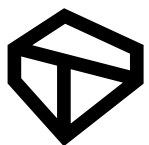




c@inspect
You build, we defend.



TARI

Smart Contract Audit
wXTM Tari Bridge
May 2025



wXTM Tari Bridge Smart Contract Audit

Version: v250528

Prepared for: Tari

May 2025

Security Assessment

1. Executive Summary
2. Summary of Findings
 - 2.3 Solved issues & recommendations
3. Scope
4. Assessment
 - 4.1 Security assumptions
 - 4.2 Decentralization
 - 4.3 Code quality & Testing
5. Detailed Findings
 - WXTM-001 - Adversary can steal bridged funds due to single confirmation required

WXTM-002 - Unchecked safeTransferFrom return value

WXTM-003 - Zero-value transfers allowed

WXTM-004 - Using reinitializer instead of initializer modifier

WXTM-005 - Unused contract implementation

6. Disclaimer

1. Executive Summary

In **May 2025**, **Tari** engaged **Coinspect** to conduct a Smart Contract Security Audit of the wXTM bridge, which includes an Omnichain Fungible Token (OFT) implementation and the bridge contract.

The wXTM bridge enables users to wrap and unwrap Tari's XTM tokens into OFT tokens on any EVM-compatible chain where the OFT is deployed, and vice versa. The OFT setup allows seamless bridging of wrapped tokens across chains allowed by Tari.

 Solved	 Caution Advised	 Resolution Pending
High 0	High 0	High 0
Medium 2	Medium 0	Medium 0
Low 0	Low 0	Low 0
No Risk 3	No Risk 0	No Risk 0
Total 5	Total 0	Total 0

As a result, Coinspect identified two medium-risk issues: one that could allow an attacker to steal funds due to the use of a single confirmation, and another involving the lack of return value checks on ERC20 transfers, which could enable token theft in the event of a contract upgrade.

The audit also flagged three informational issues: zero-value transfers being permitted, use of `reinitializer` instead of `initializer`, and the presence of an unused contract implementation.

2. Summary of Findings

This section provides a concise overview of all the findings in the report grouped by remediation status and sorted by estimated total risk.

2.3 Solved issues & recommendations

These issues have been fully fixed or represent recommendations that could improve the long-term security posture of the project.

Id	Title	Risk
WXTM-001	Adversary can steal bridged funds due to single confirmation required	Medium
WXTM-002	Unchecked safeTransferFrom return value	Medium
WXTM-003	Zero-value transfers allowed	None
WXTM-004	Using reinitializer instead of initializer modifier	None
WXTM-005	Unused contract implementation	None

3. Scope

The scope was set to be the repository at <https://github.com/tari-project/wxtm-bridge-contracts> on commit **1f379c162027fee583595ee506fb3375a9328a83**. During the review, the Tari team shared commit **ca4339efcfab8a908bcc67bbc6575e77e64c93ed**, which Coinspect used for a differential analysis.

Specifically, the following contracts were targeted:

```
8fbfe3c5c1ef43d61066f95f6b272ddb701a821b85812d2cca01fd6e7521edc1  wXTM.sol
b7bca7b99d0b35dc1ec86ec4bdc00fe4eaf55d0906a52a54d343783dd75805a3
wXTMBridge.sol
948e7db74b4bedae0476b709a8348c8c64d26015cfbe93d9d0a9b282dbe1a77a
wXTMController.sol
```

4. Assessment

The contracts outlined in the previous section enable the wrapping and bridging of Tari XTM tokens. Each repository or module is described in detail in the sections that follow.

Notably, the components reviewed appear to be under active development. Coinspect did not find any configurations indicating an imminent deployment to production.

The wXTM Tari bridge enables wrapping Tari XTM tokens for use on EVM-compatible chains which can later be used in cross-chain transfers. To achieve this, Tari employs an upgradeable Omnichain Fungible Token (OFT) that builds on the ERC-20 standard and integrates LayerZero's cross-chain capabilities.

