CERTIK

Security Assessment

UncleMine

Jun 13th, 2022

Table of Contents

About

Summary

This report has been prepared for UncleMine to discover issues and vulnerabilities in the source code of the UncleMine 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	UncleMine
Platform	Solana
Language	Rust
Codebase	https://github.com/unclemine/fixedlock-staking-reward
Commit	• bf7b2f363f0b22f02968ca61c48ebce081215226 • 9670fc1560f0d4dbb7623963bb175e3b8586cba2

Audit Summary

Delivery Date	Jun 13, 2022 UTC
Audit Methodology	Static Analysis, Manual Review
Key Components	fixedlock-staking-reward

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Mitigated	Partially Resolved	Resolved
Critical	0	0	0	0	0	0	0
Major	2	0	0	1	0	0	1
Medium	2	0	0	0	0	0	2
Minor	6	0	0	3	0	0	3
Optimization	0	0	0	0	0	0	0
Informational	10	0	0	10	0	0	0
Discussion	0	0	0	0	0	0	0

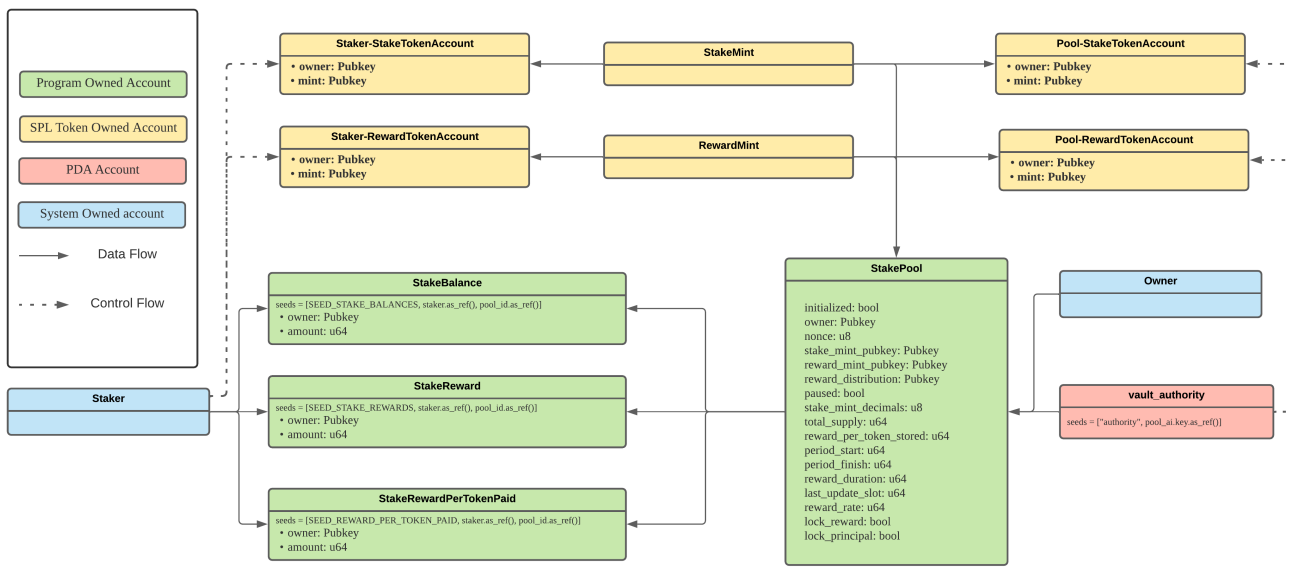
Audit Scope

ID	File	SHA256 Checksum
SRC		
STA	state	
STK	state/stake_pool.rs	70a4859be9521d8aa24c26fd8b65fac5ce2845a4bff699e2aed5a3fa45f48181
STE	state/stake_states.rs	155833f474dc7fb8852b99bce14619b073b24ce4676f465044af7cdcc56eb963
MOD	state/mod.rs	c7607c3cb5803c3b6e1226156b01ab81fd8ed25b36ebb82024eacc33c07ca93b
ENT	entrypoint.rs	ab21c37ed97197eace5abf84729fb2141b8863cfa102d890e2a0b2e5dfa2798c
INS	instruction.rs	9c9c82d54488c28927eba9098e7dc75505a4947bb53c6f245b7a9bd164bcf323
UTI	utils.rs	b2c22e83c3ade73256cd90eaf0463abf411db81ddebdb8e19001bfdce0e666a0
PRO	processor.rs	9249762c8a269070f591060a69bd16049d01c89208c0b91c694f47e33ce1416e
ERR	error.rs	8391c50324cde35c7fc27f058732f842ed4db1d324f81e687f8ddcb77616a342
LIB	lib.rs	340eddd7c04480469ed62276424ca649bf549a4f8662d2ec7956317ca4fb9408

