// HALBORN

Astroport.fi Governance

CosmWasm Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **April 18th, 2022 - May 10th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	04/18/2022	Thiago Mathias
0.2	Document Update	04/27/2022	Michal Bazyli
0.3	Document Update	04/29/2022	Thiago Mathias
0.4	Draft Version	05/10/2022	Thiago Mathias
0.5	Draft Review	05/11/2022	Gabi Urrutia
1.0	Remediation Plan	05/23/2022	Michal Bazyli
1.1	Remediation Plan Review	05/24/2022	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
Michal Bazyli	Halborn	Michal.Bazyli@halborn.com
Thiago Mathias	Halborn	Thiago.Mathias@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Astroport.fi engaged Halborn to conduct a security audit on their smart contracts beginning on April 18th, 2022 and ending on May 10th, 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 two weeks and a half for the engagement and assigned a two full-time security engineers 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

In summary, Halborn identified some improvements to reduce the likelihood and impact of risks, which were mostly addressed by Astroport team. The main ones are the following:

- Enforce the limit for pools to vote for.
- Ensure a minimum and maximum value of configuration parameters.
- Add appropriate checks in each contract and package.
- Implement a validation routine to validate the characters entered the fields and whitelisted links.
- Enforce usage of appropriate arithmetical methods.

1.3 TEST APPROACH & METHODOLOGY

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.

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL

10 - CRITICAL
9 - 8 - HIGH

7 - 6 - MEDIUM

5 - 4 - LOW

3 - 1 - VERY LOW AND INFORMATIONAL

1.4 SCOPE

Code repository: astroport-governance

1. CosmWasm Governance Smart Contracts
   (a) Commit ID: 0b1a4282b062cc2f9e8fb3684f203e6fad1f9fd2
   (b) Contracts in scope:
      - assembly
      - escrow_fee_distributor
      - generator_controller
      - voting_escrow

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

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	1	1	3

LIKELIHOOD

IMPACT

	(HAL-02)	(HAL-01)		
(HAL-03) (HAL-04) (HAL-05)				

EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
HAL01 - THE POOL LIMIT IS NOT ENFORCED FOR VOTING	Medium	NOT APPLICABLE
HAL02 - THRESHOLDS VALUES NOT ENFORCED	Low	SOLVED - 05/19/2022
HAL03 - OVERFLOW CHECKS NOT SET FOR PROFILE RELEASE	Informational	ACKNOWLEGED
HAL04 - LACK OF VERIFICATION FOR MARKETING INFOS	Informational	SOLVED - 05/19/2022
HAL05 - MULTIPLE INSTANCES OF UNCHECKED MATH	Informational	SOLVED - 05/19/2022

FINDINGS & TECH DETAILS

3.1 (HAL-01) THE POOL LIMIT IS NOT ENFORCED FOR VOTING - MEDIUM

Description:

The handle_vote function in **contracts/generator_controller/src/contract.rs** allows users to vote for multiple pools. The function does not enforce a maximum pool limit for voting. Consequently, users who vote for multiple pools, say 100, will reach the maximum message size.

```
    message: 'failed to execute message; message index: 0: execute msg size is
too huge: exceeds max contract msg size limit: invalid request',
    details: []
```

Code Location:

Listing 1: contracts/generator_controller/src/contract.rs (Line 166)

```
162 fn handle_vote(
163     deps: DepsMut,
164     env: Env,
165     info: MessageInfo,
166     votes: Vec<(String, u16)>,
167 ) -> ExecuteResult {
168     let user = info.sender;
169     let block_period = get_period(env.block.time.seconds())?;
170     let escrow_addr = CONFIG.load(deps.storage)?.escrow_addr;
171     let user_vp = get_voting_power(deps.querier, &escrow_addr, &
 ↳ user)?;
172 
173     if user_vp.is_zero() {
174         return Err(ContractError::ZeroVotingPower {});
175     }
176 
177     let user_info = USER_INFO.may_load(deps.storage, &user)?.
 ↳ unwrap_or_default();
178     // Does the user eligible to vote again?
179     if env.block.time.seconds() - user_info.vote_ts <
 ↳ VOTE_COOLDOWN {
180         return Err(ContractError::CooldownError(VOTE_COOLDOWN /
 ↳ DAY));
```

```
181     }
182
183     // Check duplicated votes
184     let addrs_set = votes
185         .iter()
186         .cloned()
187         .map(|(addr, _)| addr)
188         .collect::<HashSet<_>>();
189     if votes.len() != addrs_set.len() {
190
```

Risk Level:

Likelihood - 3
Impact - 3

Recommendation:

Enforce the limit of pools to vote to avoid reaching the maximum message size limit.

