// HALBORN

Nexus Protocol

CosmWasm Smart Contract
Security Audit

Prepared by: **Halborn**

Date of Engagement: **August 9th, 2021 - September 3rd, 2021**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY 7
CONTACTS 7
1 EXECUTIVE OVERVIEW 8
1.1 AUDIT SUMMARY 9
1.2 TEST APPROACH & METHODOLOGY 10
RISK METHODOLOGY 10
1.3 SCOPE 12
2 ASSESSMENT SUMMARY & FINDINGS OVERVIEW 13
3 FINDINGS & TECH DETAILS 15
3.1 (HAL-01) MISCALCULATION OF BALANCE LEADS TO OVERRATED REWARDS DISTRIBUTION - HIGH 16
Description 16
Code Location 17
Risk Level 17
Recommendation 18
Remediation plan 18
3.2 (HAL-02) MISCALCULATION OF NASSET TOKENS TO MINT LEADS TO UNFAIR WITHDRAWING / DOS - HIGH 19
Description 19
Code Location 20
Risk Level 20
Recommendation 20
Remediation plan 21
3.3 (HAL-03) NO UPDATE OF LOAN REPAYMENT STATE ON REBALANCE FUNCTION - HIGH 22
Description 22

Code Location 22

Risk Level 23

Recommendation 23

Remediation plan 23

3.4 (HAL-04) PRIVILEGED ADDRESSES CAN BE TRANSFERRED WITHOUT CONFIRMATION - MEDIUM 24

Description 24

Code Location 24

Risk Level 25

Recommendation 25

Remediation plan 25

3.5 (HAL-05) DECIMAL RATES COULD BE UPDATED TO A VALUE GREATER OR EQUAL THAN 1 - LOW 27

Description 27

Code Location 27

Risk Level 27

Recommendation 28

Remediation plan 28

3.6 (HAL-06) LOAN REPAYMENT COULD TAKE MORE ITERATIONS THAN ALLOWED - LOW 29

Description 29

Code Location 29

Risk Level 29

Recommendation 30

Remediation plan 30

3.7 (HAL-07) ROUNDING ISSUES WHEN DEPOSITING / WITHDRAWING BASSET TOKENS - LOW 31

Description 31

Code Location 31
Risk Level 32
Recommendation 32
Remediation plan 32
3.8 (HAL-08) UNAUTHORIZED TOKEN REWARDS CONTRACT ADDRESS MODIFICATION - INFORMATIONAL 33
Description 33
Code Location 33
Risk Level 34
Recommendations 34
Remediation plan: 34
3.9 (HAL-09) GLOBAL INDEX MANIPULATION - INFORMATIONAL 35
Description 35
Code Location 35
Risk Level 37
Recommendations 37
Remediation plan: 37
3.10 (HAL-10) INSUFFICIENT MARKETING DATA VALIDATION - INFORMATIONAL 38
Description 38
Code Location 38
Risk Level 39
Recommendations 39
Remediation plan: 39
3.11 (HAL-11) NO UNLOCKING MECHANISM FOR EMERGENCY SITUATIONS - INFORMATIONAL 40
Description 40

Code Location 40

Risk Level 41

Recommendation 41

Remediation plan 41

3.12 (HAL-12) POSSIBLE EXCESSIVE ACCESS TO REBALANCE FUNCTION - INFORMATIONAL 42

Description 42

Code Location 42

Risk Level 42

Recommendation 43

Remediation plan 43

3.13 (HAL-13) ADDRESS VALIDATION MISSING - INFORMATIONAL 44

Description 44

Code Location 44

Risk Level 44

Recommendations 45

Remediation plan 45

3.14 (HAL-14) OVERFLOW CHECKS NOT SET FOR PROFILE RELEASE - INFORMATIONAL 46

Description 46

Code Location 46

Risk Level 46

Recommendation 46

Remediation plan 47

3.15 (HAL-15) ARITHMETIC OVERFLOW - INFORMATIONAL 48

Description 48

Code Location 48

Risk Level 49

Recommendations 49

Remediation plan 50

3.16 (HAL-16) MINTER ADDRESS NOT UPDATEABLE - INFORMATIONAL 51

Description 51

Code Location 51

Risk Level 52

Recommendations 52

Remediation plan 52

3.17 (HAL-17) INACCURATE ERROR MESSAGES - INFORMATIONAL 53

Description 53

Code Location 53

Risk Level 53

Recommendation 54

Remediation plan 54

4 FUZZING 55

Introduction 56

Description 56

PoC 57

Results 57

5 AUTOMATED TESTING 58

5.1 VULNERABILITIES AUTOMATIC DETECTION 59

Description 59

Results 59

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	08/09/2021	Piotr Cielas
0.2	Document Updates	08/20/2021	Piotr Cielas
0.3	Document Updates	08/31/2021	Luis Quispe Gonzales
1.0	Final Version	09/03/2021	Luis Quispe Gonzales
1.1	Remediation Plan	09/16/2021	Luis Quispe Gonzales

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
Piotr Cielas	Halborn	Piotr.Cielas@halborn.com

EXECUTIVE OVERVIEW

1.1 AUDIT SUMMARY

Nexus Protocol engaged Halborn to conduct a security assessment on smart contracts beginning on August 9th, 2021 and ending September 3rd, 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 Nexus team. The main ones are the following:

- Update calculus of total stablecoin balance to distribute rewards without affecting vault funds.
- Have in consideration bAsset tokens previously transferred to vault into the calculus of nAsset tokens to mint.
- Reset loan repayment state when this repayment completes.
- Split privileged address transfer functionality to allow transfer to be completed by recipient.

