Audit Report

Router Voyager Forwarder

v1.0

April 29, 2024

Table of Contents

License

Disclaimer

This audit has been performed by

Oak Security GmbH

https://oaksecurity.io/
info@oaksecurity.io

Introduction

Purpose of This Report

Oak Security has been engaged by Kailaasa Infotech Pte Ltd to perform a security audit of the Router Voyager Forwarder.

The objectives of the audit are as follows:

1. Determine the correct functioning of the protocol, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine smart contract bugs, which might lead to unexpected behavior.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

Codebase Submitted for the Audit

The audit has been performed on the following target:

Repository	https://github.com/router-protocol/voyager-forwarder
Commit	`ad2e63969a72bd4195961fca6069738659a91f61`
Scope	All files were in scope. Note that commented code has not been audited.
Fixes verified at commit	`16a88a2ef40bcc8b5504e893864eab4709d556e0` Note that only fixes to the issues described in this report have been reviewed at this commit. Any further changes such as additional features have not been reviewed.

Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the code base's intended purpose by reading the available documentation.
2. Automated source code and dependency analysis.
3. Manual line-by-line analysis of the source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
   a. Race condition analysis
   b. Under-/overflow issues
   c. Key management vulnerabilities
4. Report preparation

Functionality Overview

Router protocol is a layer one chain focusing on blockchain interoperability, enabling cross-chain communications.

The forwarder is a component of the protocol that is responsible for relaying messages between different chains.

How to Read This Report

This report classifies the issues found into the following severity categories:

Severity	**Description**
Critical	A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service.
Major	A vulnerability or bug that can affect the correct functioning of the system, lead to incorrect states or denial of service.
Minor	A violation of common best practices or incorrect usage of primitives, which may not currently have a major impact on security, but may do so in the future or introduce inefficiencies.
Informational	Comments and recommendations of design decisions or potential optimizations, that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary. This category may also include opinionated recommendations that the project team might not share.

The status of an issue can be one of the following: **Pending, Acknowledged**, or **Resolved**.

Note that audits are an important step to improving the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**. We include a table with these criteria below.

Note that high complexity or low test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than in a security audit and vice versa.

Code Quality Criteria

The auditor team assesses the codebase's code quality criteria as follows:

Criteria	Status	Comment
Code complexity	**Medium**	-
Code readability and clarity	**Low**	The codebase does not follow the idiomatic Go writing style. There are many outstanding TODO comments throughout the codebase, along with unimplemented functionalities and duplicated code.
Level of documentation	**Medium**	The client provided recorded videos but no detailed documentation was available.
Test coverage	**Low**	There were minimal test cases in the codebase and some of them failed.

Summary of Findings

No	Description	Severity	Status
1	Missing HTTP timeouts enable Slowloris DoS attacks	Critical	Resolved
2	Improper usage of goroutines leads to memory and thread leaks as well as deadlocks	Critical	Partially Resolved
3	Using `BlockHeight` instead of `Timestamp` calculates incorrect transaction expirations	Critical	Resolved
4	NEAR `eventProcessor` disregards user configurations and forces the application to connect to untrusted nodes hosted on AWS	Critical	Resolved
5	The `isTransactionProfitable` function always returns true leading the forwarder's operator to relay transactions at a loss	Critical	Resolved
6	Limitations in `Network` configuration and untrusted hardcoded endpoint usage	Critical	Partially Resolved
7	Improper error handling in `Processor`'s `Start` method stops forwarders' operations	Major	Resolved
8	`SR25519` key generation requires an external untrusted binary	Major	Resolved
9	Unhandled errors in the codebase	Major	Partially Resolved
10	HTTP and GRPC services do not use TLS	Major	Resolved
11	Missing query pagination handling could lead to partial data retrieval	Major	Resolved
12	Unthrottled query retries allow DoS attacks on Tron nodes	Major	Resolved
13	Incorrect usage of named return parameters in `QueryVoyagerEvents` makes the function not exit in case of error	Major	Resolved
14	The execution does not stop if contract ABI is incorrect	Major	Resolved
15	Missing handling of out-of-index errors	Major	Resolved

