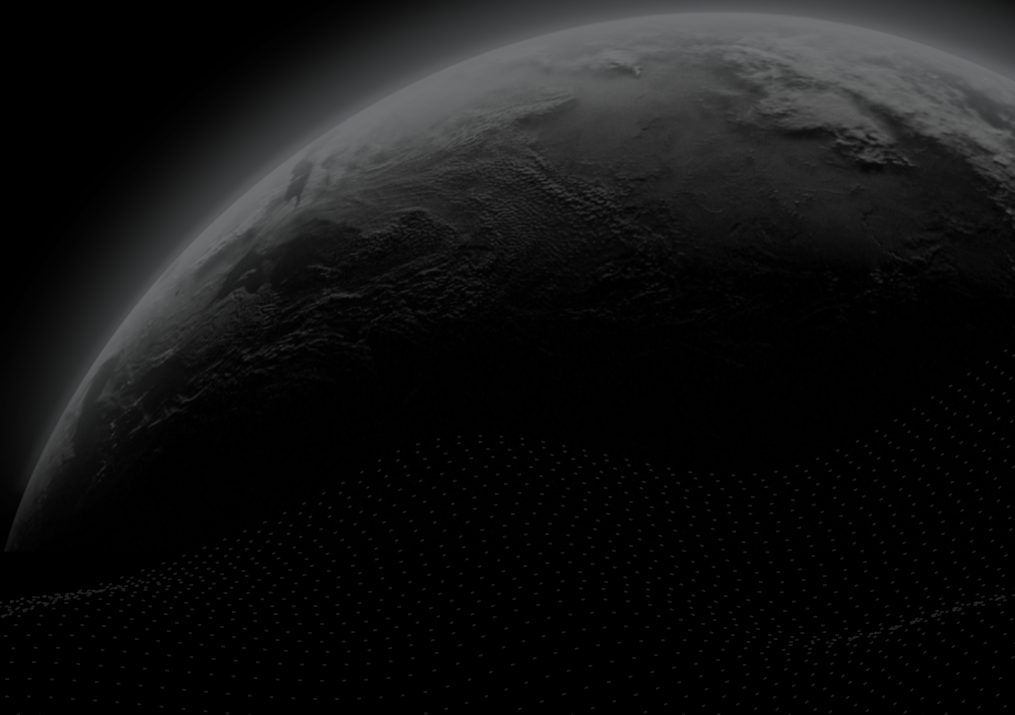




Security Assessment

pyth2wormhole - Governance

CertiK Verified on Dec 13th, 2022





Certik Verified on Dec 13th, 2022

pyth2wormhole - Governance

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

Executive Summary

TYPES

Bridge

ECOSYSTEM

Solana

METHODS

Manual Review, Static Analysis

LANGUAGE

Rust, Solidity

TIMELINE

Delivered on 12/13/2022

KEY COMPONENTS

N/A

CODEBASE

<https://github.com/pyth-network/governance>[...View All](#)

COMMITTS

- 805b9a4ac1c75a81eb5ccdaaf873d0403e49b2bc
- 21f87e5b35d984f9eb335a52cf18445982cff5d6

[...View All](#)

Vulnerability Summary



12

Total Findings

4

Resolved

0

Mitigated

0

Partially Resolved

8

Acknowledged

0

Declined

0

Unresolved

0 Critical

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.

2 Major

1 Resolved, 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.

0 Medium

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

3 Minor

1 Resolved, 2 Acknowledged



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.

