PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For LayerZero (Verifier Network)

26 August 2023

paladinsec.co   info@paladinsec.co

Table of Contents

Disclaimer

Paladin Blockchain Security ("Paladin") has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocation for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Paladin.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and Paladin is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Paladin or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

Paladin retains the right to re-use any and all knowledge and expertise gained during the audit process, including, but not limited to, vulnerabilities, bugs, or new attack vectors. Paladin is therefore allowed and expected to use this knowledge in subsequent audits and to inform any third party, who may or may not be our past or current clients, whose projects have similar vulnerabilities. Paladin is furthermore allowed to claim bug bounties from third-parties while doing so.

1 Overview

This report has been prepared for LayerZero's Verifier Network contracts on the Ethereum network. Paladin provides a user-centred examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

1.1 Summary

Project Name	LayerZero
URL	https://layerzero.network/
Platform	Ethereum
Language	Solidity
Preliminary Contracts	https://github.com/LayerZero-Labs/monorepo/tree/bddcada8ab86e1708ac57f960b58a9fbaad1759c/packages/layerzero-v2/evm/messagelib/contracts
Resolution #1	https://github.com/LayerZero-Labs/monorepo/tree/084abacfea540a83086713dbe8c5826e435ca47e/packages/layerzero-v2/evm/messagelib/contracts
Resolution #2	https://github.com/LayerZero-Labs/monorepo/tree/f8b2e33e5adc0743b42e89e140a1a510cd31fe5c/packages/layerzero-v2/evm/messagelib/contracts

1.2 Contracts Assessed

Name	Contract	Live Code Match
`VerifierNetwork`		
`Worker`		
`MultiSig`		

1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
🔴 High	0	-	-	-
🟠 Medium	1	1	-	-
🟡 Low	4	4	-	-
🟣 Informational	5	1	-	4
Total	**10**	**6**	**-**	**4**

Classification of Issues

Severity	Description
🔴 High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
🟠 Medium	Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
🟡 Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
🟣 Informational	Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.

1.3.1 VerifierNetwork

ID	Severity	Summary	Status
01	MEDIUM	Admin can re-enter during the execute function, potentially maliciously executing an operation twice	✓ RESOLVED
02	LOW	`_shouldCheckHash` could cause certain functions to not be checked, allowing them to be executed multiple times	✓ RESOLVED
03	INFO	Gas optimization	ACKNOWLEDGED

1.3.2 Worker

ID	Severity	Summary	Status
04	LOW	`address(0)` unnecessarily bears the `DEFAULT_ADMIN_ROLE` which has full control over this contract	✓ RESOLVED
05	INFO	Gas optimization	ACKNOWLEDGED
06	INFO	Typographical issues	✓ RESOLVED

1.3.3 MultiSig

ID	Severity	Summary	Status
07	LOW	Signatures do not incorporate a domain separator by default	✓ RESOLVED
08	LOW	Signatures are malleable as low-level `ecrecover` is used	✓ RESOLVED
09	INFO	Gas optimization	ACKNOWLEDGED
10	INFO	Typographical issues	ACKNOWLEDGED

2 Findings

2.1 VerifierNetwork

`VerifierNetwork` is used within LayerZero as a crucial component for the delivery of messages on the destination chains.

The contract has multi-signature logic to validate ECDSA signatures — if sufficient signatures are provided, it can submit a validation confirmation to the ULN (`UltraLightNode302`) and ask it to execute the message delivery which allows the message to subsequently be executed in a separate transaction. It is configured as a `MultiSig` (using the `MultiSig` dependency) which can be configured to have any number of signers and an arbitrary required quorum to execute messages.

It should be explicitly noted that when a user sets the `VerifierNetwork` as their desired verifier, they specifically rely on the security of the multi-signature set. If the multi-signature set-up is compromised, so is the application if the `VerifierNetwork` is the only verifier.