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: basset-vault-contracts

   (b) Commit ID: 6e1244da15c8c9c5b660e5a93e9098966e83d23d

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	3	1	3	10

LIKELIHOOD

IMPACT

🟨	🟨	🟥	🟥	🟪
🟩	🟨	🟨	🟥 (HAL-02)	🟥 (HAL-01)
🟩 (HAL-06)	🟩 (HAL-05)	🟨 (HAL-04)	🟨	🟥 (HAL-03)
⬜ (HAL-08) (HAL-09) (HAL-10) (HAL-11) (HAL-12)	🟩	🟩	🟨	🟨
⬜ (HAL-13) (HAL-14) (HAL-15) (HAL-16) (HAL-17)	⬜	🟩 (HAL-07)	🟩	🟨

SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) MISCALCULATION OF BALANCE LEADS TO OVERRATED REWARDS DISTRIBUTION	High	SOLVED - 09/07/2021
(HAL-02) MISCALCULATION OF NASSET TOKENS TO MINT LEADS TO UNFAIR WITHDRAWING / DOS	High	SOLVED - 08/26/2021
(HAL-03) NO UPDATE OF LOAN REPAYMENT STATE ON REBALANCE FUNCTION	High	SOLVED - 08/26/2021
(HAL-04) PRIVILEGED ADDRESSES CAN BE TRANSFERRED WITHOUT CONFIRMATION	Medium	SOLVED - 09/14/2021
(HAL-05) DECIMAL RATES COULD BE UPDATED TO A VALUE GREATER OR EQUAL THAN 1	Low	SOLVED - 08/25/2021
(HAL-06) LOAN REPAYMENT COULD TAKE MORE ITERATIONS THAN ALLOWED	Low	SOLVED - 09/14/2021
(HAL-07) ROUNDING ISSUES WHEN DEPOSITING / WITHDRAWING BASSET TOKENS	Low	SOLVED - 09/12/2021
(HAL-08) UNAUTHORIZED TOKEN REWARDS CONTRACT ADDRESS MODIFICATION	Informational	ACKNOWLEDGED
(HAL-09) GLOBAL INDEX MANIPULATION	Informational	ACKNOWLEDGED
(HAL-10) INSUFFICIENT MARKETING DATA VALIDATION	Informational	ACKNOWLEDGED
(HAL-11) NO UNLOCKING MECHANISM FOR EMERGENCY SITUATIONS	Informational	ACKNOWLEDGED
(HAL-12) POSSIBLE EXCESSIVE ACCESS TO REBALANCE FUNCTION	Informational	ACKNOWLEDGED
(HAL-13) ADDRESS VALIDATION MISSING	Informational	ACKNOWLEDGED
(HAL-14) OVERFLOW CHECKS NOT SET FOR PROFILE RELEASE	Informational	SOLVED - 09/12/2021
(HAL-15) INTEGER OVERFLOW	Informational	ACKNOWLEDGED
(HAL-16) MINTER ADDRESS NOT UPDATEABLE	Informational	ACKNOWLEDGED
(HAL-17) INACCURATE ERROR MESSAGES	Informational	SOLVED - 09/12/2021

FINDINGS & TECH DETAILS

3.1 (HAL-01) MISCALCULATION OF BALANCE LEADS TO OVERRATED REWARDS DISTRIBUTION - HIGH

Description:

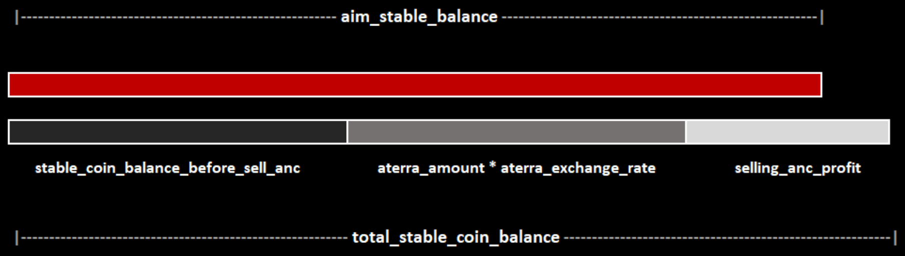
The split_profit_to_handle_interest function from **contracts/basset_vault/src/utils.rs** always miscalculates the value of total_stable_coin_balance, which produces the following consequences:

- The aforementioned function wrongly calculates rewards and overrates them, at expenses of vault funds.
- Decisions regarding rewards distribution will be wrongly made: BuyPsi, DepositToAnc or Split.
- Total balance could not reach aim_stable_balance, whereby the vault would not be able to repay loans to Anchor.

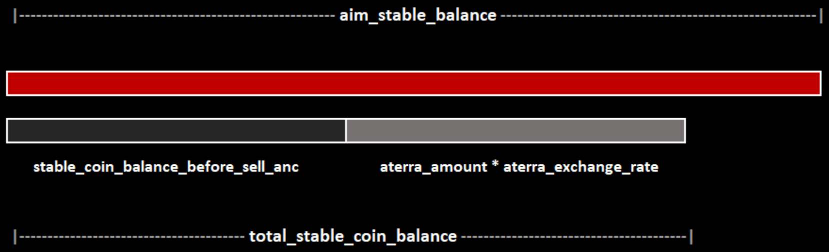
It is important to note that is not possible to revert, undo or correct the logic of the split_profit_to_handle_interest function, unless the vault contract is redeployed, which could lead to a significant loss of users' and Nexus vault's funds.

Attack scenario:

1. Total stablecoin balance always is miscalculated and wrongly considers selling_anc_profit in the calculus, see image below.


aim_stable_balance
stable_coin_balance_before_sell_anc
aterra_amount * aterra_exchange_rate
selling_anc_profit
total_stable_coin_balance

2. The split_profit_to_handle_interest function overrates rewards values and, when they are distributed, the remaining total balance could not reach aim_stable_balance as it should, see image below.


aim_stable_balance
stable_coin_balance_before_sell_anc
aterra_amount * aterra_exchange_rate
total_stable_coin_balance

Code Location:

Listing 1: contracts/basset_vault/src/utils.rs (Lines 434)

```
pub fn split_profit_to_handle_interest(
    borrowed_amount: Uint256,
    aterra_amount: Uint256,
    aterra_exchange_rate: Decimal256,
    stable_coin_balance: Uint256,
    stable_coin_balance_before_sell_anc: Uint256,
    over_loan_balance_value: Decimal256,
) -> ActionWithProfit {
    if stable_coin_balance <= stable_coin_balance_before_sell_anc
        {
        return ActionWithProfit::Nothing;
    }

    let total_stable_coin_balance = aterra_amount *
        aterra_exchange_rate + stable_coin_balance;
    let selling_anc_profit = stable_coin_balance -
        stable_coin_balance_before_sell_anc;

    let aim_stable_balance = borrowed_amount *
        over_loan_balance_value;
```

Risk Level:

Likelihood - 5
Impact - 4

Recommendation:

Update the total_stable_coin_balance formula not to include the value of **selling_anc_profit**. Below is a proposed sample formula:

$$total\_stable\_coin\_balance = aterra\_amount * aterra\_exchange\_rate + stable\_coin\_balance\_before\_sell\_anc$$

Remediation plan:

SOLVED: Issue fixed in commit cbaeb5bd108c030d5993145d080495c1ccf1719f.

