GA GUARDIAN

Olympus

LZ Integration

Security Assessment

June 4th, 2026

Summary

Audit Firm Guardian

Prepared By Ciphky, CrypticDefense

Client Firm Olympus

Final Report Date June 4, 2026

Audit Summary

Olympus engaged Guardian to review its LayerZero bridge integration. From April 16, 2026 to April 21, 2026, Guardian reviewed the LZ bridge gateway, cross-chain bridge periphery, LayerZero configuration library, security upgrade proposal, and activation proposal.

Confidence Ranking

Given the lack of High and Critical issues detected during the main review, Guardian assigns a Confidence Ranking of 5 to the protocol. Guardian advises the protocol to consider periodic review with future changes. For detailed understanding of the Guardian Confidence Ranking, please see the rubric on the following page.

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

Guardian Confidence Ranking

Confidence Ranking	Definition and Recommendation	Risk Profile
5: Very High Confidence	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. Low risk of latent critical issues.	0 High/Critical findings and few Low/Medium severity findings.
4: High Confidence	Code is clean, well-structured, and adheres to best practices. Only 1 Significant issue was uncovered per week. 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 issues. Varied Low severity findings.
3: Moderate Confidence	Medium-severity and occasional High-severity issues found. Code is functional, but there are concerning areas (e.g., weak modularity, risky patterns). No critical design flaws, though some patterns could lead to issues in edge cases. **Recommendation:** Address issues thoroughly and consider a targeted follow-up audit depending on code changes.	1-2 High/Critical findings per engagement week.
2: Low Confidence	Code shows frequent emergence of Critical/High vulnerabilities. Audit revealed recurring anti-patterns, weak test coverage, or unclear logic. These characteristics suggest a high likelihood of latent issues. **Recommendation:** Post-audit development and a second audit cycle are strongly advised.	2-4 High/Critical findings per engagement week. Or additional High/Critical findings uncovered in remediation review which have not been resolved and confirmed by Guardian.
1: Very Low Confidence	Code has systemic issues. Multiple High/Critical findings (≥5/week), poor security posture, and design flaws that introduce compounding risks. Safety cannot be assured. **Recommendation:** Halt deployment and seek a comprehensive re-audit after substantial refactoring.	≥5 High/Critical findings and overall systemic flaws.

Table of Contents

Project Overview

Project Summary

Project Name	Olympus
Codebase	https://github.com/OlympusDAO/olympus-v3
Commit(s)	Main Review: aaae73e623307712c952179a38cfb66637c0ae21 Remediation: 3068400436abf9946655ae175541e193ec76e599

Audit Summary

Delivery Date	June 4, 2026
Audit Methodology	Static Analysis, Manual Review

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	1	0	0	0	0	1
Low	3	0	0	0	0	3
Info	16	0	0	2	0	14

Audit Scope & Methodology

Contract,source,total,comment

olympus-v3/src/periphery/bridge/LZCrossChainBridge.sol,60,106,24
olympus-v3/src/policies/bridge/LZBridgeGateway.sol,297,568,177
olympus-v3/src/libraries/LZConfigLib.sol,161,280,67
olympus-v3/src/proposals/LZBridgeSecurityUpgradeProposal.sol,329,448,66
olympus-v3/src/proposals/LZBridgeActivator.sol,150,235,55
source count: {
  total: 1637,
  source: 997,
  comment: 389
}

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
M-01	Optional DVNs Inherit LayerZero Defaults	Logical Error	Medium	Resolved
L-01	Peer Update Strands Verified Messages	Compatibility	Low	Resolved
L-02	Bridge Fee Event Can Overstate Paid Fee	Events	Low	Resolved
L-03	L2 Post-Batch Validation Omits Endpoint Checks	Validation	Low	Resolved
I-01	No Receive-Side Validation Of Recipient	Validation	Info	Resolved
I-02	Skip/Burn Bypass bridgedSupply Accounting	Trust Assumptions	Info	Resolved
I-03	No Token Rescue On Gateway	Best Practices	Info	Resolved
I-04	Dual-Bridge Window During Migration	Trust Assumptions	Info	Acknowledged
I-05	sendOhm Does Not Validate Recipient Address	Gas Optimization	Info	Resolved
I-06	Consider Additional DVN For Added Resilience	Suggestion	Info	Resolved
I-07	Unused V1 Chain ID Constants In LZConfigLib	Superfluous Code	Info	Acknowledged
I-08	Old Bridge Deactivated Without Disabled Check	Trust Assumptions	Info	Resolved
I-09	Unresolved TODO Comments	Best Practices	Info	Resolved