Remediation Plan:

NOT APPLICABLE: The Astroport team claimed that the issue is a limitation of the Terra Platform and appears before entering the smart contract code. So, there is no way to control it on contract side. The limitation will be applied on the front-end side.

3.2 (HAL-02) THRESHOLDS VALUES NOT ENFORCED - LOW

Description:

There is no validation routine in the instantiate and update_config functions of **contracts/assembly/src/contract.rs** to check if the correct value has been set for the configuration parameters, which could result in the following situations:

- The voting period may be too short, making it impossible to vote, or too long a period that takes too long to close, depending on the value entered in proposal_voting_period.
- The proposal_effective_delay can be set too large and take too long to perform a proposal.
- If proposal_expiration_period is too short, the proposal will have little time to be effective.
- The proposal_required_deposit if it is too high, it will not be feasible to send proposals.
- A vote can be closed with few participants if the proposal_required_quorum is too small, or never be closed because it does not reach a very large quorum.
- proposal required threshold can cause a proposal to be approved with few votes in favor, or rejected with many votes in favor.

In the **contracts/escrow_fee_distributor/src/contract.rs** contract, the maximum number of addresses when claiming fees for multiple users, represented by the claim_many_limit parameter, is not validated against a minimum acceptable value when set. Consequently, this value can be set to 0, which will cause claim many feature to always fail.

Code Location:

instantiate code:

Listing 2: contracts/assembly/src/contract.rs (Lines 66-71)

```
62     let mut config = Config {
63         xastro_token_addr: addr_validate_to_lower(deps.api, &msg.
 ↳ xastro_token_addr)?,
64         vxastro_token_addr: None,
65         builder_unlock_addr: addr_validate_to_lower(deps.api, &msg
 ↳ .builder_unlock_addr)?,
66         proposal_voting_period: msg.proposal_voting_period,
67         proposal_effective_delay: msg.proposal_effective_delay,
68         proposal_expiration_period: msg.proposal_expiration_period
 ↳ ,
69         proposal_required_deposit: msg.proposal_required_deposit,
70         proposal_required_quorum: Decimal::from_str(&msg.
 ↳ proposal_required_quorum)?,
71         proposal_required_threshold: Decimal::from_str(&msg.
 ↳ proposal_required_threshold)?,
72         whitelisted_links: msg.whitelisted_links,
73     };
74 }
```

update_config code:

Listing 3: contracts/assembly/src/contract.rs (Lines 525,529,533,537,541,545)

```
524     if let Some(proposal_voting_period) = updated_config.
 ↳ proposal_voting_period {
525         config.proposal_voting_period = proposal_voting_period;
526     }
527
528     if let Some(proposal_effective_delay) = updated_config.
 ↳ proposal_effective_delay {
529         config.proposal_effective_delay = proposal_effective_delay
 ↳ ;
530     }
531
532     if let Some(proposal_expiration_period) = updated_config.
 ↳ proposal_expiration_period {
```

```
533             config.proposal_expiration_period =
 ↳ proposal_expiration_period;
534         }
535
536         if let Some(proposal_required_deposit) = updated_config.
 ↳ proposal_required_deposit {
537             config.proposal_required_deposit = Uint128::from(
 ↳ proposal_required_deposit);
538         }
539
540         if let Some(proposal_required_quorum) = updated_config.
 ↳ proposal_required_quorum {
541             config.proposal_required_quorum = Decimal::from_str(&
 ↳ proposal_required_quorum)?;
542         }
543
544         if let Some(proposal_required_threshold) = updated_config.
 ↳ proposal_required_threshold {
545             config.proposal_required_threshold = Decimal::from_str(&
 ↳ proposal_required_threshold)?;
546         }
```

instantiate code:

Listing 4: contracts/escrow_fee_distributor/src/contract.rs (Line 53)

```
38 pub fn instantiate(
39     deps: DepsMut,
40     _env: Env,
41     _info: MessageInfo,
42     msg: InstantiateMsg,
43 ) -> StdResult<Response> {
44     set_contract_version(deps.storage, CONTRACT_NAME,
 ↳ CONTRACT_VERSION)?;
45
46     CONFIG.save(
47         deps.storage,
48         &Config {
49             owner: addr_validate_to_lower(deps.api, &msg.owner)?,
50             astro_token: addr_validate_to_lower(deps.api, &msg.
 ↳ astro_token)?,
51             voting_escrow_addr: addr_validate_to_lower(deps.api, &
 ↳ msg.voting_escrow_addr)?,
```

```
52             is_claim_disabled: msg.is_claim_disabled.unwrap_or(
↳ false),
53             claim_many_limit: msg.claim_many_limit.unwrap_or(
↳ CLAIM_LIMIT),
54         },
55     )?;
56
57     Ok(Response::new())
58 }
```

update_config code:

