CERTIK

Security Assessment

The Open Network - Audit 1 (Phase 2)

Aug 1st, 2022

Table of Contents

About

Summary

This report has been prepared for The Open Network to discover issues and vulnerabilities in the source code of the The Open Network - Audit 1 (Phase 2) project. A comprehensive examination has been performed, utilizing Manual Review technique.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Minimize the usage of `auto` keyword, use explicit type specification;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function, specify what `Class::method()` is supposed to call it;
- Provide more comments per each member field;
- Provide more transparency on general communication workflow in code comments.

Overview

Project Summary

Project Name	The Open Network - Audit 1 (Phase 2)
Platform	TON
Language	C++
Codebase	https://github.com/newton-blockchain/ton
Commit	53ac5ee9b6262a7b7fd1568538612095b076ffa8

Audit Summary

Delivery Date	Aug 01, 2022 UTC
Audit Methodology	Manual Review

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Mitigated	Partially Resolved	Resolved
● Critical	0	0	0	0	0	0	0
● Major	1	0	0	0	0	0	1
● Medium	2	0	0	0	0	0	2
● Minor	9	0	0	0	0	0	9
● Informational	7	0	0	0	0	0	7
● Discussion	0	0	0	0	0	0	0

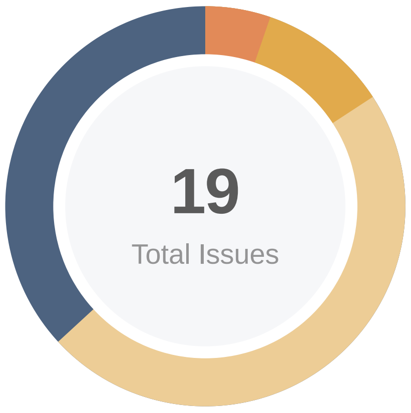
Audit Scope

ID	File	SHA256 Checksum
CAA	catchain/catchain-received-block.cpp	d9eb42eeba577cdac1eae65c8feda121354e633e4fd47c9133aa2dc0d456617b
CA4	catchain/catchain.hpp	00961ef5dcbab3d7cb120c09d63fc3728a461f715584475d2865834e2ba93978
CAI	catchain/catchain-received-block.h	d2e44e321dba5208b587595d3e18e45cc89b3fa67103fca820fed909b757c8aa
CAE	catchain/catchain-receiver-source.cpp	59a658e30f69e4ecc36ef1c3861db90f03adbe793d2da0afa7ec79eba37b2991
CAN	catchain/catchain-received-block.hpp	5fb452004a2a09b8bcb16647831dd6f1f8d887546d8d0b1e596f3a3152edee25
CA0	catchain/catchain-receiver.h	d75df729fee9233e8bd83ea13542fdbb6a19927c3f06b823910451a92afdc499
CAV	catchain/catchain-receiver-source.h	16b0e9e9786c8df60bfcb5345d4bc8969bfc45715cb2dcd22c7af7ec2a414cb6
CA7	catchain/catchain-receiver.hpp	a0529e2bee89198bfd2e1920ca80f2c52ebe421a9760913ec34876524064bd25
CAC	catchain/catchain-block.cpp	84bce86bd6ff6ceb526a799a1bbe60b1ea7ad9ec2d40facb42c5c95d47224a5d
CAS	catchain/catchain-receiver-source.hpp	e1bc0097bd7f84a708aa03b80032c59c9758bd901f1ebf17c6330ef567dd648b
CA1	catchain/catchain.h	63421006799b6d1c5776cd018b136bdc96b0bbea9b17ff0bdc191790e4a14851
CA2	catchain/catchain.cpp	6288081b27b277a7d7e0ec1e60b409131ef771cacaf39873d72aa6dd0df6e387
CA5	catchain/catchain-receiver.cpp	a024dbda47c07fc5d47e09d68dfe8fdb24b4b922d9b081154fe2fa96c9ed7dd9
CAR	catchain/catchain-receiver-interface.h	71aa355db7ce240970f694bbef9d69821d16cdd80d034f1cc339b5d36785a05b
CAY	catchain/catchain-types.h	7f57fb4165e99e5409cccbc9f33a6af2b8b36cb073d784e013fe5f0ece7b0076
CAH	catchain/catchain-block.hpp	9285d6886c0696de35305204de5cfe2e71057652888e1ae57923858d80e6f454

