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

PackOpener Contract Audit Notes

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

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.

Remark	Minor	Major	Critical
4	4	0	0

Participants

Primary Auditor was 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 packopener 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

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

Minor issues

- The value SECONDS_PER_DAY is a #define which obscures a variable of the same name defined in config.hpp. That variable should be a constexpr.
- Should the various clear*() actions limit a maximum batch size like those in the mining contract in order to not exceed any blockchain enforced time limit?
- “Todo” comments found. Indicates unimplemented requirements.
- Iteration over a collection in a “while” loop with the index incrementer is within a conditional block of code. It is unclear that if the condition were false the loop could terminate.

Remarks

- Common use of magic numbers and string literals. Use of constexpr symbols would improve maintainability.
- A potential for numeric overflow exists in addcard() although the consequence is negligible.
- A memberwise assignment from a vector may be suitably replaced with a std::move().
- It may be better to replace the shifting of bytes from the calculated trx_id into signing_value with a call to memmove().

Contract actions from the source:

Action Signature	**In ClickUp**	**Notes**
addpack(name pack_name, symbol pack_symbol, extended_asset bonus_ft, bool active);	yes	A “todo” comment indicates the need to verify the supplied symbol exists.
editpack(name pack_name, symbol pack_symbol, extended_asset bonus_ft);	yes	A “todo” comment indicates the need to verify the supplied symbol exists.
activatepack(name pack_name, bool active);	yes	No issues.
delpack(name pack_name);	yes	No issues.
addcard(name pack_name, uint64_t card_id, vector<cardprob> card_probabilities);	yes	Remarks 1. use of magic numbers.

		2. Potential for numeric overflow, vector of uint16's summed into a uint16. 3. Since the vector is passed in by value, could a std::move() call be used rather than the memberwise assignment operator?
editcard(uint64_t card_id, vector<cardprob> card_probabilities);	yes	Same notes as for addcard()
delcard(uint64_t card_id);	yes	No issues.
addcrate(name crate_name, name type, vector<uint64_t> ids);	yes	Remarks 1. Use of magic numbers 2. Memberwise vector assignment
editcrate(name crate_name, vector<uint64_t> template_ids);	yes	Same notes as for addcrate().
addasset(name crate_name, uint64_t asset_id);	yes	No issues.
emptycrate(name crate_name);	yes	No issues.
delcrate(name crate_name);	yes	No issues.
clearcrates();	no	No issues.
clearpacks();	no	No issues.
clearcards();	no	No issues.
open(name account);	yes	Remark: 1. use of magic numbers. 2. The conversion to a uint64 from the 0 most significant bytes could be done in a 1-line for loop. Or a memmove().
receiverand(uint64_t assoc_id, checksum256 random_value);	yes	Minor: 1. while loop iterates through cards and increments inside a conditional.
claim(name account, name pack_name);	yes	Remark: 1. use of magic numbers.
delclaim(name account);	no	No issues.
logopen(name opener, name pack_name, vector<chosen_card> chosen_cards, asset ft_bonus);	no	It is an empty implementation.
[callback function] void transfer(name from, name to, asset quantity, string memo);	no	No issues.
[callback function] void tlmtransfer(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.