



SOLIDProof
Bring trust into your projects

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

MADE IN GERMANY

Elevate Pad

Audit

Security Assessment

26. April, 2023

For



SolidProof_io



@solidproof_io

Disclaimer	3
Description	5
Project Engagement	5
Logo	5
Contract Link	5
Methodology	7
Tested Contract Files	8
Source Lines	9
Risk Level	9
Capabilities	10
Inheritance Graph	11
CallGraph	12
Scope of Work/Verify Claims	13
Modifiers and public functions	17
Source Units in Scope	22
Critical issues	23
High issues	23
Medium issues	23
Low issues	24
Informational issues	24
Audit Comments	25
Alleviation	25
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	19. April 2023 - 21. Apr 2023	<ul style="list-style-type: none">• Layout project• Automated- /Manual-Security Testing• Summary
1.1	26. April 2023	<ul style="list-style-type: none">• Reaudit

Network

Ethereum

Website

<https://www.elevatepad.io/>

Telegram

<https://t.me/elevatePAD>

Twitter

twitter.com/elevatepad



Description

ElevatePAD was initially developed by the founders of Ponyo impact, an innovative Auto-Impact token, with a focus on Coral Restoration. The Ponyo Impact team had a greater vision to help other legitimate ESG impact projects come to life. Beyond the initial development and conception of ElevatePAD, our future lies in the collective voice of our community that will push forward the ethos of DeFi. The backbone of Elevate is our strong community and potential global reach.

Project Engagement

During the Date of 17 April 2023, **Elevate Pad 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://github.com/elevate-pad/smart-contracts>
- Commit: [0429b06](#)

Note - This Audit report consists of security analysis of the Elevate smart contracts, functional testing (or unit testing) of the contract's logic was not included in this analysis.

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.

Tested Contract Files

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

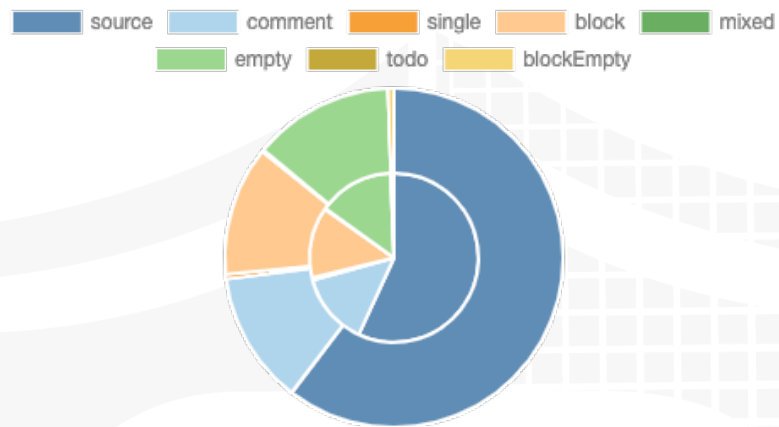
A file with a different Hash has been modified, intentionally or otherwise, after the security review. A different Hash could be (but not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of this review.

v1.0

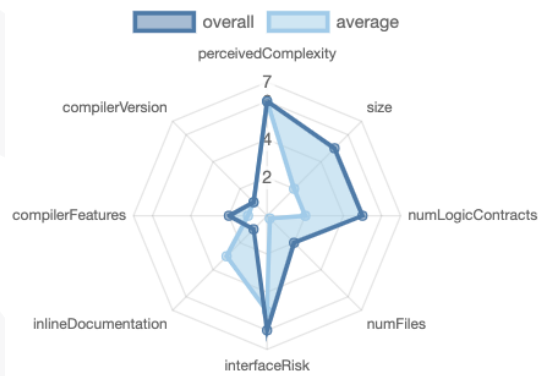
File Name	SHA-1 Hash
contracts/TokenSale/ PoolFactory.sol	0823c25c91fa5cf58ada0739340f 25d64b32ddef
contracts/TokenSale/ PresalePool.sol	7b7f67b9ad4099c4fb32fb423905 8ccc507f017c
contracts/TokenSale/ PoolManager.sol	4fd8c6663644f70befeac01b192d 26c9cb473d74
contracts/TokenCreate/ TokenFactory.sol	626389524142695500734d943e6 51f901519a02c
contracts/TokenCreate/ BabyToken.sol	f989324b2f344ee40e87a1c6c6fb ef0a4d1a3ce5
contracts/TokenCreate/ BuybackBabyToken.sol	da3a8f323617b630eeea8e46142 5d8a73f8cb595
contracts/TokenCreate/ StandardToken.sol	a0df3d3dfdb3c98b75408b7adf4fc 31ae45ecb94
contracts/TokenCreate/ LiquidityGeneratorToken.sol	b6da49ce8ed6a04864459998bc5 f47b27fb73e58
contracts/TokenLock/ ElevateLocker.sol	1bf1ef3748632689be0e16c6ad0e cc7c559bd0d1

