// HALBORN

Astroport.fi bLUNA Stableswap and Lockdrop

CosmWasm Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **March 21st, 2022 - April 8th, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	03/21/2022	Jose C. Ramirez
0.2	Document Update	04/08/2022	Jakub Heba
0.3	Draft Version	04/08/2022	Jose C. Ramirez
0.4	Draft Review	04/12/2022	Gabi Urrutia
1.0	Remediation Plan	05/09/2022	Jakub Heba
1.1	Remediation Plan	05/10/2022	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
Jakub Heba	Halborn	Jakub.Heba@halborn.com
Jose Ramirez	Halborn	Jose.Ramirez@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Astroport.fi engaged Halborn to conduct a security audit on their smart contracts beginning on March 21st, 2022 and ending on April 8th, 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 three weeks for the engagement and assigned two full-time security engineers to audit the security of the smart contract. The security engineers are blockchain and smart-contract security experts 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 has been mostly addressed by the Astroport.fi team. The main ones are the following:

- Review the order in which the parameters of the reward distribution function are supplied.
- Explore the possibility of splitting the change-owner functionality into two steps, including set_owner and accept_owner functions.
- Normalize all addresses provided by users in contracts.
- Enforce a minimum value of at least one for the maximum number of simultaneous locks.
- Consider changing the logic of creating LP tokens' name based on similar names of the components in the pair.

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.

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 repositories:

1. CosmWasm Smart Contract - bLuna Stableswap
    (a) Repository: astroport-core
    (b) Commit ID: 49edeaff6341a4d2f7cd8ef00e003b27e1bbab20
    (c) Contract in scope:
        i. pair_stable_bluna
2. CosmWasm Smart Contract - Lockdrop
    (a) Repository: astroport-bootstrapping
    (b) Commit ID: 7e11bd4727e3cd825169d8ba736fbcaf7921e1ab
    (c) Contracts in scope:
        i. lockdrop

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

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	1	2	2	6

LIKELIHOOD

IMPACT

	(HAL-02) (HAL-03)			
(HAL-04)				(HAL-01)
	(HAL-05)			
(HAL-07) (HAL-08) (HAL-09) (HAL-10) (HAL-11)	(HAL-06)			

EXECUTIVE OVERVIEW

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) USER REWARDS NOT CLAIMABLE DUE TO TYPO	High	SOLVED - 04/18/2022
(HAL-02) PRIVILEGED ADDRESS CAN BE TRANSFERRED WITHOUT CONFIRMATION	Medium	SOLVED - 04/18/2022
(HAL-03) LACK OF ADDRESS NORMALIZATION	Medium	SOLVED - 04/18/2022
(HAL-04) MINIMUM VALUE OF CONFIG PARAMETER NOT ENFORCED	Low	SOLVED - 04/18/2022
(HAL-05) LP TOKEN NAME COLLISION	Low	RISK ACCEPTED
(HAL-06) DUPLICATED CONFIG LOADED INSIDE A FUNCTION	Informational	SOLVED - 04/18/2022
(HAL-07) MULTIPLE INSTANCES OF UNCHECKED MATH	Informational	SOLVED - 04/18/2022
(HAL-08) OVERFLOW CHECKS NOT SET FOR PROFILE RELEASE	Informational	ACKNOWLEDGED
(HAL-09) NEVER USED FUNCTION	Informational	SOLVED - 04/18/2022
(HAL-10) FUNCTION RESPONSIBLE FOR SLIPPAGE TOLERANCE IS EMPTY	Informational	ACKNOWLEDGED
(HAL-11) UNMANTAINED DEPENDENCY	Informational	PARTIALLY SOLVED

FINDINGS & TECH DETAILS

3.1 (HAL-01) USER REWARDS NOT CLAIMABLE DUE TO TYPO - HIGH

Description:

Parameters are incorrectly passed from the CallbackMsg::DistributeAssetRewards message to its handler. The recipient address and LP token address have a mixed order when supplied to the callback_distribute_asset_reward function.

This issue causes the function to fail some validation steps, preventing users from successfully receiving rewards.

