CERTIK

Security Assessment

Token Relay

Apr 19th, 2021

Summary

This report has been prepared for Stafi , to discover issues and vulnerabilities in the source code of their relay mechanisms. A comprehensive examination has been performed, utilising Dynamic Analysis, Static Analysis, and Manual Review techniques.

The security assessment resulted in findings that ranged from major to informational. The team opted to fix part of the findings and acknowledge the rest to be resulted in future iterations. The audit took under consideration regarding alleviations the branch `newrdot` up to the commit hash `0c7f0073ee1ef766aa055e73975f9af74190bd28`.

Overview

Project Summary

Project Name	Token Relay
Description	A bridge relay solution.
Platform	Custom
Language	Golang
Codebase	https://github.com/stafiprotocol/rtoken-relay
Commits	2de63ed08dc8a23b2561d8877dc58fb03a0acaa9

Audit Summary

Delivery Date	Apr 19, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Total Issues	52
● Critical	0
● Major	3
● Minor	13
● Informational	35
● Discussion	1

Audit Scope

ID	file	SHA256 Checksum
INT	chains/interface.go	85d34d771178d743e8947fdbd2119b6268ad2c97b2e9f040704c41b636c80b1f
CHA	chains/substrate/chain.go	c7431ca5d7b73fbaa4162c0de3aee8e9268b9d135d501b44e550fe0fd93cc9c9
CON	chains/substrate/connection.go	0282eda49c1bb66f424287814e29ca9d13a78374c941014882651b496ec3bac9
EVE	chains/substrate/event.go	31eaeeff63916544a5eee08dbacebac11b0ec70dfd602d440b8f923bb77e35a8
EVN	chains/substrate/event_handler.go	a6e136ed598fd8619d97f13097b5738989f85eb8c96bee530bac5bd2cb69bf9e
LIS	chains/substrate/listener.go	7ece3c6baab59f6f18786fe7f0c1d05b99a1020a3a81276510d1fae31740c286
PRO	chains/substrate/proposal.go	6289fa743ef46afb1d50caf78f8a45edbfa421d1262d9ef92dcb1520bc7067ff
WRI	chains/substrate/writer.go	bd73e8e490e028ff3705f63d6ae3babd1aa1f1e9408757df25e3a0bff6533906
ACC	cmd/relay/account.go	87d24c6a913912e89b2de9d0c7d9e60d0682620c0d041a8b1f1dd6d1e8c0fb66
MAI	cmd/relay/main.go	c818b3d3c2b29d8362924446f35f7f62096cc70d31c5872c2141f336b5184ee1
COF	config/config.go	9d4ab1756116324c26551597714fb1c4f49b6b12bd7e64eeeadac2af80a9ec1b
FLA	config/flags.go	0a0d6e6b9a653466c77aee4c0b208c1cd3a0e658e54c2bb953972bfb444cd3f7
STA	config/stafi.go	f4d4d18bba0b089f392da550de51adbb124dcddbcbc0fbdd400164b35a5e8563
SUB	config/substrate.go	8931ef4d1ae5a83f825028808c704a2dfaf653c252469215879a84358b866151
CHI	core/chain.go	32cdc634c8e7d81dca2057c2047c7515df9d8decfb9c9238cf6b7799d2738389
COR	core/core.go	59a90c35578955a76710f7766538881d08c6c0cc0bf8cd4bfa6a73146d1a8225
MES	core/message.go	a1b5358fbbc6bdac5cb3f2b9dd38156e64d3b62bdac0d7636e5267c7a2412935
MOD	core/model.go	4507d3fd344657a9f8c9869c318a527127a759ed13a36a2592d644bbf5d89867
ROU	core/router.go	62c11ed0d5bbc2e2e452ac8beefe085036c76baca785b219a911447591ebe6d0
EVT	models/submodel/event_parser.go	1e5cee824806eb25e4ccbf4efcc99a7ed267d7b442f4291591514b3d8edaf122
MOE	models/submodel/model.go	7357912e7bde67422561e34b1cdfe1f10ad2f85762f5379049ff90906cdd5531

