GUARDIAN

USDT0

Arbitrum Upgrade

Security Assessment

January 23rd, 2025

Summary

Audit Firm Guardian

Prepared By Owen Thurm, Daniel Gelfand, Xu Winnie, Roberto Reigada,
Zdravko Hristov

Peer Reviewed Osman Ozdemir, Mark Jonathas, Kiki, 3b, Arnie, Cosine, Nicholas Chew, 0xPhaze

Client Firm USDT0

Final Report Date January 23, 2025

Audit Summary

USDT0 engaged Guardian to review the security of the USDT to USDT0 migration on Arbitrum. From the 16th of January to the 20th of January, a team of 5 auditors, peer reviewed by 8 auditors reviewed the source code in scope. All findings have been recorded in the following report.

For a detailed understanding of risk severity, source code vulnerability, and potential attack vectors, refer to the complete audit report below.

🔗 Blockchain network: **Arbitrum**

✅ Verify the authenticity of this report on Guardian’s GitHub: https://github.com/guardianaudits

📊 Code coverage & PoC test suite: https://github.com/GuardianAudits/usdt0-oft-contracts/tree/usdt0-fuzzing

Table of Contents

Project Overview

Project Summary

Project Name	USDT0
Language	Solidity
Codebase	https://github.com/Everdawn-Labs/usdt0-tether-contracts-hardhat
Commit(s)	e6cffe572e9c92e9778c465c04cfae3526a06109

Audit Summary

Delivery Date	January 23, 2025
Audit Methodology	Static Analysis, Manual Review, Test Suite, Contract Fuzzing

Vulnerability Summary

Vulnerability Level	Total	Pending	Declined	Acknowledged	Partially Resolved	Resolved
● Critical	0	0	0	0	0	0
● High	0	0	0	0	0	0
● Medium	0	0	0	0	0	0
● Low	5	0	0	2	0	3
● Info	5	0	0	4	0	1

Audit Scope & Methodology

Vulnerability Classifications

Severity	Impact: *High*	Impact: *Medium*	Impact: *Low*
Likelihood: *High*	● Critical	● High	● Medium
Likelihood: *Medium*	● High	● Medium	● Low
Likelihood: *Low*	● Medium	● Low	● Low

Impact

High	Significant loss of assets in the protocol, significant harm to a group of users, or a core functionality of the protocol is disrupted.
Medium	A small amount of funds can be lost or ancillary functionality of the protocol is affected. The user or protocol may experience reduced or delayed receipt of intended funds.
Low	Can lead to any unexpected behavior with some of the protocol's functionalities that is notable but does not meet the criteria for a higher severity.

Likelihood

High	The attack is possible with reasonable assumptions that mimic on-chain conditions, and the cost of the attack is relatively low compared to the amount gained or the disruption to the protocol.
Medium	An attack vector that is only possible in uncommon cases or requires a large amount of capital to exercise relative to the amount gained or the disruption to the protocol.
Low	Unlikely to ever occur in production.

Audit Scope & Methodology

Methodology

Guardian is the ultimate standard for Smart Contract security. An engagement with Guardian entails the following:

- Two competing teams of Guardian security researchers performing an independent review.
- A dedicated fuzzing engineer to construct a comprehensive stateful fuzzing suite for the project.
- An engagement lead security researcher coordinating the 2 teams, performing their own analysis, relaying findings to the client, and orchestrating the testing/verification efforts.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross-referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts. Comprehensive written tests as a part of a code coverage testing suite.
- Contract fuzzing for increased attack resilience.

