Echo Bridge
Audit Report

Thu May 15 2025

contact@bitslab.xyz https://twitter.com/scalebit_

ScaleBit

Echo Bridge Audit Report

1 Executive Summary

1.1 Project Information

Description	A cross-chain bridge built on Solana
Type	Bridge
Auditors	ScaleBit
Timeline	Fri May 09 2025 - Thu May 15 2025
Languages	Rust
Platform	Solana
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/echo-proto/bridge-sol
Commits	7489c996ed32db36417a1fe9f5138b147a1e6707 9ffdf0396686a89dc88ff1ec75937e3181f8d762 585889dfbbe7f70e1755113a842952ac96523acd a762950a419a27a69b1f18be74792944516c3476

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
ERR	programs/bridge/src/errors.rs	042f9f20e57246b2789f759b6a729efac697760b
CON	programs/bridge/src/constants.rs	b66bf0807b25dc89ccf4279074361fd2e510aec0
MOD	programs/bridge/src/instructions/echo_bridge/mod.rs	0ffeba6f055ba05c2b7ce124dca7ec7283ed0ce9
MSWS	programs/bridge/src/instructions/echo_bridge/mint_sbtc_with_signatures.rs	3b0c4b3c80b3fda68dfb519ea3b7fe361c22e5ee
E25	programs/bridge/src/instructions/utils/ed25519.rs	495c62f79305950d010f55c2486f418c2f9d5615
PBSIUMR	programs/bridge/src/instructions/utils/mod.rs	5f23ed8cb66e2fe52488f6e16ce367fc68e31944
UTI	programs/bridge/src/instructions/utils/utils.rs	9bfbf872b3930222c49bf9ef5ac07de63814a6fd
MSM	programs/bridge/src/instructions/utils/msgs/mint_sbtc_msg.rs	f14ccb091748c594d3965dfe86a67a8fa257b6ef
TRA	programs/bridge/src/instructions/utils/msgs/traits.rs	c51f135e24d1c3c14b1fa20b6382e87e23b1d7fd
PBSIUMMR	programs/bridge/src/instructions/utils/msgs/mod.rs	d665d5e457729ee2165aa8bbea7a6048c28fe907

COM	programs/bridge/src/instructions/utils/msgs/common.rs	72e9e7b4ca8b99ef3199a448960a695bae99c9e5
ULM	programs/bridge/src/instructions/utils/msgs/update_limiter_msg.rs	4fdbe8d4610046c0bc560683a0aa5f5dc9342512
WBM	programs/bridge/src/instructions/utils/msgs/withdraw_btc_msg.rs	7bcb302c46dc2ae3d85ce73b4ae96e14376dd000
PBSILMR	programs/bridge/src/instructions/limiter/mod.rs	e837535c63ad7754d4e5abc9aa8b1f2d613e6bac
STA	programs/bridge/src/instructions/limiter/state.rs	ca57ab1247dd795138827bcaaa72f19b1b3f8a1e
AOULWS	programs/bridge/src/instructions/limiter/add_or_update_limiter_with_signatures.rs	5ce3b98095f54e0655092e4a3ecf52f4b672539f
AOUC	programs/bridge/src/instructions/committee/add_or_update_committee.rs	6dc3797806e2cd8ce102d533197f58268813cb41
PBSICMR	programs/bridge/src/instructions/committee/mod.rs	9d88ab9166ba092709ac2292c8375ba428a62513
PBSICSR	programs/bridge/src/instructions/committee/state.rs	f1be73cf30428705833c85b8d4233342a9ecd348
AOUS	programs/bridge/src/instructions/committee/add_or_update_submitter.rs	2627faafbb55e55573e6e081a629381ffcbd6065
PBSIMR	programs/bridge/src/instructions/mod.rs	2ea7d6032da0d9c2d8b6c5dcd0ae97a023e50c9b

