CANTINA

Maker DAO: Arbitrum Token Bridge

Security Review

Cantina Managed review by:
Christoph Michel, Lead Security Researcher
M4rio.eth, Security Researcher
Shung, Associate Security Researcher

July 3, 2024

Contents

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity	Description
Critical	*Must* fix as soon as possible (if already deployed).
High	Leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
Medium	Global losses <10% or losses to only a subset of users, but still unacceptable.
Low	Losses will be annoying but bearable. Applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.
Gas Optimization	Suggestions around gas saving practices.
Informational	Suggestions around best practices or readability.

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

The Maker Protocol, also known as the Multi-Collateral Dai (MCD) system, allows users to generate Dai (a decentralized, unbiased, collateral-backed cryptocurrency soft-pegged to the US Dollar) by leveraging collateral assets approved by the Maker Governance, which is the community organized and operated process of managing the various aspects of the Maker Protocol.

From Jun 24th to Jun 28th the Cantina team conducted a review of arbitrum-token-bridge on commit hash 7fd37185.

The Cantina team reviewed MakerDao's Arbitrum Token Bridge changes holistically on commit hash 6248966bd8dac6261fb296f9743a0b9432cfec71 and determined that all issues were resolved and no new issues were identified.

The team identified a total of **12** issues in the following risk categories:

- Critical Risk: 0
- High Risk: 0
- Medium Risk: 0
- Low Risk: 1
- Gas Optimizations: 1
- Informational: 10

3 Findings

3.1 Low Risk

3.1.1 Closing L2 gateways and misconfigurations can lead to L2 tokens being unbacked

Severity: Low Risk

Context: L2TokenGateway.sol#L139

Description: There are several ways to configure the L1 and L2 gateway `l1ToL2Token` mappings that end up with the L2 tokens becoming unredeemable. They are essentially unbacked and will lose value.

1. `L2TokenGateway.close()` will disable withdrawals for all tokens.
2. `L2TokenGateway.registerToken(l1Token, address(0))` will disable redeeming the `l2Token` for the `l1Token`.
3. When multiple L1 tokens map to the same, single L2 token, the L2 withdrawer can choose what L1 token to redeem for. However, if the escrow contract is depleted for one token (because the L2 token is now fully backed by the other L1 token), the withdrawal transaction succeeds on L2 but will fail on L1, losing the funds.
4. If the L2 contract has a minter besides the gateway, not all of them can be redeemed and they will be partially unbacked.

Recommendation: Ensure no two L1 tokens map to the same L2 token, even across gateways as they use the same `escrow` contract. Ensure the L2 tokens only use the gateway as their authorized minter. When disabling withdrawals on L2, consider the consequences of the L2 tokens now being unbacked and potentially losing value.

Maker: We consider this not to be an issue. The spell crafting process is trusted not to result in a misconfiguration of the bridge. In particular, the instructions in the README are assumed to be followed to upgrade the bridge; L2 tokens are assumed to be mapped 1:1 with L1 tokens; and in case more than one L2 gateway is given the right to mint an L2 token, it is assumed that all the bridges involved share the same escrow.

Cantina Managed: Acknowledged.

3.2 Gas Optimization

3.2.1 Aliased counterpart gateway address can be stored as an immutabe variable

Severity: Gas Optimization

Context: L2TokenGateway.sol#L71

Description: `L2TokenGateway.finalizeInboundTransfer()` checks if the `msg.sender` is the aliased version of the counterpart gateway address at L1. Per Arbitrum docs, the aliased version is calculated each time by adding `0x1111000000000000000000000000000000001110` to the counterpart gateway address. Due to the `L2TokenGateway.counterpartGateway` address being an immutable variable, the aliased address will always be the same. Therefore, the aliased address can be calculated in the constructor once and stored as an immutable address along with the `counterpartGateway`.

Recommendation: Consider calculating the aliased address only once in the constructor and storing it as an immutable variable along with the `counterpartGateway`. Then, during the access control check, this immutable aliased address can be used directly instead of recalculating the counterpart gateway alias in each call.

