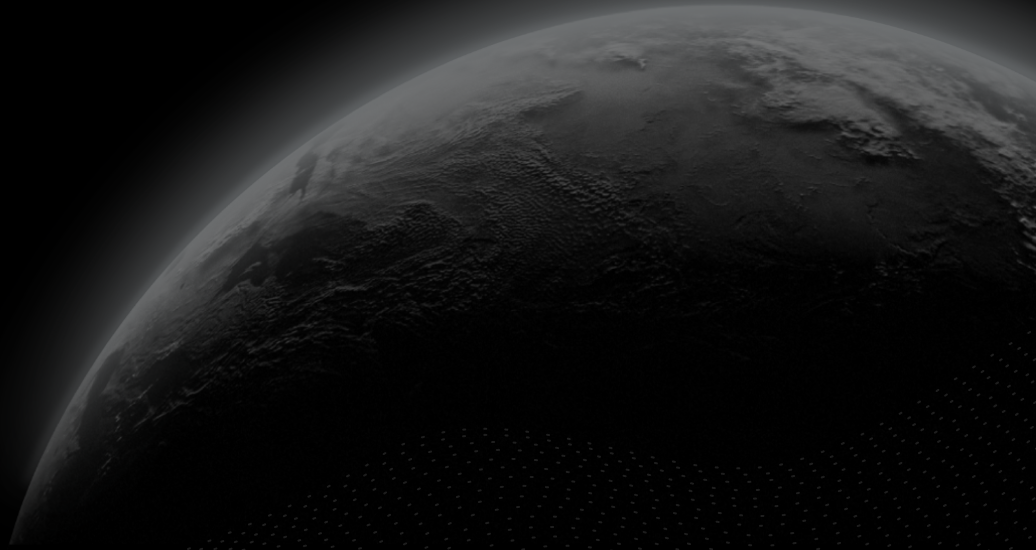




Security Assessment

TON Stake

CertiK Assessed on Apr 16th, 2024





CertiK Assessed on Apr 16th, 2024

TON Stake

The security assessment was prepared by CertiK, the leader in Web3.0 security.

Executive Summary

TYPES
Staking

ECOSYSTEM
TON

METHODS
Manual Review

LANGUAGE
FunC

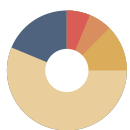
TIMELINE
Delivered on 04/16/2024

KEY COMPONENTS
N/A

CODEBASE
[update](#)
[base](#)
View All in Codebase Page

COMMITTS
[c6b3a713e24e4c2dcffc5e33ca47533df115c3dc](#)
[074d7275d0641ad211f8b4868cdd1dc1be704c4e](#)
View All in Codebase Page

Vulnerability Summary



16

Total Findings

15

Resolved

0

Mitigated

0

Partially Resolved

1

Acknowledged

0

Declined

1 Critical

1 Resolved

Critical risks are those that impact the safe functioning of a platform and must be addressed before launch. Users should not invest in any project with outstanding critical risks.

1 Major

1 Acknowledged

Major risks can include centralization issues and logical errors. Under specific circumstances, these major risks can lead to loss of funds and/or control of the project.

2 Medium

2 Resolved

Medium risks may not pose a direct risk to users' funds, but they can affect the overall functioning of a platform.

9 Minor

9 Resolved

Minor risks can be any of the above, but on a smaller scale. They generally do not compromise the overall integrity of the project, but they may be less efficient than other solutions.

3 Informational

3 Resolved

Informational errors are often recommendations to improve the style of the code or certain operations to fall within industry best practices. They usually do not affect the overall functioning of the code.

