OBJECT COMPUTING

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

Alien Worlds dacproposals Contract Audit (EOS)

Source for this analysis is github: Alien-Worlds/eosdac-contracts
Commit tag 722e38267153bb53b37431aa86bb0959bd5485e7
Files: contracts/dacproposals/dacproposals.[ch]pp
Test: contracts/dacproposals/dacproposals.test.ts
Ricardian contracts: contracts/dacproposals/dacproposals.contracts.md
Business requirements and other engineering artifacts: https://eosdac.io/tools/smart-contracts-explained/

Executive Summary

No issues of any severity were detected in this audit. Test coverage is good at almost 75% and Ricardian contracts are provided for most of the 19 contract actions.

Remark	**Minor**	**Major**	**Critical**
0	**0**	**0**	**0**

Participants

Primary auditors are Jack DiSalvatore and 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 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
- **Reentrancy Attack:** A reentrancy attack can occur when you create a function that makes an external call to another untrusted contract before it resolves any effects. If the attacker can control the untrusted contract, they can make a recursive call back to the original function, repeating interactions that would have otherwise not run after the effects were resolved.

  This simplest example is when a contract does internal accounting with a balance variable and exposes a withdraw function. If the vulnerable contract transfers funds before it sets the balance to zero, the attacker can recursively call the withdraw function repeatedly and drain the whole contract.

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 3 tables, 19 actions and 6 helper functions. Test coverage is a decent amount, roughly 74% of actions have at least one test scenario. The contract actions are documented on eosdac.io.

Remarks
- None.

Minor
- None.

Major
- None.

Critical
- None.

Table/Singleton	Notes
Proposal	No issues.
ProposalVote	No issues.
config	No issues.

Action	Notes	Ricardian Contract
createprop	No issues.	yes

voteprop	No issues.	yes
votepropfin	No issues.	no
delegatevote	No issues.	yes
delegatecat	No issues.	yes
undelegateca	No issues.	yes
arbapprove	No issues.	yes
arbdeny	No issues.	no
startwork	No issues.	yes
completework	No issues.	yes
finalize	No issues.	yes
cancelprop	No issues.	no
cancelwip	No issues.	no
dispute	No issues.	no
comment	No issues.	yes
updateconfjg	No issues.	yes
clearconfig	No issues.	no
clearexpprop	No issues.	yes
updpropvotes	Used as a helper as well as an action. Otherwise no issues.	no

Helper Functions	**Notes**
clearprop	No issues.
transferfunds	No issues.
check_proposal_can_start	No issues.
count_votes	No issues.
arbitrator_rule_on_proposal	No issues.
_voteprop	No issues.

Test Case	**Scenario**	**Notes**
createprop	Without valid permissions	
	With valid auth AND With invalid title	

	With valid auth AND With invalid summary	
	With valid auth AND With invalid pay symbol	
	With valid auth AND With no pay symbol	
	With valid auth AND With negative amount	
	With valid auth AND With no arbitrator	
	With valid auth AND with valid params	
	With valid auth AND with duplicate id	
	With valid auth AND with valid params as an additional proposal	
voteprop	Without valid auth	
	With valid auth AND with invalid proposal id	
	With valid auth AND proposal in pending approval state	
	with valid auth and proposal in work_in_progress state	
votepropfin		No tests.
delegatevote	Without valid auth	
	With valid auth AND with invalid proposal id	
	With valid auth AND delegating to self	
	With valid auth AND proposal in pending_apporval state AND delegate vote	
delegatecat	created proposal but still needing one vote for approval	
	created a proposal but still need 2 votes for approval for complex case	
	undelegate vote	
	with correct auth	
undelegateca		No tests.
arbapprove	with invalid prop	
	with valid prop id	
arbdeny	with invalid prop	
	with valid prop id	
startwork	With valid auth AND with invalid proposal id	
	proposal in pending_approval state AND with insufficient votes	

	with more denied than approved votes AND with insufficient votes	
	with enough votes to approve the proposal	
	Without valid auth	
	start work with enough votes	
	proposal not in pending_approval state	
	proposal with short expiry	
completework	without existing proposal	
	with incorrect auth	
	proposal in pending approval state	
	proposal in work_in_progress state	
finalize	without valid auth	No test body
	with valid auth	Commented out
	with invalid proposal id	
	proposal not in pending_finalize state	
	proposal in pending_finalize state	
cancelprop	tests	
cancelwip	tests	
dispute		No tests.
comment	Without valid auth	
	With valid auth AND with invalid proposal id	
	With custodian only auth	
updateconfjg	Without valid auth	
	With valid auth	
clearconfig		No tests.
clearexpprop		No tests.
updpropvotes		No tests directly called but it is tested indirectly

Test coverage = (number of tested actions) / (number of actions)
14 / 19 = ~74%

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.