ID	file	SHA256 Checksum
GSR	shared/substrate/gsrpc.go	d9fb9808ed9df7fc794dd8e88a23152a5c5cded968f73d838772eb74cf3bba45
MOL	shared/substrate/model.go	33a5fa0d3aecd8ba801400262ffbca329ef1956c605997f1622f424f1cdf2698
SAR	shared/substrate/sarpc.go	79152161b1bf57791e5b21b4163b1743250e3e2a5d2505691115b1da42ea63fc
TYP	shared/substrate/types.go	d3a8448d9a4d8923fbbcd74ceecb87e16f363f64c3442f77a0d9d9d03459703a
BIG	utils/bigint.go	68439ef783c48b585ecbead111728717a33dc3c0de9347f90d0216802d0b0643
BLA	utils/blake2.go	4695b4d872abc0a335261d8803223f6164ef8944a93a677e1266ce6f1ed32b29
BLO	utils/blockstore.go	1d0b8a17082eab0313e2b61ed19b36f91fca15e16b7e8e3ec2cdbf06ced4e01b

System

The system describes a relay service acting as a chain bridge based on the implementation-defined initially here https://github.com/ChainSafe/ChainBridge with extended functionality, bridging the substrate stafi chain and ethereum based stafi smart contract.

The team revised the code structure by adding needed structural updates and performed changes that align with the current implementation protocol's needs.

The code audited meets the specification and the intended results but lacks the testing and documentation to provide a concrete security assumption.

Golang Specific

The Golang specific examination revealed that the code is in good condition regarding language specifications and best practices with room for improvement.

With some minor exceptions that we highlight in the findings section, the code presented is consistent with best practices and language-specific idioms. Project-specific changes are in alignment with the ethos of the rest of the code.

The team should consider revising the codebase's error handling as it contains global variables and naming that are not parallel with the language specifications and best practices.

Documentation

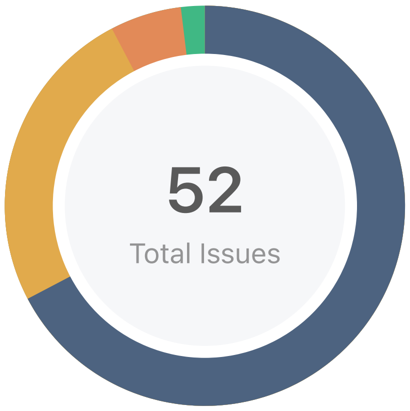
The level of documentation on the code is very minimal, with comments deviating from the language specifications making it harder to follow the code.

The code can fundamentally benefit from detailed commenting that will help support the code's maintainability and readability. Additionally, this will make it easier to onboarding new members or community contributors, or even future auditors, drastically reducing the time needed to understand and navigate the codebase.

Testing

The team should extend the project's testing suite and achieve as much code coverage as possible. The testing suite is currently minimal, and the client should address this before any production release.

Findings



🟥	Critical	**0** (0.00%)
🟧	Major	**3** (5.77%)
🟨	Minor	**13** (25.00%)
🟦	Informational	**35** (67.31%)
🟩	Discussion	**1** (1.92%)

ID	Title	Category	Severity	Status
BIG-01	Potential Overflow of U128	Mathematical Operations	● Major	ⓘ Acknowledged
BLO-01	Panic	Logical Issue	● Minor	ⓘ Acknowledged
CHA-01	Error Naming Issue	Coding Style	● Informational	ⓘ Acknowledged
CHA-02	Global Variable Declaration	Coding Style	● Informational	ⓘ Acknowledged
CHA-03	Naming Issue	Coding Style	● Informational	ⓘ Acknowledged
CHA-04	Panic	Language Specific	● Informational	ⓘ Acknowledged
COF-01	Naming Issue	Coding Style	● Informational	ⓘ Acknowledged
COF-02	Missing Check For Close	Language Specific	● Minor	✓ Resolved
COF-03	Redundant Return Value	Logical Issue	● Informational	✓ Resolved
COF-04	Unsafe Type Assertion	Logical Issue, Language Specific	● Informational	ⓘ Acknowledged
CON-01	Error Naming Issue	Coding Style	● Informational	ⓘ Acknowledged
CON-02	Global Variable Declaration	Coding Style	● Informational	ⓘ Acknowledged
CON-03	Inefficient Error Design	Coding Style	● Informational	ⓘ Acknowledged
CON-04	Confusing Comment	Logical Issue	● Informational	ⓘ Acknowledged
CON-05	Not Returning Error	Logical Issue	● Minor	ⓘ Acknowledged
CON-06	Redundant Block	Coding Style	● Informational	ⓘ Acknowledged