ID	File	SHA256 Checksum
CML	catchain/CMakeLists.txt	fcd8dc002bb46fcc2ed99539baadb826873cc5cf7ff653407d634c2bb41a053d
CAB	projects/ton-catchain/catchain-block.cpp	d5a7312c34e041fafc7832f948abb4debcaa73659f4979f8856274f0729c2051
CAL	projects/ton-catchain/catchain-block.hpp	99ca92bb664232a763d8eab53f6fba9ddff88f4c67336204fb7051e78c0026fa
CAD	projects/ton-catchain/catchain-received-block.cpp	1ff2c5056d949522d17a7c92bbb880d63464a09efa32593a3e6249fdca533a7d
CAO	projects/ton-catchain/catchain-received-block.h	5bb702c62f84cfe4c7562685e2b5f2327b9173e3178ee34c542a947fcf2bd7f5
CAK	projects/ton-catchain/catchain-received-block.hpp	0266e31c9bfc1aeba0532ed6e48088a340bf781a609548c9f6730277fd02043a
CAF	projects/ton-catchain/catchain-receiver-interface.h	79dd2085953d78e352490683ddd33b7f7fb33c93625eb3730b3240246e63ef24
CAU	projects/ton-catchain/catchain-receiver-source.cpp	95d72869fd8182ee9df642626a9df2917ade58bed9e135d1479004b2e9b560f1
CAP	projects/ton-catchain/catchain-receiver-source.h	bc7b333cbff2d11dff1a7d95de3ac15203647d6233c69b399523a11db82bad1e
CAJ	projects/ton-catchain/catchain-receiver-source.hpp	71bd905adf88b1bad8a2a8398c4d7513cb3c89075ff1912a5c7ca4b18b2309c4
CA3	projects/ton-catchain/catchain-receiver.cpp	882698e7f44e63d9a0305f3ae5d2d7b16f287c341e5d4f3877d6c6e86f6b20d5
CA6	projects/ton-catchain/catchain-receiver.h	570ceb8db8d6e54b694220a0e0e6836bc6e50b5aaea4950c3c449603ae17a9a8
CA9	projects/ton-catchain/catchain-receiver.hpp	3131d85f83d8958415e46886b2202b4f09ad147793f482d83247ae80c63e07f9
CA8	projects/ton-catchain/catchain-types.h	88bb1b341db60e995a8db812af60e9c52b6fec4585be3c690395bedbec23e261
CAG	projects/ton-catchain/catchain.cpp	feb87c543d4149148ae517f515fe8f44917785da4bfa20e02f322c879b64a444
CTC	projects/ton-catchain/catchain.h	3e2775d4eec185f8a221f8e370469a3b1ae822d4bcc9d02a9be6ff87d0bfdb77

ID	File	SHA256 Checksum
CTH	projects/ton-catchain/catchain.hpp	ad9db3653a552f2ac58b003729617a024f61a51373172fa78766ffd0ee201b77

Findings



🟥 **Critical**	**0** (0.00%)
🟧 **Major**	**1** (5.26%)
🟨 **Medium**	**2** (10.53%)
🟨 **Minor**	**9** (47.37%)
🟦 **Informational**	**7** (36.84%)
🟩 **Discussion**	**0** (0.00%)

ID	Title	Category	Severity	Status
CA2-01	`top_blocks_` Element Source Bounds Are Not Checked	Logical Issue	● Minor	✓ Resolved
CA2-02	`src_id` Bounds Are Not Checked Before Indexing `blamed_sources_[src_id]`	Logical Issue	● Minor	✓ Resolved
CA5-01	`block->data->deps` Can Be Altered By A Malicious Actor	Logical Issue	● Major	✓ Resolved
CA5-02	Catchain DoS Can Be Provoked By Valid Blocks Flood	Logical Issue	● Medium	✓ Resolved
CA5-03	The Code Assumes The Hash Is 32 Bytes Long	Volatile Code	● Minor	✓ Resolved
CA5-04	Pointer Dereferenced Before Check	Volatile Code	● Minor	✓ Resolved
CA5-05	Pointer Is Not Checked Before Dereferencing In `send_custom_query_data()`	Volatile Code	● Minor	✓ Resolved
CA5-06	Catchain Hangs If Can't `td::RocksDb::destroy()`	Logical Issue	● Minor	✓ Resolved
CA5-09	Variable Name `l` Is Not Recommended	Coding Style	● Informational	✓ Resolved
CA5-10	Redundant Statements	Volatile Code	● Informational	✓ Resolved
CA5-11	Typo	Coding Style	● Informational	✓ Resolved
CAA-01	Cycles In `deps` Are Not Prevented	Logical Issue	● Medium	✓ Resolved
CAA-02	`deps` Size Is Not Limited For Outgoing Blocks	Inconsistency	● Minor	✓ Resolved
CAE-01	`validate_dep_sync()` Is Never Used	Inconsistency	● Minor	✓ Resolved

