CERTIK

Security Assessment

Shade Protocol

Aug 1st, 2022

Table of Contents

Summary

This report has been prepared for Shade Protocol to discover issues and vulnerabilities in the source code of the Shade Protocol project as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

- Enhance general coding practices for better structures of source codes;
- Add enough unit tests to cover the possible use cases;
- Provide more comments per each function for readability, especially contracts that are verified in public;
- Provide more transparency on privileged activities once the protocol is live.

Overview

Project Summary

Project Name	Shade Protocol
Platform	CosmWasm
Language	Rust
Codebase	https://github.com/securesecrets/shade/tree/dev/contracts/bonds
Commit	• 55ae240a32bb1440c8324dc109c17e4b5210f8e5 • 746b3505014cc3b87b44ae72d4eb04c6abeb29d6

Audit Summary

Delivery Date	Aug 01, 2022 UTC
Audit Methodology	Static Analysis, Manual Review

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Mitigated	Partially Resolved	Resolved
Critical	0	0	0	0	0	0	0
Major	1	0	0	1	0	0	0
Medium	1	0	0	1	0	0	0
Minor	2	0	0	1	0	0	1
Informational	11	0	0	3	0	0	8
Discussion	0	0	0	0	0	0	0

Audit Scope

ID	File	SHA256 Checksum
CON	src/contract.rs	bd827dbd4fa7d95bd98c65a1bead22bec7d3a4746030bf5bfef99d9c65e15e18
HAN	src/handle.rs	2f930b53e9e891aec36090ab2bc3a5d29e89b1cfb5310b6753e385e91c9aa33c
LIB	src/lib.rs	e8502a218cba6fbed031254e77eb9cce6f456b7c8197db22376a2a5da2339eef
QUE	src/query.rs	67f3018c746f41f16ac2757e3f701a4532efd14049e228505c3c7d0b549643e7
STA	src/state.rs	42cfd1c6e8eeb501568b73984435957e5ca46e51a1df28559fdecf6fdd40401e

Review Notes

Shade Protocol is an array of connected privacy-preserving dApps built on Secret Network.
The `bonds` module of **Shade Protocol** is mainly used to issue bonds for assets. Privileged roles can add bond exchange opportunities for specific assets. Users can deposit assets with bond opportunities to exchange bonds. After the bond claim period, users can claim the issue assets. The `bonds` module consists of the following four contracts:

- contract
- handle
- query
- state

The `contract` contract acts as the controller of the **Shade Protocol**, which includes bond configuration initialization, message processing, and data query functions.

The `handle` contract is the core business contract, it contains some privileged functions. Privileged roles can modify the bond configuration and add bond exchange opportunities for assets. Users can store assets for exchange and receive bonds.

The `query` contract provides data query functions for other contracts.

The `state` contract defines the keys to store and query, and encapsulates the input and return data structures uniformly.

External Dependencies

There are a few depending injection contracts or addresses in the current project:

- Oracle: provide asset price query
- ValidateAdminPermission: provide authentication function to verify whether the account has admin permission
- SetViewingKey: set a key for the contract for viewing
- TokenInfo: query the basic information of the issued asset, including name, symbol, decimals, etc.
- TokenConfig: query the configuration information of the issued asset
- CompleteTask: create an account for the user and send this message to update the airdrop status

We assume these vulnerable actors are implementing proper logic to collaborate with the current project.

Privileged Functions

Contract `bonds/src/handle.rs`

the role `limit_admin` has authority over the following function:

- try_update_limit_config(): update limit configuration of bonds

the role `admin` has authority over the following function:

- try_update_config(): update configuration of bonds
- try_open_bond(): add bond opportunities for Snip20 assets
- try_close_bond(): cancel bond opportunities for Snip20 assets

To improve the trustworthiness of the project, dynamic runtime updates in the project should be notified to the community. Any plan to invoke the aforementioned functions should be also considered to move to the execution queue of the `Timelock` contract.