Metrics

Source Lines v1.0



Risk Level v1.0



Capabilities

Components

 Contracts	 Libraries	 Interfaces	 Abstract
18	31	53	33


Exposed Functions

This section lists functions that are explicitly declared public or payable. Please note that getter methods for public stateVars are not included.





 Public	 Payable
761	47

External	Internal	Private	Pure	View
568	993	74	164	384


StateVariables

Total	 Public
313	185

Capabilities

Solidity Versions observed	 Experimental Features	 Can Receive Funds	 Uses Assembly	 Has Destroyable Contracts
<code>=0.8.4</code> <code>^0.8.4</code> <code>0.8.17</code>	<code>ABIEncoderV2</code>	<code>yes</code>	<code>yes</code> (41 asm blocks)	

 Transfers ETH	 Low-Level Calls	 DelegateCall	 Uses Hash Functions	 ECRrecover	 New/Create/Create2
<code>yes</code>		<code>yes</code>	<code>yes</code>		<code>yes</code> → <code>AssemblyCall:Name:create</code> → <code>AssemblyCall:Name:create2</code> → <code>NewContract:DividendDistributor</code>

 TryCatch	 Unchecked
<code>yes</code>	<code>yes</code>

Inheritance Graph

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. Is contract an upgradeable
2. Correct implementation of Token standard
3. Overall checkup (Smart Contract Security)



Is contract an upgradeable

Name	
Is contract an upgradeable?	Yes

Comments:

v1.0

- Owner can deploy a new version of the Pool and Token contracts which can change any limit and give owner new privileges
 - Be aware of this and do your own research for the contract which is the contract pointing to

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	✓	✓	✓

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

BabyToken

- ◆ setSwapTokensAtAmount
- Ⓜ onlyOwner
- ◆ updateDividendTracker
- Ⓜ onlyOwner
- ◆ updateUniswapV2Router
- Ⓜ onlyOwner
- ◆ excludeFromFees
- Ⓜ onlyOwner
- ◆ excludeMultipleAccountsFromFees
- Ⓜ onlyOwner
- ◆ setMarketingWallet
- Ⓜ onlyOwner
- ◆ setTokenRewardsFee
- Ⓜ onlyOwner
- ◆ setLiquiditFee
- Ⓜ onlyOwner
- ◆ setMarketingFee
- Ⓜ onlyOwner
- ◆ setAutomatedMarketMakerPair
- Ⓜ onlyOwner
- ◆ updateGasForProcessing
- Ⓜ onlyOwner
- ◆ updateClaimWait
- Ⓜ onlyOwner
- ◆ updateMinimumTokenBalanceForDividends
- Ⓜ onlyOwner
- ◆ excludeFromDividends
- Ⓜ onlyOwner
- ◆ processDividendTracker
- ◆ claim

