SOLIDProof

Bring trust into your projects

Blockchain Security | Smart Contract Audits | KYC Development | Marketing

MADE IN GERMANY

GraphLinq Chain

Audit

Security Assessment

15. February, 2023

For

graphlinqchain

Disclaimer

SolidProof.io reports are not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. These reports are not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team. SolidProof.io do not cover testing or auditing the integration with external contract or services (such as Unicrypt, Uniswap, PancakeSwap etc’...)

SolidProof.io Audits do not provide any warranty or guarantee regarding the absolute bug- free nature of the technology analyzed, nor do they provide any indication of the technology proprietors. SolidProof Audits should not be used in any way to make decisions around investment or involvement with any particular project. These reports in no way provide investment advice, nor should be leveraged as investment advice of any sort.

SolidProof.io Reports represent an extensive auditing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology. Blockchain technology and cryptographic assets present a high level of ongoing risk. SolidProof’s position is that each company and individual are responsible for their own due diligence and continuous security. SolidProof in no way claims any guarantee of security or functionality of the technology we agree to analyze.

Date	Description
9. February 2023	• Listing of all the changes after forking • Layout project
10. February 2023	• Checking for vulnerabilities in the GraphLinqChain
13. February 2023	• Automated- /Manual-Security Testing • Summary
15. February 2023	- RPC pentesting in progress -

Network
GraphLinq Chain (mainnet)
Native (fee) coin: GLQ
Chain ID: 614
RPC: https://glq-dataseed.graphlinq.io

Website
https://graphlinq.io/
https://docs.graphlinq.io/graphlinq-chain/networks

Twitter
https://twitter.com/graphlinq_proto

Explorer
https://explorer.graphlinq.io

Status
https://status.graphlinq.io

Description

GraphLinq is a platform for automation and process optimization using Graphs. It provides a powerful and intuitive way for users to automate various tasks and workflows by creating and deploying graphs. The platform is built on a decentralized and EVM-compatible blockchain, with a Proof-of-Stake consensus mechanism, allowing for a more secure and transparent platform for automation.

Project Engagement

During the Date of 9th February 2023, **GraphLinq Team** engaged Solidproof.io to audit smart contracts that they created. The engagement was technical in nature and focused on identifying security flaws in the design and implementation of the contracts. They provided Solidproof.io with access to their code repository and whitepaper.

Logo

Project Link

v1.0

- https://github.com/GraphLinq/GraphLinq-Chain
- Commit: 87bc0deedb82335cca55043568bd0835313c3633
- Date: Feb 14, 2023 at 7:17 PM GMT+1
    - This branch is 9 commits ahead, 39 commits behind ethereum:master.
    - Note: the report has been audited until the Commit above. Extended commits are not part of the audit

Vulnerability & Risk Level

Risk represents the probability that a certain source-threat will exploit vulnerability, and the impact of that event on the organization or system.

Level	Vulnerability	Risk (Required Action)
Critical	A vulnerability that can disrupt the contract functioning in a number of scenarios, or creates a risk that the contract may be broken.	Immediate action to reduce risk level.
High	A vulnerability that affects the desired outcome when using a contract, or provides the opportunity to use a contract in an unintended way.	Implementation of corrective actions as soon aspossible.
Medium	A vulnerability that could affect the desired outcome of executing the contract in a specific scenario.	Implementation of corrective actions in a certain period.
Low	A vulnerability that does not have a significant impact on possible scenarios for the use of the contract and is probably subjective.	Implementation of certain corrective actions or accepting the risk.
Informational	A vulnerability that have informational character but is not effecting any of the code.	An observation that does not determine a level of risk

Auditing Strategy and Techniques Applied

Throughout the review process, care was taken to check the repository for security-related issues, code quality, and compliance with specifications and best practices. To this end, our team of experienced pentesters and smart contract developers reviewed the code line by line and documented any issues discovered.

Methodology

The auditing process follows a routine series of steps:

1. Code review that includes the following:
    i) Review of the specifications, sources, and instructions provided to SolidProof to make sure we understand the size, scope, and functionality of the blockchain.
    ii) Manual review of changed code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
    iii) Comparison with the specification, i.e. checking if the code does what is described in the specifications, sources and instructions provided to SolidProof.

2. Testing and automated analysis that includes the following:
    i) Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    ii) Symbolic execution, which is analyzing a program to determine what inputs causes each part of a program to execute.

