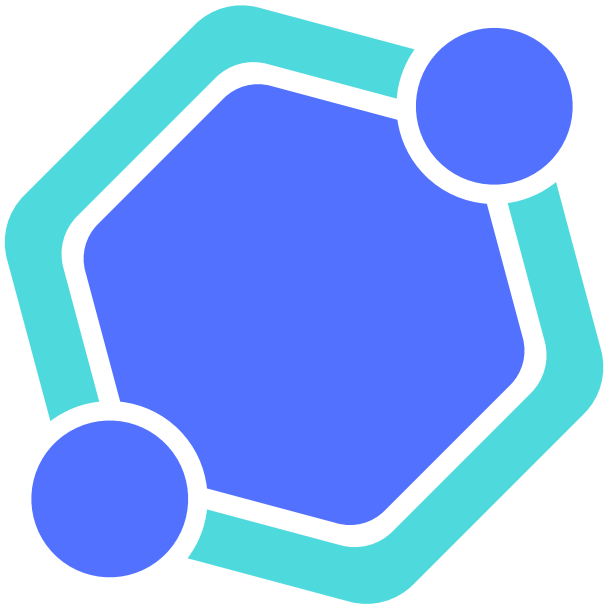
AUDIT REPORT

April 2023



Audit conducted by

RICARDO PONTES

Summary

Auditing Firm	Crypto Hub
Architecture	Crypto Hub Auditing Standard
Smart Contract Audit Approved By	Ricardo | Blockchain Dev at Crypto Hub
Platform	Solidity
Mandatory Audit Check	Static, Software & Manual Analysis
Consultation Request Date	April 29, 2023
Report Date	April 29, 2023

Audit Summary

Crypto Hub team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks. According to the smart contract audit:

- ★ Baby Tate smart contract source code has **LOW RISK SEVERITY**.
- ★ Baby Tate has **PASSED** the smart contract audit.

For the detailed understanding of risk severity, source code vulnerability, and functional test, kindly refer to the audit.

✅ Verify the authenticity of this report on Crypto Hub Website: https://www.cryptohub.agency/

Table Of Contents

Project Overview

Crypto Hub was consulted by Baby Tate to conduct the smart contract security audit of their solidity source code.

Project	Baby Tate
Blockchain	Ethereum Mainnet
Language	Solidity
Contracts	0x069714f1AFe6464472667EA83f43B22E0c866158
Website:	http://babytate.website/

Public logo:

Solidity Source Code On Blockchain (Verified Contract Source Code)

https://etherscan.io/token/0x069714f1AFe6464472667EA83f43B22E0c866158#code

Contract Name: BabyTate

Compiler Version: v0.8.17

Optimization Enabled: yes

SHA-1 Hash

Solidity source code is audited at hash

#04ddfbb34692aa909d50985454c102ec2cf70392

Audit Scope & Methodology

The scope of this report is to audit the above smart contract source code and Crypto Hub has scanned the contract and reviewed the project for common vulnerabilities, exploits, hacks, and back-doors. Below is the list of commonly known smart contract vulnerabilities, exploits, and hacks:

Smart Contract Vulnerabilities

- ☐ Re-entrancy
- ☐ Unhandled Exceptions
- ☐ Transaction Order Dependency
- ☐ Integer Overflow
- ☐ Unrestricted Action
- ☐ Incorrect Inheritance Order
- ☐ Typographical Errors
- ☐ Requirement Violation

Source Code Review

- ☐ Ownership Takeover
- ☐ Gas Limit and Loops
- ☐ Deployment Consistency
- ☐ Repository Consistency
- ☐ Data Consistency
- ☐ Token Supply Manipulation

Functional Assessment

- ☐ Access Control and Authorization
- ☐ Operations Trail and Event Generation
- ☐ Assets Manipulation
- ☐ Liquidity Access

Crypto Hub Audit Standard

The aim of Crypto Hub standard is to analyze the smart contract and identify the vulnerabilities and the hacks in the smart contract. Mentioned are the steps used by ECHELON-1 to assess the smart contract:

1. Solidity smart contract source code reviewal:
   - Review of the specifications, sources, and instructions provided to Crypto Hub to make sure we understand the size, scope, and functionality of the smart contract.
   - Manual review of code, which is the process of reading source code line-by-line to identify potential vulnerabilities.

