CERTIK

Preliminary Comments

sad baby token

Sept 29th, 2021

Table of Contents

Summary

This report has been prepared for sad baby token to discover issues and vulnerabilities in the source code of the sad baby token project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review and Static Analysis techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	sad baby token
Description	Deflationary BEP20 token
Platform	BSC
Language	Solidity
Codebase	https://bscscan.com/token/0x3ad405ef7aea80ccb41beef0a74510e18feef190
Commit	N/A

Audit Summary

Delivery Date	Sept 29, 2021
Audit Methodology	Manual Review, Static Analysis
Key Components	

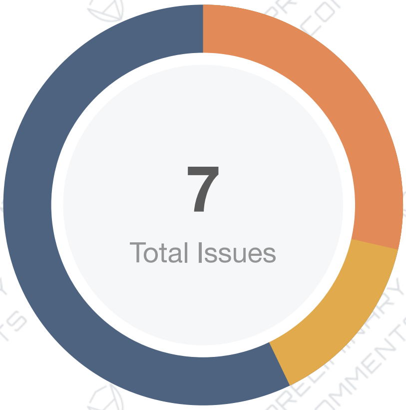
Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	2	2	0	0	0	0
Medium	1	1	0	0	0	0
Minor	0	0	0	0	0	0
Informational	4	4	0	0	0	0
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
CTC	CoinToken.sol	1a8347acf66ad0d1030b2852d38b34de92c748771e4b3931e0bc6c304085ad11

Findings



Critical	0 (0.00%)
Major	2 (28.57%)
Medium	1 (14.29%)
Minor	0 (0.00%)
Informational	4 (57.14%)
Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
CTC-01	Privileged Ownership	**Centralization / Privilege**	**Major**	Pending
CTC-02	Centralized Token Holding Position	**Centralization / Privilege**	**Major**	Pending
CTC-03	Variable `_rOwned[account]` Not Updated in Function `includeAccount()`	Control Flow	Medium	Pending
CTC-04	Redundant Code	Logical Issue	Informational	Pending
CTC-05	No Need to Use Library `SafeMath`	Language Specific	Informational	Pending
CTC-06	Missing Events Emitting	Coding Style	Informational	Pending
CTC-07	Potential Risks on Approve/TransferFrom Methods	Logical Issue	Informational	Pending

CTC-01 | Privileged Ownership

Category	Severity	Location	Status
Centralization / Privilege	Major	CoinToken.sol (2)	Pending

Description

Based on the record on chain, we could identify the owner as an EOA (externally owned account), and the owner address is 0x1158971539f52e2386Be22df6DAab80D79f15CB2.

In the contract `CoinToken`, the owner has the authority over the following functions:

- `excludeAccount()`
- `includeAccount()`
- `setAsCharityAccount()`
- `updateFee()`

The owner of the contract has significant privileges. For example, address `FeeAddress` can be set by the owner after contract deployment. The `_TAX_FEE`, `_BURN_FEE`, and `_CHARITY_FEE` each has a cap of 100% in the `updateFee()` function. Over time the `FeeAddress` wallet controlled by the owner will be distributed a significant amount of tokens. Any compromise to the owner or the `FeeAddress`'s private key may allow the hacker to take advantage of this, gaining access to funds and jeopardize the project.

Recommendation

We advise the client to carefully manage the owner's and `FeeAddress`'s private key to avoid any potential risks of being hacked. We recommend setting a reasonable cap on the `_TAX_FEE`, `_BURN_FEE`, and `_CHARITY_FEE`, as well as letting the community monitor the activities of the `FeeAddress` to ensure it is operating in accordance with the whitepaper.

In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;

- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

CERTIK PRELIMINARY COMMENTS

CTC-02 | Centralized Token Holding Position

Category	Severity	Location	Status
Centralization / Privilege	Major	CoinToken.sol (2): 496	Pending

Description

```
_rOwned[tokenOwner] = _rTotal;
```

All of the tokens are distributed to the contract owner when deploying the contract. And the owner can distribute tokens without obtaining the consensus of the community.

Recommendation

Once the token goes live, we assume many transactions would involve the wallet unlock of the owner address and the team shall make enough efforts to restrict the access of the private key.

CTC-03 | Variable `_rOwned[account]` Not Updated in Function `includeAccount()`

Category	Severity	Location	Status
Control Flow	Medium	CoinToken.sol (2): 604~615	Pending

Description

```
function includeAccount(address account) external onlyOwner() {
    require(_isExcluded[account], "Account is already included");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
            _tOwned[account] = 0;
            _isExcluded[account] = false;
            _excluded.pop();
            break;
        }
    }
}
```

The `_rOwned` variable is not updated within the function `includeAccount()`. This could cause a potential discrepancy in account balance if an address is excluded, and subsequently included again. When an account is excluded, the return value of the `_getRate()` function could change. If the `_rOwned[account]` value is not updated on inclusion, `balanceOf(account)` could change as a result. This appears to be inconsistent with the business logic, as an account balance shouldn't change during the period of exclusion. Theoretically, the `_owner` is able to exclude its address or any arbitrary address and later include the same address, siphoning tokens from other holders.