16	Files are not closed after I/O operations leading to descriptor leakage, memory waste, and potential data corruption	Major	Resolved
17	Private keys are stored in plaintext in the configuration file	Major	Resolved
18	Administrators updating on-chain parameters could cause all connected forwarders to crash	Minor	Acknowledged
19	Health check is not implemented	Minor	Resolved
20	Hardcoded high gas price leads to inefficiencies and potential stop of forwarder operations	Minor	Acknowledged
21	Accessing `nil` leads to panic	Minor	Resolved
22	Inappropriate use of `context.Background` in multiple code instances	Minor	Partially Resolved
23	The application gets stuck if the user executes it with incorrect arguments or with `-h` or `-v` flags	Minor	Acknowledged
24	User-defined `logLevel` is ignored	Minor	Resolved
25	Potential leaked `Ticker`	Minor	Resolved
26	Usage of panics instead of errors	Minor	Resolved
27	Hardcoded gas limit leads to unnecessary forwarder operators' expenses	Minor	Resolved
28	Unzeroized secret data in `Listener` initializers and `EventProcessor`	Minor	Acknowledged
29	Integer overflow when creating big numbers	Minor	Resolved
30	Lack of input file path validation in `ToJSON` function	Minor	Acknowledged
31	Incomplete validation of the `Config` leads to possible unexpected errors and race conditions	Minor	Acknowledged
32	Silent failure for missing chains	Minor	Resolved
33	Inadequate use of unbuffered channels	Informational	Resolved
34	Hardcoded Tron chain ID	Informational	Acknowledged
35	Incorrect usage of `Sleep` slows down the `RequestProcessor`	Informational	Partially Resolved
36	`EncryptAndWriteToFile` writes secret data to a file with unknown access rights	Informational	Acknowledged

37	Unutilized function arguments	Informational	Partially Resolved
38	Incorrect logger labeling	Informational	Resolved
39	Double import of the `logrus` package	Informational	Resolved
40	Type inconsistency for `ChainType` struct parameter	Informational	Resolved
41	Unused variables in various code instances	Informational	Resolved
42	Remove redundant `ChainClient` type from the `fetchTokenPriceList` arguments	Informational	Resolved
43	Incorrect logging level	Informational	Resolved
44	Incorrect struct tag syntax	Informational	Resolved
45	Outdated dependencies	Informational	Acknowledged
46	Usage of deprecated functions	Informational	Partially Resolved
47	Private key fields are not used	Informational	Resolved
48	Hardcoded sleep time ignores `waitPeriod` parameter	Informational	Resolved
49	Miscellaneous comments	Informational	Partially Resolved

Detailed Findings

1. Missing HTTP timeouts enable Slowloris DoS attacks

Severity: Critical

In `cmd/voyager-forwarder/main.go:86`, an HTTP server is instantiated and enabled to listen for incoming requests on port `metricPort`.

However, since no HTTP timeouts are in place to handle idle connections the server is vulnerable to Slowloris Denial-of-Service (DoS) attacks. This attack method operates by transmitting large amounts of data slowly, which might succeed in keeping the connection alive in the event of a timeout, ultimately resulting in a DoS.

Recommendation

We recommend defining timeouts for the HTTP server.

Status: Resolved

2. Improper usage of goroutines leads to memory and thread leaks as well as deadlocks

Severity: Critical

The application is designed as a main process that instantiates several goroutines depending on the configuration parameters. However, they are improperly orchestrated using context and channels.

In `cmd/voyager-forwarder/main.go:105-289`, listeners and event processors are initialized, but it is not possible to stop them gracefully using the implemented mechanisms for the following reasons:

- A parent context in `cmd/voyager-forwarder/main.go:105` is created without a cancellable function
- `NearChainRelayer` in `chains/near/relayer/relayer.go:59` and `TronChainRelayer` in `chains/tron/relayer/relayer.go:59` ignore messages from the error channel
- Listeners like `listener/listener.go:36` do not use context and error channels
- Watcher in `watcher/watcher.go:38` panics on getting an error from the error channel
- No component propagates errors by sending them to the error channel
- There is no handling of the `errChn` to synchronize goroutines

As a result, even if an error happens, all components keep running until a panic occurs that stops the whole process without terminating goroutines gracefully.

Additionally, spawned goroutines are logically grouped in groups of three, including a listener, a dispatcher, and a relayer. If one of them stops, the other two will be deadlocked since the process that feeds data stopped. This would cause a waste of memory and resources to keep the remaining goroutines alive.

Recommendation

We recommend implementing proper management of gorutines using context and channels according to best practices and established patterns.

Status: Partially Resolved

3. Using `BlockHeight` instead of `Timestamp` calculates incorrect transaction expirations

Severity: Critical

The `ValidateTxExpiry` function expects a `Timestamp` as input to evaluate if a transaction is expired.

However, in the following lines of the code, the function is invoked with `BlockHeight` instead of `Timestamp`:

- `chains/tron/relayer/tranformer.go:90`
- `chains/tron/relayer/tranformer.go:137`
- `chains/evm/relayer/transformer.go:145`
- `chains/evm/relayer/transformer.go:193`
- `chains/near/relayer/transformer.go:81`
- `chains/near/relayer/transformer.go:129`

Consequently, the expiration check is ineffective and the result is unpredictable leading the forwarder to discard valid transactions.

Recommendation

We recommend passing `Timestamp` values to the `ValidateTxExpiry` function.

Status: Resolved

4. NEAR `eventProcessor` disregards user configurations and forces the application to connect to untrusted nodes hosted on AWS

Severity: Critical

In `listener/near/eventprocessor/querier.go:61-73`, the `ChainRpc` field defined in the `config.json` for NEAR chains, intended for instantiating the `nearClient` remains unused.

