DECENTRALIZED APPLICATION CODE REVIEW AND SECURITY ANALYSIS REPORT

Customer: Wormhole

Date: October 25th, 2022

ver. 2

Hacken OÜ
Parda 4, Kesklinn, Tallinn,
10151 Harju Maakond, Eesti,
Kesklinna, Estonia
support@hacken.io

Document

Name	Decentralized Application Code Review and Security Analysis Report for Wormhole.
Approved By	Andrew Matiukhin | CTO at Hacken OÜ
Type	Server, Oracle
Methods	Architecture Review, Functional Testing, Computer-Aided Verification, Manual Review
Website	wormhole.com
Timeline	03 OCTOBER 2022 - 25 OCTOBER 2022
Changelog	11 OCTOBER 2022 - Review[1] 25 OCTOBER 2022 - Remediation[2]

Table of contents

Introduction

Hacken OÜ (Consultant) was contracted by Wormhole (Customer) to conduct a Decentralized Application Code Review and Security Analysis. This report presents the security assessment findings of the Customer's applications.

Scope

The scope of the project is applications in the following repositories:

1. https://github.com/wormhole-foundation/wormhole

    - **Subdirectory:** node/pkg/watchers/near
    - **Platform:** Go
    - **Commit:**
      b622e76e8b3baf077ec8020a8452ec22d3ee11df (Review[1])
      5a0b33699423027c27ac5afaf03fff1a2caecede (Remediation[2])

These applications were scanned for commonly known and more specific issues. Considered items include but are not limited to:

- Overconfidence in a node (or node provider)
- Failure to account for blockchain branching out
- Incorrect validation of ENS records
- Weak authentication via message signing
- Unsafe private key storage
- XSS/SQL injections from the blockchain data
  Misuse of checksum addresses
- Blockchain data inconsistency
- Incorrect integration with a smart contract and/or blockchain platform
- Usage of wrong data types
- Application architecture
- Repository consistency
- Code style consistency
- Deprecated, vulnerable, or outdated Web3 libraries

System Overview

The following interactions between the applications and blockchain platforms were identified and reviewed.

NEAR Watcher

This application integrates with the NEAR blockchain platform. Its primary purpose is to monitor, validate and propagate smart contract events.

Functionality

1. Blockchain node polling
2. Transaction parsing and events validation
3. Publishing validated events and other blockchain-related information

Executive Summary

This report presents the findings of a code review and security analysis conducted between October 03, 2022 - October 25, 2022.

The purpose of this engagement was to evaluate the stability and security of the application against the best practices and possible attack vectors.

The score is based only on the applications that were subject to “Code Review and Security Analysis” (see Scope).

For more details about the scoring, see Methodology.

Documentation quality

Technical documentation and functional requirements were sufficient.

The documentation quality score is **10** out of **10**.

Code quality

The code is well-written, easy to understand and contains helpful comments. Core functionality is covered by tests that were implemented after the initial review.

The code quality score is **10** out of **10**.

Architecture quality

There are no apparent design flaws.

The architecture quality score is **10** out of **10**.

Security and Stability

As a result of the audit, security engineers found **1** low severity issue that was deemed to be mitigated after the initial review.

A detailed description of issues can be found in the Issue Overview section.

The security and stability score is **10** out of **10**.

Summary

According to the assessment, the Customer's applications have the following score: **10**

Checked Items

We have reviewed decentralized applications for commonly known and more specific vulnerabilities. Here are some of the items that were considered:

#	Item	Status
1	**Private keys are safely used and stored** Private keys should be encrypted, stored in a secure secret manager, provided to the application with a secure method.	Not Relevant
2	**Strong authentication via message signing** A signed authentication message should be exclusive to the application. It should have an expiration date and time-specific nonce provided by the system.	Not Relevant
3	**Blockchain branching out is accounted for** The application should wait an adequate number of blocks to reduce the risk of a chain re-organization.	Passed
4	**Healthy integration with the blockchain nodes**	Passed
4.1	**Blockchain node redundancy** The application should not rely on a single blockchain data provider for a trusted and stable connectivity.	Passed
4.2	**Low-trust integration with the blockchain nodes** The application should expect that each interaction with a blockchain data provider can fail and handle these cases accordingly.	Passed
5	**Blockchain data is sanitized** The application should not trust the inputs from the blockchain when they can be considered the same as user inputs.	Passed
6	**Secure integration with the blockchain**	Passed
6.1	**Events are validated** The application should check the event origin, its type and structure.	Passed
6.2	**Proper mapping of types** Blockchain-specific types (uint256, bytes, etc.) should be mapped to the correct language-specific types without a data loss.	Passed

6.3	**Error-proof data indexing** The application should not silently skip anything it could not process at some point in time. It should have a retry mechanism or sufficient logging.	Passed
6.4	**Secure integration with the smart contracts** The application should follow the smart contract specification and expect all relevant and possible outcomes.	Passed
7	**Dependencies are up-to-date and not vulnerable**	Passed

Severity Definitions

Risk Level	Description
Critical	Critical issues are usually straightforward to exploit, can lead to asset loss, data manipulations, or greatly impact the application's stability.
High	High-level issues are difficult to exploit. However, they also have a significant impact on the application execution.
Medium	Medium-level issues are important to fix. However, they cannot lead to asset loss or data manipulations.
Low	Low-level issues are mostly related to outdated, unused code snippets that cannot significantly impact execution.

Issue Overview

■ Low

1. Shard Assumption in the Observation Request Processing

For observation requests, Wormhole account is always used as the sender. This is an issue already acknowledged in code comments. It can lead to observation requests not processing properly when using a transaction from a different shard. It does not affect the user or regular functions of the application.

watcher.go

```
// TODO !!!! IMPORTANT !!!! e.wormholeContract is not the correct value for senderAccountId.
// This may work for now, but eventually we could end up hitting the wrong shard, leading to errors.
// we also need to know the block height. I think if we don't know it we could just set it to 0, but need to think more about that.
job := newTransactionProcessingJob(txHash, e.wormholeAccount)
e.transactionProcessingQueue.Schedule(job, time.Now())
```

Source: watcher.go

Recommendation: Supply the sender account id in the request.

Status: Mitigated

Mitigation: This issue is acknowledged by the Customer and does not affect the security or stability of the application.

Disclaimers

Hacken Disclaimer

The application given for the audit has been analyzed by the best industry practices at the date of this report, in relation to cybersecurity vulnerabilities and issues in the application's source code, the details of which are disclosed in this report (Source Code); the Source Code compilation, deployment, and functionality (performing the intended functions).

The audit makes no statements or warranties on the security of the code. It also cannot be considered a sufficient assessment regarding the utility and safety of the code, bug-free status, or any other contract statements. 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 the security of the decentralized application.

Technical Disclaimer

Decentralized applications are closely integrated with a blockchain platform. The platform, its programming language, and other software related to the application can have vulnerabilities that can lead to hacks. Thus, the audit cannot guarantee the explicit security of the audited application.