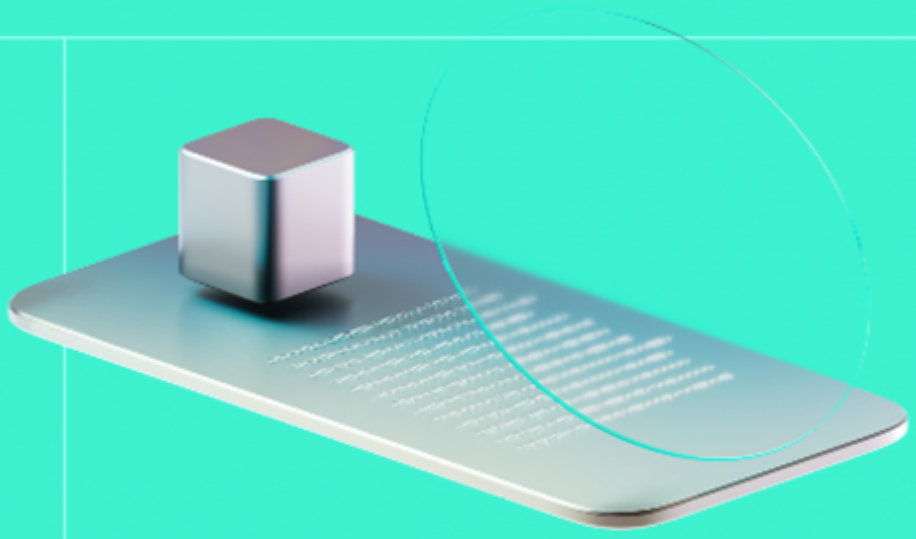
HACKEN

Smart Contract Code Review And Security Analysis Report

Customer: Ignition Staking 2.0

Date: 19/01/2024



We express our gratitude to the Ignition Staking 2.0 team for the collaborative engagement that enabled the execution of this Smart Contract Security Assessment.

Ignition Staking 2.0 is a staking platform featuring an off-chain reward mechanism. It enables the deposit of `PAID` tokens in exchange for `sPAID` tokens.

Platform: EVM

Language: Solidity

Tags: ERC20, Staking

Timeline: 15/01/2024 - 16/01/2024

Methodology: https://hackenio.cc/sc_methodology

Review Scope

Repository	https://github.com/PAIDNetwork/ignition-sc-staking
Commit	9478499

Audit Summary

10/10	10/10	100%	10/10
Security Score	Code quality score	Test coverage	Documentation quality score

Total 10/10

The system users should acknowledge all the risks summed up in the risks section of the report

1	1	0	0
Total Findings	Resolved	Accepted	Mitigated

Findings by severity

Critical	0
High	0
Medium	0
Low	1

Vulnerability	Status
F-2024-0463 - Unchecked Transfers Operations	Fixed

This report may contain confidential information about IT systems and the intellectual property of the Customer, as well as information about potential vulnerabilities and methods of their exploitation.

The report can be disclosed publicly after prior consent by another Party. Any subsequent publication of this report shall be without mandatory consent.

Document

Name	Smart Contract Code Review and Security Analysis Report for Ignition Staking 2.0
Audited By	Eren Gonen
Approved By	Grzegorz Trawinski
Website	https://paidnetwork.com/
Changelog	16/01/2024 - Preliminary Report 09/01/2024 - Remediation

Table of Contents

System Overview

Ignition Staking 2.0 is a staking platform where users can lock their `PAID` tokens to receive `sPAID` tokens in exchange. This system allows users the option to withdraw their deposited `PAID` tokens by burning the corresponding `sPAID` tokens. A tax is applied to both deposit and withdrawal transactions. Notably, the reward mechanism off-chain. The relevant contracts are as follows:

ERC20 — A simple ERC-20 token mints the deposited amount to the user when they execute a deposit and burns the amount when the user executes a withdrawal. The owner does not have the right to mint or burn tokens.

It has the following attributes:

- Name: Staked PAID
- Symbol: sPAID
- Decimals: 18
- Total supply: N/A.

SPAID —A contract that mints `sPAID` tokens when a user deposits `PAID` tokens and burns `sPAID` when a user wishes to withdraw the deposited `PAID` token amount.