Findings

Critical	**0** (0.00%)
Major	**1** (6.67%)
Medium	**1** (6.67%)
Minor	**2** (13.33%)
Informational	**11** (73.33%)
Discussion	**0** (0.00%)

ID	Title	Category	Severity	Status
CON-01	Design Of Block Size For Padding Message	Logical Issue	Informational	Resolved
HAN-01	Centralization Related Risks	**Centralization / Privilege**	**Major**	Acknowledged
HAN-02	Assumption On Oracle Price Decimals	Logical Issue	Medium	Acknowledged
HAN-03	Assets Are Sent To Different Addresses	Logical Issue	Informational	Acknowledged
HAN-04	`unwrap()` Function Is Discouraged	Logical Issue	Informational	Acknowledged
HAN-05	Compensation For Deposit Asset Price Exceeding Maximum Price	Logical Issue	Informational	Acknowledged
HAN-06	Parameter Used To Create Accounts	Logical Issue	Informational	Resolved
HAN-07	Could Add More Details To Events	Coding Style	Informational	Resolved
HAN-08	Unnecessary Borrow	Coding Style	Informational	Resolved
HAN-09	Unnecessary `clone()`	Coding Style	Informational	Resolved
HAN-10	Unnecessary `let` Binding	Coding Style	Informational	Resolved
QUE-01	Unnecessary `return`	Coding Style	Informational	Resolved
SRC-01	External Dependency	Logical Issue	Minor	Acknowledged
SRC-02	Lack Of Input Validation	Logical Issue	Minor	Resolved

ID	Title	Category	Severity	Status
SRC-03	Usage Of Magic Number	Coding Style	Informational	Resolved

CON-01 | Design Of Block Size For Padding Message

Category	Severity	Location	Status
Logical Issue	Informational	src/contract.rs: 70	Resolved

Description

In the `contract.rs` contract, the `init()`, `handle()` and `query()` functions will pad the message to blocks of a certain size (1 and 256).

```
68      let token_info = token_info_query(
69          &deps.querier,
70          1,
71          state.issued_asset.code_hash.clone(),
72          state.issued_asset.address.clone(),
73      )?;
74
75      let token_config = token_config_query(
76          &deps.querier,
77          256,
78          state.issued_asset.code_hash.clone(),
79          state.issued_asset.address.clone(),
80      )?;
```

The audit team has the following questions about the block size setting:

1. The `TokenInfo` message is padded to the block of size 1, is the size sufficient?
2. What is the size of the padded blocks based on?

Recommendation

The audit team would like to confirm that the aforementioned padding size is indeed the same by design.

Alleviation

[Shade Protocol]:

- That `token_info_query` is getting its padded value changed to 256, so as to stay consistent with other query sizes.
- The size of the padded blocks refers to what size you want the message to be rounded to. Passing 256, for example, means the message (msg) will be padded with spaces until msg.lenth() % 256 == 0.

[Certik]: The Shade Protocol team set the padding size to be consistent as `RESPONSE_BLOCK_SIZE` in the commit 746b3505014cc3b87b44ae72d4eb04c6abeb29d6, and the padding size of `token_info_query` is `256`.

HAN-01 | Centralization Related Risks

Category	Severity	Location	Status
Centralization / Privilege	**Major**	src/handle.rs: 36, 95, 445, 598	Acknowledged

Description

In the contract `bonds/src/handle.rs`, the role `limit_admin` has authority over the following function:

- try_update_limit_config(): update limit configuration of bonds.

In the contract `bonds/src/handle.rs`, the role `admin` has authority over the following function:

- try_update_config(): update configuration of bonds.
- try_open_bond(): add bond opportunities for Snip20 asset.
- try_close_bond(): cancel bond opportunities for Snip20 asset.

Any compromise to the privileged accounts may allow a hacker to take advantage of this to update project configurations and issue or close bonds.

Recommendation

The risk describes the current project design and potentially makes iterations to improve in the security operation and level of decentralization, which in most cases cannot be resolved entirely at the present stage. We advise the client to carefully manage the privileged account's private key to avoid any potential risks of being hacked. In general, we strongly recommend centralized privileges or roles in the protocol be improved via a decentralized mechanism or smart-contract-based accounts with enhanced security practices, e.g., multi-signature wallets.

Indicatively, here are some feasible suggestions that would also mitigate the potential risk at a different level in terms of short-term, long-term and permanent:

Short Term:

Timelock and Multi sign (⅔, ⅗) combination *mitigate* by delaying the sensitive operation and avoiding a single point of key management failure.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to the private key compromised;

AND

- A medium/blog link for sharing the timelock contract and multi-signers addresses information with the public audience.

Long Term:

Timelock and DAO, the combination, *mitigate* by applying decentralization and transparency.

