OBJECT COMPUTING

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

LandHolder Contract Audit Notes

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

Our review of the LandHolder Contract revealed no Major or Critical issues, but we did find 2 minor issues and a couple dozen remarks. There are a couple of anomalies to point out regarding this contract. First is the lack of a description in the business requirements and engineering artifacts (A.K.A. the "ClickUp") document. This, along with the lack of Ricardian contract source, makes the validation of the source code with respect to the intentions for it an impossible task. Second, the definitions and declarations for the LandHolder contract are combined into a single source file. Typically the declarations and definitions are maintained in "*.hpp" and "*.cpp" files respectively.

Remark	Minor	Major	Critical
28	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 landboost 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
- Reentrancy 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

Remarks:

- Many functions rely on literal strings and “magic” numbers, affecting maintainability and to some extent the size of the resulting WASM file. These are all indicated in the table below.
- The signature of the boost action has a separate declaration whether or not the IS_DEV compiler macro is defined. This could be refactored to be more maintainable.
- Both notification callback handlers, ftransfer and withdraw have blocks of code that could be refactored into a single check() macro.
- Test case “Openslot - without insufficient deposit should fail” appears to be misnamed. Should “without” really be “with” so the test should be “Openslot - with insufficient deposit should fail” or “without sufficient deposit should fail.”

Minor issues:

- Helper function start_calculate_land_ratings should verify the existence of the id value before using it on line 168.
- Helper function startpay should verify the existence of the landId value before using it on line 222.

Contract actions, helper functions, and tests from the source:

Action Signature	**In ClickUp**	**Notes**
ACTION setconfig(uint16_t numberOfLands, vector<name> skippedAccounts, asset startPayThreshold)	N/A	Phil No issues.
ACTION run(uint8_t batchSize)	N/A	Phil Remark: Relies on helper functions listed below
ACTION claimpay(name receiver)	N/A	Phil No issues.
ACTION openslot(const name &owner, const uint64_t land_id, const uint8_t number)	N/A	Phil No issues.

ACTION boost(const uint64_t land_id, const asset &amount, const name &payer, const time_point_sec &current_time, const uint32_t nonce)	N/A	Used if IS_DEV is defined Remark: The guarded portion of the signature should be confined to only the parts that are different
ACTION boost(const uint64_t land_id, const asset &amount, const name &payer)	N/A	Used if IS_DEV is not defined Remark: The guarded portion of the signature should be confined to only the parts that are different
ACTION setminboost(const name &owner, const uint64_t land_id, const asset &minboost)	N/A	Phil No issues.
Helper Functions		
void start_calculate_land_ratings()		Jack Minor: line#168 check to make sure `_landregs.begin()->id` exists before using it.
void calculate_land_ratings(uint8_t batchSize)		Jack No issues.
void startpay()		Jack Remark: line#219 use of magic numbers Minor: line#222 check to make sure `_landratings.begin()->landId` exists before using it.
void processBatch(uint8_t batchSize)		Jack No issues.
void nft_update_mutable_data(const name &owner, const uint64_t land_id, atomicdata::ATTRIBUTE_MAP attrs)		Jack No issues.
asset get_balance()		Jack No issues.
void reduce_deposit(const name &owner, const asset &quantity)		Jack No issues.
void add_deposit(const name &owner, const asset &quantity)		Jack No issues.
Notification Callback Handlers		

[[eosio::on_notify("alien.worlds::transfer")]] void ftransfer(const name &from, const name &to, const asset &quantity, const string &memo)		Jack
		Remark: You could refactor lines#391-394 into
		```
		check(quantity > ZERO_TRILIUM);
		```

[[eosio::on_notify(LANDBOOST_CONTRACT_STR "::withdraw")]] void withdraw(const name &user, const asset &quantity)		Jack
		Remark: You could refactor lines#403-405 into
		```
		check(quantity > ZERO_TRILIUM);
		```

Tests		

Configure NFTs - create land schema - should succeed		Jack No issues.
Configure NFTs - create land preset - should succeed		Jack Remark: magic number \`100\`
Configure NFTs - mint land nfts - should succeed		Jack Remark: line#138 magic number \`-1\`
Configure NFTs - mint land nfts - should have created the landregs entries		Jack Remark: line#164 magic number \`21\`
Adding additional attributes to nfts - should succeed		Jack No issues.
Adding additional attributes to nfts - should have added the mutable attributes		Jack Remark: line#178 magic number \`1000000\`
When changing the config - with incorrect auth - should fail with auth error		Jack No issues.
When changing the config with correct auth - should succeed		Jack No issues.
When changing the config with correct auth - should change the globals		Jack No issues.
Run pay batch - with wrong auth - should fail with auth error		Jack No issues.
Run pay batch - with correct auth - starting landrating calculation - should succeed		Jack
Run pay batch - with correct auth - starting landrating calculation - should update the globals		Jack Remark: line#243 magic number \`21\`.
Run pay batch - with correct auth - starting landrating calculation - should have populated the landratings table		Jack Remark lines#257-269 magic numbers.
Run pay batch - with correct auth - starting landrating calculation - batch process should have switched to WaitingForPayment		Jack No issues.
Run pay batch - with correct auth - starting landrating calculation - should update totalLandRating		Jack Remark line#278 magic numbers.
Run pay batch - with correct auth - with no pay to distribute - start pay batch should succeed		Jack No issues.
Run pay batch - with correct auth - with pay to distribute - should update the globals		Jack Remark: line#304 magic number 21.
Run pay batch - with correct auth - with pay to distribute - should update the globals2		Jack Remark: line#323 magic numbers.
Run pay batch - with correct auth - with pay to distribute - after start pay has succeeded first process		Jack No issues.

