Report

Ackee Solana Bootcamp

Auditor: Felipe Donato

Initial setup

Fuzzing: I've started by going through the code base and selecting every potential critical function to be tested:

- Initialize
- Withdraw
- Deposit
- SwapBaseInput
- SwapBaseOutput
- CollectFundFee

After selecting a first function or functions, I bring the target program function under close inspection to gather the collection of accounts and arguments that are expected by the given function.

With the list of expected accounts in hand, I go through each anchor constraint that will likely tell in what state the given account is expected to be passed in to function. Then I start populating the already initialized **trident-tests/fuzz-instructions.rs get_accounts()** function from the given Targets function I am testing at moment, what follows now is quick break down how I approached Accounts configuration in my test:

Following the bootcamp methodology I have abused of methods within the framework, as

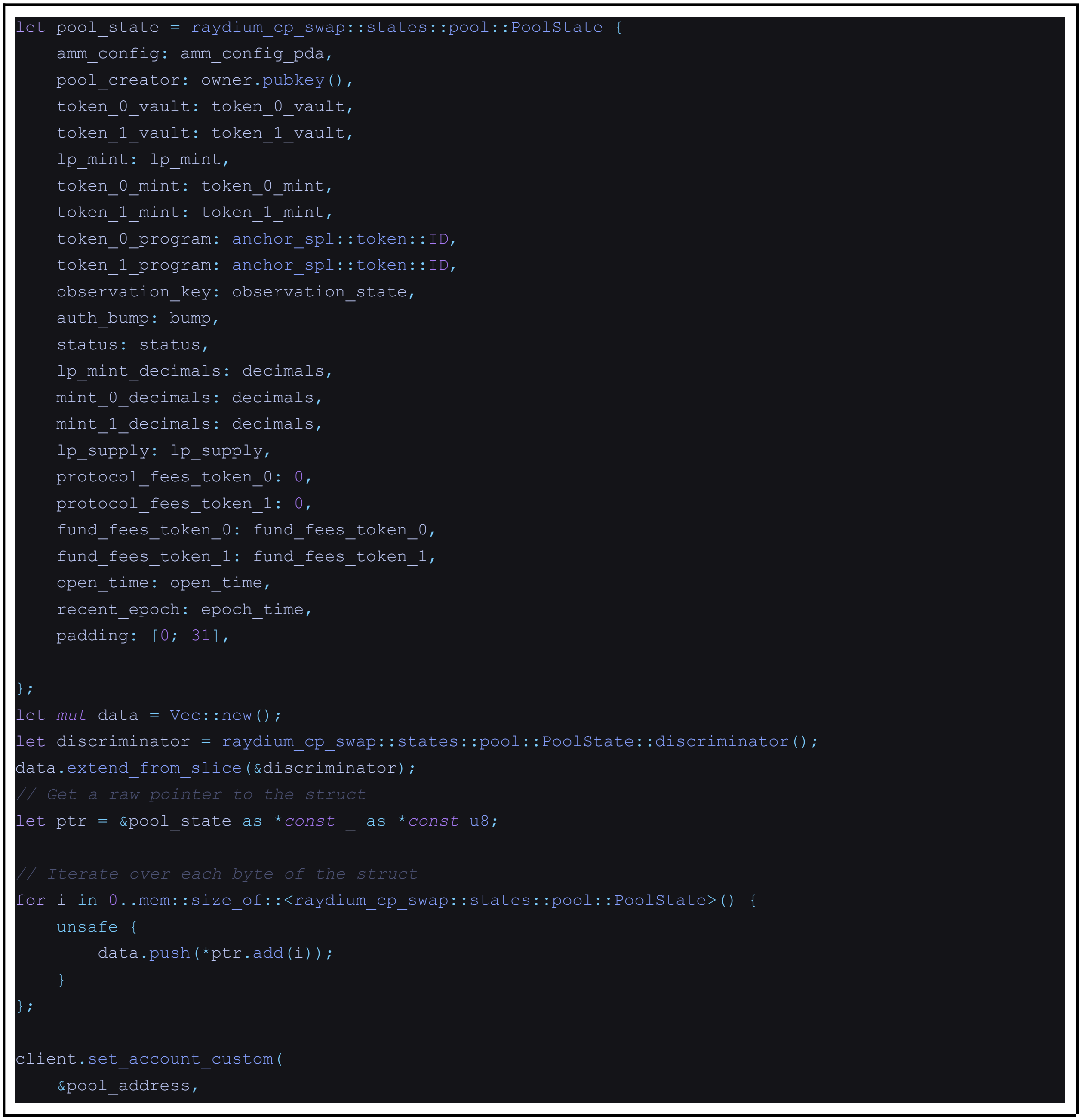
- *get_or_create_account()*, which I have used in this case mostly not exclusively for Key Pairs.
- *get_account()* for when there was a possibility of the account having already initialized.
- set_account_custom() mostly but not exclusively used for PDA account creation with custom data, although in a given instance facing a Mint supply check I have used this method to create a Mint that would pass the given check a specific test.
- Lastly, well known program Addresses.