Privileged roles

- The owner of the contract can update the deposit and withdrawal tax amounts within a range of 1-10.
- The owner has the capability to update the treasury wallet address
- The owner can withdraw the all deposited `PAID` tokens from the contract.

Executive Summary

This report presents an in-depth analysis and scoring of the customer's smart contract project. Detailed scoring criteria can be referenced in the scoring methodology.

Documentation quality

The total Documentation Quality score is **10** out of **10**.

- Functional requirements are provided.
- Technical description is provided.

Code quality

The total Code Quality score is **10** out of **10**.

- The development environment is configured.

Test coverage

Code coverage of the project is **100%** (branch coverage).

- Deployment and user interactions are thoroughly tested, covering both negative and positive cases.

Security score

Upon auditing, the code was found to contain **0** critical, **0** high, **0** medium, and **1** low severity issues, leading to a security score of **10** out of **10**.

All identified issues are detailed in the “Findings” section of this report.

Summary

The comprehensive audit of the customer's smart contract yields an overall score of **10**. This score reflects the combined evaluation of documentation, code quality, test coverage, and security aspects of the project.

Risks

- The owner of the contract has the authority to transfer all deposited PAID tokens after a user's deposit. **The withdrawal of deposited PAID tokens is not guaranteed.**
- The reward mechanism is entirely off-chain and **cannot be verified.**

Findings

Vulnerability Details

F-2024-0463 - Unchecked Transfers Operations - Low

Description: In the `SPAID.sol` contract, there are instances where the `transferFrom()` function is called to transfer tokens. However, the contract does not check the return values of these `transferFrom()` calls. Not all `ERC20` tokens are guaranteed to revert on failure; some may return a boolean value (false) instead. If the `SPAID.sol` contract interacts with such tokens, a failed transfer would not cause the transaction to revert, potentially leading to discrepancies in the contract's state.

The functions that do not check the return value of transfers include:

- `deposit()`
- `withdraw()`
- `withdrawAllStaked()`

Assets:

- contracts/SPAID.sol

Status: Fixed

Classification

Severity: Low

Impact: Likelihood [1-5]: 1

Impact [1-5]: 4

Exploitability [1,2]: 1

Complexity [0-2]: 0

Final Score: 2.0 [Low]

Recommendations

Recommendation: Either use the `SafeERC20` library in the mentioned functions to interact with tokens safely, or implement a return value check for the transfer functions.

Remediation (Revised commit: 9478499): The `SafeERC20` library from OpenZeppelin was implemented.

Observation Details

F-2024-0460 - Solidity version 0.8.20 might not work on all chains due to `PUSH0` - Info

Description: Solidity version 0.8.20 employs the recently introduced `PUSH0` opcode in the Shanghai EVM, this opcode might not be universally supported across all blockchain networks and Layer 2 solutions. It is advisable to use an earlier version of solidity to ensure compatibility.

Assets:

- contracts/SPAID.sol

Status: Fixed

Recommendations

Recommendation: To ensure compatibility with a wide range of blockchain networks and Layer 2 solutions, consider using an earlier version of Solidity that does not rely on the `PUSH0` opcode introduced in Solidity version 0.8.20. Using a more widely supported Solidity version can help avoid potential compatibility issues and ensure the smooth deployment and execution of smart contracts.

Remediation (Revised commit: d5fd4cb): The team fixed this issue by downgrading the Solidity version to `0.8.19`.

F-2024-0461 - Redundant Check in `deposit()` Function Due to Tautology in Requirement - Info

Description:

In the `deposit()` function of the contract, there is an `if` statement that checks if the amount parameter is less than or equal to zero. The statement is structured as follows:

```
if (amount <= 0)
```

However, this check contains a logical redundancy, as the amount variable is defined as a **uint256**. In Solidity, **uint256** (unsigned integer) variables inherently cannot hold negative values; they are always zero or positive. Therefore, the condition `amount < 0` is always false, making the **<** part of the **<=** operator redundant.

Assets:

- contracts/SPAID.sol

Status: Fixed

Recommendations

Recommendation:

Remove the **<** operator and reconfigure the `if` statement to eliminate redundancy. For instance, implement `if (amount == 0)` as an alternative.

