Security Assessment

Stader

Dec 6th, 2021

Table of Contents

Summary

Overview

- Project Summary
- Audit Summary
- Vulnerability Summary
- Audit Scope

Findings

- GLOBAL-01 : Centralization Risk For Role `manager`
- CKP-01 : Centralization Risk for Role `delegator`
- CKP-02 : Incorrect Event Log
- COA-01 : Pseudo-random Redelegation Destination
- COA-02 : Centralization Risk for role `scc`
- COA-03 : Potential Ineffective Validator Number Check
- COA-04 : Inconsistent Result Types
- COA-05 : Potential Integer Overflow
- COA-06 : Lack of Sanity Check
- COC-01 : Lack of Sanity Check
- CON-01 : Centralization Risk for role `pools_contract`
- CON-02 : Lack of Input Validation
- COR-01 : Centralization Risk
- COR-02 : Inaccurate Query Result
- COR-03 : Potential Integer Overflow
- COR-04 : Inconsistent Airdrop Amount Calculation
- COR-05 : Lack of Input Validation
- COS-01 : Centralization Risk for role `pools_contract`
- COS-02 : Potential Integer Overflow
- COS-03 : Logic of Accrued Rewards
- COT-01 : Potential Integer Overflow
- HEL-01 : Potential Zero Shares Per Token Ratio

Appendix

Disclaimer

About

Summary

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

- 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.

Overview

Project Summary

Project Name	Stader
Platform	Terra
Language	Rust
Codebase	https://github.com/stader-labs/stader-protocol-v1
Commit	• 3ca5b916850acdc0b5406b5c1f70c6045a662a65

Audit Summary

Delivery Date	Dec 06, 2021
Audit Methodology	Static Analysis, Manual Review
Key Components	

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
Critical	0	0	0	0	0	0
Major	7	0	0	4	0	3
Medium	2	0	0	0	0	2
Minor	8	0	0	2	0	6
Informational	5	0	0	1	0	4
Discussion	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
CKP	contracts/cf-scc/src/contract.rs	10793d553544c8ec61f1a878ad6f1e564989ac543df5837e7cfd3ece275fd468
ERR	contracts/cf-scc/src/error.rs	354ade4f12130c9ad887f93fe80ac67d3e1a5a8f1cb80d5a3a9200b1b624abe5
LIB	contracts/cf-scc/src/lib.rs	f19dd9f92dbdcfaf94e19491336606a46a930e767da1b49b0ce6c775020b061e
MSG	contracts/cf-scc/src/msg.rs	735b668588e72614fdfd80a011f17dac9467428777bf7e229fe63747e8805804
STA	contracts/cf-scc/src/state.rs	2a14c470ea6405d3647cbd4a7bc462133390f36256aed960ef7fd852f8abf335
CON	contracts/delegator/src/contract.rs	f299ca3dbe1a6e12f90b4e47cab8e3a33d4ea8c7973a2a470e94e5efee7e5c0b
ERO	contracts/delegator/src/error.rs	144d70445aa29f3df6ce16187fbe4bef73740dc33103eb280ddb20208c2c2701
LIS	contracts/delegator/src/lib.rs	608d2d252f602d37929016a73821564b8f074f2dc738a8d5a00addb92cb9b0e6
MSS	contracts/delegator/src/msg.rs	f1ea3c69cb7b78f08044781beadb5b6938b115f11287ad085368242fde39474a
REQ	contracts/delegator/src/request_validation.rs	ca7011c36c2250cadc7a43b347bb6905c044040be86615d3e756574aff25f4ee
STT	contracts/delegator/src/state.rs	efa07b9d69c841f53280993262993fb089bb1d721851a35cf79381409d14c8d7
COT	contracts/pools/src/contract.rs	7d74f854e18294bd4a28171c5b25a124668b8418b0225c8d785a5964bad92335
ERS	contracts/pools/src/error.rs	70f187d6dc12d78d73b5f3e8c8899c5c6a57607b8d009ae8d050ead19c763cd2
LIR	contracts/pools/src/lib.rs	608d2d252f602d37929016a73821564b8f074f2dc738a8d5a00addb92cb9b0e6
MSR	contracts/pools/src/msg.rs	9172c6971c94d3f6b887f6a54e84d9af47ee0c459cbeae0b74709bbe90b7f0f7
REU	contracts/pools/src/request_validation.rs	8d95c2f895c6aa4fe8b5b664b7d9953192ac696f9d78af0e0e7b821593a9c1f1

ID	File	SHA256 Checksum
STE	contracts/pools/src/state.rs	1cf743cb37f79e2e47243cdbb092ee68a049dba56c5ddb8bb929475c244e0605
COR	contracts/scc/src/contract.rs	5da1312497eb1f51a2b345347761254089112c1ce59f910b3745b63f575dab39
ERC	contracts/scc/src/error.rs	e86ca082e4824a20226d9a995fe27849db5c56d2420118fca133f025055ecd06
HEL	contracts/scc/src/helpers.rs	043750c8647a93f74fcdece83677b128c64a48a4a2deee8e075a628395d38454
LIC	contracts/scc/src/lib.rs	c50b7465a24e09ba3d941565beefaba6ebbb5d3389f6f955e7ceb4d27ea8d5cc
MSC	contracts/scc/src/msg.rs	c56db695857cb4aaa588acaec1350b650ac432ff74eb880477cff6391098 01d9
STS	contracts/scc/src/state.rs	30d36c7933c3b6ecf7da6d65d75b8c54372a7ba598b8df6be9f2ee60a522dc09
USE	contracts/scc/src/user.rs	96d04aa29347dc220558d3d243fc97ec9c408fbcf11600a3a984907b65631ba4
COA	contracts/sic-auto-compound/src/contract.rs	3b46bbaec842cdfcae855f0205f9def9d40df861d2ba9ba44baebe0c29ec7e91
ERI	contracts/sic-auto-compound/src/error.rs	ab6c0847e617ca78126248cf94e6bd36a1763f93660cc329fcf3fd132201675f
HEP	contracts/sic-auto-compound/src/helpers.rs	4c52a408b122e16b284fc53a5f24ff72e7e61f8ce8ce958a64d45e7d976debb6
LII	contracts/sic-auto-compound/src/lib.rs	473a971c7b097df6d7d279b80fa66ef6fda2e2ebd431a27fc7348394e7b76bca
MSI	contracts/sic-auto-compound/src/msg.rs	09e68b5dc4ca5ad2737c7d15db5b2061c174102178371ab59ac55dcfd4405022
STR	contracts/sic-auto-compound/src/state.rs	b7129145b5a728ab0e32969d0528d34a8093453ce225ddf96259d7aed6c21cc9
COC	contracts/stader-hub/src/contract.rs	7adbeae3a8b6aaf6d7d7289fd3f83933f6dcf9016997cc9b628a36a07fdcf6f1
ERT	contracts/stader-hub/src/error.rs	de7f6723a83834c87387fc54fb5eb57c018e91dd581dab8dbdbf42397fbf2692
LIT	contracts/stader-hub/src/lib.rs	3e62212e93a5008c18b7fc78e39daf6821c044ee6b54ae2f8d0f30ac15f20fd8

