SALUS SECURITY

JUL 2024

CODE SECURITY ASSESSMENT

LORENZO PROTOCOL

Overview

Project Summary

- Name: Lorenzo Protocol - relayer
- Language: Go
- Repository:
  - https://github.com/Lorenzo-Protocol/lorenzo-relayer
- Audit Range: See Appendix - 1

Project Dashboard

Application Summary

Name	Lorenzo Protocol - relayer
Version	v1
Type	Go
Dates	Jul 05 2024
Logs	Jul 05 2024

Vulnerability Summary

Total High-Severity issues	1
Total Medium-Severity issues	4
Total Low-Severity issues	1
Total informational issues	0
Total	6

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	Critical security issues discovered in btcd version < 0.24.2	High	Business Logic	Pending
2	Lightningnetwork/lnd v0.16.4-beta.rc1 has an issue which could lead to a denial of service	Medium	Configuration	Pending
3	Go lang 1.21.12 should be used as it has multiple security fixes	Medium	Configuration	Pending
4	Lack of handling error when reading CA files before creating a new rpc client	Medium	Business Logic	Pending
5	On SIGINT handling, it doesn't wait for the reporter to be shutdown	Medium	Business Logic	Pending
6	Lack of nil check on zfront of subscription	Low	Business Logic	Pending

2.2 Notable Findings

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

1. Critical security issues discovered in btcd version < 0.24.2	
Severity: High	Category: Configuration
Target: - go.mod	

Description

Bugs and issues related to subtle interactions related to re-orgs and the UTXO set cache exist in btcd with versions lower than 0.24.2. The current version used by Lorenzo is v0.24.0

Full details to be disclosed in 90 days (since 0.24.2 release) as mentioned by btcd contributors.

Recommendation

Consider upgrading to v0.24.2

2. Lightningnetwork/lnd v0.16.4-beta.rc1 has an issue which could lead to a denial of service	
Severity: Medium	Category: Configuration
Target: - go.mod	

Description

A parsing vulnerability in lnd's onion processing logic led to a DoS vector due to excessive memory allocation.

Recommendation

Consider upgrading to v0.17.0-beta

3. Go lang 1.21.12 should be used as it has multiple security fixes	
Severity: Medium	Category: Configuration
Target: - go.mod	

Description

Go v1.21 is used in the relayer. This version could be any version 1.21.X which likely to have multiples secuirty issues that are fixed by latest version, a few examples:

- DoS due to improper 100-continue handling in net/http
- Unexpected behavior from Is methods for IPv4-mapped IPv6 addresses in net/netip
- HTTP/2 CONTINUATION flood in net/http which could lead to DoS.
- Incorrect forwarding of sensitive headers and cookies on HTTP redirect in net/http

Please note. that, when 1.21 version is set, it will take the minor version based on the installed version on the device.

Recommendation

Specify the version in go.mod as 1.21.12

4. Lack of handling error when reading CA files before creating a new rpc client	
Severity: Medium	Category: Business Logic
Target: - btcclient/client_block_subscriber.go - config/bitcoin.go	

Description

When creating a new rpc client, it reads certificate file.

btcclient/client_block_subscriber.go:L90

```
Certificates: cfg.ReadCAFile(),
```

If TLS is disabled, returns nil.

config/bitcoin.go:L69

```
if cfg.DisableClientTLS { return nil }
```

Otherwise, read certificate file. However, If there's an error reading the CA file, it continues with nil returned. This might cause an error when using rpcclient

btcclient/client_block_subscriber.go:L93

```
rpcClient, err := rpcclient.New(connCfg, &notificationHandlers)
```

Recommendation

Before rpcclient.New check if cfg.DisableClientTLS is false and Certificates is nil, then return error.