Findings & Resolutions

ID	Title	Category	Severity	Status
I-10	Enabled/Disable Events Omit Data Parameters	Events	Info	Resolved
I-11	Non-Canonical Chains Have No Minting Cap	Trust Assumptions	Info	Resolved
I-12	Periphery Can Use Stale Gateway	Informational	Info	Resolved

M-01 | Optional DVNs Inherit LayerZero Defaults

Category	Severity	Location	Status
Logical Error	Medium	LZConfigLib.sol	Resolved

Description

The bridge intends to pin LayerZero verification to exactly two required DVNs per route, but LZConfigLib.encodeUlnConfig() leaves optional DVNs in LayerZero's default-inherited state.

The helper encodes optionalDVNCount = 0 with an empty optionalDVNs array. In LayerZero V2 OApp ULN config, 0 does not mean "zero optional DVNs"; it means "use the default config." To explicitly configure no optional DVNs, the value must be type(uint8).max.

This conflicts with the proposal's stated goal of eliminating LayerZero default drag-along and using dual-DVN verification: LayerZero Labs + Google Cloud for non-Berachain routes, and LayerZero Labs + Nethermind for Berachain routes. If current defaults have no optional DVNs, the resolved config may look correct today, but the raw app config still inherits defaults.

If LayerZero later adds default optional DVNs or changes the optional DVN threshold for a route, Olympus can inherit that change without governance action. Messages may then require an extra default DVN, increasing fees or blocking delivery if that DVN is unavailable or unsupported. In the bridge flow, users can burn OHM on the source chain while the destination mint remains stuck until the inherited requirement is satisfied or admins recover the message.

Recommendation

Use LayerZero's NIL sentinel for no optional DVNs: set optionalDVNCount = type(uint8).max, optionalDVNThreshold = 0, and keep optionalDVNs empty.

Resolution

Olympus: Resolved.

L-01 | Peer Update Strands Verified Messages

Category	Severity	Location	Status
Compatibility	Low	src/policies/bridge/LZBridgeGateway.sol:319	Resolved

Description

Calling `setPeer` can update or clear the peer for an EID. If messages from that peer are already verified on the LZ endpoint but not yet executed, they become permanently undeliverable, and `lzReceive` reverts on the peer check. Admin must proceed to manually recover each stranded message (i.e., skip, clear).

Recommendation

Document this behavior. Consider a two-step peer removal (disable receives → wait for in-flight messages to settle → remove peer) or check for pending messages before clearing.

Resolution

Olympus: Resolved.

L-02 | Bridge Fee Event Can Overstate Paid Fee

Category	Severity	Location	Status
Events	Low	src/periphery/bridge/LZCrossChainBridge.sol:68	Resolved

Description

LZCrossChainBridge.sendOhm() emits msg.value as the fees field in Bridged, but msg.value is the native value supplied upfront, not necessarily the actual LayerZero fee paid.

LayerZero V2 refunds excess native value to the configured refund address. In this bridge flow, the gateway passes the user as the refund address, so any overpayment is returned, but the event still records the full original msg.value.

The actual fee charged by LayerZero is returned in MessagingReceipt.fee, which the gateway receives from EndpointV2.send(). However, that value is not surfaced in the Bridged event.

This could also lead to misaccounting in off-chain accounting systems that rely on the emitted fees value.

Recommendation

Emit MessagingReceipt.fee.nativeFee as the paid fee, or rename/document the field as native value supplied.

Resolution

Olympus: Resolved.

L-03 | L2 Post-Batch Validation Omits Endpoint Checks

Category	Severity	Location	Status
Validation	Low	src/scripts/ops/batches/LZBridgeGatewayL2Batch.sol:311	Resolved

Description

`LZBridgeGatewayL2Batch._validateConfigureAndEnable` verifies the gateway is enabled, peers are set, and enforced options exist. However, unlike the Ethereum proposal, it does not verify that the LZ libraries are pinned (SendUln302/ReceiveUln302), ULN config (DVN addresses, confirmations), or executor config, The Ethereum proposal's `_validate` reads these values and endpoint state directly. A misconfigured L2 endpoint setup would therefore not be caught by the L2 post-batch validation.

Recommendation

Extend `_validateConfigureAndEnable` to read endpoint configuration via the gateway's `ILZEndpointV2Admin` view functions and verify libraries, DVNs, confirmations, and executor match the expected values. Mirror the checks in `LZBridgeSecurityUpgradeProposal._validateLZConfig`.

