Security Audit

Shiba Classic (Token)

December 2024 info@hashlock.com.au Hashlock Pty Ltd

Table of Contents

CAUTION

THIS DOCUMENT IS A SECURITY AUDIT REPORT AND MAY CONTAIN CONFIDENTIAL INFORMATION. THIS INCLUDES IDENTIFIED VULNERABILITIES AND MALICIOUS CODE WHICH COULD BE USED TO COMPROMISE THE PROJECT. THIS DOCUMENT SHOULD ONLY BE FOR INTERNAL USE UNTIL ISSUES ARE RESOLVED. ONCE VULNERABILITIES ARE REMEDIATED, THIS REPORT CAN BE MADE PUBLIC. THE CONTENT OF THIS REPORT IS OWNED BY HASHLOCK PTY LTD FOR USE OF THE CLIENT.

Executive Summary

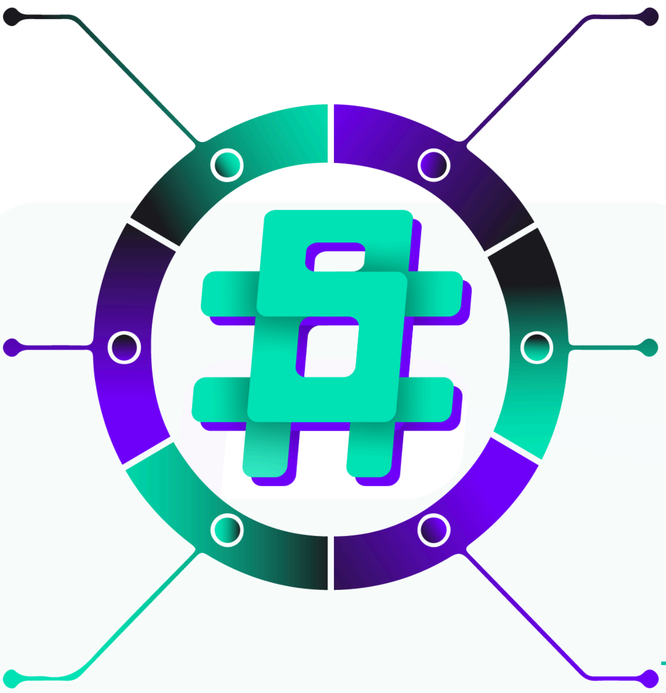
The Shiba Classic team partnered with Hashlock to conduct a security audit of their SHIBC.sol smart contract. Hashlock manually and proactively reviewed the code in order to ensure the project's team and community that the deployed contracts are secure.

Project Context

Shiba Classic ($SHIBC) is a decentralized, community-driven cryptocurrency that revitalizes the pioneering spirit of the original Doge killer. Drawing inspiration from the iconic Shiba Inu logo of 2021, $SHIBC embodies a collective commitment to decentralization and innovation within the Web3 ecosystem. With renounced ownership and locked liquidity, it ensures transparency and security for its holders. By integrating features such as governance, staking, and NFTs, Shiba Classic transcends its meme token origins, offering real-world applications and long-term utility. Join us in reviving the essence of the Shiba phenomenon and be part of a community united by a shared vision for the future of decentralized finance.

Project Name: Shiba Classic
Project Type: Token
Compiler Version: 0.8.28
Website: www.shibaclassic.io
Logo:

Visualised Context:



