SHERLOCK

Security Review For Wormhole

Private Audit Contest Prepared For:	Wormhole
Lead Security Expert:	Obsidian
Date Audited:	February 25 - March 18, 2025
Final Commit:	091d70b

Introduction

MultiGov: The industry's first multichain governance system.

Scope

Repository: wormhole-foundation/multigov

Audited Commit: ff2c8752eaf785b32f9c6802a03b88e744d5563c

Final Commit: 091d70b7f7dce47d225b3ea63c6a1ea041def430

Files:

- solana/programs/staking/Cargo.toml
- solana/programs/staking/Xargo.toml
- solana/programs/staking/src/context.rs
- solana/programs/staking/src/contexts/cancel_vesting.rs
- solana/programs/staking/src/contexts/claim_vesting.rs
- solana/programs/staking/src/contexts/create_vesting.rs
- solana/programs/staking/src/contexts/create_vesting_balance.rs
- solana/programs/staking/src/contexts/finalize.rs
- solana/programs/staking/src/contexts/initialize.rs
- solana/programs/staking/src/contexts/mod.rs
- solana/programs/staking/src/contexts/transfer_vesting.rs
- solana/programs/staking/src/contexts/withdraw_surplus.rs
- solana/programs/staking/src/error.rs
- solana/programs/staking/src/lib.rs
- solana/programs/staking/src/state/checkpoints.rs
- solana/programs/staking/src/state/ext/mod.rs
- solana/programs/staking/src/state/global_config.rs
- solana/programs/staking/src/state/guardian_signatures.rs
- solana/programs/staking/src/state/mod.rs
- solana/programs/staking/src/state/proposal.rs
- solana/programs/staking/src/state/proposal_voters_weight_cast.rs
- solana/programs/staking/src/state/spoke_airlock.rs
- solana/programs/staking/src/state/spoke_message_executor.rs

- solana/programs/staking/src/state/spoke_metadata_collector.rs
- solana/programs/staking/src/state/stake_account.rs
- solana/programs/staking/src/state/vesting.rs
- solana/programs/staking/src/state/vesting_balance.rs
- solana/programs/staking/src/state/vesting_config.rs
- solana/programs/staking/src/state/vote_weight_window_lengths.rs
- solana/programs/staking/src/utils/clock.rs
- solana/programs/staking/src/utils/execute_message.rs
- solana/programs/staking/src/utils/mod.rs
- solana/programs/staking/src/wasm.rs

Final Commit Hash

091d70b7f7dce47d225b3ea63c6a1ea041def430

Findings

Each issue has an assigned severity:

- Medium issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- High issues are directly exploitable security vulnerabilities that need to be fixed.

Issues Found

High	Medium
0	1

Issues Not Fixed and Not Acknowledged

High	Medium
0	0

Security experts who found valid issues

EgisSecurity	Kose	Obsidian

Issue M-1: Users Can Prevent Vesting Cancellation

Source: https://github.com/sherlock-audit/2025-02-stealth-judging/issues/59

Found by

EgisSecurity, Kose, Obsidian

Summary

`cancel_vesting` can be prevented by vesters via transferring token account ownership to any other address.

Root Cause

In cancel_vesting.rs associated token authorithy is checked to be sure it is `vester_ta.owner` while deriving `vester_ta`, this owner variable is also used as a seed to confirm `vesting_balance` matches with the address provided:

```
    #[account(
        associated_token::mint = mint,
        associated_token::authority = vester_ta.owner,
        associated_token::token_program = token_program
    )]
    vester_ta: InterfaceAccount<'info, TokenAccount>,

...

    #[account(
        mut,
        seeds = [VESTING_BALANCE_SEED.as_bytes(), config.key().as_ref(),
 ↪  vester_ta.owner.key().as_ref()],
        bump = vesting_balance.bump
    )]
    vesting_balance: Account<'info, VestingBalance>,
```

However, while the first check for `vester_ta` is pointless, considering it will pass even if the owner changed, the check in `vesting_balance` will fail because derived address from new owner won't match the `vesting_balance` expected.

Same vulnerability with smaller impact exists in `close_vesting_balance` as well, hence fix has to be provided for both functions.

Internal Pre-conditions

None, every `cancel_vesting` call is vulnerable without any internal pre-condition.

External Pre-conditions

None, every `cancel_vesting` call is vulnerable without any external pre-condition.

Attack Path

1. Vester changes the authority of their SPL token to some other address. This can be done for multiple reasons: a. User can expect their vest to be cancelled via seeing off-chain actions. b. User can see the `cancel_vesting` call for their account will be executed (they can be validator/rpc provider etc.) c. And most importantly, anyone can transfer their token account's ownership to their side account just to be sure their vest won't be cancelled up until the vesting is finalized. Then they can transfer back to be able to claim.

Impact

It can have two different impacts related to if admin provided the tokens already or not.

1. If admin provided the tokens to vault already, then there won't be any way to receive back those tokens and attacker can claim their tokens when their maturation time arrives via transferring ownership back to their main vesting address.
2. If admin didn't provided the tokens yet, then admin has to cancel all other vests and close all other vesting balance accounts in order to get back the rent and then create a new vesting config with all account creations and vesting creations once more. This would lead to hundreds if not thousands of unnecessary transactions in order to prevent malicious user from receiving tokens that they shouldn't receive. The costs for these actions could easily exceed the attacker's vesting balance.

Both scenarios result in financial losses: either direct loss as attackers receive funds they shouldn't, or implicit loss as the entire vesting process must be reset at vesting_admin's expense.

PoC

No response

Mitigation

Do not use `vester_ta.owner` in any constraint and seeds for accounts since SPL token account ownerships can be transferred. This is true for both `cancel_vesting` and `close_v` `esting_balance` functions.

Discussion

sherlock-admin2

The protocol team fixed this issue in the following PRs/commits:
https://github.com/wormhole-foundation/multigov/pull/265

Disclaimers

Sherlock does not provide guarantees nor warranties relating to the security of the project.

Usage of all smart contract software is at the respective users’ sole risk and is the users’ responsibility.