Findings & Resolutions

ID	Title	Category	Severity	Status
L-01	Missing Storage Gaps	Storage	● Low	Acknowledged
L-02	Incorrect Confirmations Comment	Documentation	● Low	Resolved
L-03	In Flight Bridges Are Halted	Warning	● Low	Resolved
L-04	Incorrect _EIP712NameHash	Logical Error	● Low	Resolved
L-05	Signatures Invalidated By Name Update	Unexpected Behavior	● Low	Acknowledged
I-01	Depeg Risk	Warning	● Info	Acknowledged
I-02	Bridging From Arbitrum After Migration	Warning	● Info	Acknowledged
I-03	Unnecessary Require Statement	Superfluous Code	● Info	Resolved
I-04	Address Aliasing Allows Blocklist Bypass	Access Control	● Info	Acknowledged
I-05	Name Update Risk	Documentation	● Info	Acknowledged

L-01 | Missing Storage Gaps

Category	Severity	Location	Status
Storage	● Low	ArbitrumExtension.sol	Acknowledged

Description

The current storage layout for `TetherTokenV2Arbitrum` is `ArbitrumExtension` -> `EIP3009`, as it inherits these 2 contracts:

`abstract contract TetherTokenV2Arbitrum is ArbitrumExtension, EIP3009 {`.

However the issue here is that `ArbitrumExtension` is used as a parent contract for `TetherTokenV2Arbitrum`, but it lacks a gap, as it's gap is inside `TetherTokenV2Arbitrum` meaning the storage layout is:

`l2Gateway -> l1Address -> _authorizationStates -> gap49 -> gap48`

Adding a variable inside `TetherTokenV2Arbitrum` would cause a storage collision between it and `EIP3009`'s map: `_authorizationStates`.

Recommendation

Introduce a storage gap in the `ArbitrumExtension` contract to ensure no storage collisions if variables were to be added to that contract in future upgrades.

Resolution

USDT0 Team: Acknowledged.

L-02 | Incorrect Confirmations Comment

Category	Severity	Location	Status
Documentation	● Low	layerzero-arbitrum-prod-step-1.config.ts.sol	Resolved

Description

In the runbook it is mentioned as a step to: Adjust send confirmations for ETH-to-Arbitrum from 100M to 20 blocks after migration. However the configuration for the Ethereum and Arbitrum connection includes 100 Billion confirmations from the Arbitrum-to-ETH connection.

The configuration is correct as it prevents users from bridging their USDT back from arbitrum to ETH before the migration is completed. However this is not in agreement with the comments directionality or magnitude of the confirmations.

The exact magnitude of the confirmations count is not important as long as it is sufficiently large, however the desired direction of the large confirmations configurations should be corrected in the comment.

Recommendation

Update the runbook to read: Adjust send confirmations for Arbitrum-to-ETH from 100B to 20 blocks after migration.

Resolution

USDT0 Team: Resolved.

L-03 | In Flight Bridges Are Halted

Category	Severity	Location	Status
Warning	● Low	Global	Resolved

Description

As a part of the migration process bridges through the Arbitrum gateway will cease to function on the Arbitrum side, however will still be possible from the Ethereum side.

This may be especially harmful for users who have initiated bridges from Ethereum to Arbitrum through the gateway right before the migration takes place.

Any users in this scenario will experience a loss of funds as their USDT on Arbitrum can no longer be credited with the `bridgeMint` function and their USDT on Ethereum cannot be retrieved from the custom gateway contract.

Recommendation

In the migrate function of the `ArbitrumExtensionV2` contract assign the `l1Address` storage value to address(0) after the migration has been initiated by the gateway `outboundTransfer` invocation.

This will trigger refund logic in the `L2ArbitrumGateway` contract where a l2 to l1 token withdrawal is initiated to refund the user upon the l2 receipt of their gateway bridge.

Notice that users will still have to wait 7 days to recoup their funds bridged through the gateway this way.

Therefore consider putting forth a DAO proposal in the future to disable the gateway on Ethereum using the setGateways function on the `L1GatewayRouter` contract to assign the new gateway for the USDT address to the DISABLED `address(1)`.

Resolution

USDT0 Team: Resolved.

L-04 | Incorrect _EIP712NameHash

Category	Severity	Location	Status
Logical Error	● Low	ArbitrumExtension.sol: 503	Resolved

Description

The `_EIP712NameHash` function is overridden to return the `_newName` value, however the `name` function returns the hardcoded value of `super.name()` in the case where the `_newName` is assigned to an empty string.