Listing 5: contracts/escrow_fee_distributor/src/contract.rs (Line 399)

```
398     if let Some(claim_many_limit) = claim_many_limit {
399         config.claim_many_limit = claim_many_limit;
400         attributes.push(Attribute::new(
401             "claim_many_limit",
402             claim_many_limit.to_string(),
403         ));
404     };
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

A validation routine should be added within the update_config and instantiate functions to enforce that the values are within the expected range.

Remediation Plan:

SOLVED: The issue was fixed in commit a88815bf8e4b364a1858c85f305d09bb1431e24a.

3.3 (HAL-03) OVERFLOW CHECKS NOT SET FOR PROFILE RELEASE - INFORMATIONAL

Description:

Although the overflow-checks parameter is set to **true** in profile.release and is implicitly applied to all contracts and packages in the workspace, it is not explicitly enabled in **Cargo.toml** for each individual contract and package, which could have unexpected consequences if the project is refactored.

Code Location:

Listing 6: Resources affected

```
contracts/assembly/Cargo.toml
contracts/escrow_fee_distributor/Cargo.toml
contracts/generator_controller/Cargo.toml
contracts/voting_escrow/Cargo.toml
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

It is recommended that you explicitly enable overflow checks on each individual contract and package. That measure helps when the project is refactored to avoid unintended consequences.

Remediation Plan:

ACKNOWLEGED: The Astroport team acknowleged the risk of this finding.

3.4 (HAL-04) LACK OF VERIFICATION FOR MARKETING INFOS - INFORMATIONAL

Description:

The instantiate and execute_update_marketing functions of the **contracts/voting_escrow/src/contract.rs** does not validate data entered in marketing variables (Logo::Url(_), project and description). When the instantiate function receives the Logo::Url(_), marketing.project and marketing.description variables, no sanitization is performed to prevent problems in the front-end of the web application with malformed data/links, in this way, the contract stores the information in the respective variables for storage.

Code Location:

instantiate code:

Listing 7: contracts/voting_escrow/src/contract.rs (Lines 115,123,124)

```
110     if let Some(marketing) = msg.marketing {
111         let logo = if let Some(logo) = marketing.logo {
112             LOGO.save(deps.storage, &logo)?;
113
114             match logo {
115                 Logo::Url(url) => Some(LogoInfo::Url(url)),
116                 Logo::Embedded(_) => Some(LogoInfo::Embedded),
117             }
118         } else {
119             None
120         };
121
122         let data = MarketingInfoResponse {
123             project: marketing.project,
124             description: marketing.description,
125             marketing: marketing
126                 .marketing
```

```
127                 .map(|addr| addr_validate_to_lower(deps.api, &addr
 ↳ ))
128                 .transpose()?,
129             logo,
130         };
131         MARKETING_INFO.save(deps.storage, &data)?;
132     }
133
```

execute_update_marketing code from **CW20-base-0.8.1**:

Listing 8: cw20-base-0.8.1/src/contract.rs (Lines 385,391,397)

```
362 pub fn execute_update_marketing(
363     deps: DepsMut,
364     _env: Env,
365     info: MessageInfo,
366     project: Option<String>,
367     description: Option<String>,
368     marketing: Option<String>,
369 ) -> Result<Response, ContractError> {
370     let mut marketing_info = MARKETING_INFO
371         .may_load(deps.storage)?
372         .ok_or(ContractError::Unauthorized {})?;
373
374     if marketing_info
375         .marketing
376         .as_ref()
377         .ok_or(ContractError::Unauthorized {})?
378         != &info.sender
379     {
380         return Err(ContractError::Unauthorized {});
381     }
382
383     match project {
384         Some(empty) if empty.trim().is_empty() => marketing_info.
 ↳ project = None,
385         Some(project) => marketing_info.project = Some(project),
386         None => (),
387     }
388
389     match description {
390         Some(empty) if empty.trim().is_empty() => marketing_info.
```

```
 ↳ description = None,
391         Some(description) => marketing_info.description = Some(
 ↳ description),
392         None => (),
393     }
394
395     match marketing {
396         Some(empty) if empty.trim().is_empty() => marketing_info.
 ↳ marketing = None,
397         Some(marketing) => marketing_info.marketing = Some(deps.
 ↳ api.addr_validate(&marketing)?),
398         None => (),
399     }
400
401     if marketing_info.project.is_none()
402         && marketing_info.description.is_none()
403         && marketing_info.marketing.is_none()
404         && marketing_info.logo.is_none()
405     {
406         MARKETING_INFO.remove(deps.storage);
407     } else {
408         MARKETING_INFO.save(deps.storage, &marketing_info)?;
409     }
410
411     let res = Response::new().add_attribute("action", "
 ↳ update_marketing");
412     Ok(res)
413 }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Implement a validation routine similar to the one implemented in the **contracts/assembly/assembly.rs** contract in the submit_proposal function to validate the characters entered in the fields and whitelisted links.

Remediation Plan:

SOLVED: The issue was fixed in commit a88815bf8e4b364a1858c85f305d09bb1431e24a.

3.5 (HAL-05) MULTIPLE INSTANCES OF UNCHECKED MATH - INFORMATIONAL

Description:

While many instances of proven arithmetic were observed, some calculations omitted these checks. Additional verification performed when using the checked functions ensures that under/overflow states are detected and handled appropriately.

While these instances were not found to be directly exploitable, they should be reviewed to ensure a defence-in-depth approach is achieved.

Code Location:

receive_cw20 code:

Listing 9: contracts/escrow_fee_distributor/src/contract.rs (Line 161)