Maker: Generally the need to gas optimise is counterbalanced with the preference to keep the code closer to the Arbitrum generic custom gateway implementation in `token-bridge-contracts`.

This is a case here where the gas savings are very small (a simple + operation on an L2) so would prefer to keep the code more similar to that in L2ArbitrumGateway.sol#L56C14-L56C36.

Cantina Managed: Acknowledged.

3.3 Informational

3.3.1 Use a unique deployer address on Arbitrum

Severity: Informational

Context: N/A **Description:** Using the same deployer address can lead to unrelated contracts on two chains to have the same address. This can lead to confusion and mistakes.

There is no indication of the deployer address for this deployment, but a deployer address on Ethereum was reused on Arbitrum previously which lead to Optimism: DAI bridge on Ethereum and L2GovernanceRelay on Arbitrum having the same address.

Recommendation: Make sure to use a unique deployer address for Arbitrum deployments.

Maker: Acknowledged.

Cantina Managed: Acknowledged.

3.3.2 Tokens can get lost when bridging L1 to L2 due to ticket expiry

Severity: Informational

Context: Manual redemption - Arbitrum docs

Description: If auto-redemption of a retryable ticket fails, it must be manually redeemed within 7 days. Failure to do so results in the ticket expiring, preventing the corresponding L2 token from being minted. This locks the L1 token in escrow, making it inaccessible and effectively causing token loss.

Recommendation: Consider monitoring retryable tickets to ensure they are redeemed within the 7 days, mitigating the risk of token loss.

Maker: Acknowledged.

Cantina Managed: Acknowledged.

3.3.3 Unused functions

Severity: Informational

Context: L1ArbitrumMessenger.sol#L133-L157, L1ArbitrumMessenger.sol#L81-L103, L1ArbitrumMessenger.sol#L55-L79, L1ArbitrumMessenger.sol#L49-L53, L1TokenGateway.sol#L210-L223

Description: The two `sendTxToL2()` functions in `L1ArbitrumMessenger` contract are unused and can safely be removed.

`L1TokenGateway.createOutboundTxCustomRefund()` calls the `L1ArbitrumMessenger.sendTxToL2CustomRefund()` with custom `L2GasParams` type as one of the arguments. `sendTxToL2CustomRefund()` calls its namesake with flattened arguments. If `L1TokenGateway.createOutboundTxCustomRefund()` directly calls the `L1ArbitrumMessenger.sendTxToL2CustomRefund()` with flattened arguments, then `L2GasParams` struct and the `sendTxToL2CustomRefund()` function with non-flattened arguments can be removed from the `L1ArbitrumMessenger` contract.

Recommendation: Consider using the flattened `L1ArbitrumMessenger.sendTxToL2CustomRefund()` function within `L1TokenGateway.createOutboundTxCustomRefund()`. Then the following can be removed from the `L1ArbitrumMessenger` contract:

- `sendTxToL2CustomRefund()` (non-flattened function).
- `L2GasParams` struct.
- `sendTxToL2()` (flattened function).
- `sendTxToL2()` (non-flattened function).

Maker: The `arbitrum` directory contains files that are taken from github.com/OffchainLabs/token-bridge-contracts . Generally we prefer to keep the files unchanged to make the process of diffing against the original files easier.

Code changed in commit 6248966b to use the flattened version of `L1ArbitrumMessenger.sendTxToL2CustomRefund()`.

Cantina Managed: Unused functions were kept but the code was updated to use the `sendTxToL2CustomRefund()` with flattened arguments.

3.3.4 Prefer `encodeCall` to `encodeWithSelector`

Severity: Informational

Context: L1TokenGateway.sol#L191-L198, L2TokenGateway.sol#L166-L173

Description: Both of the gateway contracts create the data to be passed to the `Inbox` and `ArbSys` contracts using `abi.encodeWithSelector`.

Recommendation: Consider using `abi.encodeCall` as it type-checks that the correct arguments for the `finalizeInboundTransfer()` function is used. Wrong or missing argument types can be caught during compile time, whereas `encodeWithSelector` does not give this guarantee.

