ITHEUM

SMART CONTRACT AUDIT
Certification

ITHEUM BRIDGE SOLANA CONTRACT

xaudits
SMART CONTRACT AUDITS
Jun 19th, 2024 / v.0.2
Audited source code version:
00ba562434b54db7a449bfed39a133036f2593d0

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. 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 a user wants to send tokens from sol to mvx, he would have to, on the sol side, supply the destination address and signature correspondent to its mvx address when he does the **'send_to_liquidity'** action on the sol contract, which is great. On the other hand, when the mvx relayer calls **'send_from_liqudity'** there is no link to the initial sol 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.
```

2. 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.
```

3. 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. This way, no ‘temporary ATA’ should be required.
```

! Response

```
Fixed.
```

! Status

```
Accepted & Closed.
```

4. Desync with the mvx version

Not Addressed / INFORMATIONAL

Description: The sol contract allows for a single whitelisted tokens, while the mvx version allows for multiple ones.

! Possible fix to research

```
Make the contracts work on a 1:1 ITHEUM bridge basis or make the same
logic (multiple tokens) on both contracts (alginate them - in sync
version is less error prone)
```

! Status

```
Not Addressed.
```

5. 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.
```

6. Improve ops verification

Not Addressed / INFORMATIONAL

Description: On the sending side (maybe on the receiving side too) a step of verifying the user's intention might be useful.

! Possible fix to research

```
Investigate if there's a way to verify the user intention by crafting a
message like: address + send_to_liquidity + destination_address + token_
type + number_of_tokens and verifying its signature (made off-chain using
the sender's private key) in the contract using its public key.
```

! Status

```
Not Addressed.
```

Verification Conditions

1 Admin functions are marked accordingly

```
#[account(
    mut,
    address=ADMIN_PUBKEY,
)]
pub authority: Signer<'info>,
```

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

```
    require!(
           ctx.accounts.bridge_state.public_state == State::Active.to_code(),
           Errors::ProgramIsPaused
    );
```

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

```
require!(
    check_amount(amount, ctx.accounts.mint_of_token_sent.decimals),
    Errors::NotWholeNumber
);

require!(
    ctx.accounts.bridge_state.minimum_deposit <= amount
        && amount <= ctx.accounts.bridge_state.maximum_deposit,
    Errors::PaymentAmountNotInAcceptedRange
);


#[account(
  constraint=mint_of_token_sent.key()==bridge_state.mint_of_token_whitelisted,
)]
pub mint_of_token_sent: Box<Account<'info, Mint>>,
```

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

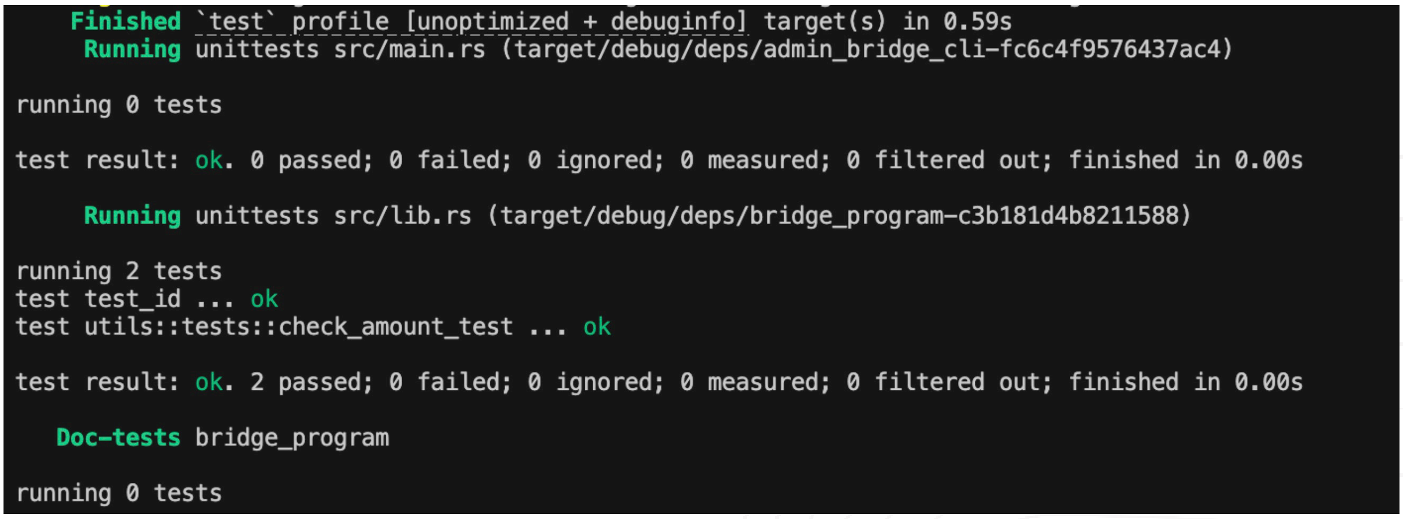
```
#[account(
    mut,
    address=bridge_state.relayer_pubkey.key() @ Errors::NotPrivileged,
)]
pub authority: Signer<'info>,
```