Instead, the application relies on specific AWS endpoints and configurations hardcoded within the `github.com/router-protocol/near-lake-framework-go/core` package.

Additionally, in `listener/near/eventprocessor/querier.go:61-74`, the `QueryEventsFromVoyagerContract` function in the NEAR `eventProcessor` forces the forwarder to connect to an unspecified NEAR testnet disregarding user configurations.

This limitation could prevent the forwarder from establishing a connection with the NEAR mainnet and poses risks about connecting to untrusted nodes.

Recommendation

We recommend allowing the user to specify the endpoints and the network for NEAR.

Status: Resolved

5. The `isTransactionProfitable` function always returns `true` leading the forwarder's operator to relay transactions at a loss

Severity: Critical

In `chains/evm/relayer/executor.go:275-301`, the `isTransactionProfitable` function is used to determine if forwarding a transaction benefits the `forwarder`.

However, the function always returns `true`, even in cases where the execution is not financially advantageous.

Consequently, the current implementation exposes the forwarder's operator to losses for relaying unprofitable transactions.

Recommendation

We recommend assessing profitability by utilizing the prices retrieved from the middleware contract.

Status: Resolved

6. Limitations in `Network` configuration and untrusted hardcoded endpoint usage

Severity: Critical

In `config/config.go:151`, the `loadNetwork` method accepts a string and returns a `Network` structure with hardcoded endpoints for connecting to specific networks.

These endpoints are not configurable by the user and when dealing with a `local` network, it returns a hardcoded configuration that may be not suitable for all the deployment environments.

Furthermore, for some networks, untrusted static IP addresses are hardcoded, necessitating a binary update for endpoint changes and potentially compelling users to connect to untrusted nodes. Additionally, some of them use HTTP and could expose sensitive information.

Recommendation

We recommend refactoring the `loadNetwork` method to allow users to configure endpoints, remove hardcoded IP addresses, and ensure secure HTTPS connections.

Status: Partially Resolved

7. Improper error handling in `Processor`'s `Start` method stops forwarders' operations

Severity: Major

In `processor/processor.go:44`, the `Start` function triggers a panic when it encounters an error, resulting in stopping the whole forwarder process.

Consequently, attackers could leverage this behavior to send improperly formatted messages to disrupt forwarders' operations.

Recommendation

We recommend implementing a Dead Letter Queue (DLQ) for handling messages with errors, or logging these errors and proceeding to process remaining messages in the queue.

Status: Resolved

8. `SR25519` key generation requires an external untrusted binary

Severity: Major

In `utils/crypto/sr25519/sr25519.go:31`, `go-substrate-rpc-client` generates a key.

However, this just implements a client command to make the corresponding RPC call, which does not work without the server side installed on the same machine.

Consequently, this will require the forwarder to contact a potentially untrusted process to generate its key.

The failing `TestDecryptIncorrectType` test in `utils/keystore/encrypt_test.go:132` confirms this.

Recommendation

We recommend adapting Substrate's implementation of `SR25519` or installing all necessary infrastructure locally.

Status: Resolved

9. Unhandled errors in the codebase

Severity: Major

In the following locations, the functions that return an error are not checked nor handled correctly.
For example, some errors are ignored and neither checked nor propagated to the caller.

- `chains/evm/relayer/transformer.go:62`
- `chains/evm/relayer/transformer.go:73`
- `chains/evm/relayer/transformer.go:145`
- `chains/evm/relayer/transformer.go:178`
- `chains/evm/relayer/transformer.go:193`
- `chains/evm/relayer/transformer.go:227`
- `chains/evm/relayer/executor.go:62`
- `chains/evm/relayer/executor.go:67`
- `chains/near/relayer/executor.go:79`
- `chains/near/relayer/transformer.go:81`
- `chains/near/relayer/transformer.go:114`
- `chains/near/relayer/transformer.go:129`
- `chains/near/relayer/transformer.go:163`
- `chains/tron/initializer/initializer.go:37`
- `chains/tron/calls/gateway/gateway.go:177`
- `chains/tron/relayer/executor.go:26`
- `chains/tron/relayer/executor.go:85`
- `chains/tron/relayer/tranformer.go:27`
- `chains/tron/relayer/tranformer.go:59`
- `chains/tron/relayer/tranformer.go:67`
- `chains/tron/relayer/tranformer.go:90`