2. Static, Manual, and Software analysis:
   - Test coverage analysis, which is the process of determining whether the test cases are covering the code and how much code is exercised when we run those test cases.
   - Symbolic execution, which is analyzing a program to determine what inputs cause 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

Automated 3P frameworks used to assess the smart contract vulnerabilities

- Slither
- Consensys MythX
- Consensus Surya
- Open Zeppelin Code Analyzer
- Solidity Code Complier

Crypto Hub's Risk Classification

Smart contracts are generally designed to manipulate and hold funds denominated in ETH/BNB. This makes them very tempting attack targets, as a successful attack may allow the attacker to directly steal funds from the contract. Below are the typical risk levels of a smart contract: Vulnerable:

A contract is vulnerable if it has been flagged by a static analysis tool as such. As we will see later, this means that some contracts may be vulnerable because of a false-positive.

Exploitable: A contract is exploitable if it is vulnerable and the vulnerability could be exploited by an external attacker. For example, if the "vulnerability" flagged by a tool is in a function which requires to own the contract, it would be vulnerable but not exploitable.

Exploited: A contract is exploited if it received a transaction on the main network which triggered one of its vulnerabilities. Therefore, a contract can be vulnerable or even exploitable without having been exploited.

Risk severity	Meaning
! Critical	This level of vulnerability could be exploited easily, and can lead to asset loss, data loss, asset manipulation, or data manipulation. They should be fixed right away.
! High	This level vulnerabilities are hard to exploit but very important to fix, they carry an elevated risk of smart contract manipulation, which can lead to critical risk severity
! Medium	This level of vulnerabilities should be fixed, as they carry an inherent risk of future exploits, and hacks which may or may not impact the smart contract execution.
! Low	This level of vulnerability can be ignored. They are code style violations, and informational statements in the code. They may not affect the smart contract execution

Smart Contract Risk Assessment

Contract Snapshot

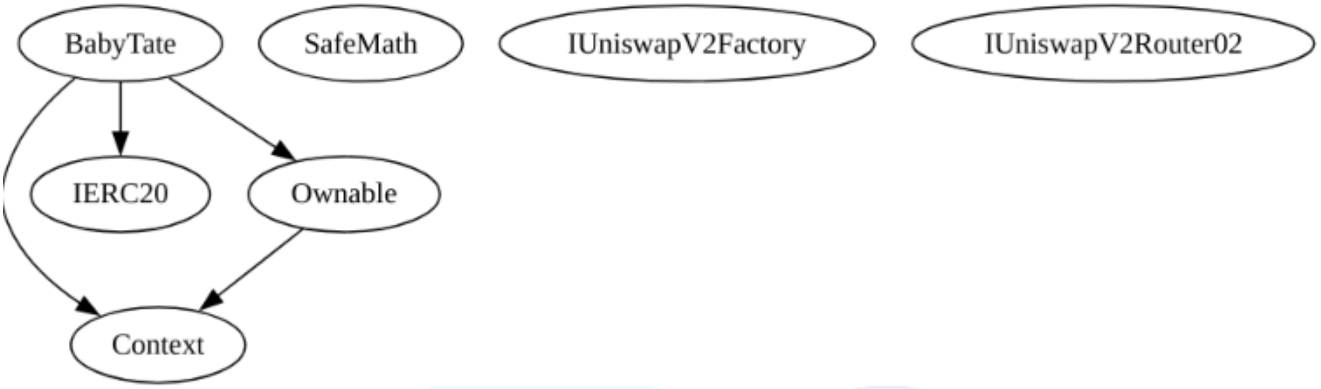
```
/**
 *Submitted for verification at Etherscan.io on 2023-04-28
*/

/**
```

