Zbra Domains Audit

Core Contracts

Nov 8th, 2021

Audited By: BlockChainSights (BCS)

1. Summary

1.1 Introduction

Zbra is a name service with a magic in-protocol pricing mechanism that provides high liquidity and benefits both early birds and residents moving in in 2050.

Zbra provides the a new rent-base registrar for the .zbra top-level domain. Instead of providing a fixed price, Zbra provides a demand-based recurring pricing. More discussion about domains pricing can be found at Vitalik's posts.

1.2 Project Summary

Project Name	Zbra Land
Client Name	Zbra Name Service
Description	Grab .zbra domains and get profit!
Client Contact	Fisher Z (zebraweb3@gmail.com)
Version	v1.0
Codebase	Github

1.3 Audit Summary

The focus of the audit was to verify that the smart contract system is secure and working according to its specifications.

There are other components of the pending upgrade that were *not* part of this audit.

Delivery Date	Nov. 2022
Method of Audit	Static Analysis, Manual Review
Tool based audits	Ethlint, Hardhat
Auditors	BlockChainSights (BCS)

2. Review Details

2.1 Architecture

Zbra has three principal components: the zbra domain name services (royalties NFT), referral program and ChicagoPot1830. Zbra provides methods to decide a demand-based pricing.

2.2 External Call - Refund Ethers to Buyer

Description:

`Controller.register` `Controller.grab` `Controller.renew` transaction will fail if the contract buyer pays extra fee and needs a refund.

Example:

The buyer can be either EOA account or Smart Contract accounts. We need to make sure the Smart Contract accounts also works.

```
// Refund any extra payment
  if(msg.value > cost) {
      msg.sender.transfer(msg.value - cost);
  }
```

Remediation:

Send Ether to other contracts by `call` (forward all gas or set gas, returns bool). `call` in combination with re-entrancy guard is the recommended method to use.

Guard against re-entrancy by

- Use the Checks-Effects-Interactions Pattern - making all state changes before calling other contracts
- using re-entrancy guard modifier

2.3 Denial of Service

Description:

`Controller.grab` transaction will fail if the previous owner reject to receive Ether, especially the previous owner is the smart contracts who rejects the Ether.

External calls can fail accidentally or deliberately. To do this, the attacker can issue several transactions which will consume the entire gas limit, with a high enough gas

price to be included as soon as the next block is mined. No gas price can guarantee inclusion in the block, but the higher the price is, the higher is the chance.

Example:

To minimize the damage caused by such failures, it is better to isolate each external call into its own transaction that can be initiated by the recipient of the call. This is especially relevant for payments, where it is better to let users withdraw funds rather than push funds to them automatically. (This also reduces the chance of problems with the gas limit)

```
// Send back previous amount
payable(previousOwner).transfer(refundAmount);
```

Remediation:

Added a pull payment system. If the previous contract owner fails to receive the grab refund, add the amount to user balance for their withdrawals.

2.4 Front Running - Registration Displacement

Description:

`Controller.register` `Controller.grab` is vulnerable to front running. Since all transactions are visible in the mempool for a short while before being executed, observers of the network can see and react to an action before it is included in a block

Example:

1. Alice calls `register(name)`.
2. Eve observes this transaction in the transaction pool.
3. Eve submits `register(name)` with a higher gas price and wins the race.

Zbra Team Answer:

This is allowed by design. By design there is no commit/reveal scheme in Zbra Phase ChicagoPot1830. During the first phase (Chicago1830), the focus is on population growth. Alice can register another new name, or grab the previous name.

2.5 Front Running - Pot Suppression

Description:

When the countdown reaches 0, the ChicagoPot1830 stops increasing and will be distributed.

In a suppression attack, a.k.a *Block Stuffing* attacks, after Mallory runs her function, she tries to delay Alice from running her function.

The best remediation is to remove the benefit of front-running in your application, mainly by removing the importance of transaction ordering or time. The other option is to mitigate the cost of front-running by specifying a maximum or minimum acceptable price range on a trade, thereby limiting price slippage.

Zbra Team Answer:

To make it healthier, we set the pot end time as 3 minutes to prevent lower smooth the *Block Stuffing* attacks.

3 Audit Measurement

3.1 Test Coverage

Testing is implemented using Hardhat, and Mocha is the test runner.

The test environment is set by Hardhat Network, a local Ethereum network designed for development. The test is covered by automated tests. 13 groups of tests are included for the contract, and they all pass.

The code coverage numbers for the tests are unable to obtain, while the testing covers most of code branches.

3.2 Tool based analysis

The issues from the tool based analysis have been reviewed.

ETHLINT

Analyse Solidity for Style and Security

We used Ethlint, which is an open source project for linting Solidity code. Only security-related issues were reviewed by the audit team. The raw output of the Ethlint vulnerability scan are digested, and relevant issues have been listed in the previous chapter.

4 Disclaimer

BlockChainSights (BCS) reports are not, nor should be considered, an "endorsement" or "disapproval" of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any "product" or "asset" created by any team or project that contracts BCS to perform a security review.

BCS Reports do not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

BCS Reports should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

BCS Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

The content contained in the Reports is current as of the date appearing on the Report and is subject to change without notice. Unless indicated otherwise, by BCS. Blockchain technology and cryptographic assets present a high level of ongoing risk. BCS's position is that each company and individual are responsible for their own due diligence and continuous security. BCS's goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.