// HALBORN

Sienna.Network AMM Protocol (Updated code)

CosmWasm Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **February 7th, 2022 - February 16th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	02/07/2022	Alexis Fabre
0.2	Document Updates	02/15/2022	Alexis Fabre
0.3	Draft Version	02/16/2022	Luis Quispe Gonzales
0.4	Draft Review	02/17/2022	Gabi Urrutia
1.0	Remediation Plan	02/18/2022	Alexis Fabre
1.1	Remediation Plan Review	02/18/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com
Luis Quispe Gonzales	Halborn	Luis.QuispeGonzales@halborn.com
Alexis Fabre	Halborn	Alexis.Fabre@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Sienna.Network engaged Halborn to conduct a security audit on their smart contracts beginning on February 7th, 2022 and ending on February 16th, 2022. The security assessment was scoped to the smart contracts provided to the Halborn team.

1.2 AUDIT SUMMARY

The team at Halborn was provided 1 week for the engagement and assigned a full-time security engineer to audit the security of the smart contract. The security engineer is a blockchain and smart-contract security expert with advanced penetration testing, smart-contract hacking, and deep knowledge of multiple blockchain protocols.

The purpose of this audit is to:

- Ensure that smart contract functions operate as intended
- Identify potential security issues with the smart contracts
- Review codebase changes since last audit

In summary, Halborn identified some improvements to reduce the likelihood and impacts of the risks, which were accepted by Sienna team. The main ones are:

- Ensure that the factory cannot create a pair of the same token by detecting same addresses with lower/upper case.
- Take fees into account in the computation of an exchange swap.
- Protect the users from data leakage by applying best practices relative to Secret Network security model.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual review of the code and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of the smart contract audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of smart contracts and can quickly identify items that do not follow security best practices. The following phases and associated tools were used throughout the term of the audit:

- Research into architecture, purpose, and use of the platform.
- Manual code read and walkthrough.
- Manual assessment of use and safety for the critical Rust variables and functions in scope to identify any contracts logic related vulnerability.
- Fuzz testing (Halborn custom fuzzing tool)
- Checking the test coverage (cargo tarpaulin)
- Scanning of Rust files for vulnerabilities (cargo audit)

RISK METHODOLOGY:

Vulnerabilities or issues observed by Halborn are ranked based on the risk assessment methodology by measuring the **LIKELIHOOD** of a security incident and the **IMPACT** should an incident occur. This framework works for communicating the characteristics and impacts of technology vulnerabilities. The quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that were used to generate the Risk scores. For every vulnerability, a risk level will be calculated on a scale of 5 to 1 with 5 being the highest likelihood or impact.

RISK SCALE - LIKELIHOOD

5 - Almost certain an incident will occur.
4 - High probability of an incident occurring.

3 - Potential of a security incident in the long term.
2 - Low probability of an incident occurring.
1 - Very unlikely issue will cause an incident.

RISK SCALE - IMPACT

5 - May cause devastating and unrecoverable impact or loss.
4 - May cause a significant level of impact or loss.
3 - May cause a partial impact or loss to many.
2 - May cause temporary impact or loss.
1 - May cause minimal or un-noticeable impact.

The risk level is then calculated using a sum of these two values, creating a value of 10 to 1 with 10 being the highest level of security risk.

10 - CRITICAL
9 - 8 - HIGH
7 - 6 - MEDIUM
5 - 4 - LOW
3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