ID	Title	Category	Severity	Status
CON-07	Should Be Switch Statement	Coding Style	● Informational	ⓘ Acknowledged
CON-08	Panic	Language Specific	● Minor	ⓘ Acknowledged
CON-09	No Return On Error	Logical Issue	● Minor	✓ Resolved
EVE-01	Error Naming Issue	Coding Style	● Informational	ⓘ Acknowledged
EVE-02	Global Variable Declaration	Coding Style	● Informational	ⓘ Acknowledged
EVE-03	Unused Error	Logical Issue	● Minor	✓ Resolved
EVE-04	Unchecked Error	Logical Issue	● Informational	✓ Resolved
EVE-05	Unchecked Error	Logical Issue	● Minor	✓ Resolved
EVN-01	Error Design	Coding Style	● Informational	ⓘ Acknowledged
EVT-01	Error Naming Issue	Coding Style	● Informational	ⓘ Acknowledged
EVT-02	Global Variable Declaration	Coding Style	● Informational	ⓘ Acknowledged
EVT-03	Unused Function	Logical Issue	● Informational	ⓘ Acknowledged
EVT-04	Unsafe Type Assertion	Logical Issue, Language Specific	● Informational	✓ Resolved
EVT-05	Missing Check	Logical Issue, Inconsistency	● Major	✓ Resolved
EVT-06	Return Variable Declaration	Language Specific	● Informational	ⓘ Acknowledged
GSR-01	Confusing Comment	Logical Issue	● Informational	ⓘ Acknowledged
GSR-02	Should Be Switch Statement	Coding Style	● Informational	ⓘ Acknowledged
LIS-01	Unused Error	Logical Issue	● Major	✓ Resolved
LIS-02	Should Be Switch Statement	Coding Style	● Informational	ⓘ Acknowledged
LIS-03	Naming Issue	Coding Style, Language Specific	● Informational	ⓘ Acknowledged
MAI-01	Flow Control	Logical Issue, Control Flow	● Informational	✓ Resolved
MAI-02	Redundant Return Value	Logical Issue	● Informational	✓ Resolved
MOL-01	Error Naming Issue	Coding Style	● Informational	ⓘ Acknowledged

ID	Title	Category	Severity	Status
PRO-01	Unused Error	Logical Issue	● Minor	✓ Resolved
PRO-02	Coding Style Inconsistency	Coding Style	● Informational	ⓘ Acknowledged
PRO-03	Confusing Logic	Logical Issue	● Discussion	! Pending
SAR-01	Panic	Logical Issue	● Minor	✓ Resolved
SAR-02	Return Variable Declaration	Language Specific	● Informational	ⓘ Acknowledged
SAR-03	Ugly due to declaring return variable	Coding Style	● Informational	ⓘ Acknowledged
WRI-01	Unused Declaration	Logical Issue	● Informational	ⓘ Acknowledged
WRI-02	Unused Error	Logical Issue	● Minor	✓ Resolved
WRI-03	Redundant Select For Single Case	Logical Issue	● Informational	✓ Resolved
WRI-04	Inefficient Break	Logical Issue	● Minor	✓ Resolved
WRI-05	Error Not Used	Language Specific	● Minor	✓ Resolved
WRI-06	Inefficient Error Message	Language Specific	● Informational	ⓘ Acknowledged
WRI-07	Silenced Error	Logical Issue	● Minor	✓ Resolved

BIG-01 | Potential Overflow of U128

Category	Severity	Location	Status
Mathematical Operations	● Major	utils/bigint.go: 14~17	ⓘ Acknowledged

Description

The function can cause the two `types.U128` arguments to "overflow" as it performs no bounds check on the resulting `c` variable nor does the `types.NewU128` constructor do so.

Recommendation

We advise the function to properly assess whether the addition of the two `types.U128` members caused an overflow.

BLO-01 | Panic

Category	Severity	Location	Status
Logical Issue	● Minor	utils/blockstore.go: 51	ⓘ Acknowledged

Description

The code uses panic.

Recommendation

It is recommended to log the err and gracefully exit instead of panic.

