OBJECT COMPUTING

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

Alien Worlds dacescrow Contract Audit (EOS)

Source for this analysis is github: Alien-Worlds/contracts/dacescrow
Commit tag: 722e38267153bb53b37431aa86bb0959bd5485e7
Forked github: eosdac/eosdac-contracts
Forked Commit tag: f213b0f07c425912692c38e8b7e07ad99da9ddca
Files: contracts/dacescrow/dacescrow.[ch]pp
Tests: N/A
Ricardian contracts: contracts/dacescrow/dacescrow.contracts.md
Business requirements and other engineering artifacts:
https://eosdac.io/tools/smart-contracts-explained/
contracts/daccustodian/README.md

Executive Summary

No critical security violations were detected in this audit.

Remark	Minor	Major	Critical
0	0	0	0

Participants

Primary auditor is Jack DiSalvatore and Phil Mesnier provided 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

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 1 table, 8 actions and 2 helper functions.

Remarks

-

Minor

-

Major

-

Critical

Table/Singleton	Notes
using escrows_table = multi_index<"escrows"_n, escrow_info, indexed_by<"bysender"_n, const_mem_fun<escrow_info, uint64_t, &escrow_info::by_sender>>>;	No issues.

Action	Notes	Ricardian

		Contract
ACTION init(name sender, name receiver, name arb, time_point_sec expires, string memo, name ext_reference);	No delta.	Yes
[[eosio::on_notify("*::transfer")]] void transfer(name from, name to, asset quantity, string memo);	No issues.	Yes
ACTION approve(name key, name approver);	No delta.	Yes
ACTION disapprove(name key, name disapprover);	No delta.	Yes
ACTION refund(name key);	No delta.	Yes
ACTION dispute(name key);	No delta.	No
ACTION cancel(name key);	No delta.	Yes
ACTION clean();	No delta.	Yes

Helper Functions	Notes
void pay_arbitrator(const escrows_table::const_iterator esc_itr);	No delta.
void refund_arbitrator_pay(const escrows_table::const_iterator esc_itr);	No delta.

Test Case	Notes
N/A	

Test Coverage

The amount of actions tested divided by the number of total actions.
No tests were found.

0 / 8 = 0%

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.