SecurityDAO

CW-Unity-Prop Audit

CW Unity Proposal CosmWasm Contract Audit

Prepared by: Logan Cerkovnik
Date: March 29th, 2022

TVL : **$>=120M USD equivalent**

... + more about threat model

Outline

Document Revision History

Version	Modification	Date	Author
0.1	**Created**	**3/29/2022**	**Logan Cerkovnik**

Contact

Contact	Organization	Email
Logan Cerkovnik	Security DAO	logan@secdao.xyz
Paul Wagner	Security DAO	paul@secdao.xyz
Barton Rhodes	Security DAO, DAO DAO	barton@secdao.xyz

Executive Overview

Audit Summary

Security Dao worked on an engagement with the Juno Community and Juno Core Dev group from 3/25/2022 through 3/30/2022 to conduct a security assessment of the unity proposal contracts to ensure security, trust, and communication for prop 18.

The security engineers involved with the audit are security and blockchain smart contract security experts with advanced knowledge of smart contract exploits.

The purpose of this audit is to achieve the following:

- **Ensure** that smart contract functions work as intended
- **Identify** potential security issues with the smart contracts

In summary, Security Dao identified impactful improvements to reduce the likelihood and scope of risks, which were addressed by the WasmSwap team.

The **primary ones** are as follows:

- Misuse of entry_point macro with sudo function

External threats such as intercontract functions and calls should be validated for expected logic and state and are not covered within the scope of this audit. Only direct rpc contract interaction is considered here not any UI components or frontend wasm interactions are excluded.

Test Approach and Methodology

Security DAO performed a combination of manual review of the code and automated security testing.

The following phases were used throughout the audit:

- Research into the architecture, purpose, and use of the platform
- Manual code review and walkthrough

- Manual Assessment of the use and safety for critical rust variables and functions in scope to identify any contracts logic related vulnerability
- Fuzz Testing (securitydao fuzzing tool)
- Check Test Coverage (cargo tarpaulin)

Contract	% coverage	Lines Covered
cw20-unity-prop	99.35%	457/460

- Scanning of Rust files for vulnerabilities (cargo audit)

Contract	Dependency	Version	Warning
cw20-unity-prop	const-oid	0.6.0	yanked
cw20-unity-prop	crypto-bigint	0.2.2	yanked

Risk Methodology

Risk Likelihood and impact scales 1 through 5 where 5 is the most severe

Risk Likelihood Scale

1 low	2 unlikely	3 possible	4 likely to happen	5 high

least severe most severe

A low likelihood risk indicates that the likelihood of attack is low because of obscurity or requiring additional exploits to utilize, a possible attack is one that is possible but not an attack method commonly seen in the wild or well-known, and high risk likelihood represents an exploit extremely likely to be used, readily apparent, or commonly been used in the past against similar systems

Risk Impact Scale

1 low	2 limited	3 Impactful	4 critical	5 severe

least severe most severe

In the context of smart contracts, a low risk impact might be something associated with limited scope or a preventive best practice, an impactful risk may result in large loss of funds but not in a systematic way, and a severe risk impact could result in substantial loss of funds in a systematic way.

Scope

Cosmwasm Smart Contracts

The primary target for the audit is cw-unity-prop crate. User interface and cross contract messages are considered out of scope for this work.

- **Repo:** `Private:` https://github.com/CosmosContracts/cw-unity-prop
- **Commit hash: 54c1fb6fa5157bcb76b8e9b9736acc6c59a99575**

Action Plan

Sudo function's purpose and use needs to be addressed and fixed in some way before deploying the contract to mainnet with appropriate fixes for authorization and validation of the sudo entry point functions.

Low Level of Effort to Fix and Low Impact

- **(SEC - 13)** Upgrade Yanked Dependencies

High Level of Effort to Fix and High Impact

- **(SEC - 15) Misuse of entry_point macro with sudo function**

Assessment Summary and Findings Overview

Findings and Tech Details

(SEC - 13) Yanked Dependencies

Severity Low / Impact Low

Description

Dependencies for const-oid version 0.6.0 and crypto-bigint 0.2.2 are both yanked. This is a cosmwasm-std 1.0.0beta dependency issue.

Code Location

https://github.com/CosmosContracts/cw-unity-prop/blob/main/Cargo.lock

Risk Level

The risk likelihood is low and the impact is low

Recommendation

Upgrade cosmwasm-std to latest version 1.0.0beta7

Remediation Plan

(SEC - 15) Misuse of entry_point macro with sudo function

High Severity / High Impact

Description

The cosmwasm_std::entry_point macro is misused with the sudo function. Officially only 3 functions (instantiate, execute, and query) are supported according to cosmwasm_std source code and docs. Use of a non-standard entrypoint macro and calling it “sudo” could introduce security vulnerabilities and will reduce trust from community members reviewing source and possibly in wasm/wat on-chain.

execute_burn , execute_send, and execute_send_all functions have no authorization in them also so they cannot be dropped into the execute entrypoint as is. Currently those functions will be unusable in the current contract because they are called outside of the execute entrypoint.

The cw-multi-test crate does have a mock_app.wasm_sudo(integration_tests.rs line line 168), but that is only for testing behavior with error handling in the test environment and no authorization functionality.

Code Location
https://github.com/CosmosContracts/cw-unity-prop/blob/main/src/contract.rs line 141

```
#[cfg_attr(not(feature = "library"), entry_point)]
pub fn sudo(deps: DepsMut, env: Env, msg: SudoMsg) -> Result<Response,
ContractError> {
    match msg {
        SudoMsg::ExecuteBurn {} => execute_burn(deps, env),
        SudoMsg::ExecuteSend { recipient, amount } => execute_send(deps,
env, recipient, amount),
        SudoMsg::ExecuteSendAll { recipient } => execute_send_all(deps,
env, recipient),
    }
}
```

Risk Level

Risk likelihood is high and severity is high

Recommendation

Either this code should not be included, included as a test helper function, or integrated with the execute function. Because the function is named sudo it will likely be targeted by anyone looking at .wasm/wat code on chain if it has any functionality. Another concern is that the sudo function is supposed to perform some essential authorization functionality that isn't integrated currently right now since it's not inside the execute function. The functions called in the sudo entrypoint (execute_burn , execute_send, and execute_send_all) need security features added around address verification, time expiration validation etc.)

Remediation Plan