



Spectrum Protocol

CosmWasm Smart Contract
Security Audit

Prepared by: Halborn

Date of Engagement: August 2nd, 2021 - September 17th, 2021

Visit: Halborn.com

DOCUMENT REVISION HISTORY	6
CONTACTS	6
1 EXECUTIVE OVERVIEW	7
1.1 AUDIT SUMMARY	8
1.2 TEST APPROACH & METHODOLOGY	9
RISK METHODOLOGY	9
1.3 SCOPE	11
2 ASSESSMENT SUMMARY & FINDINGS OVERVIEW	12
3 FINDINGS & TECH DETAILS	13
3.1 (HAL-01) PRIVILEGED ADDRESSES CAN BE TRANSFERRED WITHOUT CONFIRMATION - MEDIUM	15
Description	15
Code Location	15
Risk Level	16
Recommendation	17
Remediation plan	17
3.2 (HAL-02) NOT ENFORCING SLIPPAGE TOLERANCE COULD LEAD TOKENS LOSS - LOW	18
Description	18
Code Location	19
Risk Level	20
Recommendation	20
Remediation plan	20
3.3 (HAL-03) NO MINIMUM THRESHOLD FOR EFFECTIVE DELAY - LOW	21
Description	21

Code Location	21
Risk Level	22
Recommendation	22
Remediation plan	22
3.4 (HAL-04) SPECTRUM PLATFORM COULD BE INITIALIZED WITH INSECURE QUORUM AND THRESHOLD - LOW	23
Description	23
Code Location	23
Risk Level	24
Recommendation	24
Remediation plan	24
3.5 (HAL-05) NO VERIFICATION THAT LOCK END MUST BE GREATER OR EQUAL THAN LOCK START - LOW	25
Description	25
Code Location	25
Risk Level	27
Recommendation	27
Remediation plan	27
3.6 (HAL-06) DEPOSIT FEE RATE COULD BE SET TO A VALUE GREATER THAN 1 - LOW	28
Description	28
Code Location	28
Risk Level	29
Recommendation	29
Remediation plan	29
3.7 (HAL-07) SPEC TOKENS MINTING START COULD BE GREATER THAN MINTING END - LOW	30
Description	30

Code Location	30
Risk Level	31
Recommendation	31
Remediation plan	31
3.8 (HAL-08) NO MECHANISM TO UPDATE ASSETS IF ARE INCORRECTLY REGISTERED - LOW	32
Description	32
Code Location	32
Risk Level	32
Recommendation	33
Remediation plan	33
3.9 (HAL-09) BOND FUNCTION COULD SEND LP TOKENS TO INVALID FARM CONTRACTS - LOW	34
Description	34
Code Location	34
Risk Level	34
Recommendation	35
Remediation plan	35
3.10 (HAL-10) HARVEST AND REINVEST FUNCTIONALITIES ARE NOT RESTRICTED ENOUGH - LOW	36
Description	36
Code Location	36
Risk Level	38
Recommendation	38
Remediation plan	38
3.11 (HAL-11) WITHDRAWAL OF SPEC TOKENS FAILS IF TOO MANY LOCKED BALANCE ENTRIES EXIST - INFORMATIONAL	39
Description	39

Code Location	39
Risk Level	40
Recommendation	40
Remediation plan	40
3.12 (HAL-12) PASSED POLLS WITHOUT EXECUTION MESSAGES BECOME LOCKED – INFORMATIONAL	41
Description	41
Code Location	41
Risk Level	42
Recommendation	42
Remediation plan	42
3.13 (HAL-13) SPECTRUM SPEC FARM COULD HAVE MORE THAN ONE ASSET – INFORMATIONAL	43
Description	43
Code Location	43
Risk Level	44
Recommendation	44
Remediation plan	44
3.14 (HAL-14) MISUSE OF HELPER METHODS – INFORMATIONAL	45
Description	45
Code Location	45
Risk Level	46
Recommendation	46
Remediation plan	46
3.15 (HAL-15) OVERFLOW CHECKS NOT SET FOR PROFILE RELEASE – INFORMATI- TIONAL	47
Description	47

Code Location	47
Risk Level	47
Recommendation	47
Remediation plan	47

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	08/02/2021	Luis Quispe Gonzales
0.2	Document Updates	08/12/2021	Luis Quispe Gonzales
0.3	Document Updates	08/16/2021	Luis Quispe Gonzales
0.4	Document Updates	09/09/2021	Luis Quispe Gonzales
1.0	Draft Version	09/17/2021	Luis Quispe Gonzales
1.1	Remediation Plan	09/27/2021	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



EXECUTIVE OVERVIEW



1.1 AUDIT SUMMARY

Spectrum Protocol engaged Halborn to conduct a security assessment on CosmWasm smart contracts beginning on August 2nd, 2021 and ending September 17th, 2021.

The security engineers involved on the audit 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 achieve the following:

- Ensure that smart contract functions work 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 **Spectrum** team. The main ones are the following:

- Enforce slippage tolerance and validate that does not exceed its max value allowed.
- Split privileged address transfer functionality to allow transfer to be completed by recipient.
- Validate parameters in contracts: effective delay, quorum, threshold, lock start / end, deposit fee, etc.
- Restrict access to harvest and reinvest functionalities.