ID	Title	Category	Severity	Status
CAN-01	`data_hash_` Should Be Named `payload_hash_`	Inconsistency	● Minor	✓ Resolved
CAN-02	`CatChainReceivedBlockImpl::deps_` Should Be Named `vt_`	Inconsistency	● Informational	✓ Resolved
CAT-02	Unmodified Arguments Should Be Declared As `const`	Inconsistency	● Informational	✓ Resolved
CAT-03	`empty()` Can Be Used Instead Of `!size()`	Coding Style	● Informational	✓ Resolved
CAT-04	Magic Numbers	Magic Numbers	● Informational	✓ Resolved

CA2-01 | `top_blocks_` Element Source Bounds Are Not Checked

Category	Severity	Location	Status
Logical Issue	● Minor	catchain/catchain.cpp (base): 38~39	✓ Resolved

Description

```
    auto B = *top_blocks_.get_random();
    CHECK(B != nullptr);
    top_blocks_.remove(B->hash());
    if (B->source() == sources_.size() || !blamed_sources_[B->source()]) {
```

The code allows `B->source() == sources_.size()`, however, all the insertions to `top_blocks_` ensure the valid `src_`.

Recommendation

We recommend replacing the check with explicit:

```
37   CHECK(B->source() < sources_.size());
```

CA2-02 | `src_id` Bounds Are Not Checked Before Indexing

`blamed_sources_[src_id]`

Category	Severity	Location	Status
Logical Issue	● Minor	catchain/catchain.cpp (base): 139~140, 146~147	✓ Resolved

Description

Indexing `blamed_sources_[src_id]` is done without validation of `src_id` correctness.

Recommendation

We recommend moving of

```
146        CHECK(src_id < sources_.size());
```

upper in the code.

CA5-01 | `block->data->deps` Can Be Altered By A Malicious Actor

Category	Severity	Location	Status
Logical Issue	● Major	catchain/catchain-receiver.cpp (base): 220~221	✓ Resolved

Description

Malicious actor can receive a valid block, alter the `block->data->deps` array and rebroadcast the block to other nodes. New block will be valid for catchain nodes but invalid for BCP.

The block validation works this way:

1. Overlay gets the block and calls `CatChainReceiverImpl::receive_message_from_overlay(src, data)`
2. `catchain_block` is fetched from `data`, the rest is treated `payload`
3. `CatChainReceiverImpl::receive_block(src, catchain_block, payload)` is called
4. `CatChainReceiverImpl::validate_block_sync(block, payload)` is called
5. `CatChainReceivedBlock::pre_validate_block(block, payload)` is called
    - `block->incarnation` is verified
    - `block->src` is verified
    - `block->data->deps.size()` is verified to be not more than `max_deps`
    - `block->data->prev->src` is verified to be the same as `block->src`
    - `block->data->prev->height + 1 == block->height` is verified
    - it is verified that there are no `deps` from the same source
    - it is verified that `payload` is not empty
    - **`block->data->deps` array is not checked and not covered by signature**
6. For each `deps` and `prev` as argument `CatChainReceiverImpl::validate_block_sync(dep)` is called
    - Signature (by `dep.src`) of `block_dep_id` is checked. `block_dep_id = { incarnation, source_hash(dep.src), dep.height, dep.data_hash }`
7. Signature (by `src`) of `block_id` is checked. `block_id = { incarnation, source_hash(src), height, hash(payload) }` of original block
8. `CatChainReceiverImpl::create_block(block, payload)` is called and updates `blocks_[hash(block_id)]`

This can lead to network stalling without ability to punish the malicious actor. For example:

1. Malicious actor can clear the `deps` array of received block and force `CatChainReceiverImpl::synchronize_with()` all other nodes. They will accept the modified block and `CatChainReceiverImpl::deliver_block()` immediately to `ValidatorSessionImpl::preprocess_block()` with undefined result.
2. Malicious actor can replace one `dep` with unexisting correctly signed `block_id` and push it to other nodes. They will accept it, will never be able to deliver that block to `ValidatorSession`, and will reject the same block with correct `deps` from other nodes as duplicate.

