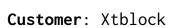


SMART CONTRACT CODE REVIEW AND SECURITY ANALYSIS REPORT



Date: August 24th, 2021



This document 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 containing confidential information can be used internally by the Customer, or it can be disclosed publicly after all vulnerabilities are fixed — upon a decision of the Customer.

Document

Name	Smart Contract Code Review and Security Analysis Report for		
	Xtblock.		
Approved by	Andrew Matiukhin CTO Hacken OU		
Туре	Locker contract		
Platform	Ethereum / Solidity		
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review		
Repository	https://github.com/xtblock/locker-contract		
Commit	7D92506FB4E68CB31C721534B946823989C78ED6		
Technical	NO		
Documentation			
JS tests	NO		
Timeline	20 AUGUST 2021 - 24 AUGUST 2021		
Changelog	23 AUGUST 2021 - INITIAL AUDIT		
	24 AUGUST 2021 - SECOND REVIEW		

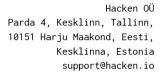




Table of contents

Introduction	4
Scope	4
Executive Summary	5
Severity Definitions	6
Audit overview	7
Conclusion	9
Disclaimers	10



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 August $20^{\rm th}$, 2021 - August $23^{\rm rd}$, 2021. The second review conducted on August $24^{\rm th}$, 2021.

Scope

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

Repository:

https://github.com/xtblock/locker-contract

Commit:

7d92506fb4e68cb31c721534b946823989c78ed6

Technical Documentation: No

JS tests: No Contracts:

Address.sol IERC20.sol SafeERC20.sol SafeMath.sol XTT-TokenTimeLock.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	•	Busines
		_	Functio

- 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 well-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 6 low severity issues.

After the second review security engineers found no issues in the code.



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

No high severity issues were found.

Medium

No medium severity issues were found.

Low

1. Incorrect versions of Solidity

The pragma header specifies a solidity version starting from $\underline{0.6.0}$ and up to $\underline{0.8.0}$ excluding it. But actually, the contract could not be compiled because constructor visibility was removed only since $\underline{0.7.0}$, therefore you cannot compile the contract using the most common $\underline{0.6.12}$ solidity version.

Recommendation: Please specify the strict version of solidity which were used to write code for. The recommended version is 0.7.6

Fixed before the second review.

2. Missing event on release time change

Updating the release time for tokens is recommended to follow up with the emitting an event. This will make it easier to track the updates off-chain.

Recommendation: Please emit the event on updating the release time.

Fixed before the second review.

3. Missing event on updating the unlocker

Updating the unlocker address is recommended to follow up with the emitting an event. This will make it easier to track the updates off-chain.

Recommendation: Please emit the event on updating the unlocker address.

Fixed before the second review.

4. Missing zero-address validation

In both places, constructor and setUnlocker(address) functions there is no check for zero address which could lead to mistakenly setting some of them to 0x0

Recommendation: Please add a zero-check for addresses.



Fixed before the second review.

5. A public function that could be declared external

public functions that are never called by the contract should be declared external to save gas.

Recommendation: Use the **external** attribute for functions never called from the contract.

Fixed before the second review.

6. Maximum line length

The solidity provides style guides as well as code layout recommendations where they have a recommended maximum line length.

Recommendation: Please follow the recommended <u>maximum line length</u>.

Fixed before the second review.



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 6 low severity issues.

After the second review security engineers found no issues in the code.



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.