Initialize function Context Accounts and its given constraints can be found at: *raydium-cp-swap/programs/cp-swap/instructions/initialize.rs*

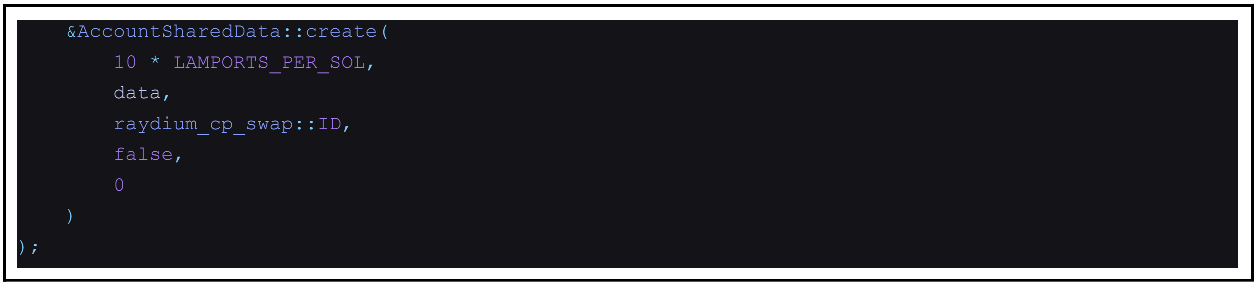
From here I shall take a brief tour in the little that has been made to adapt tests for very specific situations at times in which a potential bug/vulnerabilities could have been tested, instead of going through the specifics of configuration in every test, as I believe I have mostly summarized it above.

Creating standalone PDA's state Accounts for specific tests:
Using the very powerful *set_account_custom()* method, I could meticulously craft an account to test edge cases, as example.

PoolState

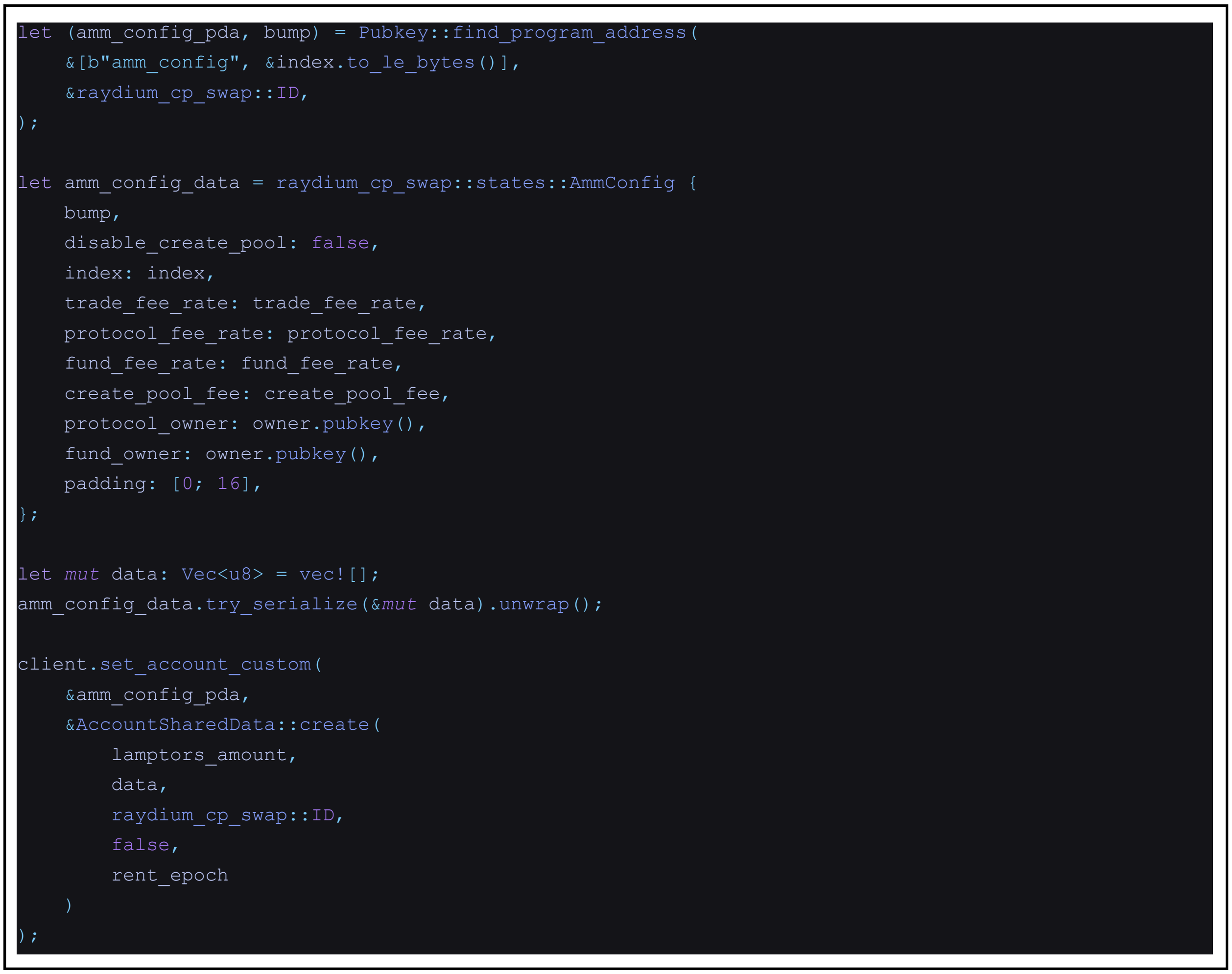


```
let pool_state = raydium_cp_swap::states::pool::PoolState {
    amm_config: amm_config_pda,
    pool_creator: owner.pubkey(),
    token_0_vault: token_0_vault,
    token_1_vault: token_1_vault,
    lp_mint: lp_mint,
    token_0_mint: token_0_mint,
    token_1_mint: token_1_mint,
    token_0_program: anchor_spl::token::ID,
    token_1_program: anchor_spl::token::ID,
    observation_key: observation_state,
    auth_bump: bump,
    status: status,
    lp_mint_decimals: decimals,
    mint_0_decimals: decimals,
    mint_1_decimals: decimals,
    lp_supply: lp_supply,
    protocol_fees_token_0: 0,
    protocol_fees_token_1: 0,
    fund_fees_token_0: fund_fees_token_0,
    fund_fees_token_1: fund_fees_token_1,
    open_time: open_time,
    recent_epoch: epoch_time,
    padding: [0; 31],

};
let mut data = Vec::new();
let discriminator = raydium_cp_swap::states::pool::PoolState::discriminator();
data.extend_from_slice(&discriminator);
// Get a raw pointer to the struct
let ptr = &pool_state as *const _ as *const u8;

// Iterate over each byte of the struct
for i in 0..mem::size_of::<raydium_cp_swap::states::pool::PoolState>() {
    unsafe {
        data.push(*ptr.add(i));
    }
};

client.set_account_custom(
    &pool_address,
```