5. On SIGINT handling, it doesn't wait for the reporter to be shutdown

Severity: Medium	Category: Business Logic
Target: - cmd/lrzrelayer/cmd/reporter.go	

Description

On SIGINT handling, it waits for the btc client to be shutdown

cmd/lrzrelayer/cmd/reporter.go:L93-L94

```
addInterruptHandler(func() {
    rootLogger.Info("Stopping BTC client...")
    btcClient.Stop()
    btcClient.WaitForShutdown()
    rootLogger.Info("BTC client shutdown")
})
```

However, it doesn't wait for the reporter to be shutdown

cmd/lrzrelayer/cmd/reporter.go:L86-L89

```
rootLogger.Info("Stopping reporter...")
vigilantReporter.Stop()
rootLogger.Info("Reporter shutdown")
```

Recommendation

Consider adding logic to wait for repoter to be shutdown

6. Lack of nil check on zfront of subscription

Severity: Low	Category: Business Logic
Target: - zmq/subscribe.go	

Description

c.subs.zfront should be checked if nil before calling SendMessage.

zmq/subscribe.go:L52-L53

```
_, err = c.subs.zfront.SendMessage("subscribe", "sequence")
```

Recommendation

Consider adding the correct checks.

2.3 Informational Findings

No informational issues were found.

Appendix

Appendix 1 - Files in Scope

This audit covered the following files in commit f573d1e:

File	SHA-1 hash
btcclient/client.go	22fcd7f64c6c9cf680df36731c81c581260f49f5
btcclient/client_block_subscriber.go	a41bea626a9372cddf3244193363a30f5b6e39c5
btcclient/notifier.go	25b14b541f64024bca6e56cf8f5322eb2bf74ac6
btcclient/query.go	82656cfd10c6c4f32613f1573f3a5094294286b1
config/bitcoin.go	129929734e3656d19f69c6876fb6f2b1e6bf1047
config/common.go	e39de8e2a08d6d23d795a2f7932e576f8fcd4dd8
config/config.go	b1f8fbaf5620dc772f5da7df32dfb7c892b984ae
config/log.go	81d074df35098d06a236fe7cff6e900dc51b96d1
config/metrics.go	f173979f6c829c2d221205910b5daf0b96a590ca
config/reporter.go	81864c0ffb921eb7480c07e5282968c97d973d41
metrics/prometheus.go	7b6335096bbb7b3e015e0bb65331e1feeead352f
metrics/reporter.go	b08a23952706dcda3333700f2425240e78a78794
netparams/bitcoin.go	91953c7ca894390bc7d91d1fa3646fdd8d041300
reporter/block_handler.go	a71e141fb03d78bff2baf0b6106f211bd1729186
reporter/bootstrapping.go	57055180f5ddd7322b167a068d88363c7810a1bd
reporter/expected_lorenzo_client.go	094ddebe0ecc48451cca0f3b4636152433841bca
reporter/reorg_list.go	651d3aeeda7055f92d4bd7a112f22820c2419054
reporter/reporter.go	e86cc9ec39da4c47283c0da016628ab20c056e70
reporter/utils.go	e86cc9ec39da4c47283c0da016628ab20c056e70
types/blockevent.go	5d3e9766faa452cb45e1f7c8172d7e31c2aa7ca0
types/btccache.go	ee99379bbd98f78b46a59fcda7165956b9df439d
types/btclightclient.go	a9407e9fe5042e852beb16f94e0c26255b29294a
types/errors.go	023f3702a5503fae0b9e7eac9c56fdaf6ed6f1c5

types/indexed_block.go	dbac0af41650ab11c37eebc3c1c8c2711d832083
types/utils.go	ba8ab276bd074fbad716290faedfb347e5413e39
types/utxo.go	0b7ea0de5c1d75a2a03b95bfa50668d5ca350e79
zmq/client.go	90e643aac77397177178d7438e70105106fca6d8
zmq/subscribe.go	6414411912d3698ef09ad8c95d3f27725b83d2c3
cmd/lrzrelayer/main.go	504ab25540bb999a668db9fff1949d3246b67b90
cmd/lrzrelayer/cmd/reporter.go	50f4deee42bd9927fd0e23dc8ae72c2a8496beae
cmd/lrzrelayer/cmd/root.go	14f46a2b5aeddfd13da9a93a330a5bb50a00129b
cmd/lrzrelayer/cmd/utils.go	e2e87104b3e6fb70d0e8893c40d925c17ecf656e