CERTIK

Security Assessment

SHIBSC

May 11th, 2021

Summary

This report has been prepared for SHIBSC smart contracts, to discover issues and vulnerabilities in the source code of their Smart Contract 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 given they are currently missing in the repository;
- Provide more comments per each function for readability, especially contracts are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	SHIBSC
Platform	BSC
Language	Solidity
Codebase	https://github.com/Shiba-bsc/Shiba-bsc-core/tree/audit
Commits	7f04c42050a383156fe46218ed02e0867f39df39

Audit Summary

Delivery Date	May 11, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

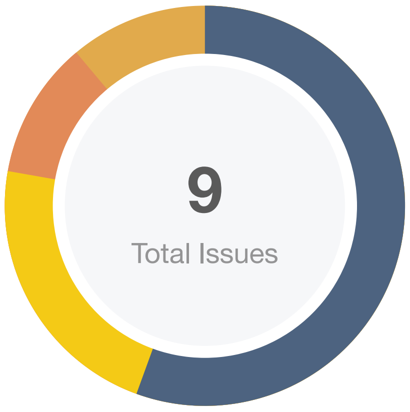
Vulnerability Summary

Total Issues	9
● Critical	0
● Major	1
● Medium	1
● Minor	2
● Informational	5
● Discussion	0

Audit Scope

ID	file	SHA256 Checksum
SHI	contracts/SHIBSC.sol	0137528e723c9d84f35b3b62975dcc49c051d61b8d0e09c17165652ef21625e6

Findings



🟥	**Critical**	**0** (0.00%)
🟧	**Major**	**1** (11.11%)
🟧	**Medium**	**1** (11.11%)
🟨	**Minor**	**2** (22.22%)
🟦	**Informational**	**5** (55.56%)
🟩	**Discussion**	**0** (0.00%)

ID	Title	Category	Severity	Status
SHI-01	Contract gains non-withdrawable BNB via the `swapAndLiquify` function	Logical Issue	● Medium	ⓘ Acknowledged
SHI-02	Centralized risk in `addLiquidity`	**Centralization / Privilege**	● **Major**	**Partially Resolved**
SHI-03	Variable could be declared as `constant`	Gas Optimization	● Informational	ⓘ Acknowledged
SHI-04	Return value not handled	Volatile Code	● Informational	ⓘ Acknowledged
SHI-05	3rd party dependencies	Control Flow	● Minor	ⓘ Acknowledged
SHI-06	Missing event emitting	Coding Style	● Informational	ⓘ Acknowledged
SHI-07	Privileged ownership	**Centralization / Privilege**	● **Minor**	**Partially Resolved**
SHI-08	Typos in the contract	Coding Style	● Informational	ⓘ Acknowledged
SHI-09	Function and variable naming doesn't match the operating environment	Coding Style	● Informational	ⓘ Acknowledged

SHI-01 | Contract gains non-withdrawable BNB via the `swapAndLiquify` function

Category	Severity	Location	Status
Logical Issue	● Medium	contracts/SHIBSC.sol: 343~357	ⓘ Acknowledged

Description

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

Recommendation

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

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

Alleviation

[SHIBSC Team]: Yes, technically, tha't possibol to use both reserves to calculate the perfect amount to add liquidity without any remainings. However, that's sophisticated and may lead to bug. Besides it causes more gas. There indeed is little bnb left, but it is tolerable

SHI-02 | Centralized risk in `addLiquidity`

Category	Severity	Location	Status
Centralization / Privilege	● Major	contracts/SHIBSC.sol: 375~386	Partially Resolved

Description

```
    function addLiquidity(uint256 tokenAmount, uint256 ethAmount) private {
        _approve(address(this), address(uniswapV2Router), tokenAmount);

        uniswapV2Router.addLiquidityETH{value : ethAmount}(
            address(this),
            tokenAmount,
            0,
            0,
            owner(),
            block.timestamp
        );
    }
```

The `addLiquidity` function calls the `uniswapV2Router.addLiquidityETH` function with the `to` address specified as `owner()` for acquiring the generated LP tokens from the `SHIBSC-BNB` pool. As a result, over time the `_owner` address will accumulate a significant portion of LP tokens.If the `_owner` is an EOA (Externally Owned Account), mishandling of its private key can have devastating consequences to the project as a whole.

Recommendation

We advise the `to` address of the `uniswapV2Router.addLiquidityETH` function call to be replaced by the contract itself, i.e. `address(this)`, and to restrict the management of the LP tokens within the scope of the contract's business logic. This will also protect the LP tokens from being stolen if the `_owner` account is compromised. 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

[SHIBSC Team]: Yes, those part of tokens shall then go to DAO or governance or boardroom. If community decides to burn them, then the tokens will be sent to blackhold address.