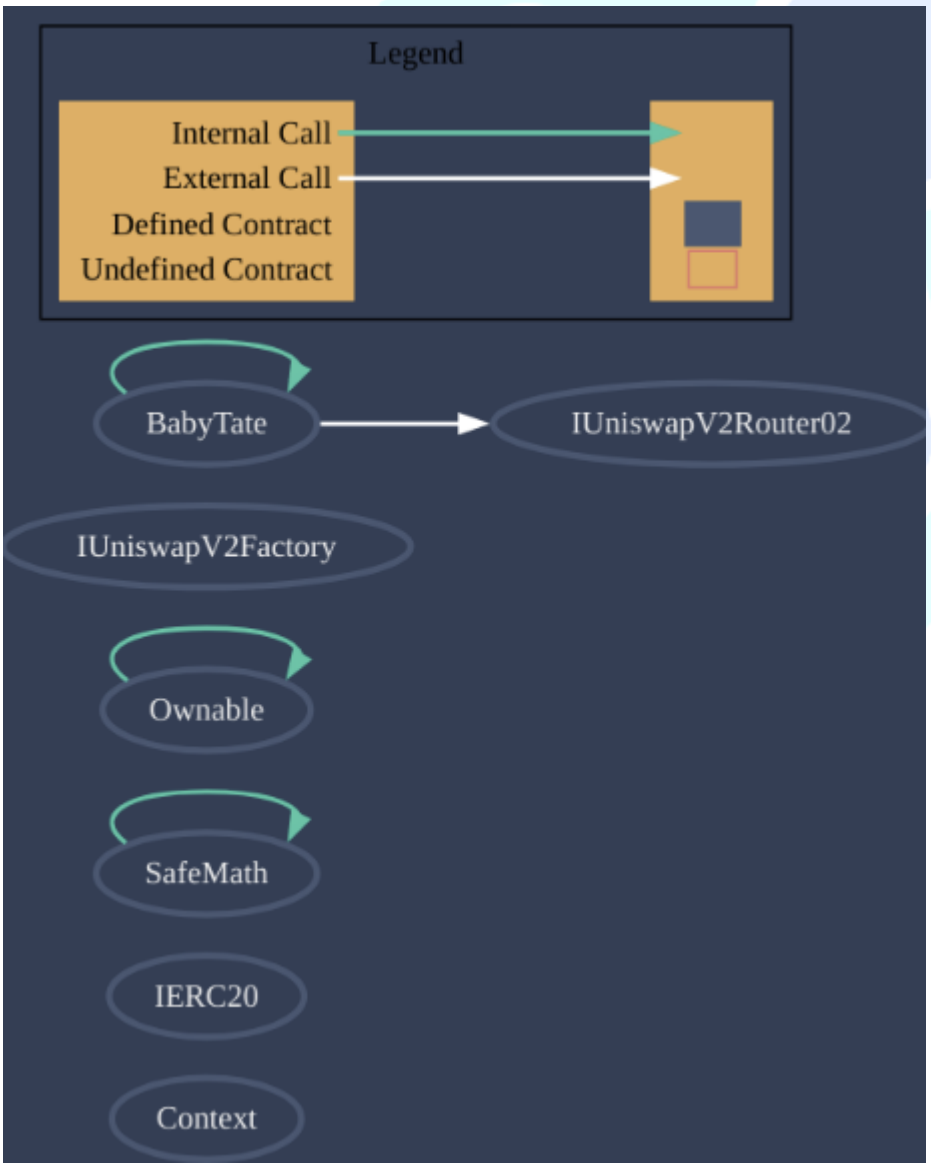


```


Telegram: https://t.me/babytateeth
Twitter: https://twitter.com/BabyTateEth
Website: babytate.website


*/

// SPDX-License-Identifier: MIT

pragma solidity 0.8.17;

abstract contract Context {
    function _msgSender() internal view virtual returns (address) {
        return msg.sender;
    }
}

interface IERC20 {
    function totalSupply() external view returns (uint256);
    function balanceOf(address account) external view returns (uint256);
    function transfer(address recipient, uint256 amount) external returns (bool);
    function allowance(address owner, address spender) external view returns (uint256);
    function approve(address spender, uint256 amount) external returns (bool);
    function transferFrom(address sender, address recipient, uint256 amount) external returns (bool);
    event Transfer(address indexed from, address indexed to, uint256 value);
    event Approval(address indexed owner, address indexed spender, uint256 value);
}

library SafeMath {
    function add(uint256 a, uint256 b) internal pure returns (uint256) {
        uint256 c = a + b;
        require(c >= a, "SafeMath: addition overflow");
        return c;
    }

    function sub(uint256 a, uint256 b) internal pure returns (uint256) {
        return sub(a, b, "SafeMath: subtraction overflow");
    }
```