7 Informational

2 Resolved, 5 Acknowledged



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 | PYTH2WORMHOLE - GOVERNANCE

I **Summary**

[Executive Summary](#)

[Vulnerability Summary](#)

[Codebase](#)

[Audit Scope](#)

[Approach & Methods](#)

I **Review Notes**

[Understandings](#)

[System Overview](#)

[External Dependencies](#)

I **Findings**

[COT-01 : Initialization of `MaxVoterWeightRecord`](#)

[LIB-01 : Centralization Related Risks](#)

[LIB-02 : Validity Checks on Vesting Schedule](#)

[LIB-03 : Validations for Proposal](#)

[POS-01 : Incorrect Index When Removing a Position](#)

[COT-02 : Missing Error Messages For Account Check](#)

[COT-03 : Missing Mint Check](#)

[LIB-04 : Inappropriate Program ID](#)

[LIB-05 : Missing Emit Events](#)

[LIB-06 : Missing Index Check](#)

[LIB-07 : Inconsistent Error Code](#)

[SRS-01 : Checked Math Not Used](#)

I **Optimizations**

[LIB-08 : Redundant Check](#)

I **Appendix**

I **Disclaimer**

CODEBASE | PYTH2WORMHOLE - GOVERNANCE

Repository













<https://github.com/pyth-network/governance>




Commit

- 805b9a4ac1c75a81eb5ccdaaf873d0403e49b2bc
- 21f87e5b35d984f9eb335a52cf18445982cff5d6

AUDIT SCOPE | PYTH2WORMHOLE - GOVERNANCE

17 files audited ● 4 files with Acknowledged findings ● 13 files without findings

ID	File	SHA256 Checksum
● POS	 src/state/positions.rs	354adea313c07f5c92eafbc3af835e0992417732b66a8f093b738b8afb2d5aab
● VES	 src/state/vesting.rs	97223328ee78eac186e2d2f2d5512dbb31b29c782f7a90cdd0f3201caf98a8b7
● COT	 src/context.rs	8139ba3878baeb68f3a9c12dbb12a7eae876da2b65c887767c297e632783d823
● LIB	 src/lib.rs	e7787d250dcea768b4b9509ab9a16107f0ed8b00a331984594319de2bad8e2b9
● GLO	 src/state/global_config.rs	28d19bd10af14da6a8a2eb35124149d35e033465011b120fb4b05080eafe3322
● MAX	 src/state/max_voter_weight_record.rs	0933113e6be9296117ce210052470c7cc23cb5df806ad9e48aff96b42a911bcf
● MOD	 src/state/mod.rs	0970c10d6d9fbdd5ea06e3a9a755ccb0fed1dc4721a6e904df9a95d4081eaaba
● STC	 src/state/stake_account.rs	df5593917c33749dc5f13ca511f84bab917477d92c17704379df5686193f76e2
● TAR	 src/state/target.rs	8e2209e89c72b297917241e2c5b9a8cdd2c41d2f4c752716363425eb9dceda59
● VOT	 src/state/voter_weight_record.rs	15c91ee1550c7d89768fa01dea091dddb58e1e3c89e6c5f8c1346d9d008dcaad
● CLO	 src/utlis/clock.rs	b663687d7a1ee2bbf57d5d23a965e9cdf21c18a1ca4909f8cc5ee0fb09330757
● MOU	 src/utlis/mod.rs	97ce6888ea74b12abf11ff8a5bfdea7fc9eb41baf81cad468202fff65d959f48
● RIS	 src/utlis/risk.rs	f90e78487e39ecfc763e792a9160c4455cabdb2825cb767da195bd548ba23f3e
● VOE	 src/utlis/voter_weight.rs	3203e7fbcbee68a9bb4abcf039d611382147a11521464e49810c6d945cbf8beb

ID	File	SHA256 Checksum
● CON	 src/constants.rs	01ba4719c80b6fe911b091a7c05124b64eeece964e09c058ef8f9805daca546b
● ERR	 src/error.rs	12fe156c8c1f305f735001bdc097a520711a8227225c376e1792e46f0230f105
● WAS	 src/wasm.rs	a9d01a6ce0100534e3977f80ca13462a59e1df2ea40de66a5a3e9030a25bcd63

APPROACH & METHODS | PYTH2WORMHOLE - GOVERNANCE

This report has been prepared for Wormhole to discover issues and vulnerabilities in the source code of the pyth2wormhole - Governance 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 and Static Analysis 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.