External threats, such as financial related attacks, oracle attacks, and inter-contract functions and calls should be validated for expected logic and state.

1.2 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. It's quantitative model ensures repeatable and accurate measurement while enabling users to see the underlying vulnerability characteristics that was 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

1.3 SCOPE

1. CosmWasm Smart Contracts

(a) Repository: [spectrumprotocol-contracts](#)

(b) Commit ID: [cca09c4da31580ceecbedc52d25d15b08322bdf7](#)

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	1	9	5

LIKELIHOOD

IMPACT

(HAL-02) (HAL-03) (HAL-04)				
(HAL-09)	(HAL-05) (HAL-06) (HAL-07) (HAL-08)	(HAL-01)		
(HAL-11)				
(HAL-12) (HAL-13) (HAL-14) (HAL-15)		(HAL-10)		

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) PRIVILEGED ADDRESSES CAN BE TRANSFERRED WITHOUT CONFIRMATION	Medium	SOLVED - 08/17/2021
(HAL-02) NOT ENFORCING SLIPPAGE TOLERANCE COULD LEAD TOKENS LOSS	Low	SOLVED - 09/22/2021
(HAL-03) NO MINIMUM THRESHOLD FOR EFFECTIVE DELAY	Low	SOLVED - 09/23/2021
(HAL-04) SPECTRUM PLATFORM COULD BE INITIALIZED WITH INSECURE QUORUM AND THRESHOLD	Low	SOLVED - 09/23/2021
(HAL-05) NO VERIFICATION THAT LOCK END MUST BE GREATER OR EQUAL THAN LOCK START	Low	SOLVED - 08/17/2021
(HAL-06) DEPOSIT FEE RATE COULD BE SET TO A VALUE GREATER THAN 1	Low	SOLVED - 08/19/2021
(HAL-07) SPEC TOKENS MINTING START COULD BE GREATER THAN MINTING END	Low	SOLVED - 08/17/2021
(HAL-08) NO MECHANISM TO UPDATE ASSETS IF ARE INCORRECTLY REGISTERED	Low	SOLVED - 09/23/2021
(HAL-09) BOND FUNCTION COULD SEND LP TOKENS TO INVALID FARM CONTRACTS	Low	SOLVED - 09/22/2021
(HAL-10) HARVEST AND REINVEST FUNCTIONALITIES ARE NOT RESTRICTED ENOUGH	Low	SOLVED - 08/17/2021
(HAL-11) WITHDRAWAL OF SPEC TOKENS FAILS IF TOO MANY LOCKED BALANCE ENTRIES EXIST	Informational	ACKNOWLEDGED
(HAL-12) PASSED POLLS WITHOUT EXECUTION MESSAGES BECOME LOCKED	Informational	NOT APPLICABLE
(HAL-13) SPECTRUM SPEC FARM COULD HAVE MORE THAN ONE ASSET	Informational	NOT APPLICABLE
(HAL-14) MISUSE OF HELPER METHODS	Informational	ACKNOWLEDGED
(HAL-15) OVERFLOW CHECKS NOT SET FOR PROFILE RELEASE	Informational	ACKNOWLEDGED



FINDINGS & TECH DETAILS



3.1 (HAL-01) PRIVILEGED ADDRESSES CAN BE TRANSFERRED WITHOUT CONFIRMATION - MEDIUM

Description:

An incorrect use of the `update_config` function in contracts can set owner to an invalid address and inadvertently lose control of the contracts, which cannot be undone in any way. Currently, the owner of the contracts can change **owner address** using the aforementioned function in a `single transaction` and `without confirmation` from the new address.

The affected smart contracts are the following:

- `spectrum_anchor_farm`
- `spectrum_gov`
- `spectrum_mirror_farm`
- `spectrum_platform`
- `spectrum_pylon_farm`
- `spectrum_spec_farm`
- `spectrum_wallet`

Code Location:

Listing 1: `contracts/spectrum_anchor_farm/src/contract.rs`

```
170 if let Some(owner) = owner {  
171     config.owner = deps.api.canonical_address(&owner)?;  
172 }
```

Listing 2: `contracts/spectrum_gov/src/contract.rs`

```
199 if let Some(owner) = owner {  
200     config.owner = deps.api.canonical_address(&owner)?;  
201 }
```


Listing 3: contracts/spectrum_mirror_farm/src/contract.rs

```
174 if let Some(owner) = owner {
175     config.owner = deps.api.canonical_address(&owner)?;
176 }
```

Listing 4: contracts/spectrum_platform/src/contract.rs

```
119 if let Some(owner) = owner {
120     config.owner = deps.api.canonical_address(&owner)?;
121 }
```

Listing 5: contracts/spectrum_pylon_farm/src/contract.rs

```
170 if let Some(owner) = owner {
171     config.owner = deps.api.canonical_address(&owner)?;
172 }
```

Listing 6: contracts/spectrum_spec_farm/src/contract.rs

```
144 if let Some(owner) = owner {
145     config.owner = deps.api.canonical_address(&owner)?;
146 }
```

Listing 7: contracts/spectrum_wallet/src/contract.rs

```
369 if let Some(owner) = owner {
370     config.owner = deps.api.canonical_address(&owner)?;
371 }
```

Risk Level:

Likelihood - 3

Impact - 3

Recommendation:

It is recommended to split **ownership transfer** functionality into **set_owner** and **accept_ownership** functions. The latter function allows the transfer to be completed by recipient.

