OBJECT COMPUTING

Object Computing, Inc.
12140 Woodcrest Executive Dr.
Suite 300
St. Louis, MO 63141
+1 314.579.0066

Alien Worlds Federation Contract Audit (EOS) - Revision

Source for this analysis is github: Alien-Worlds/alienworlds-contracts
Commit tag 1a656355b7c2c217898bde31c1929ab7029fe3b6
Files contracts/federation/federation.[ch]pp
Test contracts/federation/federation.test.ts
No ricardian contract sources were available
Business requirements and other engineering artifacts were found in the “ClickUp” document found at
https://doc.clickup.com/d/h/hfd6b-9788/49593a316fe4392/hfd6b-1448

Executive Summary

This document is a revision of a previous version. Some previous commentary has been removed (strike-throughs) with explanations in yellow highlighted text.

~~Four critical security violations were detected in this audit. Each critical violation would leave open the potential for a reentrancy attack which could steal funds or at the very least corrupt data inside the contract. These violations occur in the `stake`, `withdraw`, `refund` and `claim` actions.~~

~~In the `stake` action, DAC tokens are transferred to the user before their deposits table is updated. A hacker could deploy code onto their account that watches for DAC token transfers and upon receiving the token the code could recursively call `stake` again, using their same deposited balance to generate new DAC tokens. Eventually the function would resolve and clear out their deposit, but not after the user had generated many DAC tokens for free.~~

~~In the `withdraw` action, the users deposited funds are transferred back to them and then their deposited balance is updated. An attacker could use the same method as described above to steal other deposited tokens from this contract before the contract eventually updates their deposited balance.~~

~~The `refund` action could be attacked in the same way as the `withdraw` action, stealing deposited tokens from the contract before updating the hackers refund balance.~~

~~The `claim` action could also be attacked in the same manner allowing an attacker to steal Trilium from the reverse claiming pool.~~

The reentrancy attack violations are not applicable to the EOSIO blockchain due to its transaction model. In EOSIO the `action.send()` method does not run the action inline, but instead pushes the action onto the action queue within the transaction. The actions are then executed sequentially preventing any possibility of an action recursively calling itself, and therefore avoiding reentrancy attacks.

Remarks and other minor coding improvements are described further down in this report.

Lack of ricardian contracts is not necessarily a risk but is a lost opportunity for ensuring that the coded logic is congruent with the identified business needs. Test coverage is poor, only 8 out of the 38 total actions have unit tests.

Remark	Minor	Major	Critical
20	4	0	0

Participants

Primary auditors are Jack DiSalvatore and Phil Mesnier, with Ciju John providing a review.

Tools

Static analyzers

- EOSafe appears to be an Academic exercise last updated nearly 4 years ago
- EOSlime is a javascript tool that is also at least 2 years old
- Klevoya is a wasm analyzer tool using simple pattern matches

No static analyzers were used in this exercise. I tried to build the EOSafe tool, but it would not compile. The EOSlime tool was too complicated for me to determine how to build it. Klevoya was not useful because I could not generate an ABI file for the mining contract.

Known Vulnerabilities

List origin github:slowmisg/**eos-smart-contract-security-best-practices**/Readme_EN.md commit tag 5f77e19e50373d341e17a003c492388e9891a2c0

- **Numerical Overflow:** when doing token math, use the `asset` object operands and do not perform operations on the `uint256` that comes from `asset.amount`
- **Authorization Check:** make sure function parameters are consistent with the caller and use `require_auth ` to check
- **Apply Check:** if the contract implements an `apply` function, bind each key action and code to meet the requirements, in order to avoid abnormal and illegal calls.
- **Transfer Error Prompt:** When processing a notification triggered by `require_recipient`, ensure that `transfer.to` is `_self`
- **Random Number Practice:** Random number generator algorithm should not introduce controllable or predictable seeds
- ~~**Reentrancy Attack:** A reentrancy attack can occur when you create a function that makes an external call to another untrusted contract before it resolves any effects. If the attacker can control the untrusted contract, they can make a recursive call back to the original function, repeating interactions that would have otherwise not run after the effects were resolved.~~

~~This simplest example is when a contract does internal accounting with a balance variable and exposes a withdraw function. If the vulnerable contract transfers funds before it sets the balance to zero, the attacker can recursively call the withdraw function repeatedly and drain the whole contract.~~

We relied on visual inspection to look for instances of code that were similar to the examples found in the above document as well as best practices accumulated through many years of experience.