TABLE OF CONTENTS | TON Stake

■ Summary

[Executive Summary](#)

[Vulnerability Summary](#)

[Codebase](#)

[Audit Scope](#)

[Approach & Methods](#)

■ Findings

[SLU-01 : Single owner can control the `multisig`](#)

[SLH-03 : Centralization Risks](#)

[SLB-03 : `nominator_proxy` can steal the deposit](#)

[SLT-03 : `flood` is not decreased for expired `pending_queries`](#)

[SLG-01 : `OP::RETURN_UNSTAKE_REQUEST` is unused by `unstake_request`](#)

[SLH-04 : `end_parse\(\)` Is Missing](#)

[SLH-05 : `ton_amount` argument of `financial.mint\(\)` is misleading](#)

[SLH-06 : Initial `jetton_total_supply` of `financial` is unclear](#)

[SLI-03 : Lack of `validator_wc` validation](#)

[SLU-03 : `multisig` is vulnerable to `Replay-Failed` attack](#)

[SLU-04 : Lack of `commission_factor` validation](#)

[SLU-05 : Arguments don't have type specifiers](#)

[SLV-01 : Inconsistent type of `creator_addr` in `pending_queries` item](#)

[SLI-04 : `wallet_id` variable is shadowed](#)

[SLU-09 : Inaccurate comments](#)

[SLU-10 : Usage of Magic Numbers](#)

■ Optimizations

[SLB-02 : No reason to parse `in_msg_full`](#)

■ Appendix

■ Disclaimer

CODEBASE | TON Stake

Repository

update

base









Commit

c6b3a713e24e4c2dcffc5e33ca47533df115c3dc

074d7275d0641ad211f8b4868cdd1dc1be704c4e

AUDIT SCOPE | TON Stake

8 files audited ● 8 files without findings

ID	Repo	File	SHA256 Checksum
● JET	tonstakeapp/smart-contracts	 imports/jetton-utils.fc	a3a63d94189cd4393c2c5bf0aadbb511d98 1325a457b483f9e25dfaf0697c1f7
● NOM	tonstakeapp/smart-contracts	 imports/nominator-proxy-utils.fc	573dabaa7027f486cbc96ceba19fa16bdd56 9931f74dacfe7b9550c49acb7900
● UNS	tonstakeapp/smart-contracts	 imports/unstake-request-utils.fc	93548e7af409166a8c43de45503e04fd031 7fba37bba472d958870dbfdc65af
● ADM	tonstakeapp/smart-contracts	 admin_multisig.fc	46bfae18aa3cd4c50b457d4fc5b16cbf4b61 bc84ef40d66cf7736af1c334bb43
● FIN	tonstakeapp/smart-contracts	 financial.fc	c1877c8293ac6509ad760f6792cc41e993d ba31b4cbba8360cef43b10cd725c3
● NOI	tonstakeapp/smart-contracts	 nominator_proxy.fc	4b3218977225f2fdb1e1edf95ecf3242f87f0 15ff49a8aeb4eb04b269d63b486
● TRA	tonstakeapp/smart-contracts	 transaction_multisig.fc	7b8c7c6e383953ecccf2b5965e6bb4c31d3 ef50cfcbc3940c5bed58c08a43b5a
● UNT	tonstakeapp/smart-contracts	 unstake_request.fc	06872b56a113cd07dd486e6ff63aa237f257 053211233a271d2ebeb5cd84b663

APPROACH & METHODS | TON Stake

This report has been prepared for TON Stake to discover issues and vulnerabilities in the source code of the TON Stake project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Manual Review techniques.

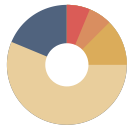
The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

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:

- Testing the smart contracts against both common and uncommon attack vectors;
- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

FINDINGS | TON Stake



16

Total Findings

1

Critical

1

Major

2

Medium

9

Minor

3

Informational

This report has been prepared to discover issues and vulnerabilities for TON Stake. Through this audit, we have uncovered 16 issues ranging from different severity levels. Utilizing the techniques of Manual Review to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
SLU-01	Single Owner Can Control The <code>multisig</code>	Logical Issue	Critical	● Resolved
SLH-03	Centralization Risks	Centralization	Major	● Acknowledged
SLB-03	<code>nominator_proxy</code> Can Steal The Deposit	Volatile Code	Medium	● Resolved
SLT-03	<code>flood</code> Is Not Decreased For Expired <code>pending_queries</code>	Denial of Service	Medium	● Resolved
SLG-01	<code>OP::RETURN_UNSTAKE_REQUEST</code> Is Unused By <code>unstake_request</code>	Volatile Code	Minor	● Resolved
SLH-04	<code>end_parse()</code> Is Missing	Coding Style	Minor	● Resolved
SLH-05	<code>ton_amount</code> Argument Of <code>financial.mint()</code> Is Misleading	Inconsistency	Minor	● Resolved
SLH-06	Initial <code>jetton_total_supply</code> Of <code>financial</code> Is Unclear	Volatile Code	Minor	● Resolved
SLI-03	Lack Of <code>validator_wc</code> Validation	Volatile Code	Minor	● Resolved
SLU-03	<code>multisig</code> Is Vulnerable To <code>Replay-Failed</code> Attack	Denial of Service	Minor	● Resolved
SLU-04	Lack Of <code>commission_factor</code> Validation	Volatile Code	Minor	● Resolved