Resolution

Olympus: Resolved.

I-01 | No Receive-Side Validation of Recipient

Category	Severity	Location	Status
Validation	Info	src/policies/bridge/LZBridgeGateway.sol:301	Resolved

Description

`_receiveBridgeOhm` decodes `(address to, uint256 amount)` from the cross-chain message but does not validate `to`. The send side checks `to != address(0)`, but the receive side trusts the payload entirely. If a situation occurs where `to = address(0)` is sent to the receiver, the OHM mint reverts, stranding the message. If `to = address(gateway)`, OHM is minted to the gateway and locked with no recovery path.

Recommendation

Add `_requireNonzeroAddress(to, "to")` in `_receiveBridgeOhm`, consider also rejecting `to == address(this)`.

Resolution

Olympus: Resolved.

I-02 | Skip/Burn Bypass bridgedSupply Accounting

Category	Severity	Location	Status
Trust Assumptions	Info	src/policies/bridge/LZBridgeGateway.sol:465	Resolved

Description

The LZ V2 message recovery functions `skip`, `burn`, and `clear` are executed directly to the endpoint without adjusting `bridgedSupply` or mint approval on the canonical chain. When `bridge_admin` uses skip or burn to permanently discard an inbound message, the OHM burned on the source chain is never minted on the destination, but `bridgedSupply` remains inflated. Admin must manually call `decreaseBridgedSupply` to enforce the `bridgedSupply` and mint approval are up-to-date, while nothing in the code enforces or signals this.

Recommendation

Document that `skip` and `burn` of messages require a corresponding `decreaseBridgedSupply` call on the canonical chain.

Resolution

Olympus: Resolved.

I-03 | No Token Rescue on Gateway

Category	Severity	Location	Status
Best Practices	Info	LZBridgeGateway.sol LZCrossChainBridge.sol	Resolved

Description

Neither `LZBridgeGateway` nor `LZCrossChainBridge` has a mechanism to recover tokens accidentally sent to them. ERC20 and native token is permanently locked, and since the gateway's lzReceive is payable, this can increase the chance user stuck funds.

Recommendation

Consider adding an admin-gated rescue function for stuck tokens.

Resolution

Olympus: Resolved.

I-04 | Dual-Bridge Window During Migration

Category	Severity	Location	Status
Trust Assumptions	Info	src/proposals/LZBridgeSecurityUpgradeProposal.sol:73	Acknowledged

Description

The OCG proposal enables the new `LZBridgeGateway` but does not deactivate the old `CrossChainBridge`. Deactivation is deferred to a separate DAO multisig batch after proposal execution. During this window both bridges are active. The old bridge lacks `bridgedSupply` tracking, DVN pinning, and receive-side disable checks. Transfers through the old bridge during this window bypass bridgedSupply tracking, explicit DVN pinning, and receive-side disable controls.

Recommendation

Minimize the window between proposal execution and old bridge deactivation. Document the maximum acceptable delay and ensure monitoring is in place to alert if the old bridge is used after the new one is live.

Resolution

Olympus: Acknowledged.

I-05 | sendOhm Does Not Validate Recipient Address

Category	Severity	Location	Status
Gas Optimization	Info	src/policies/bridge/LZBridgeGateway.sol:209	Resolved

Description

`LZCrossChainBridge.sendOhm` validates `amount_ != 0` but does not check `to_ != address(0)`. The gateway's `burnAndSend` does validate this, but the user has already paid gas for the OHM transfer before hitting that revert.

Recommendation

Add `_requireNonzeroAddress(to_, "to")` in `sendOhm` to fail fast.

Resolution

Olympus: Resolved.

I-06 | Consider Additional DVN for Added Resilience

Category	Severity	Location	Status
Suggestion	Info	src/libraries/LZConfigLib.sol:271-L272	Resolved

Description

The bridge uses 2-of-2 required DVNs per route, which meets LayerZero's multi-DVN recommendation and would have added an extra layer of protection against the recent KelpDAO exploit (April 2026) which relied on a 1-of-1 configuration. However, a 2-of-2 setup means compromising both DVNs still enables message forgery. A 3-of-3 configuration would provide additional resilience against attackers targeting DVN infrastructure, as demonstrated in the KelpDAO incident.

Recommendation

Evaluate whether adding a third required DVN is operationally feasible to add an extra layer of security.