3.2 (HAL-02) MISCALCULATION OF NASSET TOKENS TO MINT LEADS TO UNFAIR WITHDRAWING / DOS - HIGH

Description:

The deposit_basset function from **contracts/basset_vault/src/commands.rs** miscalculates the value of nasset_to_mint if someone has previously transferred bAsset tokens directly to the vault, which produces the following consequences:

- When a legitimate user deposits bAsset tokens to the vault, the nasset_to_mint value will be lower than it should be, which leads to unfair withdrawing.
- If nAsset supply is zero, legitimate users will not be able to deposit or withdraw bAssets anymore, which causes an unrecoverable denial of service (DoS) of the Nexus protocol.

It is important to note that it is not possible to revert, undo or correct the logic of the deposit_basset function, unless the vault contract is redeployed, which could lead to a significant loss of users' or Nexus vault's funds.

Attack scenario:

1. User #1 uses the deposit_basset function to deposit 2000 bLuna tokens into the vault and receives 2000 nAsset tokens in return.
2. Someone transfers 2000 bLuna tokens directly to the vault.
3. User #2 uses the deposit_basset function to deposit 2000 bLuna tokens in the vault and receives 1000 nAsset tokens in return, instead of 2000 like User #1.
4. User #2 uses the withdraw_basset function to withdraw all its bLuna, but receives 1333 bLuna tokens, instead of 2000 he had deposited.

Code Location:

Listing 2: contracts/basset_vault/src/commands.rs (Lines 169,178)

```
158  let basset_in_contract_address =
159      query_token_balance(deps.as_ref(), &config.basset_token, &env
             .contract.address)?;
160
161  let basset_balance: Uint256 = basset_in_custody +
         basset_in_contract_address.into();
162  if basset_balance == Uint256::zero() {
163      //impossible because 'farmer' already sent some basset
164      return Err(StdError::generic_err(
165          "basset balance is zero (impossible case)".to_string(),
166          ));
167  }
168  let farmer_basset_share: Decimal256 =
169      Decimal256::from_ratio(deposit_amount.0, basset_balance.0);
170
171  // nAsset tokens to mint:
172  // user_share = (deposited_basset / total_basset)
173  // nAsset_to_mint = nAsset_supply * user_share / (1 - user_share)
174  let nasset_to_mint = if farmer_basset_share == Decimal256::one()
         {
175      deposit_amount
176  } else {
177      // 'nasset_supply' can't be zero here, cause we already mint
             some for first farmer
178      nasset_supply * farmer_basset_share / (Decimal256::one() -
             farmer_basset_share)
179  };
```

Risk Level:

Likelihood - 4
Impact - 4

Recommendation:

Update deposit_amount value to include bAsset tokens directly transferred to the vault. Below are proposed sample formulas:

$$deposit\_amount = basset\_in\_contract\_address$$

$$nasset\_to\_mint = \frac{nasset\_supply * deposit\_amount}{basset\_balance - deposit\_amount}$$

Remediation plan:

SOLVED: Issue fixed in commit dd6c2467c5e1f7bdc90c7b87b3c7fb4ee8c6244d.

3.3 (HAL-03) NO UPDATE OF LOAN REPAYMENT STATE ON REBALANCE FUNCTION - HIGH

Description:

The rebalance function from **contracts/basset_vault/src/commands.rs** never resets repaying_loan_state value when loan repayment completes, so its **iteration_index** will continue increasing until reaching the maximum number of iterations allowed. Once it happens, loan repayments could be wrongly marked as completed or throw error messages, instead of iterating to actually complete the repayment.

It is important to note that is not possible to revert, undo or correct the logic of the rebalance function, unless the vault contract is redeployed, which could lead to a significant loss of users' or Nexus vault's funds.

Code Location:

Listing 3: contracts/basset_vault/src/commands.rs (Lines 389)

```
384 BorrowerActionResponse::Repay {
385             amount,
386             advised_buffer_size,
387 } => {
388        store_aim_buffer_size(deps.storage, &advised_buffer_size)?;
389        let mut repaying_loan_state = load_repaying_loan_state(deps
               .as_ref().storage)?;
390        repaying_loan_state.to_repay_amount = amount;
391        repaying_loan_state.aim_buffer_size = advised_buffer_size;
392        repay_logic(deps, env, config, repaying_loan_state)
393       }
```

Risk Level:

Likelihood - 5
Impact - 3

Recommendation:

Update the logic of rebalance function to reset repaying_loan_state value when loan repayment completes.

Remediation plan:

SOLVED: Issue fixed in commit c5714f1d8d73dac552820a3180cd598b33bb2bcb.

3.4 (HAL-04) 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 **governance contract address (owner)** using the aforementioned function in a single transaction and without confirmation from the new address.