ID	File	SHA256 Checksum
MST	contracts/stader-hub/src/msg.rs	6d10282b27a75046a82ca7ec79e931a40a5f0f9bf3f73522bc68b37ac9e9d8d6
STC	contracts/stader-hub/src/state.rs	739ec2eedf0a404cfde388a3254bb263d82f334a6421f43e975465a92015a693
COS	contracts/validator/src/contract.rs	17ee4a66b051f5cc218a4d5afd562e7a79b65b23edf331bad41cce984d63d8cf
ERV	contracts/validator/src/error.rs	24ec8f1f0a064cdfbb17e1a3f0427cd83be6cf375ab6b90df81ab5e42b1b4cdd
LIV	contracts/validator/src/lib.rs	c5c033e466a62c07f2d45d8f092ceb315ccc484e0c48b8977260cff614f35a15
MSV	contracts/validator/src/msg.rs	73d1d2b525c01c1680f68af7be46bf629f702074df03a10006c3ae0be09d51cd
OPE	contracts/validator/src/operations.rs	d52cd103486c7812789170cc6c5244eeb24cd9cba355499733a9385bc8e99eb4
REE	contracts/validator/src/request_validation.rs	27b0dfb36e490168107633a8e3847289537cac6c2c4b1c89793a7135001f9765
STV	contracts/validator/src/state.rs	c019812cb940f5ea71784ff3b958749ded5cb3edea0a7a92e080bbe68be7e188

Review Notes

External Dependencies

There are a few depending injection contracts or addresses in the current project:

- `delegator_contract` for the contract `cf-scc`;
- `pools_contract`, `scc_contract`, `airdrops_contract` and `protocol_fee_contract` for the contract `delegator`;
- `validator_contract` and `delegator_contract` for the contract `pools`;
- `delegator_contract`, `sic_contract_address`, `cw20_contract` and `airdrop_contract` for the contract `scc`;
- `airdrop_token_contract` and `cw20_token_contract` for the contract `sic-auto-compound`;
- `pools_contract`, `scc_contract`, `airdrops_contract` and `delegator_contract` for the contract `validator`.

We assume these contracts or addresses are valid and non-vulnerable actors and implement proper logic to collaborate with the current project.

Privilledged Functions

To set up the project correctly, improve overall project quality and preserve upgradability, the following roles are adopted in the codebase.

In the contract `cf-scc`, the role `manager` has the authority over the following function:

- `withdraw_funds()` to withdraw funds to an arbitrary address.

The role `delegator` has the authority over the following function:

- `update_user_rewards()` to update state variable `USER_REWARDS`.

In the contract `delegator`, the role `pools_contract` has the authority over the following functions:

- `deposit()` to update user's deposits;
- `redelegate()` to update user's redelegations;
- `undelegate()` to update user's undelegations;
- `withdraw_funds()` to withdraw funds to the user address and protocol fees to the corresponding contract.

The role `manager` has the authority over the following functions:

- `update_feature_flags()` to enable or disable redelegations;
- `allocate_rewards_and_airdrops()` to update user rewards and airdrops;
- `update_config()` to update contract configurations.

In the contract `Pools`, the role `manager` has the authority over the following functions:

- `update_feature_flags()` to enable or disable redelegations;
- `create_new_redelegation_batch()` to create a new delegation batch;
- `fulfill_redelegation_batch()` to fulfill a redelegation batch;
- `fulfill_pool_rebalancing()` to fulfill multiple redelegations to balance pools;
- `add_pool()` to add a new pool;
- `toggle_pool_active_status()` to toggle pool active status;
- `add_validator_to_pool()` to register a validator to a pool;
- `remove_validator_to_pool()` to remove a validator from a pool;
- `redeem_rewards()` to redeem rewards;
- `swap()` to swap pool funds;
- `undelegate_from_pool()` to undelegate pool funds;
- `reconcil_fund()` to settle withdrawed funds after undelegations;
- `update_airdrop_pointers()` to update airdrop pointers;
- `update_config()` to update configurations.

In the contract `scc`, the role `manager` has the authority over the following functions:

- `update_config()` to update configurations;
- `update_strategy()` to update configurations of a strategy;
- `fetch_undelegated_rewards_from_strategies()` to fetch the rewards of undelegated funds;
- `undelegate_from_strategies()` to undelegate funds from strategies given strategy ids;
- `update_cw20_contracts_registry()` to register cw20 token contracts;
- `claim_airdrops()` to claim airdrops;
- `register_strategy()` to register a new strategy.

The contract `delegator` has the authority over the following function:

- `update_user_rewards()` to update user rewards;
- `update_user_airdrops()` to update user airdrops.

In the contract `sic-auto-compound`, the role `manager` has the authority over the following functions:

- `update_config()` to update contract configurations;

- `remove_validator()` to remove a validator;
- `replace_validator()` to replace a validator;
- `add_validator()` to add a new validator;
- `swap()` to swap funds;
- `reinvest()` to delegate unused funds in the contract.

The role `scc` has the authority over the following functions:

- `claim_airdrops()` to claim airdrops;
- `transfer_rewards()` to transfer rewards and reinvest them;
- `undelegate_rewards()` to process undelegation requests;
- `transfer_undelegated_rewards()` to transfer undelegated rewards to the `scc` contract.

In the contract `Stader-hub`, the role `manager` has the authority over the following function:

- `remove_contract()` to remove a contract given name and address;
- `add_contract()` to add a new contract by name.

In the contract `validator`, the role `pools_contract` has the authority over the following functions:

- `add_validator()` to register a new validator;
- `remove_validator()` to remove validator;
- `stake_to_validator()` to stake funds to a validator;
- `redeem_rewards()` to withdraw delegation rewards;
- `redelegate()` to redelegate funds from a validator to another validator;
- `undelegate()` to undelegate funds from a validator;
- `swap_and_transfer()` to swap funds and transfer to the `scc` contract;
- `transfer_reconciled_funds()` to transfer reconciled funds to the `delegator` contract after undelegation.

