BlockApex

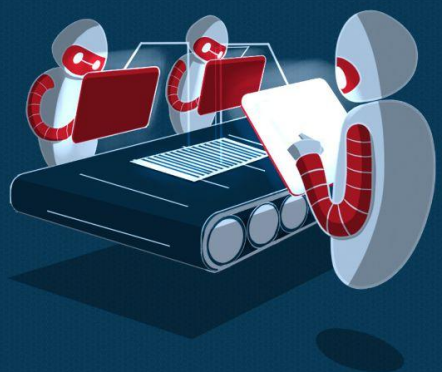
SMART CONTRACT SECURITY ANALYSIS REPORT

```
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

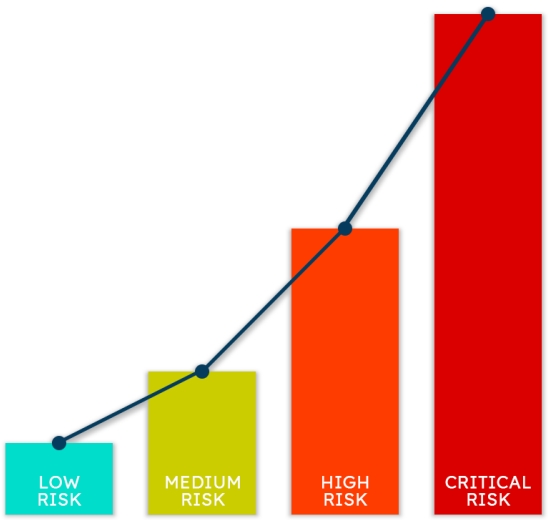
PREFACE

Objectives

The purpose of this document is to highlight the 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



CRITICAL RISK xxxx	Critical vulnerabilities are too easy to exploit and can lead to damages/loss in assets or manipulations.
HIGH RISK xxx	High-level vulnerabilities are difficult to exploit; however, they also have a significant impact on smart contract execution.
MEDIUM RISK xx	Medium-level vulnerabilities are equally imperative to fix but they tend to have minimal impact on asset loss or data manipulations.
LOW RISK x	Lowest-level vulnerabilities, informational errors, violating code styles/practices usually can't affect smart contract execution; hence they can be ignored.

TABLE OF CONTENTS

INTRODUCTION

BlockApex (Auditor) was contracted by SONAR (Client) for the purpose of conducting a Smart Contract Audit/Code Review. This document presents the findings of our analysis which took place on 28th September 2021.

Name
Sonar BSC-ETH bridge v2
Auditor
Moazzam Arif | Kaif Ahmed
Platform
Ethereum/Solidity
Type of review
Bridge
Methods
Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Git repository
https://github.com/XORD-one/sonar-bridge-contract/tree/17cb255444153945b66c5da8672f2ca06ec91efd
White paper/ Documentation
Not provided
Document log
Bridge v1 Audit (08-09-2021)
Bridge v2 Audit (28-09-2021)*

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

This is a POA bridge application that provides ETH to BSC and BSC to ETH bridging. It allows its native tokens to be accessed on both ETH and BSC blockchains.

System Architecture

Steps to burn and mint (BSC => ETH):

1. User calls the burn method on BSC bridge (BSC blockchain)
2. With the burn method, tokens are transferred to the admin account
3. Transfer event is emitted on successful burn
4. Relayer listens the event and call the mint method on ETH bridge
5. With the mint method, the tokens are transferred to the user on the ethereum blockchain

For ETH => BSC similar steps are followed.