PoolFactory

- ◆ initializeVesting
- Ⓜ onlyOperator
- ◆ setMinLockDays
- Ⓜ onlyOwner
- ◆ addWhitelistedUsers
- ◆ addWhitelistedUser
- ◆ removeWhitelistedUsers
- ◆ removeWhitelistedUser
- ◆ contribute 💰
- ◆ claim
- ◆ withdrawContribution
- Ⓜ noReentrant
- ◆ finalize
- Ⓜ onlyOperator
- Ⓜ noReentrant
- ◆ cancel
- ◆ withdrawLeftovers
- Ⓜ onlyOperator
- Ⓜ noReentrant
- ◆ withdrawLiquidity
- Ⓜ onlyOperator
- ◆ emergencyWithdrawLiquidity
- Ⓜ onlyOwner
- ◆ emergencyWithdrawToken
- Ⓜ onlyOwner
- ◆ emergencyWithdraw
- Ⓜ onlyOwner
- ◆ updatePoolDetails
- Ⓜ onlyOperator
- ◆ updateCompletedKyc
- Ⓜ onlyOwner
- ◆ setGovernance
- Ⓜ onlyOwner
- ◆ setKycAudit
- Ⓜ onlyOwner
- ◆ setWhitelisting
- Ⓜ onlyOperator

PresalePool

- ◆ initializeVesting
- Ⓜ onlyOperator
- ◆ setMinLockDays
- Ⓜ onlyOwner
- ◆ addWhitelistedUsers
- ◆ addWhitelistedUser
- ◆ removeWhitelistedUsers
- ◆ removeWhitelistedUser
- ◆ contribute 💰
- ◆ claim
- ◆ withdrawContribution
- Ⓜ noReentrant
- ◆ finalize
- Ⓜ onlyOperator
- Ⓜ noReentrant
- ◆ cancel
- ◆ withdrawLeftovers
- Ⓜ onlyOperator
- Ⓜ noReentrant
- ◆ withdrawLiquidity
- Ⓜ onlyOperator
- ◆ emergencyWithdrawLiquidity
- Ⓜ onlyOwner
- ◆ emergencyWithdrawToken
- Ⓜ onlyOwner
- ◆ emergencyWithdraw
- Ⓜ onlyOwner
- ◆ updatePoolDetails
- Ⓜ onlyOperator
- ◆ updateCompletedKyc
- Ⓜ onlyOwner
- ◆ setGovernance
- Ⓜ onlyOwner
- ◆ setKycAudit
- Ⓜ onlyOwner
- ◆ setWhitelisting
- Ⓜ onlyOperator

PoolManager

- ◆ initialize
- ◆ addPoolFactory
- Ⓜ onlyAllowedFactory
- ◆ addAdminPoolFactory
- Ⓜ onlyOwner
- ◆ addPoolFactories
- Ⓜ onlyOwner
- ◆ removePoolFactory
- Ⓜ onlyOwner
- ◆ registerPool
- Ⓜ onlyAllowedFactory
- ◆ increaseTotalValueLocked
- Ⓜ onlyAllowedFactory
- ◆ decreaseTotalValueLocked
- Ⓜ onlyAllowedFactory
- ◆ recordContribution
- Ⓜ onlyAllowedFactory
- ◆ removePoolForToken
- Ⓜ onlyAllowedFactory
- ◆ initializeTopPools
- Ⓜ onlyOwner
- ◆ addTopPool
- Ⓜ onlyAllowedFactory
- ◆ removeTopPool
- Ⓜ onlyAllowedFactory
- ◆ ethLiquidity
- Ⓜ onlyOwner
- ◆ transferAnyERC20Token
- Ⓜ onlyOwner

