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

Mining Contract Audit Notes - Revision

Source for this analysis is github:Alien-Worlds/alienworlds-contracts
Commit tag 1a656355b7c2c217898bde31c1929ab7029fe3b6
Files contracts/mining/mining.[ch]pp
Tests found in contracts/mining/mining.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.

No obvious signs of severe or critical security violations were detected in this audit, however there are a few examples of coding style or practice which could lead to issues in the future when changes are required to accommodate design changes.

The lack of Ricardian contract source is not necessarily a risk but is a lost opportunity for ensuring that the coded logic is congruent with the identified business needs.

Remark	Minor	Major	Critical
2	2	0	0

Participants

Primary Auditors were Phil Mesnier and Jack DiSalvatore, 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
- Authentication Check
- Apply Check
- Transfer Error Prompt
- Random Number Practice
- Rollback Attack

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 29 contract actions and 2 notification callback handlers. Of these only 12 are identified in the engineering reference document. As stated in the summary we found no significant security concerns, but we found instances of coding practices that could be troublesome in the future.

- Magic numbers. The use of literal numeric values should be avoided as they lose context and make maintenance updates more failure prone. Use “constexpr” defined symbols instead.
- “While” loops for iteration with multiple control paths. Using a for loop syntax is better in these cases as it encapsulates the initializer, exit criteria and incrementer, thus avoiding the risk of overlooking something in the future.
- ~~Some of the iterative actions, such as clearparams, are intended to operate on all members of a data collection, unless there are too many to act on during a single invocation. However there is no way for the client to know if the action operated on the entire set or it needs a subsequent call to complete. I think the action should use the action().send function to schedule a call to itself as many times as necessary to ensure the operation is applied to all elements of the set. These invocations are asynchronous so there is no worry about recursion.~~

- Previous remark deleted, due to invalid assumption. Subsequent action invocations must still be completed within the overall transaction CPU processing time budget. Second, as long as tables are defined in the contract ABI, then client code can determine aspects of the table including the current size.
- Commented-out code should not be archived. Particularly when using an advanced revision management tool like git, provisional or temporary code should be in a branch. Even during a single coding episode, using C-style comments are risky since they do not nest and using c++ style comments can make it difficult to disambiguate multiple commented sections and ordinary comments. Use conditional compilation instead. Framing a provisional block of code with #if 0.. #endif prevents compilation but is nestable and does not impact bona fide comments already in that code.

Contract actions from the source:

Action Signature	**In ClickUp**	**Notes**
setparam(uint64_t key, uint64_t value);	yes	No issues
testparam(name key);	yes	No issues
clearparams();	yes	Use a for loop rather than the while, and don't use magic numbers (50 in this case) ~~If total params > 50, perhaps another action request should be sent.~~ Suggestion is not valid.
setnfttarget(double target);	no	Currently commented out
mine(name miner, vector<char> nonce);	yes	Heavy use of magic numbers in code, risk for future maintainability. Excessive commenting out sections of obsolete or tentative code. Is_wam is now redundant, handling of "bot" access is inconsistent.
fill(name account, name planet_name);	yes	No issues
setbag(name account, vector<uint64_t> items);	yes	No issues
setland(name account, uint64_t land_id);	yes	lines#130-131 can be moved into the #ifndef IS_DEV block, since it is never used elsewhere line#140 looks like an old comment lines#152-153,156 remove commented out code
setlandnick(name account, uint64_t land_id, string nickname);	no	No issues
setnfts(name rarity, vector<uint32_t> template_ids);	yes	No issues

claimnfts(name miner);	no	No issues
fixmint(name miner, vector<uint32_t> oldtmps, vector<uint32_t> newtmps);	no	No issues
insertclaims(name miner, vector<uint32_t> tmps);	no	Only available when IS_DEV is defined
logmine(name miner, mining_data2 params, asset bounty, uint64_t land_id, name planet_name, name landowner, vector<uint64_t> bag_items, uint32_t offset);	no	NO-OP action, otherwise no issues
rand(uint8_t oracle_id, uint32_t time, uint32_t index, uint32_t max_index, vector<result> results);	no	No issues
clearnftwins();	no	Only available when IS_DEV is defined
logrand(name miner, uint64_t land_id, mining_data2 params, double rand1, double rand2, double rand3, uint32_t template_id);	no	NO-OP action, otherwise no issues
resetstate(name planet_name);	no	No issues
clearminers();	no	No issues
clearbags();	no	No issues
clearrandos();	no	See note for clearparams().
unlockbag(name miner);	no	No issues
clearnftmine(name owner, uint64_t asset_id);	no	This action triggers action “setassetdata” on the 3rd party “atomicassets” contract, referred to as NFT_CONTRACT.
delnft(name miner);	no	No issues
procrand();	no	Depends on orngwax::requestrand() action which is currently a NO-OP.
receiverand(uint64_t assoc_id, checksum256 random_value);	yes	Risky use of a deep while loop with multiple iteration criteria. Contains several blocks of code commented out with /*..*/ style comments. Such provisional code should not be pushed to version control, If that is unavoidable, use conditional compilation to avoid accidental comment nesting issues. Invoked from the Orng.wax contract but it is unclear how that contract is invoked.
receiverand2(uint64_t assoc_id, checksum256 random_value);	no	Risky use of a deep while loop with multiple iteration criteria. Appears unused.
addrando(name miner, uint64_t land_id, uint16_t luck);	no	Only available when IS_DEV is defined
[callback function] Logtransfer(name collection_name, name from, name to, vector<uint64_t> asset_ids, string memo);	no	No issues

[callback function] transfer(name from, name to, asset quantity, string memo);	no	No issues

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.