HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: GovWorld
Date: April 13th, 2022

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Document

Name	Smart Contract Code Review and Security Analysis Report for GovWorld.
Approved by	Andrew Matiukhin | CTO Hacken OU Evgeniy Bezuglyi | SC Department Head at Hacken OU
Type	ERC20 token; White Label Exchange
Platform	EVM
Language	Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/GovWorld/sun-claimboard-contracts
Commit	40860b82e317787ffb9e6bdbb39ee248182c375f - Initial Audit 613a7e568557547eaf7211502b0b431e6e763fc5 - Remediation Check
Deployed contract	NO
Technical Documentation	YES - https://docs.google.com/document/d/1uIowuBkfembXG-qzIHoYg9cZZorp5LyJXOL7JiDQIyk/edit
JS tests	YES - https://github.com/GovWorld/sun-claimboard-contracts/tree/master/test - Connect to preview
Website	https://www.govworld.io
Timeline	22 MARCH 2022 - 13 APRIL 2022
Changelog	28 MARCH 2022 - Initial Audit 05 APRIL 2022 - Remediation Check 13 APRIL 2022 - Remediation Check 2

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Table of contents

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Introduction

Hacken OÜ (Consultant) was contracted by GovWorld (Customer) to conduct a Smart Contract Code Review and Security Analysis. This report presents the findings of the security assessment of the Customer's smart contracts.

Scope

The scope of the project is smart contracts in the repository:

Repository:
https://github.com/GovWorld/sun-claimboard-contracts
Commit:
40860b82e317787ffb9e6bdbb39ee248182c375f - Initial Audit
613a7e568557547eaf7211502b0b431e6e763fc5 - Revised Commit
Technical Documentation: Yes - https://docs.google.com/document/d/1uIowuBkfembXG-qzIHoYg9cZZorp5LyJXOL7JiDQIyk/edit
JS tests: Yes - https://github.com/GovWorld/sun-claimboard-contracts/tree/master/test
Contracts:
SunToken.sol
ClaimBoard.sol
ClaimBoardBase.sol
LaunchpadFactory.sol
LaunchpadBoard.sol
ClaimBoardFactory.sol
TokenFactory.sol
ClaimData.sol
IGovWorldAdminRegistry.sol
IBoard.sol
ITokenFactory.sol
IClaimBoardExtras.sol
IERC20Extras.sol
IClaimBoardFactory.sol
ISunToken.sol

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

We have scanned this smart contract for commonly known and more specific vulnerabilities. Here are some of the commonly known vulnerabilities that are considered:

Category	Check Item
Code review	▪ Reentrancy ▪ Ownership Takeover ▪ Timestamp Dependence ▪ Gas Limit and Loops ▪ DoS with (Unexpected) Throw ▪ DoS with Block Gas Limit ▪ Transaction-Ordering Dependence ▪ Style guide violation ▪ Costly Loop ▪ ERC20 API violation ▪ Unchecked external call ▪ Unchecked math ▪ Unsafe type inference ▪ Implicit visibility level ▪ Deployment Consistency ▪ Repository Consistency ▪ Data Consistency
Functional review	▪ Business Logics Review ▪ Functionality Checks ▪ Access Control & Authorization ▪ Escrow manipulation ▪ Token Supply manipulation ▪ Assets integrity ▪ User Balances manipulation ▪ Data Consistency manipulation ▪ Kill-Switch Mechanism ▪ Operation Trails & Event Generation

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Executive Summary

The score measurements details can be found in the corresponding section of the methodology.

Documentation quality

The Customer provided detailed functional requirements and technical requirements. The total Documentation Quality score is **10** out of **10**.

Code quality

The total CodeQuality score is **10** out of **10**. Code follows official language style guides. Unit tests were provided.

Architecture quality

The architecture quality score is **10** out of **10**. Smart contracts of the project follow the best practices, and the project has a clear architecture.

Security score

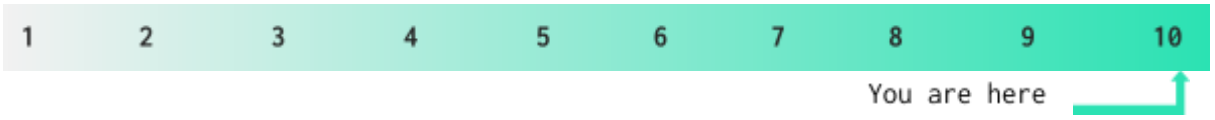
As a result of the audit, security engineers found **2** medium and **8** low severity issues. The security score is **10** out of **10**. All found issues are displayed in the “Issues overview” section.

As a result of the remediations review, Customers’ smart contracts contain **2** low severity issues.