Recommendation

We recommend covering all the block data with the signature. `block_id` in addition to `hash(payload)` should contain `hash(block->data)`.

Alleviation

[CertiK]: The team heeded to our advice and included `block->data` to `block_id` calculation. That is controlled by `block_hash_covers_data` option and will be turned on after the update of catchain protocol.

CA5-02 | Catchain DoS Can Be Provoked By Valid Blocks Flood

Category	Severity	Location	Status
Logical Issue	● Medium	catchain/catchain-receiver.cpp (base): 97~98	✓ Resolved

Description

Malicious actor can create many valid blocks to overflow internal structures of honest nodes and provoke their crash.

1. Many blocks with increasing `height` can be formed and sent by the malicious actor
2. Honest node Overlay gets the block and calls `CatChainReceiverImpl::receive_message_from_overlay(src, data)`
3. `CatChainReceiverImpl::receive_block()` is called
4. `CatChainReceiverImpl::create_block()` is called
5. `CatChainReceivedBlock::create()` allocates memory in heap

Old blocks unloading is not implemented in general.

Recommendation

We recommend limiting the data structure sizes that accept the data received from untrusted sources (other nodes). We recommend implementing of old blocks unloading mechanism to prevent storing of all the history in memory.

Alleviation

[TON]: To prevent DoS attack we limit maximal height of the block which will be processed by node by `catchain_lifetime * natural_block_production_speed * (1 + number_of_catchain_participants / max_dependencies_size)`, where `catchain_lifetime` is set by `ConfigParam 28 (CatchainConfig)`, `natural_block_production_speed` and `max_dependencies_size` are set by `ConfigParam 29 (ConsensusConfig)` (natural_block_production_speed is calculated as `catchain_max_blocks_coeff / 1000`) and `number_of_catchain_participants` is set from catchain group configuration.

By default, before the catchain protocol update, `catchain_max_blocks_coeff` is set to zero: special value which means that there is no limitation on catchain block height.

CA5-03 | The Code Assumes The Hash Is 32 Bytes Long

Category	Severity	Location	Status
Volatile Code	● Minor	catchain/catchain-receiver.cpp (base): 283~284, 957~958	✓ Resolved

Description

```
283    td::BufferSlice raw_data{32};
284    raw_data.as_slice().copy_from(as_slice(id));
```

Magic number 32 assumes the hash value fits 32 bytes.

Recommendation

We recommend using of `id.as_array().size()` instead.

CA5-04 | Pointer Dereferenced Before Check

Category	Severity	Location	Status
Volatile Code	● Minor	catchain/catchain-receiver.cpp (base): 568~570	✓ Resolved

Description

```
B->written();
CHECK(B);
```

The pointer `B` is dereferenced before it is checked.

Recommendation

We recommend moving of `CHECK(B)` upper.

CA5-05 | Pointer Is Not Checked Before Dereferencing In `send_custom_query_data()`

Category	Severity	Location	Status
Volatile Code	● Minor	catchain/catchain-receiver.cpp (base): 1019~1020, 1029~1030	✓ Resolved

Description

The code assumes `dst` argument is valid and source `S` is non-zero. No check is performed before dereferencing.

Recommendation

We recommend checking the validity of function arguments for a better code maintainability.

CA5-06 | Catchain Hangs If Can't `td::RocksDb::destroy()`

Category	Severity	Location	Status
Logical Issue	● Minor	catchain/catchain-receiver.cpp (base): 1058~1059	✓ Resolved

Description

`CatChainReceiverImpl::destroy()` is supposed to stop trying after 10 failed attempts. However, the counter is not incremented - it will try endlessly.

The same problem affects also `validator/db/archive-slice.cpp` (out of audit scope).

Recommendation

We recommend increasing the `attempt` counter after each try.

CA5-09 | Variable Name `l` Is Not Recommended

Category	Severity	Location	Status
Coding Style	● Informational	catchain/catchain-receiver.cpp (base): 763~764	✓ Resolved

Description

It is not recommended to use `l`, `o` and some other variable names for better code readability.

Recommendation

We recommend renaming the variable to `left`.

CA5-10 | Redundant Statements

Category	Severity	Location	Status
Volatile Code	● Informational	catchain/catchain-receiver.cpp (base): 784~785	✓ Resolved

Description

The linked statement does not affect the functionality of the codebase. Both `r` and `my_vt[i] - vt[i]` are checked to be positive.

Recommendation

We recommend removing of redundant statements.