PBSICMR	programs/bridge/src/instructions/config/mod.rs	de5eb9de8713d2c711a17b69840b08acff575039
PBSICSR	programs/bridge/src/instructions/config/state.rs	359f4191717fe58876cd934cad0727b02d5eae68
AOUCT	programs/bridge/src/instructions/config/add_or_update_chain_token.rs	0334232ddf54dc333459b01aab1696a0893396e4
PBSICCBCR	programs/bridge/src/instructions/config/create_bridge_config.rs	cdafc77f8d6516bafc8e7a572f602d4faa9521f2
LIB	programs/bridge/src/lib.rs	10e6820ec5e03e0f0eeebfb44936f4e5e202693f
WBT	programs/bridge/src/instructions/echo_bridge/withdraw_btc.rs	f7a9b4e0d9998e5d24b58bb4d4e7c0df3e985f74
CBC	programs/bridge/src/instructions/committee/create_bridge_committee.rs	f4757f0aaee9fcfd5678d20897778f9d39dee15c
PBSICAOUCR	programs/bridge/src/instructions/config/add_or_update_chain.rs	95ea5a6597de73f83d38201603df1751dfaead1b

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	7	7	0
Informational	1	1	0
Minor	2	2	0
Medium	4	4	0
Major	0	0	0
Critical	0	0	0

1.4 ScaleBit Audit Breakdown

ScaleBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow
- Number of rounding errors
- Unchecked External Call
- Unchecked CALL Return Values
- Functionality Checks
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic issues
- Replay attacks
- Coding style issues

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** and **"Formal Verification"** strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by Echo Bridge to identify any potential issues and vulnerabilities in the source code of the Echo Bridge smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 7 issues of varying severity, listed below.

ID	Title	Severity	Status
CBC-1	Missing `withdraw_paused` Initialization when Creating the Bridge Configuration	Medium	Fixed
COM-1	Hardcoded `is_freezing` in `EmergencyOp`	Medium	Fixed
LIB-1	Missing Function to Update the `withdraw_paused` Field in the Bridge Configuration	Medium	Fixed
MSW-1	Keep the Constraint Style Consistent	Informational	Fixed
WBT-1	Missing Check to Ensure the Fee is greater than 0	Minor	Fixed
AOU1-1	Buggy Implementation of the `check_transfer` Method	Medium	Fixed
AOU1-2	`is_initialized` Not Set	Minor	Fixed

3 Participant Process

Here are the relevant actors with their respective abilities within the Echo Bridge Smart Contract :

Supper admin

- `create_bridge_config` : Create a bridge configuration for a specific chain

Admin

- `add_or_update_chain` : Add or update a supported chain configuration
- `add_or_update_chain_token` : Add or update token configurations for a specific chain
- `create_bridge_committee` : Create a governance committee
- `add_or_update_committee` : Add or update a committee member's stake and blocklist status
- `add_or_update_submitter` : Add or update a submitter's permission to execute operations

Submitter

- `mint_sbtc_with_signatures` : Mint sBTC on Solana with multi-signature verification
- `add_or_update_limiter_with_signatures` : Add or update token transfer limits for a chain with multi-signature verification

User

- `withdraw_btc` : Bridging assets back to the original chain

4 Findings

CBC-1 Missing withdraw_paused Initialization when Creating the Bridge Configuration

Severity: Medium

Status: Fixed

Code Location:

programs/bridge/src/instructions/config/create_bridge_config.rs#50-54

Descriptions:

In the `withdraw_btc()` function, the protocol checks that `!bridge_config.withdraw_paused` to ensure withdrawals are not paused.

```
 /// 1. load BridgeConfig
  #[account(
    seeds = [
      GLOBAL_CONFIG.as_bytes(),
      &msg.chain_id.to_be_bytes()
    ],
    bump,
    constraint = bridge_config.is_initialized @ ErrorCode::BridgeConfigNotInitialized,
    constraint = !bridge_config.withdraw_paused @ ErrorCode::BridgeWithdrawPaused,
    constraint = msg.chain_id != msg.to_chain_id @
ErrorCode::ChainIdShouldDiffFromSolanaChainId
  )]
  pub bridge_config: Box<Account<'info, BridgeConfig>>,
```

However, when the bridge configuration is created, the `withdraw_paused` value is not explicitly set, which could lead to unexpected behavior.

```
  bridge_config.chain_id = chain_id;
  bridge_config.admin = administrator;
  bridge_config.fee_recipient = fee_recipient;
```

```
    bridge_config.sbtc_mint = ctx.accounts.sbtc_mint.key();
    bridge_config.is_initialized = true;
```