Code Location:

Message CallbackMsg::DistributeAssetRewards, while calling its handler passes the terraswap_lp_token parameter before the recipient parameter:

Listing 1: lockdrop/src/contract.rs (Lines 264,265)

```
254  CallbackMsg::DistributeAssetReward {
255              previous_balance,
256              terraswap_lp_token,
257              user_address,
258              recipient,
259              lock_duration,
260          } => callback_distribute_asset_reward(
261              deps,
262              env,
263              previous_balance,
264              terraswap_lp_token,
265              recipient,
266              user_address,
267              lock_duration,
268          ),
269      }
270  }
```

However, the callback_distribute_asset_reward function expects the recipient parameter to be sent before terraswap_lp_token. Thus, when it is called, the values of the parameters are swapped, which disturbs the logic of the contract:

Listing 2: lockdrop/src/contract.rs (Lines 1708,1709)

```
1704 fn callback_distribute_asset_reward(
1705     deps: DepsMut,
1706     env: Env,
1707     previous_balance: Uint128,
1708     recipient: Addr,
1709     terraswap_lp_token: Addr,
1710     user_address: Addr,
1711     lock_duration: u64,
1712 ) -> StdResult<Response> {
```

Risk Level:

Likelihood - 5
Impact - 3

Recommendation:

Review the order in which the parameters are passed. The recipient and terraswap_lp_token parameters should be swapped to achieve the intended order.

Remediation plan:

SOLVED: The issue was fixed in commit cdab941d09de44ab9ead870debdcd52e0e00a429.

3.2 (HAL-02) PRIVILEGED ADDRESS CAN BE TRANSFERRED WITHOUT CONFIRMATION - MEDIUM

Description:

An incorrect use of the lockdrop contract's handle_update_config function could set the owner to an invalid address, unintentionally losing control of the contract, which cannot be undone in any way. Currently, the contract owner can change its address using the aforementioned function in a single transaction and without confirmation of the new address.

Code Location:

Listing 3: lockdrop/src/contract.rs (Lines 366,376,377,378,379,424)

```
363 pub fn handle_update_config(
364     deps: DepsMut,
365     info: MessageInfo,
366     new_config: UpdateConfigMsg,
367 ) -> StdResult<Response> {
368     let mut config = CONFIG.load(deps.storage)?;
369     let mut attributes = vec![attr("action", "update_config")];
370 
371     // CHECK :: Only owner can call this function
372     if info.sender != config.owner {
373         return Err(StdError::generic_err("Unauthorized"));
374     }
375 
376     if let Some(owner) = new_config.owner {
377         config.owner = deps.api.addr_validate(&owner)?;
378         attributes.push(attr("new_owner", owner.as_str()))
379     };
```

Risk Level:

Likelihood - 2
Impact - 4

Recommendation:

The `handle_update_config` function should follow a two-step process, splitting into the `set_owner` and `accept_owner` functions. The latter requires the recipient to complete the transfer, effectively protecting the contract against possible typographical errors, compared to one-step owner transfer mechanisms.

Remediation plan:

SOLVED: The issue was fixed in commit cdab941d09de44ab9ead870debdcd52e0e00a429.

3.3 (HAL-03) LACK OF ADDRESS NORMALIZATION - MEDIUM

Description:

The multiple features of the lockdrop contract do not take into account that Terra addresses are valid both upper and all lower case. Although valid, a strict comparison between the same address in its all uppercase version (e.g.: **TERRA1KG...XNL8**) and its all lowercase version (e.g.: **terra1kg...xnl8**) fail to.

The likelihood of this issue was reduced as the affected functions were owner-only functionalities, therefore much less prone to error or Queries. Queries affected by this issue will only cause inconvenience rather than a security issue.

Undesired situations could occur, such as setting the new contract owner to an invalid new_config.owner since the UpdateConfigMsg structure stores addresses as a String but the value is never converted to_lower(). If the new owner's address is sent as uppercase characters, validation will succeed, but info.sender will never be the same as the new owner's, effectively blocking access to administrative functions for the entire contract.

