



SOLIDProof
Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC


MADE IN GERMANY

Audit

Security Assessment
08. December, 2021

For



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
Scope of Work	14
Inheritance Graph	14
Verify Claims	15
Modifiers	21
CallGraph	22
Source Units in Scope	23
Critical issues	25
High issues	25
Medium issues	25
Low issues	25
Informational issues	26
Audit Comments	26
SWC Attacks	27

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	08. December 2021	<ul style="list-style-type: none">• Layout project• Automated- /Manual-Security Testing• Summary

Network

Binance Smart Chain (BEP20)

Website

<https://instantxrp.finance/>

Telegram

<https://t.me/joininstant65>

<https://t.me/joininstant>

Twitter

<https://twitter.com/InstantXrp>

Reddit

<https://www.reddit.com/user/INSTANTXRPFINANCE/>

Description

TBA

Project Engagement

During the Date, **Teamname 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/0x0D1d31a3eA2FC33e79F1A2f0E2Da817042f20B58#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 as possible.
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:

BabyToken

```
@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol
@openzeppelin/contracts/token/ERC20/IERC20.sol
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol
@openzeppelin/contracts/math/SafeMath.sol
@openzeppelin/contracts/proxy/Clones.sol
./interfaces/IUniswapV2Factory.sol
./interfaces/IUniswapV2Router02.sol
./interfaces/IBabyToken.sol
./BabyTokenDividendTracker.sol
```

BabyTokenDividendTracker

```
@openzeppelin/contracts/token/ERC20/ERC20.sol
@openzeppelin/contracts-upgradeable/token/ERC20/ERC20Upgradeable.sol
@openzeppelin/contracts/token/ERC20/IERC20.sol
@openzeppelin/contracts/access/Ownable.sol
@openzeppelin/contracts-upgradeable/access/OwnableUpgradeable.sol
@openzeppelin/contracts/math/SafeMath.sol
@openzeppelin/contracts/proxy/Clones.sol
./interfaces/IUniswapV2Factory.sol
./interfaces/IUniswapV2Router02.sol
./interfaces/IUniswapV2Pair.sol
./libs/SafeMathInt.sol
./libs/SafeMathUint.sol
./libs/IterableMapping.sol
./antibot/IPinkAntiBot.sol
./interfaces/IAntiBotBabyToken.sol
```


Tested Contract Files

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

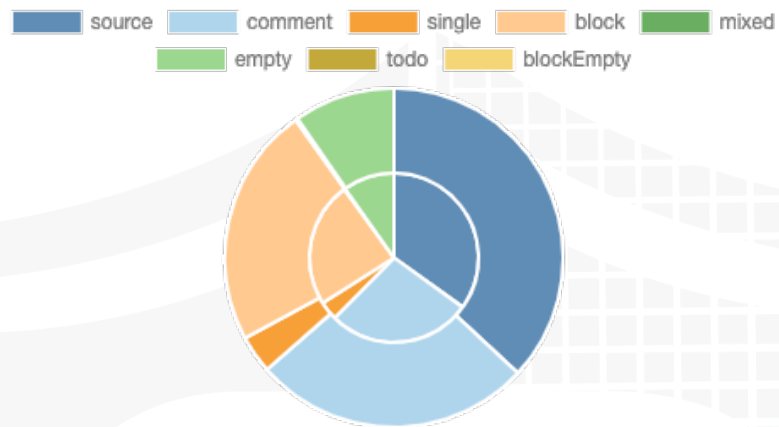
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

