ITHEUM

SMART CONTRACT AUDIT
Certification

ITHEUM BRIDGE MULTIVERSX CONTRACT

xaudits

SMART CONTRACT AUDITS

Jun 19th, 2024 / v.0.2

Audited source code version:

618d3cfe71474adb7a2503fc291e130a8265c739

Structure and Organization of the Document

Some sections are more important than others. The most critical areas are at the top, and the less critical sections are at the bottom. The issues in these sections have been fixed or addressed and will show by the "Resolved" or "Unresolved" tags. Each case is written so you can understand how serious it is, with an explanation of whether it is a risk of exploitation or unexpected behavior.

CRITICAL

These issues can have a dangerous effect on the ability of the contract to work correctly.

HIGH

These issues significantly affect the ability of the contract to work correctly.

MEDIUM

These issues affect the ability of the contract to operate correctly but do not hinder its behavior.

LOW

These issues have a minimal impact on the contract's ability to operate.

INFORMATIONAL

These issues do not impact the contract's ability to operate.

Issues

1. Missing events

Fixed / LOW

Description: There are events that are configured in **'events.rs'** file but are unused (eg **'setFeeValueEvent'**). Also, there are certain endpoints in **'admin.rs'** that don't have events at all (eg **'addToWhitelist'**).

! **Possible fix to research**

```
Use the existing events or implement new ones where it fits.
```

! **Response**

```
Fixed.
```

! **Status**

```
Accepted & Closed.
```

2. Improve transparency

Not Addressed / INFORMATIONAL

Description: Right now, the code heavily relies on an off-chain component to do the settlement between mvx and sol transactions and traceability is not the best. When an user wants to send tokens from mvx to sol, he would have, on the mvx side, to supply the destination address and signature correspondent to its sol address when he does the **'send_to_liquidity'** action on the mvx contract, which is great. On the other hand, when the sol relayer calls **'send_from_liqudity'** there is no link to the initial mvx transaction, and will most likely become at some point a problem of traceability.

! **Possible fix to research**

```
When the relayer sends the tokens on the destination chain, supply the
deposit tx hash on the other chain as a parameter. This way it would be
more transparent, the 'in' and 'out' transactions will be linked directly
on the chain for everyone to see, and traceability will become much
easier.
```

! **Status**

```
Not Addressed.
```

3. Avoid double spending

Not Addressed / INFORMATIONAL

Description: There is no check in the smart contracts to see if a bridge transaction has already been made. There could be cases where a relayer is running incorrectly (desynchronized and restarted) and it could miss some transactions (that could be handled by sending the tokens later) and it could send tokens twice for the same bridged tx. While there's no way of ensuring everything on-chain and trust in the relayer is a must, there are some checks that can be done in order to avoid those cases.

! Possible fix to research

```
In case the fix for the first suggestion is implemented, the smart
contract could save the tx hash of the other chain in its storage and
later check future hashes against it. This check will help in preventing
cases like double spending caused by problems in the relayer (bugs,
desyncs, etc).
```

! Status

```
Not Addressed.
```

4. Remove non-priority operations and dependencies

Fixed / INFORMATIONAL

Description: It is a rule of thumb that the smart contracts should have minimal external dependencies because it minimises the potential attack surface. This rule could be applied to the fee collection mechanism, meaning that the swapping (unwrapping) intermediary step can be eliminated. This would greatly improve the transaction cost and will eliminate redundancy (best to do one larger unwrap than a lot of smaller unwraps).

! Possible fix to research

```
Eliminate the unwrapping operation and send the fees as they are
received. Move the responsibility of unwrapping to the fees collector or
its administrator.
```

! Response

```
Fixed.
```

! Status

```
Accepted & Closed.
```

5. Desync with the sol version

Fixed / INFORMATIONAL

Description: The mvx contract allows for multiple whitelisted tokens, while the sol version allows for just one. It might be best to start with a 1:1 bridge. Building a bridge is a challenging task, from a security standpoint, the first version should have the minimal required operations for the launch.