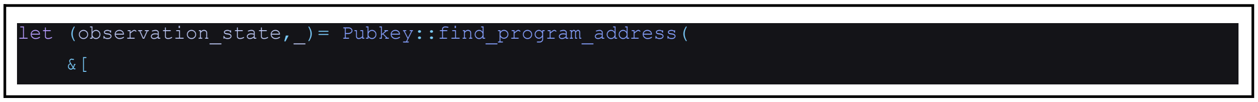
```
    &AccountSharedData::create(
        10 * LAMPORTS_PER_SOL,
        data,
        raydium_cp_swap::ID,
        false,
        0
    )
);
```

AmmConfig

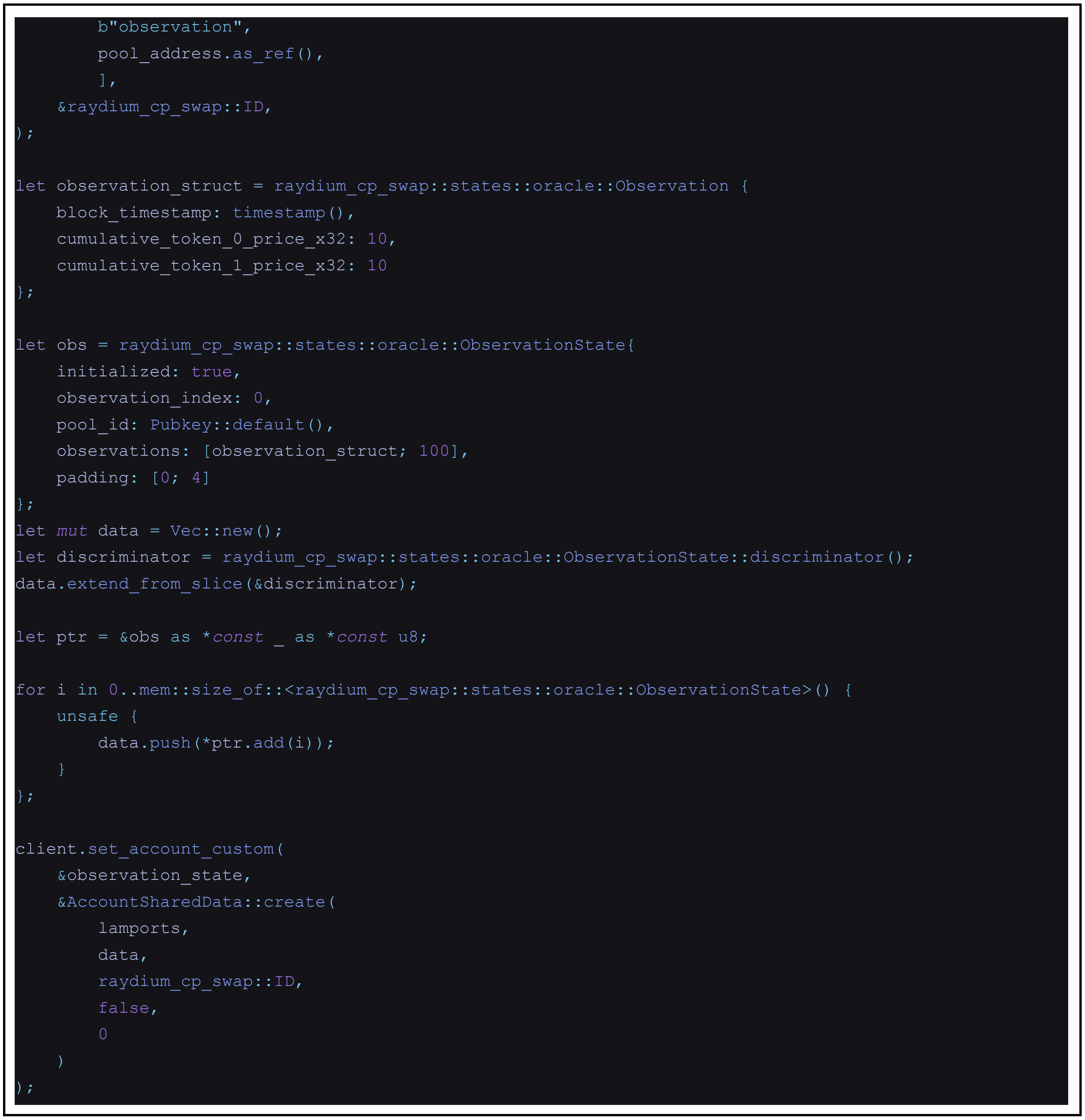


```
let (amm_config_pda, bump) = Pubkey::find_program_address(
    &[b"amm_config", &index.to_le_bytes()],
    &raydium_cp_swap::ID,
);

let amm_config_data = raydium_cp_swap::states::AmmConfig {
    bump,
    disable_create_pool: false,
    index: index,
    trade_fee_rate: trade_fee_rate,
    protocol_fee_rate: protocol_fee_rate,
    fund_fee_rate: fund_fee_rate,
    create_pool_fee: create_pool_fee,
    protocol_owner: owner.pubkey(),
    fund_owner: owner.pubkey(),
    padding: [0; 16],
};

let mut data: Vec<u8> = vec![];
amm_config_data.try_serialize(&mut data).unwrap();

client.set_account_custom(
    &amm_config_pda,
    &AccountSharedData::create(
        lamptors_amount,
        data,
        raydium_cp_swap::ID,
        false,
        rent_epoch
    )
);
```