Findings And Recommendations

We examined 38 contract actions and 4 notification callback handlers, 4 private functions, 978 lines of code. The contract actions are documented in clickup.

Remarks

- Test case indentation should be consistent for readability.
- Replace “magic numbers” with constants or variables.
- Use the `asset` whenever possible instead of just `uint64_t` balances. This is useful to compare an entire token (symbol and quantity) instead of just comparing quantities.

Minor

- Add correct variable type casting to `by_total_luck()` accessor function in the `userpoints` table
- In the `setmap` action there is no check on existing maps before emplacing a new row, which could lead to collisions
- No auth check on `setlandregs` action
- No check for existing `landregs` row before emplacing a new one
- Use of deferred transactions are deprecated

Critical

- ~~Multiple incorrect action -> table update cadence leading to the possibility of reentrancy attacks on the following functions:~~
  - ~~`stake`~~
  - ~~`withdraw`~~
  - ~~`refund`~~
  - ~~`claim`~~

Table/Singleton Signatures	**In ClickUp**	**Notes**
typedef multi_index<"landregs"_n, landreg_item> **landregs_table**;	Yes	**Jack** No issues.
typedef eosio::singleton<"state"_n, state_item> **state_singleton**;	Yes	**Jack** Remark Line#35 perhaps change the data type of `total_stake` to `asset` to avoid staking different tokens.
typedef eosio::singleton<"reserve"_n, reserve_item> **reserve_singleton**;	No	**Jack** No issues.
typedef multi_index<"userterms"_n, userterms_item> **userterms_table**;	Yes	**Jack** No issues.

typedef multi_index<"planets"_n, planet_item> **planets_table**;	Yes	**Jack** Remark Line#81 perhaps change the data type of `total_stake` to `asset` to avoid staking different tokens.
using **user_points_table** = multi_index<"userpoints"_n, user_points, indexed_by<"bytotalpts"_n, const_mem_fun<user_points, uint64_t, &user_points::by_total_points>>, indexed_by<"bydailypts"_n, const_mem_fun<user_points, uint64_t, &user_points::by_daily_points>>, indexed_by<"byweeklypts"_n, const_mem_fun<user_points, uint64_t, &user_points::by_weekly_points>>, indexed_by<"bytotalluck"_n, const_mem_fun<user_points, uint64_t, &user_points::by_total_luck>>>;	No	**Jack** Minor line#113 return value should be casted to `uint64_t`.
using **point_offers** = multi_index<"pointoffers"_n, points_offer>;	No	**Jack** No issues.
using **level_offers** = multi_index<"leveloffers"_n, level_offer>;	No	**Jack** No issues.
typedef multi_index<"maps"_n, map_item> **maps_table**;	Yes	**Jack** No issues.
typedef multi_index<"deposits"_n, deposit_item> **deposits_table**;	Yes	**Jack** No issues.
typedef multi_index<"refunds"_n, refund_item, indexed_by<"byaccount"_n, const_mem_fun<refund_item, uint64_t, &refund_item::by_account>>> **refunds_table**;	Yes	**Jack** No issues.
typedef multi_index<"players"_n, player_item> **players_table**;	Yes	**Jack** No issues.
typedef multi_index<"accounts"_n, account> **accounts_table**;	No	**Jack** No issues.

Private Functions	**In ClickUp**	**Notes**
asset get_dac_tokens(name planet_name, uint64_t amount);	No	**Jack** No issues.
name planet_from_symbol(symbol dac_symbol);	No	**Jack** No issues.

void reg_land(uint64_t asset_id, name owner);	No	**Jack** No issues.
void mint_asset(uint32_t template_id, name destn_user);	No	**Jack** No issues.