Methodology & Scope

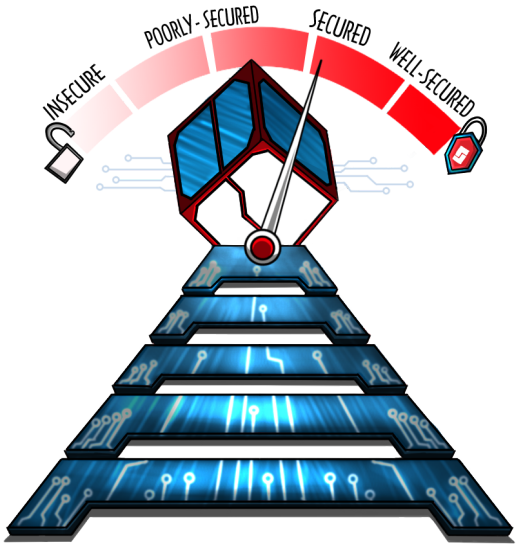
Smart contract for the native token (PING) was provided. As $PING is similar to $RFI and $SAFEMOON. So the swapping and fee distributions were primarily focused during this security assessment (as they had already been audited by Certik). Only minting and burning functionality of $ePING (ethereum representation of $PING) was considered in scope. Manual Review and Static Analysis tools were used to produce this report.

AUDIT REPORT

Executive Summary

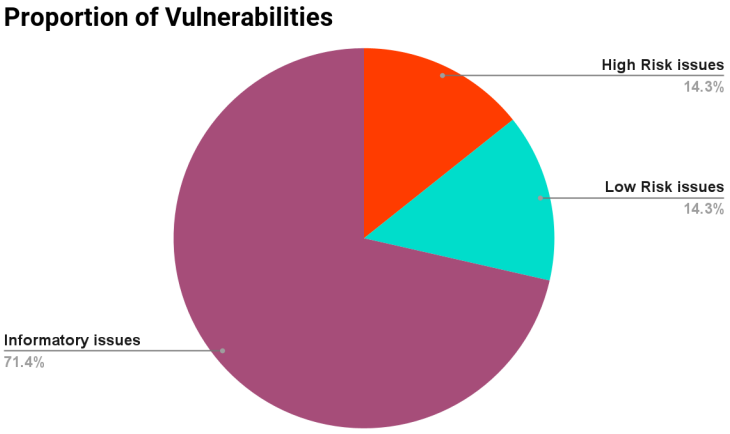
The analysis indicates that some of the functionalities in the contracts audited are **not working properly**.

Our team performed a technique called “Filtered Audit”, where the contract was separately audited by two individuals. After their thorough and rigorous process of manual testing, an automated review was carried out using Mythril, MythX and Slither. All the flags raised were manually reviewed and re-tested.



Our team found:

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



Findings

Critical-risk issues

No issues were found

High-risk issues

1. **No Minting and Burning in ePING**
   File: ePING.sol
   _tTotal represents the totalSupply. Initially on the second chain _tTotal is set to 0 in the constructor. _tTotal is used to calculate the currentRate and currentRate is used while transferring and burning. Setting _tTotal to zero gives math errors. While minting _rTotal is so high (~uint(0 => ~MAX value of uint256)). Adding any value to _rTotal will cause underflow.
   NOTE: Please carefully set the _rTotal. Also while minting do not change the currentRate, as it will be reflected in other users' balances

Medium-risk issues

No issues were found.

Low-risk issues

1. **GasFee Griefing Attack**
   Assuming that the relayer server (owned by the client) pays the gasFee to mint/unlock tokens on the alternate bridge. As there is no min_amount limit on bridging tokens. Consider the following Attack Scenario
   Attack Scenario:
   1. User deposits 0/1 wei amount of token on BscBridge.
   2. Relayer server picks-up the deposit event and calls the mint function on etheremBridge.
   3. User receives 0/1 wei tokens. And Relayer pays the gas.
   4. As gas fees are high, we can deduce that 1 wei tokens < gasFee (paid by relayer).
   Remedy:
   1. Charge gas fee from the user. While bridging the tokens, mint amount - gasFeeEquivalent on alternate chain.
   2. Put a minimum threshold on the amount of tokens to be bridged.

Informatory issues and Optimization

1. transactionID is used while depositing into the bridges. transactionID is not verified in the smart contracts. It is used when the deposit event is emitted. We advise you to validate transactionID at the backend (relayer servers).
2. Destination chainId should be used while depositing. This will improve user experience if multiple chains are supported for bridging in future.
3. Remove console.log from smart contract.
4. IToken.sol has a wrong implementation of interface. Use Openzeppelin's ERC20 interface.
5. Fees are deducted on every transfer. Please make sure that the bridge addresses are excluded from the fees.

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.