SecurityDAO

WasmSwap audit

Uniswap v1 + Synthetix in CosmWasm

Prepared by: Logan Cerkovnik
Date: April 20th 8th, 2022

Largest pool: **$25M equivalent**

... + more about threat model

Outline

Document Revision History

Version	Modification	Date	Author
0.1	Created	3/08/2022	Logan Cerkovnik
0.2	Revised	4/08/2022	Logan Cerkovnik
0.3	Revised	4/20/2022	Logan Cerkovnk

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

Junø Dev and InterWasm Foundation engaged with Security Dao from 2/20/2022 through 3/15/2022 to conduct a security assessment of smart contracts supporting WasmSwap.

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.

Principal takeaways:

- Absence of max slippage parameter
- Absence of ownership module for pause or upgrade of contracts
- Absence of standardizing logic to parse addresses and token string to lowercase. s
- Failure to validate that tokens aren't the same token or that tokens are from allowlist of tokens
- Failure to validate min and max range for rates and duration variables
- Hook failures can freeze contract through DOS on hook upgrade resulting in loss of funds
- Loss of funds during instantiation of contract with a zero token pool for a token pair or incorrect initial parameters including use of native tokens

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 code reviews 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)
  - Wasmswap: 91.41% coverage, 787/861 lines covered
  - stake-cw20-external-rewards: 99.12% coverage, 1014/1023 lines covered
- Scanning of Rust files for vulnerabilities (cargo audit)
  - No known vulnerabilities were identified in 3rd party dependencies

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 subject matter for the audit is WasmSwap. Additional targets are the staking rewards module and the staking hooks module. Sandwich attacks or other types of trading vulnerabilities inherent in uniswap-v1 are out of scope. The user interface and wasm bridge are also out of scope.

- **Repo:**
  https://github.com/securitydao/WasmSwap-contracts/blob/main/src/contract.rs
  https://github.com/securitydao/wasmswap-contracts/commit/27cf54fd3f3e4d098bfc52e4dbc6ac883b24ebc2

  Public: https://github.com/securitydao/wasmswap-contracts
- **Commit hash:**
- **Repo:**
  https://github.com/DA0-DA0/dao-contracts/blob/external-staking-rewards/contracts/stake-cw20-external-rewards/src/contract.rs

  https://github.com/DA0-DA0/dao-contracts/blob/1ca9963393830db35bd31e86d6b864c5278fbdc4/contracts/stake-cw20-external-rewards/src/contract.rs

- **Commit hash:**
- **Repo:** https://github.com/DA0-DA0/dao-contracts/blob/staking-contract-hooks/contracts/stake-cw20/src/hooks.rs#L42
- **Commit hash:**

Action Plan

Low Level of Effort to Fix and High Impact

- **(SEC - 02)** Validate and parse addresses **to** lowercase
- **(SEC - 03)** Validate, lowercase and allowlist tokens
- **(SEC - 08)** Max token swaps can crash contract on multi operations
- **(SEC - 09)** Numeric overflow in staking rewards

High Level of Effort to Fix and High Impact

- **(SEC - 01)** Missing Upgradeability / No Patching Features
- **(SEC - 04)** Implement max slippage parameter

Low Level of Effort to Fix and Low Impact

- **(SEC - 07)** Validate ranges for duration and rate variables
- **(SEC - 10)** Validate that token pairs are not the same token

High Level of Effort to Fix and Low Impact

- **(SEC - 05)** Loss of funds with contract instantiation with 0 funds
- **(SEC - 06)** Hook Failure Denial of Service

Assessment Summary and Findings Overview

Findings and Tech Details

(SEC - 01) Immutable Contract Deployment Approach

Medium Severity and Impact

Description

WasmSwap offers no way for upgrading, pausing or canceling contracts easily without individual contributors removing capital. There is an ownership module, but with WasmSwap developers recommending against its use makes it difficult to rely on.

Code Location

https://github.com/securitydao/WasmSwap-contracts/blob/main/src/contract.rs

Risk Level

The risk likelihood is 3 and impact is 4.