As `VerifierNetwork` serves as one of the verifiers for the recipient library, it also implements functions for the sending library to calculate the fee. This means that a verifier must be deployed at the same address on all used chains to allow for the fee related functions to be called on the sending chain.

2.1.1 Privileged Functions

- withdrawFee [ADMIN_ROLE]
- setDefaultMultiplierBps [ADMIN_ROLE]
- setWorkerFeeLib [ADMIN_ROLE]
- setPriceFeed [ADMIN_ROLE]
- setPaused [Not callable due to DEFAULT_ADMIN_ROLE not being assigned]
- setDstConfig [ADMIN_ROLE]
- grantRole [callable via execute and by ADMIN_ROLE]
- revokeRole [callable via execute and by ADMIN_ROLE]
- verifyAndDeliver [callable via execute]
- setSigner [callable via execute]
- setQuorum [callable via execute]
- quorumChangeAdmin [quorum required]
- execute [quorum required]

2.1.2 Issues & Recommendations

Issue #01	**Admin can re-enter during the execute function, potentially maliciously executing an operation twice**
Severity	● MEDIUM SEVERITY

Location	Lines 173-181
	`(bool success, bytes memory rtnData) = param.target.call(param.callData);`
	`if (success) {`
	`      if (shouldCheckHash) {`
	`      // store usedHash only on success`
	`      usedHashes[hash] = true; // prevent reentry and replay attack`
	`      }`
	`} else {`
	`      emit ExecuteFailed(i, rtnData);`
	`}`

Description	`VerifierNetwork` uses a MultiSig structure to allow for executing arbitrary transactions if sufficient signers approve them. It is of course essential that such signatures can only be used once, otherwise, if the signers wish to execute for example a swap, the executor could force the network to execute said swap over and over again. This is enforced by consuming the hash of the transaction data that is signed as soon as the transaction has been executed. However, in a specific case, this single-use is not properly enforced: As the hash is marked as used only after the external call, a malicious account with the `ADMIN_ROLE` could use this to re-enter and re-execute the same transaction.
Recommendation	Consider marking the hash as used before the external call and setting it back to false if the call failed. Interestingly, setting it back to `false` on failure still adheres to checks-effects-interactions as a failed transaction cannot affect state.
Resolution	✓ RESOLVED This has now been written in checks-effects-interactions

Issue #02	`_shouldCheckHash` could cause certain functions to not be checked, allowing them to be executed multiple times
Severity	● LOW SEVERITY
Description	The `_shouldCheckHash` function allows a specific selector to skip the hash check to save some gas. However, as the selector is only 4 bytes, some functions that should be checked could collide with one selector that is skipped.
Recommendation	Consider being extra careful on these selectors to avoid any kind of collision.
Resolution	✓ RESOLVED The client has manually checked the signatures.

Issue #03	Gas optimization
Severity	● INFORMATIONAL
Description	Within the `verifyAndDeliver` function, the `deliverable` and `deliver` functions are both called if the message is deliverable. However if we look at those 2 functions, they both call `_deliverable` which will check all the signatures twice. Consider updating the logic so that the expensive signature check is done only once.
Recommendation	Consider implementing the gas optimization mentioned above.
Resolution	● ACKNOWLEDGED

2.2 Worker

`Worker` is a utility dependency inherited by the `VerifierNetwork` contract. It stores various variables used within this contract and has other utilities which are not used within that contract. Its primary purpose is setting up Access Control roles (`ALLOWLIST`, `DENYLIST`, `ADMIN_ROLE`, `MESSAGE_LIB_ROLE`) alongside storing a couple of variables used for sending libraries like the `priceFeed` address, `workerFeeLib`, `defaultMultiplierBps`, etc. These variables are configurable by any account with the `ADMIN_ROLE`.

The contract also defines an interesting access control modifier which is used for several sending related functions: `onlyAcl`. This modifier allows for the worker contract to optionally guard a contract's function using an allowlist. However, the admin is also free to set the whitelist size to zero which would result in allowing anyone to call this function. Finally, the admin is permitted to denylist specific contracts from accessing this function, even if the allowlist size is equal to zero.