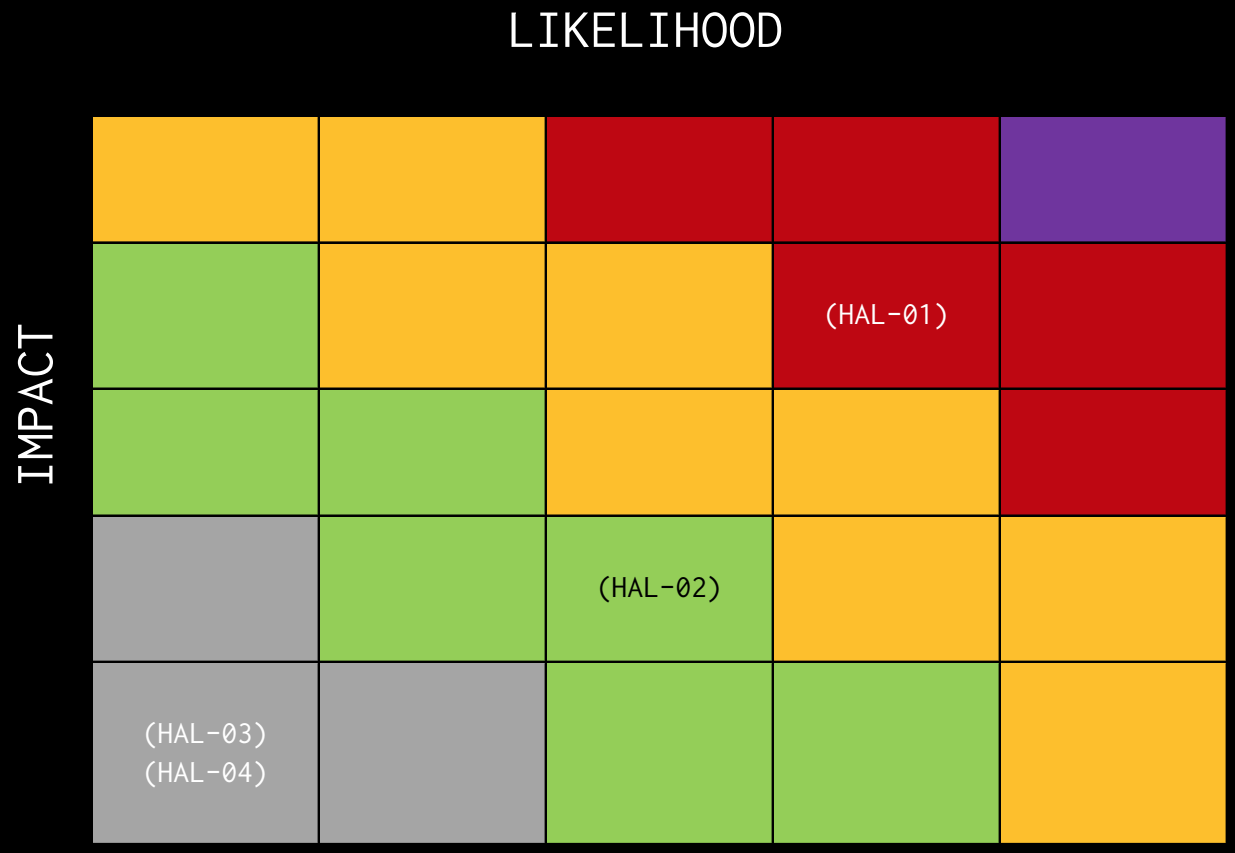
1. CosmWasm Smart Contracts
   (a) Repository: https://git.sienna.network/SiennaNetwork/contracts
   (b) Commit ID: 1cea0c7989825ee8ca2279942dedf4ca29a0f880
   (c) Contracts in scope:
      i. exchange
      ii. factory

It is worth noting that the results of this audit are a complement to the information provided in a previous report for the security audit performed to the codebase with commit id 13ce1ac9728e16b3d79d64caca603fe029882371.

Out-of-scope: External libraries and financial related attacks

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	1	0	1	2



EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) POSSIBILITY TO CREATE POOLS WITH THE SAME PAIR	High	RISK ACCEPTED
(HAL-02) EXCHANGE FEES ARE NOT STABLE	Low	RISK ACCEPTED
(HAL-03) DATA LEAKAGE ATTACKS BY ANALYZING METADATA OF CONTRACTS USAGE	Informational	ACKNOWLEDGED
(HAL-04) MISMATCH BETWEEN ASSERTION AND ERROR DESCRIPTION	Informational	ACKNOWLEDGED

FINDINGS & TECH DETAILS

3.1 (HAL-01) POSSIBILITY TO CREATE POOLS WITH THE SAME PAIR - HIGH

Description:

The AMM module allow users to create pairs (called **exchanges**) from native or custom assets by calling create_exchange function on **factory** contract. It relies on the token_type PartialEq implementation, that allows users to pair the same asset twice, which generates unexpected situations, e.g.: a user could withdraw more tokens than his fair share and affect other users in the pool.