Recommendation

Add UI / governance integration to allow for a group to pause, remove, or upgrade smart contracts and for official communication channels.

Remediation Plan

Version 1.1 will likely try to implement authorization with implementing a dao multisig used to control upgrade and pause features. For coordination in response to a cybersecurity incident, an official communication channel will be established such as website and/or discord or element channel.

(SEC - 02) Validate and lowercase addresses

Medium Severity / High Impact

Description

Addresses should be set to lowercase and validated before use

Validation of addresses is discussed in this PR:
https://github.com/DA0-DA0/dao-contracts/pull/186

Code Location

https://github.com/securitydao/WasmSwap-contracts/blob/main/src/contract.rs

https://github.com/DA0-DA0/dao-contracts/commit/08cdc986b525d4a82297e0caf6bc8032878a8a63

Risk Level

Risk likelihood is **4**: likely to happen and severity is **2**: limited

Tokens can be added to a treasury which are invalid. Admin address for the staking contracts can also be to invalid addresses.

Recommendation

Add validation and lowercasing to addresses and remove usage of Addr in message types where possible. Remove strings where a custom type may be used instead.

Remediation Plan

Improved addr validation for external rewards contract was added in commit c8bee72439b94387a4fc2fdaa822794dda3afc73

(SEC - 03) Validate, lowercase and allowlist tokens

Medium Severity and Impact

Description

Tokens should be set to lowercase and validated before use. Additionally , tokens should be checked against a allowlist of acceptable tokens. Examples of attacks could be adding contracts with UsDt vs usdt .

Code Location

https://github.com/securitydao/WasmSwap-contracts/blob/main/src/contract.rs

Risk Level: risk likelihood is 3 and the impact is 3.

Recommendation

Add validation and lowercasing at a minimum to token inputs. A allowlist check would also be helpful to prevent using namespace squatting in contracts.

Remediation Plan

The user interface addresses the token validation and allowlisting needs currently. In version 1.1 further validation and token lowercasing should be implemented in the smart contracts. Use of a dao or multisig for controlling a allowlist tokens should also be implemented in the future with the administration functions for upgrading and pausing contracts

(SEC - 04) Implement max slippage parameter

Low Severity / High Impact

Description

Certain AMM security breaches and front running can often be at least partially mitigated by allowing specification of max slippage parameter that cancels trades if slippage is greater than specified threshold specified by trader.

Code Location

https://github.com/securitydao/WasmSwap-contracts/blob/main/src/contract.rs

Risk Level: the risk likelihood is low and the impact is medium severity.

Recommendation

Add a max slippage parameter to system

Remediation Plan

This has already been implemented as the min_token parameter in line 579 of contract.rs which is a discrete alternative to max slippage float where the user may specify the minimum amount of 2nd token to receive in a given swap.

(SEC - 05) Loss of funds with contract instantiation with 0 funds

Low Likelihood and Severity

Description

Contracts can become stuck when initialized with 0 funds or other incorrect parameters. This bug was originally discovered by @ben2x4 from the wasmswap team .

Code Location

https://github.com/securitydao/WasmSwap-contracts/blob/main/src/contract.rs

Risk Level: the risk likelihood is 3 and the severity is 2.

Recommendation

Fix and validate input on contract instantiation for new liquidity pools.

Remediation Plan

The Wasmswap team acknowledges that incorrect contract instantiation can result in loss of funds currently and accepts that contract instantiation may require expert knowledge. Version 1.1 may include further validation of contract instantiation.

(SEC - 06) Hook Failure Denial of Service on Upgrade

Medium Likelihood / Low Severity

Description

Contracts can become stuck in a repeated failure loop by updating a hook to always fail. Whomever controls hooks can freeze staking contract by adding a bad hook on update. This bug was originally discovered by @ben2x4 from the wasmswap team .

