BlockApex

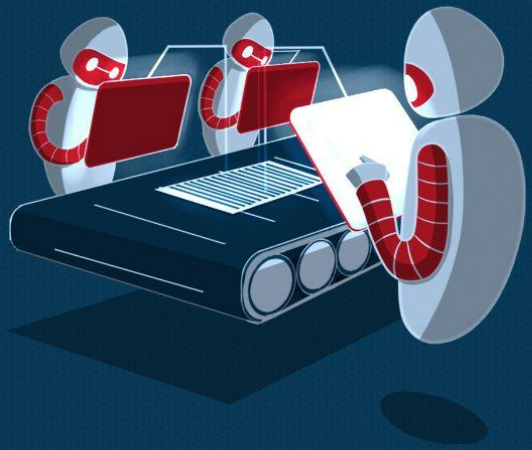
SMART CONTRACT SECURITY ANALYSIS REPORT

```
pragma solidity 0.7.0;
contract Contract {

    function hello() public returns (string) {
        return "Hello World!";
    }

    function findVulnerability() public returns (string) {
        return "Finding Vulnerability";
    }

    function solveVulnerability() public returns (string) {
        return "Solve Vulnerability";
    }
}
```



Powered by XORD

PREFACE

Objectives

The purpose of this document is to highlight any identified bugs/issues in the provided codebase. This audit has been conducted in a closed and secure environment, free from influence or bias of any sort. This document may contain confidential information about IT systems/architecture and intellectual property of the client. It also contains information about potential risks and the processes involved in mitigating/exploiting the risks mentioned below. The usage of information provided in this report is limited, internally, to the client. However, this report can be disclosed publicly with the intention to aid our growing blockchain community; under the discretion of the client.

Key understandings

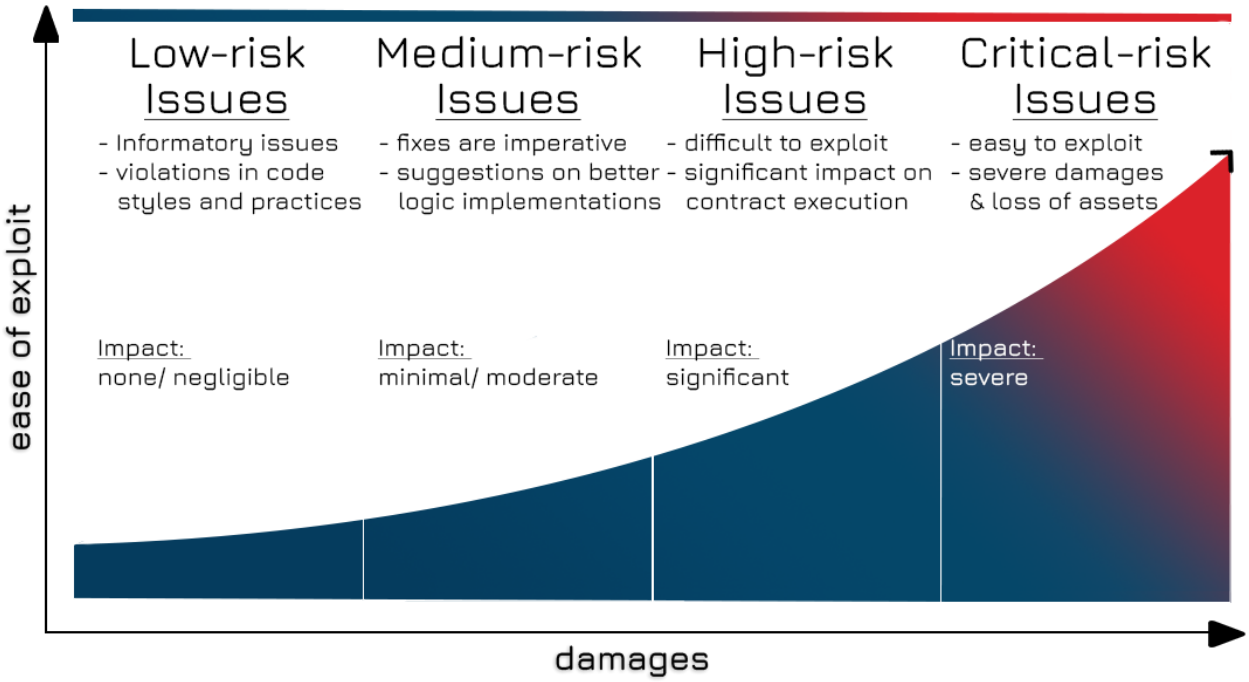


TABLE OF CONTENTS

INTRODUCTION

BlockApex (Auditor) was contracted by Borderless Money (Client) for the purpose of conducting a Smart Contract Audit/Code Review. This document presents the findings of our analysis which started on Septemebr 12th, 2022 .

Name
Borderless Money
Audited by
BlockApex
Platform
Ethereum | Solidity
Type of review
Manual Code Review | Automated Tools Analysis
Methods
Architecture Review | Functional Testing | Computer-Aided Verification
Git repository/ Commit Hash
https://github.com/BorderlessMoney-BOM/contracts/commit/e9765d2811c50b373d91b0b930d6aa2d86ec4a3c
Contract
https://polygonscan.com/address/0xc59132FBdF8dE8fbE510F568a5D831C991B4fC38#code
Website URL
https://borderless.money/
Document log
Audit Completed: September 16th, 2022

Scope

The git-repository shared was checked for common code violations along with vulnerability-specific probing to detect **major issues/vulnerabilities**. Some specific checks are as follows:

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

Project Overview

Borderless Money is a decentralized finance protocol redefining how Social Investments are made, using yield-generating strategies and contributing to social causes. An open, borderless digital society, with borderless money, where the goods, services, technology, information, opportunities, and capital can flow through the borders from one hand to many, fairly, transparently.