A detailed explanation and example are shown in the link below:
https://perafinance.medium.com/safemoon-is-it-safe-though-a-detailed-explanation-of-frictionless-yield-bug-338710649846

Recommendation

We recommend recalculating the value of `_rOwned[account]` in function `includeAccount()`.

```
function includeAccount(address account) external onlyOwner() {
    require(_isExcluded[account], "Account is already included");
    for (uint256 i = 0; i < _excluded.length; i++) {
        if (_excluded[i] == account) {
            _excluded[i] = _excluded[_excluded.length - 1];
```

```
                _rOwned[account] = _tOwned[account].mul(_getRate());    //recalculate
_rOwned
                _tOwned[account] = 0;
                _isExcluded[account] = false;
                _excluded.pop();
                break;
            }
        }
    }
```

CTC-04 | Redundant Code

Category	Severity	Location	Status
Logical Issue	● Informational	CoinToken.sol (2): 658~668	ⓘ Pending

Description

A code snippet in function `_transfer()`:

```
        if (_isExcluded[sender] && !_isExcluded[recipient]) {
            _transferFromExcluded(sender, recipient, amount);
        } else if (!_isExcluded[sender] && _isExcluded[recipient]) {
            _transferToExcluded(sender, recipient, amount);
        } else if (!_isExcluded[sender] && !_isExcluded[recipient]) {
            _transferStandard(sender, recipient, amount);
        } else if (_isExcluded[sender] && _isExcluded[recipient]) {
            _transferBothExcluded(sender, recipient, amount);
        } else {
            _transferStandard(sender, recipient, amount);
        }
```

The condition `!_isExcluded[sender] && !_isExcluded[recipient]` can be included in `else` .

Recommendation

We recommend removing the following code:

```
        else if (!_isExcluded[sender] && !_isExcluded[recipient]) {
            _transferStandard(sender, recipient, amount);
        }
```

CTC-05 | No Need to Use Library `SafeMath`

Category	Severity	Location	Status
Language Specific	● Informational	CoinToken.sol (2): 108~249, 446	ⓘ Pending

Description

Solidity v0.8.0 and later versions check underflow/overflow by default, and therefore the library `SafeMath` is not necessary.

Source: link

Recommendation

We recommend using the default arithmetic check instead of the library `SafeMath`.

CTC-06 | Missing Events Emitting

Category	Severity	Location	Status
Coding Style	Informational	CoinToken.sol (2)	Pending

Description

When a function affects the status of sensitive variables, it should be able to emit events to notify users of the contract. The following functions in contract `CoinToken` should emit events:

- `excludeAccount()`
- `includeAccount()`
- `setAsCharityAccount()`
- `updateFee()`

Recommendation

We recommend adding events for the sensitive actions of the above functions and emitting them accordingly.

CTC-07 | Potential Risks on Approve/TransferFrom Methods

Category	Severity	Location	Status
Logical Issue	● Informational	CoinToken.sol (2): 531~534	ⓘ Pending

Description

```
function approve(address spender, uint256 amount) public override returns (bool) {
    _approve(_msgSender(), spender, amount);
    return true;
}
```

The function `approve()` could be used in a front-running attack that allows a spender to transfer more tokens than the owner of the tokens ever wanted to allow the spender to transfer.

Here is a possible attack scenario:

Alice allows Bob to transfer N of Alice's tokens (N>0) by calling approve method on a Token smart contract passing Bob's address and N as method arguments. After some time, Alice decides to change from N to M (M>0) number of her tokens that Bob is allowed to transfer, so she calls the `approve()` method again, this time passing Bob's address and M as method arguments. Bob notices Alice's second transaction before it is mined and quickly sends another transaction that calls `transferFrom()` method to transfer N Alice's tokens to somewhere. If Bob's transaction is executed before Alice's, Bob will successfully transfer N Alice's tokens and gain the ability to transfer another M tokens.

So, Alice's attempt to change Bob's allowance from N to M (N>0 and M>0) made it possible for Bob to transfer N+M of Alice's tokens, while Alice never wanted to allow so many of her tokens to be transferred by Bob.

Recommendation

We recommend using functions `increaseAllowance()` and `decreaseAllowance()` instead.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you ("Customer" or the "Company") in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK's prior written consent in each instance.

This report is not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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. CertiK's position is that each company and individual are responsible for their own due diligence and continuous security. CertiK's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED "AS IS" AND "AS

AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING

MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

CERTIK