3. Best practices review, which is a review of the whole codebase to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

4. Specific, itemized, actionable recommendations to help you take steps to secure the project.

Used Code from other Frameworks/Smart Contracts (direct imports)

External Imported packages

- github.com/ethereum/go-ethereum
- github.com/stretchr/testify/require
- github.com/deckarep/golang-set/v2
- github.com/google/uuid

Internal packages

- context
- errors
- math
  - big
  - rand
- bytes
- testing
- fmt
- golang.org/x/crypto/sha3
- reflect
- encoding
  - Json
  - Strings
  - Hex
  - binary
- Sort
- Sync
  - atomic
- io
- Regexp
- Strconv
- Strings
- Unicode
  - Utf8
- time
- Crypto
  - ecdsa
  - rand
- Ox
  - Exec
- Path
  - Filepath
- Go
  - Format
- Text
  - template
- runtime

Sources in Scope and Tested Files

- Scope of work is to check the modified version of GETH, a Go implementation of Ethereum. We checked especially the modified files of the GETH fork and tested the whole codebase in our own infrastructure.

- The project cannot be listed here because it exists of a huge amount of files. If you want to look over the project size please visit the following url: *https://github.com/GraphLinq/GraphLinq-Chain*

Vulnerabilities of Concern:

We have verified the following known vulnerabilities in the GraphLinq Chain:

1. Considerations for individual nodes participating in the network
2. Amplification attacks (e.g via spoofing make peers transmit data to a target),
3. Eclipse attacks (manipulate a target to connect only to malicious peers)
4. Suitability and correct use of the chosen algorithms
5. DoS attacks
6. Wrong Control Flow and Exception Handling
7. Consensus Bugs
8. Overall checkup (Blockchain Security)

Amplification Attacks

Name	
Is the protocol prone to bandwidth depletion/DDoS attack	No

v1.0

- Comments: The 51% attack, where a majority of processing nodes are hacked preventing new transactions from gaining confirmations would still be hypothetically possible but practically not because in the case of GraphLinq chain, the nodes will be controlled privately.

Eclipse Attacks

Name	
Is the protocol prone to eclipse attack	No

v1.0

- Comments: The eclipse attack is also not feasible right now for the GraphLinq chain because of the Privately Operated nodes but if the GraphLinq team plans to extend the number of nodes and make it public then we have a few suggestions to prevent eclipse attacks:
    - **Increased node connections**: If each node in the network gets connected to a large number of nodes, it will get difficult for the attacker to isolate the target in the network, thereby reducing the possibility of an Eclipse attack.
    - **Random node selection:** The network should be designed in such a way, that each node connects to a random set of nodes when it comes in sync with the network.

Consensus Bug

Name	
Is the protocol prone to consensus bugs that may lead to chain splitting	No

v1.0

- Comments: There are no known consensus bug till date for the GETH client. As GraphLinq is a fork of GETH, the same is applicable here also.

Resource Exhaustion in EVM (DDoS)

Name	
Is the protocol prone to resource exhaustion caused by the VM	No

Audit Results

Critical issues

No critical issues

High issues

No high issues

Medium issues

No medium issues

Low issues

Issue	File	Type	Line
#1	Light/txpool.go Core/vm/evm.go	Type is Validation of the returned Sign of an Int typed value	378 609

Description
During the Call function in the core/vm/evm.go file L175 the call of the Transfer function can cause logical issues because of the passed int type of the value. Comparing it to the declared Transfer function in the core/evm.go file L126 you can see that the amount is type of big.Int that means that a call is valid when the amount is below 0. if the canTransfer not executed in the same line which checking the balance is equal or above 0 that indicates that only if the balance of an address is above or equal to 0 can transfer. In the core/txpool/txpool.go file in L609 there is only a check of the sign is under 0 but the value of 0 is allowed. Just for a little explanation if a value is 0 the Sign function will return 0, if its under 0 it will return -1 and if its above 0 it will return 1. We recommend you the validation of the TX value be done also during block processing. The unsignedness of a tx amount by using an unsigned type can help

Informational issues

No informational issues

Audit Comments
15. February 2023:

- Read whole report for more information
- We highly recommend you to read the claims sections also because it is part of the issues table
- Hence, the issue mentioned in the table above has existed in the GETH codebase, then this will also affect the GraphLinq Chain
- This branch is 9 commits ahead, 39 commits behind ethereum:master.
- Note: the report has been audited until the Commit in the Project link section on page 5. Extended commits are not part of the audit