Observation



```
let (observation_state,_)= Pubkey::find_program_address(
    &[
```



```
        b"observation",
        pool_address.as_ref(),
        ],
    &raydium_cp_swap::ID,
);

let observation_struct = raydium_cp_swap::states::oracle::Observation {
    block_timestamp: timestamp(),
    cumulative_token_0_price_x32: 10,
    cumulative_token_1_price_x32: 10
};

let obs = raydium_cp_swap::states::oracle::ObservationState{
    initialized: true,
    observation_index: 0,
    pool_id: Pubkey::default(),
    observations: [observation_struct; 100],
    padding: [0; 4]
};
let mut data = Vec::new();
let discriminator = raydium_cp_swap::states::oracle::ObservationState::discriminator();
data.extend_from_slice(&discriminator);

let ptr = &obs as *const _ as *const u8;

for i in 0..mem::size_of::<raydium_cp_swap::states::oracle::ObservationState>() {
    unsafe {
        data.push(*ptr.add(i));
    }
};

client.set_account_custom(
    &observation_state,
    &AccountSharedData::create(
        lamports,
        data,
        raydium_cp_swap::ID,
        false,
        0
    )
);
```

After fixing accounts to meet every potential test needs, I move onto *get_data()* where much of the fuzzing was conducted as well.

There is not much to explain here, each instructions has its Data struct which is allow for easy fuzzing setup, using: self.data.arg we have moved a placeholder into place that will be used by the fuzzer to iterate over the specific data type and effect permutation in value.

In *FuzzAccounts* I’ve picked which account I would have the need to store or fuzz, by setting them up correctly I could make use of the very convenient methods such as: get_or_create_account() and etc..

Testing

Having thoroughly fuzzed(guided) instructions both in their both happy and unhappy paths, with certain goals in place, such as not only running complete integration tests that could perhaps lead me to an edge case but also considering low hanging fruits.

Proceeded through Solana’s most common vulnerabilities:

- Missing ownership checks
- Missing signer checks
- Signed invocation of unverified programs
- Solana account confusions
- Missing freeze authority checks
- Insufficient SPL account verification
- Arithmetic over and underflows
- Loss of precision

But only Low severity was found during the assessment, in which code review played a bigger part:

No freeze authority checks

Description: During Initialization, code makes a call to create_pool, where create the pool, note that initialization makes an almost complete set of checks, including:

- Decimal
- Authority
- Token_program
- Payer

It lacks freeze authority checks, while not much of an attack matter, but more of potential scam, where a malicious pool creator could retrieve its content and freeze them, although very unlikely.

Overflow on Initialization (edge case crash likely cause by adding lamports to an)

No Mint check on recipient token account:

Description: During analyses of both collect_fund_fee and collect_protocol_fund_fee, it is noted that while the function itself is well guarded and can be called by the owner or admin, both recipient_token_0_account and recipient_token_1_account are lacking mint checks, but in this very case that is not an issue since its check for a TokenAccount, and mint_to will accuse the incompatibilities of mint in the account without any losses.

No owner check on recipient token account:

Description:

Both collect_fund_fee and collect_protocol_fund_fee are supposed to Only be called by its admin or fund_owner can that will be collecting the fee.

Although very unlikely any account with the correct mint could be passed here, which means an account who is not intended to be the recipient of the fees.

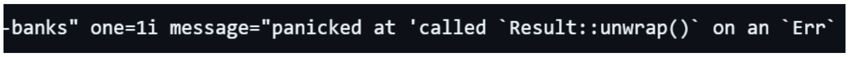
Loss of precision:

Description: the program relies heavily on integer arithmetics which could have a degree of loss of precision, although this would potentially only show at very edge cases, and it would not incur in relevant losses, nonetheless I would recommend the use of fixed point arithmetics instead.