CHA-01 | Error Naming Issue

Category	Severity	Location	Status
Coding Style	● Informational	chains/substrate/chain.go: 15	ⓘ Acknowledged

Description

Errors in Golang should be in form `ErrorFoo`.

Recommendation

Refactor the code and follow the language best practices.

CHA-02 | Global Variable Declaration

Category	Severity	Location	Status
Coding Style	● Informational	chains/substrate/chain.go: 15	ⓘ Acknowledged

Description

The code declares global variables.

Recommendation

Remove global variables and create a proper error implementation respecting the language best practices.

CHA-03 | Naming Issue

Category	Severity	Location	Status
Coding Style	● Informational	chains/substrate/chain.go: 80	ⓘ Acknowledged

Description

The linked code naming deviates from the language naming specifications.

Recommendation

Refactor the code with respect to the language specifications and best practices.

CHA-04 | Panic

Category	Severity	Location	Status
Language Specific	● Informational	chains/substrate/chain.go: 111, 115	ⓘ Acknowledged

Description

The code uses panic.

Recommendation

This is acceptable as we are still in the initialisation phase, still it would be better to log and gracefully exit.

COF-01 | Naming Issue

Category	Severity	Location	Status
Coding Style	● Informational	config/config.go: 27	ⓘ Acknowledged

Description

The linked struct field name deviates from the language specifications.

Recommendation

Rename the struct field as `RSymbol`.

COF-02 | Missing Check For Close

Category	Severity	Location	Status
Language Specific	● Minor	config/config.go: 65	✓ Resolved

Description

The linked code does not check if the access to the file opened is closed properly.

Recommendation

Check if f.Close returns an error.

Alleviation

Fixed by the team in commit 0c7f0073ee1ef766aa055e73975f9af74190bd28

COF-03 | Redundant Return Value

Category	Severity	Location	Status
Logical Issue	● Informational	config/config.go: 50	✓ Resolved

Description

The code returns a value along the error.

Recommendation

We can return nil error here instead.

Alleviation

Fixed by the team in commit 0c7f0073ee1ef766aa055e73975f9af74190bd28.

COF-04 | Unsafe Type Assertion

Category	Severity	Location	Status
Logical Issue, Language Specific	● Informational	config/config.go: 50	ⓘ Acknowledged

Description

The code performs a type assertion that does not check if it was ok.

Recommendation

Even if we are sure here that we are ok to assert for code maintainability factor only we should still perform the check.

CON-01 | Error Naming Issue

Category	Severity	Location	Status
Coding Style	● Informational	chains/substrate/connection.go: 38	ⓘ Acknowledged

Description

Errors in Golang should be in form `ErrorFoo`.

Recommendation

We recommend to refactor the code and follow the language best practices.

CON-02 | Global Variable Declaration

Category	Severity	Location	Status
Coding Style	● Informational	chains/substrate/connection.go: 38	ⓘ Acknowledged

Description

The code declares global variables.

Recommendation

Remove global variables and create a proper error implementation respecting the language best practices.

CON-03 | Inefficient Error Design

Category	Severity	Location	Status
Coding Style	● Informational	chains/substrate/connection.go: 47, 73	ⓘ Acknowledged

Description

The code base contains a single global error declaration and a series of error creations.

Recommendation

We advise for a redesign of the error handling. Introducing an error.go would be advised.

CON-04 | Confusing Comment

Category	Severity	Location	Status
Logical Issue	● Informational	chains/substrate/connection.go: 112	ⓘ Acknowledged

Description

The linked comment is confusing and deviates from the functionality following.

Recommendation

Refactor the comment providing insight to the functionality.

CON-05 | Not Returning Error

Category	Severity	Location	Status
Logical Issue	● Minor	chains/substrate/connection.go: 150, 160, 166, 170, 181, 186, 191, 195, 199, 205, 209, 215	ⓘ Acknowledged

Description

The linked code segment does not return the error.

Recommendation

We recommend to return the error. In this case the Bond Reason represents also failed statuses but it would be nice to return the err also since it's there. Additionally due to the fact that the function returns an error future development might introduce problems based on the assumption that err is nil.

CON-06 | Redundant Block

Category	Severity	Location	Status
Coding Style	● Informational	chains/substrate/connection.go: 197	ⓘ Acknowledged

Description

The code contains a redundant block.