Resolution

Olympus: Resolved.

I-07 | Unused V1 Chain ID Constants in LZConfigLib

Category	Severity	Location	Status
Superfluous Code	Info	src/libraries/LZConfigLib.sol:117-L125	Acknowledged

Description

`LZConfigLib` defines V1 LayerZero chain ID constants (`ETH_CHAIN_ID = 101`, `ARB_CHAIN_ID = 110`, etc.) that are not referenced anywhere in the codebase. All active configuration uses the V2 endpoint IDs (`ETH_EID = 30101`, `ARB_EID = 30110`, etc.).

Recommendation

Remove the unused V1 chain ID constants to reduce confusion between V1 and V2 identifiers.

Resolution

Olympus: Acknowledged.

I-08 | Old Bridge Deactivated Without Disabled Check

Category	Severity	Location	Status
Trust Assumptions	Info	src/scripts/ops/batches/LZCrossChainBridgeBatch.sol:33-L59	Resolved

Description

`LZCrossChainBridgeBatch.setup` deactivates the old `CrossChainBridge` in the Kernel without first verifying that `bridgeActive == false`. A separate `disableOldBridge` entry point exists and is expected to run beforehand, but `setup` does not enforce this ordering. If setup is ran before `disableOldBridge`, user transactions submitted in the intervening window could still interact with the old bridge.

Recommendation

Add a pre-condition check in setup that verifies the old bridge's `bridgeActive` state is already false, or include the `disable` call in the same batch to guarantee ordering.

Resolution

Olympus: Resolved.

I-09 | Unresolved TODO Comments

Category	Severity	Location	Status
Best Practices	Info	src/scripts/deploy/DeployV3.s.sol:67 src/scripts/ops/batches/LZBridgeGatewayBatch.sol:90 src/proposals/LZBridgeSecurityUpgradeProposal.sol:64 src/proposals/LZBridgeSecurityUpgradeProposal.sol:95-L98	Resolved

Description

Several TODO comments remain in the proposal, deployment scripts, and batch scripts that should be resolved before deployment:

`LZBridgeSecurityUpgradeProposal.sol`: proposal ID is hardcoded to 15 with a TODO. The proposal description contains TODO placeholders for the audit report link, PR link, and RFC/OIP reference.

`LZBridgeGatewayBatch.sol`: initBridgedSupply reads from an args file with a TODO noting the value must be set before execution.

`DeployV3.s.sol`: TODOs flag refactoring and error handling improvements.

Recommendation

Resolve all TODO comments before deployment, followed by removing the comments to avoid confusion.

Resolution

Olympus: Resolved.

I-10 | Enabled/Disable Events Omit Data Parameters

Category	Severity	Location	Status
Events	Info	LZBridgeGateway.sol PolicyEnabler.sol	Resolved

Description

`PolicyEnabler.enable` and `PolicyEnabler.disable` each accept a bytes calldata data parameter (enableData_ and disableData_) but do not include these in the Enabled and Disabled events. For policies that use this data for configuration at enable/disable time, the values are not captured in on-chain logs.

Recommendation

Consider emitting `enableData_` and `disableData_` as part of the Enabled and Disabled events.

Resolution

Olympus: Resolved.

I-11 | Non-Canonical Chains Have No Minting Cap

Category	Severity	Location	Status
Trust Assumptions	Info	src/policies/bridge/LZBridgeGateway.sol:602	Resolved

Description

In non-canonical chains, `_receiveBridgeOhm` uses JIT self-approval with no upper bound:

```
    MINTR.increaseMintApproval(address(this), amount);
    MINTR.mintOhm(to, amount);
```

If for example LayerZero verification infrastructure for a specific route is compromised (e.g. DVN compromised as demonstrated by the KelpDAO incident in April 2026), unlimited `OHM` can be minted on the non-canonical chain. The canonical chain's `bridgedSupply` cap does not protect non-canonical chains.

The `RateLimiter` inherited by the gateway caps outbound transfers only. `_inflow` reduces consumed outbound capacity, freeing it for further sends, rather than capping inbound volume. Rate limits therefore do not provide an effective cap on inbound mints if LZ verification is compromised.

Recommendation

Consider whether a `maximum-mint cap` should be introduced at the gateway level on non-canonical chains as an additional defense layer.

Resolution

Olympus: Resolved.

I-12 | Periphery Can Use Stale Gateway

Category	Severity	Location	Status
Informational	Info	DeployV3.s.sol	Resolved

Description

The saved LZ bridge deployment sequences deploy LZCrossChainBridge before LZBridgeGateway.

