OBJECT COMPUTING

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

Alien Worlds External Trillium Allocation Audit

Source for this analysis is github.com/Alien-Worlds/alienworlds-contracts
Commit tag: 1a656355b7c2c217898bde31c1929ab7029fe3b6
Files:

- scripts/landholder-airdrop_daemon.ts
- scripts/landholder-airdrop_recover.ts
- scripts/landholder-airdrop.js

Business requirements and other engineering artifacts: X

Executive Summary

No severe issues found.

Remark	Minor	Major	Critical
0	0	0	0

Participants

Primary auditor is 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:** 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

The examination looked at one TypeScript file.

Remarks
- Mixing snake and camel case.

Minor
- None.

Major
- None.

Critical
- None.

landholder-airdrop_daemon.ts

Function	**Notes**
const generateLandholdersList = async (date)	No issues.
const calculateAmounts = async (date, filterJSON: FilterAmount[] | undefined)	No issues.
const send = async (batch_size: number, dry_run: boolean)	No issues.
const run = async ()	No issues.

landholder-airdrop_recover.ts

Function	**Notes**
const send = (batch_size: number, filename: string, dry_run: boolean)	No issues.
const run = async ()	No issues.

landholder-airdrop.js

Function	**Notes**
const sleep = async (ms)	No issues.
const generate = async ()	No issues.
const filter = async (filename)	No issues.
const send = async (filename, batch_size, dry_run = false, land_allocation = null)	No issues. **Question**: line#227 is there a maximum amount of actions that can be packed into a single transaction?
const sendPrecalculatedList = async (filename, batch_size, dry_run = false)	No issues. **Question:** line#320 is there a max amount of actions that can be packed into a single transaction?
function get_excplicit_land_allocation()	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.