ID	Title	Category	Severity	Status
SLU-05	Arguments Don't Have Type Specifiers	Volatile Code	Minor	● Resolved
SLV-01	Inconsistent Type Of <code>creator_addr</code> In <code>pending_queries</code> Item	Inconsistency	Minor	● Resolved
SLI-04	<code>wallet_id</code> Variable Is Shadowed	Coding Style	Informational	● Resolved
SLU-09	Inaccurate Comments	Coding Issue	Informational	● Resolved
SLU-10	Usage Of Magic Numbers	Coding Style	Informational	● Resolved

SLU-01 | SINGLE OWNER CAN CONTROL THE `multisig`

Category	Severity	Location	Status
Logical Issue	● Critical	<code>admin_multisig.fc (base): 216; transaction_multisig.fc (base): 70</code>	● Resolved

Description

`transaction_multisig` and `admin_multisig` work this way:

1. `(wallet_id, query_id, payload)` message is signed by several owners with public keys from `owner_infos` number `i1, i2, etc.`
2. `(root_i, signatures, (wallet_id, query_id, payload))` message is signed by owner with public key `root_i` from `owner_infos`
3. `(root_signature, (root_i, signatures, (wallet_id, query_id, payload))` message is sent to the multisig wallet
4. the wallet ensures that all the signatures are valid and the number of them is `>= k`
5. the wallet doesn't ensure that the signatures belong to different owners, all of them can be duplicating the `root_i` signature only

Scenario

1. The owner X of public key `owner_infos[x]` signs any valid `payload` and gets `signatureX`
2. X constructs the `signatures` cell like `(signatureX, x, ref to (signatureX, x, ... k times))`
3. X signs the message `(x, signatures, payload)` and gets `root_signatureX`
4. X sends the external message to the multisig wallet
5. The multisig wallet executes the message

Recommendation

We recommend ensuring the signatures are from different owners this way:

```
int check_signatures(cell public_keys, cell signatures, int hash) inline_ref {
    int cnt = 0;
    int cnt_bits = 0;
    do {
        slice cs = signatures.begin_parse();
        slice signature = cs~load_bits(512);

        int i = cs~load_uint(8);
        signatures = cs~load_dict();

        (slice public_key, var found?) = public_keys.udict_get?(8, i);
        throw_unless(ERROR::PUBLIC_KEY_NOT_FOUND, found?);
        throw_unless(ERROR::INVALID_SIGNATURE, check_signature(hash, signature,
public_key.preload_uint(256)));

        int mask = (1 << i);
        throw_unless(ERROR::DUPLICATING_PUBLIC_KEY, (cnt_bits & mask) == 0);
        cnt_bits |= mask;
        cnt += 1;
    } until (cell_null?(signatures));

    return cnt;
}
```

SLH-03 | CENTRALIZATION RISKS

Category	Severity	Location	Status
Centralization	● Major	financial.fc (base): 411	● Acknowledged

Description

In the contract `financial` the role `admin_address` has authority over the functions:

- change `admin_address`, `commission_address`, and `transaction_address`
- change jetton `content`
- change `commission_factor`, which can be set even higher than 100%
- withdraw commission to `commission_address`
- update the `financial` code

The role `transaction_address` can send a transaction from `financial` to any address with any amount with any payload. `ton_total_supply` is not updated in this case.

Any compromise to the `admin_address` and/or `transaction_address` may allow the hacker to take advantage of this authority and steal all the assets.