The affected smart contracts are the following:

- basset_vault
- basset_vault_strategy
- nasset_token_config_holder
- nasset_token_rewards
- psi_distributor

Code Location:

Listing 4: contracts/basset_vault/src/commands.rs

```
if let Some(ref gov_addr) = gov_addr {
    current_config.governance_contract = deps.api.addr_validate(
        gov_addr)?;
}
```

Listing 5: contracts/basset_vault_strategy/src/commands.rs

```
if let Some(ref governance_addr) = governance_addr {
    current_config.governance_contract = deps.api.addr_validate(
        governance_addr)?;
}
```

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

```
if let Some(ref governance_addr) = governance_contract_addr {
    current_config.governance_contract = deps.api.addr_validate(
        governance_addr)?;
}
```

Listing 7: contracts/nasset_token_rewards/src/commands.rs

```
if let Some(ref governance_contract) = governance_contract {
    current_config.governance_contract = deps.api.addr_validate(
        governance_contract)?;
}
```

Listing 8: contracts/psi_distributor/src/commands.rs

```
if let Some(ref governance_contract_addr) =
    governance_contract_addr {
    current_config.governance_contract = deps.api.addr_validate(
        governance_contract_addr)?;
}
```

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 the following commits:

- 561ab303f514ea9b3d68940cbe4f864ccc7cce12

- 14e76524135074757aae4e635dc2c352d3c611ca
- 9523bf781294e8134aeb8bbdb152ea9ba90eeb49
- cf9709ba279507ae2951039fed8368f504184291
- b551a6cd48864af1874356a05cf5984aa6bc4e22

3.5 (HAL-05) DECIMAL RATES COULD BE UPDATED TO A VALUE GREATER OR EQUAL THAN 1 - LOW

Description:

The update_config function from **contracts/psi_distributor/src/commands.rs** changes all fields directly, so does not restrict that values of manual_ltv, fee_rate or tax_rate are greater or equal than 1.

The aforementioned values are used to calculate reward distribution and if are not correctly set, the formula will always panic and won't allow legitimate users to claim their rewards, thus generating a denial of service (DoS) in Nexus protocol.

Code Location:

Listing 9: contracts/psi_distributor/src/commands.rs

```rust
if let Some(manual_ltv) = manual_ltv {
    current_config.manual_ltv = manual_ltv;
}

if let Some(fee_rate) = fee_rate {
    current_config.fee_rate = fee_rate;
}

if let Some(tax_rate) = tax_rate {
    current_config.tax_rate = tax_rate;
}
```

Risk Level:

Likelihood - 2
Impact - 3

Recommendation:

Add a validation routine inside update_config function to ensure that values of manual_ltv, fee_rate and tax_rate are lesser than 1.

Remediation plan:

SOLVED: Issue fixed in commit 36c2395e68ee805426dccf46d6cb1e98f2bd3834.

3.6 (HAL-06) LOAN REPAYMENT COULD TAKE MORE ITERATIONS THAN ALLOWED - LOW

Description:

The set_buffer_part function from **contracts/basset_vault_strategy/src/state.rs** changes the value of buffer_part directly and does not verify if this value does not exceed the number of iterations allowed by the LOAN_REPAYMENT_MAX_RECURSION_DEEP constant.

If the number of iterations reaches its maximum value, the loan repayment will be wrongly marked as completed (because of previous partial payments), instead of throwing an error message for loan repayment failure.

Code Location:

Listing 10: contracts/basset_vault_strategy/src/state.rs

```
pub fn set_buffer_part(&mut self, value: Decimal256) ->
    ContractResult<()> {
    if value.is_zero() || value > Decimal256::one() {
        return Err(ContractError::InappropriateValue);
    }

    self.buffer_part = value;
    Ok(())
}
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

Add a validation routine inside set_buffer_part function to ensure that loan repayment will be able to complete within the number of iterations allowed. Below is a proposed sample validation for this routine:

$$buffer\_part >= \frac{max\_ltv - aim\_ltv}{LOAN\_REPAYMENT\_MAX\_RECURSION\_DEEP}$$

Remediation plan:

SOLVED: Issue fixed in commit cfe2ef43d778d2b363abd6bd5d287066bfe86aca.

3.7 (HAL-07) ROUNDING ISSUES WHEN DEPOSITING / WITHDRAWING BASSET TOKENS - LOW

Description:

When calculating nasset_to_mint in **deposit_basset** function and basset_to_withdraw in **withdraw_basset** function, the “multiply before divide” principle is not followed, which generates rounding issues.

Although the actual difference for rounding in each operation is very small, in the long run and with enough operations, it could cause a significant imbalance.

Code Location:

Calculating the nAsset tokens to mint:

Listing 11: contracts/basset_vault/src/commands.rs (Lines 169,178)

```
168 let farmer_basset_share: Decimal256 =
169     Decimal256::from_ratio(deposit_amount.0, basset_balance.0);
170
171 // nAsset tokens to mint:
172 // user_share = (deposited_basset / total_basset)
173 // nAsset_to_mint = nAsset_supply * user_share / (1 - user_share)
174 let nasset_to_mint = if farmer_basset_share == Decimal256::one() {
175     deposit_amount
176 } else {
177     // 'nasset_supply' can't be zero here, cause we already mint
            some for first farmer
178     nasset_supply * farmer_basset_share / (Decimal256::one() -
            farmer_basset_share)
179 };
```

Calculating the bAsset tokens to withdraw:

Listing 12: contracts/basset_vault/src/commands.rs (Lines 280,283)

```
let share_to_withdraw: Decimal256 = Decimal256::from_ratio(
    nasset_to_withdraw_amount.0,
    Uint256::from(nasset_token_supply).0,
);
let basset_to_withdraw: Uint256 = basset_in_custody *
    share_to_withdraw;
```

Risk Level:

Likelihood - 3
Impact - 1

Recommendation:

The formulas to calculate nasset_to_mint and basset_to_withdraw should be rewritten to reduce rounding issues. Below are proposed sample formulas:

$$nasset\_to\_mint = \frac{nasset\_supply * deposit\_amount}{basset\_balance - deposit\_amount}$$

$$basset\_to\_withdraw = \frac{basset\_in\_custody * nasset\_to\_withdraw\_amount}{nasset\_token\_supply}$$

Remediation plan:

SOLVED: Issue fixed in commit d80eef123844c614c9eb43180828ddc7ea8ac49c.

3.8 (HAL-08) UNAUTHORIZED TOKEN REWARDS CONTRACT ADDRESS MODIFICATION - INFORMATIONAL

Description:

The set_nasset_token_rewards_contract method in contracts/nasset_token_config_holder/src/state.rs can be used to update the contract configuration and set the nAsset rewards contract address if it hasn't been already set. This config option is world-writeable and can be accessed by sending a SetTokenRewardsContract message to the handler in contracts/nasset_token_config_holder/src/contract.rs. This may lead to unauthorized config modification and possible loss of users' rewards.

Code Location:

Listing 13: contracts/nasset_token_config_holder/src/contract.rs (Lines 42,47)