- `chains/tron/relayer/tranformer.go:123`
- `chains/tron/relayer/tranformer.go:137`
- `chains/tron/relayer/tranformer.go:171`
- `cmd/voyager-forwarder/main.go:48`
- `cmd/voyager-forwarder/main.go:86`
- `config/config.go:114`
- `config/config.go:118`
- `config/config.go:239`
- `listener/evm/eventprocessor/eventprocessor.go:132`
- `listener/evm/eventprocessor/eventprocessor.go:141`
- `listener/evm/eventprocessor/eventprocessor.go:157`
- `listener/evm/eventprocessor/eventprocessor.go:167`
- `listener/evm/eventprocessor/transformer.go:20`
- `listener/evm/eventprocessor/transformer.go:47`
- `listener/evm/eventprocessor/transformer.go:75`
- `listener/near/eventprocessor/eventprocessor.go:115`
- `listener/near/eventprocessor/eventprocessor.go:123`
- `listener/near/eventprocessor/eventprocessor.go:134`
- `listener/near/eventprocessor/eventprocessor.go:136`
- `listener/near/eventprocessor/eventprocessor.go:145`
- `listener/near/eventprocessor/eventprocessor.go:153`
- `listener/near/eventprocessor/eventprocessor.go:164`
- `listener/near/eventprocessor/eventprocessor.go:166`
- `listener/tron/eventprocessor/eventprocessor.go:149`
- `listener/tron/eventprocessor/eventprocessor.go:157`
- `listener/tron/eventprocessor/eventprocessor.go:168`
- `listener/tron/eventprocessor/eventprocessor.go:170`
- `listener/tron/eventprocessor/eventprocessor.go:180`
- `listener/tron/eventprocessor/eventprocessor.go:188`
- `listener/tron/eventprocessor/eventprocessor.go:199`
- `listener/tron/eventprocessor/eventprocessor.go:201`
- `listener/tron/eventprocessor/querier.go:23`
- `listener/tron/eventprocessor/transformer.go:22`
- `listener/tron/eventprocessor/transformer.go:26`
- `listener/tron/eventprocessor/transformer.go:30`
- `listener/tron/eventprocessor/transformer.go:31`
- `listener/tron/eventprocessor/transformer.go:32`
- `listener/tron/eventprocessor/transformer.go:72`
- `listener/tron/eventprocessor/transformer.go:76`
- `listener/tron/eventprocessor/transformer.go:80`
- `listener/tron/eventprocessor/transformer.go:81`
- `listener/tron/eventprocessor/transformer.go:82`
- `listener/tron/eventprocessor/transformer.go:127`
- `listener/tron/eventprocessor/transformer.go:128`
- `listener/tron/eventprocessor/transformer.go:129`
- `listener/listener.go:65`

- `store/store.go:31`
- `store/store.go:32`
- `store/store.go:33`
- `watcher/watcher.go:34`
- `watcher/watcher.go:40`
- `utils/utils.go:133`

Consequently, this would cause silent failures as errors are not raised leading to unexpected behaviors.

Recommendation

We recommend handling the error of the above functions according to the Go error handling best practices. If a function encounters an error, it should wrap the error and propagate it up the call stack as much as possible instead of handling it immediately. The high-level functions should handle received errors by panicking or calling `log.Fatal` to exit the process.

Status: Partially Resolved

10. HTTP and GRPC services do not use TLS

Severity: Major

In `cmd/voyager-forwarder/main.go:86`, `listener/tron/initializer/initialize.go:21`, and `chains/tron/initializer/initializer.go:29` the configurations for exposed services do not use TLS.

As a result, sensitive information could be transmitted in plaintext, significantly increasing the risk of Man-In-The-Middle (MITM) attacks. In such scenarios, attackers could intercept and alter the communication between Tron relayers and clients. This breach would allow them to inject malicious data or reroute connections to servers under their control.

Recommendation

We recommend implementing HTTPS-secure GRPC.

Status: Resolved

11. Missing query pagination handling could lead to partial data retrieval

Severity: Major

In `config/config.go:182`, the `GetAllChainConfig` function queries the `ChainConfigAll` of the Router chain defined in https://github.com/router-protocol/router-chain/blob/67c97a6bf33be0c00b1719f443bd13eac1383aa5/x/multichain/keeper/grpc_query_chain_config.go#L14-L40.

However, since the query is paginated, and the code does not handle pagination explicitly, this function will return only a partial number of instances of the available chain configurations disregarding user input and without throwing errors.

Recommendation

We recommend enhancing data retrieval by implementing pagination controls within the `GetAllChainConfig` function.

Status: Resolved

12. Unthrottled query retries allow DoS attacks on Tron nodes

Severity: Major

In `listener/tron/eventprocessor/eventprocessor.go:74-78`, in case the `GetStartEndBlockTimestamp` function fails to perform a query to a Tron node, the execution retries indefinitely to run the function.

However, reattempting failed queries to the Tron node without introducing any delay poses the risk of the forwarder potentially launching a Denial of Service (DoS) attack on the node.

Recommendation

We recommend implementing a delay mechanism following an error when reattempting failed queries to the Tron node to prevent the forwarder from inadvertently launching Denial of Service (DoS) attacks.

Status: Resolved

13. Incorrect usage of named return parameters in `QueryVoyagerEvents` makes the function not exit in case of error

Severity: Major

In `listener/evm/eventprocessor/querier.go:15-79`, the function `QueryVoyagerEvents` incorrectly utilizes named return parameters.

In cases of error, the function does not promptly `return` but continues the execution until it reaches the `return` statement at line `78`.

