hashlock.

Security Audit

The Winners Circle (Token)

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

Table of Contents

Executive Summary	4
Project Context	4
Audit scope	7
Security Rating	8
Intended Smart Contract Behaviours	9
Code Quality	10
Audit Resources	10
Dependencies	10
Severity Definitions	11
Audit Findings	12
Centralisation	14
Conclusion	15
Our Methodology	16
Disclaimers	18
About Hashlock	19

CAUTION

THIS DOCUMENT IS A SECURITY AUDIT REPORT AND MAY CONTAIN CONFIDENTIAL INFORMATION. THIS INCLUDES IDENTIFIED VULNERABILITIES AND MALICIOUS CODE THAT 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 THE USE OF THE CLIENT.

Executive Summary

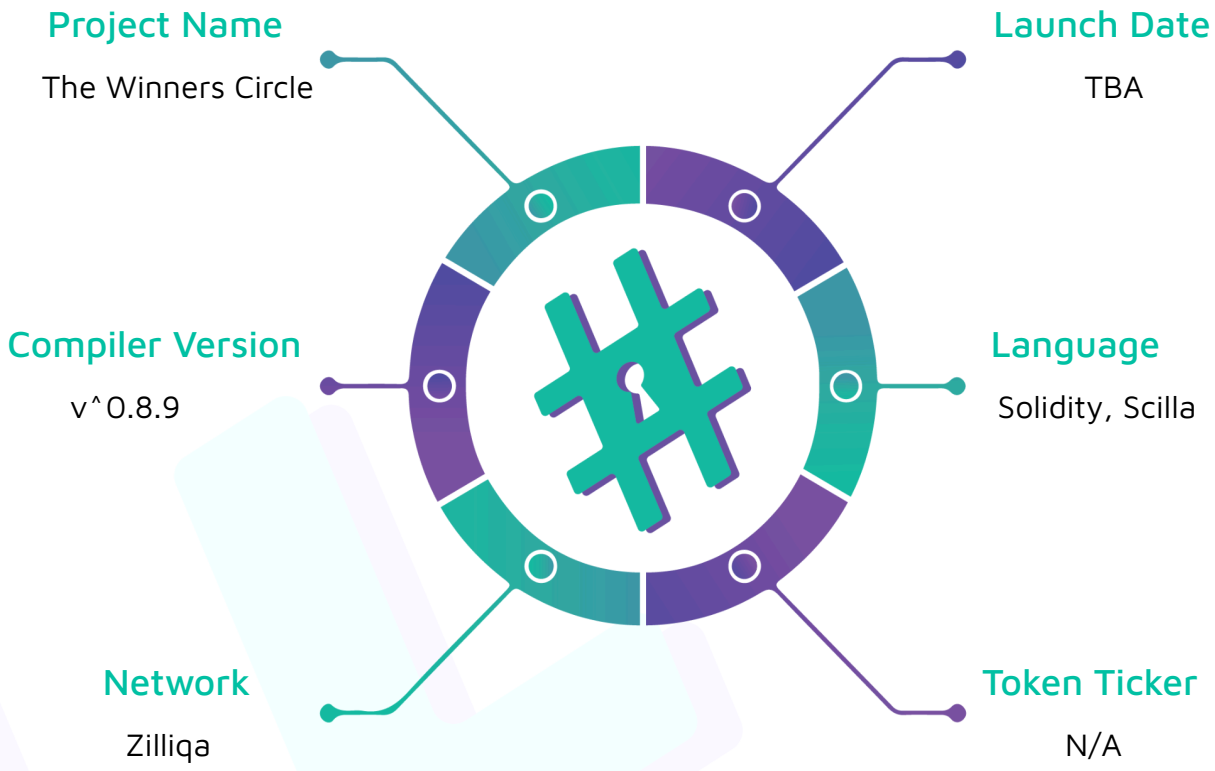
The Winners Circle team partnered with Hashlock to conduct a security audit of their HRSE.sol and HRSE.scilla smart contracts. Hashlock manually and proactively reviewed the code to ensure the project's team and community that the deployed contracts were secure.