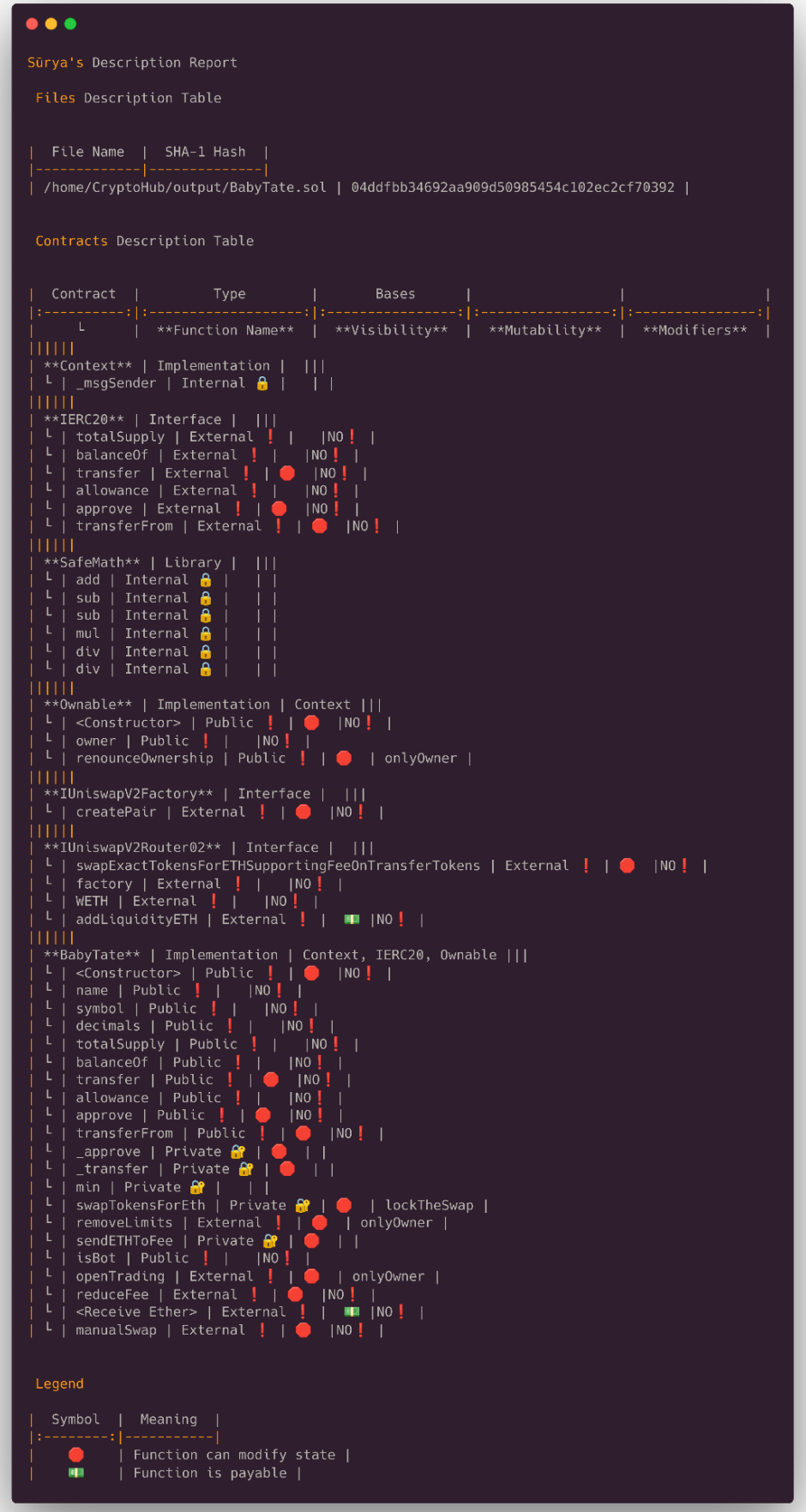
Static / Quick Analysis

Software Analysis





```
Sighash   |   Function Signature
========================
119df25f  =>  _msgSender()
18160ddd  =>  totalSupply()
70a08231  =>  balanceOf(address)
a9059cbb  =>  transfer(address,uint256)
dd62ed3e  =>  allowance(address,address)
095ea7b3  =>  approve(address,uint256)
23b872dd  =>  transferFrom(address,address,uint256)
771602f7  =>  add(uint256,uint256)
b67d77c5  =>  sub(uint256,uint256)
e31bdc0a  =>  sub(uint256,uint256,string)
c8a4ac9c  =>  mul(uint256,uint256)
a391c15b  =>  div(uint256,uint256)
b745d336  =>  div(uint256,uint256,string)
8da5cb5b  =>  owner()
715018a6  =>  renounceOwnership()
c9c65396  =>  createPair(address,address)
791ac947  =>
swapExactTokensForETHSupportingFeeOnTransferTokens(uint256,uint256,address[],address,uint256)
c45a0155  =>  factory()
ad5c4648  =>  WETH()
f305d719  =>  addLiquidityETH(address,uint256,uint256,uint256,address,uint256)
06fdde03  =>  name()
95d89b41  =>  symbol()
313ce567  =>  decimals()
104e81ff  =>  _approve(address,address,uint256)
30e0789e  =>  _transfer(address,address,uint256)
7ae2b5c7  =>  min(uint256,uint256)
b28805f4  =>  swapTokensForEth(uint256)
751039fc  =>  removeLimits()
06b50197  =>  sendETHToFee(uint256)
3bbac579  =>  isBot(address)
c9567bf9  =>  openTrading()
ec1f3f63  =>  reduceFee(uint256)
51bc3c85  =>  manualSwap()
```



Sürya's Description Report

Files Description Table

File Name	SHA-1 Hash
/home/CryptoHub/output/BabyTate.sol	04ddfbb34692aa909d50985454c102ec2cf70392