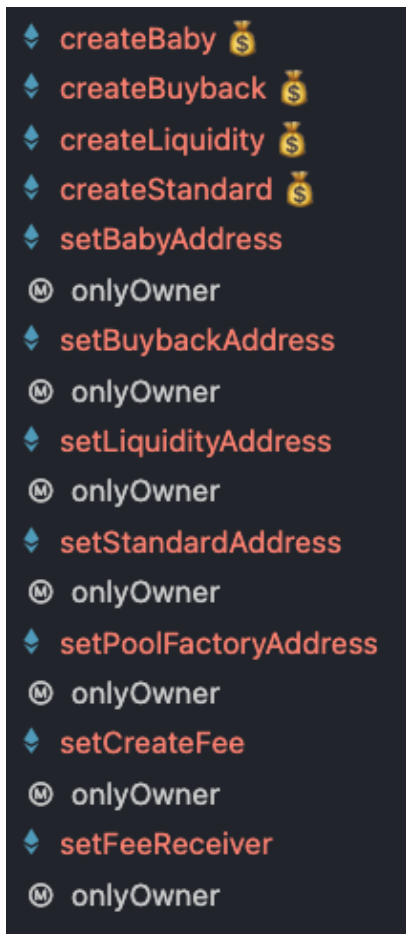
BuybackBabyToken

- ◆ approve
- ◆ approveMax
- ◆ transfer
- ◆ transferFrom
- ◆ triggerZeusBuyback
- Ⓜ authorized
- ◆ clearBuybackMultiplier
- Ⓜ authorized
- ◆ setAutoBuybackSettings
- Ⓜ authorized
- ◆ setBuybackMultiplierSettings
- Ⓜ authorized
- ◆ setIsDividendExempt
- Ⓜ authorized
- ◆ setIsFeeExempt
- Ⓜ authorized
- ◆ setBuyBacker
- Ⓜ authorized
- ◆ setFees
- Ⓜ authorized
- ◆ setFeeReceivers
- Ⓜ authorized
- ◆ setSwapBackSettings
- Ⓜ authorized
- ◆ setTargetLiquidity
- Ⓜ authorized
- ◆ setDistributionCriteria
- Ⓜ authorized
- ◆ setDistributorSettings
- Ⓜ authorized

LiquidityGeneratorToken

- ◆ transfer
- ◆ approve
- ◆ transferFrom
- ◆ increaseAllowance
- ◆ decreaseAllowance
- ◆ deliver
- ◆ excludeFromReward
- Ⓜ onlyOwner
- ◆ includeInReward
- Ⓜ onlyOwner
- ◆ excludeFromFee
- Ⓜ onlyOwner
- ◆ includeInFee
- Ⓜ onlyOwner
- ◆ setTaxFeePercent
- Ⓜ onlyOwner
- ◆ setLiquidityFeePercent
- Ⓜ onlyOwner
- ◆ setCharityFeePercent
- Ⓜ onlyOwner
- ◆ setMarketingFeePercent
- Ⓜ onlyOwner
- ◆ setMarketingWallet
- Ⓜ onlyOwner
- ◆ setCharityWallet
- Ⓜ onlyOwner
- ◆ setSwapAndLiquifyEnabled
- Ⓜ onlyOwner

TokenFactory



Ownership/Authorized Privileges

- BabyToken.sol
 - Set amount to swap tokens at, to any arbitrary value
 - Update dividend tracker, AMM, and uniswapV2 router address
 - Include/Exclude accounts from fees
 - Set Marketing Wallet address
 - Set Rewards, Liquidity, and Marketing fees, but not more than 25%
 - Update minimum balance for dividends to any arbitrary value
 - Set dividend balance of an account
 - Exclude from Dividends
 - Update claim wait to anywhere between 1 and 24 hrs
 - Update minimum token balance to any arbitrary value
- BuybackBabyToken.sol (authorised addresses privileges)
 - Authorized addresses can trigger buyback
 - Set buy back multiplier to zero
 - Set auto buyback settings in which the authorised address can set the following to any arbitrary value

- Status (true or false)
 - Cap
 - Amount
 - Period
- Include/Exclude accounts from dividends, and fees
- Set buyback address
- Set fees, but not more than 25%
- Set fee receiver addresses
- Set distribution criteria of the dividends
- Set target liquidity to any arbitrary value
- [LiquidityGenerator.sol](#)
 - Owner can include/exclude accounts from the reward and fee
 - Set tax, liquidity, marketing, and charity fee percent, but not more than 25.
 - Set charity, and marketing wallet address
 - Enable/Disable swap and liquify
- [TokenFactory.sol](#)
 - Set baby token, and buyback baby token address
 - Set liquidity token, standard token, and pool factory contract addresses
 - Set token create fee address
 - Set token creation fee to any arbitrary value but it will not affect users because this fee is not implemented anywhere
- [PoolFactory.sol](#)
 - Owner can set the following addresses
 - Master, Elevate, Admin Wallet
 - Token Factory
 - Pool Owner
 - Pool Manager
 - Owner can set the following to any arbitrary value
 - Version
 - Contribute withdraw fee
 - KYC and Audit Price
 - Presale pool price
 - Minimum lock days
 - Minimum ETH to raise
 - Bonus Base Per tier
 - Withdraw ETH and any other ERC20 from the contract
 - Update KYC and Audit status for a given address
 - Withdraw liquidity, and tokens from any given pool address which is not recommended