Suggestion:

It is recommended to set a default value for `withdraw_paused` when creating the bridge configuration.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

COM-1 Hardcoded `is_freezing` in `EmergencyOp`

Severity: Medium

Status: Fixed

Code Location:

programs/bridge/src/instructions/utils/msgs/common.rs#99-119

Descriptions:

In the `required_stake` function, the `is_freezing` flag for the `EmergencyOp` operation is hardcoded to `true`:

```
Operation::EmergencyOp => {
    let is_freezing = true;
    decode_emergency_op_payload(&message.payload())?;
    if is_freezing {
        Ok(FREEZING_STAKE_REQUIRED)
    } else {
        Ok(UNFREEZING_STAKE_REQUIRED)
    }
}
```

As a result, the function always returns `FREEZING_STAKE_REQUIRED`, regardless of the actual intent in the message payload.

Suggestion:

Properly parse and determine the `is_freezing` value from the decoded payload to ensure the correct stake requirement (`FREEZING_STAKE_REQUIRED` or `UNFREEZING_STAKE_REQUIRED`) is applied based on the operation's semantics.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

LIB-1 Missing Function to Update the `withdraw_paused` Field in the Bridge Configuration

Severity: Medium

Status: Fixed

Code Location:

programs/bridge/src/lib.rs#41-61

Descriptions:

There are functions to update the chain, chain tokens, and the bridge committee, but there is no function to update the `withdraw_paused` field in the bridge configuration.

```
 pub fn add_or_update_chain<'info>(
        ctx: Context<'_, '_, 'info, 'info, AddChain<'info>>,
        _chain_id: u8,
        supported_chain_id: u8,
        init_nonce: u64,
        supported: bool,
    ) -> Result<()> {
        instructions::add_or_update_chain(ctx, _chain_id, supported_chain_id, init_nonce, 
supported)
    }

    pub fn add_or_update_chain_token<'info>(
        ctx: Context<'_, '_, 'info, 'info, AddChainToken<'info>>,
        _chain_id: u8,
        supported_chain_id: u8,
        token_id: u8,
        token_fee_percentages: u64,
        token_min_amount: u64,
        withdraw_paused: bool,
    ) -> Result<()> {
        instructions::add_or_update_chain_token(
            ctx,
            _chain_id,
            supported_chain_id,
            token_id,
            token_fee_percentages,
            token_min_amount,
            withdraw_paused,
        )
    }
```

Suggestion:

It is recommended to implement a function to update the `withdraw_paused` field in the bridge configuration.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

MSW-1 Keep the Constraint Style Consistent

Severity: Informational

Status: Fixed

Code Location:

programs/bridge/src/instructions/echo_bridge/mint_sbtc_with_signatures.rs#139

Descriptions:

In the `withdraw_btc()` function, the constraint is written as `constraint = supported_chain_config.supported`,

```
  #[account(
      mut,
      seeds = [
          SUPPORTED_CHAINS_CONFIG.as_ref(),
          msg.to_chain_id.to_be_bytes().as_ref(),
      ],
      bump,
      constraint = supported_chain_config.is_initialized @
ErrorCode::SupportedChainConfigNotInitialized,
      constraint = supported_chain_config.supported @
ErrorCode::SupportedChainConfigNoSupported,
      constraint = (msg.amount as u128) <= supported_chain_config.mint_total @
ErrorCode::LackTargetMintOfChain,
  )]
  pub supported_chain_config: Box<Account<'info, SupportedChainConfig>>,
```

while in the `mint_sbtc_with_signatures()` function, the same constraint is written as `constraint = supported_chain_config.supported == true`.

```
  #[account(
      mut,
      seeds = [
          SUPPORTED_CHAINS_CONFIG.as_ref(),
            msg.source_chain_id.to_be_bytes().as_ref(),
        ],
        bump,
        constraint = supported_chain_config.is_initialized @
ErrorCode::SupportedChainConfigNotInitialized,
        constraint = supported_chain_config.supported == true @ ErrorCode::InvalidChain
    )]
    pub supported_chain_config: Box<Account<'info, SupportedChainConfig>>,
```