Project Name

Shiba Classic

Launch Date

13th Nov. 2024

Compiler Version

v0.8.28

Language

Solidity

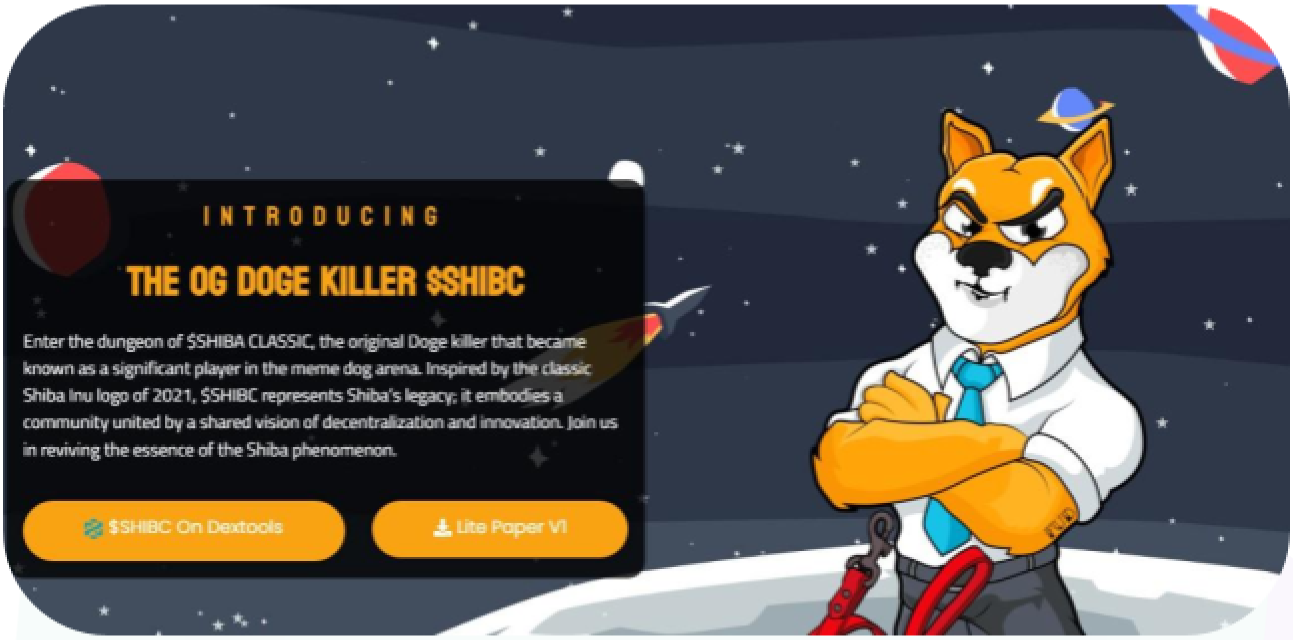
Network

ETHEREUM

Token Ticker

SHIBC

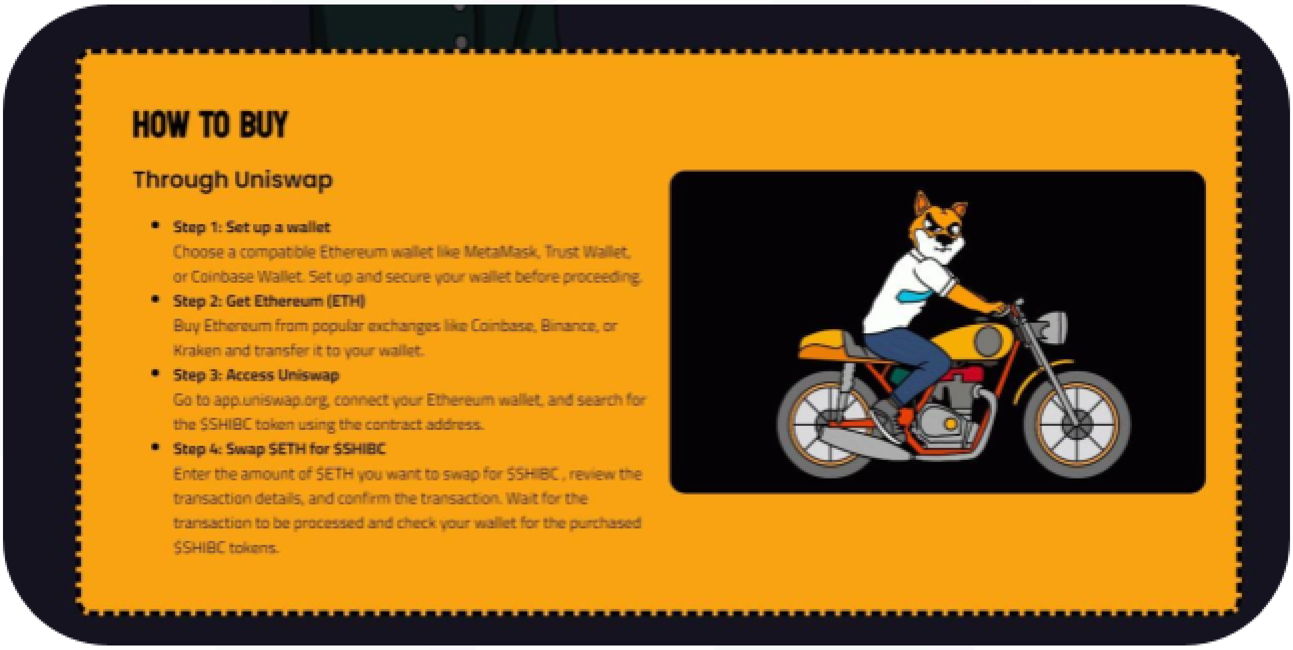
Project Visuals:



INTRODUCING

THE OG DOGE KILLER $SHIBC

Enter the dungeon of $SHIBA CLASSIC, the original Doge killer that became known as a significant player in the meme dog arena. Inspired by the classic Shiba Inu logo of 2021, $SHIBC represents Shiba's legacy; it embodies a community united by a shared vision of decentralization and innovation. Join us in reviving the essence of the Shiba phenomenon.