It is supposed that `admin_address` and `transaction_address` are controlled by a contract with multi-signature functionality. `admin_address` also has a 12-hour timelock.

Recommendation

The risk describes the current project design and potentially makes iterations to improve the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked.

Renouncing the ownership or removing the risky functionality can be considered fully resolved.

Alleviation

[Project Team]: Now `admin_address` and `transaction_address` are managed by multisigs.

Only `admin_multisig` can change both multisigs and other parameters but to do this you need to wait 90 hours and send the request again. During this time such a request can be canceled. This protects against compromising the private keys of multisig owners and against erroneous changes. We may cancel any requests while notifying users of potential problem.

`Nominator_proxy` also implements decentralized withdrawal of funds from the nominator. Any user can send a withdrawal request. As a result users can easily withdraw their funds if the private keys of multisig owners are compromised.

`Transaction_multisig` can only request a transfer to the `nominator_pool` address. This protects the financial contract from transfers to any address. The smart contract also prohibits the transfer of funds intended for withdrawal after 36-72 hours which further protects the user's funds.

Yes, there is still a risk of updating the contract but in the future this will be solved by adding a DAO contract.

SLB-03 | `nominator_proxy` CAN STEAL THE DEPOSIT

Category	Severity	Location	Status
Volatile Code	● Medium	<code>nominator_proxy.fc</code> (base): 159	● Resolved

Description

`nominator_proxy` works this way:

1. gets at least `MIN_NOMINATOR_STAKE` from `financial`
2. sends them to `nominator` with `d` message
3. at least `TIME::TEN_HOURS` passes
4. withdrawer sends `OP::SIMPLE_TRANSFER` to `nominator_proxy` with [0.5, 2] tons
5. `nominator_proxy` sends a `w` message to `nominator`
6. `nominator` sends back the deposited amount with a reward
7. if the amount is at least `MIN_NOMINATOR_STAKE - 1`, `nominator_proxy` forwards it to `financial` with `OP::ACCEPT_REWARD` message
8. else `nominator_proxy` forwards it to the withdrawer as excesses

`nominator` is out of this audit scope. `nominator` doesn't guarantee that the returned stake with a reward is not less than the deposited amount. The amount can be less due to fine distribution or other reasons. In this case, all the deposit will be forwarded to the withdrawer.

Also, if the withdrawal request is processed longer than 1 minute, another withdrawer can gather excesses.

Recommendation

We recommend not returning the excesses to the withdrawer or returning only if `msg_value < TWO_TON`.

SLT-03 | `flood` IS NOT DECREASED FOR EXPIRED `pending_queries`

Category	Severity	Location	Status
Denial of Service	● Medium	admin_multisig.fc (update1): 290	● Resolved

Description

In `admin_multisig`, `flood` parameter of corresponding `owners_addresses_info` is increased each time a new `pending_query` is created via internal message. If the query gets enough signatures from other owners, it is executed, and `dec_flood` is performed. However, if the query doesn't pass and gets deleted via `cleanup_queries()`, the `flood` is not updated for the creator. After 10 failed queries, the owner is no longer able to submit queries.

Recommendation

We recommend updating the `owners_addresses_info` during `cleanup_queries()` if query starts from 1.

SLG-01 | `OP::RETURN_UNSTAKE_REQUEST` IS UNUSED BY `unstake_request`

Category	Severity	Location	Status
Volatile Code	● Minor	<code>unstake_request.fc (base): 14</code>	● Resolved

Description

`OP::RETURN_UNSTAKE_REQUEST` is never used by `unstake_request`. `recv_internal()` assumes that the message is either `OP::DEPLOY_UNSTAKE_REQUEST`, or `OP::RETURN_UNSTAKE_REQUEST`, but doesn't check that.

Recommendation

We recommend explicitly checking if `op == OP::RETURN_UNSTAKE_REQUEST` in `recv_internal()`.

SLH-04 | `end_parse()` IS MISSING

Category	Severity	Location	Status
Coding Style	● Minor	financial.fc (base): 96	● Resolved

Description

`end_parse()` checks if slice is empty, otherwise throws an exception. It allows to ensure the slice has the expected data structure.

Several contracts affected.

