Movement Atomic Bridge

Audit Report

MOVEBIT

contact@bitslab.xyz

https://twitter.com/movebit_

Fri Nov 01 2024

Movement Atomic Bridge Audit Report

1 Executive Summary

1.1 Project Information

Description	An atomic swap is a trustless token exchange mechanism between two parties A (the initiator) and B (the counter-party) and without dependencies on a third party.
Type	Bridge
Auditors	MoveBit
Timeline	Wed Aug 28 2024 - Fri Nov 01 2024
Languages	Solidity, Move, Rust
Platform	Ethereum,Aptos
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/movementlabsxyz/movement
Commits	28ee823e248b5e70410d5b820baf8ced07f1bd15 41ec467d303819ad5f38f408cb1605028c04999b 388ada0b2d10318348aac4d55bd50f3b02e397cd

1.2 Files in Scope

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

ID	File	SHA-1 Hash
CAR28	protocol-units/bridge/chains/movement/Cargo.toml	b864ecf7044b1320e94817cb7979845472925083
CAR29	protocol-units/bridge/chains/ethereum/Cargo.toml	7e0c00fb3462c432ecf2dfbf49b99ec094a0ac6a
CAR30	protocol-units/bridge/shared/Cargo.toml	1a11f016e4b43990bdd056b5bce3154469f1caf0
ABC	protocol-units/bridge/move-modules/sources/atomic_bridge_counterparty.move	9b37167952fad18ba2114f0bb9fd005d09796d15
ABI	protocol-units/bridge/move-modules/sources/atomic_bridge_initiator.move	a07d249169c4ab1d931e5fa2234193a750bf34e4
MOVETH	protocol-units/bridge/move-modules/sources/MOVETH.move	d844955446e260d2e4cb72e5fba94ac971afb5db
CAR32	protocol-units/bridge/cli/Cargo.toml	d232fc40eeacdce867e02143614eae270369a05e
ABI1	protocol-units/bridge/contracts/src/AtomicBridgeInitiator.sol	ce64e27fcd38396165f784ca938ba00621114a28
ABC1	protocol-units/bridge/contracts/src/AtomicBridgeCounterparty.sol	1188e20c5de05c6c5665d8b4a546c1c3c2e69e5f

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	13	13	0
Informational	3	3	0
Minor	0	0	0
Medium	1	1	0
Major	6	6	0
Critical	3	3	0

1.4 MoveBit Audit Breakdown

MoveBit 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 by bit operations
- Number of rounding errors
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting
- Unchecked CALL Return Values
- The flow of capability
- Witness Type

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) Formal Verification

Perform formal verification for key functions with the Move Prover.

(4) 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 Movement Atomic Bridge to identify any potential issues and vulnerabilities in the source code of the Movement Atomic 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 13 issues of varying severity, listed below.

ID	Title	Severity	Status
ABC-1	`abortBridgeTransfer` Has No Access Control.	Medium	Fixed
ABC-2	Lack of Events Emit	Informational	Fixed
ABI-1	No Upper Limit for `time_lock`	Critical	Fixed
ABI-2	`time_lock` Can Be Set to 0	Critical	Fixed
BSE-1	BridgeService Lacks Restart Recovery Mechanism	Major	Fixed
BSE-2	Two `Poll::Ready` Calls Could Cause Critical Event Loss	Major	Fixed
CAR-1	Code Error Causes Normal Functionality Failure	Informational	Fixed
ELO-1	Any Attacker Can Invoke Contract Code, Causing BridgeService Node to Crash	Critical	Fixed
MOV-1	Signature Replay	Major	Fixed
ABC1-1	`time_lock` Unit Inconsistency	Major	Fixed

ABC1-2	`complete_bridge_transfer` Function Missing `timeLock` Check	Major	Fixed
ABC1-3	Hard Code Error Code	Informational	Fixed
BSE1-1	BridgeService Lacks On-Chain Event Loss Recovery Mechanism	Major	Fixed

3 Participant Process

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

- **User**: Initiates a swap transaction. Locks assets on the first blockchain and unlocks assets from the second blockchain
- **First Blockchain**: The user locks assets on it, and BridgeService unlocks assets
- **BridgeService**: Locks assets on the second blockchain and unlocks assets on the first blockchain
- **Second Blockchain**: BridgeService locks assets on it, and the user unlocks assets

4 Findings

ABC-1 `abortBridgeTransfer` Has No Access Control.

