Mars Protocol - Red Bank

CosmWasm Smart Contract Security Audit

Prepared by: **Halborn**

Date of Engagement: **October 3rd, 2022 - October 21st, 2022**

Visit: **Halborn.com**

DOCUMENT REVISION HISTORY

VERSION	MODIFICATION	DATE	AUTHOR
0.1	Document Creation	10/03/2022	Michal Bazyli
0.2	Document Update	10/17/2022	Michal Bazyli
0.3	Draft Version	10/21/2022	Michal Bazyli
0.4	Draft Review	10/25/2022	Gabi Urrutia
0.5	Document Update	10/27/2022	Michal Bazyli
1.0	Remediation Plan	10/31/2022	Michal Bazyli
1.1	Remediation Plan Review	11/01/2022	Gabi Urrutia

CONTACTS

CONTACT	COMPANY	EMAIL
Rob Behnke	Halborn	Rob.Behnke@halborn.com
Steven Walbroehl	Halborn	Steven.Walbroehl@halborn.com
Gabi Urrutia	Halborn	Gabi.Urrutia@halborn.com

Luis Quispe Gonzales	Halborn	Luis.QuispeGonzales@halborn.com
Michal Bazyli	Halborn	Michal.Bazyli@halborn.com

EXECUTIVE OVERVIEW

1.1 INTRODUCTION

Mars Protocol engaged Halborn to conduct a security audit on their smart contracts beginning on October 3rd, 2022 and ending on October 21st, 2022. The security assessment was scoped to the smart contracts provided in the GitHub repository outposts, commit hashes and further details can be found in the **Scope** section of this report.

1.2 AUDIT SUMMARY

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

The purpose of this audit is to:

- Ensure that smart contract functions operate as intended
- Identify potential security issues with the smart contracts

In summary, Halborn identified some security risks which were accepted by the Mars Protocol team:

- Ensure that there are no open positions in the markets before modifying them.
- Add a safe transfer ownership logic, preferably in a form of a two -steps process.
- Enhance security of the contract by implementing validation routines when initializing assets or updating them.

1.3 TEST APPROACH & METHODOLOGY

Halborn performed a combination of manual and automated security testing to balance efficiency, timeliness, practicality, and accuracy in regard to the scope of this audit. While manual testing is recommended to uncover flaws in logic, process, and implementation; automated testing techniques help enhance coverage of the Rust code 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 and purpose.
- Smart contract manual code review and walkthrough.
- Manual testing by custom scripts and fuzzers.
- Scanning of Rust files for vulnerabilities, security hotspots or bugs.
- Static Analysis of security for scoped contract, and imported functions.
- Testnet deployment.

