercentages  
updateTaxTiers  
setSupplyController  
setBeneficiary  
setFeeExclusion  
setLiquidityPools  
increaseSupply  
redeemGold  
pause  
unpause  
transfer  
transferFrom

renounceOwnership  
transferOwnership

transfer  
approve  
transferFrom  
increaseAllowance  
decreaseAllowance

## Deployer cannot mint any new tokens

Name	Exist	Tested	Status
Deployer cannot mint	✓	✓	✗
Max / Total Supply	500.000.000		

Comments:

### v1.0

- Only supply controller can mint tokens itself with `increaseSupply` function L160 GIFT.sol



## Deployer cannot burn or lock user funds

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

Comments:

### v1.0

- Only supply controller can burn tokens for a certain address without permission with redeemGold function L168 GIFT.sol
- Deployer can lock user funds by pausing the contract

## Deployer cannot pause the contract

Name	Exist	Tested	Status
Deployer cannot pause	✓	✓	✗

Comments:

**v1.0**

- Only owner can enable/disable pause



## Overall checkup (Smart Contract Security)














Tested	Verified
✓	✓



### Legend






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

# Modifiers and public functions

v1.0

- ✓  snapshot
  - Ⓜ onlyOwner
- ✓  updateTaxPercentages
  - Ⓜ onlyOwner
- ✓  updateTaxTiers
  - Ⓜ onlyOwner
- ✓  setSupplyController
  - Ⓜ onlyOwner
- ✓  setBeneficiary
  - Ⓜ onlyOwner
- ✓  setFeeExclusion
  - Ⓜ onlyOwner
- ✓  setLiquidityPools
  - Ⓜ onlyOwner
- ✓  increaseSupply
  - Ⓜ onlySupplyController
- ✓  redeemGold
  - Ⓜ onlySupplyController
- ✓  pause
  - Ⓜ onlyOwner
- ✓  unpause
  - Ⓜ onlyOwner
- ✓  transfer
  - Ⓜ whenNotPaused
- ✓  transferFrom
  - Ⓜ whenNotPaused

- ✓  renounceOwnership
  - Ⓜ onlyOwner
- ✓  transferOwnership
  - Ⓜ onlyOwner

-  transfer
-  approve
-  transferFrom
-  increaseAllowance
-  decreaseAllowance
























## Comments

- Deployer can set following state variables without any limitations
  - tierOneTaxPercentage
  - tierTwoTaxPercentage
  - tierThreeTaxPercentage
  - tierFourTaxPercentage
  - tierFiveTaxPercentage
  - tierOneMax
  - tierTwoMax
  - tierThreeMax
  - tierFourMax
- Deployer can enable/disable following state variables
  - \_isExcludedFromFees
  - \_isLiquidityPool[\_liquidityPool]
  - \_paused
- Deployer can set following addresses
  - supplyController
  - beneficiary

**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/ERC20Snapshot.sol	1	————	195	191	76	89	46	————
	contracts/Math.sol	1	————	43	43	15	23	4	————
	contracts/Context.sol	1	————	24	24	9	12	1	————
	contracts/IERC20Metadata.sol	————	1	28	17	4	16	9	
	contracts/GIFT.sol	1	————	254	230	132	68	97	
	contracts/ERC20Pausable.sol	1	————	33	29	9	16	8	
	contracts/Arrays.sol	1	————	48	48	24	17	6	
	contracts/SafeMath.sol	1	————	227	215	69	131	10	
	contracts/Ownable.sol	1	————	76	76	28	38	23	————
	contracts/Counters.sol	1	————	43	43	24	14	2	
	contracts/Pausable.sol	1	————	91	91	29	51	16	————
	contracts/ERC20.sol	1	————	356	336	103	194	80	
	contracts/IERC20.sol	————	1	82	27	17	58	13	
	<b>Totals</b>	<b>11</b>	<b>2</b>	<b>1500</b>	<b>1370</b>	<b>539</b>	<b>727</b>	<b>315</b>	

## 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

## Informational issues

**No informational issues**

## Commented Code exist

There are some instances of code being commented out in the following files that should be removed:

File	Line	Comment
Math	29	// (a + b) / 2 can overflow.
	40	// (a + b - 1) / b can overflow on addition, so we distribute.

## Recommendation

Remove the commented code, or address them properly.