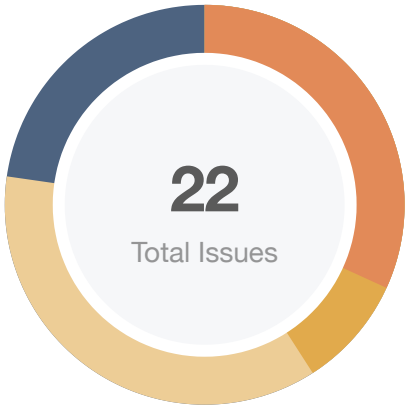
The role `manager` has the authority over the following function:

- `redelegate()` to redelegate funds from a validator to another validator;
- `update_airdrop_registry()` to update the airdrop registry information;
- `redeem_airdrop_and_transfer()` to redeem airdrops and transfer the rewards to the `airdrops` contract;
- `add_slashing_funds()` to add slashing funds;
- `remove_slashing_funds()` to remove slashing funds;
- `update_config()` to update contract configurations.

The Stader Team ensures all the above admin address will imply with the multi-sig solution. At least **3/5 minimum** signing will be required for any of the sensitive operations.

To improve the trustworthiness of the project, dynamic runtime updates in the project should be notified to the community. Any plan to invoke the aforementioned functions should be also considered to move to the execution queue of timelock contract.

Findings



Critical	**0** (0.00%)
Major	**7** (31.82%)
Medium	**2** (9.09%)
Minor	**8** (36.36%)
Informational	**5** (22.73%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
GLOBAL-01	Centralization Risk For Role `manager`	**Centralization / Privilege**	**Major**	Acknowledged
CKP-01	Centralization Risk for Role `delegator`	**Centralization / Privilege**	**Major**	Resolved
CKP-02	Incorrect Event Log	Logical Issue	Minor	Resolved
COA-01	Pseudo-random Redelegation Destination	Logical Issue	Major	Resolved
COA-02	Centralization Risk for role `scc`	**Centralization / Privilege**	**Major**	Acknowledged
COA-03	Potential Ineffective Validator Number Check	Logical Issue	Medium	Resolved
COA-04	Inconsistent Result Types	Logical Issue	Minor	Resolved
COA-05	Potential Integer Overflow	Mathematical Operations	Minor	Resolved
COA-06	Lack of Sanity Check	Volatile Code	Informational	Resolved
COC-01	Lack of Sanity Check	Volatile Code	Informational	Acknowledged
CON-01	Centralization Risk for role `pools_contract`	**Centralization / Privilege**	**Major**	Resolved
CON-02	Lack of Input Validation	Volatile Code	Minor	Resolved

ID	Title	Category	Severity	Status
COR-01	Centralization Risk	**Centralization / Privilege**	**Major**	Acknowledged
COR-02	Inaccurate Query Result	Logical Issue	Minor	Resolved
COR-03	Potential Integer Overflow	Mathematical Operations	Minor	Acknowledged
COR-04	Inconsistent Airdrop Amount Calculation	Logical Issue	Informational	Resolved
COR-05	Lack of Input Validation	Logical Issue	Informational	Resolved
COS-01	Centralization Risk for role `pools_contract`	**Centralization / Privilege**	**Major**	Acknowledged
COS-02	Potential Integer Overflow	Mathematical Operations	Minor	Resolved
COS-03	Logic of Accrued Rewards	Logical Issue	Informational	Resolved
COT-01	Potential Integer Overflow	Mathematical Operations	Minor	Acknowledged
HEL-01	Potential Zero Shares Per Token Ratio	Logical Issue	Medium	Resolved

GLOBAL-01 | Centralization Risk For Role `manager`

Category	Severity	Location	Status
Centralization / Privilege	Major	Global	Acknowledged

Description

In the contract `Validator`, the role `manager` has the authority over the following functions:

- `redelegate()` to redelegate funds from a validator to another one;
- `update_airdrop_registry()` to update the airdrop registry information;
- `redeem_airdrop_and_transfer()` to redeem airdrops and transfer the rewards to the `airdrops` contract;
- `add_slashing_funds()` to add slashing funds;
- `remove_slashing_funds()` to remove slashing funds;
- `update_config()` to update contract configurations.

In the contract `Stader-hub`, the role `manager` has the authority over the following functions:

- `remove_contract()` to remove a contract given name and address;
- `add_contract()` to add a new contract by name.

In the contract `Sic-auto-compound`, the role `manager` has the authority over the following functions:

- `update_config()` to update contract configurations;
- `remove_validator()` to remove a validator;
- `replace_validator()` to replace a validator;
- `add_validator()` to add a new validator;
- `swap()` to swap funds;
- `reinvest()` to delegate unused funds in the contract.

In the contract `Scc`, the role `manager` has the authority over the following functions:

- `update_config()` to update configurations;
- `update_strategy()` to update configurations of a strategy;
- `fetch_undelegated_rewards_from_strategies()` to fetch the rewards of undelegated funds;
- `undelegate_from_strategies()` to undelegate funds from strategies given strategy ids;
- `update_cw20_contracts_registry()` to register cw20 token contracts;
- `claim_airdrops()` to claim airdrops;

- `register_strategy()` to register a new strategy.

In the contract `Pools`, the role `manager` has the authority over the following functions:

- `update_feature_flags()` to enable or disable redelegations;
- `create_new_redelegation_batch()` to create a new delegation batch;
- `fulfill_redelegation_batch()` to fulfill a redelegation batch;
- `fulfill_pool_rebalancing()` to fulfill multiple redelegations to balance pools;
- `add_pool()` to add a new pool;
- `toggle_pool_active_status()` to toggle pool active status;
- `add_validator_to_pool()` to register a validator to a pool;
- `remove_validator_to_pool()`to remove a validator from a pool;
- `redeem_rewards()` to redeem rewards;
- `swap()` to swap pool funds;
- `undelegate_from_pool()` to undelegate pool funds;
- `reconcil_fund()` to settle withdrawed funds after undelegations;
- `update_airdrop_pointers()` to update airdrop pointers;
- `update_config()` to update configurations.