- Time-lock with reasonable latency, e.g., 48 hours, for awareness on privileged operations;
  AND
- Introduction of a DAO/governance/voting module to increase transparency and user involvement;
  AND
- A medium/blog link for sharing the timelock contract, multi-signers addresses, and DAO information with the public audience.

Permanent:

Renouncing the ownership or removing the function can be considered *fully resolved*.

- Renounce the ownership and never claim back the privileged roles;
  OR
- Remove the risky functionality.

Alleviation

[Shade Protocol]: Issue acknowledged. We're discussing the introduction of a timelock mechanism for later use, but don't intend to include it in this audit engagement.

In regards to decentralized mechanisms, we currently have a multi-signature wallet on Secret Network that is going to be holding admin privileges, with 5/7 signatures on the wallet required for transactions. The "Add Migration Info" button above these comments didn't seem to have an option to provide evidence for Secret Network addresses.

This admin privilege will be renounced and transferred over to the Shade Protocol Governance/voting module when the bulk of the Protocol launches later this year. Specifically within that Governance, the Limit Admin role will be held by one Governance Assembly (functioning similarly to a multi-signature wallet requiring a majority of signatures to act), and the Admin role will be held by a different Bonds Governance Assembly (another multi-signature wallet with a majority of signatures needed to act).

This Governance module, as well as the fact that our current multi-signature wallet will never claim back the privileged role once it transfers to Protocol Governance, is our main effort to implement decentralized mechanisms and prevent bad actors from compromising the contract.

HAN-02 | Assumption On Oracle Price Decimals

Category	Severity	Location	Status
Logical Issue	Medium	src/handle.rs: 801~802	Acknowledged

Description

In `handle.rs`, the `calculate_issuance()` function calculates the amount of issued asset to bond based on the price of the deposit asset and the issued asset, which is obtained by the oracle.

```
807  let issued_amount = deposit_amount.multiply_ratio(deposit_price, discount_price);
808      let difference: i32 = i32::from(issued_decimals)
809          .checked_sub(i32::from(deposit_decimals))
810          .unwrap();
811      match difference.cmp(&0) {
812          Ordering::Greater => (
813              issued_amount
814
.checked_mul(Uint128::new(10u128.pow(u32::try_from(difference).unwrap())))
815                  .unwrap(),
816              discount_price,
817          ),
818          Ordering::Less => (
819              issued_amount
820                  .multiply_ratio(1u128,
10u128.pow(u32::try_from(difference.abs()).unwrap())),
821              discount_price,
822          ),
823          Ordering::Equal => (issued_amount, discount_price),
824      }
```

The function `calculate_issuance()` handles the decimal inconsistency for the token assets, but it doesn't consider the potential inconsistency of assets' price coming from the oracle. It assumes that prices from the oracle have the same 18 decimals, which may not always be true. For instance, in Chainlink, the decimal of the token price that is paired with USD has the decimal of 8, but the decimal is 18 when the token is paired with ETH. If assets' prices have different decimals, the function will return the incorrect amount of issued assets, which could bring unexpected loss to the project or the users.

Recommendation

Recommend adding certain logic to handle the inconsistency of price decimals when the oracle price may have different decimals.

Alleviation

[Shade Protocol]: We are planning to handle the potential inconsistency of different asset decimal counts via our oracle. As our protocol's oracle pulls the asset prices from whatever feeds it may, the oracle will execute the logic to convert it to 18 decimals so that other contracts such as this one can stay consistent with 18.

[Certik]: This solution can solve the problem, but if the contract references other oracles with inconsistent price decimals, it still faces the problem. We still recommend processing price decimals in the contract, which can be decoupled from oracles, replacing other oracles can also avoid this problem.

[Shade Protocol]: We don't plan to ever use this contract with other oracles, and the decimal count will be controlled within our oracle. We'll keep the recommendation in mind for future iterations of bonds that may interact with multiple oracles, though.

