HACKEN

SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: XTblock
Date: December 08th, 2021

Document

Name	Smart Contract Code Review and Security Analysis Report for XTblock.
Approved by	Andrew Matiukhin | CTO Hacken OU
Type	Staking
Platform	Binance / Solidity
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Repository	https://github.com/xtblock/binosaur/blob/main/contracts/SmartChef.sol
Commit	59160193bc2ab5c7d7198fb80f3685be60c2b440
Technical Documentation	NO
JS tests	NO
Website	Xtblock.io
Timeline	18 NOVEMBER 2021 – 08 DECEMBER 2021
Changelog	26 NOVEMBER 2021 – Initial Audit 03 DECEMBER 2021 - Second Review 08 DECEMBER 2021 - Third Review

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by XTblock (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 contract and its code review conducted between November 11th, 2021 - November 26th, 2021.

Second review conducted on December 3rd, 2021.

Third review conducted on December 7th, 2021.

Scope

The scope of the project is smart contracts in repository:
Repository:
   https://github.com/xtblock/binosaur/blob/main/contracts/SmartChef.sol
Commits:
   59160193bc2ab5c7d7198fb80f3685be60c2b440
Technical Documentation: No
JS tests: No
Contracts:
   SmartChef.sol
   math/SafeMath.sol
   token/BEP20/IBEP20.sol
   token/BEP20/SafeBEP20.sol
   access/Ownable.sol
   utils/Address.sol
   utils/Context.sol

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

Executive Summary

According to the assessment, the Customer's smart contracts are secured.

| Insecure	Poor secured	Secured	Well-secured

You are here ⬆
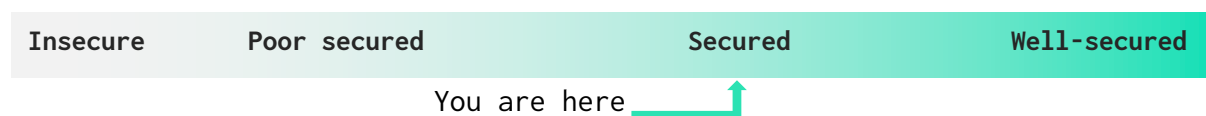
Our team performed an analysis of code functionality, manual audit, and automated checks with Mythril and Slither. All issues found during automated analysis were manually reviewed, and important vulnerabilities are presented in the Audit overview section. All found issues can be found in the Audit overview section.

As a result of the audit, security engineers found **1** high and **1** medium severity issue.

After the second review security engineers found **1** high and **1** medium severity issue.

After the third review security engineers found **1** medium severity issue.

www.hacken.io

Graph 1. The distribution of vulnerabilities after the audit.

Medium
100,0%
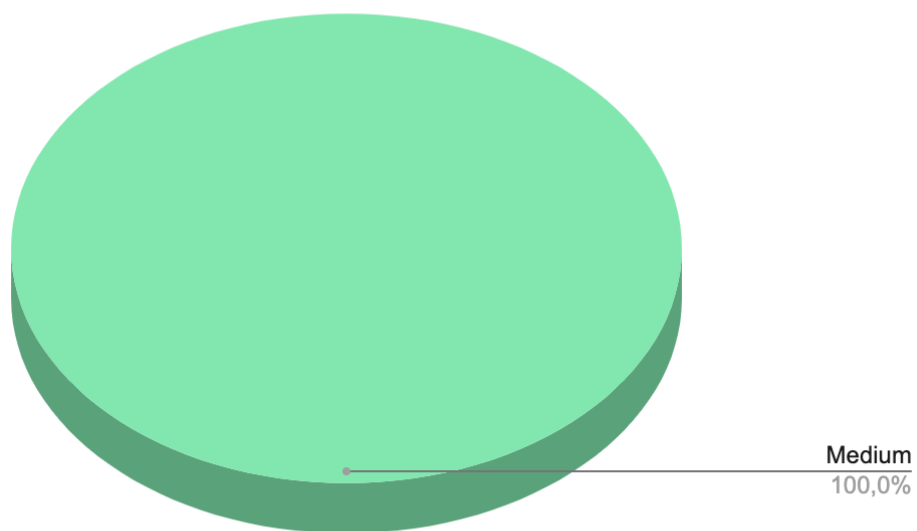
Severity Definitions

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 can't lead to assets loss or data manipulations.
Low	Low-level vulnerabilities are mostly related to outdated, unused, etc. code snippets that can't have a significant impact on execution

Audit overview

■ ■ ■ ■ Critical

No critical issues were found.

■ ■ ■ High

Essential methods could become inoperable

Contracts: SmartChef.sol

Function: getTotalSupply, rewardDistribution

Some functions iterate over array *userList*. It's constantly growing therefore calls will become more gas-consuming over time up to inoperability. A malicious user could speed it up by calling *deposit* (+ *emergencyRewardWithdraw*) from different accounts.

Recommendation: Track overall total supply; calculate rewards on a user basis, during his/her call.

Status: fixed

■ ■ Medium

No tests were provided

It's recommended to cover all non-trivial contracts with tests.

The recommended coverage is minimum 95% for branches, while it should be definitely 100% for the main logic contracts.

■ Low

No low severity issues were found.

Conclusion

Smart contracts within the scope were manually reviewed and analyzed with static analysis tools.

The audit report contains all found security vulnerabilities and other issues in the reviewed code.

As a result of the audit, security engineers found **1** high and **1** medium severity issue.

After the second review security engineers found **1** high and **1** medium severity issue.

After the third review security engineers found **1** medium severity issue.

Disclaimers

Hacken Disclaimer

The smart contracts given for audit have been analyzed in accordance with the best industry practices at the date of this report, in relation to 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 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 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 can't guarantee the explicit security of the audited smart contracts.