In the contract `Delegator`, the role `manager` has the authority over the following functions:

- `update_feature_flags()`to enable or disable redelegations;
- `allocate_rewards_and_airdrops()` to update user rewards and airdrops;
- `update_config()` to update contract configurations.

In the contract `Cf-scc`, the role `manager` has the authority over the following function:

- `withdraw_funds()` to withdraw funds to an arbitrary address.

Any compromise to the `manager` account may allow the hacker to take advantage of this and manipulate the project.

Recommendation

We advise the client to carefully manage the `manager` account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[**Stader Team**] (11/15/2021): For all the contracts, we have made `Cf-scc` manager keys to be different from Pools, Delegator, Validator, Reward contracts' manager keys.

For the role manager in the contract `Stader-hub`, we don't want to allow any external parties to modify the contract address mappings although there are no user assets staked here.

For the role manager in the contract `Sic-auto-compound`, we don't want functions like `update_config()`, `add_validator()`, `replace_validator()` and `remove_validator()` to be done by external parties.

For the role manager in the contract `Scc`, we have removed permissioned execution from functions `undelegate_from_strategies()`, `fetch_undelegated_rewards_from_strategies()`, `claim_airdrops()`, `update_config()`, `update_strategy_info()`, `register_strategy()` need to be only called by manager.

For the role manager in the contract `Pool`, we are not supporting any of the messages related to redelegation as of now. Funcitons `add_pool()`, `toggle_pool_active_status()`, `add_validator_to_pool()`, `remove_validator_to_pool()`, `update_config()` need to be done by a manager key as of now. In the future, we can make it multi-sig and allow a DAO to vote on these changes. We have made functions `undelegate_from_pool()`, `reconcile_funds()`, `claim_airdrops()`, and `transfer_airdrop()` decentralized. None of the user's base capital is staked here.

[**CertiK**] (11/16/2021): For role manager in the contract `Scc`, the Stader team removed the following permissioned execution from functions in the commit 6dc0b38c9944bcdfd6feafcdcac1420e9e537a56:

- `undelegate_from_strategies()`
- `fetch_undelegated_rewards_from_strategies()`
- `claim_airdrops()`
- `update_config()`
- `update_strategy_info()`
- `register_strategy()`

[**Stader Team**] (11/23/2021): Here are our production contracts

1. pools: https://finder.terra.money/mainnet/address/terra1r2vv8cyt0scyxymktyfuudqs3lgtypk72w6m3m
2. delegator: https://finder.terra.money/mainnet/address/terra1t9ree3ftvgr70fvm6y67zsqxjms8jju8kwcsdu
3. cf-ssc: https://finder.terra.money/mainnet/address/terra1dvpafadh3wdn3dfv6njhhfa6ew99fyqq54k9c4

All of them have a multi-sig admin, with 3/5 minimum signing requirements for any operation. And there are no privileged operations that will lead to the draining of user capital.

[**CertiK**] (11/24/2021):

Pools

The contract `Pools` deployed at terra1r2vv8cyt0scyxymktyfuudqs3lgtypk72w6m3m has the following init message:

```
{
  "min_deposit": "1000",
  "max_deposit": "1000000000000"
}
```

The **manager** of the contract `Pools` is an EOA (Externally Owned Account) terra1uayfvx2zkf23tfz2l6s6q7vvt96jsymyd26phd.

The **admin** of the contract `Pools` has the power to migrate the contract and it is a multi-sig contract terra1alxgc922ylxp0lfk8vs7aqmc504430p9aum36m. The following voters in the multi-sig contract have equal weight:

- terra1mdr46s7nvftlchyruh6rags08qqtt470vceu4v
- terra14xgv060e0v0ww2ljelhy5598vwlmkwv8v4w6lq
- terra1z0wlt2affs5l7ljpzlsnk55w5x384v2te3wdq8
- terra139cugulh83wqq52aj7nt9vt4ss5ltu0lwh0d5g
- terra1dwylleadzwtz08hvesutuvmpxcw8yamtj2a5w8

The admin of the multi-sig contract is an EOA terra1mdr46s7nvftlchyruh6rags08qqtt470vceu4v and has the power to migrate the multi-sig contract.

To pass one proposal, the multi-sig contract requires at least 3 votes.

Delegator

The contract `Delegator` is deployed at terra1t9ree3ftvgr70fvm6y67zsqxjms8jju8kwcsdu with the following init message:

```
{
  "undelegation_max_limit": 20,
  "protocol_fee": "0.01",
  "protocol_fee_contract": "terra16xcytyuaatus8xlgl7fxascvshhn9h8cfq6p08"
}
```

The **manager** of the contract `Delegator` is an EOA terra1uayfvx2zkf23tfz2l6s6q7vvt96jsymyd26phd.

The **admin** of the contract `Delegator` has the power to migrate the contract and it is a multi-sig contract terra1alxgc922ylxp0lfk8vs7aqmc504430p9aum36m. The following voters in the multi-sig contract have equal weight:

- terra1mdr46s7nvftlchyruh6rags08qqtt470vceu4v
- terra14xgv060e0v0ww2ljelhy5598vwlmkwv8v4w6lq
- terra1z0wlt2affs5l7ljpzlsnk55w5x384v2te3wdq8
- terra139cugulh83wqq52aj7nt9vt4ss5ltu0lwh0d5g
- terra1dwylleadzwtz08hvesutuvmpxcw8yamtj2a5w8

The admin of the multi-sig contract is an EOA terra1mdr46s7nvftlchyruh6rags08qqtt470vceu4v and has the power to migrate the multi-sig contract.

To pass one proposal, the multi-sig contract requires at least 3 votes.

Cf-scc

The contract `Cf-scc` is deployed at terra1dvpafadh3wdn3dfv6njhhfa6ew99fyqq54k9c4 with the following init message:

```
{
  "strategy_denom": "uluna",
  "delegator_contract": "terra1t9ree3ftvgr70fvm6y67zsqxjms8jju8kwcsdu"
}
```

The **admin** of the contract `Cf-scc` has the power to migrate the contract and it is a multi-sig contract terra1alxgc922ylxp0lfk8vs7aqmc504430p9aum36m. The following voters in the multi-sig contract have

equal weight:

- terra1mdr46s7nvftlchyruh6rags08qqtt470vceu4v
- terra14xgv060e0v0ww2ljelhy5598vwlmkwv8v4w6lq
- terra1z0wlt2affs5l7ljpzlsnk55w5x384v2te3wdq8
- terra139cugulh83wqq52aj7nt9vt4ss5ltu0lwh0d5g
- terra1dwylleadzwtz08hvesutuvmpxcw8yamtj2a5w8

The admin of the multi-sig contract is an EOA terra1mdr46s7nvftlchyruh6rags08qqtt470vceu4v and has the power to migrate the multi-sig contract.