```
143 fn receive_cw20(
144     deps: DepsMut,
145     env: Env,
146     info: MessageInfo,
147     cw20_msg: Cw20ReceiveMsg,
148 ) -> Result<Response, ContractError> {
149     let config: Config = CONFIG.load(deps.storage)?;
150     if info.sender != config.astro_token {
151         return Err(ContractError::Unauthorized {});
152     }
153
154     let curr_period = get_period(env.block.time.seconds())?;
155
156     REWARDS_PER_WEEK.update(
157         deps.storage,
158         U64Key::new(curr_period),
159         |period| -> StdResult<_> {
160             if let Some(tokens_amount) = period {
161                 Ok(tokens_amount + cw20_msg.amount)
162             } else {
163                 Ok(cw20_msg.amount)
```

```
164             }
165         },
166     )?;
167
168     Ok(Response::new())
169 }
```

claim_many code:

Listing 10: contracts/escrow_fee_distributor/src/contract.rs (Line 260)

```
227 fn claim_many(
228     mut deps: DepsMut,
229     env: Env,
230     receivers: Vec<String>,
231 ) -> Result<Response, ContractError> {
232     let config: Config = CONFIG.load(deps.storage)?;
233
234     if config.is_claim_disabled {
235         return Err(ContractError::ClaimDisabled {});
236     }
237
238     if receivers.len() > config.claim_many_limit as usize {
239         return Err(ContractError::ClaimLimitExceeded {});
240     }
241
242     let mut claim_total_amount = Uint128::zero();
243     let mut transfer_msg = vec![];
244
245     for receiver in receivers {
246         let receiver_addr = addr_validate_to_lower(deps.api, &
 ↳ receiver)?;
247         let claim_amount = calc_claim_amount(
248             deps.branch(),
249             env.clone(),
250             receiver_addr.clone(),
251             config.clone(),
252         )?;
253
254         if !claim_amount.is_zero() {
255             transfer_msg.extend(transfer_token_amount(
256                 config.astro_token.clone(),
```

```
257                 receiver_addr,
258                 claim_amount,
259             )?);
260             claim_total_amount += claim_amount;
261         };
262     }
263
264     let response = Response::new()
265         .add_attributes(vec![
266             attr("action", "claim_many"),
267             attr("amount", claim_total_amount.to_string()),
268         ])
269         .add_messages(transfer_msg);
270
271     Ok(response)
272 }
```

calculate_reward code:

Listing 11: contracts/escrow_fee_distributor/src/contract.rs (Line 363)

```
353 fn calculate_reward(
354     deps: Deps,
355     period: u64,
356     user_vp: Uint128,
357     total_vp: Uint128,
358 ) -> StdResult<Uint128> {
359     let rewards_per_week = REWARDS_PER_WEEK
360         .may_load(deps.storage, U64Key::from(period))?
361         .unwrap_or_default();
362
363     Ok(user_vp.multiply_ratio(rewards_per_week, total_vp))
364 }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

In “release” mode, Rust does not panic! due to overflows and overflowed values simply “wrap” without any explicit feedback to the user. Then, it is recommended to use vetted safe math libraries for arithmetic operations consistently throughout the smart contract system. Consider replacing the addition operator with Rust’s checked_add method, the subtraction operator with Rust’s checked_subs method, and so on.

Remediation Plan:

SOLVED: The issue was fixed in commits d534338b1bcd9f5be4ee127d5f033184f87f222c and a88815bf8e4b364a1858c85f305d09bb1431e24a.

THANK YOU FOR CHOOSING

// HALBORN