Security Audit Report

STAR ATLAS

5/9/2022

PREPARED FOR:
STAR ATLAS
https://staratlas.com

ARCADIA CONTACT INFO
Email: audits@arcadiamgroup.com
Telegram: https://t.me/thearcadiagroup

Table of Contents

Executive Summary

A Representative Party of **STAR ATLAS** ("**ATLAS**") engaged The Arcadia Group ("Arcadia"), a software development, research, and security company, to conduct a review of the following **FACTION-FLEET** program on the **STARATLASMETA/FACTION-FLEET** github repository at Commit #**f65a21c1a48074071cb2befe561224c8cf11ec29**.

The scope of this audit included the following directories and filetypes:

1. **Programs/score/*.rs**

Arcadia completed this security review using various methods primarily consisting of dynamic analysis. This process included a line-by-line analysis of the in-scope contracts, optimization analysis, analysis of key functionalities and limiters, and reference against intended functionality.

Severity Rating	Number of Original Occurrences	Number of Remaining Occurrences
CRITICAL	0	0
HIGH	0	0
MEDIUM	3	0
LOW	1	0
INFORMATIONAL	10	0

Findings

Duplicate verification for score_vars_accounts data initialization

Issue: **ATLAS-1**
Severity: **INFORMATIONAL**

Target: **process_register_ship.rs, process_deregister_ship.rs**
Finding Type: **DYNAMIC**

score_vars_account initialization status gets verified in context[1]

```
constraint = score_vars_account.to_account_info().data_len() > 0
```

Within the handler, there is another verification[2], this verification is unable to be executed.

```
    if ctx.accounts.score_vars_account.to_account_info().data_len()
== 0 {
        msg!("Scorevars auth account is not created");
        return Err(ErrorCode::ScorevarsNotInitialized.into());
    }
```

This issue persists throughout the entire validation section

1 https://github.com/staratlasmeta/faction-fleet/blob/main/programs/score/src/instructions/process_register_ship.rs#L13

2 https://github.com/staratlasmeta/faction-fleet/blob/main/programs/score/src/instructions/process_register_ship.rs#L45-L48

Action Recommended:

Due to the lack of use of the verification, it may make sense in future iterations to remove the redundant code.

Code Improvement: Move verification in the context as a constraint

Issue: **ATLAS-2**
Severity: **INFORMATIONAL**

Target: **process_register_ship.rs**
Finding Type: **DYNAMIC**

```
let score_vars_account = &ctx.accounts.score_vars_account;
    if ctx.accounts.update_authority_account.key() !=
score_vars_account.update_authority_master {
        msg!("Update authority is not correct");
        return Err(ErrorCode::ScorevarsAuthInvalid.into());
    }
```

This script is used to verify authority[3], this can be verified in `score_vars_account` account constraint. This issue persists in all of the authority validation elements of the code.

3 https://github.com/staratlasmeta/faction-fleet/blob/main/programs/score/src/instructions/process_register_ship.rs#L51-L55

Action Recommended:

Add authority verification constraint in the following lines.[4]

```
    #[account(
        seeds = [b"SCOREVARS".as_ref()],
        bump = scorevars_bump,
        constraint = score_vars_account.to_account_info().data_len()
> 0
    )]
```

No validation for reward_rate, max_reserve, milliseconds_to_burn.

Issue: **ATLAS-3**
Severity: **INFORMATIONAL**

Target: **process_register_ship.rs**
Finding Type: **DYNAMIC**

4 https://github.com/staratlasmeta/faction-fleet/blob/main/programs/score/src/instructions/process_register_ship.rs#L10-L14

```
    score_vars_ship_account.ship_mint =
ctx.accounts.ship_mint.key();
    score_vars_ship_account.reward_rate_per_second =
reward_rate_per_second;
    score_vars_ship_account.fuel_max_reserve = fuel_max_reserve;
    score_vars_ship_account.food_max_reserve = food_max_reserve;
    score_vars_ship_account.arms_max_reserve = arms_max_reserve;
    score_vars_ship_account.toolkit_max_reserve =
toolkit_max_reserve;
    score_vars_ship_account.milliseconds_to_burn_one_fuel =
milliseconds_to_burn_one_fuel;
    score_vars_ship_account.milliseconds_to_burn_one_food =
milliseconds_to_burn_one_food;
    score_vars_ship_account.milliseconds_to_burn_one_arms =
milliseconds_to_burn_one_arms;
    score_vars_ship_account.milliseconds_to_burn_one_toolkit =
milliseconds_to_burn_one_toolkit;
```

There is no validation for these params. Something like a zero-check[5]

Action Recommended:

Add validation.