```
37 ExecuteMsg::Anyone { anyone_msg } => match anyone_msg {
38     AnyoneMsg::SetTokenRewardsContract {
39         nasset_token_rewards_contract_addr,
40     } => {
41         let config = load_config(deps.storage)?;
42         if config.nasset_token_rewards_contract.to_string().
               is_empty() {
43             let addr = deps
44                 .api
45                 .addr_validate(&nasset_token_rewards_contract_addr
                       )?;
46
47             set_nasset_token_rewards_contract(deps.storage, addr)
                   ?;
48
49             Ok(Response::default())
50         } else {
51             return Err(ContractError::Unauthorized {});
```

```
        }
```

Listing 14: contracts/nasset_token_config_holder/src/state.rs (Lines 20)

```
pub fn set_nasset_token_rewards_contract(
    storage: &mut dyn Storage,
    addr: Addr,
) -> StdResult<Config> {
    singleton(storage, KEY_CONFIG).update(|mut cfg: Config| ->
        StdResult<_> {
        cfg.nasset_token_rewards_contract = addr;
        Ok(cfg)
    })
}
```

Risk Level:

Likelihood - 1
Impact - 2

Recommendations:

All sensitive operations on configuration data should require prior authorization in order not to be modified by malicious individuals.

Remediation plan::

ACKNOWLEDGED: the Nexus team acknowledged this finding. This nasset_config_holder contract is initialized on the bAsset token contract initialization which makes this vulnerability highly unlikely to be exploited.

3.9 (HAL-09) GLOBAL INDEX MANIPULATION - INFORMATIONAL

Description:

The calculate_global_index function defined in contracts/nasset_token_rewards/src/commands.rs determines the index which is then used to calculate the rewards distributed to users. This function is indirectly available from the update_global_index function which can be called by sending an anonymous UpdateGlobalIndex message to the handler in contracts/nasset_token_rewards/src/contract.rs.

This global_index can be manipulated by sending rewards to the nasset_token_rewards contract with the claim_rewards and claim_rewards_for_someone functions.

Code Location:

Listing 15: contracts/nasset_token_rewards/src/commands.rs (Lines 75,89,90,91,92)

```
69 fn calculate_global_index(
70     deps: Deps,
71     env: Env,
72     config: &Config,
73     state: &mut State,
74 ) -> ContractResult<Uint128> {
75     let balance = query_token_balance(deps, &config.psi_token, &
           env.contract.address)?;
76
77     let previous_balance = state.prev_reward_balance;
78
79     // claimed_rewards = current_balance - prev_balance;
80     let claimed_rewards = balance.checked_sub(previous_balance)?;
81
82     if claimed_rewards.is_zero() || state.total_balance.is_zero()
           {
83         return Ok(claimed_rewards);
84     }
85
86     state.prev_reward_balance = balance;
87
88     // global_index += claimed_rewards / total_balance;
89     state.global_index = decimal_summation_in_256(
90         state.global_index,
91         Decimal::from_ratio(claimed_rewards, state.total_balance),
92     );
93
94     Ok(claimed_rewards)
95 }
```

Listing 16: contracts/nasset_token_rewards/src/commands.rs (Lines 51)

```
41 pub fn update_global_index(deps: DepsMut, env: Env) ->
       ContractResult<Response> {
42     let mut state: State = load_state(deps.storage)?;
43
44     // Zero nasset balance check
45     if state.total_balance.is_zero() {
46         return Err(StdError::generic_err("nAsset balance is zero")
               .into());
47     }
48
49     let config = load_config(deps.storage)?;
50
51     let claimed_rewards = calculate_global_index(deps.as_ref(),
           env, &config, &mut state)?;
```

Listing 17: contracts/nasset_token_rewards/src/contract.rs (Lines 52)

```
43 #[entry_point]
44 pub fn execute(
45     deps: DepsMut,
46     env: Env,
47     info: MessageInfo,
48     msg: ExecuteMsg,
49 ) -> ContractResult<Response> {
50     match msg {
51         ExecuteMsg::Anyone { anyone_msg } => match anyone_msg {
52             AnyoneMsg::UpdateGlobalIndex {} => commands::
                   update_global_index(deps, env),
```

Risk Level:

Likelihood - 1
Impact - 2

Recommendations:

Both claim_rewards and claim_rewards_for_someone should block sending PSI tokens to the nasset_token_reward contract in order to prevent global index manipulation.

Remediation plan::

ACKNOWLEDGED: the Nexus team acknowledged this finding. global_index manipulation could incur losses to the attacker which makes this vulnerability less likely to be exploited.

3.10 (HAL-10) INSUFFICIENT MARKETING DATA VALIDATION - INFORMATIONAL

Description:

The basset-vault-nasset-token contract allows the contract owner to upload the nAsset token marketing information, including description and logo. This data is stored on-chain and can be modified via relevant utility functions of the cw20-base contract. The cw-20 contract however does not validate the marketing data sufficiently making it possible to insert descriptions of arbitrary length and content as well as set arbitrary URLs or update XML files for the logo.

Code Location:

Listing 18: contracts/nasset_token/src/contract.rs (Lines 9)

```
6 use cw20_base::allowances::{execute_decrease_allowance,
     execute_increase_allowance};
7 use cw20_base::contract::instantiate as cw20_instantiate;
8 use cw20_base::contract::query as cw20_query;
9 use cw20_base::contract::{execute_update_marketing,
     execute_upload_logo};
10 use cw20_base::msg::{ExecuteMsg, InstantiateMsg as
     TokenInstantiateMsg, QueryMsg};
```

Listing 19: contracts/nasset_token/src/contact.rs (Lines 100,109)

```
96  ExecuteMsg::UpdateMarketing {
97      project,
98      description,
99      marketing,
100 } => Ok(execute_update_marketing(
101     deps,
102     env,
103     info,
104     project,
    description,
    marketing,
)?),