Contracts Description Table

Contract	Type	Bases		
└	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
Context	Implementation			
└	_msgSender	Internal 🔒		
IERC20	Interface			
└	totalSupply	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	transfer	External ❗	🛑	NO❗
└	allowance	External ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
SafeMath	Library			
└	add	Internal 🔒		
└	sub	Internal 🔒		
└	sub	Internal 🔒		
└	mul	Internal 🔒		
└	div	Internal 🔒		
└	div	Internal 🔒		
Ownable	Implementation	Context		
└	<Constructor>	Public ❗	🛑	NO❗
└	owner	Public ❗		NO❗
└	renounceOwnership	Public ❗	🛑	onlyOwner
IUniswapV2Factory	Interface			
└	createPair	External ❗	🛑	NO❗
IUniswapV2Router02	Interface			
└	swapExactTokensForETHSupportingFeeOnTransferTokens	External ❗	🛑	NO❗
└	factory	External ❗		NO❗
└	WETH	External ❗		NO❗
└	addLiquidityETH	External ❗	💵	NO❗
BabyTate	Implementation	Context, IERC20, Ownable		
└	<Constructor>	Public ❗	🛑	NO❗
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗
└	decimals	Public ❗		NO❗
└	totalSupply	Public ❗		NO❗
└	balanceOf	Public ❗		NO❗
└	transfer	Public ❗	🛑	NO❗
└	allowance	Public ❗		NO❗
└	approve	Public ❗	🛑	NO❗
└	transferFrom	Public ❗	🛑	NO❗
└	_approve	Private 🔐	🛑	
└	_transfer	Private 🔐	🛑	
└	min	Private 🔐		
└	swapTokensForEth	Private 🔐	🛑	lockTheSwap
└	removeLimits	External ❗	🛑	onlyOwner
└	sendETHToFee	Private 🔐	🛑	
└	isBot	Public ❗		NO❗
└	openTrading	External ❗	🛑	onlyOwner
└	reduceFee	External ❗	🛑	NO❗
└	<Receive Ether>	External ❗	💵	NO❗
└	manualSwap	External ❗	🛑	NO❗

