SALUS SECURITY

JAN 2024

CODE SECURITY ASSESSMENT

POLYHEDRA

Overview

Project Summary

- Name: Polyhedra - Bitcoin Oracle
- Platform: EVM-compatible Chains
- Language: Solidity
- Repository: https://github.com/zkBridge-integration/bitcoin-oracle-audit
- Audit Scope: See Appendix - 1

Project Dashboard

Application Summary

Name	Polyhedra - Bitcoin Oracle
Version	v2
Type	Solidity
Date	Jan 22 2024
Logs	Jan 18 2024; Jan 22 2024

Vulnerability Summary

Total High-Severity issues	0
Total Medium-Severity issues	0
Total Low-Severity issues	1
Total informational issues	1
Total	2

Contact

E-mail: support@salusec.io

Risk Level Description

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients' reputations or serious financial implications for clients and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to a moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
Informational	The issue does not pose an immediate risk, but is relevant to security best practices or defense in depth.

Content

Introduction

1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (https://t.me/salusec), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for included (but are not limited to):

- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

Findings

2.1 Summary of Findings

ID	Title	Severity	Category	Status
1	Possible rejection of the initial block leading to DOS risks	Low	Business Logic	Resolved
2	Redundant code	Informational	Redundancy	Acknowledged

2.2 Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

1. Possible rejection of the initial block leading to DOS risks	
Severity: Low	Category: Business Logic
Target: - src/BitcoinOracle.sol	

Description

src/BitcoinOracle.sol:L50-L88

```
constructor(uint256 _initBlockHeight, bytes memory _initHeader, bool _checkPoW) {
    ...
    blockHeaders[blockHash] = Header({
        prevBlock: prevBlock,
        merkleRoot: merkleRoot,
        version: version,
        timestamp: timestamp,
        bits: bits,
        nonce: nonce,
        height: uint64(_initBlockHeight),
        isCanonical: true,
        chainWorkSinceInitBlock: bitsToWork(bits)
    });
    latestBlockHash = blockHash;
    heightToHash[_initBlockHeight] = blockHash;
    firstBlockHash = blockHash;
    initBlockHeight = _initBlockHeight;

    emit NewBlockHeader(blockHash, _initBlockHeight, _initHeader, true);
}
```

If the BitcoinOracle contract uses, as its initial block, a block that was initially mined but later rejected by the network (due to being discarded on a shorter chain), no new blocks will be generated after that block.
This leads to the system being unable to add new blocks, causing a denial-of-service (DoS) situation for the contract.

Recommendation

It is recommended to use the block that is at least 6 blocks ago as the init block when deploying the contract.

Status

The team has resolved this issue in commit 780d8b7.

2.3 Informational Findings

2. Redundant code	
Severity: Informational	Category: Redundancy
Target: - src/BitcoinOracle.sol	

Description

src/BitcoinOracle.sol:L42-L43

```
// This is used to disable PoW check for testing purpose. Must be true in a production deployment.
bool public immutable checkPoW;
```

In the codebase, there is a variable named "*checkPoW*" that is exclusively used for testing purposes. During the actual deployment process, it is recommended to remove this variable and adjust the associated logic to ensure the proper execution of PoW checks.

Recommendation

It is recommended to remove variables that are only used for testing.

Status

This issue has been acknowledged by the team.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit 7b23f8e:

File	SHA-1 hash
BitcoinOracle.sol	4f6adfa100eb1310a3d62c817ff759bd3a909a6f
BytesLib.sol	d87ffc729628ebf36a54d9c4e66bc5cf4ce4b5f9
Endian.sol	1fcf68abcf08dc52451d93dba51e8f86310ce756