Understandings

There are two time-locks on the reward and the principal when claiming the reward or withdrawing the principal, users should have an understanding of the two time-locks before any operations, the project owner should also expose the actual values of the two time-locks and inform them to users.

System overview



External Dependencies

The project mainly contains the following dependencies:

Dependency	Version
solana-program	1.8.2
spl-token	3.2.0
arrayref	0.3.6
borsh	0.9.1
borsh-derive	0.9.1
num=derive	0.3
num-traits	0.2
num_enum	0.5.4

Dependency	Version
thiserror	1.0
bytes	1.1.0
spl-associated-token-account	1.0.3
metaplex-token-metadata	
bincode	1.3.3
serde	1

It should also be noted here that the code dependencies are 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 the account should be carefully set.

We assume these dependencies are valid and non-vulnerable factors and implement proper logic to collaborate with the current project.

Privileged Functions

The program `reward_pool` contains the following privileged functions/instructions that are restricted by multiple roles, and they are used to modify the contract configurations and address attributes:

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

- `process_change_reward_distribution()` will change the account for `reward_distribution` role.
- `process_change_owner()` will change the ownership of the pool.
- `process_set_paused()` update the `pause` status.

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

- `process_notify_reward_amount()` will update the metadata of the pool.

Additionally, if the program is upgradeable, the upgrade authority account can upgrade the account, thus causing unexpected consequences.

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

Findings

Critical	**0** (0.00%)
Major	**2** (10.00%)
Medium	**2** (10.00%)
Minor	**6** (30.00%)
Informational	**10** (50.00%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
GLOBAL-01	Mint Account Not Check	Logical Issue	Informational	Acknowledged
PRO-01	Centralization Related Risks	**Centralization / Privilege**	**Major**	Acknowledged
PRO-02	Lack Of Token Owner Check	Logical Issue	Major	Resolved
PRO-03	Lack Of The Mint Check On Token Accounts	Logical Issue	Medium	Resolved
PRO-04	Invalid Return Value	Logical Issue	Medium	Resolved
PRO-05	Program Account Not Included	Logical Issue	Minor	Resolved
PRO-06	Potential Overflow/Underflow	Mathematical Operations	Minor	Resolved
PRO-07	Inaccurate `unpack` Method	Logical Issue	Minor	Resolved
PRO-08	Lack Of Check The System Level Account	Volatile Code	Minor	Acknowledged
PRO-09	Discussion On `stake_mint` And `reward_mint`	Logical Issue	Minor	Acknowledged
PRO-10	Lack Of Validation On Associated Token Accounts	Logical Issue	Minor	Acknowledged
PRO-11	Redundant Code	Logical Issue	Informational	Acknowledged
PRO-12	Missing Emit Events	Coding Style	Informational	Acknowledged

ID	Title	Category	Severity	Status
PRO-13	Unused Functions	Gas Optimization	Informational	Acknowledged
PRO-14	Lack Of Key Parameters In The `msg!`	Coding Style	Informational	Acknowledged
PRO-15	Lack Of Checking `lamports` Of `pool_ai` Account	Logical Issue, Volatile Code	Informational	Acknowledged
PRO-16	Invalid Error Handling	Logical Issue	Informational	Acknowledged
PRO-17	Lack Of Check The Reward Lock	Logical Issue	Informational	Acknowledged
PRO-18	Lack Of Checking The Owner Of Associated Token Accounts	Volatile Code	Informational	Acknowledged
STE-01	Type Cosplay	Language Specific	Informational	Acknowledged

GLOBAL-01 | Mint Account Not Check

Category	Severity	Location	Status
Logical Issue	Informational		Acknowledged

Description

The `reward_mint` and `stake_mint` accounts used in this project should be checked whether it is a valid mint account and whether it is owned by the spl token program.