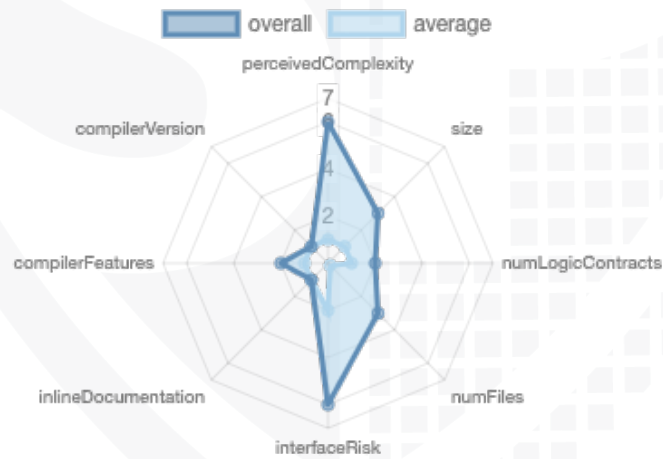
File Name	SHA-1 Hash
contracts/ERC20Upgradeable.sol	6ee6ab632ded06e0ac6d2bd829a90aab95ad32aa
contracts/OwnableUpgradeable.sol	e835454821934f07f42ad705d8b3c7d9b95d2928
contracts/BabyTokenDividendTracker.sol	f05703bd355c936c29bc9fc02662840485b5cbeb
contracts/Clones.sol	2b50bb3d43f7111b4ec12fc68120f65c497795bf
contracts/Ownable.sol	a3e98006adf649f9761b7f384c8468fc7ee52831
contracts/BabyToken.sol	126559e0743781445e3f28030d75b45ddca1b75c
contracts/antibot/IPinkAntiBot.sol	edbc021f096c0dc3d96c8ab5e2cb43c7b5d4f61a
contracts/libs/SafeMathUint.sol	e02a24b9f2774385a8f9054b030a8bc215a89733
contracts/libs/SafeMathUpgradeable.sol	c3fc02887779f0de3ab4c368a5bd8c6196b1926
contracts/libs/SafeMathInt.sol	bffbb5ad74575fb9e4be2a3c884c5ab626ccfa34
contracts/libs/SafeMath.sol	e8d52e78bd679fcdd64bb69f573da6b814259f46
contracts/libs/IterableMapping.sol	27c872599ec7b456030b0ed0c1eac3d1ecbd287
contracts/interfaces/IERC20Upgradeable.sol	b310903de7a32c0cd3631971a9113188ac411347
contracts/interfaces/IBabyToken.sol	743eebef127857688367cb86d52acd30b6233998
contracts/interfaces/IUniswapV2Pair.sol	db023a1046186f34ee91c63dc0f6d9c7715f7cc4
contracts/interfaces/IUniswapV2Factory.sol	9360ecb8c004c1a459a7f1d0e3a57e7d83b25383
contracts/interfaces/IAntiBotBabyToken.sol	b90148b564aac9633f605a21cda1a8b9486f8016
contracts/interfaces/IUniswapV2Router02.sol	a1d006daf0a8b0f8e403fa37d073edf19034d40a
contracts/interfaces/IERC20.sol	647e4efd865a1f1be76579b056382c04bac01406
contracts/ERC20.sol	197ecb9797306e1a47524d003e7723349c5369e2
contracts/utils/AddressUpgradeable.sol	522674d9d63da6c735286f3a85f9bbd4d8c8af02
contracts/utils/Context.sol	a34cc2179b2da819d60afa9d711d0094d5a72799
contracts/utils/ContextUpgradeable.sol	7c8676f62f5224a79b5f8d9cc0639a6914c27c42
contracts/proxy/Initializable.sol	b4ea9be81cf5755124ca4c530b62d35721708bdd

Metrics

Source Lines v1.0



Risk Level v1.0



Capabilities

Components

Version	Contracts	Libraries	Interfaces	Abstract
1.0	5	7	11	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	158	5

Version	External	Internal	Private	Pure	View
1.0	104	194	10	47	74

State Variables

Version	Total	Public
1.0	48	20

Capabilities

Version	Solidity Versions observed	Experimental Features	Can Receive Funds	Uses Assembly	Has Destroyable Contracts
1.0	>=0.6.0 <0.8.0 ^0.7.6 ^0.7.0 >=0.5.0 >=0.6.2 >=0.6.2 <0.8.0 >=0.4.2 4 <0.8.0		yes	yes (5 asm blocks)	

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

1.0	yes					yes → Assembly Call: Name: create → Assembly Call: Name: create2
-----	-----	--	--	--	--	------------------------------------------------------------------------------------------