! Possible fix to research

```
Make the contract only accept one token (one token to be allowed to be
whitelisted) besides from the fee token and do the bridge on a 1:1 basis.
Plans of adding more tokens could be implemented later (by upgrading the
contract or deploying a new one).
```

! Response

```
Fixed.
```

! Status

```
Accepted & Closed.
```

6. Eliminate the Add/Remove liquidity

Not Addressed / INFORMATIONAL

Description: This suggestion is tied with the one above, in case of choosing the path of a 1:1 bridge, there's no need for adding / removing liquidity aside for strictly the bridged transactions. In this case, the tokens that are locked in the contract match exactly with the number of tokens minted on the sol side. There's no need for additional operations.

! Possible fix to research

```
Remove the Add/Remove liquidity endpoints and rely strictly on the tokens
that are locked in the contract.
```

! Status

```
Not Addressed.
```

Verification Conditions

1 Admin functions are marked accordingly

```
    only_privileged!(self, ERR_NOT_PRIVILEGED);
```

2 User actions are guarded by active state checks on the contract.

```
         require_contract_ready!(self, ERR_CONTRACT_NOT_READY);
         require!(
                self.is_state_active(self.public_state().get()),
         ERR_CONTRACT_NOT_READY
         );
```

3 Valid tokens are accepted (token id and amount are being checked)

```
require!(
      self.tokens_whitelist().contains(&deposit.token_identifier),
      ERR_TOKEN_NOT_WHITELISTED
);
require!(
      self.check_amount(
            &deposit.amount,
            self.token_decimals(&deposit.token_identifier).get()),
      ERR_NOT_WHOLE_NUMBER
);
require!(
      self.minimum_deposit(&deposit.token_identifier).get() <= deposit.amount
      && deposit.amount <= self.maximum_deposit(&deposit.token_identifier).get(),
      ERR_PAYMENT_AMOUNT_NOT_IN_ACCEPTED_RANGE
);
```

4 Sending tokens from the contract to accounts can be done only by the relayer

```
           require!(self.relayer().get() == caller, ERR_NOT_PRIVILEGED);
```

Suggestions (Optional)

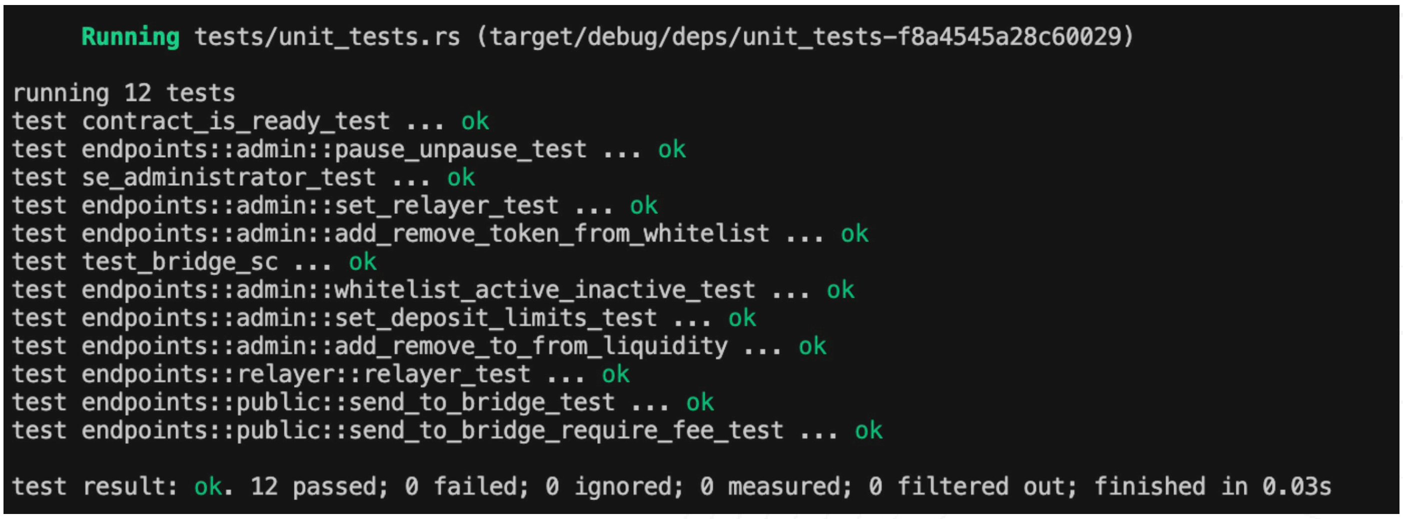
1. Simplify the logic and reduce the contract to its fundamentals (1:1 ITHEUM bridge, 1 token on each side, no liquidity being added or removed besides from token TXes).

```
Response: Not Addressed.

Status: Accepted & Closed. (only Optional)
```