Consequently, the function does not exit in line `26` when it encounters issues parsing the ABI, nor in line `32` when it fails to locate the required event in the contract ABI, or even in line `49` when the Ethereum client cannot fetch events.

Recommendation

We recommend correcting the use of named return parameters in the `QueryVoyagerEvents` function to ensure prompt returns on error conditions.

Status: Resolved

14. The execution does not stop if contract ABI is incorrect

Severity: Major

In `listener/evm/eventprocessor/querier.go:23`, if the `abi.JSON` function returns an error, the implementation logs the error in `listener/evm/eventprocessor/querier.go:26` and continues processing the received type, which is `ABI{}`.

Additionally, since the abi package does not return `(nil, err)`, but `(ABI{}, err)` the program will not panic leading to unexpected behavior.

Recommendation

We recommend returning an error if the contract ABI cannot be parsed.

Status: Resolved

15. Missing handling of out-of-index errors

Severity: Major

In `listener/evm/eventprocessor/querier.go:96`, `eventLog.Topics` element access starts with an index equal to one.

However, since there is no previous check of the length of the slice and it is not guaranteed that the slice will contain at least two elements, this could lead to an out-of-index error and the corresponding unhandled panic.

Recommendation

We recommend adding a check on the length of the `Topics`.

Status: Resolved

16. Files are not closed after I/O operations leading to descriptor leakage, memory waste, and potential data corruption

Severity: Major

In multiple sections of the codebase, files are not closed after I/O operations.

Specifically in lines:

- `config/config.go:253-274`, the `loadConfig` function does not close the file after reading `config.json`.
- `config/config.go:101-125`, the `ToJSON` function does not flush and close the file after creating and writing to it.

The consequence is that file descriptors are leaked. The operating system uses descriptors and has resources associated with that open file. If the file is not closed, the corresponding descriptor will not be cleaned up and will persist until the program closes.

Recommendation

We recommend implementing `f.Flush()` and a `defer` statement for `f.Close()` where needed.

Status: Resolved

17. Private keys are stored in plaintext in the configuration file

Severity: Major

The JSON configuration file contains fields for plaintext Ethereum and Cosmos private keys. At the same time, this file can be read by any user on the machine.

Consequently, any user accessing the machine can access the forwarder's private keys which could lead to a loss of funds.

Recommendation

We recommend restricting access to the configuration file or leveraging environment variables or OS keyrings.

Status: Resolved

18. Administrators updating on-chain parameters could cause all connected forwarders to crash

Severity: Minor

During the forwarder initialization, some parameters are fetched from contracts and the Router chain in the following lines:

- In `config/config.go:276-292`, `ValidatorFees` are retrieved from the `Middleware` contract.
- In `config/config.go:294-314`, the `ExpiryPeriod` is retrieved from the `Middleware` contract.
- In `config/config.go:181-209` and `config/config.go:211-218`, the query retrieves all configurations associated with the chain registered on the router chain.

As these configurations can be updated or removed on-chain by administrators, if the forwarder does not react to the update and modifies its parameters, administrators' actions would cause all forwarders to crash.

We are reporting this with minor severity since only on-chain by administrators can cause this issue.

Recommendation

We recommend implementing event monitoring in the `forwarder` to promptly update the mentioned parameters when necessary while considering additional safeguards to prevent potential crashes caused by on-chain updates.

Status: Acknowledged

The client states that a mechanism to refresh the configuration after admin updates will be implemented in future versions.

19. Health check is not implemented

Severity: Minor

In `health/health.go`, the implementation of the health check is entirely commented out and does not return any data.

The `healthCheck` function is in `cmd/voyager-forwarder/main.go:60` simply starts an HTTP server and does not provide any information in terms of health monitoring.

Consequently, clients fetching the health of the process will always get unreliable responses.

Recommendation

We recommend implementing the health check service.

Status: Resolved

20. Hardcoded high gas price leads to inefficiencies and potential stop of forwarder operations

Severity: Minor

In `config/config.go:162` and `cmd/voyager-forwarder/main.go:152`, the `routerChainClient` is configured to use `3000route` as the gas price.

This is problematic since having this value hardcoded and not configurable will not allow operators to react to the validator's required gas price changes.

Additionally, this parameter enforces the gas price to `3000route`, which is a large price compared to other Cosmos SDK chains, which is usually `0.0025udenom`, causing operators to pay for costly operations.

Recommendation

We recommend allowing the user to define the gas price in the configuration file.

Status: Acknowledged

21. Accessing `nil` leads to panic

Severity: Minor

In the following instances of the code, in case an error is raised, then `iRelayMsg` will be `nil`, and accessing its attributes can lead to panics:

- `chains/near/relayer/transformer.go:20`
- `chains/near/relayer/transformer.go:50`
- `chains/tron/relayer/tranformer.go:23`
- `chains/tron/relayer/tranformer.go:55`

Recommendation

We recommend checking variables to be not `nil` before accessing their attributes or invoking their methods.