Action Signatures	In ClickUp	Notes
ACTION addplanet(name planet_name, string title, symbol dac_symbol, string metadata);	Yes	**Jack** No issues.
ACTION updateplanet(name planet_name, string title, string metadata, bool active);	Yes	**Jack** No issues.
ACTION removeplanet(name planet_name);	Yes	**Jack** No issues.
ACTION setmap(name planet_name, uint16_t x, uint16_t y, uint64_t asset_id);	Yes	**Jack** Minor Line#75 add a check to make sure that an existing map does not already exist on the x coordinate before emplacing it to avoid collisions. ``` maps_table _maps(get_self(), planet_name.value); auto map = _maps.find(x); check(map == _maps.end(), “map already exists”); ```
ACTION clearmap(name planet_name);	No	**Jack** No issues. Remark perhaps call this `clearmap**s**` since it deletes all the maps on a planet.
ACTION clearrefunds();	No	**Jack** No issues.
ACTION clearplanets();	No	**Jack** No issues.
ACTION setreserve(uint64_t total);	No	**Jack** No issues.
ACTION clearstate();	No	**Jack** No issues.
ACTION deldeposit(name account);	No	**Jack** No issues.
ACTION fixstake(name planet_name, int64_t total_stake);	Yes	**Jack** No issues.
ACTION settag(name account, string tag);	Yes	**Jack** No issues.
ACTION setavatar(name account, uint64_t avatar_id);	Yes	**Jack** Remark Line#227 replace magic numbers `1` and `2` with `MALE` and `FEMALE`.
ACTION stake(name account, name planet_name, asset quantity);	Yes	**Jack** Remark Line#408 use asset comparison to check that the token symbols match as well. `check(quantity > asset(“TLM”, 0));`

		~~Critical~~ ~~Line#432-438 move these lines to the end of the function, after updating the deposits row, to prevent reentrancy attacks.~~
ACTION withdraw(name account);	Yes	**Jack** ~~Critical~~ ~~Line#455-457 move these lines to the end of the function, after removing the deposits row, to prevent reentrancy attacks.~~
ACTION refund(uint64_t id);	Yes	**Jack** ~~Critical~~ ~~Line#470-473 move these lines to the end of the function, after removing the refunds row, to prevent reentrancy attacks~~
ACTION claim(name planet_name);	Yes	**Jack** Remark Lines#490-491 `binance_name` and `binance_daily` can be constants. Remark Line#540 magic numbers. ~~Critical~~ ~~move Lines#599-605 to after reserve update on line#610 to prevent reentrancy attacks.~~
ACTION logclaim(name planet_name, asset planet_quantity, asset mining_quantity);	Yes	**Jack** Remark function is not implemented.
ACTION miningnft(name receiver, name rarity, name planet_name);	Yes	**Jack** Remark remove commented out code. Or move lines#181-184 to after the `_miningnfts` row modification (line#188) to avoid reentrancy attacks.
ACTION awardnft(name receiver, uint8_t randomValue);	Yes	**Jack** Remark Function code body commented out, also this is the same commented out code from miningnft lines#173-188
ACTION miningstart(name receiver, uint32_t preset_id);	Yes	**Jack** Remark change line#300 to `check(suffix == "wam"_n)`
ACTION setprofitshr(name owner, uint64_t land_id, uint16_t profit_share);	Yes	**Jack** No issues.
ACTION agreeterms(name account, uint16_t terms_id, checksum256 terms_hash);	Yes	**Jack** No issues.
ACTION setlandnick(name owner, uint64_t land_id, string nickname);	Yes	**Jack** No issues.
ACTION filllandpot();	Yes	**Jack** Remark Line#660 magic numbers.
ACTION resetclaim();	No	**Jack** No issues.
ACTION addpntsluck(name user, optional<double> luck, optional<uint32_t> points);	No	**Jack** Remark Line#890 mixing camel and snake case.
ACTION addpoints(name user, uint32_t points);	No	**Jack** Remark just a wrapper for `addpntsluck`

ACTION redeempntnft(name user, uint64_t offer_id);	No	**Jack** Remark line#838 mixing camel and snake case.
ACTION setptsreward(uint64_t id, time_point_sec start, time_point_sec end, uint64_t template_id, uint32_t required);	No	**Jack** No issues.
ACTION delptsreward(uint64_t id);	No	**Jack** No issues.
ACTION setlvlreward(uint64_t level, string lvl_name, uint64_t template_id, uint32_t required);	No	**Jack** No issues.
ACTION redeemlvlnft(name user);	No	**Jack** Remark Line#852 mixing camel and snake case.
ACTION setmilestone(name user, uint8_t key, uint16_t value);	No	**Jack** Remark Line#867 mixing camel and snake case.
ACTION setlandregs(vector<landreg_item> landRegs);	No	**Jack** Minor (since this is dev only) no auth check and no check to see if the land reg already exists before emplacing a new row.