The verification asserts that both tokens addresses are not exactly equal, but a lowercase version of the address will be considered different from the uppercase one.

Code Location:

Listing 1: libraries/amm-shared/src/token_type.rs (Line 32)

```
26 impl<A: PartialEq> PartialEq for TokenType<A> {
27     fn eq(&self, other: &Self) -> bool {
28         match (self, other) {
29             (
30                 Self::CustomToken { contract_addr: l_contract_addr
                       , .. },
31                 Self::CustomToken { contract_addr: r_contract_addr
                       , .. }
32             ) => l_contract_addr == r_contract_addr,
33             (
34                 Self::NativeToken { denom: l_denom },
35                 Self::NativeToken { denom: r_denom }
36             ) => l_denom == r_denom,
37             _ => false
38         }
39     }
40 }
```

Risk Level:

Likelihood - 4
Impact - 4

Recommendation:

The PartialEq implementation for TokenType must be able to compare custom tokens correctly, taking the case into account.

Remediation plan:

RISK ACCEPTED: The Sienna.Network team accepted the risk for this finding.

3.2 (HAL-02) EXCHANGE SWAP FEES ARE NOT STABLE - LOW

Description:

The **exchange** contract allows users to perform swaps between different assets, and comes with a predefined swap fee that capture part of the input amount and leaves it in the pool. However, that captured amount is not considered when performing the swap. To better explain this situation, we take a scenario with little liquidity:

- Alice offers 1000 A to a (1000 A; 1000 B) pool with 0.3% swap fees.
- 3 A are captured from the offer, and the swap is performed on the (1000 A; 1000 B) pool with 997 A instead of a (1003 A; 1000 B) pool.

This realizes the swap on a 1997 total A pool instead of 2000 A pool:

1) Without fee:

- returned = 1000B - (1000A * 1000B)/ (1000A + 1000A)= 500B
- balance after swap: 2000A, 500B

2) Excluding the fee from the pool (current scenario):

- returned = 1000B - (1000A * 1000B)/ (1000A + 997A)= 499.25B
- balance after swap: 2000A, 500.75B
- commission: (500 - 499.25)/500 = 0.15%

3) Including the fee to the pool:

- returned = 1000B - (1003A * 1000B)/ (1003A + 997A)= 498.5B
- balance after swap: 2000A, 501.5B

- commission: (500 - 498.5)/500 = 0.3%

With a 0.3% commission, the user should receive 500B * (1-0.3%)= 498.5B, but the current scenario uses in fact a fee of 0.15%, that is function of the liquidity in the pool. The protocol is therefore failing to capture the wanted amount of commission, at expenses of the pool.

Code Location:

Listing 2: contracts/amm/exchange/src/contract.rs (Lines 676,680,684)

```
673 if !is_simulation {
674     // If not a simulation, need to subtract the incoming amount
675     // from the pool
676     offer_pool = (offer_pool - offer.amount)?;
677 }
678 let total_commission = swap_commission + sienna_commission;
679 let offer_amount = (offer.amount - total_commission)?;
680 Ok(SwapInfo {
681     total_commission,
682     swap_commission,
683     sienna_commission,
684     result: compute_swap(offer_pool, balances[token_index ^ 1],
            offer_amount)?,
685 })
```

Risk Level:

Likelihood - 3
Impact - 2

Recommendation:

Include the retained commission within the offer_pool of swap computation, or only apply the fee on the returned amount.

Remediation plan:

RISK ACCEPTED: The Sienna.Network team accepted the risk of this finding and also stated that fees should be deducted from the input amount as a business decision.

3.3 (HAL-03) DATA LEAKAGE ATTACKS BY ANALYZING METADATA OF CONTRACTS USAGE - INFORMATIONAL

Description:

Depending on a contract's implementation, an attacker could deanonymize information about the contract and its clients. In all the following scenarios, assume that an attacker has a local full node in its control.