HOW TO BUY

Through Uniswap

- **Step 1: Set up a wallet**
  Choose a compatible Ethereum wallet like MetaMask, Trust Wallet, or Coinbase Wallet. Set up and secure your wallet before proceeding.
- **Step 2: Get Ethereum (ETH)**
  Buy Ethereum from popular exchanges like Coinbase, Binance, or Kraken and transfer it to your wallet.
- **Step 3: Access Uniswap**
  Go to app.uniswap.org, connect your Ethereum wallet, and search for the $SHIBC token using the contract address.
- **Step 4: Swap $ETH for $SHIBC**
  Enter the amount of $ETH you want to swap for $SHIBC , review the transaction details, and confirm the transaction. Wait for the transaction to be processed and check your wallet for the purchased $SHIBC tokens.

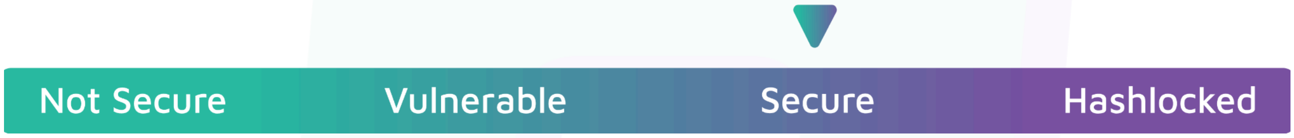
Audit scope

We at Hashlock audited the solidity code within the Shiba Classic project, the scope of work included a comprehensive review of the smart contracts listed below. We tested the smart contracts to check for their security and efficiency. These tests were undertaken primarily through manual line-by-line analysis and were supported by software-assisted testing.

Description	**Shiba Classic Smart Contract**
Platform	**Ethereum / Solidity**
Audit Date	**December, 2024**
Contract	SHIBC.sol
Contract MD5 Hash	4847ad67c490a51f082f8684c9232c45
Contract Address	0x9562e2063122eaa4d7c2d786e7ca2610d70ca8b8

Security Rating

After Hashlock’s Audit, we found the smart contracts to be **“Secure”**. The contracts all follow simple logic, with correct and detailed ordering. They use a series of interfaces, and the protocol uses a list of Open Zeppelin contracts. We initially identified some significant vulnerabilities that have since been addressed.



The ‘Hashlocked’ rating is reserved for projects that ensure ongoing security via bug bounty programs or on chain monitoring technology.

All issues uncovered during automated and manual analysis were meticulously reviewed and applicable vulnerabilities are presented in the Audit Findings section. The general security overview is presented in the Standardised Checks section and the project's contract functionality is presented in the Intended Smart Contract Functions section.

All vulnerabilities initially identified have now been resolved or acknowledged.

Hashlock found:

1 Low severity vulnerability

1 Gas Optimisation

4 QAs

Caution: *Hashlock’s audits do not guarantee a project's success or ethics, and are not liable or responsible for security. Always conduct independent research about any project before interacting.*

Intended Smart Contract Functions

Claimed Behaviour	Actual Behaviour
SHIBC.sol An ERC20 token with commission logic for buying and selling	**Contract achieves this functionality.**

Code Quality

This audit scope involves the smart contracts of the Shiba Classic project, as outlined in the Audit Scope section. All contracts, libraries, and interfaces mostly follow standard best practices and to help avoid unnecessary complexity that increases the likelihood of exploitation, however, some refactoring was required.

The code is very well commented on and closely follows best practice nat-spec styling. All comments are correctly aligned with code functionality.

Audit Resources

We were given the Shiba Classic project smart contract code in the form of Contract address.

As mentioned above, code parts are well commented. The logic is straightforward, and therefore it is easy to quickly comprehend the programming flow as well as the complex code logic. The comments are helpful in providing an understanding of the protocol's overall architecture.