ExecuteMsg::UploadLogo(logo) => Ok(execute_upload_logo(deps, env,
    info, logo)?),
```

Risk Level:

Likelihood - 1
Impact - 2

Recommendations:

All external data should be validated to match the expected size and content, for example only HTTP or HTTPS logo URLs should be allowed.

Remediation plan::

ACKNOWLEDGED: the Nexus team acknowledged this finding. Since this is an owner-only function it is highly unlikely to be abused.

3.11 (HAL-11) NO UNLOCKING MECHANISM FOR EMERGENCY SITUATIONS - INFORMATIONAL

Description:

The deposit_basset and withdraw_basset functions from **contracts/basset_-vault/src/commands.rs** lock the vault contract if there are no **bAssets** in custody but some circulating **nAssets** exist. Although this scenario is highly unlikely, if it happens legitimate users will not be able to interact (deposit or withdraw) with the vault contract anymore and it will be kept locked forever because there are no mechanisms to manage emergency situations (e.g.: unlock contract).

Code Location:

Conditional expression in deposit_basset function that locks contract if there are no **bAssets** in custody but exist some **nAssets**:

Listing 20: contracts/basset_vault/src/commands.rs

```
if basset_in_custody.is_zero() && !nasset_supply.is_zero() {
    //read comments in 'withdraw_basset' function for a reason to
        return error here
    return Err(StdError::generic_err(
            "bAsset balance is zero, but nAsset supply is not!
                Freeze contract.",
    ));
}
```

Conditional expression in withdraw_basset function that locks contract if there are no **bAssets** in custody:

Listing 21: contracts/basset_vault/src/commands.rs

```
if basset_in_custody.is_zero() {
    //interesting case - user owns some nAsset, but bAsset balance
```

```
            is zero
    //what we can do here:
    //1. Burn his nAsset, cause they do not have value in that
        context
    //2. return error. In that case if someone will deposit bAsset
         those nAsset owners will
    //   own share of his tokens. But I prevent deposists in that
        case, so contract is kinds "frozen" -
    //   no withdraw and deposits available when bLuna balance is
        zero. Looks like the best
    //   solution.
    //3. Burn all nAsset supply (not possible with cw20 messages)
    //
    //Second choice is best one in my opinion.
    return Err(StdError::generic_err(
            "bAsset balance is zero, but nLuna supply is not!
               Freeze contract.",
    ));
}
```

Risk Level:

Likelihood - 1
Impact - 2

Recommendation:

It is recommended to add locking / unlocking mechanisms in contract to manage emergency situations, such as handling unexpected states on contract, loss or steal of tokens, etc. The aforementioned mechanisms should be only accessible to **Emergency** role.

Remediation plan:

ACKNOWLEDGED: Nexus team has stated that having **bAssets** in custody when exist some circulating **nAssets** is a extremely remote situation, and in case it happens and locks the vault contract, they prefer to redeploy a new one because of existing data corruption.

3.12 (HAL-12) POSSIBLE EXCESSIVE ACCESS TO REBALANCE FUNCTION - INFORMATIONAL

Description:

When a user deposits bAsset tokens, rebalance function is called as part of regular operations through Anyone::Rebalance message, which allows that anyone can call the function anytime.

Because there is no need that external users or smart contracts other than **vault contract** call the aforementioned function, it is important to apply the principle of least privilege in this case.

Code Location:

Listing 22: contracts/basset_vault/src/contract.rs (Lines 286)

```
AnyoneMsg::Rebalance {} => {
    let config = load_config(deps.storage)?;

    // basset balance in custody contract
    let basset_in_custody = get_basset_in_custody(
        deps.as_ref(),
        &config.anchor_custody_basset_contract,
        &env.contract.address.clone(),
    )?;

    commands::rebalance(deps, env, &config, basset_in_custody,
        None)
}
```

Risk Level:

Likelihood - 1
Impact - 2

Recommendation:

It is recommended to update logic of Anyone::Rebalance message to restrict that only **vault contract** could call rebalance function.

Remediation plan:

ACKNOWLEDGED: Nexus team stated that rebalance function can be called by anyone by design. Also, they will try to achieve decentralization by sharing script that periodically calls Anyone::Rebalance message with community.

3.13 (HAL-13) ADDRESS VALIDATION MISSING - INFORMATIONAL

Description:

One thing the holder_to_response function defined in contracts/nasset_token_rewards/src/queries.rs does is it converts an array of bytes to an account address. This function assumes that the array of bytes provided contains a valid address but does not verify if that's actually true.

Code Location:

Listing 23: contracts/nasset_token_rewards/src/queries.rs (Lines 118)

```
pub fn holder_to_response(
    holder_with_address: StdResult<(Vec<u8>, Holder)>,
) -> StdResult<HolderResponse> {
    let (addr_bytes, holder) = holder_with_address?;
    let address = std::str::from_utf8(&addr_bytes)?.to_string();

    Ok(HolderResponse {
        address,
        balance: holder.balance,
        index: holder.index,
        pending_rewards: holder.pending_rewards,
    })
}
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendations:

Consider using the addr_validate utility function to check if the address recovered is a valid one.

Remediation plan:

ACKNOWLEDGED: the Nexus team acknowledged this finding. Currently, all data retrieved with this function is validated on insert.

3.14 (HAL-14) 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 contract and package, which could lead to unexpected consequences if the project is refactored.