Fuzzing

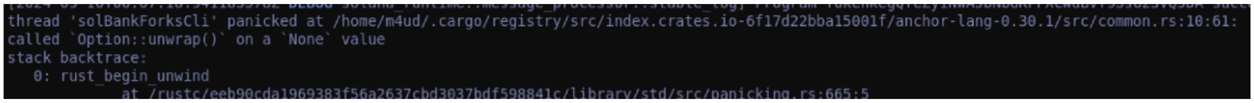
Appropriate error handling:

Description:
During a run of fuzzing unhappy paths, I came across a crash which was caused by *unwrap()*, during improper data account creation.
Throughout the code there are multiple instances of *unwrap()* method being used, while it's well known that in instances where *unwrap()* over a None value will likely cause the program to panic.



```
-banks" one=1i message="panicked at 'called `Result::unwrap()` on an `Err`
```

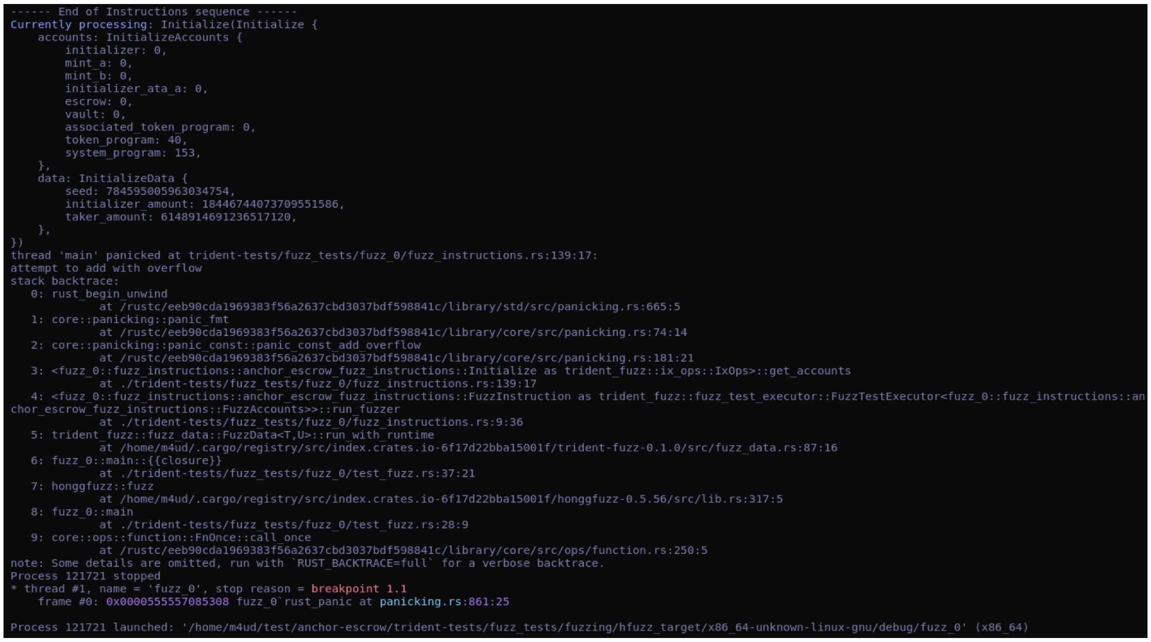
and



```
thread 'solBankForksCli' panicked at /home/m4ud/.cargo/registry/src/index.crates.io-6f17d22bba15001f/anchor-lang-0.30.1/src/common.rs:10:61:
called `Option::unwrap()` on a `None` value
stack backtrace:
   0: rust_begin_unwind
             at /rustc/eeb90cda1969383f56a2637cbd3037bdf598841c/library/std/src/panicking.rs:665:5
```