Dependencies

As per our observation, the libraries used in this smart contracts infrastructure are based on well-known industry standard open source projects.

Severity Definitions

The severity levels assigned to findings represent a comprehensive evaluation of both their potential impact and the likelihood of occurrence within the system. These categorizations are established based on Hashlock's professional standards and expertise, incorporating both industry best practices and our discretion as security auditors. This ensures a tailored assessment that reflects the specific context and risk profile of each finding.

Significance	Description
High	High-severity vulnerabilities can result in loss of funds, asset loss, access denial, and other critical issues that will result in the direct loss of funds and control by the owners and community.
Medium	Medium-level difficulties should be solved before deployment, but won't result in loss of funds.
Low	Low-level vulnerabilities are areas that lack best practices that may cause small complications in the future.
Gas	Gas Optimisations, issues, and inefficiencies
QA	Quality Assurance (QA) findings are informational and don't impact functionality. Supports clients improve the clarity, maintainability, or overall structure of the code.

Status Definitions

Each identified security finding is assigned a status that reflects its current stage of remediation or acknowledgment. The status provides clarity on the handling of the issue and ensures transparency in the auditing process. The statuses are as follows:

Significance	Description
Resolved	The identified vulnerability has been fully mitigated either through the implementation of the recommended solution proposed by Hashlock or through an alternative client-provided solution that demonstrably addresses the issue
Acknowledged	The client has formally recognized the vulnerability but has chosen not to address it due to the high cost or complexity of remediation. This status is acceptable for medium and low-severity findings after internal review and agreement. However, all high-severity findings must be resolved without exception.
Unresolved	The finding remains neither remediated nor formally acknowledged by the client, leaving the vulnerability unaddressed.

Audit Findings

Low

[L-01] SHIBC#openTrading, excludeFromMaxTransaction, removeLimits, removeSellLimit, SetFees - Lack of emitting events

Description

The above functions are missing events when the key variables are updated.

Recommendation

Add relevant events in the functions.

Note

The Shiba Classic team stated that only the function `removeSellLimit` can be executed by `deployerWallet`, other mentioned functions can't be called because the ownership is renounced.

Status

Acknowledged

Gas

[G-01] SHIBC - State variables that could be made immutable and constant

Description

The `initialTotalSupply`, `maxTransactionAmountPercent`, `maxWalletPercent`, and `swapTokensAtAmountPercent` variables are only set with hardcoded values during contract deployment. Such variables could be made constant to save gas when they are read.

The `swapTokensAtAmount` variable is only set in the `constructor` and this variable could be made immutable.

Recommendation

Make the `initialTotalSupply`, `maxTransactionAmountPercent`, `maxWalletPercent`, and `swapTokensAtAmountPercent` variables constant and the `swapTokensAtAmount` variable immutable.

Status

Acknowledged

QA

[Q-01] SHIBC#clearStuckEth - Deprecated Ether transfer function

Description

The `clearStuckEth()` function uses the `.transfer()` function to send the native coins.

However, the Istanbul update made some changes to the EVM, which made the `.transfer()` function deprecated for the ETH transfer.

Recommendation

Use `.call()` function to send ether instead of `.transfer()` function and check the return value.

Status

Acknowledged

[Q-02] SHIBC#clearStuckTokens - Return value of `transfer` function not checked

Description

The `clearStuckTokens` function calls the `transfer` function of `tokenContract` contract.

Not all ERC20 implementations perform reverting when there is an error in `transfer` or `transferFrom`.

If the return value is not checked, a transaction that should be marked as failed may succeed without an actual transfer.

Recommendation

Check the return values of the `transfer` function or use the `safeTransfer` function instead.

Status

Acknowledged

[Q-03] SHIBC - Unnecessary use of SafeMath

Description

Solidity 0.8.x versions support SafeMath as standard.

Recommendation

Remove the SafeMath import.

Status

Acknowledged

[Q-04] SHIBC#_transfer - Unnecessary check

Description

There are duplicate checks that compare `to` value and `address(0)`.

```
function _transfer(address from, address to, uint256 amount) internal {
    ...
    require(to != address(0), “ERC20: transfer to the zero address”);
    ...

     if (from != owner() && to != owner() && to != address(0) && to != address(0xdead)
&& !swapping) {
```

Recommendation

Remove the check for the `to` value.