For example, it is possible for an attacker to create a list of every account that performs a swap. This attack is based on the length of the (encrypted) message sent to the attacker's node: there is a significant difference of size between the only two queries. On one side, `pair_info` is short and doesn't take any parameter. On the other side, `swap_simulation` is longer and takes a parameter.

This means that the inputs for queries on this contract would look like:

```
"pair_info"
{"swap_simulation": {"offer": some data}}
```

Therefore, the attacker (a validator node or internet provider) can distinguish what query was called given the length of the message. If the message is long, the user might have asked for a swap simulation and could perform a swap transaction after that.

Code Location:

Listing 3: contracts/amm/exchange/src/contract.rs (Lines 182,202)

```
180 pub fn query<S: Storage, A: Api, Q: Querier>(deps: &Extern<S, A, Q
      >, msg: QueryMsg) -> QueryResult {
181     match msg {
182         QueryMsg::PairInfo => {
183             let config = load_config(deps)?;
184
```

```
185             let balances = config.pair.query_balances(
186                 &deps.querier,
187                 config.contract_addr,
188                 config.viewing_key.0,
189             )?;
190             let total_liquidity = query_liquidity(&deps.querier, &
                    config.lp_token_info)?;
191
192             to_binary(&QueryMsgResponse::PairInfo {
193                 liquidity_token: config.lp_token_info,
194                 factory: config.factory_info,
195                 pair: config.pair,
196                 amount_0: balances[0],
197                 amount_1: balances[1],
198                 total_liquidity,
199                 contract_version: CONTRACT_VERSION,
200             })
201         }
202         QueryMsg::SwapSimulation { offer } => {
203             let config = load_config(deps)?;
204             to_binary(&swap_simulation(deps, config, offer)?)
205         }
206     }
207 }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Consider applying remediation from the SCRT privacy model (same length in function names, outputs, same state accessing orders. . .)

Remediation plan:

ACKNOWLEDGED: The Sienna.Network team acknowledged this finding.

3.4 (HAL-04) MISMATCH BETWEEN ASSERTION AND ERROR DESCRIPTION - INFORMATIONAL

Description:

In the **exchange** contract, the description of the slippage check and the code do not match:

- Description explains that slippage tolerance must be bewteen 0.1 and 0.9.
- Verification ensures that slippage is contained between 0 and 1 excluded.

The confusion between the code and the description could confuse users interacting with the contract.

Code Location:

Listing 4: contracts/amm/exchange/src/contract.rs (Lines 741,743)

```
741 if slippage.is_zero() || slippage >= Decimal256::one() {
742     return Err(StdError::generic_err(
743         format!("Slippage tolerance must be between 0.1 and 0.9,
                got: {}", slippage))
744     );
745 }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Either change the code, or the error description, so that both are matching.

Remediation plan:

ACKNOWLEDGED: The Sienna team acknowledged this finding and also stated that if a mistake is made for whatever reason, it can quickly be corrected.

AUTOMATED TESTING

4.1 AUTOMATED ANALYSIS

Description:

Halborn used automated security scanners to assist with detection of well-known security issues and vulnerabilities. Among the tools used was cargo audit, a security scanner for vulnerabilities reported to the RustSec Advisory Database. All vulnerabilities published in https://crates.io are stored in a repository named The RustSec Advisory Database. cargo audit is a human-readable version of the advisory database which performs a scanning on Cargo.lock. Security Detections are only in scope. All vulnerabilities shown here were already disclosed in the above report. However, to better assist the developers maintaining this code, the auditors are including the output with the dependencies tree, and this is included in the cargo audit output to better know the dependencies affected by unmaintained and vulnerable crates.

ID	package	Short Description
RUSTSEC-2020-0159	chrono	Potential segfault in ‘localtime_r‘
RUSTSEC-2021-0076	libsecp256k1	libsecp256k1 allows overflowing
RUSTSEC-2020-0071	time	Potential segfault in the time crate

THANK YOU FOR CHOOSING

// HALBORN