Severity: Medium

Status: Fixed

Code Location:

protocol-units/bridge/contracts/src/AtomicBridgeCounterparty.sol#76

Descriptions:

`abortBridgeTransfer` has no permission control in the `Counterparty` module, which is inconsistent with the `abort_bridge_transfer` permission control in `atomic_bridge_counterparty.move`.

Suggestion:

It is recommended that the function of the two Counterparty modules be aligned.

ABC-2 Lack of Events Emit

Severity: Informational

Status: Fixed

Code Location:

protocol-units/bridge/contracts/src/AtomicBridgeCounterparty.sol#33;

protocol-units/bridge/contracts/src/AtomicBridgeInitiator.sol#42

Descriptions:

The contract lacks appropriate events for monitoring operations, such as `setAtomicBridgeInitiator` and so on, which could make it difficult to track sensitive actions or detect potential issues.

Suggestion:

It is recommended to emit events for the function.

ABI-1 No Upper Limit for `time_lock`

Severity: Critical

Status: Fixed

Code Location:

protocol-units/bridge/move-modules/sources/atomic_bridge_initiator.move#83;

protocol-units/bridge/contracts/src/AtomicBridgeInitiator.sol#47

Descriptions:

1. In the contract function `initiateBridgeTransfer`, there is no limit on the upper bound of the `time_lock` parameter.
2. An attacker could pass in an extremely large `time_lock`, such as 10,000 years, causing BridgeService to be unable to unlock its assets until 10,000 years later.

Suggestion:

Limit the maximum value of `time_lock`.

ABI-2 `time_lock` Can Be Set to 0

Severity: Critical

Status: Fixed

Code Location:

protocol-units/bridge/move-modules/sources/atomic_bridge_initiator.move#83;

protocol-units/bridge/contracts/src/AtomicBridgeInitiator.sol#47

Descriptions:

1. In the contract function `initiateBridgeTransfer`, there is no limit on the lower bound of the `time_lock` parameter.
2. An attacker could set `time_lock` to 0, allowing the `Initiator` to immediately refund their staked assets and instantly obtain the target tokens from the transaction.

Suggestion:

Set a minimum value for `time_lock`.

BSE-1 BridgeService Lacks Restart Recovery Mechanism

Severity: Major

Status: Fixed

Code Location:

protocol-units/bridge/shared/src/blockchain_service.rs#149

Descriptions:

1. While running, BridgeService must not lose important events. For example, losing the `BridgeContractCounterpartyEvent::Completed` event could allow the `Initiator` to both refund their staked assets and obtain the target tokens from the transaction.
2. During operation, BridgeService maintains a state for each transaction to ensure proper execution. After a restart, these states are completely lost, causing confusion in many transactions and even leading to asset loss.
3. In the long-term operation of the server, events such as power outages or natural disasters could cause the machine to restart, which is highly likely to occur.

Suggestion:

Add a restart recovery mechanism.

BSE-2 Two `Poll::Ready` Calls Could Cause Critical Event Loss

Severity: Major

Status: Fixed

Code Location:

protocol-units/bridge/shared/src/blockchain_service.rs#45

Descriptions:

1. While running, BridgeService must not lose important events. For example, losing the `BridgeContractCounterpartyEvent::Completed` event could allow the `Initiator` to both refund their staked assets and obtain the target tokens from the transaction.
2. In the `bridge_shared::blockchain_service::BlockchainService::poll_next_event()` function, the code is as follows:

```
fn poll_next_event(&mut self, cx: &mut Context<'_>) -> Poll<Option<Self::Item>> {
    match (
        self.initiator_monitoring().poll_next_unpin(cx),
        self.counterparty_monitoring().poll_next_unpin(cx),
    ) {
        (Poll::Ready(Some(event)), _) => {
            Poll::Ready(Some(ContractEvent::InitiatorEvent(event)))
        }
        (_, Poll::Ready(Some(event))) => {
            Poll::Ready(Some(ContractEvent::CounterpartyEvent(event)))
        }
        _ => Poll::Pending,
    }
}
```

   If the case `(Poll::Ready(Some(event)), Poll::Ready(Some(event)))` occurs, only the first branch of the match statement is executed, meaning the second event is ignored. This could lead to asset loss in BridgeService.