[5] https://github.com/staratlasmeta/faction-fleet/blob/main/programs/score/src/instructions/process_register_ship.rs#L59-L68

ship_mint validation is not required

Issue: **ATLAS-4**
Severity: **INFORMATIONAL**

Target: **process_deregister_ship.rs**
Finding Type: **DYNAMIC**

We can deregister ships whey they registered already.

And score_vars_ship_account is a PDA with ship_mint address.

So ship_mint validation is not required[6].

```
    if score_vars_ship_account.ship_mint !=
ctx.accounts.ship_mint.key() {
        msg!("Invalid Ship Mint");
        return Err(ErrorCode::InvalidShipError.into());
    }
```

There are other instructions which validate ship_mint.

Action Recommended:

Remove ship_mint validation.

6 https://github.com/staratlasmeta/faction-fleet/blob/main/programs/score/src/instructions/process_deregister_ship.rs#L51-L54

Use `close` keyword in context to close account and get lamports back.

Issue: **ATLAS-5**
Severity: **INFORMATIONAL**

Target: **process_deregister_ship.rs**
Finding Type: **DYNAMIC**

```
    let score_vars_ship_account_info =
&score_vars_ship_account.to_account_info();
    write_data(score_vars_ship_account_info, &vec![0;
score_vars_ship_account_info.data_len()], 0);

    let update_authority_account =
&ctx.accounts.update_authority_account.to_account_info();
    let source_amount: &mut u64 = &mut
score_vars_ship_account_info.lamports.borrow_mut();
    let dest_amount: &mut u64 = &mut
update_authority_account.lamports.borrow_mut();
    let new_dest_amount = dest_amount
        .checked_add(*source_amount)
        .ok_or(ErrorCode::NumericalOverflowError)?;
    *dest_amount = new_dest_amount;
    *source_amount = 0;
```

Instead of these lines, we can simply use `close` keyword to close account and get lamports back[7]

[7] https://github.com/staratlasmeta/faction-fleet/blob/main/programs/score/src/instructions/process_deregister_ship.rs#L58-L68

Please check the following snippet[8]

```
pub struct ClosePool<'info> {
    #[account(mut)]
    refundee: UncheckedAccount<'info>,
    #[account(mut)]
    staking_refundee: Box<Account<'info, TokenAccount>>,
    #[account(mut)]
    reward_a_refundee: Box<Account<'info, TokenAccount>>,
    #[account(mut)]
    reward_b_refundee: Box<Account<'info, TokenAccount>>,
    #[account(
        mut,
        close = refundee,
        has_one = authority,
        has_one = staking_vault,
        has_one = reward_a_vault,
        has_one = reward_b_vault,
        constraint = pool.paused,
        constraint = pool.reward_duration_end > 0,
        constraint = pool.reward_duration_end <
sysvar::clock::Clock::get().unwrap().unix_timestamp.try_into().unwra
p(),
        constraint = pool.user_stake_count == 0,
    )]
```

Action Recommended:

Remove ship_mint validation.

8 https://github.com/step-finance/reward-pool/blob/main/programs/reward-pool/src/lib.rs#L901

Manual calculation of memory size could break a program.

Issue: **ATLAS-6**
Severity: **INFORMATIONAL**

Target: **instructions/*.rs**
Finding Type: **DYNAMIC**

```
pub struct ProcessInitialDeposit<'info> {
    pub player_account: Signer<'info>,
    #[account(
        init,
        payer = player_account,
        seeds = [b"SCORE_INFO".as_ref(),
player_account.key.as_ref(),
ship_mint.to_account_info().key.as_ref()],
        bump = staking_bump,
        space = 209 + 8
    )]
```

When initializing an account, we set the size of the account with the `space` keyword. For now it calculates space manually, so when there are any updates in the future, we could forget updating it, and this will break the program.

Please check the following solution [9]

```
    pub funder_to_authorize: AccountInfo<'info>,
    #[account(init_if_needed, seeds = [
            b"authorization".as_ref(),
            farm.key().as_ref(),
            funder_to_authorize.key().as_ref(),
        ],
        bump,
        payer = farm_manager,
        space = 8 + std::mem::size_of::<AuthorizationProof>())]
    authorization_proof: Box<Account<'info, AuthorizationProof>>,
```

Action Recommended:

Use **std:mem** to calculate space of account.

9 https://github.com/gemworks/gem-farm/blob/main/programs/gem_farm/src/instructions/authorize_funder.rs#L26

TODO comments still exist.

Issue: **ATLAS-7**
Severity: **INFORMATIONAL**

Target: **utils.rs**
Finding Type: **DYNAMIC**

https://github.com/staratlasmeta/faction-fleet/blob/main/programs/score/src/util.rs#L59

There is a `TODO` comment.

