HACKEN

PoR Implementation Security Assessment

Gate.io

January 3, 2024

Hacken OU
Parda 4, Kesklinn, Tallinn
10151 Harju Maakond, Eesti
Kesklinna, Estonia

Repositories:	https://github.com/gateio/proof-of-reserves
Commit:	ea647e23bad94e6417df3f05277c6caa8959d0f4
Auditors:	Luciano Ciattaglia Sofiane Akermoun Nino Lipartiia Bartosz Barwikowski
References:	• Customer's Website • https://www.gate.io/proof-of-reserves/ • https://www.gate.io/learn/articles/gate-io-proof-of-reserve-upgrade/855 • https://www.gate.io/article/33123

Goal and Objectives of Engagement

Hacken team analyzes the documentation, repository codebase, code and architecture quality, new releases tags functionalities and performs necessary checks against known vulnerabilities.

The assessment's goal is to determine whether the code is vulnerable to known attacks or malicious code and to ensure that there are no issues, build, deployment, or architecture flaws.

Hacken blockchain protocol and security analysis methodology

Hacken OU
Parda 4, Kesklinn, Tallinn
10151 Harju Maakond, Eesti
Kesklinna, Estonia

Project Summary

This project is a derivative of the "zkmerkle-proof-of-solvency," retaining much of the original logic but introducing significant structural modifications in file organization and naming conventions.

Key aspects of the project include:

1. **Dependency Management:** The project relies on 1,157 dependencies, each verified through checksum validation. A comprehensive security audit revealed 42 vulnerabilities across these dependencies, with 16 posing risks due to publicly available exploits. The severity of these vulnerabilities is categorized as 22 high-impact and 20 medium-impact.

2. **Cryptographic Frameworks and Structures:**
    - The project utilizes an outdated fork of GNARK (version 0.7.0) to construct cryptographic circuits.
    - For hashing user data and the Sparse Merkle Tree (SMT) structure, it employs the Poseidon hash function with the BN254 curve.
    - The SMT, crucial for storing hashes, is implemented using the BSMT library. The tree's maximum depth is set at 28, enabling the Proof Of Solvency system to accommodate over 250 million users.

3. **Code Quality and Documentation:**
    - The code is well-organized and concise, with minimal commented lines, enhancing readability and maintainability.
    - Despite its clean structure, the project currently has 0% test coverage, highlighting a potential area for improvement in terms of code reliability.
    - The README.md file provides detailed instructions for tool setup and operation, although it lacks insight into the motivations behind the chosen computational circuits for user batch commitments.

4. **Error Handling:** The project employs the Panic strategy for main function error handling. This approach leads to tool crashes accompanied by a stack trace whenever an error occurs, which can be useful for debugging but may affect robustness in production environments.

5. **Testing and Sample Data:** For manual testing, sample user data, including balance sheets, is provided. This aids in evaluating the tools in a controlled environment with real-world data scenarios.

Overall, the project stands as a testament to a strong foundation in cryptographic implementation and exceptional code organization. Its current state reflects a well-considered balance between functionality and complexity, offering a comprehensive setup for users with its detailed documentation and provided sample data. The project's existing framework and features already contribute significantly to the field, demonstrating a clear understanding and application of key principles in blockchain technology and cryptographic systems.

Findings

[Critical]
No critical severity issues were found

[High]
No high severity issues were found

[Medium]
No medium severity issues were found

[Low]
No low severity issues were found

[Informational] Merkle Root hash integrity

When users download the Merkle tree and each user config from the frontend, the Merkle root hash is included in the user_config.json file, but there is no way to check the integrity of this hash across all Gate.io users in order to be sure that this root hash wasn't tampered depending on the client IP or other parameters of the users.


```
{
  "AccountIndex": 9,
  "AccountIdHash": "0000041cb7323211d0b356c2fe6e79fdaf0c27d74b3bb1a4635942f9ae92145b",
  "Root": "29591ef3a9ed02605edd6ab14f5dd49e7dbe0d03e72a27383f929ef3efb7514f",
  "Assets": [{"Index":7,"Equity":123456000,"Debt":0}],
  "Proof": ["DrPpFsm4/5HntRTf8M3dbgpdrxq3Q8lZkB2ngysW2js=","G1WgD/CvmGApQgmIX0rE0BlSifkw6IfNwY
  "TotalEquity": 123456000,
  "TotalDebt": 0
}
```


To counteract this, the Merkle root should be signed by a trusted third-party auditor or be published on the blockchain as a public bulletin board, so users can easily verify the transaction's inclusion and the validity of the Merkle root hash they got from their user_config.json. It should be done in a single transaction, which will be easy to detect. It's also possible to address this issues by publishing the hash root in a social media that the proved doesn't control.

[Informational] Vulnerable and Outdated gnark Dependency

The current implementation of the ZKP system relies on an outdated version of the gnark library, specifically version v0.7.1. This poses potential security risks and efficiency drawbacks, as the latest version of gnark, v0.9.1, includes several critical security fixes and performance improvements.

It is strongly recommended to update gnark to the latest version, v0.9.1. Considering that gnark uses semantic versioning, updating from v0.7.1 to v0.9.1 should not introduce breaking changes, making the update process smoother. This upgrade will ensure that we leverage the latest security patches and performance improvements.

For a detailed list of changes, including security fixes and efficiency enhancements in the newer version, please consult the release notes available at: gnark Releases on GitHub.

[Informational] Vulnerabilities in dependencies

- Improper Signature Verification affecting golang.org/x/crypto/ssh package
- Denial of Service (DoS) affecting golang.org/x/net/html package
- NULL Pointer Dereference affecting golang.org/x/net/html package
- Authorization Bypass Through User-Controlled Key affecting github.com/emicklei/go-restful/v3 package
- Authorization Bypass affecting github.com/emicklei/go-restful/v3 package
- Improper Input Validation affecting github.com/ethereum/go-ethereum/core package
- Insecure Randomness affecting github.com/satori/go.uuid package

Upon thorough examination of each exploit associated with the identified vulnerabilities, it was determined that none of the vulnerable functions are actively utilized in the current scope of the project. However, it is advisable to update these dependencies as a proactive measure to entirely mitigate any potential risks they might pose in the future.