SHI-03 | Variable could be declared as `constant`

Category	Severity	Location	Status
Gas Optimization	● Informational	contracts/SHIBSC.sol: 30~32, 26, 47	ⓘ Acknowledged

Description

Variables `_tTotal`, `numTokensSellToAddToLiquidity`, `_name`, `_symbol` and `_decimals` could be declared as `constant` since these state variables are never to be changed.

Recommendation

We recommend declaring those variables as `constant`.

Alleviation

No alleviation.

SHI-04 | Return value not handled

Category	Severity	Location	Status
Volatile Code	● Informational	contracts/SHIBSC.sol: 378~385	ⓘ Acknowledged

Description

The return values of function `addLiquidityETH` are not properly handled.

```
        uniswapV2Router.addLiquidityETH{value : ethAmount}(
            address(this),
            tokenAmount,
            0,
            0,
            owner(),
            block.timestamp
        );
```

Recommendation

We recommend using variables to receive the return value of the functions mentioned above and handle both success and failure cases if needed by the business logic.

Alleviation

No alleviation.

SHI-05 | 3rd party dependencies

Category	Severity	Location	Status
Control Flow	● Minor	contracts/SHIBSC.sol: 8~10	ⓘ Acknowledged

Description

The contract is serving as the underlying entity to interact with third party PancakeSwap protocols. The scope of the audit would treat those 3rd party entities as black boxes and assume its functional correctness. However in the real world, 3rd parties may be compromised that led to assets lost or stolen.

Recommendation

We understand that the business logic of the SHIBSC protocol requires the interaction PancakeSwap protocol for adding liquidity to SHIBSC-BNB pool and swap tokens. We encourage the team to constantly monitor the statuses of those 3rd parties to mitigate the side effects when unexpected activities are observed.

Alleviation

[SHIBSC Team]: Yes, we will check Pancake's logic and it shall fit to more dex protocol. So we leave some space here.

SHI-06 | Missing event emitting

Category	Severity	Location	Status
Coding Style	● Informational	contracts/SHIBSC.sol: 12	ⓘ Acknowledged

Description

In contract `SHIBSC`, there are a bunch of functions can change state variables. However, these function do not emit event to pass the changes out of chain.

Recommendation

Recommend emitting events, for all the essential state variables that are possible to be changed during runtime.

Alleviation

No alleviation.

SHI-07 | Privileged ownership

Category	Severity	Location	Status
Centralization / Privilege	● **Minor**	contracts/SHIBSC.sol: 12	**Partially Resolved**

Description

The owner of contract `SHIBSC` has the permission to:

1. change the address that can receive LP tokens,
2. lock the contract,
3. exclude/include addresses from rewards/fees,
4. set `taxFee`, `liquidityFee` and `_maxTxAmount`,
5. enable `swapAndLiquifyEnabled`

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

[SHIBSC Team]: Yes, we leave those elastical configs to DAO or governance or boardroom.

SHI-08 | Typos in the contract

Category	Severity	Location	Status
Coding Style	● Informational	contracts/SHIBSC.sol: 54	ⓘ Acknowledged

Description

There is a typo in the code.

1. In the following code snippet, `tokensIntoLiqudity` should be `tokensIntoLiquidity`.

```
event SwapAndLiquify(
        uint256 tokensSwapped,
        uint256 ethReceived,
        uint256 tokensIntoLiqudity
    );
```

Recommendation

We recommend correcting the typo in the contract.

Alleviation

No alleviation.

SHI-09 | Function and variable naming doesn't match the operating environment

Category	Severity	Location	Status
Coding Style	● Informational	contracts/SHIBSC.sol: 359~373	ⓘ Acknowledged

Description

The SHIBSC contract uses Pancakeswap for swapping and adds liquidity to Pancakeswap pool, but naming it Uniswap. Function swapTokensForEth(uint256 tokenAmount) swaps SHIBSC token for BNB instead of ETH.

Recommendation

Change "Uniswap" and "ETH" to "Pancakeswap" and "BNB" in the contract respectively to match the operating environment and avoid confusion.

Alleviation

No alleviation.

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.

Mathematical Operations

Mathematical Operation findings relate to mishandling of math formulas, such as overflows, incorrect operations etc.

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.

Volatile Code

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

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a struct assignment operation affecting an in-memory struct rather than an in-storage one.

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.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as constant contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

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 string 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.