This will mislead signers and result in the crafting of invalid signatures based on the result of the `name` function.

Recommendation

Correct the `_EIP712NameHash` function to be the hash of the `name` function.

Resolution

USDT0 Team: Resolved.

L-05 | Signatures Invalidated By Name Update

Category	Severity	Location	Status
Unexpected Behavior	● Low	ArbitrumExtension.sol: 503	Acknowledged

Description

Signers can create signatures with specific deadlines with ERC20Permit. The signature is typically expected to be valid until the deadline is reached.

However since the _EIP712NameHash function is now overridden to return the latest _newName value a name update with the updateNameAndSymbol function will invalidate all signatures that were previously made based on the previous name but not yet used.

Recommendation

There is a trade off between using the original name of the token for the domain separator and allowing it to be updated. If the invalidation of past signatures is an acceptable downside to achieve the intuitiveness of using the current name() value in the domain separator, then no changes are necessary.

However note that the DOMAIN_SEPARATOR is exposed as a public function through the ERC20PermitUpgradeable contract. This is what will be used by signers and therefore it is unlikely to be beneficial to base the separator on the current name() value for intuitiveness.

If this behavior of unexpectedly invalidating signatures upon name updates is unacceptable:
For new deployments of the OFTExtension it is fine to keep the same _EIP712NameHash function without an override, since this immutable value is the desired USDT0 name which the contract is initialized with.

However for the ArbitrumExtensionV2 upgrade, the _EIP712NameHash can be overridden, but return an immutable hash which is the hash of the initial name that is assigned in the migrate function. Notice however that this update upon migration will also unexpectedly invalidate signatures, but only once during the migration.

Otherwise no function override can be used and the original token name can continue to exist in the domain separator, making no signatures invalid.

Resolution

USDT0 Team: Acknowledged.

I-01 | Depeg Risk

Category	Severity	Location	Status
Warning	● Info	Global	Acknowledged

Description

During the migration for approximately 7 days, USDT0 on Tether will be un-redeemable for USDT on Ethereum.

This may introduce heightened risk of a depeg event for USDT0 on the Arbitrum network during this period.

Although technically this is the same delay a user would typically have to wait to redeem their USDT on Arbitrum for USDT on Ethereum through the gateway, the lack of a static redemption mechanism during the in motion migration period may incite turbulence in the market.

Recommendation

Be aware of this risk and consider preparing a response plan in the event that a depeg event occurs during the migration.

Resolution

USDT0 Team: Acknowledged.

I-02 | Bridging From Arbitrum After Migration

Category	Severity	Location	Status
Warning	● Info	Global	Acknowledged

Description

During the migration process before the l2 to l1 transaction has been confirmed and executed on Ethereum the USDT0 on Arbitrum is technically unbacked in the Ethereum OFT Adapter contract.

If any USDT0 is allowed to bridge back to ETH from Arbitrum during this period using the Layerzero OFT pathway, it will cause a dearth in the backing ETH USDT for USDT0 on Ink which can prevent redemptions.

This behavior is planned to be solved by assigning a restrictively high confirmations count for bridges coming from Arbitrum to Ethereum.

This is an acceptable solution so long as users are aware that if they bridge USDT from Ethereum to Arbitrum via LayerZero that they cannot immediately bridge back until the migration is complete.

Recommendation

Be aware that users may still bridge from Ethereum to Arbitrum during the migration period. If this is acceptable behavior then ensure that these users or integrators are aware that they cannot bridge back to Ethereum until the migration is complete.

Resolution

USDT0 Team: Acknowledged.

I-03 | Unnecessary Require Statement

Category	Severity	Location	Status
Superfluous Code	● Info	ArbitrumExtension.sol: 479	Resolved

Description

The bridgeBurn function implements a require statement which asserts that isMigrating is true. However isMigrating is immediately assigned to true in the migrate function and therefore this require statement performs no useful validation and is not needed.