Code Location

https://github.com/DAO-DAO/dao-contracts/blob/staking-contract-hooks/contracts/stake-cw20/src/hooks.rs#L42

https://github.com/DAO-DAO/dao-contracts/blob/external-staking-rewards/contracts/stake-cw20-external-rewards/src/contract.rs

```
pub fn unstake_hook_msgs(
    storage: &dyn Storage,
    addr: Addr,
    amount: Uint128,
) -> StdResult<Vec<SubMsg>> {
    let msg = to_binary(&StakeChangedExecuteMsg::StakeChangeHook(
        StakeChangedHookMsg::Unstake { addr, amount },
    ))?;
    HOOKS.prepare_hooks(storage, |a| {
        let execute = WasmMsg::Execute {
            contract_addr: a.to_string(),
            msg: msg.clone(),
            funds: vec![],
        };
        Ok(SubMsg::new(execute))
    })
}
```

Risk Level: the risk likelihood is 2 and the severity is 3.

Recommendation

Add backoff or max failures to hook logic before reverting to earlier state..

Remediation Plan

WasmSwap team acknowledges that contract owners can break their own rewards and lose their staking reward funds by upgrading a contract incorrectly. Future versions 1.1 may try to improve this with better validation with hook upgrades or backoff on repeated failures of hooks.

(SEC - 07) Validate ranges for duration and rate variables

Low Severity / Low Liklihood

Description

Ranges and durations do not have validation for acceptable ranges

Code Location

```
pub fn unstake_hook_msgs(
    storage: &dyn Storage,
    addr: Addr,
    amount: Uint128,
) -> StdResult<Vec<SubMsg>> {
    let msg = to_binary(&StakeChangedExecuteMsg::StakeChangeHook(
        StakeChangedHookMsg::Unstake { addr, amount },
    ))?;
    HOOKS.prepare_hooks(storage, |a| {
        let execute = WasmMsg::Execute {
            contract_addr: a.to_string(),
            msg: msg.clone(),
            funds: vec![],
        };
        Ok(SubMsg::new(execute))
    })
}
```

https://github.com/DAO-DAO/dao-contracts/blob/staking-contract-hooks/contracts/stake-cw20/src/hooks.rs#L42

https://github.com/DAO-DAO/dao-contracts/blob/external-staking-rewards/contracts/stake-cw20-external-rewards/src/contract.rs

Risk Level: the risk likelihood is low and impact is low.

Recommendation

Duration min should be validated to be greater than 0 and with some max timeout. Rates should be greater than 0 and capped at a reasonable maximum.

Remediation Plan

Version 1.1 will implement validation on rates and duration.

Fixed in commit to main fcfcd970c94391057c9291d680e92c3b38c6e59f

(SEC - 08) Max token swaps can crash contract on mult operation

Low Likelihood / Medium Impact

Description

A large swap of tokens results in the token liquidity calculation squaring and resulting in a numeric overflow.

Code Location

```
let native_amount = amount
    .checked_mul(token1.reserve)
    .map_err(StdError::overflow)?
    .checked_div(token.total_supply)
    .map_err(StdError::divide_by_zero)?;
```

https://github.com/securitydao/WasmSwap-contracts/blob/main/src/contract.rs **line 364**

Risk Level: the risk likelihood is **1**: low and the impact is **3**: impactful.

Recommendation

Either cast to float and divide down by 10^6 losing precision before mult operations and mult by 10^6 after or cap transactions at a maximum allowed token value. This may be less of a concern with Juno due to the size of current Juno in circulation, but other parts tokens may have larger liquidity that could allow this to be used to crash a contract or some other malicious purpose.

Other WasmSwap v1 implementations such as synthetix divide by a large number to reduce the chance of overflows and then multiply by that same large number to prevent this behavior.