Suggestions (Optional)

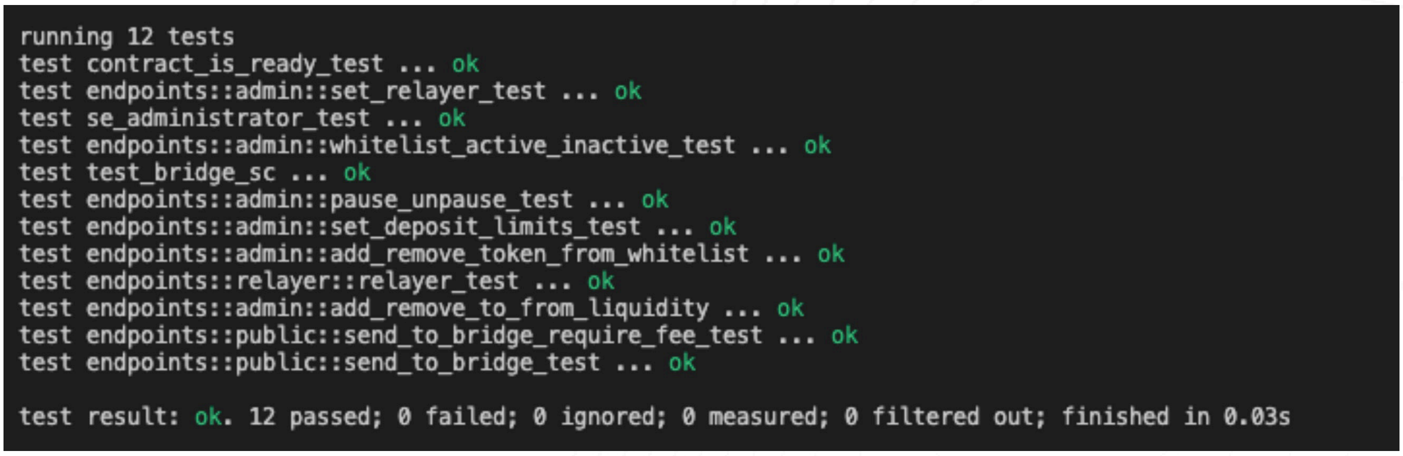
1. Simplify the logic and reduce the contract to its fundamentals (1:1 ITH bridge, 1 token on each side, no liquidity being added or removed besides from token TXes).

```
Response: Not Addressed.

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

Test results



```
    Finished `test` profile [unoptimized + debuginfo] target(s) in 0.59s
     Running unittests src/main.rs (target/debug/deps/admin_bridge_cli-fc6c4f9576437ac4)

running 0 tests

test result: ok. 0 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

     Running unittests src/lib.rs (target/debug/deps/bridge_program-c3b181d4b8211588)

running 2 tests
test test_id ... ok
test utils::tests::check_amount_test ... ok

test result: ok. 2 passed; 0 failed; 0 ignored; 0 measured; 0 filtered out; finished in 0.00s

   Doc-tests bridge_program

running 0 tests
```



```
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
630c449b2b44313a53260b6174e1f323e850d79a

Second review source code version
00ba562434b54db7a449bfed39a133036f2593d0