Maker: Code changed in commit 6248966b.

Cantina Managed: Fix verified.

3.3.5 Checks improvements on l2 tokens in the `initGateways`

Severity: Informational

Context: TokenGatewayInit.sol#L93-L100

Description: During the deploy phase, various checks are performed for the soon to be registered tokens on L1/L2 gateways:

- `cfg.l1Tokens.length == cfg.l2Tokens.length` length must match.
- Both l1/l2 tokens at position `i` in the array to not be `address(0)`.
- `l1Token` to not be registered already in the `l1Gateway`.

These checks should be improved with checking if the `l2Token` is really a token on L2 and not a misconfigured address. This check can be performed by calling the `symbol()` function on Arbitrum and making sure that it will return a value. We assume that all the registered tokens will expose the `symbol` via the public function.

Recommendation: According to foundry scripting, we can use `vm.createSelectFork("arbitrum")`, for example, to switch between rpcs to do this extra check.

Maker: The spell process is trusted to properly check that the bytecode of the registered L2 tokens is as expected.

Cantina Managed: Acknowledged.

3.3.6 Initial deposits can fail when `L2Spell` execution fails

Severity: Informational

Context: L2TokenGateway.sol#L207, TokenGatewayInit.sol#L103-L116, L2TokenGatewaySpell.sol#L47

Description: As part of the deployment spell the `L1GovernanceRelay` sends a cross-chain ticket to the `L2Spell`, registering the tokens on the L2 Gateway. The `L2Spell` execution is async and the ticket can fail depending on the chosen gas parameters. As the `L1TokenGateway` starts out in the `isOpen` state, deposits are immediately enabled and these tickets might fail as the token mapping is not yet registered on L2. These tickets must be retried. Worst case, the `L2Spell` ticket expires and is removed, and the deposit ticket also expires and is not extended, resulting in the loss of the deposited funds.

Recommendation: Ensure the `L2Spell` ticket auto-redeem succeeds, or is retried on failure. Early users might have to retry their tickets. Alternatively, disable L1 deposits until the token mapping is registered on L2.

Maker: We consider this not to be an issue. Both the deposit's `l2gateway.finalizeInboundTransfer()` and the spell's `l2gateway.registerToken()` are called via a Retryable Ticket, which involves enqueuing a message in the Delayed Inbox. Inbox transactions are processed in the order of receipt so the `l2gateway.finalizeInboundTransfer()` call would necessarily be sequenced *after* the

`l2gateway.registerTokens()` call (given that the `l1Token` won't be registered on L1 until the spell has executed).

This means that `l2gateway.finalizeInboundTransfer()` would only fail if the gas params passed to the gov relay call in the spell were inadequate. The L2 spell execution could then fail and would need to be retried. In that case if a deposit's finalization was attempted before the L2 spell execution was retried, then that deposit's finalization would also need to be retried.

The spell process is trusted to use adequate gas params, using large enough buffers to make failure unlikely.

Note however that, should the L2 spell execution fail, anyone could manually redeem the ticket. No need to involve the `L1GovernanceRelay`. So a user could simply redeem the L2 spell execution ticket before redeeming their `finalizeInboundTransfer` ticket.

Cantina Managed: Acknowledged.

3.3.7 Document address parameters that get aliased for `outboundTransfer`s

Severity: Informational

Context: L1TokenGateway.sol#L135-L145

Description: For the `L1TokenGateway.outboundTransfer` and `outboundTransferCustomRefund` functions, some of the L2 address parameters are aliased (if the address is a contract on the L1) while others are not.

- The `to` parameter is never aliased.
- The `refundTo` parameter is aliased if it's a contract on L1 (the `Inbox` performs the aliasing in `createRetryableTicket`).