Recommendation

Consider adding checks on the `reward_mint` and `stake_mint` accounts. For example,

```
if reward_mint_ai.data_len() != spl_token::state::Mint::LEN {
    return Err(ProgramError::InvalidAccountData);
}

if reward_mint_ai.owner != spl_token::id(){
    return Err(ProgramError::InvalidAccountData);
}
```

Alleviation

[UncleMine]: The team acknowledged this issue and decided not change the codebase this time.

PRO-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	**Major**	processor.rs: 201, 264, 786, 809	Acknowledged

Description

In the program `processor`, the role `owner` has authority over the following functions:

- `process_change_reward_distribution()` will change the account for `reward_distribution` role.
- `process_change_owner()` will change the ownership of the pool.
- `process_set_paused()` update the `pause` status.

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

In the program `processor`, the role `reward_distribution` has authority over the following functions:

- `process_notify_reward_amount()` will update the metadata of the pool.

Any compromise to the `reward_distribution` 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.

In commit 9670fc1560f0d4dbb7623963bb175e3b8586cba2, there is a new centralized function `process_change_duration` which can be only called by the role `reward_distribution`.

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 (⅔, ⅗) 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.

Alleviation

[UncleMine]: The team acknowledged this finding and will introduce some decentralized auth control, like multisig in the near future.

PRO-02 | Lack Of Token Owner Check

Category	Severity	Location	Status
Logical Issue	Major	processor.rs: 243, 302	Resolved

Description

The following token accounts lack token owner check and associated token address check may cause unexpected errors:

- During the `process_stake`, the staking token will be transferred from the user's token account to the pool token account `pool_stake_mint_ata_ai`. Due to there being no token owner check(at the data stage), the `pool_stake_mint_ata_ai` may not be the associated token account owned by the pool authority. The user can increase the stake balance freely by setting `pool_stake_mint_ata_ai` as another token account under the user's control.
- During the `process_notify_reward_amount`, there is a check on the balance of the `pool_reward_ata_ai` account. Due to there being no token owner check(at the data stage), the `pool_reward_ata_ai` may not be the associated token account owned by the pool authority. If a wrong account passed in, the `if` check will lose its functionality.

Recommendation

Recommend adding token owner check and associated token address check for aforementioned accounts to ensure the valid pool account being used.

Alleviation

[UncleMine]: The team resolved this issue in commit 9670fc1560f0d4dbb7623963bb175e3b8586cba2 by checking the given `pool_stake_mint_ata_ai` in `process_stake` function and removing the `pool_stake_mint_ata_ai` account logic in `process_notify_reward_amount`.

[CertiK]: For function `process_notify_reward_amount`, auditors agree that removing the `pool_stake_mint_ata_ai` account and related logic will solve the aforementioned issue, but lack of `pool.reward_rate` validation may cause not enough reward to redeem. The team should confirm if this is intended. Also, the source of reward should be transparent during actual deployment to avoid unredeemable rewards.

[UncleMine]: The team acknowledged and it is intended that reward might be inadequate. The reward manager is currently controlled by the team.

PRO-03 | Lack Of The Mint Check On Token Accounts

Category	Severity	Location	Status
Logical Issue	Medium	processor.rs: 466, 549, 651	Resolved

Description

In function `process_withdraw`, `process_get_reward`, and `process_exit`, the user can withdraw the staking token or reward token from a pool. The problem is, there is no mint-check on the token account in these functions, otherwise, users can select a token with a higher value to withdraw:

- if the staking token is withdrawn, both accounts should be `pool.stake_mint_pubkey`
- if the reward token is withdrawn, both accounts should be `pool.reward_mint_pubkey`

For example, In `process_get_reward`, assume staking token have higher value, the attacker can set `pool_reward_mint_ata_ai` and `staker_reward_ata_ai` to corresponding staking token account(`staker_stake_mint_ata_ai` and `pool_stake_mint_ata_ai`). As long as the reward check is passed and amount larger than 0, the staking token can be withdrawn at L623:

```
            invoke_signed(
                &spl_token::instruction::transfer(
                    &spl_token::ID,
                    &pool_reward_mint_ata_ai.key.clone(),
                    &staker_reward_ata_ai.key.clone(),
                    &pool_authority_ai.key.clone(),
                    &[],
                    amount,
                )?,
                &[
                    pool_reward_mint_ata_ai.clone(),
                    staker_reward_ata_ai.clone(),
                    pool_authority_ai.clone(),
                ],
                &[&[b"authority", pool_ai.key.as_ref(), &[pool.nonce]]],
            )?;
```

Due to both pool staking and pool reward token accounts sharing the same authority.