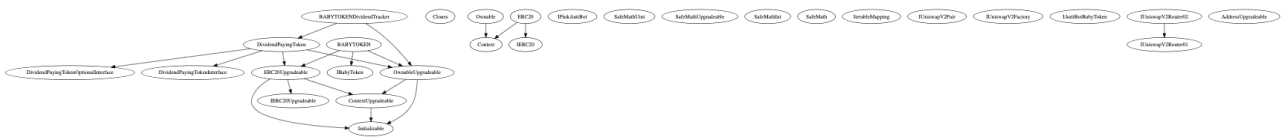
Scope of Work

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. blacklistAddress

3. claim

4. decreaseAllowance

5. excludeFromDividends

6. excludeFromFees

7. excludeMultipleAccountsFromFees

8. increaseAllowance

9. initialize

10. processDividendTracker

11. renounceOwnership

12. setAutomatedMarketMakerPair

13. setLiquidityFee

14. setMarketingFee

15. setMarketingWallet

16. setSwapTokensAtAmount

17. setTokenRewardsFee

18. transfer

19. transferFrom

20. transferOwnership

21. updateClaimWait

22. updateDividendTracker

23. updateGasForProcessing

24. updateUniswapV2Router

Deployer cannot mint any new tokens

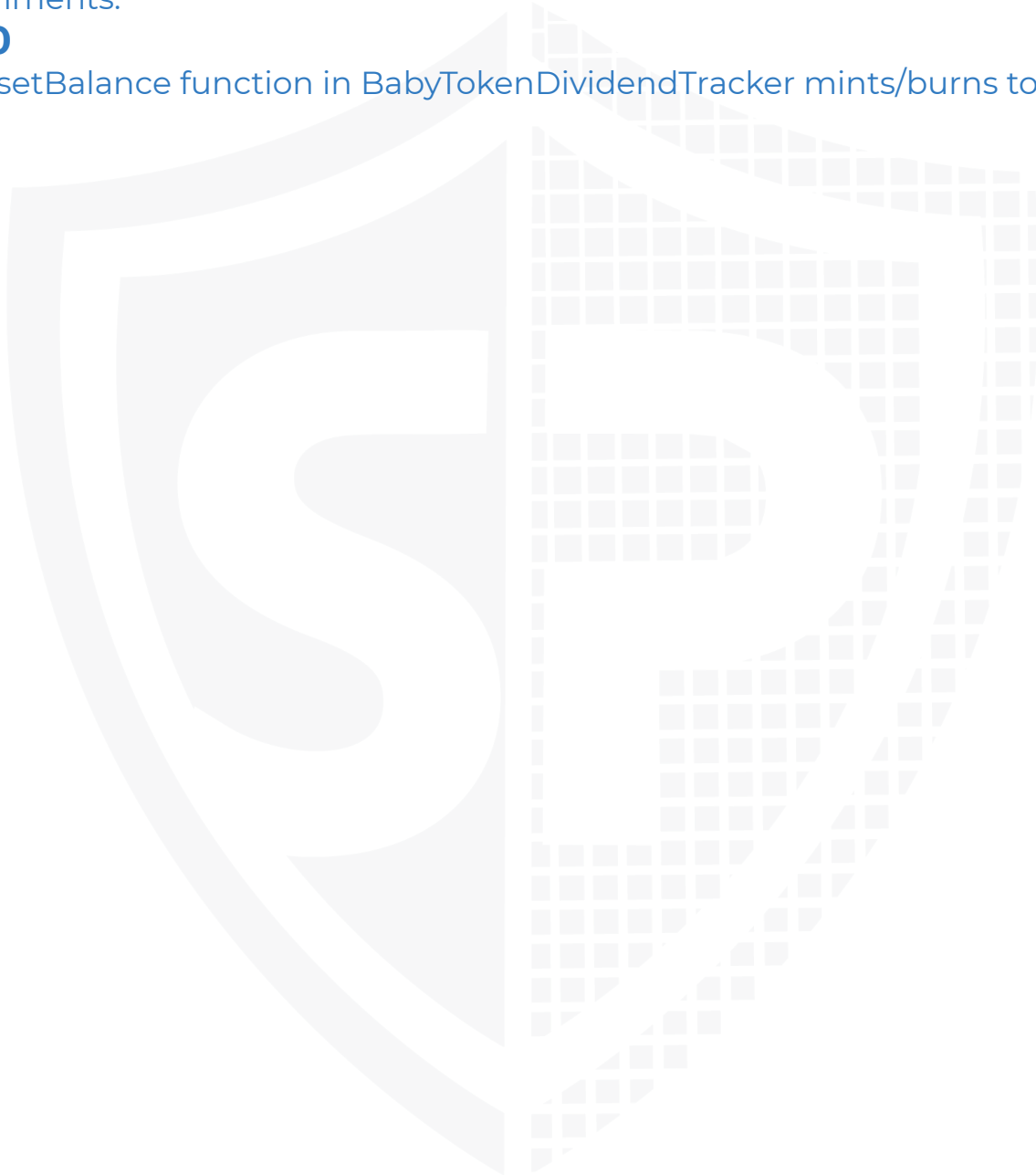
Name	Exist	Tested	Verified
Deployer cannot mint	✓	✓	✓