## 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.

### 25. February 2022:

- OpenZeppelin Contracts Version used 4.4.1. This branch is 9 commits ahead, 121 commits behind master.
- Read whole report for more information



## Unit Testing

Contract: GIFT

- ✓ checking if totalSupply returns total token supply
- ✓ updateTaxPercentages: checking that non owner cannot call onlyOwner modified function (666ms)
- ✓ updateTaxPercentages: checking that tax percentages get updated (92ms)
- ✓ updateTaxTiers: checking that non owner cannot call onlyOwner modified function (77ms)
- ✓ updateTaxTiers: checking that tax tiers get updated (90ms)
- ✓ setSupplyController: checking that non owner cannot call onlyOwner modified function (41ms)
- ✓ setSupplyController: checking that supplyController cannot be set to zero address (101ms)
- ✓ setSupplyController: checking that supplyController state variable gets set to expected address (62ms)
- ✓ setBeneficiary: checking that non owner cannot call onlyOwner modified function (78ms)
- ✓ setBeneficiary: checking that beneficiary cannot be set to zero address
- ✓ setBeneficiary: checking that beneficiary state variable gets set to expected address (102ms)
- ✓ setFeeExclusion: checking that non owner cannot call onlyOwner modified function (45ms)
- ✓ setFeeExclusion: checking if address gets set to be excluded from fee (137ms)
- ✓ setLiquidityPools: checking that non owner cannot call onlyOwner modified function (38ms)
- ✓ setLiquidityPools: checking if address gets set as a liquidity pool (82ms)
- ✓ increaseSupply: checking that non supplyController cannot call onlySupplyController modified function (92ms)
- ✓ increaseSupply: checking that it emits a Transfer event from zero address on successful call
- ✓ redeemGold: checking that non supplyController cannot call onlySupplyController modified function (129ms)
- ✓ redeemGold: checking that it emits a Transfer event to zero address on successful call
- ✓ pause: checking that non owner cannot call onlyOwner modified function (40ms)
- ✓ pause: checking that you cannot call pause function when contract is already paused (173ms)
- ✓ pause: checking that it emits a Paused event on successful call (48ms)
- ✓ unpause: checking that non owner cannot call onlyOwner modified function (57ms)
- ✓ unpause: checking that you cannot call unpause function when contract is already unpaused (213ms)
- ✓ unpause: checking that it emits a Unpaused event on successful call (126ms)
- ✓ snapshot: checking that non owner cannot call onlyOwner modified function (74ms)
- ✓ snapshot: checking that values are recorded when snapshot function is called (508ms)
- ✓ transfer: checking that you cannot call transfer when contract is paused (70ms)
- ✓ transfer: checking that you cannot call transfer to zero address (99ms)
- ✓ transfer: checking that you cannot transfer more than available balance (159ms)
- ✓ transfer: checking that it emits Transfer event on successful call (77ms)
- ✓ transfer: checking that it emits additional Transfer event to beneficiary on successful call (271ms)

## SWC Attacks

ID	Title	Relationships	Status
<a href="#">SW C-1 36</a>	Unencrypted Private Data On-Chain	<a href="#">CWE-767: Access to Critical Private Variable via Public Method</a>	PASSED
<a href="#">SW C-1 35</a>	Code With No Effects	<a href="#">CWE-1164: Irrelevant Code</a>	PASSED
<a href="#">SW C-1 34</a>	Message call with hardcoded gas amount	<a href="#">CWE-655: Improper Initialization</a>	PASSED
<a href="#">SW C-1 33</a>	Hash Collisions With Multiple Variable Length Arguments	<a href="#">CWE-294: Authentication Bypass by Capture-replay</a>	PASSED
<a href="#">SW C-1 32</a>	Unexpected Ether balance	<a href="#">CWE-667: Improper Locking</a>	PASSED
<a href="#">SW C-1 31</a>	Presence of unused variables	<a href="#">CWE-1164: Irrelevant Code</a>	PASSED
<a href="#">SW C-1 30</a>	Right-To-Left-Override control character (U+202E)	<a href="#">CWE-451: User Interface (UI) Misrepresentation of Critical Information</a>	PASSED
<a href="#">SW C-1 29</a>	Typographical Error	<a href="#">CWE-480: Use of Incorrect Operator</a>	PASSED
<a href="#">SW C-1 28</a>	DoS With Block Gas Limit	<a href="#">CWE-400: Uncontrolled Resource Consumption</a>	PASSED