Recommendation

Recommend adding a mint check to ensure the token type in aforementioned functions, or similar to PRO-14, confirm the given token address being used is equal to the reward token address and the stake token

address for the pool.

Alleviation

[UncleMine]: The team resolved this issue in commit 9670fc1560f0d4dbb7623963bb175e3b8586cba2 by adding mint-check in aforementioned functions.

PRO-04 | Invalid Return Value

Category	Severity	Location	Status
Logical Issue	Medium	processor.rs: 1123, 1127, 1141, 1146, 1159~1164	Resolved

Description

Function `balance_of`, `user_reward_per_token_paid` and `staker_reward` maintain both check for PDA account and owner check in account data. But when the check failed, it will not raise an error, instead, it will return 0, which will cause unexpected loss. For example, In function `update_staker_reward`, if user bypass a unexpected `staker_balance_ai`, At L982, the `staker_balance` will be assigned to 0, due to the aforementioned check failed:

```
        let (stake_addr, _) = ProcessorHelper::address_staker_balance(pool_id, staker,
program_id);
        if &stake_addr != staker_balance_ai.key {
            return 0;
        }
        let staker_balance =
StakeBalance::unpack(&staker_balance_ai.data.borrow()).unwrap();
        if &staker_balance.owner != staker {
            return 0;
        }
```

The function will continue with invoking the `ProcessorHelper::earned` function at L992 to determine the reward amount, due to the balance being 0 in the previous step, the amount of reward will be 0 either. And at L1012, the return value from `earned` will update the stake reward account as 0, and the user's reward will be lost.

Recommendation

Recommend throwing corresponding error, instead of value 0 in aforementioned check.

Alleviation

[UncleMine]: The team resolved this issue by throwing corresponding error in commit 9670fc1560f0d4dbb7623963bb175e3b8586cba2

PRO-05 | Program Account Not Included

Category	Severity	Location	Status
Logical Issue	Minor	processor.rs: 156~170, 174~188, 329~348, 359~378, 392~410, 441~455, 521~536, 582~597, 623~638, 678~693, 722~737, 763~778, 869~884, 918~933	Resolved

Description

During the test under "Solana 1.8.2", the linked invocations failed with an error: `An account required by the instruction is missing`.

```
/// Invoke a cross-program instruction.
///
/// Notes:
/// - RefCell checking can be compute unit expensive, to avoid that expense use
///   `invoke_unchecked` instead, but at your own risk.
/// - The program id of the instruction being issued must also be included in
///   `account_infos`.
pub fn invoke(instruction: &Instruction, account_infos: &[AccountInfo]) -> ProgramResult
{
    invoke_signed(instruction, account_infos, &[])
}
```

The source code implementation in Solana `1.8.2` implies that when calling a cross-program-invocation by the `invoke` function, the target program account should be included in `account_infos`.

Reference: solana-sdk 1.8.2

Examples,

The following code invokes the `create_associated_token_account` instruction, yet lacking the `spl_associated_token_account` as input program in `account_infos`.

```
            invoke(
                &spl_associated_token_account::create_associated_token_account(
                    signer_ai.key,
                    &pool_authority_pubkey,
                    reward_mint_ai.key,
                ),
                &[
                    signer_ai.clone(),
                    pool_reward_ata_ai.clone(),
                    pool_authority_ai.clone(),
```

```
                reward_mint_ai.clone(),
                system_program_ai.clone(),
                spl_token_program_ai.clone(),
                rent_program_ai.clone(),
            ],
        )?;
```

The following code invokes the `create_account` instruction, yet lacking the `system_program` program as input in `account_infos`.

```
          invoke_signed(
            &system_instruction::create_account(
                staker_ai.key,
                &key,
                Rent::get()?.minimum_balance(StakeBalance::LEN),
                StakeBalance::LEN as u64,
                program_id,
            ),
            &[
                staker_ai.clone(),
                staker_balance_ai.clone(),
                staker_ai.clone(),
            ],
            &[&[
                SEED_STAKE_BALANCES,
                staker_ai.key.as_ref(),
                pool_ai.key.as_ref(),
                &[nonce],
            ]],
        )?;
```

