



Security Assessment

BabyCake

Jul 31st, 2021

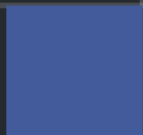


Table of Contents

Summary

Overview

Project Summary

Audit Summary

Vulnerability Summary

Audit Scope

Findings

BAB-01 : Gas optimization in function ``_transfer()``

BAB-02 : Privileged Ownership In ``BABYCAKEDividendTracker``

BAB-03 : Privileged Ownership In ``BABYCAKE``

BAB-04 : Proper Usage of “public” and “external” type

BAB-05 : Unchecked Value of ERC-20 ``transfer()`` Call

BAB-06 : Contract gains non-withdrawable ETH via the ``swapAndLiquify`` function

BAB-07 : ``Require`` Statement Never Pass

BAB-08 : ``Require`` Statement Never Pass in Function ``withdrawDividend``

BAB-09 : Redundant Codes

BAB-10 : Extra gas cost

BAB-11 : Nothing to do in ``catch`` statement

BAB-12 : Centralized risk in ``swapAndSendToFee``

DPT-01 : ``Require`` Statement Never Pass

DPT-02 : Lack Of Error Message

DPT-03 : Mismatch Between Comment and Code

DPT-04 : Redundant Codes

Appendix

Disclaimer

About

Summary

This report has been prepared for BabyCake to discover issues and vulnerabilities in the source code of the BabyCake project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review 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	BabyCake
Platform	BSC
Language	Solidity
Codebase	https://bscscan.com/address/0xdb8d30b74bf098af214e862c90e647bbb1fcc58c#code
Commit	

Audit Summary

Delivery Date	Jul 31, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

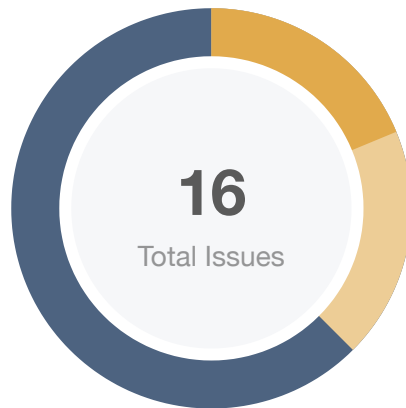
Vulnerability Level	Total	Pending	Partially Resolved	Resolved	Acknowledged	Declined
● Critical	0	0	0	0	0	0
● Major	0	0	0	0	0	0
● Medium	3	0	0	0	3	0
● Minor	3	0	0	0	3	0
● Informational	10	0	0	4	6	0
● Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
BAB	BABYCAKE.sol	bbc886ddc83bd475bc69d8883a75ca6036e879a0a9b9311c09f120b13e987d04
CCK	Context.sol	ade730fe55d7b995a6a9a81f77600d10d9ea7472be54a290a905f853495bce97
DPT	DividendPayingToken.sol	e52de8a5a6d9ce8924e443ef947ea0959145b290779fb40d6155c0858f930282
DPI	DividendPayingTokenInterface.sol	7d7301f0a6321c9a83e2544342327c9eeaffd7476424e60f2f8badd302c94053
DPO	DividendPayingTokenOptionalInterface.sol	613ef8cfcd377b92e0a456548c6560ee3dd18d9a253850fcd6b9036337feb6b
ERC	ERC20.sol	c9352c9260d5c9261d5c5449cb864887720a316ca241020d5c8a2ae0c841fb0
IER	IERC20.sol	40b62888fbeb089db2a8060f52214b2aba38abd295b9c6b90fbc8b52ba5158b6
IEC	IERC20Metadata.sol	5453d34cc9db3921a16eb83a551e1a9285d9285806c119d5caf399352f6bf1a6
IUV	IUniswapV2Factory.sol	cfac3b608fe9c5c10db6e7dfbd0e52600689b41848cbb7c3f6d074ebca8b545f
IUP	IUniswapV2Pair.sol	522717b02bc1839e9024e49d6d93ebb976be01a5d0b7e459c3d50fd3dbfe6cb2
IUR	IUniswapV2Router.sol	ac1b9a6719ad80130195805ec526188b3dd3a84ddd5b8e6ac92169abfa415a04
IMC	IterableMapping.sol	4e1661030209caf939a716c3dbc413f704fc7b6d6cdb5a957f1f48e693d30d23
OCK	Ownable.sol	fb7658fc325cceffba19f6cf9809119bf073d1d4ccccbbf9d439a34ff062934c
SMC	SafeMath.sol	253b3928dd6338470c3cc18945de79fa9ec77b12a36948aa36ae3e5771851fba