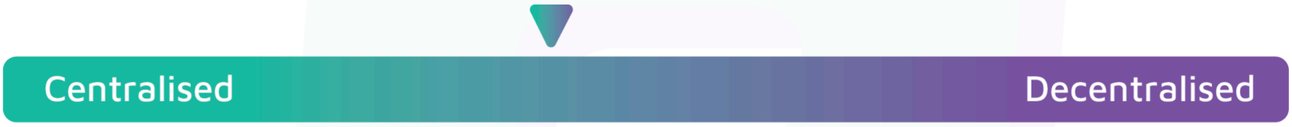
Status

Acknowledged

Centralisation

The Shiba Classic project values security and utility over decentralisation.

The owner executable functions within the protocol increase security and functionality but depend highly on internal team responsibility.



Conclusion

After Hashlock's analysis, the Shiba Classic project seems to have a sound and well-tested code base, now that our vulnerability findings have been resolved and acknowledged. Overall, most of the code is correctly ordered and follows industry best practices. The code is well commented on as well. To the best of our ability, Hashlock is not able to identify any further vulnerabilities.

Our Methodology

Hashlock strives to maintain a transparent working process and to make our audits a collaborative effort. The objective of our security audits is to improve the quality of systems and upcoming projects we review and to aim for sufficient remediation to help protect users and project leaders. Below is the methodology we use in our security audit process.

Manual Code Review:

In manually analysing all of the code, we seek to find any potential issues with code logic, error handling, protocol and header parsing, cryptographic errors, and random number generators. We also watch for areas where more defensive programming could reduce the risk of future mistakes and speed up future audits. Although our primary focus is on the in-scope code, we examine dependency code and behaviour when it is relevant to a particular line of investigation.

Vulnerability Analysis:

Our methodologies include manual code analysis, user interface interaction, and white box penetration testing. We consider the project's website, specifications, and whitepaper (if available) to attain a high-level understanding of what functionality the smart contract under review contains. We then communicate with the developers and founders to gain insight into their vision for the project. We install and deploy the relevant software, exploring the user interactions and roles. While we do this, we brainstorm threat models and attack surfaces. We read design documentation, review other audit results, search for similar projects, examine source code dependencies, skim open issue tickets, and generally investigate details other than the implementation.

Documenting Results:

We undergo a robust, transparent process for analysing potential security vulnerabilities and seeing them through to successful remediation. When a potential issue is discovered, we immediately create an issue entry for it in this document, even though we have not yet verified the feasibility and impact of the issue. This process is vast because we document our suspicions early even if they are later shown to not represent exploitable vulnerabilities. We generally follow a process of first documenting the suspicion with unresolved questions, and then confirming the issue through code analysis, live experimentation, or automated tests. Code analysis is the most tentative, and we strive to provide test code, log captures, or screenshots demonstrating our confirmation. After this, we analyse the feasibility of an attack in a live system.

Suggested Solutions:

We search for immediate mitigations that live deployments can take and finally, we suggest the requirements for remediation engineering for future releases. The mitigation and remediation recommendations should be scrutinised by the developers and deployment engineers, and successful mitigation and remediation is an ongoing collaborative process after we deliver our report, and before the contract details are made public.

Disclaimers

Hashlock's Disclaimer

Hashlock's team has analysed these smart contracts in accordance with the best industry practices at the date of this report, in relation to: cybersecurity vulnerabilities and issues in the smart contract source code, the details of which are disclosed in this report, (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

Due to the fact that the total number of test cases is unlimited, the audit makes no statements or warranties on the security of the code. It also cannot be considered as a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other statements of the contract. While we have done our best in conducting the analysis and producing this report, it is important to note that you should not rely on this report only. We also suggest conducting a bug bounty program to confirm the high level of security of this smart contract.

Hashlock is not responsible for the safety of any funds and is not in any way liable for the security of the project.

Technical Disclaimer

Smart contracts are deployed and executed on a blockchain platform. The platform, its programming language, and other software related to the smart contract can have their own vulnerabilities that can lead to attacks. Thus, the audit can't guarantee the explicit security of the audited smart contracts.

About Hashlock

Hashlock is an Australian-based company aiming to help facilitate the successful widespread adoption of distributed ledger technology. Our key services all have a focus on security, as well as projects that focus on streamlined adoption in the business sector.

Hashlock is excited to continue to grow its partnerships with developers and other web3-oriented companies to collaborate on secure innovation, helping businesses and decentralised entities alike.

Website: hashlock.com.au
Contact: info@hashlock.com.au

hashlock.

hashlock.

Hashlock Pty Ltd