Max / Total Supply:

Comments:

v1.0

- `_setBalance` function in `BabyTokenDividendTracker` mints/burns token



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 by blacklisting addresses
- Deployer can set following fees without any limitations
 - tokenRewardsFee
 - liquidityFee
 - marketingFee

Deployer cannot pause the contract

Name	Exist	Tested	Verified
Deployer cannot pause	—	—	—



Overall checkup (Smart Contract Security)

Tested	Verified
✓	✓

Legend

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

Modifiers

Initializer

initialize

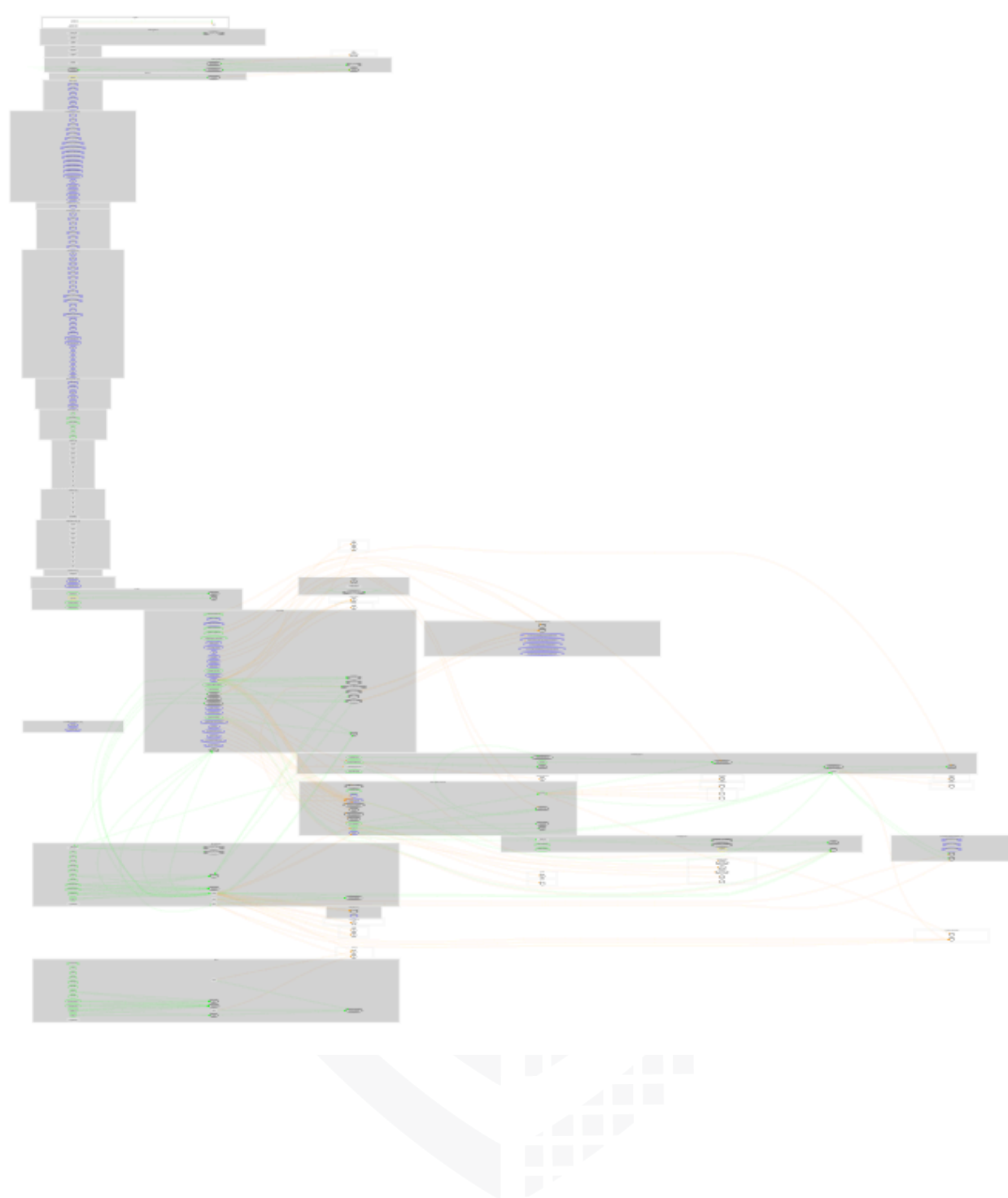
onlyOwner

```
setSwapTokensAtAmount  
updateDividendTracker  
updateUniswapV2Router  
excludeFromFees  
excludeMultipleAccountsFromFees  
setMarketingWallet  
setTokenRewardsFee  
setLiquidityFee  
setMarketingFee  
setAutomatedMarketMakerPair  
blacklistAddress  
updateGasForProcessing  
updateClaimWait  
excludeFromDividends
```

