

USDT0 · SECURITY REVIEW

USDT0

USDT0 Stellar Contracts.

A security review of USDT0 Stellar Contracts.

CLIENT	PROJECT	AUDITORS	AUDIT WINDOW	FINAL REPORT
USDT0	USDT0 Stellar Contracts	Jakub Zmyslowski	August 7 – August 11, 2026	August 11, 2026

§ 00

Contents.

13 pages: overview, confidence ranking, scope, 4 findings, disclaimer, and about Guardian.

§ 01	Overview	P. 03
§ 02	Confidence Ranking	P. 04
§ 03	Scope & Classifications	P. 05
§ 04	Findings	P. 06
	Main Review · 4 findings	P. 07
§ 05	Disclaimer	P. 12
§ 06	About Guardian	P. 13

§ 01

Overview.

Guardian reviewed USDT0 Stellar Contracts across a single review round. The final report includes 4 Informational findings.

AUDIT FIRM

Guardian

CLIENT

USDT0

PROJECT

USDT0 Stellar Contracts

PREPARED BY

Jakub Zmyslowski

AUDIT WINDOW

August 7 – August 11, 2026

FINAL REPORT

August 11, 2026

REPOSITORY

<https://github.com/Everdawn-Labs/usdt0-oft-contracts>

METHODOLOGY

Smart contract · Manual · Main Review

MAIN REVIEW COMMIT

ef3db2a637a4

1.1 Confidence Ranking

5 / 5**Very High
Confidence**

Guardian assigns a Confidence Ranking of 5 out of 5, combining finding severity, remediation outcome, and the reviewer's assessment of latent risk. The full scale is on the next page.

1VERY LOW
CONFIDENCE**2**LOW
CONFIDENCE**3**MODERATE
CONFIDENCE**4**HIGH
CONFIDENCE**5**VERY HIGH
CONFIDENCE

1.2 Findings at a Glance

SEVERITY	TOTAL	PENDING	ACK	PARTIAL	RESOLVED
 Critical	0	0	0	0	0
 High	0	0	0	0	0
 Medium	0	0	0	0	0
 Low	0	0	0	0	0
 Informational	4	0	4	0	0
All severities	4	0	4	0	0

§ 02

Confidence Ranking.

Guardian assigns a project-level confidence score using the scale below. The ranking combines finding severity, remediation outcome, and the reviewer's assessment of latent risk.

CONFIDENCE RANKING	DEFINITION AND RECOMMENDATION	RISK PROFILE
5 Very High Confidence ASSIGNED	Codebase is mature, clean, and secure. No High or Critical vulnerabilities were found. Follows modern best practices with high test coverage and thoughtful design. Recommendation: code is highly secure at time of audit.	0 High/Critical findings and few Low/Medium severity findings.
4 High Confidence	Code is clean, well-structured, and adheres to best practices. Design patterns are sound and test coverage is strong. Recommendation: suitable for deployment after remediations; consider periodic review with changes.	0-1 High/Critical findings per engagement week and little to no Medium severity findings.
3 Moderate Confidence	Code is functional but shows gaps in defensive programming or test coverage. Recommendation: remediate the findings and re-review the changes before deployment.	Some High/Critical findings, with various Medium severity findings.
2 Low Confidence	Recurring issues suggest systemic gaps rather than isolated mistakes. Recommendation: remediate, then commission a follow-up review at a frozen commit.	Over 6 High and Critical findings across the engagement.
1 Very Low Confidence	The volume and severity of findings indicate the codebase is not ready for production. Recommendation: substantial rework, then a full re-review.	Over 12 High and Critical findings across the engagement.

§ 03

Scope & Classifications.

Guardian reviewed USDT0 Stellar Contracts in <https://github.com/everdawn-labs/usdt0-ofc-contracts>.

3.1 Severity Matrix

	IMP · HIGH	IMP · MED	IMP · LOW
LIK · HIGH	Critical	High	Medium
LIK · MED	High	Medium	Low
LIK · LOW	Medium	Low	Low

IMPACT · HIGH

Significant loss of assets, harm to a group of users, or core protocol functionality is disrupted.

IMPACT · MEDIUM

Small amount of funds can be lost, or ancillary functionality is affected. Users may experience reduced or delayed receipt.