Code Location:

Listing 4: lockdrop/src/contract.rs (Lines 366,377)

```
363 pub fn handle_update_config(
364     deps: DepsMut,
365     info: MessageInfo,
366     new_config: UpdateConfigMsg,
367 ) -> StdResult<Response> {
368     let mut config = CONFIG.load(deps.storage)?;
369     let mut attributes = vec![attr("action", "update_config")];
370
371     // CHECK :: Only owner can call this function
372     if info.sender != config.owner {
```

```
373             return Err(StdError::generic_err("Unauthorized"));
374         }
375
376         if let Some(owner) = new_config.owner {
377             config.owner = deps.api.addr_validate(&owner)?;
378             attributes.push(attr("new_owner", owner.as_str()))
379         };
380 }
```

The above-mentioned lack of normalization was also noted in the following lines of the contract:

Listing 5: Affected resources

```
1 lockdrop/src/contract.rs: #385,395,420, 487, 555, 646, 647, 749,
↳ 910, 1096, 1249, 1297, 1822, 1921, 1922
```

Risk Level:

Likelihood - 2
Impact - 4

Recommendation:

One of the two approaches detailed below should be used:

- Update the cosmwasm-vm and use cosmwasm_std::Api::addr_validate (reference CWA-2022-002).
- If the update mentioned is not possible, addressees could be stored in canonical format by using the cosmwasm_std::Api::addr_canonicalize utility function.

The following considerations should be taking into account when implementing the second option:

- To successfully compare a canonical address, both ends should be in canonical format. For example, when performing access controls, the

sender (e.g.: `info.sender` or `env.message.sender`) should be canonicalized beforehand too.

- To send funds to a canonicalized address or include them into a message to a different contract, they should be first turn into its human-readable format via the `cosmwasm_std::Api::addr_humanize` utility function

Remediation plan:

SOLVED: The issue was fixed in commit `cdab941d09de44ab9ead870debdcd52e0e00a429`.

3.4 (HAL-04) MINIMUM VALUE OF CONFIG PARAMETER NOT ENFORCED - LOW

Description:

The maximum number of positions locked per user limit, represented by the max_positions_per_user parameter, is not validated against a minimum acceptable value when set. In case this value is set to 0, users will not be able to add any locked positions, which will limit access to some contract features.

Code Location:

Listing 6: lockdrop/src/contract.rs (Lines 84,93)

```
67    let config = Config {
68          owner: msg
69              .owner
70              .map(|v| deps.api.addr_validate(&v))
71              .transpose()?
72              .unwrap_or(info.sender),
73          astro_token: None,
74          auction_contract: None,
75          generator: None,
76          init_timestamp: msg.init_timestamp,
77          deposit_window: msg.deposit_window,
78          withdrawal_window: msg.withdrawal_window,
79          min_lock_duration: msg.min_lock_duration,
80          max_lock_duration: msg.max_lock_duration,
81          weekly_multiplier: msg.weekly_multiplier,
82          weekly_divider: msg.weekly_divider,
83          lockdrop_incentives: Uint128::zero(),
84          max_positions_per_user: msg.max_positions_per_user,
85      };
86
87      let state = State {
88          total_incentives_share: 0,
89          total_astro_delegated: Uint128::zero(),
90          are_claims_allowed: false,
91      };
```

```
92
93      CONFIG.save(deps.storage, &config)?;
94      STATE.save(deps.storage, &state)?;
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

Add a validation routine to ensure that a minimum value of 1 is set for simultaneously open locks.

Remediation plan:

SOLVED: The issue was fixed in commit cdab941d09de44ab9ead870debdcd52e0e00a429.

3.5 (HAL-05) LP TOKEN NAME COLLISION - LOW

Description:

Different pairs of tokens could generate the same LP token name, which could lead to confusion for users receiving those tokens and any third-party functionality that looked at the token name.

Since only the first four characters of each denom are taken when creating the LP token name, such as TKN1-TKN2-LP, token pairs that include denoms greater than four beginning with the same four letters could cause duplicate names of LP tokens by representing different pairs.

As an example, the tokens TOKNXYZ and TOKNQWE will generate a pair with ASTRO called TOKN-ASTRO-LP.

Code Location:

Listing 7: packages/astroport/src/asset.rs (Lines 314, 329, 330, 332, 333, 334, 339)

```
314 const TOKEN_SYMBOL_MAX_LENGTH: usize = 4;
315
316 /// Returns a formatted LP token name
317 /// ## Params
318 /// * **asset_infos** is an array with two items the type of [`
 ↳ AssetInfo`].
319 ///
320 /// * **querier** is an object of type [`QuerierWrapper`].
321 pub fn format_lp_token_name(
322     asset_infos: [AssetInfo; 2],
323     querier: &QuerierWrapper,
324 ) -> StdResult<String> {
325     let mut short_symbols: Vec<String> = vec![];
326     for asset_info in asset_infos {
327         let short_symbol: String;
328         match asset_info {
329             AssetInfo::NativeToken { denom } => {
```