Instead the onlyAuthorizedSender access control modifier prevents future bridges through the gateway via the bridgeBurn method as the l2Gateway variable is assigned as the OFT contract after the outboundTransfer invocation.

Recommendation

Consider removing the require statement in the bridgeBurn function entirely as it is unnecessary.

Resolution

USDT0 Team: Resolved.

I-04 | Address Aliasing Allows Blocklist Bypass

Category	Severity	Location	Status
Access Control	● Info	ArbitrumExtension.sol	Acknowledged

Description

The TetherTokenV2Arbitrum contract includes several functions which have the onlyNotBlocked modifier. These functions are intended to be uncallable by a blocked account, however l1 to l2 messages allow an address that is blocked on both l1 and l2 networks to call them.

This is because Arbitrum l1 to l2 retryableTickets use an aliased msg.sender, shifted by an OFFSET of 0x1111000000000000000000000000000000001111. Thus the sender will not be a blocked address and these invocations are allowed.

Specifically, since the transferWithAuthorization functions do not rely on the msg.sender as a signer, sender, or receiver a blacklisted address can call these addresses through aliasing.

Recommendation

There is no difference between a blacklisted address invoking this from l1 and the owner of the blacklisted address simply calling the function from a different address.

Furthermore the main functionality of a blocked address being unable to transfer funds is still held through the _beforeTokenTransfer function. Therefore this is meant only as an informational note.

Resolution

USDT0 Team: Acknowledged.

I-05 | Name Update Risk

Category	Severity	Location	Status
Documentation	● Info	Global	Acknowledged

Description

The update of the USDT token's name to USDT0 on Arbitrum may cause unexpected behavior for other applications relying on USDT in either their Smart Contracts or Frontends.

Recommendation

During our review no protocols were identified that would be directly impacted on-chain. However ideally integrators can be made aware of this update and are able to make appropriate changes if necessary. If anything this finding can serve as documentation for this risk.

Resolution

USDT0 Team: Acknowledged.

Disclaimer

This report is not, nor should be considered, an “endorsement” or “disapproval” of any particular project or team. This report is not, nor should be considered, an indication of the economics or value of any “product” or “asset” created by any team or project that contracts Guardian to perform a security assessment. This report does not provide any warranty or guarantee regarding the absolute bug-free nature of the technology analyzed, nor do they provide any indication of the technologies proprietors, business, business model or legal compliance.

This report should not be used in any way to make decisions around investment or involvement with any particular project. This report in no way provides investment advice, nor should be leveraged as investment advice of any sort. This report represents an extensive assessing process intending to help our customers increase the quality of their code while reducing the high level of risk presented by cryptographic tokens and blockchain technology.

Blockchain technology and cryptographic assets present a high level of ongoing risk. Guardian’s position is that each company and individual are responsible for their own due diligence and continuous security. Guardian’s goal is to help reduce the attack vectors and the high level of variance associated with utilizing new and consistently changing technologies, and in no way claims any guarantee of security or functionality of the technology we agree to analyze.

The assessment services provided by Guardian is subject to dependencies and under continuing development. You agree that your access and/or use, including but not limited to any services, reports, and materials, will be at your sole risk on an as-is, where-is, and as-available basis. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. The assessment reports could include false positives, false negatives, and other unpredictable results. The services may access, and depend upon, multiple layers of third-parties.

Notice that smart contracts deployed on the blockchain are not resistant from internal/external exploit. Notice that active smart contract owner privileges constitute an elevated impact to any smart contract’s safety and security. Therefore, Guardian does not guarantee the explicit security of the audited smart contract, regardless of the verdict.

About Guardian Audits

Founded in 2022 by DeFi experts, Guardian Audits is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian Audits upholds best-in-class security while achieving our mission to relentlessly secure DeFi.

To learn more, visit https://guardianaudits.com

To view our audit portfolio, visit https://github.com/guardianaudits

To book an audit, message https://t.me/guardianaudits