ID	File	SHA256 Checksum
SMI	SafeMathInt.sol	9345ec14af97a2ed2238153d853257bcd209a8d4de6de8cb1152711bc94402bd
SMU	SafeMathUint.sol	87eae8174207cfb48ac338ad99eeeeee1e989ec141144b3b30a2ef90e08d8fb9f

Findings



■ Critical	0 (0.00%)
■ Major	0 (0.00%)
■ Medium	3 (18.75%)
■ Minor	3 (18.75%)
■ Informational	10 (62.50%)
■ Discussion	0 (0.00%)

ID	Title	Category	Severity	Status
BAB-01	Gas optimization in function <code>_transfer()</code>	Gas Optimization	● Informational	ⓘ Acknowledged
BAB-02	Privileged Ownership In <code>BABYCAKEDividendTracker</code>	Centralization / Privilege	● Medium	ⓘ Acknowledged
BAB-03	Privileged Ownership In <code>BABYCAKE</code>	Centralization / Privilege	● Medium	ⓘ Acknowledged
BAB-04	Proper Usage of “public” and “external” type	Gas Optimization	● Informational	ⓘ Acknowledged
BAB-05	Unchecked Value of ERC-20 <code>transfer()</code> Call	Volatile Code	● Minor	ⓘ Acknowledged
BAB-06	Contract gains non-withdrawable ETH via the <code>swapAndLiquify</code> function	Logical Issue	● Minor	ⓘ Acknowledged
BAB-07	<code>Require</code> Statement Never Pass	Logical Issue	● Informational	ⓧ Resolved
BAB-08	<code>Require</code> Statement Never Pass in Function <code>withdrawDividend</code>	Language Specific	● Informational	ⓧ Resolved
BAB-09	Redundant Codes	Logical Issue	● Informational	ⓘ Acknowledged
BAB-10	Extra gas cost	Gas Optimization	● Informational	ⓧ Resolved
BAB-11	Nothing to do in <code>catch</code> statement	Logical Issue	● Minor	ⓘ Acknowledged
BAB-12	Centralized risk in <code>swapAndSendToFee</code>	Centralization / Privilege	● Medium	ⓘ Acknowledged
DPT-01	<code>Require</code> Statement Never Pass	Logical Issue	● Informational	ⓧ Resolved

ID	Title	Category	Severity	Status
DPT-02	Lack Of Error Message	Volatile Code	● Informational	① Acknowledged
DPT-03	Mismatch Between Comment and Code	Coding Style	● Informational	① Acknowledged
DPT-04	Redundant Codes	Logical Issue	● Informational	① Acknowledged

BAB-01 | Gas optimization in function `_transfer()`

Category	Severity	Location	Status
Gas Optimization	● Informational	BABYCAKE.sol: 349	ⓘ Acknowledged

Description

The calculation `fees += amount.mul(1).div(100);` can be optimized as below:

```
fees = fees.add(amount.div(100));
```

Recommendation

We recommend optimizing the code to save gas.

Alleviation

The development team responded that the purpose is that they are taking only 1 percent of the amount as a fee.

BAB-02 | Privileged Ownership In `BABYCAKEDividendTracker`

Category	Severity	Location	Status
Centralization / Privilege	● Medium	BABYCAKE.sol: 475	📄 Acknowledged

Description

The owner of contract `BABYCAKEDividendTracker` has the permission to call functions

1. `excludeFromDividends()`
2. `updateClaimWait()`
3. `setBalance()`
4. `processAccount()`
5. `distributeCAKEDividends()`

without obtaining the consensus of the community.

Recommendation

Renounce ownership when it is the right timing, or gradually migrate to a timelock plus multisig governing procedure and let the community monitor in respect of transparency considerations.

Alleviation

The development team responded that these functions are needed for managing this smart contract and the admin account is multi sig wallet.