Comments









































- Following state variables can set without any limitations
 - swapTokensAtAmount
 - tokenRewardsFee
 - liquidityFee
 - marketingFee

CallGraph



Source Units in Scope

v1.0

Type	File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score	Capabilities
	contracts/ERC20Upgradeable.sol	1	————	313	313	95	185	86	
	contracts/OwnableUpgradeable.sol	1	————	75	75	33	33	31	
	contracts/BabyTokenDividendTracker.sol	2	2	482	385	251	79	182	
	contracts/Clones.sol	1	————	78	78	38	35	123	
	contracts/Ownable.sol	1	————	68	68	27	33	24	————
	contracts/BabyToken.sol	1	————	519	476	350	32	302	
	contracts/antibot/IPinkAntiBot.sol	————	1	12	5	3	1	5	————
	contracts/libs/SafeMathUint.sol	1	————	14	14	8	5	3	————
	contracts/libs/SafeMathUpgradeable.sol	1	————	214	214	61	139	16	
	contracts/libs/SafeMathInt.sol	1	————	66	66	33	23	16	————
	contracts/libs/SafeMath.sol	1	————	214	214	61	139	16	
	contracts/libs/IterableMapping.sol	1	————	64	64	49	2	19	————
	contracts/interfaces/IERC20Upgradeable.sol	————	1	77	26	17	57	13	
	contracts/interfaces/IBabyToken.sol	————	1	16	5	3	2	3	————
	contracts/interfaces/IUniswapV2Pair.sol	————	1	53	8	5	1	55	————
	contracts/interfaces/IUniswapV2Factory.sol	————	1	32	12	9	1	17	————
	contracts/interfaces/IAntiBotBabyToken.sol	————	1	16	5	3	2	3	————
	contracts/interfaces/IUniswapV2Router02.sol	————	2	208	6	4	1	64	
	contracts/interfaces/IERC20.sol	————	1	77	26	17	57	13	
	contracts/ERC20.sol	1	————	306	306	89	185	79	————
	contracts/Utils/AddressUpgradeable.sol	1	————	165	149	67	100	42	
	contracts/Utils/Context.sol	1	————	24	24	10	12	1	
	contracts/Utils/ContextUpgradeable.sol	1	————	32	32	17	12	7	
	contracts/proxy/Initializable.sol	1	————	55	55	21	24	9	
	Totals	17	11	3180	2626	1271	1160	1129	

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

Issue	File	Type	Line	Description
#1	BabyToken	Reentrancy vulnerabilities	346	Apply the [`check-effects-interactions pattern`](http://solidity.readthedocs.io/en/v0.4.21/security-considerations.html#re-entrancy).or nonReentrant modifier from OpenZeppelin
#2	BabyToken	Unchecked tokens transfer	432	Use `SafeERC20`, or ensure that the transfer/transferFrom return value is checked