Notification Callback Handlers	**In ClickUp**	**Notes**
[[eosio::on_notify("alien.worlds::transfer")]] void ftransfer(name from, name to, asset quantity, string memo);	No	**Jack** No issues.
[[eosio::on_notify(DAC_TOKEN_CONTRACT_STR "::transfer")]] void dtransfer(name from, name to, asset quantity, string memo);	No	**Jack** Minor lines#398-402 use of deferred transactions is deprecated.
[[eosio::on_notify(NFT_CONTRACT_STR "::logtransfer")]] void logtransfer(name collection_name, name from, name to, vector<uint64_t> asset_ids, string memo);	No	**Jack** Remark Lines#705-712 contains commented out code.
[[eosio::on_notify(NFT_CONTRACT_STR "::logmint")]] void logmint(uint64_t asset_id, name authorized_minter, name collection_name, name schema_name, int32_t preset_id, name new_asset_owner, atomicdata::ATTRIBUTE_MAP immutable_data, atomicdata::ATTRIBUTE_MAP mutable_data, vector<asset> backed_tokens);	No	**Jack** Remark Lines#725-737 contains commented out code. Remark Line#738 contains magic numbers `1` and `3`.

Test Cases	In ClickUp	Notes
addpntsluck - with wrong auth - it should fail		**Jack** No issues.
addpntsluck - without accepting terms - with correct auth - should succeed		**Jack** No issues.
addpntsluck - without accepting terms - with correct auth - should not update table		**Jack** No issues.
addpntsluck - with accepting terms - with correct auth - should succeed		**Jack** **Remark** This should be a separate test case and not nested under “without accepting terms”.
addpntsluck - with accepting terms - with correct auth - should update table		**Jack** No issues.
addpntsluck - without accepting terms - later today - should succeed		**Jack** No issues.
addpntsluck - without accepting terms - later today - should update table		**Jack** No issues.
addpoints - with wrong auth - should fail		**Jack** No issues.
addpoints - with correct auth - should succeed		**Jack** No issues.
addpoints - with correct auth - should update table		**Jack** No issues.
addpoints - with correct auth - later today - should succeed		**Jack** No issues.
addpoints - with correct auth - later today - should update table		**Jack** No issues.
addpoints - with correct auth - tomorrow - should succeed		**Jack** No issues.
addpoints - with correct auth - tomorrow - should update table		**Jack** No issues.
addpoints - with correct auth - later tomorrow - should succeed		**Jack** No issues.
addpoints - with correct auth - later tomorrow - should update table		**Jack** No issues.
addpoints - with correct auth - next week - should succeed		**Jack** No issues.
addpoints - with correct auth - next week - should update table		**Jack** No issues.
addpoints - with correct auth - later next week - should succeed		**Jack** No issues.

addpoints - with correct auth - later next week - should update table		**Jack** No issues.
setptsreward - with wrong auth - should fail with auth error		**Jack** No issues.
setprtsreward - with valid auth - with invalid start and end dates - should fail with date error		**Jack** No issues.
setprtsreward - with valid auth - with dates in the past - should fail with date in past error		**Jack** No issues.
setprtsreward - with valid auth - with valid params - should succeed		**Jack** No issues.
setprtsreward - with valid auth - with valid params - should update the offers table		**Jack** No issues.
setprtsreward - with existing offer - should update the existing offer		**Jack** No issues.
setprtsreward - with existing offer - should update the offers table		**Jack** No issues.
delptsreward - with invalid auth - should fail with auth error		**Jack** No issues.
delptsreward - with valid auth - with no matching offer if - should fail with offer not found error		**Jack** No issues. Remark: Uses magic numbers.
delptsreward - with valid auth - with matching offer - should succeed to delete the offer		**Jack** No issues. Remark: Uses magic numbers.
delptsreward - with valid auth - with matching offer - should update the offers table		**Jack** No issues. Remark: Uses magic numbers.
redeempntnft - with wrong auth - should fail		**Jack** No issues.
redeempntnft - with no matching offer - should fail		**Jack** No issues.
redeempntnft - with no points record - should fail		**Jack** No issues.
redeempntnft - with insufficient points - should fail		**Jack** No issues.
redeempntnft - with expired offer - should fail		**Jack** No issues.
redeempntnft - with correct auth - should succeed		**Jack** No issues.
redeempntnft - with correct auth - should update table		**Jack** No issues. Remark: Uses magic numbers.