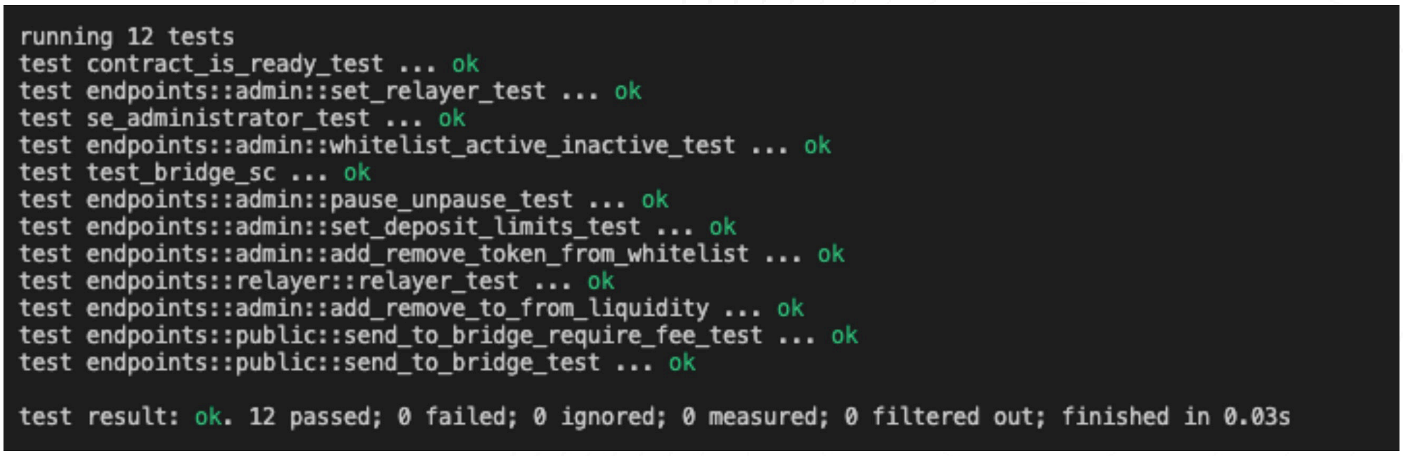
2. Run clippy on the code and solve warnings and suggesti (eg in 'check_amount' instead of the if statements and the two returns, you can use simply 'amount % &token_decimals == 0')

```
Response: Fixed.

Status: Accepted & Closed.
```

Test results



```
     Running tests/unit_tests.rs (target/debug/deps/unit_tests-f8a4545a28c60029)

running 12 tests
test contract_is_ready_test ... ok
test endpoints::admin::pause_unpause_test ... ok
test se_administrator_test ... ok
test endpoints::admin::set_relayer_test ... ok
test endpoints::admin::add_remove_token_from_whitelist ... ok
test test_bridge_sc ... ok
test endpoints::admin::whitelist_active_inactive_test ... ok
test endpoints::admin::set_deposit_limits_test ... ok
test endpoints::admin::add_remove_to_from_liquidity ... ok
test endpoints::relayer::relayer_test ... ok
test endpoints::public::send_to_bridge_test ... ok
test endpoints::public::send_to_bridge_require_fee_test ... ok

test result: ok. 12 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.03s
```



```
running 12 tests
test contract_is_ready_test ... ok
test endpoints::admin::set_relayer_test ... ok
test se_administrator_test ... ok
test endpoints::admin::whitelist_active_inactive_test ... ok
test test_bridge_sc ... ok
test endpoints::admin::pause_unpause_test ... ok
test endpoints::admin::set_deposit_limits_test ... ok
test endpoints::admin::add_remove_token_from_whitelist ... ok
test endpoints::relayer::relayer_test ... ok
test endpoints::admin::add_remove_to_from_liquidity ... ok
test endpoints::public::send_to_bridge_require_fee_test ... ok
test endpoints::public::send_to_bridge_test ... ok

test result: ok. 12 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.03s
```

Audited source code version
4af8a461b8d41e011dedfc6931d635e3e6d3f0c9

Second review source code version
618d3cfe71474adb7a2503fc291e130a8265c739