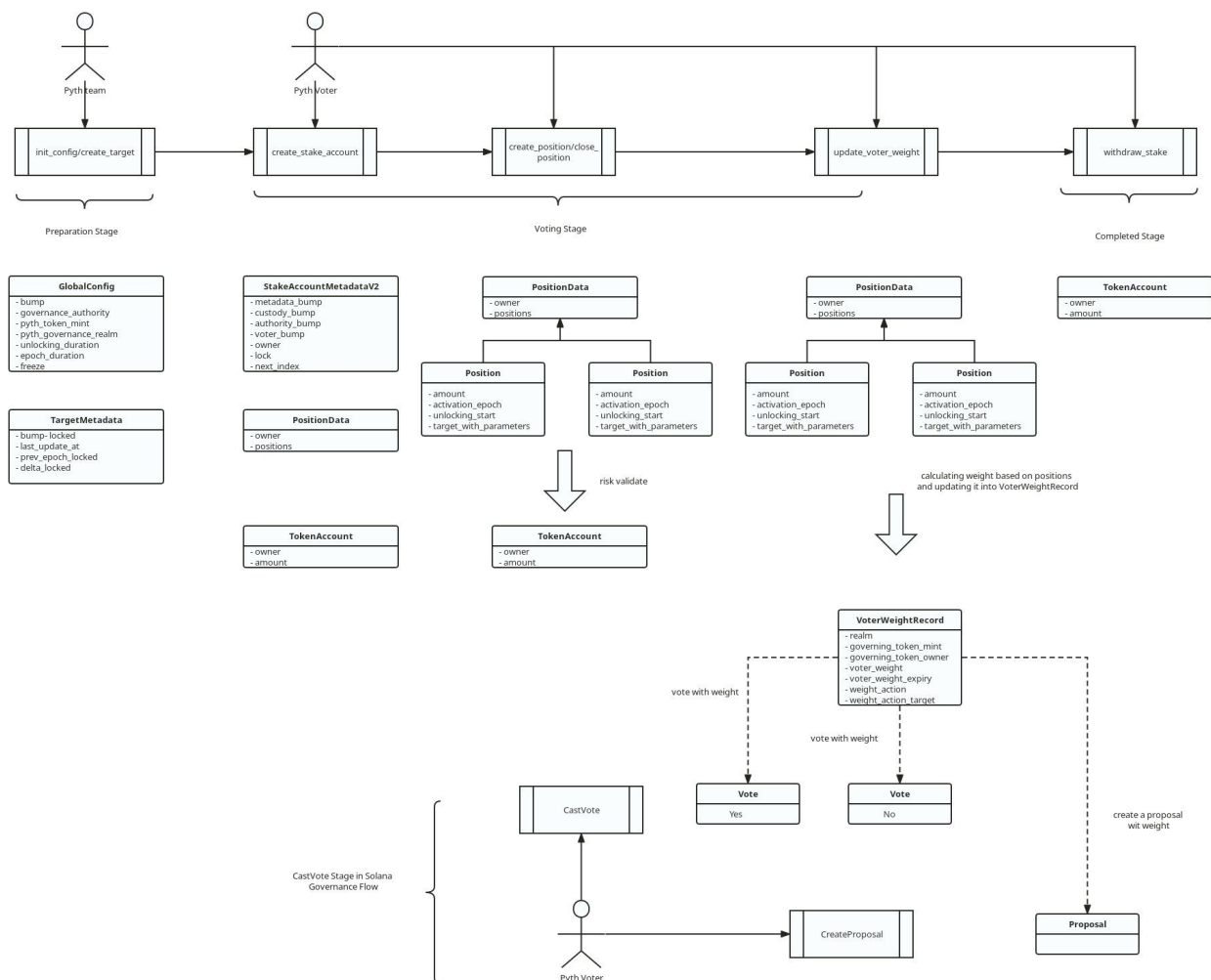
REVIEW NOTES | PYTH2WORMHOLE - GOVERNANCE

Understandings

The Governance program enables governance activities of the Pyth network. Authorities can create and configure governance targets and users can participate by staking their Pyth tokens and obtaining corresponding voting weights. The current implementation (as of June 26th, 2022) supports the voting target only.