Low issues

Issue	File	Type	Line	Description
#1	BabyToken, BabyTokenDividendTracker	A floating pragma is set	3	The current pragma Solidity directive is „`^0.7.6`”.
#2	BabyToken	Missing Zero Address Validation (missing-zero-check)	141, 108, 199, 171, 164,	Check that the address is not zero
#3	Main	State variable visibility is not set		It is best practice to set the visibility of state variables explicitly

Informational issues

Issue	File	Type	Line	Description
#1	BabyToken	Unused return values	499	Ensure that all the return values of the function calls are used and handle both success and failure cases if needed by the business logic
#2	SafeMathInt	Functions that are not used	56, 27, 15	Remove unused functions
#3	SafeMathInt	Unused state variables	10	Remove unused state variables
#4	BabyTokenDividendTracker	Unnecessary lines of code	203-205	<p>We recommend to delete lines below require statement</p> <p>Because of the require false statement in line 201 the lines below will not execute</p>

Audit Comments

08. December 2021:

- Deployer can lock user funds
- totalFees has limitation while deploying but not if its updated

SWC Attacks

ID	Title	Relationships	Status
SW C-13 6	Unencrypted Private Data On-Chain	CWE-767: Access to Critical Private Variable via Public Method	PASSED
SW C-13 5	Code With No Effects	CWE-1164: Irrelevant Code	NOT PASSED
SW C-13 4	Message call with hardcoded gas amount	CWE-655: Improper Initialization	PASSED
SW C-13 3	Hash Collisions With Multiple Variable Length Arguments	CWE-294: Authentication Bypass by Capture-replay	PASSED
SW C-13 2	Unexpected Ether balance	CWE-667: Improper Locking	PASSED
SW C-13 1	Presence of unused variables	CWE-1164: Irrelevant Code	NOT PASSED
SW C-13 0	Right-To-Left-Override control character (U+202E)	CWE-451: User Interface (UI) Misrepresentation of Critical Information	PASSED
SW C-12 9	Typographical Error	CWE-480: Use of Incorrect Operator	PASSED
SW C-12 8	DoS With Block Gas Limit	CWE-400: Uncontrolled Resource Consumption	PASSED

SW C-12 7	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
SW C-12 5	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SW C-12 4	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
SW C-12 3	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SW C-12 2	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SW C-12 1	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SW C-12 0	Weak Sources of Randomness from Chain Attributes	CWE-330: Use of Insufficiently Random Values	PASSED
SW C-11 9	Shadowing State Variables	CWE-710: Improper Adherence to Coding Standards	PASSED
SW C-11 8	Incorrect Constructor Name	CWE-665: Improper Initialization	PASSED
SW C-11 7	Signature Malleability	CWE-347: Improper Verification of Cryptographic Signature	PASSED

SW C-11 6	Timestamp Dependence	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SW C-11 5	Authorization through tx.origin	CWE-477: Use of Obsolete Function	PASSED
SW C-11 4	Transaction Order Dependence	CWE-362: Concurrent Execution using Shared Resource with Improper Synchronization ('Race Condition')	PASSED
SW C-11 3	DoS with Failed Call	CWE-703: Improper Check or Handling of Exceptional Conditions	PASSED
SW C-11 2	Delegatecall to Untrusted Callee	CWE-829: Inclusion of Functionality from Untrusted Control Sphere	PASSED
SW C-111	Use of Deprecated Solidity Functions	CWE-477: Use of Obsolete Function	PASSED
SW C-11 0	Assert Violation	CWE-670: Always-Incorrect Control Flow Implementation	PASSED
SW C-10 9	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SW C-10 8	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED
SW C-10 7	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	NOT PASSED
SW C-10 6	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SW C-10 5	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SW C-10 4	Unchecked Call Return Value	CWE-252: Unchecked Return Value	NOT PASSED
SW C-10 3	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	NOT PASSED
SW C-10 2	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SW C-10 1	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED
SW C-10 0	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED

The logo features the words "SolidProof" in a white, handwritten-style script. The "P" is large and 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 grid-like pattern on its right side and a solid blue area on its left side.

SolidProof

Blockchain Security | Smart Contract Audits | KYC

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

MADE IN GERMANY