- [PoolManager.sol](#)
 - Factory addresses and the owner can set/add a new pool factory address
 - Owner can remove factory address
 - Factory addresses can register a new pool
 - Factory addresses can increase/decrease the total value locked for a particular currency to any arbitrary value
 - Factory addresses can manually set the contribution of a user to any particular pool
 - Factory addresses can remove the pool for a token address
 - The owner can initialise top pools
 - The factory addresses can add, and remove top pools
 - Owner can withdraw ETH and any other ERC20 from the contract
- [PresalePool.sol](#)
 - Operator address has the following privileges
 - Initialise vesting
 - Add/Remove whitelisted users, and only these users will be able to contribute in the presale
 - Finalize/Finish a pool when the total raised value has reached the hardcap, or soft cap and the end time has passed
 - Cancel a pool anytime
 - Withdraw contract's balance to the governance address if the end time is reached and total raised value is less than the soft cap
 - Withdraw liquidity once the liquidity unlock time has been passed.
 - Update pool details
 - Owner address has the following privileges
 - Set minimum lock days
 - Update the status of completed KYC
 - Set governance address
 - Withdraw liquidity, and tokens from any given pool address which is not recommended
 - Set KYC and Audit status
- 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

Please check if an OnlyOwner or similar restrictive modifier has been forgotten.

Source Units in Scope v1.0

File	Logic Contracts	Interfaces	Lines	nLines	nSLOC	Comment Lines	Complex. Score
contracts/TokenSale/PoolFactory.sol	8	7	1294	942	603	292	532
contracts/TokenSale/PresalePool.sol	12	7	2072	1671	1086	426	967
contracts/TokenSale/PoolManager.sol	8	6	1630	1248	758	398	627
contracts/TokenCreate/TokenFactory.sol	6	4	577	417	309	66	348
contracts/TokenCreate/BabyToken.sol	16	10	3152	2423	1208	1003	1093
contracts/TokenCreate/BuybackBabyToken.sol	7	5	1542	1128	732	235	681
contracts/TokenCreate/StandardToken.sol	6	1	804	683	278	377	161
contracts/TokenCreate/LiquidityGeneratorToken.sol	8	4	1743	1279	789	407	559
contracts/TokenLock/ElevateLocker.sol	7	6	2142	1489	971	453	766
contracts/ElevateToken.sol	4	3	645	618	474	25	446
Totals	82	53	15601	11898	7208	3682	6180

Legend

Attribute	Description
Lines	total lines of the source unit
nLines	normalised lines of the source unit (e.g. normalises functions spanning multiple lines)
nSLOC	normalised 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

Critical issues

No critical issues

High issues

No high issues

Medium issues

Acknowledged

Issue	File	Type	Line	Description
#1	BuyBak cBabyTo ken.sol	Liquidity goes to an EOA	1354	The liquidity of the contract will be credited to an EOA that is the 'autoLiquidityReceiver' address every time there is a snapback in the transfer function. The receiver address can also be changed by the authorised addresses. This way the authorised address may be able to drain the liquidity.
#2	PoolFac tory.sol	Fees can be 100% or more	1005	The owner is able to set the contribution withdraw fee up to 100% or more, and if done so then the complete amount of refund for an user that has contributed in the pool will go to the admin wallet, and the user will get 0 refund
#3	PresalP ool.sol	Owner can withdraw funds	1946-1961	The owner is able to withdraw all funds from the Presale Pool address or any other pool address using the PoolFactory contract which is not recommended, even if it is stated as an emergency function, but the owner has the liberty to use them anytime.

