MythX

REPORT 60FE04A052E9EC0019963024

Created	Mon Jul 26 2021 00:41:04 GMT+0000 (Coordinated Universal Time)
Number of analyses	1
User	60c9637f43f2c3c61312dd0c

REPORT SUMMARY

Analyses ID	Main source file	Detected vulnerabilities
d770979c-040f-459b-9eff-8267ff4bdea0	/contracts/wagerfactory.sol	5

Analysis d770979c-040f-459b-9eff-8267ff4bdea0

MythX

Started	Mon Jul 26 2021 00:41:10 GMT+0000 (Coordinated Universal Time)
Finished	Mon Jul 26 2021 00:43:25 GMT+0000 (Coordinated Universal Time)
Mode	Quick
Client Tool	Mythx-Vscode-Extension
Main Source File	/Contracts/Wagerfactory.Sol

DETECTED VULNERABILITIES

HIGH	MEDIUM	LOW
0	0	5

ISSUES

LOW SWC-100

Function visibility is not set.

The function definition of "null" lacks a visibility specifier. Note that the compiler assumes "public" visibility by default. Function visibility should always be specified explicitly to assure correctness of the code and improve readability.

Source file

/contracts/wagerfactory.sol

Locations

```

/* CONSTRUCTOR */
constructor() {
RandomNumberConsumer randomNumberConsumer = new RandomNumberConsumer(address(this));
randomNumberConsumerAddress = address(randomNumberConsumer);
}

/* FUNCTIONS */
```

LOW SWC-103

A floating pragma is set.

The current pragma Solidity directive is ""^0.8.0"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Source file

/contracts/wagerfactory.sol

Locations

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

import "./Wager.sol";
```

LOW

SWC-107

Read of persistent state following external call.

The contract account state is accessed after an external call. To prevent reentrancy issues, consider accessing the state only before the call, especially if the callee is untrusted. Alternatively, a reentrancy lock can be used to prevent untrusted callees from re-entering the contract in an intermediate state.

Source file

`/contracts/wagerfactory.sol`

Locations

```
Wager newWager = new Wager(msg.sender, address(this));
address wagerAddress = address(newWager);
wagerAddresses.push(wagerAddress); // Update with new contract address

emit WagerCreated(wagerAddress, address(this));
```

LOW

SWC-108

State variable visibility is not set.

It is best practice to set the visibility of state variables explicitly. The default visibility for "rngMapping" is internal. Other possible visibility settings are public and private.

Source file

`/contracts/wagerfactory.sol`

Locations

```
address public randomNumberConsumerAddress;
uint16 private randomNumber;
mapping(bytes32 => address) rngMapping;

/* EVENTS */
```

LOW

SWC-120

Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Source file

`/contracts/wagerfactory.sol`

Locations

```
// Using Chainlink VRF Oracle; Random already contains LINK
bytes32 requestId = RandomNumberConsumer(randomNumberConsumerAddress).getRandomNumber(
uint(keccak256(abi.encodePacked(block.timestamp, block.number, block.difficulty)))
);
```