Remediation (Revised commit: 5ffb45b): The redundant **<** operator was removed, and the recommendation to change the statement to `if (amount == 0)` was implemented.

F-2024-0462 - Missing Events - Info

Description: Events for critical state changes should be emitted for tracking actions off-chain.

It was observed that events are missing events in the following functions:

```
setTax()
setTreasury()
```

Assets:

- contracts/SPAID.sol

Status: Fixed

Recommendations

Recommendation: Consider emitting missing events.

Remediation (Revised commit: a83ffc1): The events were added and emitted for the mentioned functions.

F-2024-0464 - Inefficient Gas Usage in `withdraw()` and `withdrawAllStaked()` Functions Due to Use of `transferFrom()` - Info

Description: In the contract, the `withdraw()` and `withdrawAllStaked()` functions utilize the `transferFrom()` function to transfer tokens from the contract to the user. However, the `transferFrom()` function includes additional allowance checks, which are not necessary when transferring tokens from the contract itself. These extra checks result in increased gas costs compared to using the `transfer()` function.

Assets:

- contracts/SPAID.sol

Status: Fixed

Recommendations

Recommendation: Modify the `withdraw()` and `withdrawAllStaked()` functions to use the `safeTransfer()` function instead of `transferFrom()` for transferring tokens from the contract to users. This change will eliminate unnecessary allowance checks and reduce gas costs.

Remediation (Revised commit: 9478499): The `safeTransfer()` function from the `SafeERC20` library was implemented.

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed based on best industry practices at the time of the writing of this report, with cybersecurity vulnerabilities and issues in 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).

The report contains no statements or warranties on the identification of all vulnerabilities and security of the code. The report covers the code submitted and reviewed, so it may not be relevant after any modifications. Do not consider this report as a final and sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements.

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 recommend proceeding with several independent audits and a public bug bounty program to ensure the security of smart contracts.

English is the original language of the report. The Consultant is not responsible for the correctness of the translated versions.

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 vulnerabilities that can lead to hacks. Thus, the Consultant cannot guarantee the explicit security of the audited smart contracts.

Appendix 1. Severity Definitions

When auditing smart contracts, Hacken is using a risk-based approach that considers **Likelihood**, **Impact**, **Exploitability** and **Complexity** metrics to evaluate findings and score severities.

Reference on how risk scoring is done is available through the repository in our Github organization:

hknio/severity-formula

Severity	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to the loss of user funds or contract state manipulation.
High	High vulnerabilities are usually harder to exploit, requiring specific conditions, or have a more limited scope, but can still lead to the loss of user funds or contract state manipulation.
Medium	Medium vulnerabilities are usually limited to state manipulations and, in most cases, cannot lead to asset loss. Contradictions and requirements violations. Major deviations from best practices are also in this category.
Low	Major deviations from best practices or major Gas inefficiency. These issues will not have a significant impact on code execution, do not affect security score but can affect code quality score.

Appendix 2. Scope

The scope of the project includes the following smart contracts from the provided repository:

Scope Details - Initial

Repository	https://github.com/PAIDNetwork/ignition-sc-staking
Commit	235a9a33f30314861e9b8b63add9ace2f5d114ba
Whitepaper	Not Provided
Requirements	https://github.com/PAIDNetwork/ignition-sc-staking/blob/main/README.md
Technical Requirements	https://github.com/PAIDNetwork/ignition-sc-staking/blob/main/README.md

Scope Details - Second

Repository	https://github.com/PAIDNetwork/ignition-sc-staking
Commit	d5fd4cb0c544b689c6688cba7d6a95e0ac8e2865
Whitepaper	Not Provided
Requirements	https://github.com/PAIDNetwork/ignition-sc-staking/blob/main/README.md
Technical Requirements	https://github.com/PAIDNetwork/ignition-sc-staking/blob/main/README.md

Scope Details - Third

Repository	https://github.com/PAIDNetwork/ignition-sc-staking
Commit	9478499ec973e64001b89401ddb1a9edd002715b
Whitepaper	Not Provided
Requirements	https://github.com/PAIDNetwork/ignition-sc-staking/blob/main/README.md

Scope Details - Third

Technical Requirements	https://github.com/PAIDNetwork/ignition-sc-staking/blob/main/README.md

Contracts in Scope

./contracts/SPAID.sol