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

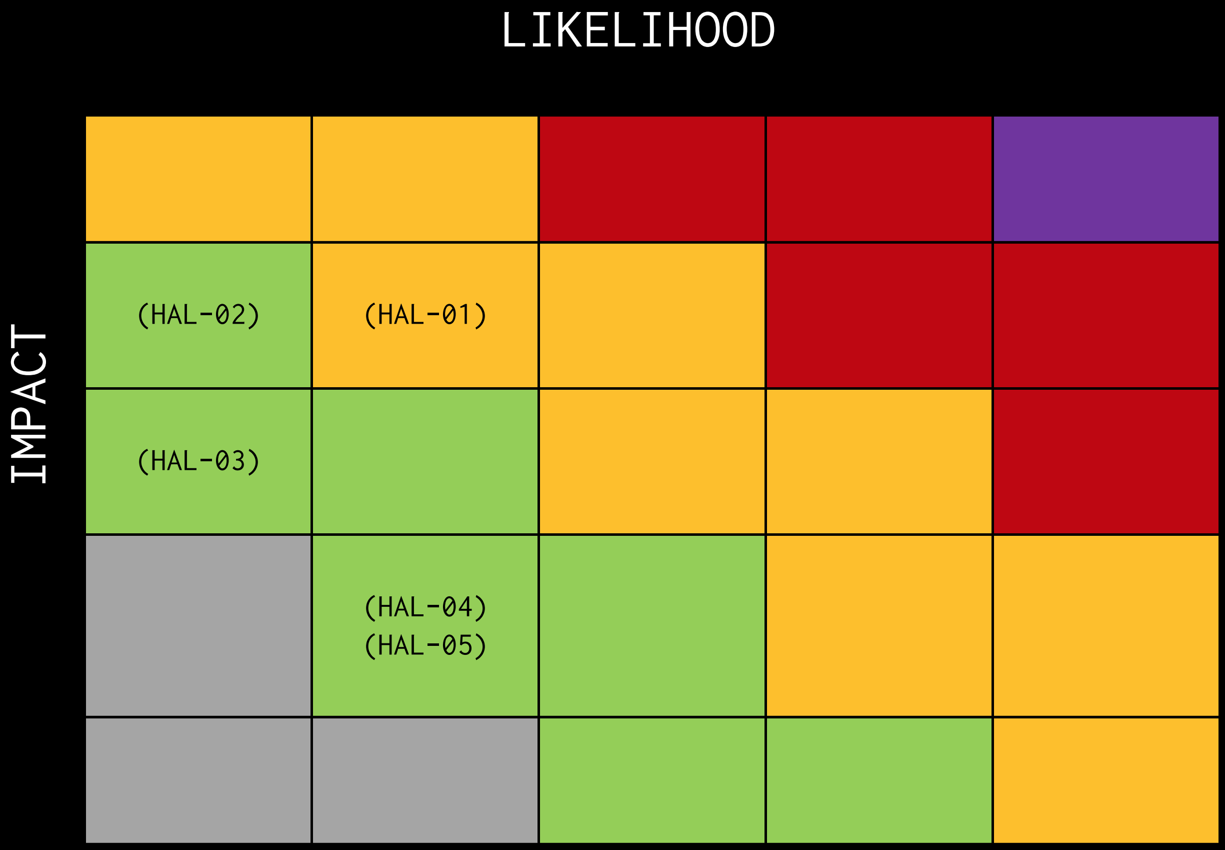
1.4 SCOPE

1. CosmWasm Smart Contracts
   (a) Repository: outpost
   (b) Commit ID: e476501a784c78de1b7f350722febe6d77d3a35d
   (c) Contracts in scope:
      i. red-bank
   (d) Packages in scope:
      i. outpost

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

2. ASSESSMENT SUMMARY & FINDINGS OVERVIEW

CRITICAL	HIGH	MEDIUM	LOW	INFORMATIONAL
0	0	1	4	0



SECURITY ANALYSIS	RISK LEVEL	REMEDIATION DATE
(HAL-01) MARKET VALUES CAN BE UPDATED WHEN POSITIONS ARE OPENED	Medium	RISK ACCEPTED
(HAL-02) PRIVILEGED ADDRESS CAN BE TRANSFERRED WITHOUT CONFIRMATION	Low	RISK ACCEPTED
(HAL-03) MISSING VALIDATION ROUTINE FOR CRITICAL MARKET PARAMETERS	Low	RISK ACCEPTED
(HAL-04) MARKETS CANNOT BE REMOVED	Low	RISK ACCEPTED
(HAL-05) LIQUIDATION CAN LEAD TO BAD DEBT	Low	RISK ACCEPTED

FINDINGS & TECH DETAILS

3.1 (HAL-01) MARKET VALUES CAN BE UPDATED WHEN POSITIONS ARE OPENED - MEDIUM

Description:

update_asset function in **contracts/red-bank/src/execute.rs** allows updating the market values without previously checking if there are open positions. Consequently, users with collateralized debts and health positions may be liquidated.

It is worth noting that likelihood for this to happen is limited because **red-bank** contract is intended to be owned by multi-sign wallet, who is the responsible one for this operation, however this could happen mistakenly.