As a result of the second remediations review, Customers’ smart contracts contain **no** issues.

Summary

According to the assessment, the Customer's smart has the following score: **10.0**



Notices

1. SUN Tokens can be burnt by some roles (*liquidator* and *tokenMarket*) that are not in the Audit scope.

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Severity Definitions

Risk Level	Description
Critical	Critical vulnerabilities are usually straightforward to exploit and can lead to assets loss or data manipulations.
High	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution, e.g., public access to crucial functions
Medium	Medium-level vulnerabilities are important to fix; however, they cannot lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that cannot have a significant impact on execution

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Audit overview

Critical

No critical issues were found.

High

No high issues were found.

Medium

Potential Out-of-Gas Exception

Iterating over user-supplied can consume much Gas as the size of the array grows.

This could lead to a potential Out-of-Gas exception.

Contracts: ClaimBoardBase.sol, ClaimBoard.sol, LaunchBoard.sol, ClaimBoardFactory.sol,

Function: addAllocationLaunchPad(), constructor(), deployClaimBoard(),

Recommendation: add an array size limit.

Status: Fixed

Low

1. **Missing Empty String Check**

   While distributions are being configured, the contract only checks if string lengths for token name and symbol are equal. Empty strings can meet this condition.

   Contracts: ClaimBoardBase.sol

   Function: configureDistributions()

   Recommendation: In addition to checking the length, whether these strings are filled or not should be checked.

   Status: Fixed

2. **Use of Hardcoded Values**

   Hardcoded values are being used in calculations.

   Contracts: ClaimBoardBase.sol

   Function: addAllocations(), getClaimableAmountbyRound(), getDays(),

   Recommendation: Move hardcoded values to constants.

Status: Fixed

3. **Missing Zero Address Check**

   Parameters of address type should be checked for potential zero address before use.

   Contracts: ClaimBoard.sol, LaunchBoard.sol,

   Function: constructor(),

   Recommendation: Implement a zero address check.

   Status: Fixed

4. **Check Should be Performed Outside of the Loop**

   Checking address and totalAmounts lengths in a loop over and over again consume unnecessary Gas.

   Contracts: ClaimBoard.sol,

   Function: addAllocationLaunchPad()

   Recommendation: Move check outside the loop.

   Status: Fixed

5. **Floating Pragma**

   The project uses floating pragma ^0.8.0.

   Contracts: SunToken.sol, ClaimBoard.sol, LaunchpadFactory.sol, LaunchpadBoard.sol, ClaimBoardFactory.sol, TokenFactory.sol, ClaimData.sol , IGovWorldAdminRegistry.sol, IBoard.so, ITokenFactory.so, IClaimBoardExtras.so, IERC20Extras.so, IClaimBoardFactory.so, ISunToken.sol

   Function: -

   Recommendation: Consider locking the pragma version whenever possible and avoid using a floating pragma in the final deployment.

   Status: Fixed

6. **Functions that can be Declared as *external***

   In order to save gas, public functions that are never called in the contract should be declared as *external*.

   Contracts: SunToken.sol, ClaimBoard.sol, ClaimBoardFactory.sol, TokenFactory.sol, ClaimBoardBase.sol, LaunchBoard.sol

Function: name(), symbol(), decimals(), totalSupply(), balanceOf(), transfer(), allowance(), approve(), transferFrom(), increaseAllowance() , burn(), burnFrom(), pause(), unpause(), claimSunToken(), claimAllNative(), activateClaimBoard(), claimTokenLaunchpadWallets(), withdrawLaunchPadWallet(), LaunchBoard.activateClaimBoard(), setLiquidator(), setTokenMarket(), setClaimBoard()

Recommendation: Consider locking the pragma version whenever possible and avoid using a floating pragma in the final deployment.

Status: Fixed

7. **Function that can be Declared as *internal or private***

External calls to the same contract are expensive in terms of Gas consumption.

Contracts: ClaimBoardBase.sol

Function: canClaim()

Recommendation: Change function visibility.

Status: Fixed

8. **Incorrect Solidity Version**

Using an old version prevents access to new Solidity security checks.

Contracts: SunToken.sol, ClaimBoard.sol, LaunchpadFactory.sol, LaunchpadBoard.sol, ClaimBoardFactory.sol, TokenFactory.sol, ClaimData.sol , IGovWorldAdminRegistry.sol, IBoard.so, ITokenFactory.so, IClaimBoardExtras.so, IERC20Extras.so, IClaimBoardFactory.so, ISunToken.sol

Function: -

Recommendation: Consider using one of these versions: *0.8.4, 0.8.6*.

Status: Fixed

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed by the best industry practices at the date 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 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 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.

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 audit cannot guarantee the explicit security of the audited smart contracts.