Remediation Plan

Fixed in commit to main fcfcd970c94391057c9291d680e92c3b38c6e59f

(SEC - 09) **numeric overflow in staking rewards when rewards become too large**

Low Severity / Medium Likelihood

Description

Rewards increase unboundedly as a function of increasing blocktime and this results a numeric overflow in

Code Location

https://github.com/DA0-DA0/dao-contracts/blob/external-staking-rewards/contracts/stake-cw20-external-rewards/src/contract.rs
line 247

```
pub fn get_reward_per_token(deps: Deps, env: &Env,
staking_contract: &Addr) -> StdResult<Uint256> {
let reward_config = REWARD_CONFIG.load(deps.storage)?;
   let total_staked = get_total_staked(deps, staking_contract)?;
   let current_block = min(env.block.height,
reward_config.periodFinish);
   let last_update_block =
LAST_UPDATE_BLOCK.load(deps.storage).unwrap_or_default();
   let prev_reward_per_token =
REWARD_PER_TOKEN.load(deps.storage).unwrap_or_default();
   let additional_reward_per_token = if total_staked ==
Uint128::zero() {
       Uint256::zero()
   } else {
       let numerator =
Uint256::from(reward_config.rewardRate.checked_mul(Uint128::from(cu
rrent_block - last_update_block))?).checked_mul(scale_factor())?;
```

```
        let denominator = Uint256::from(total_staked);
        numerator.checked_div(denominator)?
    };
```

Risk Level: the risk likelihood is low and the impact is medium.

Recommendation

Either cast to float and divide down by 10^16 losing precision before mult operations and mult by 10^16 after or cap transactions at a maximum allowed token value. This bug was originally discovered by @ben2x4 from the wasmswap team .

Remediation Plan

Fixed in commit to main fcfcd970c94391057c9291d680e92c3b38c6e59f

Using cosmwasm uint256 in type cast before and after this operation may also help reduce overflow errors. This may be less of a concern with Juno due to the size of current Juno in circulation, but other parts tokens may have larger liquidity that could allow this to be used to crash a contract or some other malicious purpose.

The contract failing after a long period of time may not be an issue if the staking reward contract owner may remove the rewards if it fails after a long period of time.

(SEC - 10) Validate that token pairs are not the same token

Low Likelihood and Severity

Description: Liquidity pools can be created with the same token. This undesirable behavior can sometimes lead to security vulnerabilities in amm systems where more tokens can be withdrawn than entered.

Code Location

https://github.com/securitydao/WasmSwap-contracts/blob/main/src/contract.rs

Lines 25:64

```
    let token1 = Token {
        reserve: Uint128(0),
        denom: msg.native_denom.to_lowercase(),
        address: None,
    };

    TOKEN1.save(deps.storage, &token1)?;

    let token2 = Token {
        address: Some(msg.token_address),
        denom: msg.token_denom.to_lowercase(),
        reserve: Uint128(0),
    };

    TOKEN2.save(deps.storage, &token2)?;
```

Risk Level: the risk likelihood is low and the severity is low.

Recommendation Plan

1. Validate that the two swapped tokens are not identical.
2. Add an integration test that **demonstrates ability to catch when you try to swap two identical tokens**

Remediation Plan

Add code with raising custom error for duplicate token usage in version 1.1

(SEC - 11) Potential Roundoff or Slippage Error

Low Severity and Likelihood

Description:

When swapping $NETA and $UST tokens there is sometimes a larger than expected net slippage. This is due to a scaling calculation bug. This bug was originally discovered by @ben2x4 from the wasmswap team .

Code Location

https://github.com/securitydao/WasmSwap-contracts/blob/main/src/contract.rs

Risk Level: the risk likelihood and severity is low.

Recommendation Plan

Add cw_multi_test framework example for testing against currencies of different scales and multiswap hop slippage. Consider scaling transaction values to mitigate round off errors. Introduce more safe math around rates and allow upscaling to uint512 where needed.

Remediation Plan

Fixed in commit to main fcfcd970c94391057c9291d680e92c3b38c6e59f

(SEC - 12) Failure at End of Reward Period

Low Likelihood and Severity

Description:

The contract fails at the end of the reward period in a local testnet environment. This bug was originally discovered by @ben2x4 from the wasmswap team .