Recommendation

Refactor the code and replace the blocks with a single else if {}.

CON-07 | Should Be Switch Statement

Category	Severity	Location	Status
Coding Style	● Informational	chains/substrate/connection.go: 174	ⓘ Acknowledged

Description

The linked code segment could be represented better with a switch statement.

Recommendation

We recommend to refactor the code using a switch statement.

CON-08 | Panic

Category	Severity	Location	Status
Language Specific	● Minor	chains/substrate/connection.go: 304	ⓘ Acknowledged

Description

The code uses panic while concurrent execution is running.

Recommendation

We advise to remove the panic use and replace it with a log entry followed by a graceful exit.

CON-09 | No Return On Error

Category	Severity	Location	Status
Logical Issue	● Minor	chains/substrate/connection.go: 376	✓ Resolved

Description

The code just logs the error and moves on.

Recommendation

Return after logging the error.

Alleviation

Fixed by the team in commit 0c7f0073ee1ef766aa055e73975f9af74190bd28.

EVE-01 | Error Naming Issue

Category	Severity	Location	Status
Coding Style	● Informational	chains/substrate/event.go: 17	ⓘ Acknowledged

Description

Errors in Golang should be in form `ErrorFoo`.

Recommendation

We recommend to refactor the code and follow the language best practices.

EVE-02 | Global Variable Declaration

Category	Severity	Location	Status
Coding Style	● Informational	chains/substrate/event.go: 17	ⓘ Acknowledged

Description

The code declares global variables.

Recommendation

Remove global variables and create a proper error implementation respecting the language best practices.

EVE-03 | Unused Error

Category	Severity	Location	Status
Logical Issue	● Minor	chains/substrate/event.go: 172	✓ Resolved

Description

Unused Error

Recommendation

Implement a check again the error.

Alleviation

Fixed by the team in commit 0c7f0073ee1ef766aa055e73975f9af74190bd28.

EVE-04 | Unchecked Error

Category	Severity	Location	Status
Logical Issue	● Informational	chains/substrate/event.go: 226	✓ Resolved

Description

The code silences an error by not taking it under consideration.

Recommendation

Log the error even if we want to act on exists variable or use _ for err and remove the check completely.

Alleviation

Fixed by the team in commit 0c7f0073ee1ef766aa055e73975f9af74190bd28.

EVE-05 | Unchecked Error

Category	Severity	Location	Status
Logical Issue	● Minor	chains/substrate/event.go: 247	✓ Resolved

Description

The code does not check an error value.

Recommendation

Implement a check against the error value.

Alleviation

Fixed by the team in commit 0c7f0073ee1ef766aa055e73975f9af74190bd28.

EVN-01 | Error Design

Category	Severity	Location	Status
Coding Style	● Informational	chains/substrate/event_handler.go: 47	ⓘ Acknowledged

Description

The code represents errors with hard coded values.

Recommendation

We recommend for a more proper design where errors live in the error.go file and are constants.

EVT-01 | Error Naming Issue

Category	Severity	Location	Status
Coding Style	● Informational	models/submodel/event_parser.go: 18~22	ⓘ Acknowledged

Description

Errors in Golang should be in form `ErrorFoo`.

Recommendation

Refactor the code and follow the language best practices.

EVT-02 | Global Variable Declaration

Category	Severity	Location	Status
Coding Style	● Informational	models/submodel/event_parser.go: 19	ⓘ Acknowledged

Description

The code declares global variables.

Recommendation

Remove global variables and create a proper error implementation respecting the language best practices.

EVT-03 | Unused Function

Category	Severity	Location	Status
Logical Issue	● Informational	models/submodel/event_parser.go: 291	ⓘ Acknowledged

Description

The function is unused.

Recommendation

Remove the function or create a comment.

EVT-04 | Unsafe Type Assertion

Category	Severity	Location	Status
Logical Issue, Language Specific	● Informational	models/submodel/event_parser.go: 29, 36, 39	Resolved

Description

The code performs a type assertion that does not check if it was ok.

Recommendation

Even if we are sure here that we are ok to assert for code maintainability factor only we should still perform the check.

Alleviation

Fixed by the team in commit 0c7f0073ee1ef766aa055e73975f9af74190bd28.