Code Location:

Fragment of update_asset:

Listing 1: contracts/red-bank/src/execute.rs (Line 267)

```
            let mut updated_market = Market {
                max_loan_to_value: max_loan_to_value.unwrap_or(
 ↳ market.max_loan_to_value),
                reserve_factor: reserve_factor.unwrap_or(market.
 ↳ reserve_factor),
                liquidation_threshold: liquidation_threshold
                    .unwrap_or(market.liquidation_threshold),
                liquidation_bonus: liquidation_bonus.unwrap_or(
 ↳ market.liquidation_bonus),
                interest_rate_model: interest_rate_model.unwrap_or
 ↳ (market.interest_rate_model),
                deposit_enabled: deposit_enabled.unwrap_or(market.
 ↳ deposit_enabled),
                borrow_enabled: borrow_enabled.unwrap_or(market.
 ↳ borrow_enabled),
                deposit_cap: deposit_cap.unwrap_or(market.
 ↳ deposit_cap),
                ..market
            };

            updated_market.validate()?;

            if should_update_interest_rates {
                response = update_interest_rates(
                    &deps,
                    &env,
                    &mut updated_market,
                    Uint128::zero(),
                    &denom,
                    response,
                )?;
            }
            MARKETS.save(deps.storage, &denom, &updated_market)?;

            Ok(response
                .add_attribute("action", "outposts/red-bank/
↳ update_asset")
                .add_attribute("denom", &denom))
        }
    }
}
```

Risk Level:

Likelihood - 2
Impact - 4

Recommendation:

Update the logic of update_asset function to restrict that there is no open positions in current market. Otherwise, it should throw an error message.

Remediation plan:

RISK ACCEPTED: The Mars Protocol team accepted the risk of this finding.

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

Description:

An incorrect use of the update_config function from the **red-bank** contract could set the owner to an invalid address, unwillingly losing control of the contract, which cannot be undone in any way. Currently, the **owner** of the contracts can change its address using the aforementioned function in a single transaction and without confirmation from the new address.

Code Location:

Fragment of update_config:

Listing 2: contracts/red-bank/src/execute.rs (Line 82)

```rust
    pub fn update_config(
        deps: DepsMut,
        info: MessageInfo,
        new_config: CreateOrUpdateConfig,
    ) -> Result<Response, ContractError> {
        let mut config = CONFIG.load(deps.storage)?;

        if info.sender != config.owner {
            return Err(MarsError::Unauthorized {}.into());
        }

        // Destructuring a structs fields into separate variables
 ↳ in order to force
        // compile error if we add more params
        let CreateOrUpdateConfig {
            owner,
            address_provider,
            close_factor,
        } = new_config;

        // Update config
        config.owner = option_string_to_addr(deps.api, owner, config.owner)?;
        config.address_provider =
            option_string_to_addr(deps.api, address_provider, config.address_provider)?;
        config.close_factor = close_factor.unwrap_or(config.close_factor);

        // Validate config
        config.validate()?;

        CONFIG.save(deps.storage, &config)?;

        Ok(Response::new().add_attribute("action", "outposts/red-bank/update_config"))
    }
```

Risk Level:

Likelihood - 1
Impact - 4

Recommendation:

The update_config function should follow a two steps process, being split into set_owner and accept_owner functions. The latter one requiring the transfer to be completed by the recipient, effectively protecting the contract against potential typing errors compared to single-step ownership transfer mechanisms.

Remediation plan:

RISK ACCEPTED: The Mars Protocol team accepted the risk of this finding.

3.3 (HAL-03) MISSING VALIDATION ROUTINE FOR CRITICAL MARKET PARAMETERS - LOW

Description:

The current implementation of validate function for Market struct validates only is:

- reserve_factor lesser than one.
- max_loan_to_value lesser or equal one.
- liquidation_threshold lesser or equal one.
- liquidation_bonus lesser or equal one.

However, the validation routines for the minimal value of max_loan_to_value, minimal value liquidation_threshold and maximum value for reserve_factor and liquidation_bonus are not enforced.