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
IUniswapV2Router02 (iface)
factory
WETH
swapExactTokensForETHSupportingFeeOnTransferTokens
addLiquidityETH
IUniswapV2Factory (iface)
createPair
BabyTate
lockTheSwap
<Constructor>
name
symbol
decimals
totalSupply
transferFrom
allowance
reduceFee
removeLimits
isBot
transfer
<Receive Ether>
payable
openTrading
manualSwap
_transfer
type
balanceOf
sendETHToFee
owner
swapTokensForEth
approve
min
_approve
amount
mul
sub
_taxWallet
transfer
SafeMath (lib)
add
sub
mul
div
IERC20 (iface)
totalSupply
balanceOf
transfer
allowance
approve
transferFrom
Ownable
renounceOwnership
owner
<Constructor>
onlyOwner
address
_msgSender
Context
_msgSender

SWC Attacks

The following table contains an overview of the SWC registry. Each row consists of an SWC identifier (ID), weakness title, CWE parent and list of related code samples.

The auditor used a MythX tool, A static analyzer that parses the Soldity AST, a symbolic analyzer that detects possible vulnerable states, and a greybox fuzzer that detects vulnerable execution paths.

ID	Description	Status
SWC - 100	Function Default Visibility	✔️ **Passed**
SWC - 101	Integer Overflow and Underflow	✔️ **Passed**
SWC - 102	Outdated Compiler Version	✔️ **Passed**
SWC - 103	Floating Pragma	✔️ **Passed**
SWC - 104	Unchecked Call Return Value	✔️ **Passed**
SWC - 105	Unprotected Ether Withdrawal	✔️ **Passed**
SWC - 106	Unprotected SELFDESTRUCT Instruction	✔️ **Passed**
SWC - 107	Reentrancy Passed	✔️ **Passed**
SWC - 108	State Variable Default Visibility	✔️ **Passed**
SWC - 109	Uninitialized Storage Pointer	✔️ **Passed**
SWC - 110	Assert Violation Passed	✔️ **Passed**
SWC - 111	Use of Deprecated Solidity Functions	✔️ **Passed**
SWC - 112	Delegatecall to Untrusted Callee	✔️ **Passed**
SWC - 113	DoS with Failed Call	✔️ **Passed**
SWC - 114	Transaction Order Dependence	✔️ **Passed**
SWC - 115	Authorization through tx.origin	✔️ **Passed**
SWC - 116	Block values as a proxy for time	✔️ **Passed**
SWC - 117	Signature Malleability	✔️ **Passed**

ID	Description	Status
SWC - 118	Incorrect Constructor Name	✔️ **Passed**

SWC - 119	Shadowing State Variables	✔️ **Passed**
SWC - 120	Weak Sources of Randomness from Chain Attributes	✔️ **Passed**
SWC - 121	Missing Protection against Signature Replay Attacks	✔️ **Passed**
SWC - 122	Lack of Proper Signature Verification	✔️ **Passed**
SWC - 123	Requirement Violation Passed	✔️ **Passed**
SWC - 124	Write to Arbitrary Storage Location	✔️ **Passed**
SWC - 125	Incorrect Inheritance Order Passed	✔️ **Passed**
SWC - 126	Insufficient Gas Griefing	✔️ **Passed**
SWC - 127	Arbitrary Jump with Function Type Variable	✔️ **Passed**
SWC - 128	DoS With Block Gas Limit	✔️ **Passed**
SWC - 129	Typographical Error	✔️ **Passed**
SWC - 130	Right-To-Left-Override control character (U+202E)	✔️ **Passed**
SWC - 131	Presence of unused variables	✔️ **Passed**
SWC - 132	Unexpected Ether balance	✔️ **Passed**
SWC - 133	Hash Collisions With Multiple Variable Arguments	✔️ **Passed**

SWC - 134	Message call with hardcoded gas amount	✔️ **Passed**
SWC - 135	Code With No Effects	✔️ **Passed**
SWC - 136	Unencrypted Private Data On-Chain	✔️ **Passed**

Manual Analysis

Note that using "tx.origin" as a security control might cause a situation where a user inadvertently authorizes a smart contract to perform an action on their behalf. It is recommended to use "msg.sender" instead.