System Overview

Workflow

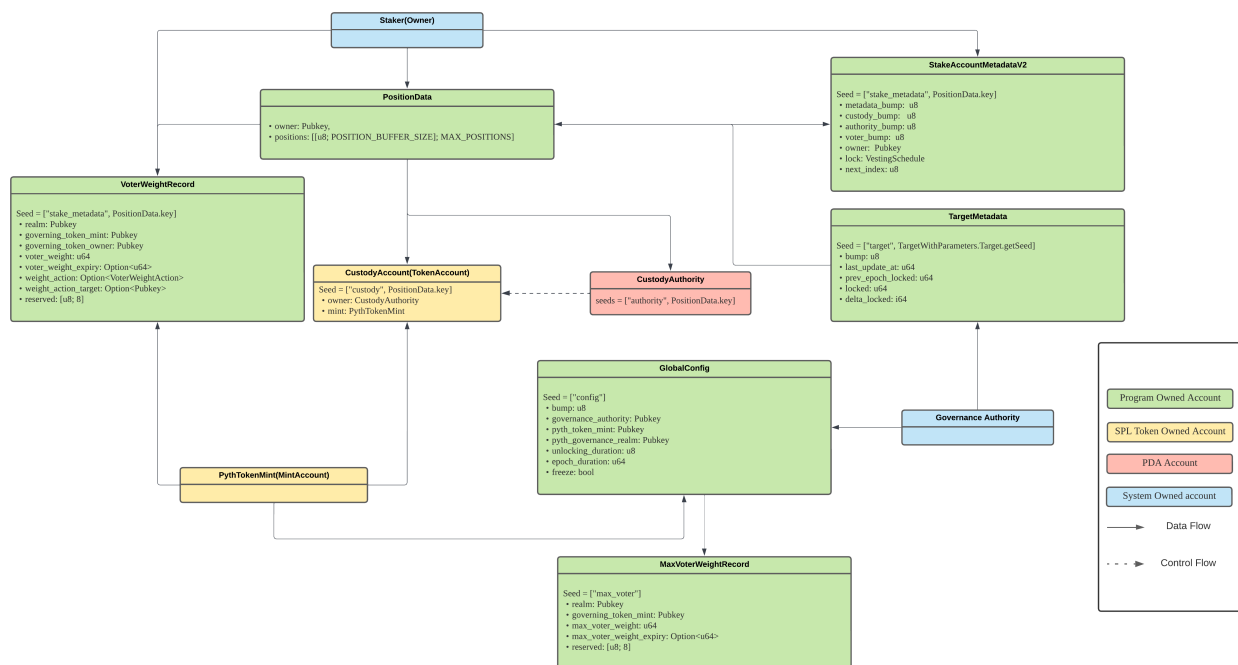


The Pyth Governance program is developed as a plugin of the SPL Governance. Here is a description of the main workflow:

1. The Pyth team should create `GlobalConfig` and `TargetMetadata`.
2. Voters should create a staking account.
3. Voters can call `create_position` instruction. Before doing this, voters should deposit the Pyth tokens to the escrow account(owned by a PDA) to ensure that these positions meet all risk requirements.

- There are five states for the position: `LOCKING` / `LOCKED` / `PREUNLOCKING` / `UNLOCKING` / `UNLOCKED`
- At any time voters want to create a proposal or do a vote, they should call `update_voter_weight` instruction, and only `LOCKED` / `PREUNLOCKING` positions will be used to calculate the weight of the vote. This weight will be written into `VoterWeightRecord` which is a plugin account in Solana Governance Program.
- After updating the weight, voters can create a proposal at the `CreateProposal1` stage or vote at the `CastVote` stage with the `VoterWeightRecord` in Solana Governance Flow.
- After the vote is completed, voters can call `close_position` instruction and the state of these positions will become `UNLOCKING`. After an epoch, the state will become `UNLOCKED`, and voters can call `close_position` instruction again to remove these positions and call `withdraw_stake` instruction to withdraw their Pyth tokens.