EVT-05 | Missing Check

Category	Severity	Location	Status
Logical Issue, Inconsistency	● Major	models/submodel/event_parser.go: 74	✓ Resolved

Description

The function deviates from the rest of the code and does not check the array length.

Recommendation

Add a check regarding the array length.

Alleviation

Fixed by the team in commit 0c7f0073ee1ef766aa055e73975f9af74190bd28.

EVT-06 | Return Variable Declaration

Category	Severity	Location	Status
Language Specific	● Informational	models/submodel/event_parser.go: 74	ⓘ Acknowledged

Description

The function declares a return variable.

Recommendation

Declare the variable inside the functions scope.

GSR-01 | Confusing Comment

Category	Severity	Location	Status
Logical Issue	● Informational	shared/substrate/gsrpc.go: 89	ⓘ Acknowledged

Description

The comment is confusing.

Recommendation

Refactor the comment so that it represents the functionality.

GSR-02 | Should Be Switch Statement

Category	Severity	Location	Status
Coding Style	● Informational	shared/substrate/gsrpc.go: 259	ⓘ Acknowledged

Description

The code segment should be represented with a switch statement.

Recommendation

Refactor the code and introduce a switch statement.

LIS-01 | Unused Error

Category	Severity	Location	Status
Logical Issue	● Major	chains/substrate/listener.go: 147	✓ Resolved

Description

The code does not check the error.

Recommendation

Implement a check against the error value.

Alleviation

Fixed by the team in commit 0c7f0073ee1ef766aa055e73975f9af74190bd28.

LIS-02 | Should Be Switch Statement

Category	Severity	Location	Status
Coding Style	● Informational	chains/substrate/listener.go: 213	ⓘ Acknowledged

Description

The linked code segment should be represented with a switch statement.

Recommendation

Refactor the code and introduce a switch statement.

LIS-03 | Naming Issue

Category	Severity	Location	Status
Coding Style, Language Specific	● Informational	chains/substrate/listener.go: 42	ⓘ Acknowledged

Description

The linked struct field deviates from the naming specifications for the language used.

Recommendation

Rename the field with respect to language specifications and best practises.

MAI-01 | Flow Control

Category	Severity	Location	Status
Logical Issue, Control Flow	● Informational	cmd/relay/main.go: 121, 124~126	✓ Resolved

Description

The linked code segment could be executed earlier.

Recommendation

Move the check inside the if chain.Type scope.

Alleviation

Fixed by the team in commit 0c7f0073ee1ef766aa055e73975f9af74190bd28.

MAI-02 | Redundant Return Value

Category	Severity	Location	Status
Logical Issue	● Informational	cmd/relay/main.go: 124~126	✓ Resolved

Description

The code returns a value along the error.

Recommendation

We can simple return nil and err here.

Alleviation

Fixed by the team in commit 0c7f0073ee1ef766aa055e73975f9af74190bd28.

MOL-01 | Error Naming Issue

Category	Severity	Location	Status
Coding Style	● Informational	shared/substrate/model.go: 10	ⓘ Acknowledged

Description

Error naming

PRO-01 | Unused Error

Category	Severity	Location	Status
Logical Issue	● Minor	chains/substrate/proposal.go: 108	✓ Resolved

Description

The code does not check against the error.

Recommendation

Implement a check against the error value.

Alleviation

Fixed by the team in commit 0c7f0073ee1ef766aa055e73975f9af74190bd28.

PRO-02 | Coding Style Inconsistency

Category	Severity	Location	Status
Coding Style	● Informational	chains/substrate/proposal.go: 28, 48, 67, 87, 178	ⓘ Acknowledged

Description

The linked code deviates from the rest of the codebase regarding the declaration of fields names.

Recommendation

Refactor the code and stay consistent with the rest of the codebase and best practices.

PRO-03 | Confusing Logic

Category	Severity	Location	Status
Logical Issue	● Discussion	chains/substrate/proposal.go: 101	ⓘ Pending

Description

The code segment returns true on a not valid state.

Recommendation

We need explanations regarding the intended outcome here.

SAR-01 | Panic

Category	Severity	Location	Status
Logical Issue	● Minor	shared/substrate/sarpc.go: 115	✓ Resolved

Description

The code uses panic.

Recommendation

It is recommended to log the err and gracefully exit instead of panic.

Alleviation

Fixed by the team in commit 0c7f0073ee1ef766aa055e73975f9af74190bd28.