```
330                 short_symbol = denom.chars().take(
 ↳ TOKEN_SYMBOL_MAX_LENGTH).collect();
331             }
332             AssetInfo::Token { contract_addr } => {
333                 let token_symbol = query_token_symbol(querier,
 ↳ contract_addr)?;
334                 short_symbol = token_symbol.chars().take(
 ↳ TOKEN_SYMBOL_MAX_LENGTH).collect();
335             }
336         }
337         short_symbols.push(short_symbol);
338     }
339     Ok(format!("{}-{}-LP", short_symbols[0], short_symbols[1]).
 ↳ to_uppercase())
340 }
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

Consider changing the way LP token names are created for pairs containing tokens with denoms longer than four characters.

Remediation plan:

RISK ACCEPTED: Astroport.fi team reported this as expected behavior. Users should always be aware of the proper contract address and not rely on the name.

3.6 (HAL-06) DUPLICATED CONFIG LOADED INSIDE A FUNCTION - INFORMATIONAL

Description:

The handle_claim_rewards_and_unlock_for_lockup function loads the current configuration into the config variable twice, overwriting it.
While this is not a security risk, redundant operations should be removed to increase code readability and reduce unnecessary operations.

Code Location:

Listing 8: lockdrop/src/contract.rs (Lines 1078,1093)

```
1070 pub fn handle_claim_rewards_and_unlock_for_lockup(
1071     mut deps: DepsMut,
1072     env: Env,
1073     info: MessageInfo,
1074     terraswap_lp_token: String,
1075     duration: u64,
1076     withdraw_lp_stake: bool,
1077 ) -> StdResult<Response> {
1078     let config = CONFIG.load(deps.storage)?;
1079     let state = STATE.load(deps.storage)?;
1080
1081     if !state.are_claims_allowed {
1082         return Err(StdError::generic_err("Reward claim not allowed
 ↳ "));
1083     }
1084
1085     if env.block.time.seconds()
1086         < config.init_timestamp + config.deposit_window + config.
 ↳ withdrawal_window
1087     {
1088         return Err(StdError::generic_err(
1089             "Deposit / withdraw windows are still open",
1090         ));
1091     }
```

```
1092
1093        let config = CONFIG.load(deps.storage)?;
1094        let user_address = info.sender;
```

Risk Level:

Likelihood - 2
Impact - 1

Recommendation:

One of the assignments to the config variable should be removed.

Remediation plan:

SOLVED: The issue was fixed in commit cdab941d09de44ab9ead870debdcd52e0e00a429.

3.7 (HAL-07) MULTIPLE INSTANCES OF UNCHECKED MATH - INFORMATIONAL

Description:

In computer programming, an overflow occurs when an arithmetic operation attempts to create a numeric value that is outside the range that can be represented by a given number of bits, either greater than the maximum or less than the minimum representable value.

This issue has been raised for information only, as it was not possible to define a clear exploitation scenario for the affected cases. However, they appeared to be potentially risky patterns and have therefore been highlighted as such.

Code Location:

Listing 9

```
lockdrop/src/contract.rs: #454, 2204, 1467, 1486, 1518, 1568, 1966
pair_stable_bluna/src/contract.rs: #1393
pair_stable_bluna/src/math.rs: #31, 37, 58, 64
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

In “release” mode Rust does not panic about overflows and overflowed values simply “wrap” without any explicit feedback to the user. It is then recommended to use vetted and safe math libraries for arithmetic operations consistently throughout the smart contract system. Consider replacing the addition operator with Rust’s checked_add method, the sub-

traction operator with Rust's checked_subs method, and so on.

Remediation plan:

SOLVED: The issue was fixed in commits cdab941d09de44ab9ead870debdcd52e0e00a429 and bdd55bfaf2326ed68c49cd86d2ccf638f9c53419.

3.8 (HAL-08) OVERFLOW CHECKS NOT SET FOR PROFILE RELEASE - INFORMATIONAL

Description:

Although the overflow-checks parameter is set to true in profile.release and implicitly applied to all contracts and packages in the workspace, it is not explicitly enabled in the Cargo.toml files of each individual package. Although not a security issue per se, it could have unintended consequences if the project is refactored.

Code Location:

Listing 10: Affected resource

```
1 pair_stable_bluna/Cargo.toml
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Overflow checks should be explicitly enabled within each individual contract and package.