Code Location:

Listing 24: Resources affected

```
contracts/basset_vault/Cargo.toml
contracts/basset_vault_strategy/Cargo.toml
contracts/nasset_token/Cargo.toml
contracts/nasset_token_config_holder/Cargo.toml
contracts/nasset_token_rewards/Cargo.toml
contracts/psi_distributor/Cargo.toml
packages/basset_vault/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:

SOLVED: Issue fixed in commit 1402dcb673fc7e9b9ae2df72a92019ef2938378b.

3.15 (HAL-15) ARITHMETIC OVERFLOW - INFORMATIONAL

Description:

An overflow happens when an arithmetic operation reaches the maximum size of a type. For instance in the decimal_multiplication_in_256() method, two Decimal256 values are multiplied which may end up overflowing the type. In computer programming, an overflow occurs when an arithmetic operation attempts to create a numeric value that is outside of the range that can be represented with a given number of bits -- either larger than the maximum or lower than the minimum representable value.

Code Location:

Listing 25: contracts/nasset_token_rewards/src/math.rs (Lines 8)

```rust
pub fn decimal_multiplication_in_256(a: Decimal, b: Decimal) ->
    Decimal {
    let a_u256: Decimal256 = a.into();
    let b_u256: Decimal256 = b.into();
    let c_u256: Decimal = (b_u256 * a_u256).into();
    c_u256
}
```

Listing 26: contracts/nasset_token_rewards/src/math.rs (Lines 16)

```rust
pub fn decimal_summation_in_256(a: Decimal, b: Decimal) -> Decimal
    {
    let a_u256: Decimal256 = a.into();
    let b_u256: Decimal256 = b.into();
    let c_u256: Decimal = (b_u256 + a_u256).into();
    c_u256
}
```

Listing 27: contracts/nasset_token_rewards/src/math.rs (Lines 24)

```rust
pub fn decimal_subtraction_in_256(a: Decimal , b: Decimal) ->
    Decimal {
     let a_u256: Decimal256 = a.into();
     let b_u256: Decimal256 = b.into();
     let c_u256: Decimal = (a_u256 - b_u256).into();
     c_u256
}
```

Listing 28: contracts/nasset_token_rewards/src/commands.rs (Lines 193,194)

```rust
holder.index = state.global_index;
holder.pending_rewards = decimal_summation_in_256(rewards, holder.
    pending_rewards);
holder.balance += amount;
state.total_balance += amount;

calculate_global_index(deps.as_ref(), env, &config, &mut state)?;
}
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendations:

In the “release” mode Rust does not panic on overflows and overflown values just “wrap” without any explicit feedback to the user. It is recommended then 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.

Remediation plan:

ACKNOWLEDGED: The overflow-checks parameter is set to **true** in profile.release and implicitly applied to all contracts and packages in the workspace. The contract is going to panic if overflow happens.

3.16 (HAL-16) MINTER ADDRESS NOT UPDATEABLE - INFORMATIONAL

Description:

The nasset_token contract is largely based on the standard cw20 token contract. A standard cw20 token contract introduces a minter role which cannot be updated after the token contract is instantiated. Thus, if the minter address is not set on nasset_token instantiation (or is set to an incorrect address) all the cw20 minter-only features will be unavailable indefinitely.

Code Location:

Listing 29: contracts/nasset_token_/src/contract.rs (Lines 31)

```
12 #[entry_point]
13 pub fn instantiate(
14     deps: DepsMut,
15     env: Env,
16     info: MessageInfo,
17     msg: InstantiateMsg,
18 ) -> ContractResult<Response> {
19     let config_holder_contract = deps.api.addr_validate(&msg.
           config_holder_contract)?;
20     save_config_holder_contract(deps.storage, &
           config_holder_contract)?;
21
22     cw20_instantiate(
23         deps,
24         env,
25         info,
26         TokenInstantiateMsg {
27             name: msg.name,
28             symbol: msg.symbol,
29             decimals: msg.decimals,
30             initial_balances: msg.initial_balances,
31             mint: msg.mint,
32             marketing: msg.marketing,
33         },
    )?;

    Ok(Response::default())
}
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendations:

Validate the mint parameter has Some value and implement a utility governance-only function to update the minter address if necessary.

Remediation plan:

ACKNOWLEDGED: This vulnerability is highly unlikely to affect the contract since by default nasset_token is instantiated from the basset_vault contract. The minter is always set to the basset_vault contract address.

3.17 (HAL-17) INACCURATE ERROR MESSAGES - INFORMATIONAL

Description:

Error messages shown in certain sections of code have inaccurate information, which could mislead legitimate users if these messages appear during a failed operation with the Nexus protocol.

Code Location:

Error message should state "... **bigger or equal to one**".

Listing 30: contracts/psi_distributor/src/contract.rs (Lines 27)

```
return Err(StdError::generic_err(
    "none of decimal numbers can be bigger or equal to zero",
)
.into());
```

Error message should state "... **but nAsset suply is not! ...**"

Listing 31: contracts/basset_vault/src/commands.rs (Lines 275)