Recommendation

We recommend calling `end_parse()` to ensure the slice doesn't contain more data.

SLH-05 | `ton_amount` ARGUMENT OF `financial.mint()` IS MISLEADING

Category	Severity	Location	Status
Inconsistency	● Minor	financial.fc (base): 149	● Resolved

Description

In `financial` contract `mint()` accepts `ton_amount` argument. The argument meaning is unclear. Zero is always passed to the function. `forward_ton_amount` is hardcoded as 100. Excesses will be returned to `to_address`.

Recommendation

We recommend removing the argument or clarifying the intended behavior.

SLH-06 | INITIAL `jetton_total_supply` OF `financial` IS UNCLEAR

Category	Severity	Location	Status
Volatile Code	● Minor	financial.fc (base): 253	● Resolved

Description

`financial` calculates the stake by the formula:

```
253      int stake_jetton_amount = muldiv(jetton_total_supply, stake_ton_amount,
ton_total_supply);
```

However, it is unclear what are the initial values of `jetton_total_supply` and `ton_total_supply`. Zero values prevent deposits, non-zero values can lead to [inflation attack](#). Inconsistent `ton_total_supply` will lead to unfair deposit calculation.

Recommendation

We recommend providing the initial values or clarifying the intended behavior.

Alleviation

[Project Team]: The initial values are set before the contract is deployed. In our case the initial value for both pools was 1 TON (1 000 000 000).

SLI-03 | LACK OF `validator_wc` VALIDATION

Category	Severity	Location	Status
Volatile Code	● Minor	<code>transaction_multisig.fc</code> (base): 119	● Resolved

Description

`transaction_multisig` gets the `validator_addr` as part of `OP::SEND_TON_FROM_FINANCIAL` message. But it doesn't ensure the validator is deployed in masterchain.

Recommendation

We recommend checking the correctness of all input values.

SLU-03 | `multisig` IS VULNERABLE TO Replay-Failed ATTACK

Category	Severity	Location	Status
Denial of Service	● Minor	<code>admin_multisig.fc (base): 257; transaction_multisig.fc (base): 111</code>	● Resolved

Description

`transaction_multisig` and `admin_multisig` are calling `set_gas_limit(100000)`.

If, after `accept_message` or `set_gas_limit`, some error is thrown (either in `ComputePhase` or `ActionPhase`), the transaction will be written to the blockchain, and fees will be deducted from the contract balance. However, storage will not be updated, and actions will not be applied.

As a result, if the contract accepts an external message and then throws an exception due to an error in the message data or the sending of an incorrectly serialized message, it will pay for processing but will have no way of preventing message replay. The same message will be accepted by the contract over and over until it consumes the entire balance.

Recommendation

We recommend updating the `completed_queries` in storage immediately after `accept_message()` / `set_gas_limit()` or using `try-catch` block:

```
set_gas_limit(100000);

try {
    ;; process refs
} catch (x, y) {
}

completed_queries~dict_set_builder(64, query_id, begin_cell().store_int(0, 1));

set_data(pack_state(completed_queries, owner_infos, k, n, wallet_id,
financial_address, nominator_proxy_code, nominator_pool_code));
commit();
```

SLU-04 | LACK OF `commission_factor` VALIDATION

Category	Severity	Location	Status
Volatile Code	● Minor	admin_multisig.fc (base): 158; financial.fc (base): 268	● Resolved

Description

`admin_multisig` allows to send `OP::CHANGE_COMMISSION_FACTOR` message to `financial` to change `commission_factor`. `financial` expects it to be less than `COMMISSION_BASE`, however, that is not checked.

`admin_multisig` stores `commission_factor` as `int16` and allows negative values, however, sends to `financial` and reads from `msg_body` as `uint16`. This allows an implicit overflow.

Recommendation

We recommend limiting the `commission_factor` by a reasonable value and making the types consistent.

SLU-05 | ARGUMENTS DON'T HAVE TYPE SPECIFIERS

Category	Severity	Location	Status
Volatile Code	● Minor	financial.fc (base): 149; nominator_proxy.fc (base): 79	● Resolved

Description

```
79 () on_bounce (slice in_msg_body, int msg_value, balance) impure {
```

`balance` argument of `on_bounce()` function of `nominator_proxy` contract doesn't have type specified. Many arguments affected.