Remediation plan::

ACKNOWLEDGED: The Astroport team acknowledged this finding.

3.9 (HAL-09) UNUSED FUNCTION - INFORMATIONAL

Description:

The amount_of function has been defined in the codebase, but there is no actual reference to it in any contract.

It is a good practice to remove any “dead code”, unused functions or variables, to improve code readability and optimization.

Code Location:

Listing 11: pair_stable_bluna/src/contract.rs

```
308 pub fn amount_of(coins: &[Coin], denom: String) -> Uint128 {
309     match coins.iter().find(|x| x.denom == denom) {
310         Some(coin) => coin.amount,
311         None => Uint128::zero(),
312     }
313 }
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

If a function is not used in the contract logic, it should be removed from the codebase.

Remediation plan:

SOLVED: The issue was fixed in commit bdd55bfaf2326ed68c49cd86d2ccf638f9c53419.

3.10 (HAL-10) FUNCTION RESPONSIBLE FOR SLIPPAGE TOLERANCE IS EMPTY - INFORMATIONAL

Description:

The functionality to provide liquidity, represented by provide_liquidity function, is defined in the specifications to accept a slippage_tolerance parameter from the user. This information is then passed as an attribute to assert_slippage_tolerance, but this is an empty function that always returns Ok(()).

This may cause incorrect assumptions by the user regarding functionality.

Code Location:

Listing 12: pair_stable_bluna/src/contract.rs (Line 471)

```
470     // Assert slippage tolerance
471     assert_slippage_tolerance(&slippage_tolerance, &deposits, &
 ↳ pools)?;
```

Listing 13: pair_stable_bluna/src/contract.rs (Line 1510)

```
1504 fn assert_slippage_tolerance(
1505     _slippage_tolerance: &Option<Decimal>,
1506     _deposits: &[Uint128; 2],
1507     _pools: &[Asset; 2],
1508 ) -> Result<(), ContractError> {
1509     //There is no slippage in the stable pool
1510     Ok(())
1511 }
```

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

We suggest that the correct logic of the function be implemented, or, if there is no need to use it, remove it from the content of the contract.

Remediation plan::

ACKNOWLEDGED: The Astroport team acknowledged this finding.

3.11 (HAL-11) UNMANTAINED DEPENDENCY - INFORMATIONAL

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. 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-2020-0025	bigint	biginit is unmaintained, use uint instead

Code Location:

Listing 14: Dependency tree

```
bigint 4.4.3
 cosmwasm-bignumber 2.2.0
     astroport-pair-stable-bluna 1.0.1
     astroport-pair-stable 1.0.0
        astroport-oracle 1.0.0
        astroport-generator 1.2.0
     astroport-pair 1.0.0
        astroport-oracle 1.0.0
        astroport-maker 1.0.1
        astroport-generator 1.2.0
        astroport-factory 1.2.0
            astroport-pair-stable-bluna 1.0.1
            astroport-pair-stable 1.0.0
```

```
            astroport-pair 1.0.0
            astroport-oracle 1.0.0
            astroport-maker 1.0.1
            astroport-generator 1.2.0
     astroport-oracle 1.0.0
     astroport-maker 1.0.1

-------

bigint 4.4.3
 cosmwasm-bignumber 2.2.0
     terraswap-pair 0.0.0
        astroport-lockdrop 1.1.0
            astroport-auction 1.0.0
                astroport-lockdrop 1.1.0
                astroport-airdrop 1.0.0
                    astroport-lockdrop 1.1.0
                    astroport-auction 1.0.0
     simple-astroport-airdrop 1.0.0
     astroport-periphery 1.1.0
        simple-astroport-airdrop 1.0.0
        astroport-lockdrop 1.1.0
        astroport-auction 1.0.0
        astroport-airdrop 1.0.0
     astroport-pair-stable 1.0.0
        astroport-lockdrop 1.1.0
     astroport-pair 1.0.0
     astroport-pair 1.0.0
     astroport-lockdrop 1.1.0
     astroport-auction 1.0.0
     astroport-airdrop 1.0.0
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Beware of using dependencies and packages that are no longer supported by developers or have publicly known security flaws, even when they are

not currently exploitable.

Remediation plan:

PARTIALLY SOLVED: Some contracts have been deprived of an outdated dependency in commit cdab941d09de44ab9ead870debdcd52e0e00a429.

THANK YOU FOR CHOOSING

// HALBORN