Remediation plan:

SOLVED: Issue fixed in commit [6010909e58197a23c4e194e296250b569e9f0564](#).
Contracts owner cannot be updated once **Gov contract** is set as owner.

3.2 (HAL-02) NOT ENFORCING SLIPPAGE TOLERANCE COULD LEAD TOKENS LOSS - LOW

Description:

The `bond` function from `contracts/spectrum_staker/src/contract.rs` does not enforce `slippage_tolerance` parameter when users provide liquidity to `spectrum_staker` contract. As a consequence, if a user mistakenly (or fooled by an attacker) provides liquidity with an imbalanced asset pair, he could lose all his excedent tokens.

Example: A user provides liquidity of `0.999258` UST and `0.006815` mAAPL for `spectrum_staker` contract, he receives `0.079814` mAAPL-UST LP.

```
execute_msg {
  "bond": {
    "assets": [
      {
        "amount": "999258",
        "info": {
          "native_token": {
            "denom": "uusd"
          }
        }
      },
      {
        "amount": "6815",
        "info": {
          "token": {
            "contract_addr": "terra16vfxm98rxlc8erj4g0sj5932dvy1gmdufnugk0"
          }
        }
      }
    ],
    "contract": "terra1hasd1716xtegnch8mjyw2g7mfh9nt3gtdtmpfu"
  }
}
```

Action

Provide 0.995061 UST 0.006815 mAAPL **Liquidity** to Terraswap (mAAPL-UST Pair Contract)
Mint 0.079814 mAAPL-UST LP
Stake LP to Mirror (Staking Contract)

On the other hand, if the user provides liquidity of **0.999258 UST** and **0.681536 mAAPL** for **spectrum_staker** contract, he also receives **0.079814 mAAPL-UST LP**, the same amount of LP tokens than previous transaction, but spending **100 times more** mAAPL tokens.

```
execute_msg {
  "bond": {
    "assets": [
      {
        "amount": "999258",
        "info": {
          "native_token": {
            "denom": "uusd"
          }
        }
      },
      {
        "amount": "681536",
        "info": {
          "token": {
            "contract_addr": "terra16vfxm98rxlc8erj4g0sj5932dvy1gmdufnugk0"
          }
        }
      }
    ],
    "contract": "terra1hasd1716xtegnch8mjyw2g7mfh9nt3gtdtmpfu"
  }
}
```

Action

Provide 0.995061 UST 0.681536 mAAPL **Liquidity** to Terraswap (mAAPL-UST Pair Contract)
Mint 0.079814 mAAPL-UST LP
Stake LP to Mirror (Staking Contract)

Code Location:

Listing 8: contracts/spectrum_staker/src/contract.rs (Lines 69)

```
64 fn bond<S: Storage, A: Api, Q: Querier>(
65     deps: &mut Extern<S, A, Q>,
66     env: Env,
67     contract: HumanAddr,
68     assets: [Asset; 2],
69     slippage_tolerance: Option<Decimal>,
70     compound_rate: Option<Decimal>,
71 ) -> StdResult<HandleResponse> {
```

Listing 9: contracts/spectrum_staker/src/contract.rs (Lines 149)

```

141 CosmosMsg::Wasm(WasmMsg::Execute {
142     contract_addr: terraswap_pair.contract_addr,
143     msg: to_binary(&PairHandleMsg::ProvideLiquidity {
144         assets: if let AssetInfo::NativeToken { .. } = assets
145             [0].info.clone() {
146                 [native_asset.clone(), assets[1].clone()]
147             } else {
148                 [assets[0].clone(), native_asset.clone()]
149             },
150         slippage_tolerance,
151     })?,
152     send: vec![Coin {
153         denom: native_asset.info.to_string(),
154         amount: native_asset.amount,
155     }],
156 })),

```

Risk Level:**Likelihood - 1****Impact - 4****Recommendation:**

Enforce `slippage_tolerance` parameter in `bond` function and add a validation routine to ensure that this value is lesser or equal than a predefined `max value`. As a reference, `max slippage tolerance` for `Uniswap liquidity pools` is 50%.

Remediation plan:

SOLVED: Issue fixed in commit [f298e93f5d0a018a03302f9a317f650d061b5020](#).

3.3 (HAL-03) NO MINIMUM THRESHOLD FOR EFFECTIVE DELAY – LOW

Description:

Timelocks are defined in **Governance contracts** to allow protocol users to react timely if a change made is bad faith or is not in the best interest of protocol and its users.

The `init` and `update_config` functions from `contracts/spectrum_gov/src/-contract.rs` do not restrict that timelock (`effective_delay`) is greater or equal than a **minimum threshold**. So, malicious changes proposed through voting could even be executed immediately if `effective_delay` is not set appropriately.