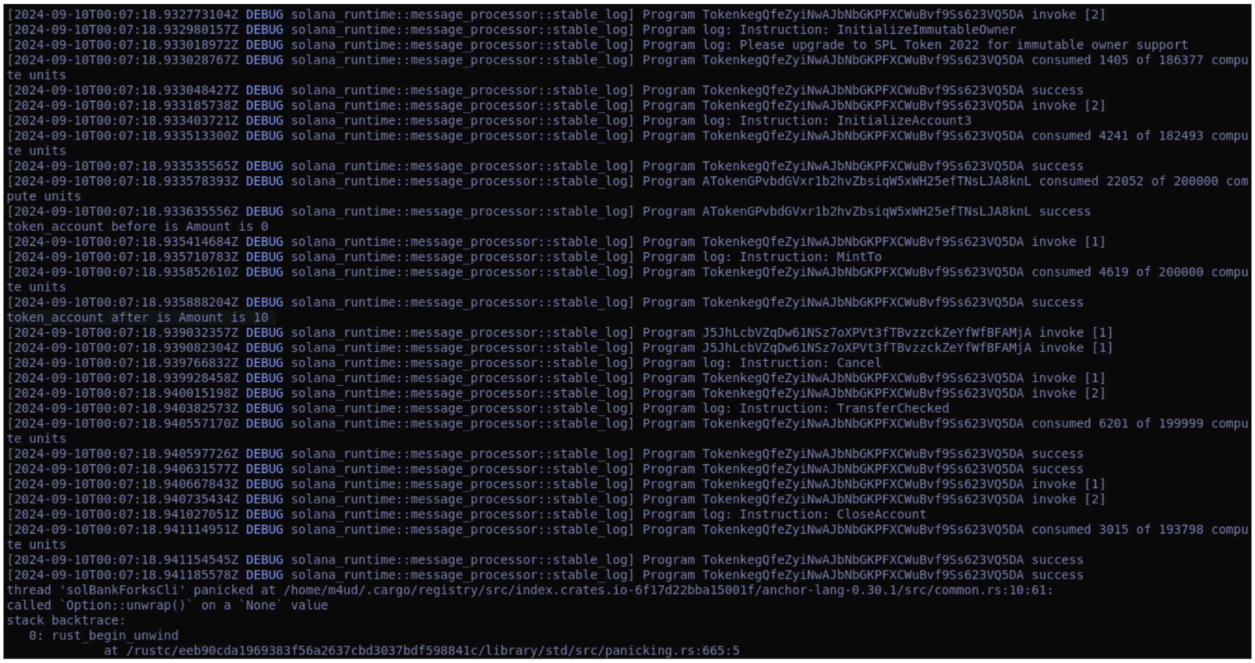
Note: as I the Raydium projects happened to find itself on a very good standing with good coding and security practices it lead me to not have any good finds with the fuzzing, so I moved onto a smaller Escrow project, which I will be sharing here some bugs found in edge case with the fuzzer help:

Overflow
When sending lamports back, it can be seen that the initialized_amout practically held the maximum value for an u64.



```
------ End of Instructions sequence ------
Currently processing: Initialize(Initialize {
    accounts: InitializeAccounts {
        initializer: 0,
        mint_a: 0,
        mint_b: 0,
        initializer_ata_a: 0,
        escrow: 0,
        vault: 0,
        associated_token_program: 0,
        token_program: 40,
        system_program: 153,
    },
    data: InitializeData {
        seed: 784595005963034754,
        initializer_amount: 18446744073709551586,
        taker_amount: 6148914691236517120,
    },
})
thread 'main' panicked at trident-tests/fuzz_tests/fuzz_0/fuzz_instructions.rs:139:17:
attempt to add with overflow
stack backtrace:
   0: rust_begin_unwind
             at /rustc/eeb90cda1969383f56a2637cbd3037bdf598841c/library/std/src/panicking.rs:665:5
   1: core::panicking::panic_fmt
             at /rustc/eeb90cda1969383f56a2637cbd3037bdf598841c/library/core/src/panicking.rs:74:14
   2: core::panicking::panic_const::panic_const_add_overflow
             at /rustc/eeb90cda1969383f56a2637cbd3037bdf598841c/library/core/src/panicking.rs:181:21
   3: <fuzz_0::fuzz_instructions::anchor_escrow_fuzz_instructions::Initialize as trident_fuzz::ix_ops::IxOps>::get_accounts
             at ./trident-tests/fuzz_tests/fuzz_0/fuzz_instructions.rs:139:17
   4: <fuzz_0::fuzz_instructions::anchor_escrow_fuzz_instructions::FuzzInstruction as trident_fuzz::fuzz_test_executor::FuzzTestExecutor<fuzz_0::fuzz_instructions::an
chor_escrow_fuzz_instructions::FuzzAccounts>>::run_fuzzer
             at ./trident-tests/fuzz_tests/fuzz_0/fuzz_instructions.rs:9:36
   5: trident_fuzz::fuzz_data::FuzzData<T,U>::run_with_runtime
             at /home/m4ud/.cargo/registry/src/index.crates.io-6f17d22bba15001f/trident-fuzz-0.1.0/src/fuzz_data.rs:87:16
   6: fuzz_0::main::{{closure}}
             at ./trident-tests/fuzz_tests/fuzz_0/test_fuzz.rs:37:21
   7: honggfuzz::fuzz
             at /home/m4ud/.cargo/registry/src/index.crates.io-6f17d22bba15001f/honggfuzz-0.5.56/src/lib.rs:317:5
   8: fuzz_0::main
             at ./trident-tests/fuzz_tests/fuzz_0/test_fuzz.rs:28:9
   9: core::ops::function::FnOnce::call_once
             at /rustc/eeb90cda1969383f56a2637cbd3037bdf598841c/library/core/src/ops/function.rs:250:5
note: Some details are omitted, run with `RUST_BACKTRACE=full` for a verbose backtrace.
Process 121721 stopped
* thread #1, name = 'fuzz_0', stop reason = breakpoint 1.1
    frame #0: 0x0000555557085308 fuzz_0`rust_panic at panicking.rs:861:25

Process 121721 launched: '/home/m4ud/test/anchor-escrow/trident-tests/fuzz_tests/fuzzing/hfuzz_target/x86_64-unknown-linux-gnu/debug/fuzz_0' (x86_64)
```