All initial mints—intended exclusively on Ethereum—are expected to be controlled by a the `wXTMController` contract, which defines different roles for minting as explained in section 4.2. When bridging from Tari to an EVM chain, XTM tokens will be locked in a Minotari wallet on Tari, which will allow calling `mint` on the corresponding `wXTM` contract. Although the Tari team plans to mint tokens only on Ethereum, the `mint` function exists on any target chain deploying this contract. For transfers from EVM back to Tari, users call the `bridgeToTari*` functions, which burn tokens on the source chain and emit an event. This event is indexed by Tari-controlled infrastructure, which then releases the equivalent tokens on the Tari chain to the desired address.

On top of this, the `wXTM` contract implements EIP-3009, enabling gas-less transfers through off-chain signed authorizations.

When minting tokens, consider the amount of decimals used by the OFT token. By default, a LayerZero OFT uses the standard ERC-20 precision of 18 decimals for its on-chain balances, and a `sharedDecimals` value of 6 for cross-chain transfers.

Coinspect noted that the gas limit in `EVM_ENFORCED_OPTIONS` is set to 80,000 and recommends profiling gas consumption on each destination chain to verify proper functionality.

4.1 Security assumptions

The review proceeded under these assumptions:

- The OFT delegate, contract owner, and proxy admin remain uncompromised and are governed by a multisig.
- All Tari addresses provided to the bridge are valid.
- The ERC-20 OFT `wXTM` token does not apply a fee on transfers. Otherwise, it will halt bridge operations since it will attempt to burn more tokens than were transferred.

4.2 Decentralization

The project under review exhibits these centralization aspects:

- The contract is a transparent, upgradeable proxy—meaning it can be upgraded at any time.
- The `wXTM` contract minter role can mint tokens arbitrarily without being required to prove XTM tokens were actually locked in Tari.
- Users must rely on the Bridging-to-Tari process, as there's no trustless mechanism for verifying receipt of their XTM tokens on the Tari chain or for reverting the transfer otherwise.

Additionally, the smart contracts employ a role-based access control system. At the highest level, the `ProxyAdmin` (initially the deployer) controls contract upgrades for both `wXTM` and `wXTMController`:

- The `wXTM` token contract has a `DEFAULT_ADMIN_ROLE` (initially granted to the `_delegate` address specified during its initialize function) that can manage other roles, including the `MINTER_ROLE` allowed to mint `wXTM` tokens (meant to be granted to the `wXTMController` contract); this `_delegate` also becomes the LayerZero OApp owner responsible for configuring cross-chain communication (like `setPeer`).
- Once the `wXTMController` possesses the `MINTER_ROLE`, it further delegates minting permissions using its own set of roles, managed by its own `DEFAULT_ADMIN_ROLE`. Specifically, it defines the `LOW_MINTER_ROLE` and `HIGH_MINTER_ROLE` roles; being the former restricted to mint amounts up to a `HIGH_MINT_THRESHOLD`.

Coinspect recommends using separate addresses for contract upgrades and administrative roles. This ensures that if the internal admin is compromised, the attacker cannot also unilaterally upgrade the contract to malicious code, thus preventing a full takeover.