Code Location:

Listing 10: `contracts/spectrum_gov/src/contract.rs` (Lines 39)

```
29 let config = Config {
30     owner: deps.api.canonical_address(&msg.owner)?,
31     spec_token: if let Some(spec_token) = msg.spec_token {
32         deps.api.canonical_address(&spec_token)?
33     } else {
34         CanonicalAddr::default()
35     },
36     quorum: msg.quorum,
37     threshold: msg.threshold,
38     voting_period: msg.voting_period,
39     effective_delay: msg.effective_delay,
40     expiration_period: msg.expiration_period,
```

Listing 11: `contracts/spectrum_gov/src/contract.rs` (Lines 225)

```
224 if let Some(effective_delay) = effective_delay {
225     config.effective_delay = effective_delay;
226 }
```

Risk Level:

Likelihood - 1

Impact - 4

Recommendation:

Add a validation routine inside `init` and `update_config` functions to ensure that timelock (`effective_delay`) is greater or equal than a **minimum threshold** that allows Spectrum users to act timely against any issue that protocol could have when changes are made. The following are some examples of timelocks used on other protocols:

- Uniswap: 48-hours timelock
- Compound: 48-hours timelock
- Aave: 24-hours timelock (Short time lock)

Remediation plan:

SOLVED: Issue fixed in commit [224e758890d84ad4fbe15c554a587e164e5f92c6](#).

3.4 (HAL-04) SPECTRUM PLATFORM COULD BE INITIALIZED WITH INSECURE QUORUM AND THRESHOLD - LOW

Description:

The `init` function from `contracts/spectrum_platform/src/contract.rs` only restrict that `quorum` and `threshold` parameters are not greater than one; however, when `spectrum_platform` is initialized, there will exist few `boards`, i.e.: who are able to vote, which allows that a malicious board can takeover `spectrum_platform` contract if the aforementioned parameters are not set appropriately.

Attack scenario:

1. The `spectrum_platform` is initialized with the following parameters: `quorum = 10%` and `threshold = 50%`.
2. There are 3 `boards`, each one has a `weight = 1`.
3. **Board 1** creates a new poll with a execution message that changes its own weight to 100.
4. **Board 1** votes for `VoteOption::yes`.
5. **Board 2** votes for `VoteOption::no` and **Board 3** does not vote.
6. Voting phase ends and **Board 1** will be able to change its own weight to 100. So, now he totally controls `spectrum_platform` for future voting, despite of vote results from other `boards`.

Code Location:

Listing 12: `contracts/spectrum_platform/src/contract.rs` (Lines 28,29)

```
18 pub fn init<S: Storage, A: Api, Q: Querier>(
19     deps: &mut Extern<S, A, Q>,
20     env: Env,
21     msg: ConfigInfo,
22 ) -> StdResult<InitResponse> {
23     validate_quorum(msg.quorum)?;
```



```
24     validate_threshold(msg.threshold)?;  
25  
26     let config = Config {  
27         owner: deps.api.canonical_address(&msg.owner)?,  
28         quorum: msg.quorum,  
29         threshold: msg.threshold,  
30         voting_period: msg.voting_period,  
31         effective_delay: msg.effective_delay,  
32         expiration_period: msg.expiration_period,  
33     };
```

Risk Level:

Likelihood - 1

Impact - 4

Recommendation:

Add a validation routine inside `init` function to ensure that `quorum` and `threshold` are greater or equal than 50%.

Remediation plan:

SOLVED: Issue fixed in commit [92d1cd57b8f78b4d562644ae1d8c64cd2e3d12ed](#).

3.5 (HAL-05) NO VERIFICATION THAT LOCK END MUST BE GREATER OR EQUAL THAN LOCK START - LOW

Description:

The `init` and `update_config` functions do not restrict that `lock_end` is greater or equal than `lock_start`. These values are used to calculate locked rewards when a user withdraws rewards from farms.

If they are not correctly set, locked rewards could be miscalculated, so rewards distributed would be unfair. The affected smart contracts are the following:

- `spectrum_anchor_farm`
- `spectrum_mirror_farm`
- `spectrum_pylon_farm`
- `spectrum_spec_farm`

Code Location:

Listing 13: `contracts/spectrum_anchor_farm/src/contract.rs`

```
63 lock_start: msg.lock_start,  
64 lock_end: msg.lock_end,
```

Listing 14: `contracts/spectrum_anchor_farm/src/contract.rs`

```
202 if let Some(lock_start) = lock_start {  
203     config.lock_start = lock_start;  
204 }  
205  
206 if let Some(lock_end) = lock_end {  
207     config.lock_end = lock_end;  
208 }
```

Listing 15: contracts/spectrum_mirror_farm/src/contract.rs

```
66 lock_start: msg.lock_start,  
67 lock_end: msg.lock_end,
```

Listing 16: contracts/spectrum_mirror_farm/src/contract.rs

```
206 if let Some(lock_start) = lock_start {  
207     config.lock_start = lock_start;  
208 }  
209  
210 if let Some(lock_end) = lock_end {  
211     config.lock_end = lock_end;  
212 }
```

Listing 17: contracts/spectrum_pylon_farm/src/contract.rs

```
63 lock_start: msg.lock_start,  
64 lock_end: msg.lock_end,
```

Listing 18: contracts/spectrum_pylon_farm/src/contract.rs

```
202 if let Some(lock_start) = lock_start {  
203     config.lock_start = lock_start;  
204 }  
205  
206 if let Some(lock_end) = lock_end {  
207     config.lock_end = lock_end;  
208 }
```

Listing 19: contracts/spectrum_spec_farm/src/contract.rs

```
27 lock_start: msg.lock_start,  
28 lock_end: msg.lock_end,
```

Listing 20: contracts/spectrum_spec_farm/src/contract.rs

```
148 if let Some(lock_start) = lock_start {  
149     config.lock_start = lock_start;  
150 }
```

```
151
152 if let Some(lock_end) = lock_end {
153     config.lock_end = lock_end;
154 }
```

Risk Level:

Likelihood - 2

Impact - 3

Recommendation:

Add a validation routine inside `init` and `update_config` functions to ensure that `lock_end` is greater or equal than `lock_start`.