setlvlreward - with wrong auth - should fail with auth error		**Jack** No issues.
setlvlreward - with valid auth - with valid params - should succeed		**Jack** No issues.
setlvlreward - with valid auth - with valid params - should update the offers table		**Jack** No issues. Remark: Uses magic numbers.
setlvlreward - with valid auth - with existing offer - should update the existing offer		**Jack** No issues.
setlvlreward - with valid auth - with existing offer - should update the offers table		**Jack** No issues. Remark: Uses magic numbers.
redeemlvlnft - with wrong auth - should fail		**Jack** No issues.
redeemlvlnft - with correct auth - with no matching level to claim - should fail		**Jack** No issues.
redeemlvlnft - with correct auth - with no points record - should fail		**Jack** No issues.
redeemlvlnft - with correct auth - with an available level to claim - with insufficient points - should fail		**Jack** No issues.
redeemlvlnft - with correct auth - with an available level to claim - with enough points - should succeed		**Jack** No issues.
redeemlvlnft - with correct auth - with an available level to claim - with enough points - should update table		**Jack** No issues. Remark: Uses magic numbers.
redeemlvlnft - with correct auth - with an available level to claim - with enough points for next level - should succeed		**Jack** No issues.
redeemlvlnft - with correct auth - with an available level to claim - with enough points for next level - should update table		**Jack** Remark: Uses magic numbers.
setmilestone - with wrong auth - should fail with auth error		Phil Remark: Uses magic numbers
setmilestone - with correct auth - for user without user points row - should fail with row error		Phil Remark: Uses magic numbers
setmilestone - with correct auth - with existing userpoints record - without existing value for key - should create new key value pair		Phil Remark: Uses magic numbers
setmilestone - with correct auth - with existing userpoints record - without existing value for key - should create new key value pair		Phil Remark: Uses magic numbers
setmilestone - with correct auth - with existing userpoints record - without existing value for key - should update userpoints table		Phil Remark: Uses magic numbers

setmilestone - with correct auth - with existing userpoints record - with existing value - should update the value		Phil Remark: Uses magic numbers
setmilestone - with correct auth - with existing userpoints record - with existing value - should update userpoints table		Phil Remark: Uses magic numbers
setmilestone - with correct auth - with existing userpoints record - when adding another value - should succeed		Phil Remark: Uses magic numbers
setmilestone - with correct auth - with existing userpoints record - when adding another value - should create an addition key value pair		Phil Remark: Uses magic numbers
secondary indexes - second index should sort by totalpoints		Phil Remark: Uses magic numbers
secondary indexes - third index should sort by dailypoints grouped by day		Phil Remark: Uses magic numbers
secondary indexes - forth index should sort by weekly grouped by week		Phil Remark: Uses magic numbers
secondary indexes - fifth index should sort by total luck		Phil Remark: Uses magic numbers
addpntsluck with high values - should not overflow		Phil Remark: Uses magic numbers
addpntsluck with high values - should not allow too big luck values		Phil Remark: Uses magic numbers

Test coverage 8 / 38 = 21%

Tests Available For:

- addpntsluck
- addpoints
- setptsreward
- delptsreward
- redeempntnft
- setlvlreward
- redeemlvlnft
- setmilestone

Missing Tests For:

- addplanet
- updateplanet
- removeplanet
- setmap
- clearmap
- clearplanets
- setreserve
- clearstate
- deldeposit
- fixstake
- settag
- setavatar
- stake
- withdraw
- refund
- claim
- logclaim
- miningnft
- awardnft
- miningstart
- setprofitshr
- agreeterms
- setlandnick
- filllandpot
- resetclaim
- setlandregs
- ftransfer
- dtransfer
- logtransfer
- Logmint

No methodology definitively proves the absence of vulnerabilities. Following assessment and remediation, modifications to an application, its platform, network environment, and new threat vectors may result in new application security vulnerabilities.