Project Context

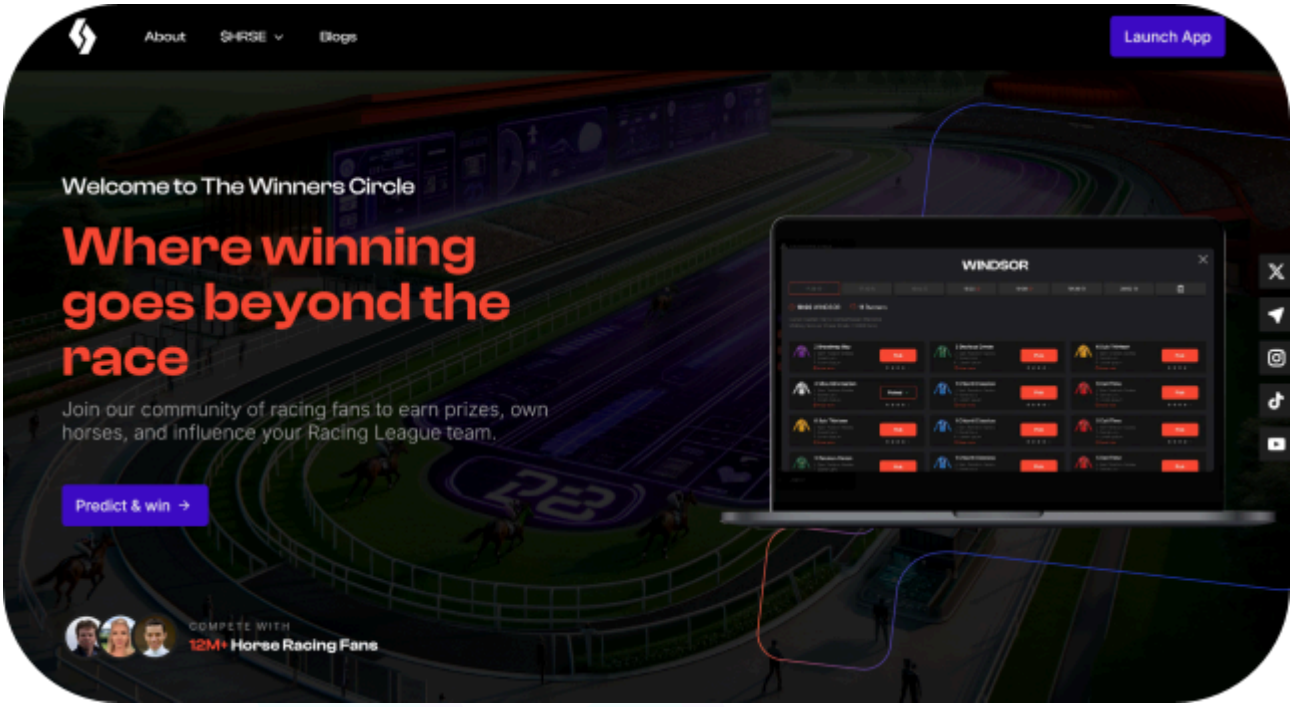
The Winners Circle is a public, permissionless blockchain that is designed to offer high throughput with the ability to complete thousands of transactions per second.

Project Name: The Winners Circle
Compiler Version: ^0.8.9
Website: https://www.thewinnerscircle.io/
Logo:

Visualised Context:



Project Visuals:



Audit scope

We at Hashlock audited the solidity and scilla codes within The Winners Circle 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	**The Winners Circle Smart Contracts**
Platform	**Zilliqa / Scilla, Solidity**
Audit Date	**July, 2024**
Contract 1	HRSE.sol
Contract 1 MD5 Hash	1250efea2ab87a6d4100462724b596e6
Contract 2	HRSE.scilla
Contract 2 MD5 Hash	d039a95a4ab69b7dfb700f876447df60

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.

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.

We initially identified some issues that have since been addressed.

Hashlock found:

3 Low-severity vulnerabilities

1 Gas Optimisations

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 Behaviours

Claimed Behaviour	Actual Behaviour
HRSE.sol - Allows users to: - Call the functions of the token contract on Zilliqa network	**Contract achieves this functionality.**
HRSE.scilla - Allows users to: - Transfer/TransferFrom the tokens	**Contract achieves this functionality.**

Code Quality

This audit scope involves the smart contracts for The Winners Circle project, as outlined in the Audit Scope section. All contracts, libraries, and interfaces mostly follow standard best practices 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 Winners Circle project smart contract code in the form of smart contract files.

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 help us understand the overall architecture of the protocol.

Dependencies

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

Severity Definitions

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

Audit Findings

QA

[Q-01] HRSE - Unnecessary use of SafeMath

Description

Solidity 0.8.x versions support SafeMath as standard.

Recommendation

Remove the SafeMath import.

Status

Acknowledged

Note:

QA refers to Quality Assurance, meaning there is no security risk present

The Winners Circle will resolve at a later date

[Q-02] HRSE - Lack of readability

Description

The `_UINT8_MAX` and `_UINT128_MAX` variables are used to store the maximum value of uint8 and uint128 types and they are assigned with 2**8 - 1 and 2**128 - 1.

However, in Solidity, there are already new keywords for returning max of certain value types like type(T).max.
Using this keyword is more useful to improve the readability.

Recommendation

Assign type(uint8).max to the `_UINT8_MAX` and type(uint128).max to the `_UINT128_MAX` .

Status

Acknowledged

Note:

QA refers to Quality Assurance, meaning there is no security risk present

The Winners Circle will resolve at a later date

[Q-03] HRSE - Wrong variable type

Description

The `_UINT8_MAX` is defined to store the maximum value of the uint8 type but the variable's type is uint128.

Recommendation

Use uint8 for the `_UINT8_MAX` variable.

Status

Acknowledged

Note:

QA refers to Quality Assurance, meaning there is no security risk present

The Winners Circle will resolve at a later date

Gas

[G-01] HRSE - Could use immutable

Description

The `zrc2_address` and `decimals` variables are only set in the constructor.

These variables could be set as immutable.

Recommendation

Make the `zrc2_address` and `decimals` variables immutable.

Status

Acknowledged

Note:

QA refers to Quality Assurance, meaning there is no security risk present

The Winners Circle will resolve at a later date

Centralisation

The Winners Circle prioritizes security and utility, relying on the internal team for certain functions. While decentralized, it values trustlessness over correctness, and its restaking functionality may lead to issues with third-party apps not adhering to guidelines.

Conclusion

After Hashlocks analysis, the Winners Circle project seems to have a sound and well-tested code base. 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 behavior 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 still need to verify the feasibility and impact of the issue. This process is vast because we document our suspicions early even if they are later shown not to 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 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.