Suggestion:

It is recommended to keep the constraint style consistent by using `constraint = supported_chain_config.supported`.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

WBT-1 Missing Check to Ensure the Fee is greater than 0

Severity: Minor

Status: Fixed

Code Location:

programs/bridge/src/instructions/echo_bridge/withdraw_btc.rs#59

Descriptions:

In the `withdraw_btc()` function, the protocol calculates the fee using the formula:

```
let fee = (((msg.amount as u128) * (token_config.token_fee_percentage as u128))
      / (FEE_DENOMINATOR as u128)) as u64;
  let amount = msg.amount - fee;
```

However, the protocol does not verify that the calculated fee is greater than 0. An attacker could exploit this by repeatedly initiating withdrawals with very small amounts to avoid paying any fee.

Suggestion:

It is recommended to ensure the fee is greater than 0.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

AOU1-1 Buggy Implementation of the `check_transfer` Method

Severity: Medium

Status: Fixed

Code Location:

programs/bridge/src/instructions/limiter/add_or_update_limiter_with_signatures.rs#140-167

Descriptions:

The logic of the function in checking whether the input funds have reached the upper limit is shown below:

```
pub fn check_transfer<'info>(
    limiter: &mut Account<'info, ChainTokenLimiter>,
    amount: u64,
) -> Result<()> {
    let current_h = current_hour();

    // Calculate elapsed hours (handles u32 overflow)
    let hours_passed = current_h.wrapping_sub(limiter.oldest_hour);

    if hours_passed > 0 {
        // Calculate the number of slots that need to be cleared (up to 24 hours)
        let slots_to_clear = std::cmp::min(hours_passed, 24) as usize;

        // Ring buffer index calculation
        let start_idx = (limiter.oldest_hour % 24) as usize;
        // Clear expired data and advance the start time
        for i in 0..slots_to_clear {
            let idx = (start_idx + i) % 24;
            limiter.hourly_transfers[idx] = 0;
        }

        limiter.oldest_hour = limiter.oldest_hour.wrapping_add(slots_to_clear as u32);
    }
...
}
```

In the above code, there is incorrect scroll cleanup logic that can cause cleanup errors to produce unintended results. For example,We assume that the number of array elements is as follows:

```
limiter.hourly_transfers=[A,B,C,D,...X]. idx=>0-23
such as limiter.hourly_transfers[0] = A,limiter.hourly_transfers[1]=B,etc.
```

line 160 will incorrectly clear the previous element by 0.

Let's assume `hours_passed==1` and `limiter.oldest_hour%24==3` : We'll see that lines 159-162 incorrectly set `limiter.hourly_transfers[3]` to 0, which will result in unintended placement of assets beyond the set amount limit.

line 165 if the update takes longer than 24h then `oldest_hour` here will not be updated to the latest subscript.

Let us assume that `hours_passed==25` . At this point, the 165 lines of code will be `limiter.oldest_hour+=24` which is the wrong calculation logic, the correct calculation logic should be `limiter.oldest_hour+=25` .

Suggestion:

It is recommended to refer to sui's logical implementation of scrolling windows.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

AOU1-2 `is_initialized` Not Set

Severity: Minor

Status: Fixed

Code Location:

programs/bridge/src/instructions/committee/add_or_update_submitter.rs#10-23

Descriptions:

In both the `create_bridge_committee` and `add_or_update_submitter` instructions, the `submitter_pda` account is updated. However, the `is_initialized` flag is never set to `true`. As a result, every time the account is updated, it is treated as uninitialized.

Example from `add_or_update_submitter`:

```
if !ctx.accounts.submitter_pda.is_initialized {
    ctx.accounts.submitter_pda.submitter = ctx.accounts.submitter.key();
    ctx.accounts.submitter_pda.is_submitter = is_submitter;
} else {
    ctx.accounts.submitter_pda.is_submitter = is_submitter;
}
```

Suggestion:

Set `is_initialized = true` the first time the account is initialized to avoid repeated initialization logic and ensure correct state handling.

Resolution:

This issue has been fixed. The client has adopted our suggestions.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.