Mem-Bridge Security Review

Reviewers
Reviewer 1, Nirlin
Reviewer 2, Amar Fares
June 7, 2024

1 Executive Summary

Total Issues

Critical Risk	0
High Risk	0
Medium Risk	2
Low Risk	0
Gas Optimizations and Informational	0

Contents

2 9Lives Labs

9LivesLabs is a team of smart contract security researchers comprising of 0xnirlin & Amar Fares. Together, we help secure the Web3 ecosystem. We offer security reviews and related services to Web3 projects.

Disclaimer: This security review does not guarantee against a hack. It is a snapshot in time of brink according to the specific commit by a three person team. Any modifications to the code will require a new security review.

3 Findings

3.1 Medium Risk

3.1.1 Wrong function signatures are used for Chainlink Client v0.8

Severity: Medium

Context: `Contract.sol#L160-L165`

Description:

ChainLinkClient v0.8 uses different signatures from chainlink v0.6 but in this case signatures from v0.6 are also used which will never work.

For example, in this snippet in the constructor

```
        setChainlinkToken(_linkTokenAddr);
        setChainlinkOracle(_oracleAddress);
```

Actual signatures are `_setChainlinkToken` and `setChainlinkOracle` in chainlink v0.8

Recommendation: Change the signatures to latest version.

Project: Fixed in PR.

9Lives Labs Resolved.

3.2 Medium Risk

3.2.1 No transfer limit set in bridge.js can lead to small amounts of tokens bridged back getting stuck

Severity: Medium

Context: `Bridge.js`

Description:

```
        require(amount > unlockFlatFee, "err_invalid_amount");
```

This means the amount is less than `unlockFlatFee`, now the architecture is such that if such a small amount is bridged, more amount cannot be added to the same transaction so it will be forever locked, which is possible because there is no such minimum transfer check in MEM `bridge.js` code. Any amount can be transferred as shown in the following snippet

```
if (input.function === "initiateUnlock") {
  const { caller, sig, amount } = input;

  const bigIntAmount = BigInt(amount);

  const normalizedCaller = _normalizeCaller(caller);
  ContractAssert(
    bigIntAmount <= BigInt(state.balances[normalizedCaller]),
    "err",
  );

  await _moleculeSignatureVerification(normalizedCaller, sig);

  state.unlocks.push({
    address: normalizedCaller,
    mid: sig,
    amount: amount,
  });

  const newBalance = BigInt(state.balances[normalizedCaller]) - bigIntAmount;

  state.balances[normalizedCaller] = newBalance.toString();

  return { state };
}
```

Recommendation: Implement a minimum amount check in bridge.js

Project: Fixed in PR.

9Lives Labs Resolved.

4 Appendix