Methodology & Scope

The BOM Token codebase was audited using a filtered audit technique. A pair of auditors scanned the codebase in an iterative process spanning over a span of 4 days.

Starting with the recon phase, a basic understanding was developed and the auditors worked on establishing presumptions for the codebase and the relevant documentation/ whitepaper provided or found publicly.

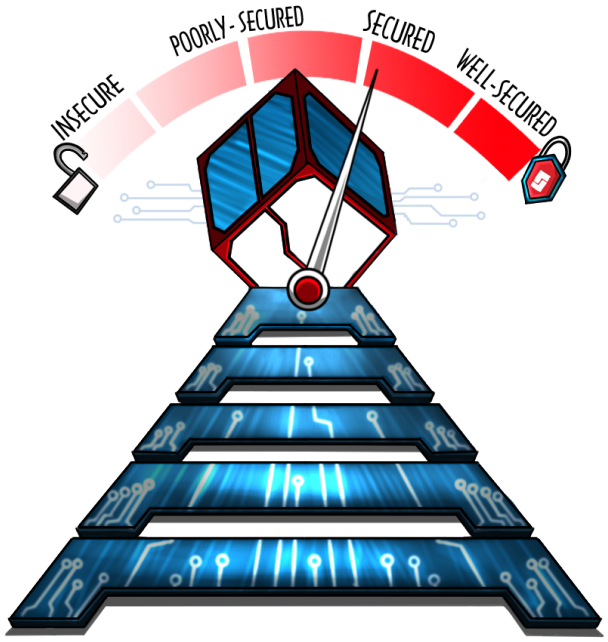
Furthermore, the audit moved on with the manual code reviews with the motive to find logical flaws in the codebase complemented with code optimizations, software and security design patterns, code styles, best practices and identifying false positives that were detected by automated analysis tools.

AUDIT REPORT

Executive Summary

The analysis indicates that the contract under scope of audit is **working properly**.

Our team performed a technique called “Filtered Audit”, where the contract was separately audited by two individuals. After a thorough and rigorous process of manual testing, an automated review was carried out using tools like slither for an extensive static analysis and foundry for property testing the invariants of the system. All the flags raised were manually reviewed in collaboration and re-tested to identify the false positives.



Issues Found:

Number of issues	Severity of the risk
0	Critical Risk issue(s)
0	High Risk issue(s)
0	Medium Risk issue(s)
0	Low Risk issue(s)
0	Informatory issue(s)

Properties Tested

#	Properties	Type	Status
1.	Compliant with ERC20 standard	Custom	Passed
2.	Should mint initial supply to msg.sender on deployment	Custom	Passed
3.	Should be able to approve tokens	Custom	Passed
4.	Should be able to increase allowance	Custom	Passed
5.	Should be able to decrease allowance	Custom	Passed
6.	Should be able to spend approved tokens	Custom	Passed
7.	Should not be able to mint more than initial supply	Custom	Passed
8.	Should update balances of sender and recipient when token transferred	Custom	Passed
9.	Reverts if sender doesn't holds enough token balance for sending	Custom	Passed
10.	Reverts if spender doesn't hold enough approval to spend someone's tokens	Custom	Passed
11.	Functions and state variables visibility should be set explicitly. Visibility levels should be specified consciously.	SWC-100 SWC-108	Passed
12.	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	SWC-101	Passed
13.	It is recommended to use a recent version of the Solidity compiler.	SWC-102	Passed
14.	Contracts should be deployed with the same compiler version and flags that they have been tested thoroughly.	SWC-103	Passed

15.	Ownership takeover should not be possible. All crucial functions should be protected. Users could not affect data that belongs to other users.	CWE-284	Passed
16.	Race Conditions and Transactions Order Dependency should not be possible.	CWE-114	Passed
17.	Tokens can be minted only according to rules specified in a whitepaper or any other documentation provided by the customer.	Custom	Passed
18.	Incorrect Inheritance Order which may result in unusual behaviour of smart contract functions	SWC-125	Passed

DISCLAIMER

The smart contracts provided by the client for audit purposes have been thoroughly analyzed in compliance with the global best practices till date w.r.t cybersecurity vulnerabilities and issues in smart contract code, the details of which are enclosed in this report.

This report is not an endorsement or indictment of the project or team, and they do not in any way guarantee the security of the particular object in context. This report is not considered, and should not be interpreted as an influence, on the potential economics of the token, its sale or any other aspect of the project.

Crypto assets/tokens are results of the emerging blockchain technology in the domain of decentralized finance and they carry with them high levels of technical risk and uncertainty. No report provides any warranty or representation to any third-Party in any respect, including regarding the bug-free nature of code, the business model or proprietors of any such business model, and the legal compliance of any such business. No third-party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. Specifically, for the avoidance of doubt, this report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and it is not a guarantee as to the absolute security of the project.

Smart contracts are deployed and executed on a blockchain. The platform, its programming language, and other software related to the smart contract can have its vulnerabilities that can lead to hacks. The scope of our review is limited to a review of the Solidity code and only the Solidity code we note as being within the scope of our review within this report. The Solidity language itself remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond Solidity that could present security risks.

This audit 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 security of smart contracts.