Remediation plan:

SOLVED: Issue fixed in commit [6010909e58197a23c4e194e296250b569e9f0564](#). Validation routine has been applied in `init` functions and changes for `lock_start` / `lock_end` have been disabled in `update_config` functions.

3.6 (HAL-06) DEPOSIT FEE RATE COULD BE SET TO A VALUE GREATER THAN 1 - LOW

Description:

The `init` function does not restrict that value of `deposit_fee` rate is greater than 1. This value is used to calculate deposit fee (and its splits) when a user provides liquidity to farm contracts.

If it is not correctly set, the operation will always panic and won't allow legitimate users to provide liquidity, thus generating a denial of service (DoS) in Spectrum protocol.

The affected smart contracts are the following:

- `spectrum_anchor_farm`
- `spectrum_mirror_farm`
- `spectrum_pylon_farm`

Code Location:

Listing 21: `contracts/spectrum_anchor_farm/src/contract.rs`

```
28 pub fn init<S: Storage, A: Api, Q: Querier>(
29     deps: &mut Extern<S, A, Q>,
30     env: Env,
31     msg: ConfigInfo,
32 ) -> StdResult<InitResponse> {
33     validate_percentage(msg.community_fee, "community_fee"?);
34     validate_percentage(msg.platform_fee, "platform_fee"?);
35     validate_percentage(msg.controller_fee, "controller_fee"?);
36
37     let api = deps.api;
```

Listing 22: contracts/spectrum_mirror_farm/src/contract.rs

```

31 pub fn init<S: Storage, A: Api, Q: Querier>(
32     deps: &mut Extern<S, A, Q>,
33     env: Env,
34     msg: ConfigInfo,
35 ) -> StdResult<InitResponse> {
36     validate_percentage(msg.community_fee, "community_fee"?);
37     validate_percentage(msg.platform_fee, "platform_fee"?);
38     validate_percentage(msg.controller_fee, "controller_fee"?);
39
40     let api = deps.api;

```

Listing 23: contracts/spectrum_pylon_farm/src/contract.rs

```

28 pub fn init<S: Storage, A: Api, Q: Querier>(
29     deps: &mut Extern<S, A, Q>,
30     env: Env,
31     msg: ConfigInfo,
32 ) -> StdResult<InitResponse> {
33     validate_percentage(msg.community_fee, "community_fee"?);
34     validate_percentage(msg.platform_fee, "platform_fee"?);
35     validate_percentage(msg.controller_fee, "controller_fee"?);
36
37     let api = deps.api;

```

Risk Level:

Likelihood - 2

Impact - 3

Recommendation:

Apply `validate_percentage` function inside `init` to ensure `deposit_fee` rate is lesser or equal than 1.

Remediation plan:

SOLVED: Issue fixed in commit [3d6bf3908ae1f0eb05aedc8a585015fbcc223120](#).

3.7 (HAL-07) SPEC TOKENS MINTING START COULD BE GREATER THAN MINTING END - LOW

Description:

The `init` and `update_config` functions from `spectrum_gov` contract do not restrict that `mint_end` is greater or equal than `mint_start`. These values are used to calculate how much SPEC tokens can be minted as reward to warchest and vaults.

If they are not correctly set, amount of SPEC tokens to mint could be miscalculated, so reward distributed would be unfair.

Code Location:

Listing 24: `contracts/spectrum_gov/src/contract.rs` (Lines 43,44)

```

20 pub fn init<S: Storage, A: Api, Q: Querier>(
21     deps: &mut Extern<S, A, Q>,
22     env: Env,
23     msg: ConfigInfo,
24 ) -> StdResult<InitResponse> {
25     validate_percentage(msg.quorum, "quorum")?;
26     validate_percentage(msg.threshold, "threshold")?;
27     validate_percentage(msg.warchest_ratio, "warchest_ratio")?;
28
29     let config = Config {
30         owner: deps.api.canonical_address(&msg.owner)?,
31         spec_token: if let Some(spec_token) = msg.spec_token {
32             deps.api.canonical_address(&spec_token)?
33         } else {
34             CanonicalAddr::default()
35         },
36         quorum: msg.quorum,
37         threshold: msg.threshold,
38         voting_period: msg.voting_period,
39         effective_delay: msg.effective_delay,
40         expiration_period: msg.expiration_period,

```

```

41         proposal_deposit: msg.proposal_deposit,
42         mint_per_block: msg.mint_per_block,
43         mint_start: msg.mint_start,
44         mint_end: msg.mint_end,

```

Listing 25: contracts/spectrum_gov/src/contract.rs (Lines 251,255)

```

250     if let Some(mint_start) = mint_start {
251         config.mint_start = mint_start;
252     }
253
254     if let Some(mint_end) = mint_end {
255         config.mint_end = mint_end;
256
257         let mut state = state_store(&mut deps.storage).load()?;
258         if validate_minted(&state, &config, env.block.height).
            is_err() {
259             state.last_mint = env.block.height;
260             state_store(&mut deps.storage).save(&state)?;
261         }
262     }

```

Risk Level:

Likelihood - 2

Impact - 3

Recommendation:

Add a validation routine inside `init` and `update_config` functions to ensure that `mint_end` is greater or equal than `mint_start`.

