Tetragaze

Smart contract audit

Tetragaze

January 19, 2022 | v 1.0

Pass

Tetragaze Security Team has concluded that there are no issues that can have an impact on contract security. The contract is well written and is production-ready.

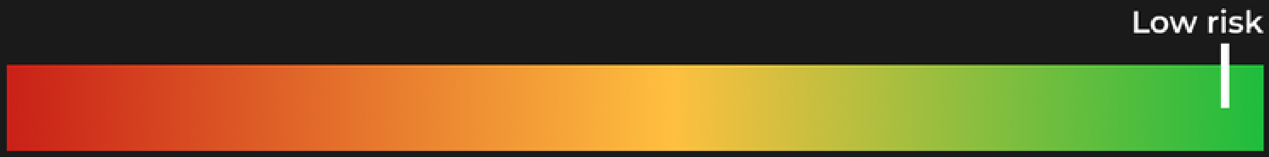
Score

96

Technical summary

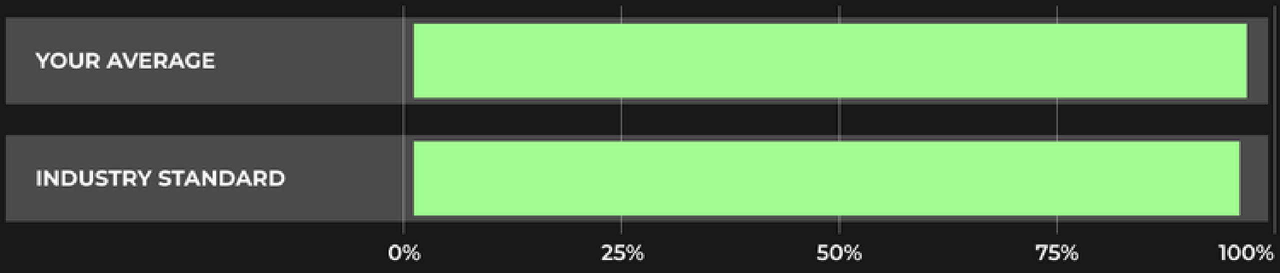
This document outlines the overall security of the SoulPiece smart contracts, evaluated by Tetragaze Blockchain Security team.
The scope of this audit was to analyze and document the SoulPiece smart contract codebase for quality, security, and correctness

Contract Status



There were no critical issues found during the audit.

Testable Code



Testable code is 96%, which is close to the industry standard of 95%.

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the Ethereum network's fast-paced and rapidly changing environment, we at Tetragaze recommend that the SoulPiece team put in place a bug bounty program to encourage further and active analysis of the smart contract.

Table of content

Auditing Strategy and Techniques Applied

Requirements: FIP8 spec

During the review process, we made sure to carefully check that the token contract:

- Adheres to established Token standards and ensures effective implementation;
- Ensures that documentation and code comments accurately reflect the logic and behavior of the code;
- Ensures that token distribution is in line with calculations;
- Utilizes best practices to efficiently use gas, avoiding unnecessary waste;
- Employs methods that are safe from reentrance attacks and immune to the latest vulnerabilities;
- Follows best practices for code readability

Tetragaze team of expert pentesters and smart contract developers carefully evaluated the repository for security issues, code quality, and compliance with specifications and best practices. They conducted a line-by-line review, documenting any issues that were identified during the process. Special care was taken to ensure that the review was thorough and comprehensive.

1	Due diligence in assessing the overall code quality of the codebase.	2	Cross-comparison with other, similar smart contracts by industry leaders.
3	Testing contract logic against common and uncommon attack vectors.	4	Thorough, manual review of the codebase, line-by-line.

Summary

Soul Piece is a token for a cryptocurrency card game that has a supportive and expanding community. However, it was discovered through a Smart Contract analysis that the owner has the ability to prevent users from making additional transactions after the first one. To prevent potential hacking, a multi-wallet signing pattern will be implemented for security. To completely eliminate any threats to the contract, the owner can temporarily lock the contract or renounce ownership.

Structure and Organization of Document

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

● Critical

The issue affects the ability of the contract to compile or operate in a significant way.

● High

The issue affects the ability of the contract to compile or operate in a significant way.

● Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

● Low

The issue has minimal impact on the contract’s ability to operate.

● Informational

The issue has no impact on the contract’s ability to operate.

Complete Analysis

Stop Transactions

Low

The contract owner has the ability to halt transactions for all users after they have made their initial transaction. For example, a user who has made a single purchase may not be able to sell again. The owner may use this power to set a high value for the antiBotSeconds variable, potentially taking advantage of the situation.

```
if (!whitelist[msg.sender]){
    uint secondsPassed = block.timestamp - lastTrade[msg.sender];
    require(secondsPassed >= antiBotSeconds, "AntiBot: You can't trade yet.");
}
```

Recomendation:

The contract could include a clause to prevent setting the antiBotSeconds for an unreasonable amount of time. It is important for the team to properly safeguard the private keys for the owner's account. To prevent a single user from accessing the contract admin functions, we highly recommend implementing a strong security mechanism. This risk can be minimized by temporarily locking the contract or renouncing ownership.

Complete Analysis

Public Function could be Declared External

Low

It is recommended to declare public functions that are not used by the contract as external in order to save gas.

```
decreaseAllowance
increaseAllowance
transferFrom
...
```

Recomendation:

Functions that are never called from the contract should utilize the external attribute.

Complete Analysis

Conformance to Solidity Naming Conventions

Low

Solidity has a specific naming convention that must be followed, with a few exceptions. Constant variables, symbols, and decimals may be lowercase. Private variables and unused parameters may have an underscore at the beginning of their mixed case names.

```
setAntiBotSeconds
setAntiBotStatus
```

Recomendation:

Follow the Solidity naming convention.
https://docs.soliditylang.org/en/v0.4.25/style-guide.html#naming-conventions

Complete Analysis

Elimination of Dead Code

Low

Unused functions in the contract that increase the code's size.

```
_burn
_beforeTokenTransfer
_afterTokenTransfer
...
```

Recomendation:

Remove unused functions.

We are thankful for the opportunity to collaborate with the Soul Piece team.

This document's statements should not be taken as investment or legal advice, and the authors cannot be held responsible for any decisions made based on them.

It is suggested by the Tetragaze Security Team that the Soul Piece team implement a bug bounty program in order to encourage external parties to scrutinize the smart contract further.