To pass one proposal, the multi-sig contract requires at least 3 votes.

CKP-01 | Centralization Risk for Role `delegator`

Category	Severity	Location	Status
Centralization / Privilege	**Major**	projects/stader/contracts/cf-scc/src/contract.rs (ff52909): 61	Resolved

Description

In the contract `Cf-scc`, the role `delegator` has the authority over the following function:

- `update_user_rewards()` to update state variable `USER_REWARDS`.

Any compromise to the `delegator` account may allow the hacker to take advantage of this and manipulate the rewards.

Recommendation

We advise the client to carefully manage the `delegator` account's private keys (or the access to `delegator` if it is a contract) to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[**Stader Team**] (11/15/2021): This is intentional. We only want the manager to be able to pull out funds after the community farming as rewards are taken by Stader to exchange for Stader tokens. The function `update_user_rewards()` should only be called by the `Delegator` contract.

We are adding protections to monitor any contract config changes using a cron job. We have a cloud trail of every manager key API/TX usage. These safety guardrails will safeguard compromising manager keys under all circumstances.

- The team ensures the admin account uses a multi-sig solution, with **3/5 minimum** signing requirements for any operations.

[**CertiK**] (11/16/2021): The auditors agree that, if the `delegator` role is correctly set to the `Delegator` contract, the rewards update will follow the logic implemented in the function `allocate_rewards_and_airdrops` of the `Delegator` contract.

[**Stader Team**] (11/23/2021): The `Cf-scc` contract is deployed at terra1dvpafadh3wdn3dfv6njhhfa6ew99fyqq54k9c4. The role `delegator` is the contract `Delegator` with a multi-sig admin and 3/5 minimum signing requirements for any operation. And there are no privileged operations that will lead to the draining of user capital.

[**CertiK**] (11/24/2021): The contract `Cf-scc` deployed at terra1dvpafadh3wdn3dfv6njhhfa6ew99fyqq54k9c4 has the following `initMsg`:

```
{
  "strategy_denom": "uluna",
  "delegator_contract": "terra1t9ree3ftvgr70fvm6y67zsqxjms8jju8kwcsdu"
}
```

The address terra1t9ree3ftvgr70fvm6y67zsqxjms8jju8kwcsdu is the address of the contract `Delegator` provided by the Stader team.

CKP-02 | Incorrect Event Log

Category	Severity	Location	Status
Logical Issue	Minor	projects/stader/contracts/cf-scc/src/contract.rs (ff52909): 73	Resolved

Description

In the function `update_user_rewards()`, an event with the description "zero_user_airdrop_requests" is emitted if `update_user_rewards_requests` is empty. Considering the function is to update user rewards, we believe the description should be "zero_user_rewards_requests".

Recommendation

We recommend changing "zero_user_airdrop_requests" in the aforementioned line to "zero_user_rewards_requests".

Alleviation

The Stader team heeded our advice and resolved this issue in the commit 00f81f56b546b62832f42044ff15c6c0516206d0. The event with description `zero_user_rewards_requests` is emitted.

COA-01 | Pseudo-random Redelegation Destination

Category	Severity	Location	Status
Logical Issue	Major	projects/stader/contracts/sic-auto-compound/src/contract.rs (ff52909): 199	Resolved

Description

In the function `remove_validator`, the delegation to the removed validator should be moved to another validator. The destination validator is decided by a pseudo-random number:

```
199     let validator_to_redelegate = new_validator_pool
200         .get((_env.block.time.seconds() as usize) % (new_validator_pool.len()))
201         .unwrap();
```

Because miners are able to manipulate `_env.block.time.seconds()` to some extent, they can affect the result of the redelegation.

Recommendation

We advise the Stader team to consider if this would be a problem of the design and re-design the aforementioned redelegation process if it is necessary.

Alleviation

The Stader team heeded our advice and resolved this issue in the commit 6e3e8645cc0c6fa999d3781f242ee51d2813817c. Now the manager can choose where he would like to redelegate to.

COA-02 | Centralization Risk for role `scc`

Category	Severity	Location	Status
Centralization / Privilege	**Major**	projects/stader/contracts/sic-auto-compound/src/contract.rs (ff5 2909): 386, 497, 552, 643	Acknowledged

Description

In the contract `Sic-auto-compound`, the role `scc` has the authority over the following functions:

- `claim_airdrops()` to claim airdrops;
- `transfer_rewards()` to transfer rewards and reinvest them;
- `undelegate_rewards()` to process undelegation requests;
- `transfer_undelegated_rewards()` to transfer undelegated rewards to the `scc` contract.

Any compromise to the `scc` account may allow the hacker to take advantage of this and manipulate the project.

Recommendation

We advise the client to carefully manage the `scc` accounts' private key (or access to `scc` if it is a contract) to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[**Stader Team**] (11/15/2021): By design, only Scc should be able to call `transfer_rewards()`, `undelegate_rewards()`, `claim_airdrops()` and `transfer_undelegated_rewards()`.

- The team ensures the admin account uses a multi-sig solution, with **3/5 minimum** signing requirements for any operations.

[**CertiK**] (11/16/2021): The auditors agree that, if the `scc` role is correctly set to the `Scc` contract, the state update will follow the logic implemented in the contract `Scc`.

COA-03 | Potential Ineffective Validator Number Check

Category	Severity	Location	Status
Logical Issue	Medium	projects/stader/contracts/sic-auto-compound/src/contract.rs (ff52909): 150~156	Resolved

Description

When the `manager` calls `remove_validator()` to remove a validator, the number of validators is checked so that the validator pool size will not be too small:

```
150     if state
151         .validator_pool
152         .len()
153         .eq(&(state.min_validator_pool_size as usize))
154     {
155         return Err(ContractError::CannotRemoveMoreValidators {});
156     }
```

However, if the current validator number is already smaller than `min_validator_pool_size`, which might happen if the validator number is never greater than `min_validator_pool_size` upon pool creation, removing a validator will be allowed.

Also, if there is only one validator left, there will not be any other validators to redelegate to, so removing the last validator is impossible.

We would like to check with the Stader team if this is the intended design.