Suggestion:

Handle the situation where two `Poll::Ready` calls occur

CAR-1 Code Error Causes Normal Functionality Failure

Severity: Informational

Status: Fixed

Code Location:

protocol-units/bridge/service/Cargo.toml#11

Descriptions:

In the `ethereum_bridge::event_logging::EthInitiatorMonitoring::run()` function, the `listener` is not associated with the `sender`, so the `listener` does not receive the data sent by the `sender`

Suggestion:

Modify the code

ELO-1 Any Attacker Can Invoke Contract Code, Causing BridgeService Node to Crash

Severity: Critical

Status: Fixed

Code Location:

protocol-units/bridge/chains/ethereum/src/event_logging.rs#65

Descriptions:

1. In the `EthInitiatorMonitoring` class, a task is spawned to listen and decode events on the Ethereum blockchain:

```
tokio::spawn(async move {
    while let Some(log) = sub_stream.next().await {
        let event = decode_log_data(log)
            .map_err(|e| {
                tracing::error!("Failed to decode log data: {:?}", e);
            })
            .expect("Failed to decode log data");
```

2. If the `decode_log_data` function returns an error, it will cause a panic. Since this task does not catch the panic, it will crash the entire process.

3. In the `decode_log_data` function, an error is returned if the `recipient_address` or `hash_lock` in the event is not 32 bytes long:

```
let recipient_address = decoded.indexed[2]
    .as_fixed_bytes()
    .map(coerce_bytes)
    .ok_or_else(|| anyhow::anyhow!("Failed to decode RecipientAddress"))?;
//
let hash_lock = decoded.indexed[4]
    .as_fixed_bytes()
    .map(coerce_bytes)
    .ok_or_else(|| anyhow::anyhow!("Failed to decode HashLock"))?;
```

4. In the `AtomicBridgeInitiator.sol` contract, a malicious attacker can pass arbitrary lengths of `recipient_address` and `hashLock`:

```
function initiateBridgeTransfer(uint256 wethAmount, bytes32 recipient, bytes32
hashLock, uint256 timeLock)
    external
    payable
    returns (bytes32 bridgeTransferId)
{
    address originator = msg.sender;
```

Suggestion:

Add length checks for `recipient_address` and `hashLock` in the `AtomicBridgeInitiator.sol` and `atomic_bridge_initiator.move` contracts.

MOV-1 Signature Replay

Severity: Major

Status: Fixed

Code Location:

protocol-units/bridge/move-modules/sources/MOVETH.move#172

Descriptions:

The current `transfer_from` function in contract's signature verification carries a risk of signature replay. `get_sequence_number` does not increase with user transactions, resulting in proof signatures that can be reused.

Suggestion:

It is recommended to change to the correct logic to track the user's nonce thus avoiding replay attacks.

ABC1-1 `time_lock` Unit Inconsistency

Severity: Major

Status: Fixed

Code Location:

protocol-units/bridge/move-modules/sources/atomic_bridge_counterparty.move#92

Descriptions:

The `timeLock` in the `bridgeTransfers` object is the height of a future block, which is obtained by `block.number` in EVM and by `get_current_block_height` in Move. But in atomic_bridge_counterparty.move module `lock_bridge_transfer_assets` function to calculate `time_lock` through `now_seconds` function, this is not consistent with the time unit recorded in `BridgeTransfer`.

Suggestion:

It is recommended to ensure that this is consistent with the protocol design and to modify it to the correct logic.

ABC1-2 `complete_bridge_transfer` Function Missing `timeLock` Check

Severity: Major

Status: Fixed

Code Location:

protocol-units/bridge/move-modules/sources/atomic_bridge_counterparty.move#109

Descriptions:

The `complete_bridge_transfer` function lacks a `time_lock` check, resulting in `complete_bridge_transfer` still being called after `time_lock` if the user does not call the `abort_bridge_transfer` function in time.

Suggestion:

It is recommended to add a `time_lock` check to the function.

ABC1-3 Hard Code Error Code

Severity: Informational

Status: Fixed

Code Location:

protocol-units/bridge/move-modules/sources/atomic_bridge_counterparty.move#120

Descriptions:

Using a hard-coded way to manage error codes may make it difficult to maintain the code at a later stage.

Suggestion:

It is recommended that a constant error code be used.

BSE1-1 BridgeService Lacks On-Chain Event Loss Recovery Mechanism

Severity: Major

Status: Fixed

Code Location:

protocol-units/bridge/shared/src/bridge_service.rs#140

Descriptions:

1. While running, BridgeService must not lose important events. For example, losing the `BridgeContractCounterpartyEvent::Completed` event could allow the `Initiator` to both refund their staked assets and obtain the target tokens from the transaction.
2. Network instability or malicious behavior from third-party service nodes we connect to could result in lost events or even erroneous events.
3. During long-term server operation, situations like network disconnection or instability are common occurrences.

Suggestion:

Implement a recovery mechanism for lost on-chain events.

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.