IMPACT · LOW

Unexpected behavior in some functionality that is notable but does not meet the criteria for higher severity.

LIKELIHOOD · HIGH

Possible with reasonable assumptions that mimic on-chain conditions; cost is low compared to amount gained.

LIKELIHOOD · MEDIUM

Possible only in uncommon cases or requires a large amount of capital relative to disruption caused.

LIKELIHOOD · LOW

Unlikely to ever occur in production.

§ 04

Findings.

Grouped by round. Visible IDs reset by severity within each round, so a finding is named the same way in the report as it was in the channel it was written in. Status is where each finding ended up; its own page says how.

ID	TITLE	CATEGORY	SEVERITY	STATUS	PAGE
MAIN REVIEW					
I-01	Stellar Trustline Gaps Block OFT Delivery	Documentation	INFORMATIONAL	ACKNOWLEDGED	p. 7
I-02	Stellar Pause Leaves Inbound Transfers Pending	Documentation	INFORMATIONAL	ACKNOWLEDGED	p. 8
I-03	Quoted Maximum Ignores Dust Rounding	Rounding	INFORMATIONAL	ACKNOWLEDGED	p. 9
I-04	Unsafe Single-Step SAC Admin Transfer	Best Practices	INFORMATIONAL	ACKNOWLEDGED	p. 11

INFORMATIONAL I-01

ACKNOWLEDGED

Stellar Trustline Gaps Block OFT Delivery

CATEGORY	Documentation
LOCATION	https://developers.stellar.org/docs/learn/fundamentals/stellar-data-structures/accounts#trustlines https://developers.stellar.org/docs/tokens/stellar-asset-contract#interacting-with-classic-stellar-assets
REVIEW	Main Review

— DESCRIPTION

The EVM OFT accepts a 32-byte destination but cannot determine whether the corresponding Stellar recipient can hold the Stellar asset. After tokens are burned on EVM, the Stellar OFT resolves the payload as a deployed C-address or, if no corresponding contract exists, as a G-address and attempts to mint through the Stellar Asset Contract.

A classic Stellar account identified by a G-address can receive and hold an issued asset only through a trustline for that asset.

Therefore, the recipient must have an authorized trustline whose remaining limit can accommodate the transferred amount. The Stellar OFT only invokes the mint operation and does not create a missing trustline, as trustline creation requires authorization from the recipient and sufficient XLM reserve.

If these requirements are not satisfied, destination execution reverts after the source tokens have already been burned. The message remains retryable, but the tokens remain unavailable until the trustline is established or the intended contract is deployed and the message is retried.

— RECOMMENDATION

Document that G-address recipients must exist and establish an authorized trustline for the Stellar asset before receiving an inbound transfer.

— RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-02

ACKNOWLEDGED

Stellar Pause Leaves Inbound Transfers Pending

CATEGORY	Documentation
LOCATION	https://github.com/GuardianOrg/usdt0-oft-contracts-team1-1785973686068/blob/ef3db2a637a43f33dc17950afe4873d7c830c793/stellar-contracts/oft/src/oft.rs#L93-L140
REVIEW	Main Review

DESCRIPTION

The Stellar pause mechanism blocks both outgoing debits and incoming credits. Outgoing transfers are pause protected indirectly, as both `debit` implementations invoke `__debit_view` before locking or burning tokens, and that function checks the pause state. Incoming transfers are rejected directly by `__credit`.

This creates a cross-chain asymmetry because the EVM OFT has no corresponding pause mechanism and cannot prevent transfers targeting a paused Stellar deployment. An EVM-to-Stellar transfer can burn tokens successfully on EVM, after which destination execution reverts when Stellar attempts to credit the recipient. Conversely, a Stellar-to-EVM transfer is rejected before any Stellar tokens are locked or burned.

The failed inbound message remains retryable, so the tokens are not lost. However, they remain unavailable until Stellar is unpaused and the message is retried.

RECOMMENDATION

Document that pausing Stellar also suspends settlement of transfers already initiated from other EVM chains. Monitor failed inbound deliveries and establish a process to retry them after unpausing.

RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-03

ACKNOWLEDGED

Quoted Maximum Ignores Dust Rounding

CATEGORY	Rounding
LOCATION	https://github.com/Everdawn-Labs/usdt0-oft-contracts/blob/c8bdaa0caf71f48f9cce2c3838809603c93bdd5a/stellar-contracts/oft/src/oft.rs#L62-L80 https://github.com/Everdawn-Labs/usdt0-oft-contracts/blob/c8bdaa0caf71f48f9cce2c3838809603c93bdd5a/stellar-contracts/oft/src/oft.rs#L93-L133
REVIEW	Main Review

DESCRIPTION

The `quote_oft` function derives the maximum input from the available outbound rate limit capacity. When a fee is configured, the contract converts the available post-fee capacity into an estimated maximum pre-fee input, intending the result to represent the largest input accepted by `send`.

```
let mut max_amount_ld = Self::rate_limit_capacity(env, &Direction::Outbound,
    send_param.dst_eid);
if max_amount_ld != i128::MAX {
    // Back-calculate pre-fee amount so max_amount_ld represents the largest input
    // to send()
    max_amount_ld = Self::__amount_before_fee(env, send_param.dst_eid,
        max_amount_ld);
} else {
    // No rate limit configured – use the bounded default
    max_amount_ld = u64::MAX as i128;
}
limit.max_amount_ld = max_amount_ld;
```

However, during execution, the contract rounds the amount down according to the decimal conversion rate before applying the rate limit.

```
let (amount_sent_ld, amount_received_ld) = if !has_fee {
    // No fee: dust stays with sender (default OFT behavior)
    let amount_sent_ld = oft_utils::remove_dust(amount_ld, conversion_rate);
    (amount_sent_ld, amount_sent_ld)
} else {
    // With fee: match EVM OFTFee behavior
    // - sender pays full amount_ld (no dust removed), dust is absorbed into the
    //   charged fee
    let fee = Self::__fee_view(env, dst_eid, amount_ld);
    let amount_received_ld = oft_utils::remove_dust(amount_ld - fee,
        conversion_rate);
    (amount_ld, amount_received_ld)
```

I-03 Quoted Maximum Ignores Dust Rounding

CONTINUED

The rate limiter consequently checks the normalized `amount_received_1d`, rather than the original input. As a result, inputs above the reported maximum can still be accepted when fee calculation and dust removal round them down to an amount within the available capacity. Consequently, integrations relying on the quoted maximum may unnecessarily reject valid transfers.

— RECOMMENDATION

Calculate `max_amount_1d` using the same fee and dust rounding as `send`, so it reflects the largest accepted input.

— RESOLUTION

Status: Acknowledged.

INFORMATIONAL I-04

ACKNOWLEDGED

Unsafe Single-Step SAC Admin Transfer

CATEGORY	Best Practices
LOCATION	https://github.com/Everdawn-Labs/usdt0-ofc-contracts/blob/c8bdaa0caf71f48f9cce2c3838809603c93bdd5a/stellar-contracts/sac-manager/src/sac_manager.rs#L69-L72 https://developers.stellar.org/docs/build/guides/tokens/custom-sac-admin
REVIEW	Main Review

DESCRIPTION

The SAC Manager allows an account holding `ADMIN_MANAGER_ROLE` to transfer SAC administration immediately to any supplied address.

The new administrator is not validated and is not required to accept the transfer. After execution, the Manager loses SAC authority and cannot restore itself because any subsequent administrative transfer must be authorized by the new administrator.

Consequently, transferring administration to an incorrect or nonfunctional address can permanently disable minting and other SAC administrative operations.

RECOMMENDATION

Implement a two-step transfer in which the current administrator proposes a new administrator and the proposed administrator authenticates an acceptance before the Manager calls the SAC's `set_admin`.

RESOLUTION

Status: Acknowledged.

§ 05

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.

§ 06

About Guardian.

Guardian is a smart contract security firm. Every engagement is worked by two independent teams competing on the same scope, each with its own tooling and methods, so no single reviewer's blind spot survives.

\$45B+DIGITAL ASSETS UNDER
REVIEW**200+**

ENGAGEMENTS COMPLETED

500+CRITICAL & HIGH FINDINGS
REPORTED

Reach us at audits@guardianaudits.com.