The following code invokes the `transfer` instruction, yet lacking the `spl_token` program as input in `account_infos`.

```
    invoke(
        &spl_token::instruction::transfer(
            &spl_token::ID,
            staker_stake_minit_ata_ai.key,
            pool_stake_mint_ata_ai.key,
            staker_ai.key,
            &[],
            amount,
        )?,
        &[
            staker_stake_minit_ata_ai.clone(),
            pool_stake_mint_ata_ai.clone(),
            staker_ai.clone(),
```

```
            ],
        )?;
```

Recommendation

Recommend consider including the target program when calling instructions if the project is using the "Solana 1.8.2". For example, the `create_associated_token_account` instruction could be modified to:

```
            invoke(
               &spl_associated_token_account::create_associated_token_account(
                   signer_ai.key,
                   &pool_authority_pubkey,
                   reward_mint_ai.key,
               ),
               &[
                   _spl_ata_program_ai.clone(),
                   signer_ai.clone(),
                   pool_reward_ata_ai.clone(),
                   pool_authority_ai.clone(),
                   reward_mint_ai.clone(),
                   system_program_ai.clone(),
                   spl_token_program_ai.clone(),
                   rent_program_ai.clone(),
               ],
           )?;
```

Alleviation

[UncleMine]: The team resolved this issue in commit [9670fc1560f0d4dbb7623963bb175e3b8586cba2](9670fc1560f0d4dbb7623963bb175e3b8586cba2)by updating to solana v1.10.

PRO-06 | Potential Overflow/Underflow

Category	Severity	Location	Status
Mathematical Operations	Minor	processor.rs: 232, 240, 251	Resolved

Description

The linked code could lead to an integer overflow/underflow. For example,

```
pool.period_finish = start_slot + pool.reward_duration;
```

Recommendation

It is advised to use `checked_add`/`checked_sub` function to avoid unexpected errors.

Alleviation

[UncleMine]: The team resolved this issue by using `checked_add`/`checked_sub` function in commit 9670fc1560f0d4dbb7623963bb175e3b8586cba2.

PRO-07 | Inaccurate `unpack` Method

Category	Severity	Location	Status
Logical Issue	Minor	processor.rs: 1143~1144	Resolved

Description

Since the account type of `staker_reward_per_token_paid_ai` is `StakeRewardPerTokenPaid`, the `StakeRewardPerTokenPaid::unpack` method should be used to deserialize the data field of `staker_reward_per_token_paid_ai`. However, current implementation uses `StakeReward::unpack` to deserialize the data.

```
let staker_reward_paid =
    StakeReward::unpack(&staker_reward_per_token_paid_ai.data.borrow()).unwrap();
```

Recommendation

Recommend using the `StakeRewardPerTokenPaid::unpack` method to unpack `staker_reward_per_token_paid_ai` account. For example,

```
let staker_reward_paid =

StakeRewardPerTokenPaid::unpack(&staker_reward_per_token_paid_ai.data.borrow()).unwrap();
```

Alleviation

[UncleMine]: The team resolved this issue by using the `StakeRewardPerTokenPaid::unpack` method to unpack `staker_reward_per_token_paid_ai` account in commit 9670fc1560f0d4dbb7623963bb175e3b8586cba2.

PRO-08 | Lack Of Check The System Level Account

Category	Severity	Location	Status
Volatile Code	Minor	processor.rs: 113~115, 167~169, 593~595, 689~691	Acknowledged

Description

In the functions `process_initialize_stakepool`, `process_initialize_stakepool`, `process_get_reward` and `process_exit`, lack check the system level account, such as `system_program_ai`, `spl_token_program_ai`, `rent_program_ai` and `_spl_ata_program_ai`. If wrong account given, the instruction will fail or generate unexpected result.

Recommendation

Recommend the team implementing aforementioned account validations.

Alleviation

[UncleMine]: The team acknowledged this issue and decided not change the codebase this time.

[CertiK]: Unchecked cross-program invocation target and system program/sysvar will cause unexpected behavior for the program more than failure. Like what we see in the wormhole exploit, the root cause is unchecked sysvar, although the problem may not be that serious in our program. For the best security practice, we highly recommend implementing the check to cross-program invocation target(program) and system program/sysvar.