Code Location

https://github.com/DAO-DAO/dao-contracts/commit/08cdc986b525d4a82297e0caf6bc8032878a8a63

Risk Level: the risk likelihood and severity are low.

Recommendation Plan

Fix logic around end of reward period

Remediation Plan

Fixed in commit to main fcfcd970c94391057c9291d680e92c3b38c6e59f

(SEC - 13) execute_stake and execute_unstake are same functions

Low Likelihood and Severity

Description:

execute_stake and execute_unstake are the same functions that only serve to update the rewards. It would be better to update the rewards immediately after the stake has changed.

Code Location

https://github.com/DAO-DAO/dao-contracts/blob/fcfcd970c94391057c9291d680e92c3b38c6e59f/contracts/stake-cw20-external-rewards/src/contract.rs

```
pub fn execute_stake_changed(
    deps: DepsMut,
    env: Env,
    info: MessageInfo,
    msg: StakeChangedHookMsg,
) -> Result<Response<Empty>, ContractError> {
    let config = CONFIG.load(deps.storage)?;
    if info.sender != config.staking_contract {
        return Err(ContractError::Unauthorized {});
    };
    match msg {
```

```
        StakeChangedHookMsg::Stake { addr, .. } => execute_stake(deps,
env, addr),
        StakeChangedHookMsg::Unstake { addr, .. } =>
execute_unstake(deps, env, addr),
    }
}

pub fn execute_stake(
    mut deps: DepsMut,
    env: Env,
    addr: Addr,
) -> Result<Response<Empty>, ContractError> {
    update_rewards(&mut deps, &env, &addr)?;
    Ok(Response::new().add_attribute("action", "stake"))
}

pub fn execute_unstake(
    mut deps: DepsMut,
    env: Env,
    addr: Addr,
) -> Result<Response<Empty>, ContractError> {
    update_rewards(&mut deps, &env, &addr)?;
    Ok(Response::new().add_attribute("action", "unstake"))
}
```

Risk Level: the risk likelihood is low and severity is low.

Recommendation Plan

Update the execute_stake / unstake or make reward updates happen in the same call that does the stake change

Remediation Plan

This behavior is intentionally only used for attribute response tagging and triggering update rewards. Wasmswap team will not change this but may add some comments for clarification.

(SEC - 14) Scaling operation breaks

High Likelihood / Low Severity

Description:

A scale factor was used before a divide operation for accrued rewards then unscaled this before calculating the rewards earned by a user. The unscaling operation was done too early and the resulting value was lower than the scale factor resulting in zero pending rewards.

Code Location

https://github.com/DAO-DAO/dao-contracts/blob/8ad861a126eddd772bd108ae1cefad7daa739637/contracts/stake-cw20-external-rewards/src/contract.r

Risk Level: the risk likelihood is high and the severity is low.

Recommendation Plan

Add some validation logic and raise an error if the reward rate is less than 1 to avoid locking funds

Remediation Plan

Fixed in commit 65dc9ea862df6a5a747b5e43210b11265704a9b8 with pr https://github.com/DAO-DAO/dao-contracts/pull/253

(SEC - 15) Validate that Reward Rate is greater than 1

Medium Likelihood / Low Severity

Description:

There is a bug in the contract where if the reward configuration results in rewards less than 1 then the reward rate is rounded down to zero and funds are lost/locked.

This bug was originally discovered by @ben2x4 from the wasmswap team .

Code Location

https://github.com/DAO-DAO/dao-contracts/blob/8ad861a126eddd772bd108ae1cefad7daa739637/contracts/stake-cw20-external-rewards/src/contract.r

Risk Level: the risk likelihood is high and the severity is low.

Recommendation Plan

Add some validation logic and raise an error if the reward rate is less than 1 to avoid locking funds

Remediation Plan

Fixed in commit 8ad861a126eddd772bd108ae1cefad7daa739637 with pr https://github.com/DAO-DAO/dao-contracts/pull/259