Low issues

Issue	File	Type	Line	Description
#1	All	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	BabyToken.sol	Missing Zero Address Validation (missing-zero-check)	2727-2785	Check that the address is not zero
#3	PoolManager.sol	Missing Zero Address Validation (missing-zero-check)	1250, 1384, 1418	Check that the address is not zero
#4	BabyToken.sol	Local variables shadowing	2107, 2108, 2200, 2174, 2186	Rename the local variables that shadow another component
#5	ElevateLocker.sol	Local variables shadowing	1514, 1553, 1587, 1620, 1349	Rename the local variables that shadow another component
#6	PoolFactory.sol	Missing Events Arithmetic	All	Emit an event for critical parameter changes. The contract has no events

Informational issues

Issue	File	Type	Line	Description
#1	BuyBackBabyToken.sol	State variables that could be declared immutable (immutable-states)	734, 755, 742	Add the `immutable` attributes to state variables that never change
#2	ElevateLocker.sol	Unused return values	1552, 1586, 1660, 2105, 1710	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
#3	PoolManager.sol	Unused return values	1223, 1219, 1250, 1233	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

#4	StandardToken.sol	Functions that are not used	736, 781	Remove unused functions. Before removing check the function, it could be possible, that you forget to implement it into the contract
#5	All	NatSpec documentation missing	—	If you started to comment your code, also comment all other functions, variables etc.

Audit Comments

We recommend you to use the special form of comments (NatSpec Format, Follow link for more information <https://docs.soliditylang.org/en/latest/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.

Alleviation

The medium bugs have been acknowledged by the Elevate Pad team, and it is informed to us that the bugs are indeed part of the 'Intended Behaviour' for the Launchpad. Comments from the ElevatePad team -

"Yeah I meant ones that are not vulnerabilities. Memepad is high risk non kyc. The Launchpad is kyc and verified teams. It is the user to trust in the teams they invest with"

26. April 2023:

- Unit tests with 100% code coverage was not provided to SolidProof so we cannot ensure complete functional correctness of the code's logic.
- We recommend Elevate team to conduct unit and fuzz tests thoroughly to rule out possibilities of an unwanted logical and calculation errors.
- There is still an owner (Owner still has not renounced ownership)
- Owner can deploy a new version of the contract which can change any limit and give owner new privileges
- Read whole report and modifiers section for more information

SWC Attacks

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

SW C-1 27	Arbitrary Jump with Function Type Variable	CWE-695: Use of Low-Level Functionality	PASSED
SW C-1 25	Incorrect Inheritance Order	CWE-696: Incorrect Behavior Order	PASSED
SW C-1 24	Write to Arbitrary Storage Location	CWE-123: Write-what-where Condition	PASSED
SW C-1 23	Requirement Violation	CWE-573: Improper Following of Specification by Caller	PASSED
SW C-1 22	Lack of Proper Signature Verification	CWE-345: Insufficient Verification of Data Authenticity	PASSED
SW C-1 21	Missing Protection against Signature Replay Attacks	CWE-347: Improper Verification of Cryptographic Signature	PASSED
SW C-1 20	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	NOT 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-11 1	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-1 09	Uninitialized Storage Pointer	CWE-824: Access of Uninitialized Pointer	PASSED
SW C-1 08	State Variable Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED
SW C-1 07	Reentrancy	CWE-841: Improper Enforcement of Behavioral Workflow	PASSED
SW C-1 06	Unprotected SELFDESTRUCT Instruction	CWE-284: Improper Access Control	PASSED

SW C-1 05	Unprotected Ether Withdrawal	CWE-284: Improper Access Control	PASSED
SW C-1 04	Unchecked Call Return Value	CWE-252: Unchecked Return Value	PASSED
SW C-1 03	Floating Pragma	CWE-664: Improper Control of a Resource Through its Lifetime	PASSED
SW C-1 02	Outdated Compiler Version	CWE-937: Using Components with Known Vulnerabilities	PASSED
SW C-1 01	Integer Overflow and Underflow	CWE-682: Incorrect Calculation	PASSED
SW C-1 00	Function Default Visibility	CWE-710: Improper Adherence to Coding Standards	PASSED

*Solid
Proofed*

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


MADE IN GERMANY