Remediation plan:

SOLVED: Issue fixed in commit [6010909e58197a23c4e194e296250b569e9f0564](#). Validation routine has been applied in `init` function and changes for `mint_start` / `mint_end` have been disabled in `update_config` function.

3.8 (HAL-08) NO MECHANISM TO UPDATE ASSETS IF ARE INCORRECTLY REGISTERED - LOW

Description:

The `register_asset` function from `contracts/spectrum_mirror_farm/src/contract.rs` does not allow updating `staking_token` parameter if asset is incorrectly registered.

Due to the fact the `spectrum_mirror_farm` handles many assets, if there is any issue with the register of one of them, the contract will never be able to use this asset because it cannot be updated, nor registered again.

Code Location:

Listing 26: `contracts/spectrum_mirror_farm/src/contract.rs` (Lines 243)

```
240 let mut pool_info = pool_info_read(&deps.storage)
241     .may_load(asset_token_raw.as_slice())?
242     .unwrap_or_else(|| PoolInfo {
243         staking_token: deps.api.canonical_address(&staking_token).
            unwrap(),
244         total_auto_bond_share: Uint128::zero(),
245         total_stake_bond_share: Uint128::zero(),
246         total_stake_bond_amount: Uint128::zero(),
247         weight: 0u32,
248         auto_compound: false,
```

Risk Level:

Likelihood - 2

Impact - 3

Recommendation:

It is recommended that `register_asset` function allows updating `staking_token` parameter only if asset's pool is empty, i.e.: bond amount and bond share are zero.

Remediation plan:

SOLVED: Issue fixed in commit [25c7941918e30a8c8f01b3c79c83c78483578803](#).

3.9 (HAL-09) BOND FUNCTION COULD SEND LP TOKENS TO INVALID FARM CONTRACTS - LOW

Description:

The `bond` function from `contracts/spectrum_staker/src/contract.rs` does not restrict that the `contract` parameter sent is an address of an actual farm contract: Anchor, Mirror, Pylon or Spec.

As a consequence, if a user mistakenly (or fooled by an attacker) sends an incorrect address in the aforementioned parameter, the tokens deposited in `spectrum_staker` contract can be totally lost.

Code Location:

Listing 27: `contracts/spectrum_staker/src/contract.rs` (Lines 159)

```
156 CosmosMsg::Wasm(WasmMsg::Execute {  
157     contract_addr: env.contract.address,  
158     msg: to_binary(&HandleMsg::bond_hook {  
159         contract,  
160         asset_token: token_addr.clone(),  
161         staking_token: terraswap_pair.liquidity_token,  
162         staker_addr: env.message.sender,  
163         prev_staking_token_amount,  
164         compound_rate,  
165     })?,  
166     send: vec![],  
167 }),
```

Risk Level:

Likelihood - 1

Impact - 3

Recommendation:

Add a validation routine inside `bond` function to ensure that `contract` parameter belongs to a allowlist of actual farm contracts; otherwise, reject the operation.

Remediation plan:

SOLVED: Issue fixed in commit [c785d1929ee09ce348feeac18b52fed51a8a7abb](#).

3.10 (HAL-10) HARVEST AND REINVEST FUNCTIONALITIES ARE NOT RESTRICTED ENOUGH - LOW

Description:

The `harvest_all`, `re_invest` and `compound` functions are restricted in such a way that only `config.controller` can call them and are used to:

- Harvest pending reward from Staking contract
- Increase pools' reinvest allowance with part of the reward
- Stake remaining reward in Gov contract
- Distribute commission fees
- Reinvest assets and stake LP in Staking contract

However, if `config.controller` has not been previously set, anyone is able to call the functions and bypass the restriction. Because there is no need that external users or smart contracts other than `config.controller` call the aforementioned functions, it is important to apply the principle of least privilege in these cases.

Code Location:

Listing 28: `contracts/spectrum_mirror_farm/src/harvest.rs` (Lines 34)

```

28 pub fn harvest_all<S: Storage, A: Api, Q: Querier>(
29     deps: &mut Extern<S, A, Q>,
30     env: Env,
31 ) -> HandleResult {
32     let config = read_config(&deps.storage)?;
33
34     if config.controller != CanonicalAddr::default()
35         && config.controller != deps.api.canonical_address(&env.
36             message.sender)?
37     {
38         return Err(StdError::unauthorized());
39     }

```

Listing 29: contracts/spectrum_mirror_farm/src/reinvest.rs (Lines 28)

```
21 pub fn re_invest<S: Storage, A: Api, Q: Querier>(
22     deps: &mut Extern<S, A, Q>,
23     env: Env,
24     asset_token: HumanAddr,
25 ) -> StdResult<HandleResponse> {
26     let config = read_config(&deps.storage)?;
27
28     if config.controller != CanonicalAddr::default()
29         && config.controller != deps.api.canonical_address(&env.
30             message.sender)?
31     {
32         return Err(StdError::unauthorized());
33     }
```