batch - should succeed		
Run pay batch - with correct auth - with pay to distribute - after start pay has succeeded first process batch - should update payments for landholders		Jack No issues.
Run pay batch - with correct auth - with pay to distribute - after start pay has succeeded first process batch - should update global fields		Jack No issues.
When user claims pay - with no pending pay - should fail with no found error for the receiver		Jack No issues.
When user claims pay - with a pending pay - with wrong auth - should fail with an auth error		Jack No issues.
When user claims pay - with a pending pay - with correct auth - should process payment		Jack No issues.
When user claims pay - with a pending pay - with correct auth - should remove pendingPay row		Jack No issues.
When user claims pay - with a pending pay - with correct auth - should update balance for claimer/receiver		Jack No issues.
When user claims pay - with a pending pay - with correct auth - should update globals pending pay amount 1		Jack Remark: instead of using hardcoded strings, it would be clearer to use variables
When processing the next batch - should succeed		Jack No issues.
Should update globals pending pay amount 2		Jack Remark: It's really hard to follow and validate the math when using hardcoded strings.
When processing the next batch - when processing the remaining batches with remaining records to process - should process all successfully		Jack No issues
When processing the next batch - when processing the remaining batches with remaining records to process - should update globals pending pay amount 3		Jack No issues.
When processing the next batch - when processing the remaining batches with remaining records to process - should have deleted all landrating table entries		Jack No issues.
When processing next cycle - should succeed to start next landrating calculation		Jack No issues.
When processing next cycle - should update the global table for the next landrating calculation		Jack No issues.
When processing next cycle - should have populated the landrating table		Jack No issues.
When processing next cycle - should update batch		Jack

process state to WaitingForPayment		No issues.
When processing next cycle - should have updated totalLandRating		Jack Remark: line#521 magic numbers.
When processing next cycle - without enough pay it should fail with error		Jack No issues.
When processing next cycle - with enough pay should succeed		No issues.
When processing next cycle - should succeed to process next pay first batch		No issues.
When processing next cycle - should update payments table		Remark: again it is hard to validate the math with hardcoded “magic” numbers.
When processing next cycle - then should succeed to process next pay batch		No issues.
When processing next cycle - should update all payments		Remark: again it is hard to validate the math with hardcoded “magic” numbers.
With accumulated pays - should claim accumulated pay		No issues.
With accumulated pays - should update all payments		Remark: hard to validate the math with hardcoded “magic” numbers.
With accumulated pays - should update the global table for the start next pay run		No issue.
With accumulated pays - balance should be slightly greater than the pendingPayout amount to avoid overpayment		No issue.
With accumulated pays - should update balance for claimer		No issues.
Claimpay from self permission - should update all payments		No issues.
Claimpay from self permission - should update all payments		Remark: again it is hard to validate the math with hardcoded “magic” numbers.
Openslot - without auth should fail		No issues.
Openslot - without deposit should fail		No issues.
Openslot - without insufficient deposit should fail		Remark: I think test case should actually be “**with** insufficient deposit should fail”
Openslot - skipping number should fail		No issues.
Openslot - with sufficient deposit should work		No issues.
Openslot - deposit should have been erased		No issues.

Openslot - should increase openslots of NFT		No issues.
Openslot - next level should work		Remark: use a for-loop to reduce duplicate code
Openslot - should fail when at level 15		No issues.
Openslot - should deduct the total cost from the deposit		No issues.
Boost - setminboost without proper auth should fail		No issues.
Boost - setminboost should succeed		No issues.
Boost - setminboost should update MinBoostAmount		No issues.
Boost - with invalid level should fail		No issues.
Boost - with level lower than MinBoostAmount should fail		No issues.
Boost - without enough deposited should fail		No issues.
Boost - with wrong symbol should fail		No issues.
Boost - with sufficiently high deposit should work		No issues.
Boost - should have increased landrating		No issues.
Boost - should have deducted amount from deposited balance		Remark: using variables instead of hardcoded numbers would make test cleaner
Boost - too many times should fail		No issues.
Boost - should have increased landrating even more		Remark: you should quantify how much the landrating increases by instead of using magic numbers.
Boost - 1 day later - it should work again		No issues.
Boost - 1 day later - should have increased landrating again		Remark: line#1233 magic number.
Boost - 1 day later - boosting too many times should fail again		Remark: looks like a redundant test, you already have a test for boosting too many times
Boost - boost for somebody else - without authorization from payer should fail		No issues.
Boost - boost for somebody else - without deposit should fail		No issues.
Boost - boost for somebody else - with deposit should work		No issues.
Boost - boost for somebody else - should have increased landrating for landowner		Remark: line#1324 uses magic number.
Boost - boost for somebody else - should have deducted the money from payer's deposit		Remark: easier to do math with variables instead of hardcoded values.

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.