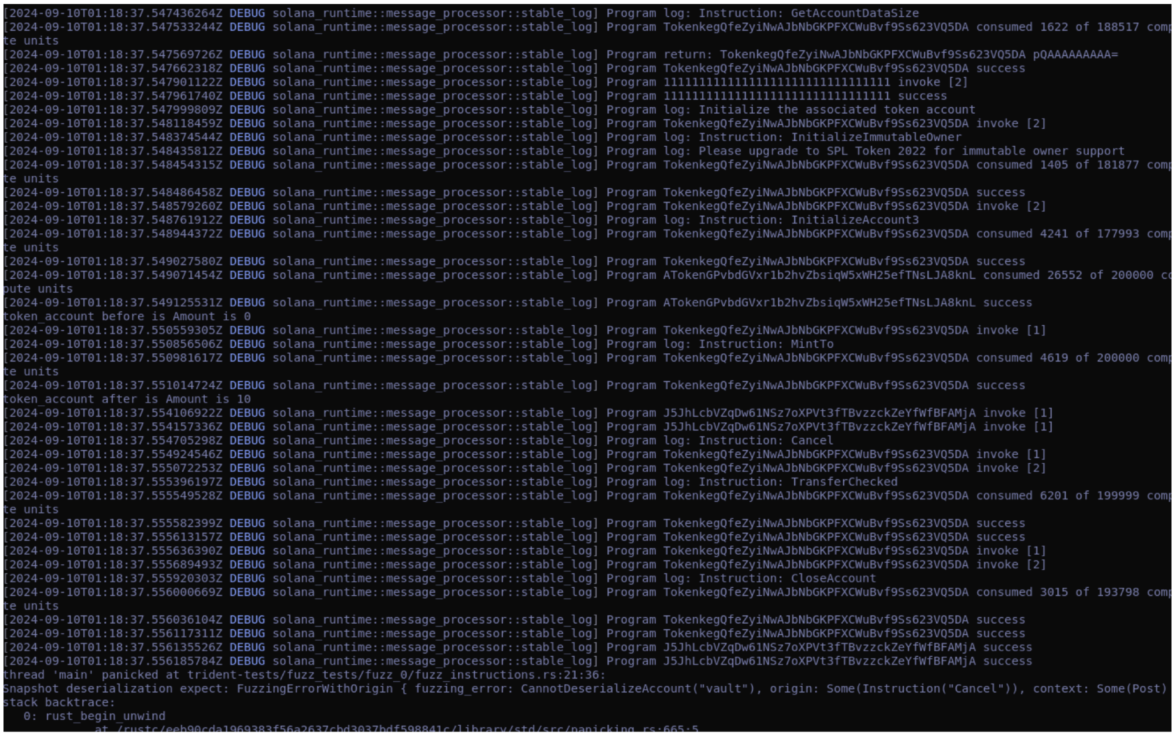
Unwrap



```
[2024-09-10T00:07:18.932773104Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA invoke [2]
[2024-09-10T00:07:18.932980157Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: InitializeImmutableOwner
[2024-09-10T00:07:18.933018972Z DEBUG solana_runtime::message_processor::stable_log] Program log: Please upgrade to SPL Token 2022 for immutable owner support
[2024-09-10T00:07:18.933028767Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA consumed 1405 of 186377 compu
te units
[2024-09-10T00:07:18.933048427Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA success
[2024-09-10T00:07:18.933185738Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA invoke [2]
[2024-09-10T00:07:18.933403721Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: InitializeAccount3
[2024-09-10T00:07:18.933513300Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA consumed 4241 of 182493 compu
te units
[2024-09-10T00:07:18.933535565Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA success
[2024-09-10T00:07:18.933578393Z DEBUG solana_runtime::message_processor::stable_log] Program ATokenGPvbdGVxr1b2hvZbsiqW5xWH25efTNsLJA8knL consumed 22052 of 200000 com
pute units
[2024-09-10T00:07:18.933635556Z DEBUG solana_runtime::message_processor::stable_log] Program ATokenGPvbdGVxr1b2hvZbsiqW5xWH25efTNsLJA8knL success
token_account before is Amount is 0
[2024-09-10T00:07:18.935414684Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA invoke [1]
[2024-09-10T00:07:18.935710783Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: MintTo
[2024-09-10T00:07:18.935852610Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA consumed 4619 of 200000 compu
te units
[2024-09-10T00:07:18.935888204Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA success
token_account after is Amount is 10
[2024-09-10T00:07:18.939032357Z DEBUG solana_runtime::message_processor::stable_log] Program J5JhLcbVZqDw61NSz7oXPVt3fTBvzzckZeYfWfBFAMjA invoke [1]
[2024-09-10T00:07:18.939082304Z DEBUG solana_runtime::message_processor::stable_log] Program J5JhLcbVZqDw61NSz7oXPVt3fTBvzzckZeYfWfBFAMjA invoke [1]
[2024-09-10T00:07:18.939766832Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: Cancel
[2024-09-10T00:07:18.939928458Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA invoke [1]
[2024-09-10T00:07:18.940015198Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA invoke [2]
[2024-09-10T00:07:18.940382573Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: TransferChecked
[2024-09-10T00:07:18.940557170Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA consumed 6201 of 199999 compu
te units
[2024-09-10T00:07:18.940597726Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA success
[2024-09-10T00:07:18.940631577Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA success
[2024-09-10T00:07:18.940667843Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA invoke [1]
[2024-09-10T00:07:18.940735434Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA invoke [2]
[2024-09-10T00:07:18.941027051Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: CloseAccount
[2024-09-10T00:07:18.941114951Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA consumed 3015 of 193798 compu
te units
[2024-09-10T00:07:18.941154545Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA success
[2024-09-10T00:07:18.941185578Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA success
thread 'solBankForksCli' panicked at /home/m4ud/.cargo/registry/src/index.crates.io-6f17d22bba15001f/anchor-lang-0.30.1/src/common.rs:10:61:
called `Option::unwrap()` on a `None` value
stack backtrace:
   0: rust_begin_unwind
             at /rustc/eeb90cda1969383f56a2637cbd3037bdf598841c/library/std/src/panicking.rs:665:5
```