BAB-03 | Privileged Ownership In `BABYCAKE`

Category	Severity	Location	Status
Centralization / Privilege	● Medium	BABYCAKE.sol: 28	ⓘ Acknowledged

Description

The owner of contract `BABYCAKE` has the permission to call functions

1. `updateDividendTracker()`
2. `updateUniswapV2Router()`
3. `excludeFromFees()`
4. `excludeMultipleAccountsFromFees()`
5. `setMarketingWallet()`
6. `setCAKERewardsFee()`
7. `setLiquiditFee()`
8. `setMarketingFee()`
9. `setAutomatedMarketMakerPair()`
10. `blacklistAddress()`
11. `excludeFromDividends()`

without obtaining the consensus of the community.

Recommendation

Renounce ownership when it is the right timing, or gradually migrate to a timelock plus multisig governing procedure and let the community monitor in respect of transparency considerations.

Alleviation

The development team responded that these functions are needed for managing this smart contract and the admin account is multi sig wallet.

BAB-04 | Proper Usage of “public” and “external” type

Category	Severity	Location	Status
Gas Optimization	● Informational	BABYCAKE.sol: 169	ⓘ Acknowledged

Description

`public` functions that are never called by the contract could be declared `external`. When the inputs are arrays, `external` functions are more efficient than `public` functions.

Recommendation

We advise the client to use the external attribute for functions never called from the contract.

Alleviation

The development team responded that they will try to update in next release.

BAB-05 | Unchecked Value of ERC-20 `transfer()` Call

Category	Severity	Location	Status
Volatile Code	● Minor	BABYCAKE.sol: 379	ⓘ Acknowledged

Description

"The linked `transfer()` invocations do not check the return value of the function call which should yield a `true` result in case of proper ERC-20 implementation.

The aforementioned lines perform the external call to `transfer` of ERC20 contracts and the return value is not checked in either case. "

Recommendation

"As many tokens do not follow the ERC-20 standard faithfully, they may not return a `bool` variable in this function's execution meaning that simply expecting it can cause incompatibility with these types of tokens. Instead, we advise that OpenZeppelin's `SafeERC20.sol` implementation is utilized for interacting with the `transfer()` and `transferFrom()` functions of ERC-20 tokens. The OZ implementation optionally checks for a return value rendering compatible with all ERC-20 token implementations.

It is recommended to use SafeERC20 or make sure that the value returned from 'transfer()' is checked."

Alleviation

The development team responded that they will try to update in next release.

BAB-06 | Contract gains non-withdrawable ETH via the `swapAndLiquify` function

Category	Severity	Location	Status
Logical Issue	● Minor	BABYCAKE.sol: 382	ⓘ Acknowledged

Description

The `swapAndLiquify` function converts half of the BABYCAKE tokens to ETH. The other half of BABYCAKE tokens and part of the converted ETH are deposited into the LP pool on Uniswap as liquidity. For every `swapAndLiquify` function call, a small amount of ETH leftover in the contract. This is because the price of BABYCAKE drops after swapping the first half of BABYCAKE tokens into ETHs, and the other half of BABYCAKE tokens require less than the converted ETH to be paired with it when adding liquidity. The contract doesn't appear to provide a way to withdraw those ETH, and they will be locked in the contract forever.

Recommendation

It's not ideal that more and more ETH are locked into the contract over time. The simplest solution is to add a `withdraw` function in the contract to withdraw ETH. Other approaches that benefit the BABYCAKE token holders can be:

- Distribute ETH to BABYCAKE token holders proportional to the amount of token they hold.
- Use leftover ETH to buy back BABYCAKE tokens from the market to increase the price of BABYCAKE.

Alleviation

The development team responded that they will try to update in next release.

BAB-07 | Require Statement Never Pass

Category	Severity	Location	Status
Logical Issue	● Informational	BABYCAKE.sol: 501	☑ Resolved

Description

The code `require(false)` in the function `_transfer` will always fail. Is that designed as expected?

Alleviation

[BabyCake team]: To prevent the sending our dividend tokens and notify users before they transfer.

BAB-08 | `require` Statement Never Pass in Function `withdrawDividend`

Category	Severity	Location	Status
Language Specific	● Informational	BABYCAKE.sol: 505	🟢 Resolved

Description

The code `require(false)` in the function `withdrawDividend` will always fail.