Account Relationship



External Dependencies

The project mainly contains the following dependencies:

Dependency	Version
borsh	0.9.3
anchor-lang	0.24.1
anchor-spl	0.24.1
wasm-bindgen	0.2.79

Dependency	Version
spl-governance	2.2.4
js-sys	0.3.56
bincode	1.3.3
solana-program	*

It should also be noted here that the code dependencies are being actively developed in the current auditing version. It is necessary to keep the dependencies up-to-date to avoid potential vulnerabilities.

The on-chain program can be upgradeable after the initial deployment based on Solana's features. Also, based on the unique rent mechanism in Solana, the balance in accounts should be carefully set.

Privileged Functions

The program contains a privileged role `governance_signer` that has the right to configure and update the whole Governance program. Specifically, it has the authority over the following functions:

- `update_governance_authority()` will change the `governance_authority` role of the `config_account`.
- `update_freeze()` will change the `freeze` status of the `config_account`.
- `create_target()` will create a voting target `target_account`.

Additionally, if the program is upgradeable, the upgrade authority account can upgrade the account, thus causing unexpected consequences. The upgrade authority should be carefully managed.

To improve the trustworthiness of the project, dynamic runtime updates in the project should be notified to the community.

FINDINGS | PYTH2WORMHOLE - GOVERNANCE



12

Total Findings

0

Critical

2

Major

0

Medium

3

Minor

7

Informational

This report has been prepared to discover issues and vulnerabilities for pyth2wormhole - Governance. Through this audit, we have uncovered 12 issues ranging from different severity levels. Utilizing the techniques of Manual Review & Static Analysis to complement rigorous manual code reviews, we discovered the following findings:

ID	Title	Category	Severity	Status
COT-01	Initialization Of <code>MaxVoterWeightRecord</code>	Logical Issue	Minor	● Resolved
LIB-01	Centralization Related Risks	Centralization / Privilege	Major	● Acknowledged
LIB-02	Validity Checks On Vesting Schedule	Logical Issue	Minor	● Acknowledged
LIB-03	Validations For Proposal	Logical Issue	Minor	● Acknowledged
POS-01	Incorrect Index When Removing A Position	Logical Issue	Major	● Resolved
COT-02	Missing Error Messages For Account Check	Volatile Code	Informational	● Acknowledged
COT-03	Missing Mint Check	Volatile Code	Informational	● Acknowledged
LIB-04	Inapropriate Program ID	Language Specific	Informational	● Acknowledged
LIB-05	Missing Emit Events	Coding Style	Informational	● Acknowledged
LIB-06	Missing Index Check	Volatile Code	Informational	● Resolved

ID	Title	Category	Severity	Status
<u>LIB-07</u>	Inconsistent Error Code	Language Specific	Informational	● Resolved
<u>SRS-01</u>	Checked Math Not Used	Mathematical Operations	Informational	● Acknowledged

COT-01 | INITIALIZATION OF MaxVoterWeightRecord

Category	Severity	Location	Status
Logical Issue	● Minor	src/context.rs: 243	● Resolved

Description

When context for `UpdateMaxVoterWeight` has the `init_if_needed` condition for the `max_voter_record` account. Since the function `update_max_voter_weight()` fixes all fields of `max_voter_record`, there is no reason to use this function twice.

Recommendation

We recommend changing `init_if_needed` to `init`, unless there exists an old `max_voter_record` account and this is meant to update that account. Furthermore, if this function is meant to update an already exiting `max_voter_record`, it may be best to place access controls on who can call this function.

Alleviation

[Pyth]: The team fixed this issue by using `init` instead of `init_if_needed` in commit [310ad3d84eb0cad8b828aa9ffcaac1e2de68153](#).