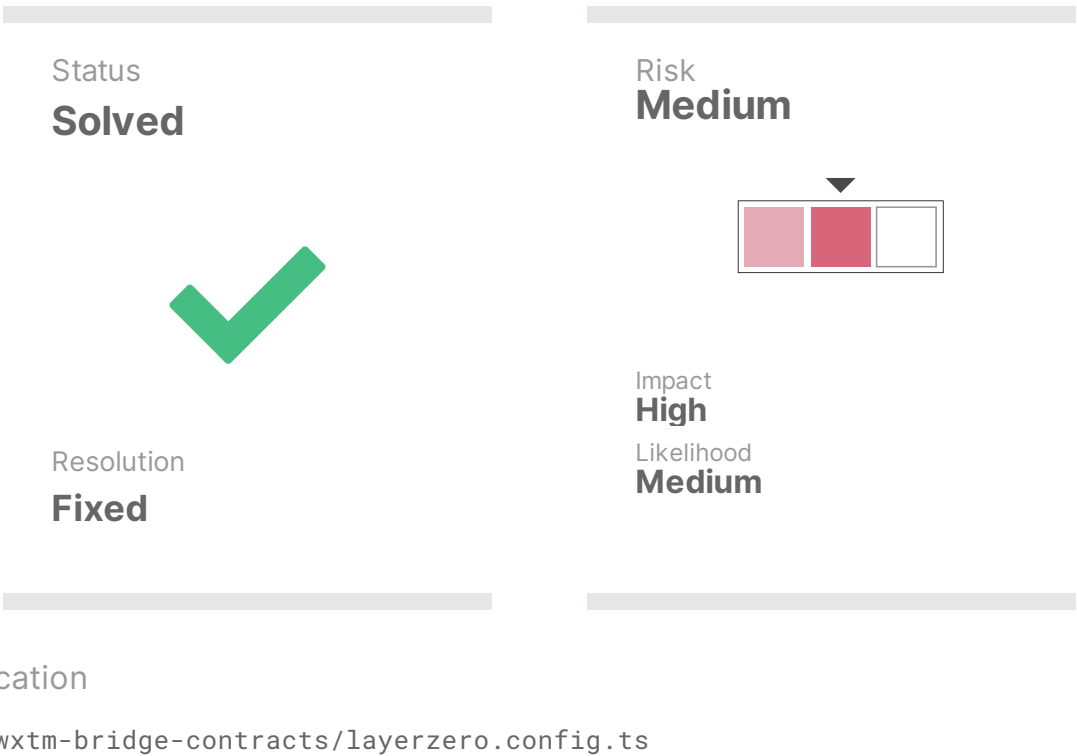
4.3 Code quality & Testing

The smart contract code is clear and easy to follow, but it lacks in-code documentation. Coinspect recommends adopting the NatSpec format for comments. The accompanying test suite achieves 100 % coverage for `wXTMBridge`, while `wXTM` sits at 68 %.

5. Detailed Findings

WXTM-001

Adversary can steal bridged funds due to single confirmation required



Description

Requiring just one confirmation lets adversaries exploit short-lived chain reorganizations to double-spend across the bridge: they send funds, trigger a cross-chain mint, and—when the reorg occurs—end up retaining assets on both chains.

Both directions wait for only one confirmation before treating a message as final. While this speeds up UX on fast chains, it also exposes you to canonical reorgs: a single-block reorg can cause the bridge to replay or revert a transfer it had already marked as settled.

```
[
  optimismContract,    // Chain A contract
  arbitrumContract,    // Chain C contract
  [['LayerZero Labs'], []], // [ requiredDVN[], [
optionalDVN[], threshold ] ]
  [1, 1], // [A→C confirmations, C→A
confirmations]
  [EVM_ENFORCED_OPTIONS, EVM_ENFORCED_OPTIONS], // Chain C
enforcedOptions, Chain A enforcedOptions
],
```

Note that the chains listed in `layerzero.config.ts` are testnets—no production networks are included.

Recommendation

Bump the amount of confirmations required for cross-chain transfer.

Status

Fixed in commit **ab6cc878f19725c183acd67aff2dbb1417081ffa**. The Tari team modified the configuration to align with LayerZero's default confirmation settings (e.g., 20 for Optimism, 20 for Arbitrum, 12 for Avalanche). However, these settings applied only to the development environments—production environment configurations were still missing at the time of the fix review.

WXTM-002

Unchecked safeTransferFrom return value

Status

Solved



Resolution

Fixed

Risk

Medium



Impact

High

Likelihood

Low

Location

wxtm-bridge-contracts/contracts/wXTMBridge.sol

Description

The `bridgeToTari` function in the bridge contract uses the `transferFrom` function without checking its return value. Even if wXTM currently reverts, future upgrades to wXTM (since it's upgradeable) or integrations with different tokens in other contexts might encounter tokens that return false.

Many modern ERC20 token implementations, especially those based on OpenZeppelin's contracts (like wXTM which uses `ERC20Upgradeable`), often revert the transaction with an error message on failure (e.g., insufficient balance or allowance) rather than returning false.

If the token always reverts on failure, then not checking the boolean output is less immediately dangerous because the transaction would halt anyway.

Recommendation

Consider implementing `safeTransferFrom` instead to prevent vulnerabilities arising from future upgrades.