HAN-03 | Assets Are Sent To Different Addresses

Category	Severity	Location	Status
Logical Issue	Informational	src/handle.rs: 333, 343	Acknowledged

Description

In `handle.rs`, users can deposit an asset that has a bond opportunity through the `try_deposit()` function. If the contract cannot mint a bond for the deposit asset, the contract will transfer the issued asset to `env.contract.address`; if it can mint new tokens, the contract will mint the issued asset to `config.contract`.

```
325     if !bond_opportunity.minting_bond {
326         // Decrease AllocatedAllowance since user is claiming
327         allocated_allowance_w(&mut deps.storage)
328             .update(|allocated|
Ok(allocated.checked_sub(amount_to_issue.clone())?))?;
329 
330         // Transfer funds using allowance to bonds
331         messages.push(transfer_from_msg(
332             config.treasury.clone(),
333             env.contract.address.clone(),
334             amount_to_issue.into(),
335             None,
336             None,
337             256,
338             config.issued_asset.code_hash.clone(),
339             config.issued_asset.address,
340         )?);
341     } else {
342         messages.push(mint_msg(
343             config.contract,
344             amount_to_issue.into(),
345             None,
346             None,
347             256,
348             config.issued_asset.code_hash,
349             config.issued_asset.address,
350         )?);
351     }
```

For assets that are only for non-minting bond opportunities, the issued asset is transferred from the treasury to `env.contract.address`; for the minting bond opportunities, the issued asset is minted to the address `config.contract`.

Later, users can claim the issued asset through `try_claim()` function after the bond maturity, and the issued asset would be transferred to users from the bond contract.

```
    messages.push(send_msg(
        env.message.sender,
        total.into(),
        None,
        None,
        None,
        256,
        config.issued_asset.code_hash.clone(),
        config.issued_asset.address,
    )?);
```

Since the issued asset is sent to different addresses and later the issued asset is transferred from another address to the users. Therefore, users may fail to claim the issued asset unless `env.contract.address == config.contract`, which is also the bind contract.

Recommendation

The audit team would like to confirm that the aforementioned three addresses are indeed the same by design.

Alleviation

[Shade Protocol]: The aforementioned addresses are indeed the same by design. `config.contract` is set equal to `env.contract.address` in the `contract.rs` `init()` function, and cannot be reset anywhere else. If it would be best practice to use `config.contract` contract over `env.contract.address` or vice-versa, it can certainly be changed.

HAN-04 | `unwrap()` Function Is Discouraged

Category	Severity	Location	Status
Logical Issue	Informational	src/handle.rs: 234, 400, 800, 808, 812~813, 818, 829, 888, 891, 894	Acknowledged

Description

The function `unwrap(self) -> T` will give the embedded `T` if there is one. If instead there is not a `T` but an `E` or None, then it will panic.

Since the `unwrap()` function may cause panic, its usage is generally discouraged. Instead, it is preferred to use pattern matching and handle the error case explicitly, and it is better to call `unwrap_or()`, `unwrap_or_else()`, or `unwrap_or_default()`.

Recommendation

Recommend using the functions `unwrap_or()`, `unwrap_or_else()` or `unwrap_or_default()` functions instead of the `unwrap()` function.

Alleviation

[Shade Protocol]: Issue acknowledged. We're very confident that those uses of unwrap will not panic, and would prefer them to panic rather than use unwrap_or, unwrap_or_else, or unwrap_or_default.

HAN-05 | Compensation For Deposit Asset Price Exceeding Maximum Price

Category	Severity	Location	Status
Logical Issue	Informational	src/handle.rs: 756	Acknowledged

Description

In `handle.rs`, the `amount_to_issue()` function calculates the amount of issued assets that can be redeemed based on the price and amount of the deposit assets and the price of the issue asset.

```
749    if deposit_price > max_accepted_deposit_price {
750          if deposit_price > err_deposit_price {
751              return Err(deposit_price_exceeds_limit(
752                  deposit_price.clone(),
753                  err_deposit_price.clone(),
754              ));
755          }
756          deposit_price = max_accepted_deposit_price;
757      }
```

However, if the price of the deposit asset exceeds `max_accepted_deposit_price` set by the admin, the price of the deposit asset is set to the predefined `max_accepted_deposit_price`, which will reduce the number of issued assets that the user can claim.

The audit team has the following questions:

1. Why set `max_accepted_deposit_price` for deposit assets?
2. If the price of the deposit asset exceeds `max_accepted_deposit_price`, is there any compensation provided to the user?

Recommendation

The auditing team would like to gain a deeper understanding of the design from the responses to the above questions.

Alleviation

[Shade Protocol]: `max_accepted_deposit_price` is used for deposit assets that fluctuate around a peg so as to keep the DAO from trading at a significant loss in the event the asset's value has risen above the peg.