```
return Err(StdError::generic_err(
    "bAsset balance is zero, but nLuna supply is not! Freeze
        contract.",
));
```

Risk Level:

Likelihood - 1
Impact - 1

Recommendation:

Correct error messages to show more accurate information and to avoid confusing users if these messages appear.

Remediation plan:

SOLVED: Issue fixed in commits 9824dc093494925dffba74aa60be9ff0e3eedbc6 and c770d69c135659c2c16d585435b17e542e3ba470.

FUZZING

Introduction:

Fuzzing or fuzz testing is an automated software testing technique that involves providing invalid, unexpected, or random data as inputs to a computer program. The program is then monitored for exceptions such as crashes, failing built-in code assertions, or potential memory leaks.

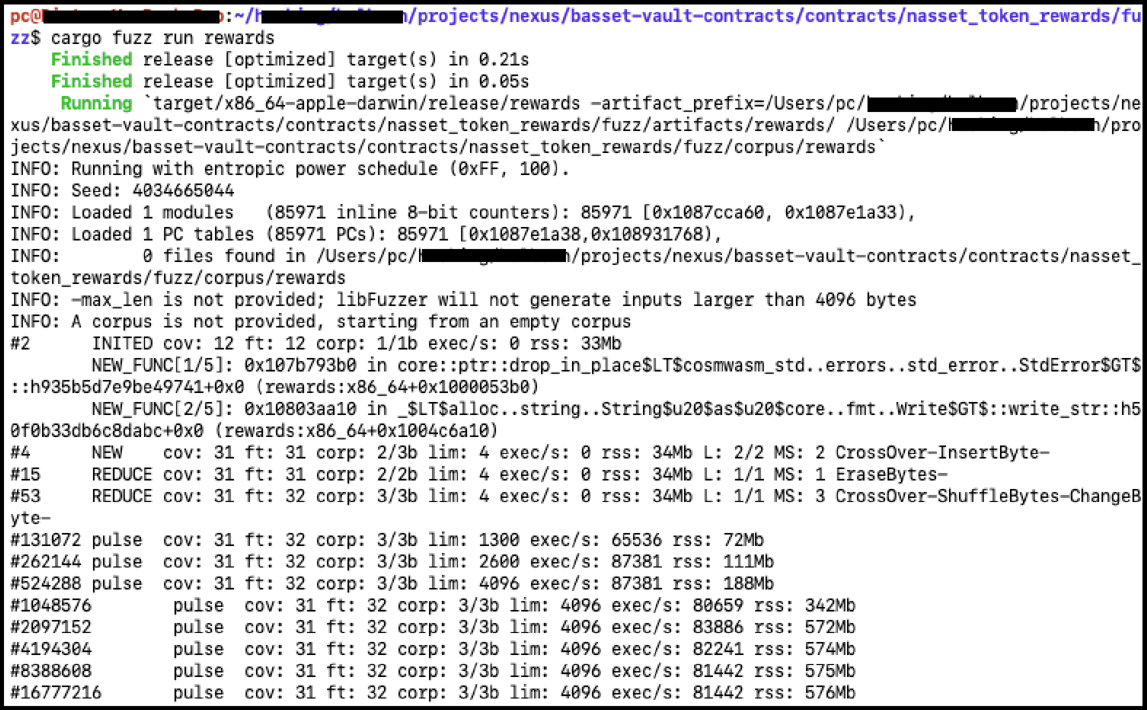
Halborn custom-built scripts leverage libFuzzer and cargo-fuzz for in-process, coverage-guided fuzz testing.

The fuzzer tracks which areas of the code are reached, and generates mutations on the corpus of input data in order to maximize the code coverage. The code coverage information is provided by LLVM's SanitizerCoverage instrumentation.

Description:

Halborn used custom fuzzing scripts, tailored to the specifics of Substrate and the Cere protocol. The methods targeted were the ones accepting vectors of bytes as input because they are potentially most likely to be vulnerable to memory-related and indexing issues.

PoC:



```
pc@[redacted]:~/h[redacted]n/projects/nexus/basset-vault-contracts/contracts/nasset_token_rewards/fu
zz$ cargo fuzz run rewards
    Finished release [optimized] target(s) in 0.21s
    Finished release [optimized] target(s) in 0.05s
     Running `target/x86_64-apple-darwin/release/rewards -artifact_prefix=/Users/pc/[redacted]/projects/ne
xus/basset-vault-contracts/contracts/nasset_token_rewards/fuzz/artifacts/rewards/ /Users/pc/H[redacted]h/pro
jects/nexus/basset-vault-contracts/contracts/nasset_token_rewards/fuzz/corpus/rewards`
INFO: Running with entropic power schedule (0xFF, 100).
INFO: Seed: 4034665044
INFO: Loaded 1 modules   (85971 inline 8-bit counters): 85971 [0x1087cca60, 0x1087e1a33),
INFO: Loaded 1 PC tables (85971 PCs): 85971 [0x1087e1a38,0x108931768),
INFO:        0 files found in /Users/pc/H[redacted]n/projects/nexus/basset-vault-contracts/contracts/nasset_
token_rewards/fuzz/corpus/rewards
INFO: -max_len is not provided; libFuzzer will not generate inputs larger than 4096 bytes
INFO: A corpus is not provided, starting from an empty corpus
#2      INITED cov: 12 ft: 12 corp: 1/1b exec/s: 0 rss: 33Mb
        NEW_FUNC[1/5]: 0x107b793b0 in core::ptr::drop_in_place$LT$cosmwasm_std..errors..std_error..StdError$GT$
::h935b5d7e9be49741+0x0 (rewards:x86_64+0x1000053b0)
        NEW_FUNC[2/5]: 0x10803aa10 in _$LT$alloc..string..String$u20$as$u20$core..fmt..Write$GT$::write_str::h5
0f0b33db6c8dabc+0x0 (rewards:x86_64+0x1004c6a10)
#4      NEW    cov: 31 ft: 31 corp: 2/3b lim: 4 exec/s: 0 rss: 34Mb L: 2/2 MS: 2 CrossOver-InsertByte-
#15     REDUCE cov: 31 ft: 31 corp: 2/2b lim: 4 exec/s: 0 rss: 34Mb L: 1/1 MS: 1 EraseBytes-
#53     REDUCE cov: 31 ft: 32 corp: 3/3b lim: 4 exec/s: 0 rss: 34Mb L: 1/1 MS: 3 CrossOver-ShuffleBytes-ChangeB
yte-
#131072 pulse  cov: 31 ft: 32 corp: 3/3b lim: 1300 exec/s: 65536 rss: 72Mb
#262144 pulse  cov: 31 ft: 32 corp: 3/3b lim: 2600 exec/s: 87381 rss: 111Mb
#524288 pulse  cov: 31 ft: 32 corp: 3/3b lim: 4096 exec/s: 87381 rss: 188Mb
#1048576        pulse  cov: 31 ft: 32 corp: 3/3b lim: 4096 exec/s: 80659 rss: 342Mb
#2097152        pulse  cov: 31 ft: 32 corp: 3/3b lim: 4096 exec/s: 83886 rss: 572Mb
#4194304        pulse  cov: 31 ft: 32 corp: 3/3b lim: 4096 exec/s: 82241 rss: 574Mb
#8388608        pulse  cov: 31 ft: 32 corp: 3/3b lim: 4096 exec/s: 81442 rss: 575Mb
#16777216       pulse  cov: 31 ft: 32 corp: 3/3b lim: 4096 exec/s: 81442 rss: 576Mb
```

Results:

Between the time constraints and lack of advanced memory manipulation in the source code **no issues were identified at this time**.

AUTOMATED TESTING

5.1 VULNERABILITIES AUTOMATIC DETECTION

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. 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.

Results:

Package	**ID**	**Short Description**
bigint	RUSTSEC-2020-0025	bigint is unmaintained, use uint instead

THANK YOU FOR CHOOSING

// HALBORN