Potential unexpected behavior in trident causing panic:

Description: test a function that would call CloseAccount at the end would cause the account to not be in the correct state for account_snapshot deserialization.



```
[2024-09-10T01:18:37.547436264Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: GetAccountDataSize
[2024-09-10T01:18:37.547533244Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA consumed 1622 of 188517 comp
te units
[2024-09-10T01:18:37.547569726Z DEBUG solana_runtime::message_processor::stable_log] Program return: TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA pQAAAAAAAAA=
[2024-09-10T01:18:37.547662318Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA success
[2024-09-10T01:18:37.547901122Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 invoke [2]
[2024-09-10T01:18:37.547961740Z DEBUG solana_runtime::message_processor::stable_log] Program 11111111111111111111111111111111 success
[2024-09-10T01:18:37.547999809Z DEBUG solana_runtime::message_processor::stable_log] Program log: Initialize the associated token account
[2024-09-10T01:18:37.548118459Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA invoke [2]
[2024-09-10T01:18:37.548374544Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: InitializeImmutableOwner
[2024-09-10T01:18:37.548435812Z DEBUG solana_runtime::message_processor::stable_log] Program log: Please upgrade to SPL Token 2022 for immutable owner support
[2024-09-10T01:18:37.548454315Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA consumed 1405 of 181877 comp
te units
[2024-09-10T01:18:37.548486458Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA success
[2024-09-10T01:18:37.548579260Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA invoke [2]
[2024-09-10T01:18:37.548761912Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: InitializeAccount3
[2024-09-10T01:18:37.548944372Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA consumed 4241 of 177993 comp
te units
[2024-09-10T01:18:37.549027580Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA success
[2024-09-10T01:18:37.549071454Z DEBUG solana_runtime::message_processor::stable_log] Program ATokenGPvbdGVxr1b2hvZbsiqW5xWH25efTNsLJA8knL consumed 26552 of 200000 co
pute units
[2024-09-10T01:18:37.549125531Z DEBUG solana_runtime::message_processor::stable_log] Program ATokenGPvbdGVxr1b2hvZbsiqW5xWH25efTNsLJA8knL success
token_account before is Amount is 0
[2024-09-10T01:18:37.550559305Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA invoke [1]
[2024-09-10T01:18:37.550856506Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: MintTo
[2024-09-10T01:18:37.550981617Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA consumed 4619 of 200000 comp
te units
[2024-09-10T01:18:37.551014724Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA success
token_account after is Amount is 10
[2024-09-10T01:18:37.554106922Z DEBUG solana_runtime::message_processor::stable_log] Program J5JhLcbVZqDw61NSz7oXPVt3fTBvzzckZeYfWfBFAMjA invoke [1]
[2024-09-10T01:18:37.554157336Z DEBUG solana_runtime::message_processor::stable_log] Program J5JhLcbVZqDw61NSz7oXPVt3fTBvzzckZeYfWfBFAMjA invoke [1]
[2024-09-10T01:18:37.554705298Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: Cancel
[2024-09-10T01:18:37.554924546Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA invoke [1]
[2024-09-10T01:18:37.555072253Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA invoke [2]
[2024-09-10T01:18:37.555396197Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: TransferChecked
[2024-09-10T01:18:37.555549528Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA consumed 6201 of 199999 comp
te units
[2024-09-10T01:18:37.555582399Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA success
[2024-09-10T01:18:37.555613157Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA success
[2024-09-10T01:18:37.555636390Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA invoke [1]
[2024-09-10T01:18:37.555689493Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA invoke [2]
[2024-09-10T01:18:37.555920303Z DEBUG solana_runtime::message_processor::stable_log] Program log: Instruction: CloseAccount
[2024-09-10T01:18:37.556000669Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA consumed 3015 of 193798 comp
te units
[2024-09-10T01:18:37.556036104Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA success
[2024-09-10T01:18:37.556117311Z DEBUG solana_runtime::message_processor::stable_log] Program TokenkegQfeZyiNwAJbNbGKPFXCWuBvf9Ss623VQ5DA success
[2024-09-10T01:18:37.556135526Z DEBUG solana_runtime::message_processor::stable_log] Program J5JhLcbVZqDw61NSz7oXPVt3fTBvzzckZeYfWfBFAMjA success
[2024-09-10T01:18:37.556185784Z DEBUG solana_runtime::message_processor::stable_log] Program J5JhLcbVZqDw61NSz7oXPVt3fTBvzzckZeYfWfBFAMjA success
thread 'main' panicked at trident-tests/fuzz_tests/fuzz_0/fuzz_instructions.rs:21:36:
Snapshot deserialization expect: FuzzingErrorWithOrigin { fuzzing_error: CannotDeserializeAccount("vault"), origin: Some(Instruction("Cancel")), context: Some(Post)
stack backtrace:
   0: rust_begin_unwind
             at /rustc/eeb90cda1969383f56a2637cbd3037bdf598841c/library/std/src/panicking.rs:665:5
```