Status: Resolved

22. Inappropriate use of `context.Background` in multiple code instances

Severity: Minor

In the following instances of the codebase, `context.Background()` is misused:

- `chains/evm/calls/gateway/gateway.go:68`
- `chains/evm/calls/gateway/gateway.go:83`
- `chains/evm/calls/gateway/gateway.go:94`
- `chains/evm/calls/gateway/gateway.go:100`
- `chains/evm/calls/gateway/gateway.go:104`
- `chains/evm/relayer/executor.go:112`
- `chains/evm/relayer/executor.go:209`
- `chains/evm/relayer/executor.go:229`
- `chains/evm/relayer/executor.go:267`
- `chains/evm/relayer/executor.go:276`
- `chains/tron/calls/gateway/gateway.go:101`
- `config/config.go:181`
- `config/config.go:282`
- `config/config.go:300`
- `listener/evm/eventprocessor/eventprocessor.go:181`
- `listener/evm/eventprocessor/transformer.go:93`
- `listener/evm/eventprocessor/querier.go:47`
- `utils/utils.go:262`
- `watcher/watcher.go:57`

While a new context is created for each of those functions, this is not bound to other contexts and, because of that, can not be used for goroutine orchestration or resource management.

Recommendation

We recommend creating a context in the high-level functions and propagating them on the call stack.

Status: Partially Resolved

23. The application gets stuck if the user executes it with incorrect arguments or with `-h` or `-v` flags

Severity: Minor

In `cmd/voyager-forwarder/main.go:38-58`, the `quitChannel` is defined to notify the `main` thread about exit signals like `SIGINT` and `SIGTERM` and exit the process.

However, this logic does not allow the application to exit if the user provides incorrect arguments in the CLI, for example, by running the program without arguments or with the `-v` or `-h` flags.

Consequently, the process will be stuck unless the user sends `SIGINT` or `SIGTERM`.

Recommendation

We recommend revising the signal channel handling to prevent process lockups.

Status: Acknowledged

24. User-defined `logLevel` is ignored

Severity: Minor

In `logger/logger.go:80-85`, the `InitLogger` function initializes the logger.

However, instead of setting the user-provided `logLevel`, it enforces the use of `log.DebugLevel`, disregarding the user's intended log level customization.

Recommendation

We recommend utilizing the log level specified by the user.

Status: Resolved

25. Potential leaked `Ticker`

Severity: Minor

In `cmd/voyager-forwarder/main.go:269`, a `Ticker` is created and started.

However, since that `Ticker` is not stopped at the end of the function, it could lead to a memory leak.

Recommendation

We recommend stopping tickers using `defer` statements.

Status: Resolved

26. Usage of panics instead of errors

Severity: Minor

The following lines contain functions that panic instead of propagating an error to the caller:

- `chains/evm/calls/gateway/gateway.go:106`
- `chains/evm/calls/gateway/gateway.go:110`
- `chains/evm/relayer/transformer.go:27`
- `chains/evm/relayer/transformer.go:140`
- `chains/evm/relayer/transformer.go:188`
- `chains/near/initializer/initialize.go:23`

However, since the signature of the functions allows them to return an error to the caller function, errors should be propagated.

Recommendation

We recommend avoiding the use of `panic` and returning errors instead.

Status: Resolved

27. Hardcoded gas limit leads to unnecessary forwarder operators' expenses

Severity: Minor

Hardcoded gas limits have been found for both `evm` and `near` chains in the following locations:

- In `chains/evm/relayer/executor.go:91` and `chains/evm/relayer/executor.go:255`, a hardcoded value of `40000` is used for `gasLimit`.
- In `chains/near/calls/gateway/gateway.go:70`, a hardcoded value of `200000000000000` is used for `gas`.

However, since the cost of the execution can vary because of multiple factors, having a defined hardcoded gas limit, could lead to inefficiencies and unnecessary forwarder operators' expenses.

Recommendation

We recommend estimating the required gas, for example using `EstimateGasLimit` to get the actual `gasLimit` value or allowing the user to define it in the configuration file.

Status: Resolved

28. Unzeroized secret data in `Listener` initializers and `EventProcessor`

Severity: Minor

In `listener/evm/initializer/initialize.go:45`, `chains/tron/initializer/initializer.go:39` and `listener/tron/eventprocessor/eventprocessor.go:27`, secret data should be zeroized after passing corresponding private keys to the functions.

If zeroization is not performed, an attacker with access to the memory may be able to retrieve non-zeroized private keys.

Recommendation

We recommend implementing data zeroization using `SetFinalizer`.

Status: Acknowledged

29. Integer overflow when creating big numbers

Severity: Minor

In the following lines, big integers are created after transforming `uint64` to `int64`:

- `listener/evm/eventprocessor/transformer.go:93`
- `chains/evm/calls/gateway/gateway.go:112`
- `utils/utils.go:252`
- `utils/utils.go:279`
- `utils/utils.go:282`

The created big integers may exhibit overflows.