Recommendation

We recommend explicitly specifying the used argument type.

SLV-01 | INCONSISTENT TYPE OF `creator_addr` IN `pending_queries` ITEM

Category	Severity	Location	Status
Inconsistency	● Minor	admin_multisig.fc (update2): 156	● Resolved

Description

In `admin_multisig` the `pending_queries` item contains `creator_addr` address.

- it is loaded by `load_int(256)` in `unpack_query_data()`
- it is stored by `store_int(creator_addr, 256)` in `update_pending_queries()`
- it is loaded by `load_uint(256)` in `cleanup_queries()`
- it is used as unsinged key of `owners_addresses_info` by `udict_set_builder(256)` in `dec_flood()`

Recommendation

We recommend using `uint(256)` representation of addresses.

SLI-04 | `wallet_id` VARIABLE IS SHADOWED

Category	Severity	Location	Status
Coding Style	● Informational	transaction_multisig.fc (base): 124	● Resolved

Description

In `transaction_multisig` the inner scope variable `wallet_id` shadows another one in outer scope. This can lead to confusion.

Recommendation

We recommend avoiding variables shadowing.

SLU-09 | INACCURATE COMMENTS

Category	Severity	Location	Status
Coding Issue	● Informational	admin_multisig.fc (base): 229; nominator_proxy.fc (base): 113	● Resolved

Description

Some comments are inaccurate or outdated.

```
83      ;; empty message triggers init
```

In fact, nothing is triggered on an empty message.

```
113      ;; ignore all bounced messages
```

In fact, bounced messages are not ignored.

Recommendation

We recommend updating the comments.

SLU-10 | USAGE OF MAGIC NUMBERS

Category	Severity	Location	Status
Coding Style	● Informational	admin_multisig.fc (base): 119; nominator_proxy.fc (base): 74, 95; unstake_request.fc (base): 67	● Resolved

Description

Different magic numbers are used as-is in code.

- `financial` op-codes are used as numbers in `admin_multisig`, `nominator_proxy`, and `unstake_request`
- 100, 119 are used instead of `ACTION::DEPOSIT` / `ACTION::WITHDRAW` in `nominator_proxy`

Recommendation

We recommend declaring and using constants to improve code maintainability and readability.

OPTIMIZATIONS | TON Stake

ID	Title	Category	Severity	Status
<u>SLB-02</u>	No Reason To Parse <code>in_msg_full</code>	Gas Optimization	Optimization	● Resolved

SLB-02 | NO REASON TO PARSE `in_msg_full`

Category	Severity	Location	Status
Gas Optimization	● Optimization	nominator_proxy.fc (base): 120-125	● Resolved

Description

```
120     cs~load_msg_addr(); ;; skip dst
121     cs~load_coins(); ;; skip value
122     cs~skip_bits(1); ;; skip extracurrency collection
123     cs~load_coins(); ;; skip ihr_fee
124     cs~load_coins(); ;; skip fwd_fee
```

There is no reason to read `in_msg_full`. No values are used.

Recommendation

We recommend removing the redundant code.

APPENDIX | TON Stake

Finding Categories

Categories	Description
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.
Coding Style	Coding Style findings may not affect code behavior, but indicate areas where coding practices can be improved to make the code more understandable and maintainable.
Coding Issue	Coding Issue findings are about general code quality including, but not limited to, coding mistakes, compile errors, and performance issues.
Denial of Service	Denial of Service findings indicate that an attacker may prevent the program from operating correctly or responding to legitimate requests.
Inconsistency	Inconsistency findings refer to different parts of code that are not consistent or code that does not behave according to its specification.
Volatile Code	Volatile Code findings refer to segments of code that behave unexpectedly on certain edge cases and may result in vulnerabilities.
Logical Issue	Logical Issue findings indicate general implementation issues related to the program logic.
Centralization	Centralization findings detail the design choices of designating privileged roles or other centralized controls over the code.

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 | CERTIK

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.

CertiK | Securing the Web3 World

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.