Alleviation

The Stader team resolved the described issue by adding an additional check for `validator_pool` length less than or equal to `min_validator_pool_size` in the commit 76fb62bdee3616819f7c059bbc246e0ff18d0db2.

COA-04 | Inconsistent Result Types

Category	Severity	Location	Status
Logical Issue	Minor	projects/stader/contracts/sic-auto-compound/src/contract.rs (ff52909): 509, 514, 519	Resolved

Description

In the function `transfer_rewards()` of the contract `sic-auto-compound`, the result "Ok()" will be returned if the fund is empty, there are more than two coins sent, or if the sent coin is not supported:

```
508     if info.funds.is_empty() {
509         return Ok(Response::new().add_attribute("no_funds_sent", "1"));
510     }
511
512     // accept only one coin
513     if info.funds.len() > 1 {
514         return Ok(Response::new().add_attribute("multiple_coins_passed", "1"));
515     }
516
517     let transferred_coin = info.funds[0].clone();
518     if transferred_coin.denom.ne(&state.strategy_denom) {
519         return Ok(Response::new().add_attribute("transferred_denom_is_wrong", "1"));
520     }
```

Meanwhile, in the function `transfer_rewards()` of the contract `scc`, similar checks are performed and the result `Err` is returned:

```
909     if info.funds.is_empty() {
910         return Err(ContractError::NoFundsSent {});
911     }
912
913     if info.funds.len() > 1 {
914         return Err(ContractError::MultipleCoinsSent {});
915     }
916
917     let funds = info.funds[0].clone();
918     if state.scc_denom != funds.denom {
919         return Err(ContractError::WrongDenomSent {});
920     }
```

Considering these are unexpected cases, so `Err()` should be returned as the result in `sic-auto-compound`.

Recommendation

We recommend returning `Err()` for the aforementioned cases.

Alleviation

The Stader team heeded our advice and resolved this issue in the commit [649d184d32d80d97929ebe5c7f5119f263c72e4b](). The team implemented a function `get_validated_coin()` in `contracts/sic-auto-compound/src/helpers.rs` to check prerequisites and throw errors when necessary.

COA-05 | Potential Integer Overflow

Category	Severity	Location	Status
Mathematical Operations	Minor	projects/stader/contracts/sic-auto-compound/src/contract.rs (ff52909): 597	Resolved

Description

In the contract `sic-auto-compound`, integer overflow might happen in the listed calculation:

```
597                 amount.u128() * stake_fraction.numerator() / stake_fraction.denominator(),
```

Reference: https://doc.rust-lang.org/stable/book/ch03-02-data-types.html#integer-overflow

Recommendation

We recommend using `checked_mul()` for the aforementioned calculation to avoid integer overflow.

Alleviation

The variable `stake_fraction` is no longer in use.

COA-06 | Lack of Sanity Check

Category	Severity	Location	Status
Volatile Code	Informational	projects/stader/contracts/sic-auto-compound/src/contract.rs (ff52909): 172, 208, 328	Resolved

Description

In the function `remove_validator()` and `replace_validator()`, records for validators need to be removed from `VALIDATORS_TO_STAKED_QUOTA` through the method `remove()`. However, the existence of the validator in `VALIDATORS_TO_STAKED_QUOTA` is not checked.

Recommendation

We recommend checking the existence of the validator before removing it from `VALIDATORS_TO_STAKED_QUOTA`.

Alleviation

The `VALIDATORS_TO_STAKED_QUOTA` mapping is no longer in use.

COC-01 | Lack of Sanity Check

Category	Severity	Location	Status
Volatile Code	Informational	projects/stader/contracts/stader-hub/src/contract.rs (ff52909): 49	Acknowledged

Description

In the function `remove_contract()`, the record for a contract is removed from `CONTRACTS` through the method `remove()`. However, the existence of the contract in `CONTRACTS` is not checked.

Recommendation

We recommend checking the existence of the contract before removing it from `CONTRACTS`.

Alleviation

[**Stader Team**]: The function `remove()` will not throw an error if `CONTRACTS` is not present. It will only remove if the record is present. We would not have to add a special case to check whether the contract is present or not.

[**CertiK**]: Checking the existence of the contract would be helpful for the operator to know the result of the operation: the contract is removed or it does not exist.

CON-01 | Centralization Risk for role `pools_contract`

Category	Severity	Location	Status
Centralization / Privilege	**Major**	projects/stader/contracts/delegator/src/contract.rs (ff52909): 214, 281, 395, 468	Resolved

Description

In the contract **Delegator**, the role `pools_contract` has the authority over the following functions:

- `deposit()` to update user's deposits;
- `redelegate()` to update user's redelegations;
- `undelegate()` to update user's undelegations;
- `withdraw_funds()` to withdraw funds to the user address and protocol fees to the corresponding contract.

In the contract **Validator**, the role `pools_contract` has the authority over the following functions:

- `add_validator()` to register a new validator;
- `remove_validator()` to remove validator;
- `stake_to_validator()` to stake funds to a validator;
- `redeem_rewards()` to withdraw delegation rewards;
- `redelegate()` to redelegate funds from a validator to another one;
- `undelegate()` to undelegate funds from a validator;
- `swap_and_transfer()` to swap funds and transfer to the `scc` contract;
- `transfer_reconciled_funds()` to transfer reconciled funds to the `delegator` contract after undelegation.

Any compromise to the role `pools_contract` may allow the hacker to take advantage of this and manipulate the project.

Recommendation

We advise the client to carefully manage the `pools_contract` account's private key (or the access to `delegator` if it is a contract) to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[**Stader Team**] (11/15/2021): The design was specifically chosen so that no other actors except the `Pools` contract can update the `Delegator` contract. Moreover, once the `Pools` contract is set to a valid address, it can no longer be changed.

[**CertiK**] (11/15/2021): The auditors agree that, if the `pools_contract` role is correctly set to the `Pools` contract, the state update will follow the logic implemented in the contract `Pools`.

[**Stader Team**] (11/23/2021): The `Delegator` contract is deployed at terra1t9ree3ftvgr70fvm6y67zsqxjms8jju8kwcsdu. The role `pools_contract` is the contract `Pools` with a multi-sig admin and 3/5 minimum signing requirements for any operation. And there are no privileged operations that will lead to the draining of user capital.