LIB-01 | CENTRALIZATION RELATED RISKS

Category	Severity	Location	Status
Centralization / Privilege	● Major	src/lib.rs: 74~87	● Acknowledged

Description

In the program `staking`, the role `governance_authority` has authority over the following functions:

- `update_governance_authority()` will change the `governance_authority` role of the `config_account`.
- `update_freeze()` will change the `freeze` status of the `config_account`.
- `create_target()` will create a voting target `target_account`.

Any compromise to the `governance_authority` account may allow the hacker to take advantage of this and result in unexpected loss.

Additionally, the Solana program could be upgradeable, and the upgrade authority is the deployer by default. Therefore, if the program is upgradable, and the upgrade authority account is compromised, it could lead to a malicious program upgrade, thus introducing centralization risk.

Recommendation

These centralization-related risks described in the current project potentially need multiple iterations to improve in the security operation and level of decentralization, and in most cases can't be resolved entirely at the present stage. We advise the client to carefully manage the aforementioned privileged accounts' keypair to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

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

Short Term:

Timelock and Multi sign ($\frac{2}{3}$, $\frac{3}{5}$) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;
AND

- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles;
OR
- Remove the risky functionality.

Noted: Recommend considering the long-term solution or the permanent solution. The project team shall make a decision based on the current state of their project, timeline, and project resources.

I Alleviation

[Pyth]: The team will fix the issue in the future, which will not be included in this audit engagement. The team are planning to onboard most of these admin features to the governance.

LIB-02 | VALIDITY CHECKS ON VESTING SCHEDULE

Category	Severity	Location	Status
Logical Issue	● Minor	src/lib.rs: 96	● Acknowledged

Description

When creating a stake account, the user provides the vesting schedule. If `PeriodVesting` is chosen, four parameters need to be provided:

- `initial_balance`
- `start_date`
- `period_duration`
- `num_periods`

Since there are operations that involve dividing by `period_duration` or `num_periods`, there should be checks to ensure that these parameters are positive, especially since they cannot be changed after the stake account is created.

Recommendation

Recommend adding checks to ensure `period_duration` and `num_periods` are positive.

Alleviation

[Pyth]: The team acknowledged this issue and decided not to change the current codebase.

LIB-03 | VALIDATIONS FOR PROPOSAL

Category	Severity	Location	Status
Logical Issue	● Minor	src/lib.rs: 393	● Acknowledged

Description

When updating a voter's weight with the `castvote` action, a proposal account is deserialized, but not sufficiently checked.

For example, since the `governance` and `governing_token_mint` fields are not checked, it is possible for the proposal account to belong to different governance. Furthermore, the `state` field is not checked, meaning the proposal may have already succeeded or failed.

Recommendation

Recommend implementing checks to ensure the proposal is valid.

Alleviation

[Pyth]: The team acknowledged this issue and decided not to change the current codebase.

POS-01 | INCORRECT INDEX WHEN REMOVING A POSITION

Category	Severity	Location	Status
Logical Issue	● Major	src/state/positions.rs: 54	● Resolved

Description

The function `make_none()` is used to remove a position, which is needed when completely closing a position. By design, it will swap the last valid position with the position to be removed.

```
51     pub fn make_none(&mut self, i: usize, next_index: &mut u8) -> Result<()> {
52         *next_index -= 1;
53         self.positions[i] = self.positions[*next_index as usize];
54         None::<Option<Position>>.try_write(&mut self.positions[i])
55     }
```

The function first decreases `next_index` by 1, so it now points to the last created position. Then the function overwrites the position at index `i` with the information of the last position, and then changes position `i` to a `None` value. This eliminates position `i`, but the value of `next_index` does not point towards an empty position.

A consequence of this is that if a new position was created, then this new position will overwrite the last created position due to the updated value of `next_index`, causing the user to completely lose a position.

Recommendation

Recommend writing the `None` value to `positions[next_index]` instead of `positions[i]`.

Alleviation

[Pyth]: The team heeded the advice and resolved this issue in commit [0c70e06edbd6420935323e1c1416531905c4ff38](#) by writing `None` value to `positions[next_index]` instead.