This also allows users to still use their pegged asset to enter a bond opportunity during one of these minor price events, rather than barring them access until their asset's value drops closer to its peg.

No compensation is provided to the user if the price of the deposit asset exceeds the `max_accepted_deposit_price`, but it will be communicated to the user ahead of time via the front end of the application what value their deposit asset will have during issuance calculation and they can elect to proceed or not.

HAN-06 | Parameter Used To Create Accounts

Category	Severity	Location	Status
Logical Issue	Informational	src/handle.rs: 300	Resolved

Description

In `handle.rs`, users can deposit asset which is available for a bond opportunity through the `try_deposit()` function, and the contract will create the pending bond and account data to record users' deposit action, as shown below:

```
299     // Find user account, create if it doesn't exist
300     let mut account = match
account_r(&deps.storage).may_load(sender.as_str().as_bytes())? {
301         None => {
302             // Airdrop task
303             if let Some(airdrop) = config.airdrop {
304                 let msg = CompleteTask {
305                     address: sender.clone(),
306                     padding: None,
307                 };
308                 messages.push(msg.to_cosmos_msg(airdrop.code_hash, airdrop.address,
None)?);
309             }
310
311             Account {
312                 address: sender,
313                 pending_bonds: vec![],
314             }
315         }
316         Some(acc) => acc,
317     };
```

where the `sender` is considered as the initiator of this function who deposits asset tokens to buy the bonds.

However, the `try_deposit()` function is handled through a `Receive` message in the `contract.rs`:

```
186             HandleMsg::Receive {
187                 sender,
188                 from,
189                 amount,
190                 msg,
191                 ..
192             } => handle::try_deposit(deps, &env, sender, from, amount, msg),
```

where `sender` is the address of the sender and `from` is the owner of the funds since the `receive` message is defined as follows:

```
{
  "receive": {
    "sender": "address of the sender",
    "from": "address of the owner of the funds",
    "amount": "funds that were sent as Uint128",
    "msg": "custom message, optional"
  }
}
```

The `sender` and `from` fields may be different if the `receive` message is sent using the `SendFrom` message, and they will be the same when sent by a `Send` call.

Recommendation

The auditing team would like to confirm that it is the intended design to use the `sender` as the owner of the bond.

Alleviation

[Shade Protocol]: After discussion with the team working with the DAO, we decided that using the "from" address for account creation closer fits our intended design. This would make the owner of the funds the owner of the account. The change is reflected in the commit 746b3505014cc3b87b44ae72d4eb04c6abeb29d6.

HAN-07 | Could Add More Details To Events

Category	Severity	Location	Status
Coding Style	Informational	src/handle.rs: 86, 170	Resolved

Description

In `handle.rs`, the functions `try_update_limit_config()` and `try_update_config()` do not provide details of execution messages. It would be better if the contract could emit detailed events so that the users can easily check the on-chain state updates through events.

Recommendation

Recommend adding more detailed descriptions of the key state variables when updating.

Alleviation

[Shade Protocol]: The team heeded the advice and details about updated config fields have been added to `HandleResponse` of related functions in the commit a8156129f8139c6736ced074960edea71566ee4d.

HAN-08 | Unnecessary Borrow

Category	Severity	Location	Status
Coding Style	Informational	src/handle.rs: 223, 241, 520, 748	Resolved

Description

The references on the linked lines would be dereferenced immediately by the compiler, so the borrow operations are unnecessary.

For example,

```
223             bond_active(&env, &prev_opp)?;
```

can be

```
223             bond_active(env, &prev_opp)?;
```

Recommendation

Recommend removing the aforementioned unnecessary borrow operations.

Alleviation

[Shade Protocol]: The team heeded the advice and resolved this issue in the commit a8156129f8139c6736ced074960edea71566ee4d.

HAN-09 | Unnecessary `clone()`

Category	Severity	Location	Status
Coding Style	Informational	src/handle.rs: 257, 289, 328, 418, 486, 527, 552, 639, 752, 753, 761, 762, 770	Resolved

Description

The following variables are of type `cosmwasm_math_compat::Uint128` which implements the `Copy` trait, so the `clone()` is redundant.