Listing 30: contracts/spectrum_anchor_farm/src/compound.rs (Lines 34)

```
28 pub fn compound<S: Storage, A: Api, Q: Querier>(
29     deps: &mut Extern<S, A, Q>,
30     env: Env,
31 ) -> StdResult<HandleResponse> {
32     let config = read_config(&deps.storage)?;
33
34     if config.controller != CanonicalAddr::default()
35         && config.controller != deps.api.canonical_address(&env.
36             message.sender)?
37     {
38         return Err(StdError::unauthorized());
39     }
```

Listing 31: contracts/spectrum_pylon_farm/src/compound.rs (Lines 34)

```
28 pub fn compound<S: Storage, A: Api, Q: Querier>(
29     deps: &mut Extern<S, A, Q>,
30     env: Env,
31 ) -> StdResult<HandleResponse> {
32     let config = read_config(&deps.storage)?;
33
34     if config.controller != CanonicalAddr::default()
35         && config.controller != deps.api.canonical_address(&env.
36             message.sender)?
37     {
38         return Err(StdError::unauthorized());
39     }
```

```
37         return Err(StdError::unauthorized());  
38     }
```

Risk Level:

Likelihood - 3

Impact - 1

Recommendation:

Update the conditional expression in `harvest_all`, `re_invest` and `compound` functions to reject calls from any address other than `config.controller`.

Remediation plan:

SOLVED: Issue fixed in commit [6010909e58197a23c4e194e296250b569e9f0564](#). The `config.controller` address is duly set when `init` function from `spectrum_mirror_farm` contract is called, which invalidates attack vector.

3.11 (HAL-11) WITHDRAWAL OF SPEC TOKENS FAILS IF TOO MANY LOCKED BALANCE ENTRIES EXIST - INFORMATIONAL

Description:

Every time a user withdraws SPEC tokens from Gov contract, the `compute_locked_balance` function is called to calculate the largest `locked_balance` value. Because this function contains an unbounded iteration over the entries in `locked_balance`, a user transaction to get its staked tokens back could fail or spend much more gas than expected. This only affects individual users and the likelihood of the list growing to the point of described issue is very low.

Code Location:

Listing 32: contracts/spectrum_gov/src/stake.rs (Lines 191,206)

```
185 fn compute_locked_balance<S: Storage, A: Api, Q: Querier>(
186     deps: &mut Extern<S, A, Q>,
187     account: &mut Account,
188     voter: &CanonicalAddr,
189 ) -> StdResult<u128> {
190     // filter out not in-progress polls
191     account.locked_balance.retain(|(poll_id, _)| {
192         let poll = read_poll(&deps.storage, &poll_id.to_be_bytes()
193             )
194             .unwrap()
195             .unwrap();
196         if poll.status != PollStatus::in_progress {
197             // remove voter info from the poll
198             poll_voter_store(&mut deps.storage, *poll_id).remove(
199                 voter.as_slice());
200         }
201         poll.status == PollStatus::in_progress
```



```
202     });  
203  
204     Ok(account  
205         .locked_balance  
206         .iter()  
207         .map(|(_, v)| v.balance.u128())  
208         .max()  
209         .unwrap_or_default())  
210 }
```

Risk Level:

Likelihood - 1

Impact - 2

Recommendation:

It is recommended to implement an additional `HandleMsg` that allows users to call `compute_locked_balance` function directly to remove polls which status are not `PollStatus::in_progress`.

Remediation plan:

ACKNOWLEDGED: Spectrum team acknowledged the finding. They have only few polls at a time. Tests were performed by Spectrum team with more than 20 polls opened at a time and have no problem. It is unlikely to have polls more than 5 at a time in general. The issue **does not represent an immediate security impact for the protocol**.

3.12 (HAL-12) PASSED POLLS WITHOUT EXECUTION MESSAGES BECOME LOCKED – INFORMATIONAL

Description:

The `poll_start` function allows to create polls even with `execute_msgs` parameter without any execution message. In the unlikely event one of them is accepted through voting, it won't be possible to change its status to `PollStatus::executed` nor `PollStatus::expired`, so it will remain locked with `PollStatus::passed` status.

Code Location:

Listing 33: `contracts/spectrum_gov/src/poll.rs` (Lines 59)

```
49 let new_poll = Poll {
50     id: poll_id,
51     creator: deps.api.canonical_address(&proposer)?,
52     status: PollStatus::in_progress,
53     yes_votes: Uint128::zero(),
54     no_votes: Uint128::zero(),
55     end_height: env.block.height + config.voting_period,
56     title,
57     description,
58     link,
59     execute_msgs,
60     deposit_amount,
61     total_balance_at_end_poll: None,
62 };
```

Listing 34: `contracts/spectrum_platform/src/poll.rs` (Lines 53)

```
43 let new_poll = Poll {
44     id: poll_id,
45     creator: deps.api.canonical_address(&env.message.sender)?,
46     status: PollStatus::in_progress,
47     yes_votes: 0u32,
```

```
48     no_votes: 0u32,  
49     end_height: env.block.height + config.voting_period,  
50     title,  
51     description,  
52     link,  
53     execute_msgs: execute_msgs,  
54     total_balance_at_end_poll: None,  
55 };
```

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

Add a validation routine inside `poll_start` function to ensure that `execute_msgs` contains at least one execution message; otherwise, reject the operation.