<a href="#">SW</a> <a href="#">C-1</a> <a href="#">27</a>	Arbitrary Jump with Function Type Variable	<a href="#">CWE-695: Use of Low-Level Functionality</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">25</a>	Incorrect Inheritance Order	<a href="#">CWE-696: Incorrect Behavior Order</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">24</a>	Write to Arbitrary Storage Location	<a href="#">CWE-123: Write-what-where Condition</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">23</a>	Requirement Violation	<a href="#">CWE-573: Improper Following of Specification by Caller</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">22</a>	Lack of Proper Signature Verification	<a href="#">CWE-345: Insufficient Verification of Data Authenticity</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">21</a>	Missing Protection against Signature Replay Attacks	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">20</a>	Weak Sources of Randomness from Chain Attributes	<a href="#">CWE-330: Use of Insufficiently Random Values</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-11</a> <a href="#">9</a>	Shadowing State Variables	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-11</a> <a href="#">8</a>	Incorrect Constructor Name	<a href="#">CWE-665: Improper Initialization</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-11</a> <a href="#">7</a>	Signature Malleability	<a href="#">CWE-347: Improper Verification of Cryptographic Signature</a>	<b>PASSED</b>

<a href="#">SW C-11 6</a>	Timestamp Dependence	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	<b>PASSED</b>
<a href="#">SW C-11 5</a>	Authorization through tx.origin	<a href="#">CWE-477: Use of Obsolete Function</a>	<b>PASSED</b>
<a href="#">SW C-11 4</a>	Transaction Order Dependence	<a href="#">CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')</a>	<b>PASSED</b>
<a href="#">SW C-11 3</a>	DoS with Failed Call	<a href="#">CWE-703: Improper Check or Handling of Exceptional Conditions</a>	<b>PASSED</b>
<a href="#">SW C-11 2</a>	Delegatecall to Untrusted Callee	<a href="#">CWE-829: Inclusion of Functionality from Untrusted Control Sphere</a>	<b>PASSED</b>
<a href="#">SW C-11 1</a>	Use of Deprecated Solidity Functions	<a href="#">CWE-477: Use of Obsolete Function</a>	<b>PASSED</b>
<a href="#">SW C-11 0</a>	Assert Violation	<a href="#">CWE-670: Always-Incorrect Control Flow Implementation</a>	<b>PASSED</b>
<a href="#">SW C-1 09</a>	Uninitialized Storage Pointer	<a href="#">CWE-824: Access of Uninitialized Pointer</a>	<b>PASSED</b>
<a href="#">SW C-1 08</a>	State Variable Default Visibility	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	<b>PASSED</b>
<a href="#">SW C-1 07</a>	Reentrancy	<a href="#">CWE-841: Improper Enforcement of Behavioral Workflow</a>	<b>PASSED</b>
<a href="#">SW C-1 06</a>	Unprotected SELFDESTRUCT Instruction	<a href="#">CWE-284: Improper Access Control</a>	<b>PASSED</b>

<a href="#">SW</a> <a href="#">C-1</a> <a href="#">05</a>	Unprotected Ether Withdrawal	<a href="#">CWE-284: Improper Access Control</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">04</a>	Unchecked Call Return Value	<a href="#">CWE-252: Unchecked Return Value</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">03</a>	Floating Pragma	<a href="#">CWE-664: Improper Control of a Resource Through its Lifetime</a>	<b>NOT PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">02</a>	Outdated Compiler Version	<a href="#">CWE-937: Using Components with Known Vulnerabilities</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">01</a>	Integer Overflow and Underflow	<a href="#">CWE-682: Incorrect Calculation</a>	<b>PASSED</b>
<a href="#">SW</a> <a href="#">C-1</a> <a href="#">00</a>	Function Default Visibility	<a href="#">CWE-710: Improper Adherence to Coding Standards</a>	<b>PASSED</b>

The logo features the words "SolidProofed" in a white, elegant script font. The "P" in "Proofed" is significantly larger and more stylized, with a long horizontal stroke that extends to the left. The background is a solid blue color with a faint, large shield emblem. The shield has a blue border and a grid-like pattern in the center, with a darker blue circle in the middle.

SolidProofed

**Blockchain Security | Smart Contract Audits | KYC**

A small horizontal bar representing the German flag, with black, red, and gold stripes.

MADE IN GERMANY