



Umi Public Report

PROJECT: Umi Review
Fall 2021

Prepared For:

Umi Protocol
up@umi.top

Prepared By:

Jonathan Haas | Bramah Systems, LLC.
jonathan@bramah.systems



Table of Contents

Executive Summary	3
Scope of Engagement	3
Engagement Goals	3
Protocol Specification	3
Overall Assessment	3
Timeliness of Content	5
General Recommendations	6
Toolset Warnings	6
Overview	6
Compilation Warnings	6
Test Coverage	6
Static Analysis Coverage	6
Directory Structure	7



Umi Security Review

Executive Summary

Scope of Engagement

Bramah Systems, LLC was engaged in Fall of 2021 to perform a comprehensive security review of the Umi Protocol Golang repository. Our review was conducted over a period of five business days by a member of Bramah Systems, LLC. executive staff.

Bramah's review pertains to Go code (*.go) as of commit **4d7f004b930454182b73a578d9d45edaebf11929**.

Engagement Goals

The primary scope of the engagement was to evaluate and establish the overall security of the Umi Protocol system, with a specific focus on trading actions. Notably, this report does not describe usability or any operational aspects of the system in any way. In specific, the engagement sought to answer the following questions (determined conclusions in bold):

- Is it possible for an attacker to manipulate the code?
 - **There does not appear to be a methodology via which an attacker would be able to manipulate the running executable without prior local privileged access.**
- Does the Go code match the specification as provided?
 - **Code appears to match the provided specification.**
- Is there a way to interfere with the software mechanisms?
 - **It does not appear there is a mechanism to interfere with the software mechanisms.**
- Are the arithmetic calculations trustworthy?
 - **Calculations appear to match the provided specification.**

Protocol Specification

A basic specification document was compiled by the review team based upon review of the Umi Protocol code and discussion with the team.



Overall Assessment

Bramah Systems was engaged to evaluate and identify multiple security concerns in the codebase of the Umi architecture. During the course of our engagement, Bramah Systems denoted no instances wherein the software deviated from established best practices and procedures of secure software development. The codebase benefits from detailed code comments throughout, which allowed for Bramah to review the codebase rapidly and without a deeper formal specification.



Disclaimer

As of the date of publication, the information provided in this report reflects the presently held, commercially reasonable understanding of Bramah Systems, LLC.'s knowledge of security patterns as they relate to the Umi Protocol Protocol, with the understanding that distributed ledger technologies (“DLT”) remain under frequent and continual development, and resultantly carry with them unknown technical risks and flaws. The scope of the review provided herein is limited solely to items denoted within “Scope of Engagement” and contained within “Directory Structure”. The report does NOT cover, review, or opine upon security considerations unique to the Go compiler, tools used in the development of the protocol, or distributed ledger technologies themselves, or to any other matters not specifically covered in this report.

The contents of this report must NOT be construed as investment advice or advice of any other kind. This report does NOT have any bearing upon the potential economics of the Umi Protocol protocol or any other relevant product, service or asset of Umi Protocol or otherwise. This report is not and should not be relied upon by Umi Protocol or any reader of this report as any form of financial, tax, legal, regulatory, or other advice.

To the full extent permissible by applicable law, Bramah Systems, LLC. disclaims all warranties, express or implied. The information in this report is provided “as is” without warranty, representation, or guarantee of any kind, including the accuracy of the information provided. Bramah Systems, LLC. makes no warranties, representations, or guarantees about the Umi Protocol Protocol. Use of this report and/or any of the information provided herein is at the users sole risk, and Bramah Systems, LLC. hereby disclaims, and each user of this report hereby waives, releases, and holds Bramah Systems, LLC. harmless from, any and all liability, damage, expense, or harm (actual, threatened, or claimed) from such use.

Timeliness of Content

All content within this report is presented only as of the date published or indicated, to the commercially reasonable knowledge of Bramah Systems, LLC. as of such date, and may be superseded by subsequent events or for other reasons. The content contained within this report is subject to change without notice. Bramah Systems, LLC. does not guarantee or warrant the accuracy or timeliness of any of the content contained within this report, whether accessed through digital means or otherwise.

Bramah Systems, LLC. is not responsible for setting individual browser cache settings nor can it ensure any parties beyond those individuals directly listed within this report are receiving the most recent content as reasonably understood by Bramah Systems, LLC. as of the date this report is provided to such individuals.



General Recommendations

Best Practices & Go Development Guidelines

The Umi codebase was not found to have any vulnerabilities of note by Bramah Systems nor the Umi team. Areas of suggested improvement were incorporated into prior code updates.

Toolset Warnings

Unique to the Umi Protocol Protocol

Overview

In addition to our manual review, our process involves utilizing concolic analysis and dynamic testing in order to perform additional verification of the presence security vulnerabilities. An additional part of this review phase consists of reviewing any automated unit testing frameworks that exist.

The following sections detail warnings generated by the automated tools and confirmation of false positives where applicable, in addition to findings generated through manual inspection.

Compilation Warnings

No warnings were found at time of compilation that presented material concern.

Test Coverage

The contract repository possesses substantial unit test coverage throughout. This testing provides a variety of unit tests which encompass the various operational stages of the protocol

Static Analysis Coverage

The contract repository underwent heavy scrutiny with multiple static analysis agents, including:

- [Semgrep](#)

In each case, the team had either mitigated relevant concerns raised by each of these tools or provided adequate justification for the risk.



Directory Structure

At time of review, the directory structure of the Umi Protocol appeared as it does below. Our review, at request of Umi Protocol, covers the Go code (*.sol) as of commit [4d7f004b930454182b73a578d9d45edaebf11929](#).

```
|— LICENSE
|— README.md
|— build
|  └─ package
|     └─ docker
|        └─ Dockerfile
|— cmd
|  └─ umid
|     └─ main.go
|— go.mod
|— go.sum
└─ pkg
   └─ config
      └─ config.go
   └─ events
      └─ events.go
   └─ generator
      └─ generator.go
   └─ ledger
      └─ account.go
      └─ confirmer.go
      └─ confirmer_legacy.go
      └─ ledger.go
      └─ structure.go
```



```
|— legacy
| |— client.go
| |— fetcher.go
| |— pusher.go
|— openlibm
| |— pow.go
| |— pow_test.go
|— restapi
| |— handler
| | |— account.go
| | |— address_create.go
| | |— block.go
| | |— event.go
| | |— helpers.go
| | |— helpers_test.go
| | |— mempool.go
| | |— structure.go
| | |— transaction.go
| | |— transaction_create.go
| |— restapi.go
| |— router.go
| |— router_test.go
|— storage
| |— blockchain.go
| |— blockchain_memory.go
| |— blockchain_mmap.go
| |— blockchain_test.go
| |— filesystem.go
| |— filesystem_test.go
| |— genesis.go
| |— index.go
```




```
| ├── mempool.go
| └── mempool_test.go
├── syncer
| └── syncer.go
└── umi
    ├── account.go
    ├── address.go
    ├── bech32.go
    ├── block.go
    ├── block_legacy.go
    ├── crypto.go
    ├── hash.go
    ├── merkle.go
    ├── prefix.go
    ├── transaction.go
    └── transaction_verify.go
```

17 directories, 54 files