Recommendation

We recommend using `big.SetUint64` to construct big integers.

Status: Resolved

30. Lack of input file path validation in `ToJSON` function

Severity: Minor

In the `ToJSON` function, located in `config/config.go:101-125`, there is no validation for the input `file` parameter used as the path of a created file. As a result, it is possible to overwrite existing files.

We report this issue as minor since the `ToJSON` function is not used in the codebase.

Recommendation

We recommend removing the `ToJSON` function or implementing validation of the input file path.

Status: Acknowledged

31. Incomplete validation of the `Config` leads to possible unexpected errors and race conditions

Severity: Minor

In `config/config.go:127-147`, the `validate` function checks the parameters provided by the user in the `config` file.

However, it currently checks only a portion of the `Chain` parameters and overlooks the validation of `GlobalConfig`.

As a result, the configuration validation remains incomplete, and the application proceeds the execution with potentially incorrect parameters.

Additionally, the `Chain` slice is not deduplicated, which could potentially cause duplicated goroutines working on the same data, leading to race conditions.

Recommendation

We recommend extending the `config` validation and deduplicating the `Chain` slice.

Status: Acknowledged

32. Silent failure for missing chains

Severity: Minor

In `config/config.go:187-224`, if a chain specified in the configuration is not located by the `GetAllChainConfig` query, the router protocol silently fails without an error. This results in part of the configuration being ignored.

Recommendation

We recommend implementing proper error handling to generate errors when a chain specified in the configuration is not found by the `GetAllChainConfig` query.

Status: Resolved

33. Inadequate use of unbuffered channels

Severity: Informational

In `cmd/voyager-forwarder/main.go:38-57`, `quitChannel` is defined as an unbuffered channel used to capture the `SIGINT` and `SIGTERM` signal.

However, the use of unbuffered channels should be avoided in this scenario, particularly when I/O operations may entail potential delays.

In fact, it is the caller's responsibility to guarantee that the channel has an adequate buffer size to accommodate the anticipated signal frequency. In situations where the channel is employed solely for notifying a single signal value, a buffer size of 1 is satisfactory.

Recommendation

We recommend defining `quitChannel` as a buffered channel with a buffer size of one.

Status: Resolved

34. Hardcoded Tron chain ID

Severity: Informational

In `cmd/voyager-forwarder/main.go:170`, the chain ID for Tron is hardcoded.

This imposes limitations on users, preventing them from utilizing alternative Tron chains without requiring a binary update to modify the chain ID.

Recommendation

We recommend adopting a more flexible approach by defining a distinct chain type and a new field in the configuration file to enable users to specify chain IDs for Tron.

Status: Acknowledged

35. Incorrect usage of `Sleep` slows down the `RequestProcessor`

Severity: Informational

In `processor/processor.go:37-66`, the `Start` method of the `RequestProcessor` starts a goroutine for processing incoming events.

However, the current implementation introduces a one-second `Sleep` after processing each event instead of at the end of the processing batch, which would slow down the execution. A more reasonable approach would be to place the `Sleep` invocation outside of the `for` loop, occurring between batches of events.

Additionally, the method starts two goroutines instead of one which is unnecessary and inefficient.

Recommendation

We recommend eliminating the redundant goroutine within the `Start` method and optimizing sleep management by placing the `Sleep` invocation outside of the `for` loop occurring between batches of events.

Status: Partially Resolved

36. `EncryptAndWriteToFile` writes secret data to a file with unknown access rights

Severity: Informational

In `utils/keystore/encrypt.go:71`, encrypted data is written to the input file with unknown privileges.

In case privileges are set to `777` then the data in the file can be overwritten by any users in the system.

Recommendation

We recommend creating a file with restricted privileges in the `EncryptAndWriteToFile` function.

Status: Acknowledged

37. Unutilized function arguments

Severity: Informational

The following lines contain instances of functions with unused arguments:

- `chains/evm/relayer/executor.go:275`
- `chains/evm/relayer/relayer.go:78`
- `chains/near/relayer/relayer.go:59`
- `chains/tron/relayer/relayer.go:59`
- `cmd/voyager-forwarder/main.go:60`
- `listener/evm/eventprocessor/transformer.go:67`
- `listener/tron/eventprocessor/transformer.go:119`
- `listener/listener.go:36`

- `logger/logger.go:80`
- `oracle/tokenPrice.go:13`
- `processor/processor.go:68`
- `processor/processor.go:80`
- `watcher/watcher.go:32`

Recommendation

We recommend implementing logic that uses those arguments, leaving them unnamed by using _ if required according to the interfaces or completely removing them.

Status: Partially Resolved

38. Incorrect logger labeling

Severity: Informational

In `logger/logger.go:80-81`, the `initLogger` function initializes the logger.

However, it erroneously assigns the `orchestrator` label to logs instead of the intended `forwarder`.

Recommendation

We recommend changing the `NewLogger` argument to be set as `forwarder`.