Code Location:

Fragment of init_asset:

Listing 3: contracts/red-bank/src/execute.rs

```rust
    pub fn init_asset(
    deps: DepsMut,
    env: Env,
    info: MessageInfo,
    denom: String,
    params: InitOrUpdateAssetParams,
) -> Result<Response, ContractError> {
    let config = CONFIG.load(deps.storage)?;

    if info.sender != config.owner {
        return Err(MarsError::Unauthorized {}.into());
    }

    if MARKETS.may_load(deps.storage, &denom)?.is_some() {
        return Err(ContractError::AssetAlreadyInitialized {});
    }

    let new_market = create_market(env.block.time.seconds(), &
↳ denom, params)?;
    MARKETS.save(deps.storage, &denom, &new_market)?;

    Ok(Response::new()
        .add_attribute("action", "outposts/red-bank/init_asset")
        .add_attribute("denom", denom))
}
```

Fragment of validate:

Listing 4: packages/outpost/src/red_bank/market.rs

```
impl Market {
    pub fn validate(&self) -> Result<(), MarsError> {
        decimal_param_lt_one(self.reserve_factor, "reserve_factor"
↳ )?;
        decimal_param_le_one(self.max_loan_to_value, "
↳ max_loan_to_value")?;
        decimal_param_le_one(self.liquidation_threshold, "
↳ liquidation_threshold")?;
        decimal_param_le_one(self.liquidation_bonus, "
↳ liquidation_bonus")?;

        // liquidation_threshold should be greater than
↳ max_loan_to_value
        if self.liquidation_threshold <= self.max_loan_to_value {
            return Err(MarsError::InvalidParam {
                param_name: "liquidation_threshold".to_string(),
                invalid_value: self.liquidation_threshold.
↳ to_string(),
                predicate: format!("> {} (max LTV)", self.
↳ max_loan_to_value),
            });
        }

        self.interest_rate_model.validate()?;

        Ok(())
    }
```

Risk Level:

Likelihood - 1
Impact - 3

Recommendation:

It is recommended to add a validation routine inside market.validate functions to ensure that:

- reserve_factor is not greater than a hardcoded maximum value.
- max_loan_to_value is not greater than a hardcoded maximum value and not lesser than a minimum one.
- liquidation_threshold is not greater than a hardcoded maximum value and not lesser than a minimum one.
- liquidation_bonus is not greater than a hardcoded maximum value.

Remediation plan:

RISK ACCEPTED: The Mars Protocol team accepted the risk of this finding.

3.4 (HAL-04) MARKETS CANNOT BE REMOVED - LOW

Description:

There is no possibility to remove markets. In case an error occurs, or a delisting is required, there will be no possibility to remove a market. Consequently, any party that is still relying on the market, may keep receiving data on a market on which it should not. Also, over time the storage might grow in an uncontrolled way, thus the ability to clear some entries may be useful.

Code Location:

Fragment of update_asset:

Listing 5: contracts/red-bank/src/execute.rs (Line 267)

```rust

            let mut updated_market = Market {
                max_loan_to_value: max_loan_to_value.unwrap_or(market.max_loan_to_value),
                reserve_factor: reserve_factor.unwrap_or(market.reserve_factor),
                liquidation_threshold: liquidation_threshold
                    .unwrap_or(market.liquidation_threshold),
                liquidation_bonus: liquidation_bonus.unwrap_or(market.liquidation_bonus),
                interest_rate_model: interest_rate_model.unwrap_or(market.interest_rate_model),
                deposit_enabled: deposit_enabled.unwrap_or(market.deposit_enabled),
                borrow_enabled: borrow_enabled.unwrap_or(market.borrow_enabled),
                deposit_cap: deposit_cap.unwrap_or(market.deposit_cap),
                ..market
            };

            updated_market.validate()?;

            if should_update_interest_rates {
                response = update_interest_rates(
                    &deps,
                    &env,
                    &mut updated_market,
                    Uint128::zero(),
                    &denom,
                    response,
                )?;
            }
            MARKETS.save(deps.storage, &denom, &updated_market)?;

            Ok(response
                .add_attribute("action", "outposts/red-bank/update_asset")
                .add_attribute("denom", &denom))
        }
    }
}
```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

Consider adding a functionality to remove markets.

Remediation plan:

RISK ACCEPTED: The Mars Protocol team accepted the risk of this finding.