Remediation plan:

NOT APPLICABLE: Spectrum team claims that they use the `text poll` concept to get community opinion without required execution message.

3.13 (HAL-13) SPECTRUM SPEC FARM COULD HAVE MORE THAN ONE ASSET – INFORMATIONAL

Description:

The `register_asset` function from `contracts/spectrum_spec_farm/src/contract.rs` allows than more than one asset can be registered, despite this contract should only handle **SPEC token** as asset. If a new asset is registered mistakenly, other functions like `withdraw_reward` or `read_reward_infos` will spend more gas than expected when called.

Code Location:

Listing 35: `contracts/spectrum_spec_farm/src/contract.rs`

```

99  if config.owner != deps.api.canonical_address(&env.message.sender
    )? {
100      return Err(StdError::unauthorized());
101  }
102
103  let mut state = read_state(&deps.storage)?;
104  deposit_reward(deps, &mut state, &config, env.block.height, false
    )?;
105
106  let mut pool_info = pool_info_read(&deps.storage)
107      .may_load(asset_token_raw.as_slice())?
108      .unwrap_or_else(|| PoolInfo {
109          staking_token: deps.api.canonical_address(&staking_token).
            unwrap(),
110          total_bond_amount: Uint128::zero(),
111          weight: 0u32,
112          state_spec_share_index: state.spec_share_index,
113          spec_share_index: Decimal::zero(),
114      });

```

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

Add a validation routine inside `register_asset` function to ensure that no more than one asset can be registered.

Remediation plan:

NOT APPLICABLE: Spectrum team claims that allowing multiple assets is a planned feature by design on Spectrum farm.

3.14 (HAL-14) MISUSE OF HELPER METHODS – INFORMATIONAL

Description:

The use of the `unwrap` and `expect` function is very useful for testing environments because a value is forcibly demanded to get an error (aka `panic!`) if the “Option” does not have “Some” value or “Result”. Nevertheless, leaving `unwrap` or `expect` functions in production environments is a bad practice because not only will this cause the program to crash out, or `panic!`, but also (in case of `unwrap`) no helpful messages are shown to help the user solve, or understand the reason of the error.

Code Location:

Listing 36: Resources affected

```

1 spectrum_anchor_farm: bond.rs (L394)
2 spectrum_anchor_farm: compound.rs (L397)
3 spectrum_anchor_farm: contract.rs (L246)
4 spectrum_anchor_farm: state.rs (L101,123)
5 spectrum_gov: poll.rs (L414,436,468)
6 spectrum_gov: stake.rs (L193,194,254,262,263,337)
7 spectrum_gov: state.rs (L160)
8 spectrum_mirror_farm: bond.rs (L406)
9 spectrum_mirror_farm: contract.rs (L243)
10 spectrum_mirror_farm: harvest.rs (L175)
11 spectrum_mirror_farm: reinvest.rs (L56)
12 spectrum_mirror_farm: state.rs (L137,159)
13 spectrum_platform: contract.rs (L213)
14 spectrum_platform: poll.rs (L347,368,400)
15 spectrum_platform: state.rs (L113,178)
16 spectrum_pylon_farm: bond.rs (L394)
17 spectrum_pylon_farm: compound.rs (L397)
18 spectrum_pylon_farm: contract.rs (L246)
19 spectrum_pylon_farm: state.rs (L101,123)
20 spectrum_spec_farm: bond.rs (L196)
21 spectrum_spec_farm: contract.rs (L109)
22 spectrum_wallet: contract.rs (L450)
```

Risk Level:**Likelihood - 1****Impact - 1****Recommendation:**

It is recommended to not use the `unwrap` or `expect` functions in a production environment because this use provokes `panic!` and may crash the Spectrum contracts without error messages. Some alternatives are possible, such as propagating the error by putting a `"?"`, using `unwrap_or` / `unwrap_or_else` / `unwrap_or_default` functions, or using `error-chain` crate for errors.

Reference: <https://crates.io/crates/error-chain>

Remediation plan:

ACKNOWLEDGED: Spectrum team acknowledged the finding, but decided not to fix it because does not represent an immediate security impact for the protocol.

3.15 (HAL-15) OVERFLOW CHECKS NOT SET FOR PROFILE RELEASE - INFORMATIONAL

Description:

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

Code Location:

Listing 37: Resources affected

```
1 packages/mirror_protocol/Cargo.toml
2 packages/spectrum_protocol/Cargo.toml
```

Risk Level:

Likelihood - 1

Impact - 1

Recommendation:

It is recommended to explicitly enable overflow checks in each individual contract and package. That measure helps when the project is refactored to prevent unintended consequences.

Remediation plan:

ACKNOWLEDGED: Spectrum team acknowledged the finding. Packages come from Mirror and Anchor code for using only interface. Real entry points are in contract folders. Even if this issue is fixed, when the code of those

packages is copied again, the issue will still happen.



THANK YOU FOR CHOOSING

 **HALBORN**