COT-02 | MISSING ERROR MESSAGES FOR ACCOUNT CHECK

Category	Severity	Location	Status
Volatile Code	● Informational	src/context.rs: 58, 67, 106, 118, 160, 187, 213, 256	● Acknowledged

Description

The `#[account(address = <expr>)]` in Anchor supports throwing an exception if the address does not match. It is better to use `#[account(address = <expr> @ custom_error)]` instead of `#[account(address = <expr>)]` to provide a customized error message and pass it back to the caller.

Reference: [Accounts in anchor_lang](#)

Recommendation

Recommend using custom errors via `@` for each `#[account(address = expr)]` for better error handling.

Alleviation

[Pyth]: The team acknowledged the finding and will fix the finding in the future, which will not be included in this audit engagement.

COT-03 | MISSING MINT CHECK

Category	Severity	Location	Status
Volatile Code	● Informational	src/context.rs: 122	● Acknowledged

Description

When withdrawing the staked token, a `destination` TokenAccount is given but missing the mint check against `config.pyth_token_mint`.

Recommendation

Recommend adding the mint check for the `destination` account.

Alleviation

[Pyth]: The team acknowledged this issue and decided not to change the current codebase.

LIB-04 | INAPPROPRIATE PROGRAM ID

Category	Severity	Location	Status
Language Specific	● Informational	src/lib.rs: 43	● Acknowledged

Description

In L43, the program is using a default program ID provided by the Anchor framework:

```
43 declare_id!("Fg6PaFpoGXkYsidMpWTK6W2BeZ7FEfcYkg476zPFsLnS");
```

Recommendation

Recommend replacing this ID during the actual deployment to avoid potential vulnerability. The address can be generated using Solana CLI, for example, `solana address -k target/deploy/[]-keypair.json`

Alleviation

[Pyth]: The team acknowledged the finding and will fix the finding in the future, which will not be included in this audit engagement.

LIB-05 | MISSING EMIT EVENTS

Category	Severity	Location	Status
Coding Style	● Informational	src/lib.rs: 49	● Acknowledged

Description

In the program `staking`, the functions that affect the status of sensitive variables should be able to emit events.

- `init_config` affects the `GlobalConfig` account
- `update_governance_authority` affects `governance_authority` of the `GlobalConfig` account
- `update_freeze` affects `freeze` of the `GlobalConfig` account
- `create_stake_account` affects `StakeAccountMetadataV2` account and `VoterWeightRecord` account
- `create_position` affects `PositionData` account
- `close_position` affects `PositionData` account
- `update_voter_weight` affects `VoterWeightRecord` account

Recommendation

Recommend adding events for sensitive actions and emitting them in the functions.

Reference: [Anchor emit macro](#)

Alleviation

[Pyth]: The team acknowledged the finding and will fix the finding in the future, which will not be included in this audit engagement.

LIB-06 | MISSING INDEX CHECK

Category	Severity	Location	Status
Volatile Code	● Informational	src/lib.rs: 196	● Resolved

Description

The position index `i`, converted from user input `index`, is not sanitized before accessing the positions in `stake_account_positions`.

Recommendation

Recommend checking if the index is in the valid array range, i.e., `[0, next_index)`.

Alleviation

[Pyth]: The team fixed this issue by throwing error when out of bound issue raised in commit [0c70e06edbd6420935323e1c1416531905c4ff38](#).

LIB-07 | INCONSISTENT ERROR CODE

Category	Severity	Location	Status
Language Specific	● Informational	src/lib.rs: 407, 427	● Resolved

Description

In the function `update_woter_weight`, both actions `CastVote` and `CreateProposal` will compare the proposal `max_voting_time` with `config.epoch_duration`. It currently checks to ensure the `max_voting_time` is at least larger than one epoch.

```
406         if config.epoch_duration > max_voting_time.into() {  
407             return Err(error!(ErrorCode::ProposalTooLong));  
408         }
```

However, the error code for the `max_voting_time` when it is less than one epoch will be `ErrorCode::ProposalTooLong`. If the max voting time is meant to be larger than one epoch, the error code should be "proposal too short" instead. Otherwise, if the max voting time is meant to be shorter than an epoch, the condition should be reversed.