3.5 (HAL-05) LIQUIDATION CAN LEAD TO BAD DEBT - LOW

Description:

liquidate function in **contracts/red-bank/src/execute.rs** allows the liquidator or bot to liquidate unhealthy market positions. However, when the collateral amount to liquidate is higher than the user collateral balance, the liquidator can liquidate the position by paying the borrower collateral discounted by a liquidation bonus and receiving 100% of the borrower's collateral leaving the bad debt. This exposes the protocol to unnecessary risk of MEV attacks.

Bad debt can accrue, more users will be unable to withdraw their funds. And due to the design of the protocols, the last to withdraw will realize the loss.

Proof of Concept:

- Alice is depositing collateral of 200$ in uosmo.
- Alice Borrow 100$ in utoken.
- The price of utoken go up 2 times.
- Alice's Debt is now 200$ which is 3x from Liquidation threshold.
- Bob is Liquidating Alice by paying 180$ in utoken.
- Bob receives 100% of Alice's position collateral by paying 90% of Alice collateral.
- Alice's debt is -120$.
- Bob extract 10% of the collateral from the protocol.
- Alice is not repaying the debt which makes it grow in time, in consequences it affects the total utilization rate .

Asset config for uosmo:

Listing 6

```
AssetConfig = {
  denom: 'uosmo',
  initial_borrow_rate: '0.1',
  max_loan_to_value: '0.5',
  reserve_factor: '0.02',
  liquidation_threshold: '0.60',
  liquidation_bonus: '0.1',
  interest_rate_model: {
    optimal_utilization_rate: '1.0',
    base: '0.05',
    slope_1: '0.0',
    slope_2: '0.0',
  },
  deposit_cap: '340282366920938463463374607431768211455',
  deposit_enabled: true,
  borrow_enabled: true,
  symbol: 'OSMO',
  }
```

Asset config for utoken:

Listing 7

```
AssetConfig = {
  denom: 'utoken',
  initial_borrow_rate: '0.1',
  max_loan_to_value: '0.6',
  reserve_factor: '0.03',
  liquidation_threshold: '1.0',
  liquidation_bonus: '0.0',
  interest_rate_model: {
    optimal_utilization_rate: '1.0',
    base: '0.05',
    slope_1: '0.0',
    slope_2: '0.0',
  },
  deposit_cap: '340282366920938463463374607431768211455',
  deposit_enabled: true,
  borrow_enabled: true,
  symbol: 'TOKEN',
  }
```

Code Location:

Fragment of liquidation_compute_amounts:

Listing 8: contracts/red-bank/src/execute.rs

```
    if collateral_amount_to_liquidate_scaled >
    ↳ user_collateral_amount_scaled {
        collateral_amount_to_liquidate_scaled =
    ↳ user_collateral_amount_scaled;
        collateral_amount_to_liquidate =
    ↳ get_underlying_liquidity_amount(
            collateral_amount_to_liquidate_scaled,
            collateral_market,
            block_time,
        )?;
        debt_amount_to_repay = math::divide_uint128_by_decimal(
            math::divide_uint128_by_decimal(
                collateral_amount_to_liquidate * collateral_price,
                debt_price,
            )?,
            Decimal::one() + collateral_market.liquidation_bonus,
        )?;
    }

```

Risk Level:

Likelihood - 2
Impact - 2

Recommendation:

We suggest an alternative solution to the current implementation which will lower the risk of being targeted by an MEV attack.
When the collateral amount to liquidate is higher than the user collateral balance, incentives should be calculated from repay debt and extracted from the user collateral if any left or offer lower liquidation spread when liquidations such a position.

This will lower the profits from such an edge case scenarios of liquidation, making the protocol less attractive for MEV attacks and the protocol will have more collateral to cover.

References:

- https://arxiv.org/pdf/2106.06389.pdf
- https://docs.compound.finance/liquidation

Remediation plan:

RISK ACCEPTED: The Mars Protocol team accepted the risk of this finding.

THANK YOU FOR CHOOSING

// HALBORN