The environment variable "block.number" looks like it might be used as a source of randomness. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Using "tx.origin" as a security control can lead to authorization bypass vulnerabilities. Consider using "msg.sender" unless you really know what you are doing.

Risk Status

Risk severity	Meaning
! Critical	None critical severity issues identified
! High	None high severity issues identified
! Medium	None medium severity issues identified
! Low	None low severity issues identified
Verified	17 functions and instances verified and checked
Safety Score	91 out of 100

Report Summary

Crypto Hub team has performed a line-by-line manual analysis and automated review of the smart contract. The smart contract was analyzed mainly for common smart contract vulnerabilities, exploits, and manipulation hacks.

Baby Tate smart contract source code has **LOW RISK SEVERITY.**
Baby Tate has **PASSED** the smart contract audit.

Note for stakeholders:

Be aware that active smart contract owner privileges constitute an elevated impact on smart contract's safety and security.
Make sure that the project team's KYC/identity is verified by an independent firm, e.g., Crypto Hub.
Always check if the contract's liquidity is locked. A longer liquidity lock plays an important role in the project's longevity. It is recommended to have multiple liquidity providers.
Examine the unlocked token supply in the owner, developer, or team's private wallets. Understand the project's tokenomics, and make sure the tokens outside of the LP Pair are vested or locked for a longer period of time.
Ensure that the project's official website is hosted on a trusted platform, and is using an active SSL certificate. The website's domain should be registered for a longer period of time.

Audit & KYC Certificates

We hereby certificate Baby Tate token smart contract as an audited project under the Crypto Hub enterprise umbrella. And to represent it as such we issued the following certificate:

Legal Advisory

Important Disclaimer

Crypto Hub provides contract auditing and project verification services for blockchain projects. The purpose of the audit is to analyze the on-chain smart contract source code, and to provide a basic overview of the project. This report should not be transmitted, disclosed, referred to, or relied upon by any person for any purposes without Crypto Hub prior written consent.

Crypto Hub provides the easy-to-understand assessment of the project, and the smart contract (otherwise known as the source code). The audit makes no statements or warranties on the security of the code. It also cannot be considered as an adequate assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have used all the data at our disposal to provide the transparent analysis, it is important to note that you should not rely on this report only — we recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts. Be aware that smart contracts deployed on a blockchain aren't resistant from external vulnerability, or a hack. Be aware that active smart contract owner privileges constitute an elevated impact to smart contract's safety and security. Therefore, Crypto Hub does not guarantee the explicit security of the audited smart contract.

The analysis of the security is purely based on the smart contracts alone. No applications or operations were reviewed for security. No product code has been reviewed.

This report should not be considered as an endorsement or disapproval of any project or team. The information provided in this report does not constitute investment advice, financial advice, trading advice, or any other sort of advice and you should not treat any of the report's content as such. Do conduct your own due diligence and consult your financial advisor before making any investment decisions.

About Crypto Hub

Crypto Hub provides intelligent blockchain solutions. Crypto Hub is developing an ecosystem that is seamless and responsive. Some of our services: Blockchain Security, Token Launchpad, NFT Marketplace, etc. Crypto Hub's mission is to interconnect multiple services like Blockchain Security, DeFi, Gaming, and Marketplace under one ecosystem that is seamless, multi-chain compatible, scalable, secure, fast, responsive, and easy-to-use.

Crypto Hub is built by a decentralized team of UI experts, contributors, engineers, and enthusiasts from all over the world. Our team currently consists of 3+ core team members, and 6+ casual contributors. Crypto Hub provides manual, static, and automatic smart contract analysis, to ensure that the project is checked against known attacks and potential vulnerabilities.