[**CertiK**] (11/24/2021): The contract `Delegator` deployed at terra1t9ree3ftvgr70fvm6y67zsqxjms8jju8kwcsdu has the following `initMsg`:

```
{
  "undelegation_max_limit": 20,
  "protocol_fee": "0.01",
  "protocol_fee_contract": "terra16xcytyuaatus8xlgl7fxascvshhn9h8cfq6p08"
}
```

The `pools_contract` role is set to terra1r2vv8cyt0scyxymktyfuudqs3lgtypk72w6m3m, which is the address of the contract `Pools` provided by the Stader team.

CON-02 | Lack of Input Validation

Category	Severity	Location	Status
Volatile Code	Minor	projects/stader/contracts/delegator/src/contract.rs (ff52909): 40, 513	Resolved

Description

In the contract `delegator`, the value of `protocol_fee` should be set between 0% to 100%. Otherwise, it might cause unexpected exceptions. For example, the function `withdraw_funds` might fail:

```
530     let protocol_fee_amount = multiply_u128_with_decimal(amount.u128(),
config.protocol_fee);
531     let user_amount = amount.u128() - protocol_fee_amount;
```

Recommendation

We recommend checking the value of `protocol_fee` when updating it.

Alleviation

The Stader team heeded our advice and resolved this issue in the commit 847e2f5363e7fbc303442f745becce913e04642f by not allowing the protocol fee to be greater than 100%.

COR-01 | Centralization Risk

Category	Severity	Location	Status
Centralization / Privilege	**Major**	projects/stader/contracts/scc/src/contract.rs (ff52909): 1021, 1215	Acknowledged

Description

In the contract `Scc`, the contract `delegator` has the authority over the following functions:

- `update_user_rewards()` to update user rewards;
- `update_user_airdrops()` to update user airdrops.

Any compromise to the `delegator` account may allow the hacker to take advantage of this and manipulate the rewards and airdrops.

Recommendation

We advise the client to carefully manage the `delegator` account's private key (or the access to `delegator` if it is a contract) to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[**Stader Team**]: Functions `update_user_rewards()` and `update_user_airdrops()` are to be only called by the delegator contract by design.

- The team ensures the admin account uses a multi-sig solution, with **3/5 minimum** signing requirements for any operations.

[**CertiK**]: The auditors agree that, if the `delegator` role is correctly set to the `Delegator` contract, the rewards and airdrops update will follow the logic implemented in the contract `Delegator`.

COR-02 | Inaccurate Query Result

Category	Severity	Location	Status
Logical Issue	Minor	projects/stader/contracts/scc/src/contract.rs (ff52909): 1431	Resolved

Description

The function `query_undelegation_batch_info()` returns the undelegation batch information given the batch id. However, the batch information might not be accurate because the estimated release time is set to the creation time when an undelegation batch is created:

```
626             if batch_opt.is_none() {
627                 return Ok(BatchUndelegationRecord {
628                     amount: Uint128::zero(),
629                     shares: Decimal::zero(),
630                     unbonding_slashing_ratio: Decimal::one(),
631                     undelegation_s_t_ratio: Decimal::from_ratio(10_u128, 1_u128),
632                     create_time: _env.block.time,
633                     // est_release_time will be filled up in the
undelegate_from_strategies call
634                     est_release_time: _env.block.time,
635                     undelegation_batch_status: UndelegationBatchStatus::Pending,
636                     released: false,
637                 });
638             }
```

And it will not be updated to the correct estimated release time until the function `undelegate_from_strategies()` is triggered:

```
468                 batch.est_release_time = _env
469                     .block
470                     .time
471                     .plus_seconds(strategy_info.unbonding_period +
strategy_info.unbonding_buffer);
```

Recommendation

We recommend setting `est_release_time` to the correct value to make the query result accurate.

Alleviation

The Stader team heeded our advice and resolved this issue in the commit 0645eb4b3beb642830d5c4d3cb83198bb088bcd4 by setting the release time only during the undelegation batch job.

COR-03 | Potential Integer Overflow

Category	Severity	Location	Status
Mathematical Operations	Minor	projects/stader/contracts/scc/src/contract.rs (ff52909): 471	Acknowledged

Description

In the contract `scc`, integer overflow might happen in the listed calculation:

```
471                     .plus_seconds(strategy_info.unbonding_period +
strategy_info.unbonding_buffer);
```

Reference: https://doc.rust-lang.org/stable/book/ch03-02-data-types.html#integer-overflow

Recommendation

We recommend using `checked_add()` for the aforementioned calculation to avoid integer overflow.

Alleviation

[Stader Team]: Unbonding periods and buffers will not be very large numbers. The maximum value for unbonding period will be around 1814400 which would not cause any overflow. Unbonding buffer will be a similar number.

[CertiK]: The integer overflow may not happen if the provided unbonding period and buffer are reasonable. However, the input for the unbonding period and buffer are not checked or restricted, so we would still consider it possible to have integer overflows.

COR-04 | Inconsistent Airdrop Amount Calculation

Category	Severity	Location	Status
Logical Issue	Informational	projects/stader/contracts/scc/src/contract.rs (ff52909): 1367	Resolved

Description

In the function `query_user()`, the value of `user_strategy_query.total_airdrops` is calculated by the sum of the user airdrops:

```
1358          total_airdrops = merge_coin_vector(
1359              &user_airdrops,
1360              CoinVecOp {
1361                  fund: total_airdrops,
1362                  operation: Operation::Add,
1363              },
1364          );
```

The pending airdrops are not considered in the calculation.

However, if we understood the logic correctly, the pending airdrops are also a part of the user's total airdrops. For example, in the function `undelegate_user_rewards()`, `user_airdrops` is converted to `pending_airdrops`:

```
655      user_reward_info.pending_airdrops = merge_coin_vector(
656          &user_reward_info.pending_airdrops,
657          CoinVecOp {
658              fund: user_airdrops,
659              operation: Operation::Add,
660          },
661      );
```

Considering the function `query_user()` is not called within the contract, nor is the message `GetUser` used by another contract, we would like to check with the Stader team if this is the intended design.

Alleviation

[Stader]: The variable `pending_airdrops` is the total airdrops accumulated for the user after updating the user's airdrop pointers. When we query for a user's airdrops, we may be in a situation where the user's pointers are not updated which would imply that the airdrops shown to the user are not all the airdrops

accrued for the user. We simulate the pending airdrops computation in query_user to get all the users accrued airdrops. The action taken is intentional.

COR-05 | Lack of Input Validation

Category	Severity	Location	Status
Logical Issue	Informational	projects/stader/contracts/scc/src/contract.rs (ff52909): 717	Resolved

Description

In the function `claim_airdrops()`, the value of input `amount` can be zero, in which case calling this function will not bring any state update.