- `amount_to_issue` and `msg.minimum_expected_amount` in line 257
- `amount_to_issue` in line 289 and line 328
- `total` in line 418
- `unspent` in line 486 and line 639
- `deposit_price` in line 752
- `err_deposit_price` in line 753
- `issued_price` in line 761
- `err_issued_price` in line 762
- `deposit_price` in line 770

The following variables are the values that are dropped without any further use, so the `clone()` is redundant.

- `env.contract.address` in line 527
- `deposit_asset` in line 552

Recommendation

Recommend removing the `clone()` from the aforementioned variables.

Alleviation

[Shade Protocol]: The team heeded the advice and resolved this issue in the commit a8156129f8139c6736ced074960edea71566ee4d.

HAN-10 | Unnecessary `let` Binding

Category	Severity	Location	Status
Coding Style	Informational	src/handle.rs: 841	Resolved

Description

It is extraneous to return the result of a `let` binding from a block. Removing the `let` binding would make the code follow the coding conventions and be more rusty.

For example,

```
841     let end = env_time.checked_add(bonding_period).unwrap();
842
843     end
```

can be

```
841     env_time.checked_add(bonding_period).unwrap()
```

Recommendation

Consider returning the expression directly.

Alleviation

[Shade Protocol]: The team heeded the advice and resolved this issue in the commit a8156129f8139c6736ced074960edea71566ee4d.

QUE-01 | Unnecessary `return`

Category	Severity	Location	Status
Coding Style	Informational	src/query.rs: 51, 73~75, 83	Resolved

Description

It is extraneous to use `return` to return the result. Removing the `return` would make the code follow the coding conventions and be more rusty.

For example,

```
51          _ => return Err(query_auth_bad_response()),
```

can be

```
51          _ => Err(query_auth_bad_response()),
```

Recommendation

Consider removing `return` on the aforementioned lines.

Alleviation

[Shade Protocol]: The team heeded the advice and resolved this issue in the commit a8156129f8139c6736ced074960edea71566ee4d.

SRC-01 | External Dependency

Category	Severity	Location	Status
Logical Issue	Minor	src/contract.rs: 59~66, 68~74, 75~81; src/handle.rs: 114, 199, 303, 461, 606, 748, 758	Acknowledged

Description

The bonds module refers to some external contract dependencies which are not within the audit scope. The external contract dependencies are as follows:

- `oracle` :
  `GetPrice` message is sent to `oracle` to query the price of deposit and issued assets, the price obtained determines the exchange ratio between assets.
- `shade_admin`:
  `ValidateAdminPermission` message is sent to `shade_admin` to verify whether the caller has the admin permission, accounts with admin privileges can modify the bond configuration.
- `issued_asset`:
  1. `SetViewingKey` message is sent to set a key for the contract for viewing.
  2. `TokenInfo` message is sent to query the basic information of the issued asset, including name, symbol, decimals, etc.
  3. `TokenConfig` message is sent to query the configuration of the issued asset.
- `airdrop`:
  `CompleteTask` message is sent to update the airdrop status of `sender`.

These external dependencies provide important data for the bond module, and it is necessary to ensure the security of this partial dependency.

Recommendation

Recommend checking the logic of the external dependency contracts and ensuring their security.

Alleviation

[Shade Protocol]: Issue acknowledged. We're making a point to ensure the logic and security of these dependencies.

SRC-02 | Lack Of Input Validation

Category	Severity	Location	Status
Logical Issue	Minor	src/contract.rs: 44, 46; src/handle.rs: 105~106	Resolved

Description

In the state of `config`, `global_minimum_bonding_period` restricts the minimum of the `bond_period` and `global_maximum_discount` restricts the maximum of `bond_discount`. The input validations in `init()` and `try_update_config()` functions for these two parameters are missing.

- `msg.bond_period` and `msg.bond_discount` in `init()` function;
- `bond_period` and `bond_discount` in `try_update_config()` function.

Due to the lack of input validations, the bond module may be prone to errors in runtime.