DeployV3.deploy() executes the sequence in order and stores each deployed address in the in-memory deployedTo map only after that deployment completes. DeployV3._getAddressNotZero() first checks this in-memory map, then falls back to env.json.

However, deployLZCrossChainBridge() reads olympus.policies.LZBridgeGateway during construction. Since LZBridgeGateway appears later in both saved LZ deployment sequences, the newly deployed gateway is not available in deployedTo when the periphery is constructed.

As a result, the periphery either reverts if env.json has no gateway address, or it is deployed with whatever gateway address was already saved in env.json. If env.json contains an older gateway address, the new periphery is wired to that stale gateway instead of the gateway deployed later in the same sequence.

Recommendation

Deploy LZBridgeGateway before LZCrossChainBridge, or explicitly require that env.json already contains the intended gateway address before deploying the periphery.

Resolution

Olympus: Resolved.

Remediation Findings & Resolutions

ID	Title	Category	Severity	Status
I-10	Enabled/Disable Events Omit Data Parameters	Events	Info	Resolved
I-11	Non-Canonical Chains Have No Minting Cap	Trust Assumptions	Info	Resolved
I-12	Periphery Can Use Stale Gateway	Informational	Info	Resolved

I-13 | Stale _subActionTargetKind Storage

Category	Severity	Location	Status
Best Practices	Info	src/policies/bridge/LZBridgeAndDelegateConfig.sol:297 src/policies/bridge/LZBridgeAndDelegateConfig.sol:93C63-L93C83	Resolved

Description

When a queued action is executed or cancelled, `_queuedActions[actionId].actions` is deleted, but the per queued sub-action `_subActionTargetKind[actionId][index]` mapping entries are not. These entries remain in storage permanently after the action completes, even though they are never read again.

Recommendation

Clear `_subActionTargetKind[actionId][index]` entries in the execution and cancellation paths.

Resolution

Olympus: Resolved.

I-14 | Queued Actions Remain After Policy Disable

Category	Severity	Location	Status
Documentation	Info	src/policies/bridge/LZBridgeAndDelegateConfig.sol:287	Resolved

Description

When `LZBridgeAndDelegateConfig` is disabled for security purposes, `_validateExecution` blocks execution of queued actions via `_requireEnabled()`, but the queued actions remain in storage.

If the policy is disabled and later re-enabled, any previously-queued actions (whose execution windows have not expired yet) become executable again. An operator who re-enables the policy without first cancelling all potentially dangerous queued actions could allow a malicious action to complete.

Recommendation

Document the operational procedure that all malicious queued actions must be cancelled by the emergency role before the policy is re-enabled.

Resolution

Olympus: Resolved.

I-15 | Cross-Batch Execution Order is Not Enforced

Category	Severity	Location	Status
Documentation	Info	src/policies/utils/TimelockBatchQueue.sol:126	Resolved

Description

Within a queued batch, sub-actions execute atomically in array order. Across separate queued actions, there is no enforced ordering, and execution is permissionless, so two independently-queued batches can be executed in any order once their timelocks elapse. Operators who queue dependent actions in separate batches cannot rely on them executing in queue order.

Recommendation

Document that actions with execution-order dependencies must be queued within a single batch, and that independent batches should not be assumed to execute in queue order.

Resolution

Olympus: Resolved.

I-16 | Delegate Clear Docs Mismatch

Category	Severity	Location	Status
Documentation	Info	ILZBridgeGateway.sol	Resolved

Description

The ILZBridgeGateway.setDelegate() interface documents that passing address(0) clears the LayerZero endpoint delegate.

However, LZBridgeGateway.setDelegate() calls validateSetDelegate(delegate_), and validateSetDelegate() rejects the zero address. The timelocked LZBridgeAndDelegateConfig path also calls the same validation helper before queueing setDelegate, so zero-address delegate clearing is not available through either direct or timelocked configuration.

As a result, operators or scripts following the interface documentation may attempt to clear the delegate with address(0) and see the action revert at validation or execution time. The implementation still supports replacing the delegate with another nonzero address, so this is a documentation/API consistency issue rather than a loss of delegate control.

Recommendation

Update the interface documentation to state that delegate_ must be nonzero.

Resolution

Olympus: Resolved.

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

Founded in 2022 by DeFi experts, Guardian is a leading audit firm in the DeFi smart contract space. With every audit report, Guardian 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