Recommendation

Recommending to use a more appropriate error message if the voting time for proposals are meant to be larger than one epoch, otherwise reverse the condition to `config.epoch_duration <= max_voting_time.into()`.

Alleviation

[Pyth]: The team fixed this issue by reversing the inequality in commit [21f87e5b35d984f9eb335a52cf18445982cff5d6](#).

SRS-01 | CHECKED MATH NOT USED

Category	Severity	Location	Status
Mathematical Operations	● Informational	src/lib.rs: 148, 214, 445; src/state/positions.rs: 167; src/state/vesting.rs: 84	● Acknowledged

I Description

The above linked locations are places where checked math is not used and integer overflow is possible.

I Recommendation

Recommend using checked math for these operations.

I Alleviation

[Pyth]: The team acknowledged this issue and decided not to change the current codebase.

OPTIMIZATIONS | PYTH2WORMHOLE - GOVERNANCE

ID	Title	Category	Severity	Status
LIB-08	Redundant Check	Logical Issue	Optimization	● Acknowledged

LIB-08 | REDUNDANT CHECK

Category	Severity	Location	Status
Logical Issue	● Optimization	src/lib.rs: 317~355	● Acknowledged

I Description

The following two risk validations at L322 to L332 and L345 to L355 in the `withdraw_stake` instruction are redundant.

```
318         let remaining_balance = stake_account_custody
319             .amount
320             .checked_sub(amount)
321             .ok_or_else(|| error!(
322 (ErrorCode::InsufficientWithdrawableBalance)))?;
323         if utils::risk::validate(
324             stake_account_positions,
325             remaining_balance,
326             unvested_balance,
327             current_epoch,
328             config.unlocking_duration,
329         )
330         .is_err()
331         {
332             return Err(error!(ErrorCode::InsufficientWithdrawableBalance));
333         }
334         transfer(
335             CpiContext::from(&*ctx.accounts).with_signer(&[&[
336                 AUTHORITY_SEED.as_bytes(),
337                 ctx.accounts.stake_account_positions.key().as_ref(),
338                 &[stake_account_metadata.authority_bump],
339             ]]),
340             amount,
341         )?;
342
343         ctx.accounts.stake_account_custody.reload()?;
344
345         if utils::risk::validate(
346             stake_account_positions,
347             ctx.accounts.stake_account_custody.amount,
348             unvested_balance,
349             current_epoch,
350             config.unlocking_duration,
351         )
352         .is_err()
353         {
354             return Err(error!(ErrorCode::InsufficientWithdrawableBalance));
355         }

```

Recommendation

It can be refactored to a single risk validation and a single value comparison as shown below:

```
let remaining_balance = stake_account_custody
    .amount
    .checked_sub(amount)
    .ok_or_else(|| error!(ErrorCode::InsufficientWithdrawableBalance));

transfer(
    CpiContext::from(&*ctx.accounts).with_signer(&[
        AUTHORITY_SEED.as_bytes(),
        ctx.accounts.stake_account_positions.key().as_ref(),
        &[stake_account_metadata.authority_bump],
    ]),
    amount,
)?;

ctx.accounts.stake_account_custody.reload()?;

assert_eq!(ctx.accounts.stake_account_custody.amount, remaining_balance);

if utils::risk::validate(
    stake_account_positions,
    ctx.accounts.stake_account_custody.amount,
    unvested_balance,
    current_epoch,
    config.unlocking_duration,
)
.is_err()
{
    return Err(error!(ErrorCode::InsufficientWithdrawableBalance));
}
```

Alleviation

[Pyth]: The team acknowledged this issue and decided not to change the current codebase.

APPENDIX | PYTH2WORMHOLE - GOVERNANCE

Finding Categories

Categories	Description
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.
Language Specific	Language Specific findings are issues that would only arise within Solidity, i.e. incorrect usage of private or delete.
Coding Style	Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

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.