Recommendation

Recommend adding verifications as below:

contract.rs

```
29        if msg.bonding_period < msg.global_minimum_bonding_period {
30          return Err(bonding_period_below_minimum_time(
31              msg.bonding_period,
32              msg.global_minimum_bonding_period,
33          ));
34        }
35
36        if msg.discount > msg.global_maximum_discount {
37          return Err(bond_discount_above_maximum_rate(
38              msg.discount,
39              msg.global_maximum_discount,
40          ));
41        }
```

handle.rs

```
 97        if bonding_period < state.global_minimum_bonding_period {
 98          return Err(bonding_period_below_minimum_time(
 99              bonding_period,
100              state.global_minimum_bonding_period,
101          ));
102        }
```

```
103
104         if discount > state.global_maximum_discount {
105           return Err(bond_discount_above_maximum_rate(
106               discount,
107               state.global_maximum_discount,
108           ));
109         }
```

Alleviation

[Shade Protocol]: The team heeded the advice and resolved this issue in the commit a8156129f8139c6736ced074960edea71566ee4d.

SRC-03 | Usage Of Magic Number

Category	Severity	Location	Status
Coding Style	Informational	src/contract.rs: 62, 77; src/handle.rs: 279, 337, 347, 429, 529, 850	Resolved

Description

To avoid data leaks, Secret Contracts can enforce constant length messages via padding. Such output of transactions can include the following: callbacks to another contract call, contract init, staking transactions, votes on proposals, instructions for sending funds from the contract's wallet, an error section, and a data section of free-form bytes to be interpreted by the client or dApp with added support for additional types in the future.

In `contract.rs`, the `init()` function uses the magic number `256` as the block size of padding messages, and in the `handle.rs`, the padding number `256` is also widely used. It is more consistent to use the defined constant `RESPONSE_BLOCK_SIZE` to represent the block size of the padding message.

In addition, in `handle.rs` a different magic number of the block size `1` is used as well. As a result, the usage of padding block size is inconsistent throughout the code base, which may be prone to error when upgrading in the future.

Recommendation

Recommend replacing the magic number `256` with `RESPONSE_BLOCK_SIZE` and unifying the padding block size throughout the code base when needed.

Alleviation

[Shade Protocol]: The team heeded the advice and resolved this issue in the commit 746b3505014cc3b87b44ae72d4eb04c6abeb29d6.

Appendix

Finding Categories

Centralization / Privilege

Centralization / Privilege findings refer to either feature logic or implementation of components that act against the nature of decentralization, such as explicit ownership or specialized access roles in combination with a mechanism to relocate funds.

Logical Issue

Logical Issue findings detail a fault in the logic of the linked code, such as an incorrect notion on how block.timestamp works.

Coding Style

Coding Style findings usually do not affect the generated byte-code but rather comment on how to make the codebase more legible and, as a result, easily maintainable.

Checksum Calculation Method

The "Checksum" field in the "Audit Scope" section is calculated as the SHA-256 (Secure Hash Algorithm 2 with digest size of 256 bits) digest of the content of each file hosted in the listed source repository under the specified commit.

The result is hexadecimal encoded and is the same as the output of the Linux "sha256sum" command against the target file.

Disclaimer

This report is subject to the terms and conditions (including without limitation, description of services, confidentiality, disclaimer and limitation of liability) set forth in the Services Agreement, or the scope of services, and terms and conditions provided to you (“Customer” or the “Company”) in connection with the Agreement. This report provided in connection with the Services set forth in the Agreement shall be used by the Company only to the extent permitted under the terms and conditions set forth in the Agreement. This report may not be transmitted, disclosed, referred to or relied upon by any person for any purposes, nor may copies be delivered to any other person other than the Company, without CertiK’s prior written consent in each instance.

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts CertiK to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. CertiK’s position is that each company and individual are responsible for their own due diligence and continuous security. CertiK’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by CertiK is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

ALL SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF ARE PROVIDED “AS IS” AND