Recommendation

Consider apply the `revert` statement.

Alleviation

[BabyCake team]: This is only for prevent users to withdraw DividendTracker token. And show the error message when they are trying to do that through metamask or sth. No need to revert in this case cause we didn't made any transaction.

BAB-09 | Redundant Codes

Category	Severity	Location	Status
Logical Issue	● Informational	BABYCAKE.sol: 671, 650, 607, 177	ⓘ Acknowledged

Description

There is no need to declare an address as `payable` if there is no native token transfer on it.

Alleviation

The development team responded that they will try to update in next release.

BAB-10 | Extra gas cost

Category	Severity	Location	Status
Gas Optimization	● Informational	BABYCAKE.sol: 361~370	🕒 Resolved

Description

There is an extra gas cost in each transfer caused by the extra logic of the distributing of dividend. Is that designed as expected?

Alleviation

[BabyCake team]: Yup, it's what we designed so far. For now we distribute CAKE token automatically to the users.

BAB-11 | Nothing to do in `catch` statement

Category	Severity	Location	Status
Logical Issue	● Minor	BABYCAKE.sol: 358~359, 367~369	ⓘ Acknowledged

Description

The fail case in try/catch statement is ignored.

Alleviation

No Alleviation.

BAB-12 | Centralized risk in `swapAndSendToFee`

Category	Severity	Location	Status
Centralization / Privilege	● Medium	BABYCAKE.sol: 373~380	① Acknowledged

Description

```
1 // add the liquidity
2     function swapAndSendToFee(uint256 tokens) private {
3
4         uint256 initialCAKEBalance = IERC20(CAKE).balanceOf(address(this));
5
6         swapTokensForCake(tokens);
7         uint256 newBalance =
8     (IERC20(CAKE).balanceOf(address(this))).sub(initialCAKEBalance);
9         IERC20(CAKE).transfer(_marketingWalletAddress, newBalance);
10    }
```

The `swapAndSendToFee` function calls the `IERC20(CAKE).transfer` function with the `to` address specified as `_marketingWalletAddress` for acquiring the generated CAKE tokens after the calling of `swapTokensForCake`. As a result, over time the `_marketingWalletAddress` address will accumulate a significant portion of CAKE tokens. If the `_marketingWalletAddress` is an EOA (Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

Recommendation

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

Indicatively, here are some feasible solutions that would also mitigate the potential risk:

- Time-lock with reasonable latency, i.e. 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent single point of failure due to the private key;
- Introduction of a DAO / governance / voting module to increase transparency and user involvement.

Alleviation

[BabyCake team]: we are using multi signature wallet address for `_marketingWalletAddress`.

DPT-01 | **Require** Statement Never Pass

Category	Severity	Location	Status
Logical Issue	● Informational	DividendPayingToken.sol: 133	🟢 Resolved

Description

The code `require(false)` in the function `_transfer` will never pass.

Recommendation

Consider refactoring the code.

Alleviation

The development team responded that it is to prevent the sending dividend tokens and notify users before they transfer.

DPT-02 | Lack Of Error Message

Category	Severity	Location	Status
Volatile Code	● Informational	DividendPayingToken.sol: 56	ⓘ Acknowledged

Description

Lack of error messages in the function `distributeCAKEDividends` makes it difficult for users to understand.

Alleviation

The development team responded that they will add more description.

DPT-03 | Mismatch Between Comment and Code

Category	Severity	Location	Status
Coding Style	● Informational	DividendPayingToken.sol: 74, 81	ⓘ Acknowledged

Description

The comments of the function and its codes are dis-match, the comments described that the function would deal with ether, but its implementation implies that the function deals with the `CAKE`.

Alleviation

The development team responded that they will update the comment in next release.

DPT-04 | Redundant Codes

Category	Severity	Location	Status
Logical Issue	● Informational	DividendPayingToken.sol: 76	ⓘ Acknowledged

Description

There is no need to declare an address as `payable` if there is no native token transfer on it.

Alleviation

The development team responded that they will try to update in next release.

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.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

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.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

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 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 without CertiK's prior written consent.

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.

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.