Status: Resolved

39. Double import of the `logrus` package

Severity: Informational

In `logger/logger.go:10-11`, the `logrus` package has been redundantly imported with different aliases which, being unnecessary, may result in inefficiencies.

Recommendation

We recommend removing one of the imports of the `logrus` package.

Status: Resolved

40. Type inconsistency for `ChainType` struct parameter

Severity: Informational

In `config/config.go:56`, the `ChainType` is defined as a `string` in the `Chain` struct.

However, the same parameter is defined as `multichainTypes.ChainType` in the `ChainSpecs` struct, resulting in incoherence.

Recommendation

We recommend ensuring consistency by using the same data type, `multichainTypes.ChainType`, for `ChainType` in both the `Chain` and `ChainSpecs` structs.

Status: Resolved

41. Unused variables in various code instances

Severity: Informational

In the following lines, the `err` variable is not used after being assigned:

- `chains/near/relayer/executor.go:79`
- `chains/tron/relayer/executor.go:26`
- `chains/tron/relayer/executor.go:85`

In the following lines, the `bytesDestChainId` variable is not used after being assigned:

- `listener/tron/eventprocessor/transformer.go:267`
- `listener/tron/eventprocessor/transformer.go:76`

Recommendation

We recommend using these variables or removing them.

Status: Resolved

42. Remove redundant `ChainClient` type from the `fetchTokenPriceList` arguments

Severity: Informational

In `watcher/watcher.go:51`, the `fetchTokenPriceList` function accepts an argument of `routerclient.ChainClient` type. However, `ChainClient` is already contained in the `Watcher` type in `watcher/watcher.go:19`.

Recommendation

We recommend removing `ChainClient` from `fetchTokenPriceList` function arguments.

Status: Resolved

43. Incorrect logging level

Severity: Informational

In the following lines of the code, the returned error is logged with `Debug` or `Info` level instead of `Error` level:

- `chains/tron/relayer/tranformer.go:30`
- `chains/tron/relayer/tranformer.go:61`
- `chains/tron/relayer/executor.go:30`
- `chains/tron/relayer/executor.go:89`

Recommendation

We recommend using `Error` level to log information about happened errors.

Status: Resolved

44. Incorrect struct tag syntax

Severity: Informational

In `config/config.go:68-88`, the `json` struct tag syntax for `MaxApprovalInWei` and `MinApprovalThresholdInWei` fields is incorrect.

Recommendation

We recommend correcting it by adding a double quote around the tag name.

Status: Resolved

45. Outdated dependencies

Severity: Informational

Several packages (e.g., `cosmos-sdk`, `net`) and the Go compiler are outdated and have known vulnerabilities (e.g., `GO-2023-2102` and `GO-2023-1878`) fixed in the newest versions.

Recommendation

We recommend updating the dependencies and the Go compiler version.

Status: Acknowledged

46. Usage of deprecated functions

Severity: Informational

In `utils/keystore/decrypt.go:79`, `listener/tron/eventprocessor/querier.go:24`, and `listener/tron/initializer.initialize.go:21` deprecated functions are used.

Recommendation

We recommend using updated functions instead of deprecated ones.

Status: Partially Resolved

47. Private key fields are not used

Severity: Informational

In `listener/tron/eventprocessor/eventprocessor.go:27`, `listener/evm/eventprocessor/eventprocessor.go:27`, `config/config.go:43` private key fields are declared but not used.

Recommendation

We recommend removing private key fields if they are unnecessary or implementing functions exercising them.

Status: Resolved

48. Hardcoded sleep time ignores `waitPeriod` parameter

Severity: Informational

In `watcher/watcher.go:43` the sleep time is hardcoded to one second, despite the presence of the `waitPeriod` parameter passed to the function.

Recommendation

We recommend using `waitPeriod` for the `Sleep` input rather than the hardcoded `1 * time.Second`.

Status: Resolved

49. Miscellaneous comments

Severity: Informational

Miscellaneous recommendations can be found below.

Recommendation

The following are some recommendations to improve the overall code quality and readability:

- `chains/tron/relayer/tranformer.go` should be renamed to `transformer.go`.
- `TODO` context in `listener/evm/eventprocessor/eventprocessor.go:112` should be defined properly.
- Consider using vendoring (`go mod vendor`) to obtain durability, reproducible building, and testability.
- Consider using `golangci-lint` tools for linting, especially `errcheck`, `govet`, `ineffassign`, `unused`, and `staticchek`.
- Do not use `fmt` package for logging.
- Break long lines of code (e.g., `listener/listener.go:52`).
- Remove non-meaningful and non-idiomatic comments (e.g., `listener/evm/eventprocessor/transformer.go:68`).
- Remove misleading leading `I` from the names of the structures in `types/message.go` (e.g., `IAssetDepositedDataWithMessage`, `IAssetDepositedData`).
- Remove superfluous error handling in `chains/evm/relayer/transformer.go:26-28`.

Status: Partially Resolved