Status

Fixed in commit **b9cc2557fa2603d1f2bf541c620b632d7bb1915**. The Tari team replaced `transferFrom` with `safeTransferFrom`.

WXTM-003

Zero-value transfers allowed

Status Solved	Risk None
	
Resolution Fixed	Impact Recommendation
	Likelihood -

Location

wxtm-bridge-contracts/contracts/wXTMBridge.sol
wxtm-bridge-contracts/contracts/wXTM.sol

Description

The `bridgeToTari` and `burn` functions in the `wXTMBridge` and `wXTM` contracts permit transactions where the value parameter is zero. Zero-value `TokenUnwrapped` events force off-chain systems to handle useless data, driving up indexing costs and clogging logs with noise. Repeated emission of these events can also be used for minor griefing, as attackers can spam the bridge and impose extra processing load on backend infrastructure.

While an ERC20 `transferFrom` of zero tokens from the user to the bridge, followed by a burn of zero tokens from the bridge, are typically no-op operations on-chain (consuming gas but not changing token balances), the contract will still proceed to emit the `TokensUnwrapped` event.

Recommendation

Disallow zero-value operations (`bridgeToTari*` and `burn`).

Status

Fixed in commit [163362a82429c95973dc442f0e8e7a739caa393a](#). The `burn` function now throws the `ZeroAmount` error when attempting to burn zero tokens.

Proof-of-Concept

Running the test below with the command `forge test --match-test test_bridge_to_tari_zero` confirms the bridge allows zero-value transfers.

```
function test_bridge_to_tari_zero() public {
    uint256 value = 0 ether;

    vm.startPrank(user);
    wxtm.approve(address(bridge), value);
    bridge.bridgeToTari("tariExampleAddress", value);
    vm.stopPrank();

    assertEq(wxtm.balanceOf(address(bridge)), 0);
    assertEq(wxtm.balanceOf(user), 10 ether);
}
```

WXTM-004

Using reinitializer instead of initializer modifier

Status Solved	Risk None
	
Resolution Fixed	Impact Recommendation
	Likelihood —

Description

Using `reinitializer(2)` in the `initialize` function of the `wXTM` contract signals that it is meant for version 2 setup—or allows re-initialization up to version 2. On a first deployment, this can confuse readers or suggest a misstep in the usual initialization flow, unless you have a clear reason to start at version 2.

Recommendation

Unless necessary, use `initializer` instead.

Status

Fixed in commit **1dd6ff7567fbe86e1a44978a92c1b076fdd21ada**. The Tari team replaced `reinitializer` by the `initializer` modifier.

WXTM-005

Unused contract implementation

Status

Solved



Resolution

Fixed

Risk

None



Impact

Recommendation

Likelihood

—

Location

wxtm-bridge-contracts/contracts/wXTMBridge.sol:8

Description

The `wXTMBridge` contract inherits from the `Ownable` contract. Note However that Coinspect did not find functionality protected by the `onlyOwner` modifier, meaning that `Ownable` is not required at all. Having unused code exposes an additional attack surface, and it increases operational costs.

Recommendation

Consider removing the `Ownable` contract inheritance.

Status

Fixed in commit **9b0b284d5ad7fe42ef52b1daa6f275dfd5487386**. The Tari team removed the `Ownable` contract from the bridge contract.

6. Disclaimer

The contents of this report are provided "as is" without warranty of any kind. Coinspect is not responsible for any consequences of using the information contained herein.

This report represents a point-in-time and time-boxed evaluation conducted within a specific timeframe and scope agreed upon with the client. The assessment's findings and recommendations are based on the information, source code, and systems access provided by the client during the review period.

The assessment's findings should not be considered an exhaustive list of all potential security issues. This report does not cover out-of-scope components that may interact with the analyzed system, nor does it assess the operational security of the organization that developed and deployed the system.

This report does not imply ongoing security monitoring or guaranteeing the current security status of the assessed system. Due to the dynamic nature of information security threats, new vulnerabilities may emerge after the assessment period.

This report should not be considered an endorsement or disapproval of any project or team. It does not provide investment advice and should not be used to make investment decisions.