Recommendation

We recommend checking if the value of `amount` is zero and continuing processing the request only when it is non-zero.

Alleviation

The Stader team heeded our advice and resolved this issue in the commit 6dc0b38c9944bcdfd6feafcdcac1420e9e537a56 by adding a check for `amount` and throwing an error when necessary.

COS-01 | Centralization Risk for role `pools_contract`

Category	Severity	Location	Status
Centralization / Privilege	**Major**	projects/stader/contracts/validator/src/contract.rs (ff52909): 132, 167, 204, 272, 363, 464, 464, 587, 735	Acknowledged

Description

In the contract `Validator`, the role `pools_contract` has the authority over the following functions:

- `add_validator()` to register a new validator;
- `remove_validator()` to remove validator;
- `stake_to_validator()` to stake funds to a validator;
- `redeem_rewards()` to withdraw delegation rewards;
- `redelegate()` to redelegate funds from a validator to another one;
- `undelegate()` to undelegate funds from a validator;
- `swap_and_transfer()` to swap funds and transfer to the `scc` contract;
- `transfer_reconciled_funds()` to transfer reconciled funds to the `delegator` contract after undelegation.

Any compromise to the `pools_contract` account may allow the hacker to take advantage of this and manipulate the project.

Recommendation

We advise the client to carefully manage the `pools_contract` account's private key (or access to `pools_contract` if it is a contract) to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol to be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., Multisignature wallets.

Indicatively, here is some feasible suggestions that would also mitigate the potential risk at the different level in term of short-term and long-term:

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key;
- Introduction of a DAO/governance/voting module to increase transparency and user involvement.

Alleviation

[**Stader Team**]: The contract `Validator` is only intended to be interacted with from `Pools` by design.

[**CertiK**]: The auditors agree that, if the `pools_contract` role is correctly set to the `Pools` contract, the state update will follow the logic implemented in the contract `Pools`.

COS-02 | Potential Integer Overflow

Category	Severity	Location	Status
Mathematical Operations	Minor	projects/stader/contracts/validator/src/contract.rs (ff52909): 643	Resolved

Description

In the contract `Validator`, integer overflows might happen in the listed calculations:

```
643             total_transfer_amount += coin.amount.u128();
```

```
667             total_transfer_amount += coin_converted;
```

Reference: https://doc.rust-lang.org/stable/book/ch03-02-data-types.html#integer-overflow

Recommendation

We recommend using `checked_add()` for the aforementioned calculations to avoid integer overflow.

Alleviation

The aforementioned code is no longer in use.

COS-03 | Logic of Accrued Rewards

Category	Severity	Location	Status
Logical Issue	Informational	projects/stader/contracts/validator/src/contract.rs (ff52909): 230	Resolved

Description

In the contract `validator`, `accrued_rewards` is recorded in `VMeta` to track the rewards from validators. For example, when the `pools` contract stakes to the `validator` contract, unclaimed rewards of the validator will be added to `VMeta.accrued_rewards` in the function `stake_to_validator()`:

```
226     let full_delegation = deps
227         .querier
228         .query_delegation(&env.contract.address, &val_addr)?;
229
230     let accrued_rewards: Vec<Coin> = if let Some(fd) = full_delegation {
231         fd.accumulated_rewards
232     } else {
233         vec![]
234     };
235
236     VALIDATOR_REGISTRY.save(
237         deps.storage,
238         &val_addr,
239         &VMeta {
240             staked: val_meta.staked.checked_add(stake_amount.amount).unwrap(),
241             accrued_rewards: merge_coin_vector(
242                 &val_meta.accrued_rewards,
243                 CoinVecOp {
244                     fund: accrued_rewards.clone(),
245                     operation: Operation::Add,
246                 },
247             ),
248         },
249     )?;
```

However, the rewards are not claimed within the function, meaning the added rewards are not excluded in the next query and will be added to `VMeta.accrued_rewards` again if the function `stake_to_validator()` is triggered again. The same issue is also in the functions `redelegate()` and `undelegate()`.

We would like to check with the Stader team if the rewards will be automatically claimed when operations like `Delegate`, `Redelegate` and `Undelegate` are done.

Alleviation

[**Stader Team**]: Rewards are auto redeemed upon deposit, undelegation and redelegation. In the new arch, we move the redeemed rewards to a separate reward contract which does not require us to compute the total rewards accrued till now.

COT-01 | Potential Integer Overflow

Category	Severity	Location	Status
Mathematical Operations	Minor	projects/stader/contracts/pools/src/contract.rs (ff52909): 813	Acknowledged

Description

In the contract `pools`, integer overflow might happen in the listed calculation:

```
813                     .plus_seconds(config.unbonding_period +
config.unbonding_buffer),
```

Reference: https://doc.rust-lang.org/stable/book/ch03-02-data-types.html#integer-overflow

Recommendation

We recommend using `checked_add()` for the aforementioned calculation to avoid integer overflow.

Alleviation

[Stader Team]: Unbonding periods and buffers will not be very large numbers. The maximum value for the unbonding period will be around 1814400 which would not cause any overflow. We do not expect the unbonding_buffer to be a very large number.

[CertiK]: The integer overflow may not happen if the provided unbonding period and buffer are reasonable. However, the input for the unbonding period and buffer are not checked or restricted, so we would still consider it possible to have integer overflows.

HEL-01 | Potential Zero Shares Per Token Ratio

Category	Severity	Location	Status
Logical Issue	Medium	projects/stader/contracts/scc/src/helpers.rs (ff52909): 82	Resolved

Description

In the function `get_strategy_shares_per_token_ratio()`, the shares per token ratio is calculated by `total_strategy_shares / total_sic_tokens`. It returns `default_s_t_ratio` when `total_sic_tokens` is zero.

However, it does not handle the case when `total_strategy_shares` is zero, meaning it returns zero if `total_strategy_shares` is zero. This will lead to zero shares for users depositing funds when there are no shares in the strategy yet.

Recommendation

We recommend returning a non-zero shares per token ratio when `total_strategy_shares` is zero.

Alleviation

The Stader team heeded our advice and resolved this issue in the commit 649d184d32d80d97929ebe5c7f5119f263c72e4b by adding checks when `total_strategy_shares` is zero.

[Stader Team]: Ideally the manager would have deposited to a strategy that would not cause shares to go to 0.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Mathematical Operations

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

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.

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.