PRO-09 | Discussion On `stake_mint` And `reward_mint`

Category	Severity	Location	Status
Logical Issue	Minor	processor.rs: 109~110	Acknowledged

Description

During initializing the stake pool, `stake_mint` and `reward_mint` can be assigned with the same address. It will allow `pool_stake_ata_ai` and `pool_reward_ata_ai` to be the same token account due to the usage of the associated token account.

Recommendation

We would like to check with the team whether it is the intended design or the project is in development.

Alleviation

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

[CertiK]: Using the same token vault for both reward and stake may cause user stake loss when there is not enough reward deposited by the team, so we want to confirm with the team to ensure unexpected circumstances do not happen.

PRO-10 | Lack Of Validation On Associated Token Accounts

Category	Severity	Location	Status
Logical Issue	Minor	processor.rs: 155, 173	Acknowledged

Description

During the `process_initialize_stakepool`, the `pool_stake_ata_ai` and `pool_reward_ata_ai` refers to `pool_authority`'s associated token account for `stake_mint` and `reward_mint`. When data is empty, the corresponding accounts will be created. Otherwise, the function will continue.

```
if pool_stake_ata_ai.data_is_empty() {
...
}

if pool_reward_ata_ai.data_is_empty() {
...
}
```

The concern is the lack of verification when the given associated token account already exists. This can lead to additional problems, such as stake and withdrawal functionality, due to the associated token account not being created.

Recommendation

Recommend to adding validation on the given addresses `pool_stake_ata_ai` and `pool_reward_ata_ai` when it already initialized. For example,

```
if pool_stake_ata_ai.key !=
spl_associated_token_account::get_associated_token_address(&pool_authority_pubkey,
stake_mint_ai.key,) {
  Err(...)
}
```

Alleviation

[UncleMine]: The team acknowledged this issue and decided not change the codebase this time.

PRO-11 | Redundant Code

Category	Severity	Location	Status
Logical Issue	Informational	processor.rs: 130~131, 145, 152	Acknowledged

Description

The code in line 145 is redundant since the code in line 152 performs the same functionality.

```
let mut stake_pool = StakePool::unpack(&pool_ai.data.borrow())
  .or::<ProgramError>(Ok(StakePool::default()))?;
```

```
stake_pool.pack_into_slice(&mut pool_ai.data.borrow_mut());
```

```
Pack::pack(stake_pool, &mut pool_ai.data.borrow_mut())?
```

Recommendation

Consider removing the redundant code in line 145.

Alleviation

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

PRO-12 | Missing Emit Events

Category	Severity	Location	Status
Coding Style	Informational	processor.rs: 264, 651, 786, 809	Acknowledged

Description

The functions that affect the status of sensitive variables should be able to emit events.

- `process_set_paused` affects `paused` of the pool
- `process_exit` affects `stake_balance`, `stake_reward` and `staker_reward_per_token_paid` of the user
- `process_change_owner` affects `owner` of the pool
- `process_change_reward_distribution` affects `reward_distribution` of the pool

Recommendation

Consider adding events for sensitive actions, and emit them in the functions.

Alleviation

[UncleMine]: The team acknowledged this issue and decided not change the codebase this time.

PRO-13 | Unused Functions

Category	Severity	Location	Status
Gas Optimization	Informational	processor.rs: 835, 891, 1104	Acknowledged

Description

The below functions are unused:

- `process_evacuate_staking_token`
- `process_evacuate_rewards`
- `get_reward_for_duration` in `ProcessorHelper`

Recommendation

It is advised remove aforementioned functions if not intended to be used.

Alleviation

[UncleMine]: The team acknowledged this issue and decided not change the codebase this time.

PRO-14 | Lack Of Key Parameters In The `msg!`

Category	Severity	Location	Status
Coding Style	Informational	processor.rs: 102, 192, 207, 255, 457, 540, 642	Acknowledged

Description

```
msg!("invoke: process_notify_reward_amount");
```

The above log lacks concrete parameters in the `msg!` macro, such as pool's key, `amount` and etc. Detailed information in the events can better help developers track the state of the program.

Recommendation

Consider adding more detailed information in the `msg!` macro.

Alleviation

[UncleMine]: The team acknowledged this issue and decided not change the codebase this time.