Recommendation: Consider documenting the parameters that can be aliased in the natspec. See the `L1ArbitrumGateway` natspec for reference.

```
/**
 * @dev L2 address alias will not be applied to the following types of addresses on L1:
 *      - an externally-owned account
 *      - a contract in construction
 *      - an address where a contract will be created
 *      - an address where a contract lived, but was destroyed
 * @param _l1Token L1 address of ERC20
 * @param _refundTo Account, or its L2 alias if it has code in L1, to be credited with excess gas refund in L2
 * @param _to Account to be credited with the tokens in the L2 (can be the user's L2 account or a contract),
↪  not subject to L2 aliasing
            This account, or its L2 alias if it has code in L1, will also be able to cancel the retryable
↪  ticket and receive callvalue refund
 * @param _amount Token Amount
 * @param _maxGas Max gas deducted from user's L2 balance to cover L2 execution
 * @param _gasPriceBid Gas price for L2 execution
 * @param _data encoded data from router and user
 * @return res abi encoded inbox sequence number
 */
//  * @param maxSubmissionCost Max gas deducted from user's L2 balance to cover base submission fee
function outboundTransferCustomRefund(
```

Maker: Natspec updated in commit 6248966b.

Cantina Managed: Natspec updated accordingly.

3.3.8 Missing ERC20 return value check

Severity: Informational

Context: L1TokenGateway.sol#L165, L1TokenGateway.sol#L243

Description: The README mentions that the bridge supports "standard tokens". The bridge does not support tokens that return `false` instead of reverting. Note that ERC20 tokens `SHOULD` but not `MUST` throw, they can also return `false`.

> Callers MUST handle false from returns (bool success). Callers MUST NOT assume that false is never returned! `https://eips.ethereum.org/EIPS/eip-20`

Recommendation: Consider checking the return values of `transfer` and `transferFrom`. Alternatively, further restrict the supported tokens to standard tokens that always revert on unsuccessful transfers.

Maker: README updated in commit 6248966b

Cantina: The support for non-reverting ERC20 tokens was not added but README was updated to reflect this.

3.3.9 Use named arguments for function calls with many arguments

Severity: Informational

Context: L1TokenGateway.sol#L210-L223

Description: The gateways perform function calls with many parameters.

```
sendTxToL2CustomRefund(
    inbox,
    counterpartGateway,
    refundTo,
    from,
    msg.value, // we forward the L1 call value to the inbox
    0, // l2 call value is 0
    L2GasParams({
        _maxSubmissionCost: maxSubmissionCost,
        _maxGas: maxGas,
        _gasPriceBid: gasPriceBid
    }),
    outboundCalldata
);
```

This becomes hard to read and it's hard to see that the argument order is correct.

Recommendation: Consider using named arguments over positional arguments for these calls.

```
sendTxToL2CustomRefund({
    _inbox: inbox,
    _to: counterpartGateway,
    _refundTo: refundTo,
    _user: from,
    _l1CallValue: msg.value,
    _l2CallValue: 0,
    _l2GasParams: L2GasParams({
        _maxSubmissionCost: maxSubmissionCost,
        _maxGas: maxGas,
        _gasPriceBid: gasPriceBid
    }),
    _data: outboundCalldata
});
```

Maker: Code changed in commit 6248966b.

Cantina Managed: Code updated appropriately.

3.3.10 Router deposit support depends on Arbitrum-privileged function

Severity: Informational

Context: L1TokenGateway.sol#L160, DefaultGateway

Description: The `L1TokenGateway` accepts indirect deposits from the router. The `L1GatewayRouter` needs to have the token to gateway path registered. The tokens can either self-register if they support this functionality or it must be registered by the router's owner contracts (controlled by Arbitrum). As the DAI, NST, and other Maker tokens do not support self-registering, Arbitrum needs to call `router.setGateways` to set this path and enable support for Router deposits.

Note that as long as the router gateway path is not registered, the default gateway is used. If users have approved this gateway for any reason, they will use the wrong gateway which will deploy and use a different standard L2 ERC20 bridge contract and funds need to be bridged back.

Recommendation: Ensure the token maps to the correct gateway in the Router before using deposits through the Router.

Maker: Yes, we'll ask the Arbitrum team to call the `onlyOwner` `setGateways`.

Cantina Managed: Acknowledged.