CA5-11 | Typo

Category	Severity	Location	Status
Coding Style	● Informational	catchain/catchain-receiver.cpp (base): 849~850, 855~856, 883~884	✓ Resolved

Description

"syncronize" is supposed to be "synchronize".

Recommendation

We recommend fixing the typo.

CAA-01 | Cycles In `deps` Are Not Prevented

Category	Severity	Location	Status
Logical Issue	● Medium	catchain/catchain-received-block.cpp (base): 332~337	✓ Resolved

Description

According to Whitepaper the Catchain blocks form a directed acyclic graph using their `deps`. The Catchain code doesn't ensure that.

Let's assume:

1. Malicious actor A creates a BlockA and shares its `block_id_A (incarnation, hash(src), height, hash(payload)` with malicious actor B
2. Malicious actor B creates a BlockB, shares its `block_id_B` with A
3. Malicious actor A uses `block_id_B` as part of `BlockA->data->deps` and broadcasts the block in the Catchain
4. Malicious actor B uses `block_id_A` as part of `BlockB->data->deps` and broadcasts the block in the Catchain
5. Honest node C gets two blocks that depend on each other

Currently the code works correctly in this situation.

Recommendation

We recommend covering all the block data with the signature. `block_id` in addition to `hash(payload)` should contain `hash(block->data)`. Or at least we recommend ensuring that the block received doesn't have cycles in dependencies. We recommend adding the test case ensuring the Catchain code works correctly with cycles in dependencies.

CAA-02 | `deps` Size Is Not Limited For Outgoing Blocks

Category	Severity	Location	Status
Inconsistency	● Minor	catchain/catchain-received-block.cpp (base): 446~448	✓ Resolved

Description

Catchain expects every full block received to have not more than `max_deps` dependencies. `max_deps` is 4 at this moment.

However, when full block is sent to network, the `deps` size is not checked.

Recommendation

We recommend to `CHECK()` that every outgoing block has a limited number of `deps` to catch the potential problem earlier.

CAE-01 | `validate_dep_sync()` Is Never Used

Category	Severity	Location	Status
Inconsistency	● Minor	catchain/catchain-receiver-source.cpp (base): 133~145	✓ Resolved

Description

`CatChainReceiverSourceImpl::validate_dep_sync()` is never used.

`CatChainReceiverImpl::validate_block_sync(dep)` is used instead.

Recommendation

We recommend removing of unused methods.

CAN-01 | `data_hash_` Should Be Named `payload_hash_`

Category	Severity	Location	Status
Inconsistency	● Minor	catchain/catchain-received-block.hpp (base): 139~144	✓ Resolved

Description

`CatChainReceivedBlockImpl` has fields `data_`, `hash_`, `payload_` and `data_hash_`. `data_` field is unused. `data_hash_` in fact holds `payload_` hash.

Recommendation

We recommend removing of unused `data_` field, renaming of `data_hash_` to `payload_hash_`, renaming `hash_` to `block_id_hash_` for better code maintainability.

CAN-02 | `CatChainReceivedBlockImpl::deps_` Should Be Named `vt_`

Category	Severity	Location	Status
Inconsistency	● Informational	catchain/catchain-received-block.hpp (base): 150~156	✓ Resolved

Description

`CatChainReceivedBlockImpl` has fields `block_deps_`, `deps_`, `rev_deps_`, `pending_deps_`.

- `block_deps_` holds pointers to blocks from other nodes
- `deps_` holds **heights** of this block source chain pieces
- `rev_deps_` holds pointers to blocks that depend on this one
- `pending_deps_` holds the number of full blocks required to `CatChainReceiverImpl::run_block()`

`deps_` name is misleading.

Recommendation

We recommend renaming `deps_` to `vt_` or commenting the fields for better code maintainability. `update_deps()` and `get_deps()` should also be renamed.

CAT-02 | Unmodified Arguments Should Be Declared As `const`

Category	Severity	Location	Status
Inconsistency	● Informational	catchain/catchain-received-block.cpp (base): 362~364, 400~401; catchain/catchain-receiver.cpp (base): 218~219, 615~616	✓ Resolved

Description

If the function doesn't modify the argument, it should accept it as `const *` or `const &`.

```
362  td::Status CatChainReceivedBlock::pre_validate_block(CatChainReceiver *chain,
363               tl_object_ptr<ton_api::catchain_block> &block, td::Slice payload) {
```

This and some other functions don't modify the arguments.

Methods that don't modify the object state should be declared as `const`.