SAR-02 | Return Variable Declaration

Category	Severity	Location	Status
Language Specific	● Informational	shared/substrate/sarpc.go: 68	ⓘ Acknowledged

Description

The function declares a return variable.

Recommendation

Declare the variable inside the functions scope.

SAR-03 | Ugly due to declaring return variable

Category	Severity	Location	Status
Coding Style	● Informational	shared/substrate/sarpc.go: 79	ⓘ Acknowledged

Description

Ugly due to declaring return variable

WRI-01 | Unused Declaration

Category	Severity	Location	Status
Logical Issue	● Informational	chains/substrate/writer.go: 34~36, 55~57	ⓘ Acknowledged

Description

The code declares an allocation that is never used.

Recommendation

Remove the unused declarations.

Alleviation

Fixed by the team in commit 0c7f0073ee1ef766aa055e73975f9af74190bd28.

WRI-02 | Unused Error

Category	Severity	Location	Status
Logical Issue	● Minor	chains/substrate/writer.go: 225, 441, 442	✓ Resolved

Description

The code fails to evaluate an error.

Recommendation

We recommend to implement a check against the error.

Alleviation

Fixed by the team in commit 0c7f0073ee1ef766aa055e73975f9af74190bd28.

WRI-03 | Redundant Select For Single Case

Category	Severity	Location	Status
Logical Issue	● Informational	chains/substrate/writer.go: 692	✓ Resolved

Description

The code introduces a select statement for a single case.

Recommendation

Refactor the code and use a send to the channel.

Alleviation

Fixed by the team in commit 0c7f0073ee1ef766aa055e73975f9af74190bd28.

WRI-04 | Inefficient Break

Category	Severity	Location	Status
Logical Issue	● Minor	chains/substrate/writer.go: 701	✓ Resolved

Description

The code tries to escape the loop with a break statement that does not refer to the right condition.

Recommendation

We advise that the whole statement is removed.

Alleviation

Fixed by the team in commit 0c7f0073ee1ef766aa055e73975f9af74190bd28.

WRI-05 | Error Not Used

Category	Severity	Location	Status
Language Specific	● Minor	chains/substrate/writer.go: 222	✓ Resolved

Description

The code does not take under consideration the case of the function returning an error.

Recommendation

Check against the case of error.

Alleviation

Fixed by the team in commit 0c7f0073ee1ef766aa055e73975f9af74190bd28.

WRI-06 | Inefficient Error Message

Category	Severity	Location	Status
Language Specific	● Informational	chains/substrate/writer.go: 181	ⓘ Acknowledged

Description

The error deviates from language specifications and best practices.

Recommendation

Redesign the errors.

WRI-07 | Silenced Error

Category	Severity	Location	Status
Logical Issue	● Minor	chains/substrate/writer.go: 696	✓ Resolved

Description

The code does not log the error cause here before returning.

Recommendation

Log the error as in previous cases.

Alleviation

Fixed by the team in commit 0c7f0073ee1ef766aa055e73975f9af74190bd28.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings refer to exhibits that do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

Mathematical Operations

Mathematical Operation exhibits entail findings that relate to mishandling of math formulas, such as overflows, incorrect operations etc.

Logical Issue

Logical Issue findings are exhibits that detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Control Flow

Control Flow findings concern the access control imposed on functions, such as owner-only functions being invoke-able by anyone under certain circumstances.

Volatile Code

Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases that may result in a vulnerability.

Data Flow

Data Flow findings describe faults in the way data is handled at rest and in memory, such as the result of a struct assignment operation affecting an in-memory struct rather than an in storage one.

Language Specific

Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete .

Coding Style

Coding Style findings usually do not affect the generated byte-code and comment on how to make the codebase more legible and as a result easily maintainable.

Inconsistency

Inconsistency findings refer to functions that should seemingly behave similarly yet contain different code, such as a constructor assignment imposing different require statements on the input variables than a setter function.

Magic Numbers

Magic Number findings refer to numeric literals that are expressed in the codebase in their raw format and should otherwise be specified as constant contract variables aiding in their legibility and maintainability.

Compiler Error

Compiler Error findings refer to an error in the structure of the code that renders it impossible to compile using the specified version of the project.

Disclaimer

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.

CERTIK