“AS AVAILABLE” AND WITH ALL FAULTS AND DEFECTS WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED UNDER APPLICABLE LAW, CERTIK HEREBY DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS. WITHOUT LIMITING THE FOREGOING, CERTIK SPECIFICALLY DISCLAIMS ALL IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT, AND ALL WARRANTIES ARISING FROM COURSE OF DEALING, USAGE, OR TRADE PRACTICE. WITHOUT LIMITING THE FOREGOING, CERTIK MAKES NO WARRANTY OF ANY KIND THAT THE SERVICES, THE LABELS, THE ASSESSMENT REPORT, WORK PRODUCT, OR OTHER MATERIALS, OR ANY PRODUCTS OR RESULTS OF THE USE THEREOF, WILL MEET CUSTOMER’S OR ANY OTHER PERSON’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULT, BE COMPATIBLE OR WORK WITH ANY SOFTWARE, SYSTEM, OR OTHER SERVICES, OR BE SECURE, ACCURATE, COMPLETE, FREE OF HARMFUL CODE, OR ERROR-FREE. WITHOUT LIMITATION TO THE FOREGOING, CERTIK PROVIDES NO WARRANTY OR UNDERTAKING, AND MAKES NO REPRESENTATION OF ANY KIND THAT THE SERVICE WILL MEET CUSTOMER’S REQUIREMENTS, ACHIEVE ANY INTENDED RESULTS, BE COMPATIBLE OR WORK WITH ANY OTHER SOFTWARE, APPLICATIONS, SYSTEMS OR SERVICES, OPERATE WITHOUT INTERRUPTION, MEET ANY PERFORMANCE OR RELIABILITY STANDARDS OR BE ERROR FREE OR THAT ANY ERRORS OR DEFECTS CAN OR WILL BE CORRECTED.

WITHOUT LIMITING THE FOREGOING, NEITHER CERTIK NOR ANY OF CERTIK’S AGENTS MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND, EXPRESS OR IMPLIED AS TO THE ACCURACY, RELIABILITY, OR CURRENCY OF ANY INFORMATION OR CONTENT PROVIDED THROUGH THE SERVICE. CERTIK WILL ASSUME NO LIABILITY OR RESPONSIBILITY FOR (I) ANY ERRORS, MISTAKES, OR INACCURACIES OF CONTENT AND MATERIALS OR FOR ANY LOSS OR DAMAGE OF ANY KIND INCURRED AS A RESULT OF THE USE OF ANY CONTENT, OR (II) ANY PERSONAL INJURY OR PROPERTY DAMAGE, OF ANY NATURE WHATSOEVER, RESULTING FROM CUSTOMER’S ACCESS TO OR USE OF THE SERVICES, ASSESSMENT REPORT, OR OTHER MATERIALS.

ALL THIRD-PARTY MATERIALS ARE PROVIDED “AS IS” AND ANY REPRESENTATION OR WARRANTY OF OR CONCERNING ANY THIRD-PARTY MATERIALS IS STRICTLY BETWEEN CUSTOMER AND THE THIRD-PARTY OWNER OR DISTRIBUTOR OF THE THIRD-PARTY MATERIALS.

THE SERVICES, ASSESSMENT REPORT, AND ANY OTHER MATERIALS HEREUNDER ARE SOLELY PROVIDED TO CUSTOMER AND MAY NOT BE RELIED ON BY ANY OTHER PERSON OR FOR ANY PURPOSE NOT SPECIFICALLY IDENTIFIED IN THIS AGREEMENT, NOR MAY COPIES BE DELIVERED TO, ANY OTHER PERSON WITHOUT CERTIK’S PRIOR WRITTEN CONSENT IN EACH INSTANCE.

NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH SERVICES, ASSESSMENT REPORT, AND ANY ACCOMPANYING MATERIALS.

THE REPRESENTATIONS AND WARRANTIES OF CERTIK CONTAINED IN THIS AGREEMENT ARE SOLELY FOR THE BENEFIT OF CUSTOMER. ACCORDINGLY, NO THIRD PARTY OR ANYONE ACTING ON BEHALF OF ANY THEREOF, SHALL BE A THIRD PARTY OR OTHER BENEFICIARY OF SUCH REPRESENTATIONS AND WARRANTIES AND NO SUCH THIRD PARTY SHALL HAVE ANY RIGHTS OF CONTRIBUTION AGAINST CERTIK WITH RESPECT TO SUCH REPRESENTATIONS OR WARRANTIES OR ANY MATTER SUBJECT TO OR RESULTING IN INDEMNIFICATION UNDER THIS AGREEMENT OR OTHERWISE.

FOR AVOIDANCE OF DOUBT, THE SERVICES, INCLUDING ANY ASSOCIATED ASSESSMENT REPORTS OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.

About

Founded in 2017 by leading academics in the field of Computer Science from both Yale and Columbia University, CertiK is a leading blockchain security company that serves to verify the security and correctness of smart contracts and blockchain-based protocols. Through the utilization of our world-class technical expertise, alongside our proprietary, innovative tech, we're able to support the success of our clients with best-in-class security, all whilst realizing our overarching vision; provable trust for all throughout all facets of blockchain.