If cycle doesn't modify the container elements, the variable `X` should be declared as `const`:

```
400    for (auto &X : block->data_->deps_) {
```

Recommendation

We recommend changing the function declarations like:

```
362  td::Status CatChainReceivedBlock::pre_validate_block(const CatChainReceiver *chain,
363     const tl_object_ptr<ton_api::catchain_block> &block, const td::Slice& payload)
const {
```

We recommend using of `const` modifier wherever it is possible.

We recommend avoiding using of `auto` typename wherever specific type is not very complex.

This increases code readability and maintainability.

CAT-03 | `empty()` Can Be Used Instead Of `!size()`

Category	Severity	Location	Status
Coding Style	● Informational	catchain/catchain-receiver-source.cpp (base): 27~28, 63~64; catchain/catchain-receiver-source.hpp (base): 85~86	✓ Resolved

Description

```
85      if (!blocks_.size()) {
```

`blocks_.empty()` can be used instead of `!blocks_.size()`.

Recommendation

We recommend changing the container method for better code readability.

CAT-04 | Magic Numbers

Category	Severity	Location	Status
Magic Numbers	● Informational	catchain/catchain-receiver.cpp (base): 92, 485~486, 604~607, 669~670, 695~697, 761~762, 869~870, 877~878, 891~892, 930~931, 944~945, 966~967, 966~967, 980~981, 991~992; catchain/catchain-receiver.h (base): 48~49	✓ Resolved

Description

Magic numbers are used directly in code.

Recommendation

We recommend introducing constant values:

```
DEFAULT_MAX_NEIGHBOURS = 5
EXPECTED_UNSAFE_INITIAL_SYNC_DURATION = 300.0
EXPECTED_INITIAL_SYNC_DURATION = 5.0
OVERLAY_MAX_ALLOWED_PACKET_SIZE = 16 * 1024 * 1024
NEIGHBOURS_ROTATE_INTERVAL_MIN = 60
NEIGHBOURS_ROTATE_INTERVAL_MAX = 120
```

etc.

Optimizations

ID	Title	Category	Severity	Status
CA5-07	`map::emplace()` Result Can Be Used Instead Of `map::find()`	Gas Optimization	● Optimization	✓ Resolved
CA5-08	`vector::reserve()` Can Minimize Relocations	Gas Optimization	● Optimization	✓ Resolved
CAT-01	Unnecessary Structures Copying	Gas Optimization	● Optimization	✓ Resolved

CA5-07 | `map::emplace()` Result Can Be Used Instead Of `map::find()`

Category	Severity	Location	Status
Gas Optimization	● Optimization	catchain/catchain-receiver.cpp (base): 177~178, 191~194	✓ Resolved

Description

`map::emplace()` returns `std::pair<iterator,bool>`. `iterator` can be used to avoid the call to `map::find()`.

Recommendation

We recommend rewriting the code this way:

```
176      auto r = blocks_.emplace(hash, CatChainReceivedBlock::create(std::move(block),
std::move(payload), this));
177      return r.first->second.get();
```

CA5-08 | `vector::reserve()` Can Minimize Relocations

Category	Severity	Location	Status
Gas Optimization	● Optimization	catchain/catchain-receiver.cpp (base): 479~480	✓ Resolved

Description

The resulting size of `ids` vector is known in advance. Using of `vector::push_back()` leads to unnecessary relocations.

Recommendation

We recommend using of `vector::reserve()`.

CAT-01 | Unnecessary Structures Copying

Category	Severity	Location	Status
Gas Optimization	● Optimization	catchain/catchain-receiver.cpp (base): 218~219, 433~434; catchain/catchain.cpp (base): 299~301; catchain/catchain.hpp (base): 76~77	✓ Resolved

Description

In many functions non-trivial arguments are passed by value. This leads to unnecessary copying.

Non-trivial arguments are `PublicKeyHash`, `CatChainSessionId`, `td::Slice`, `td::SharedSlice` and others.

Recommendation

We recommend passing non-trivial arguments with `std::move()` or declaring the arguments as `const &`.

Appendix

Finding Categories

Gas Optimization

Gas Optimization findings 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.

Logical Issue

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

Volatile Code

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

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather 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.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you (“Customer” or the “Company”) in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK’s prior written consent in each instance.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing 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. CertiK’s position is that each company and individual are responsible for their own due diligence and continuous security. CertiK’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND

"AS AVAILABLE" AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER'S OR ANY OTHER PERSON'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER'S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK'S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER'S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED "AS IS" AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK'S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

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