Action Recommended:

Figure `TODO` comments.

1. Optimise updating the **current_capacity_timestamp**.

Issue: **ATLAS-8**
Severity: **INFORMATIONAL**

Target: **utils.rs**
Finding Type: **DYNAMIC**

https://github.com/staratlasmeta/faction-fleet/blob/main/programs/score/src/util.rs#L58-L65

`time_passed = current time - old current_capacity_timestamp`.

so `current time = time_passed + old current_capacity_timestamp`.

So we don’t need conditions for updating current_capacity_timestamp.

Action Recommended:

Update L58-L65 to the following script.

`ship_staking_account_info.current_capacity_timestamp = current_timestamp;`

2. Optimise updating the **staked_time_paid**.

Issue: **ATLAS-9**	Target: **ship_staking.rs**
Severity: **INFORMATIONAL**	Finding Type: **DYNAMIC**

https://github.com/staratlasmeta/faction-fleet/blob/main/programs/score/src/state/ship_staking.rs#L180-L182

Action Recommended:

Update L180-L182 to the following script.

`self.staked_time_paid = self.total_time_staked;`

3. Users need to create too many escrow accounts if they have different ships.

Issue: **ATLAS-10**	Target: **process_deposit_resource.rs**
Severity: **LOW**	Finding Type: **DYNAMIC**

https://github.com/staratlasmeta/faction-fleet/blob/main/programs/score/src/instructions/process_deposit_resource.rs#L57

https://github.com/staratlasmeta/faction-fleet/blob/main/programs/score/src/instructions/process_deposit_resource.rs#L104 https://github.com/staratlasmeta/faction-fleet/blob/main/programs/score/src/instructions/process_deposit_resource.rs#L151

These accounts are PDA with player pubKey, shipMint, and resource mint.

So users need to make different escrow accounts for different ship mint.

This means, to get rewards, users need to create 3 * ship quantity accounts for resources, and players need to pay rents for them.

Of course, users can get back rents after close accounts.

But it would be good to use a few accounts.

The user’s staked resource amount is stored in staking info, so no need to make different token accounts per ship.

Action Recommended:

Create PDA with player pubKey and resource mint only.

4. Combine the depositing resource function.

Issue: **ATLAS-11**
Severity: **INFORMATIONAL**

Target: **process_deposit_resource.rs**
Finding Type: **DYNAMIC**

There are **refeed**, **refuel**, **rearm, repair** functions to deposit resources. And they use same calculation.

Now there are different contexts, and different functions for handling deposits. This could lead to unexpected issues when updating the deposit feature - example, update refuel deposit handler, but forget updating rearm, …

Action Recommended:

Optimise deposit context and handler.

5. Need to settle all users' accounts in one transaction.

Issue: **ATLAS-12**
Severity: **MEDIUM**

Target: **process_settle.rs**
Finding Type: **DYNAMIC**

To give fair rewards to all users, we need to settle all users' pending rewards in one transaction.

Ex, if we want to increase reward rate, and if we settle user1, and after 3 seconds, settle user2. Then user1 will get higher reward than user2, because user1 has a lower staked time with lower reward rate.

So the admin needs to combine all users' settle instructions in one transaction.

Here, there is a limit of solana transaction size, so this will prevent settlement for many users at once.

And there is no validation in updating reward rate instructions, if all users' pending reward have been settled.

Action Recommended:

Not sure what is the best way for this.

6. There is no partial withdraw resources functionality

Issue: **ATLAS-13**
Severity: **MEDIUM**

Target: **process_withdraw_resources.rs**
Finding Type: **DYNAMIC**

There is no partial withdrawal functionality, so for this, users need to withdraw all resources, and refuel back.

Action Recommended:

Think if partial withdrawal functionality is required or not.

7. There is no partial withdraw resources functionality

Issue: **ATLAS-14**	Target: **process_withdraw_resources.rs**
Severity: **MEDIUM**	Finding Type: **DYNAMIC**

There is no partial withdrawal functionality, so for this, users need to withdraw all resources, and refuel back.

Action Recommended:

Think if partial withdrawal functionality is required or not.

Conclusion

Arcadia identified issues that occurred at hash **#f65a21c1a48074071cb2befe561224c8cf11ec29**. Arcadia also reviewed the fixes at later commit hashes

Disclaimer

While best efforts and precautions have been taken in the preparation of this document, The Arcadia Group and the Authors assume no responsibility for errors, omissions, or damages resulting from the use of the provided information. Additionally, Arcadia would like to emphasize that the use of Arcadia's services does not guarantee the security of a smart contract or set of smart contracts and does not guarantee against attacks. One audit on its own is not enough for a project to be considered secure; that categorization can only be earned through extensive peer review and battle testing over an extended period.