PRO-15 | Lack Of Checking `lamports` Of `pool_ai` Account

Category	Severity	Location	Status
Logical Issue, Volatile Code	Informational	processor.rs: 106	Acknowledged

Description

The Solana documentation writes that keeping accounts alive on Solana incurs a storage cost called rent. If rent run out on the address of `pool_ai`, then the `pool_ai` account is deleted, which causes the users to lose all of their principal and reward tokens.

Recommendation

We advise the client to check whether the account `pool_ai` is rent-exempt before initializing it.

Alleviation

[UncleMine]: The team acknowledged this issue and decided not change the codebase this time.

PRO-16 | Invalid Error Handling

Category	Severity	Location	Status
Logical Issue	Informational	processor.rs: 1007~1010, 1240	Acknowledged

Description

The following error handling are invalid:

```
            let stake_reward = match StakeReward::unpack(&staker_reward_ai.data.borrow())
{
                Err(_) => StakeReward {
                    owner: *staker,
                    amount: earned,
                },
```

When an error occurred during unpacking, the whole transaction should revert and throw an error to avoid unexpected errors, instead of keeping returning the result and continuing the process.

The same problem are held at L1240:

```
        if staker_balance.owner == Pubkey::default() {
            staker_balance.owner = *stake_authority_ai.key;
        }
```

When the owner of the `staker_balance` addresses zero, which means the `staker_balance` is not valid, an error should be raised, instead of continuing the process.

Recommendation

Recommend implementing complete error handling in the above code snippet.

Alleviation

[UncleMine]: The team acknowledged this issue and it is intended design.

[CertiK]: Based on our observation, when using `unpack` function, there are two possible situations to throw an error: 1. the account is not initialized 2. the account data length does not match. Either result reflects unusual behavior from the program to auditors. We want to get more detail about the design to avoid unexpected circumstances.

PRO-17 | Lack Of Check The Reward Lock

Category	Severity	Location	Status
Logical Issue	Informational	processor.rs: 696~698	Acknowledged

Description

There are two time-locks, i.e. principal time-lock and reward time-lock, in the program. The user can withdraw principal and reward in the function `process_exit`, but the function only check on the principal lock without checking on the reward lock.

Recommendation

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

Alleviation

[UncleMine]: Expected Reward can't be locked alone without the principal. Two locks are controlled by the same duration period.

PRO-18 | Lack Of Checking The Owner Of Associated Token Accounts

Category	Severity	Location	Status
Volatile Code	Informational	processor.rs: 476~477, 551~552, 653~655	Acknowledged

Description

In the function `process_withdraw`,`process_get_reward`, `process_exit`, there are no owner-checks for the `staker_stake_mint_ata_ai`,`staker_reward_mint_ata_ai` account. If a user passes an incorrect account not owned by himself, all his principal or reward will be lost.

Recommendation

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

Alleviation

[UncleMine]: User should be responsible for his input. Since it actually requires an effort if some one really wanna use other's ATA accounts, there seems little meaning of disallowing user to do that.

[CertiK]: Auditors agree that the current design allows a user to send funds to any destination that the user intended. Users should carefully set the destination accounts in the aforementioned functions.

STE-01 | Type Cosplay

Category	Severity	Location	Status
Language Specific	Informational	state/stake_states.rs: 14	Acknowledged

Description

In program `fixedlock-staking-reward`, the staker will maintain three data accounts to hold the stake states:

```
pub struct StakeBalance {
    pub owner: Pubkey,
    pub balance: u64,
}

pub struct StakeReward {
    pub owner: Pubkey,
    pub amount: u64,
}

pub struct StakeRewardPerTokenPaid {
    pub owner: Pubkey,
    pub amount: u64,
}
```

Due to these three account types sharing a similar layout, with `Pubkey` + `u64`, the account type cosplay will be possible, which means the user may use a `StakeRewardPerTokenPaid` account as a `StakeReward` account.

Recommendation

Recommend adding discriminator for each account and check the discriminator before the account usage.

Alleviation

[UncleMine]: The team acknowledged this issue and decided not change the codebase this time.

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.

Gas Optimization

Gas Optimization findings do not affect the functionality of the code but generate different, more optimal EVM opcodes resulting in a reduction on the total gas cost of a transaction.

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

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.