2.2.1 Privileged Functions

- `withdrawFee [ ADMIN_ROLE ]`
- `setDefaultMultiplierBps [ ADMIN_ROLE ]`
- `setWorkerFeeLib [ ADMIN_ROLE ]`
- `setPriceFeed [ ADMIN_ROLE ]`
- `setPaused [ DEFAULT_ADMIN_ROLE ]`
- `grantRole [ DEFAULT_ADMIN_ROLE ]`
- `revokeRole[ DEFAULT_ADMIN_ROLE ]`

2.2.2 Issues & Recommendations

Issue #04	**address(0) unnecessarily bears the DEFAULT_ADMIN_ROLE which has full control over this contract**
Severity	● LOW SEVERITY
Description	`VerifierNetwork` will provide `address(0)` as the default admin address to burn it. However, in our opinion, it is much more theoretically secure to actually burn this role by never assigning it instead. The zero address is actually granted the role which seems like a totally unnecessary risk exposure, even though control of the zero address is pretty much impossible as long as ECDSA is not broken.

Recommendation	Consider only granting the role in case it's for a non-zero admin:
	`if(_roleAdmin != address(0))`
	`    _grantRole(DEFAULT_ADMIN_ROLE, _roleAdmin);`

Resolution	✓ RESOLVED The recommendation has been introduced in the constructor.

Issue #05	Gas optimization
Severity	● INFORMATIONAL

Location	Line 60-62
	`if (hasRole(DENYLIST, _sender)) {`
	`            revert("Worker: not allowed");`
	`      } else if (allowlistSize == 0 || hasRole(ALLOWLIST,`
	`_sender)) {`

Description	In case the `onlyAcl` function is used on critical paths, it might make sense to combine the `ALLOWLIST` and `DENYLIST` role into a single storage slot to not waste two different reads when one would be sufficient given that they are mutually exclusive. This would likely require deviating from the AccessControl dependency and introducing an enum, which is probably not worth the code bloat overhead if this is not on a critical path.
Recommendation	Consider implementing the gas optimization mentioned above.
Resolution	● ACKNOWLEDGED

Issue #06	Typographical issues
Severity	● INFORMATIONAL

Description	Line 7
	`import "./interfaces/ILayerZeroPriceFeed.sol";`
	
	This import appears unused. Consider whether it can be removed if there are not external dependencies on this.
	
	Line 79
	`function setPaused(bool _paused) external onlyRole(DEFAULT_ADMIN_ROLE) {`
	
	This function lacks an event. It may make sense to instead simply extend Pausable which has the necessary modifiers and emits events on the internal functions. This would save some lines of code and keep the main contract simple.
	
	The constructor lacks setter events for the various variables which are set.

Recommendation	Consider fixing the typographical errors.
Resolution	✓ RESOLVED

2.3 MultiSig

`MultiSig` is an internal dependency used for the `VerifierNetwork`. It contains the business logic to verify signatures on a transaction hash that would be executed. This verification ensures that at least the configured quorum (a number smaller or equal than the total registered signers) have signed a certain transaction. It also contains the business logic to add and remove signers from the multi-signature contract and to adjust the quorum.

It should be noted that essentially all business logic is provided as internal functions, which means that the actual external functions to do these things are provided within VerifierNetwork.

2.3.1 Issues & Recommendations

Issue #07	**Signatures do not incorporate a domain separator by default**
Severity	● LOW SEVERITY
Description	The `_getEthSignedMessageHash` function simply hashes the message hash with the standard `eth` prefix. This means signatures might be identical across chains and deployments, allowing for malicious actors to replay them across these chains and deployments. This is not the case for `execute` signatures within VerifierNetwork due to the packet header having some namespacing and the target containing the VerifierNetwork address typically. However, in certain execution styles, the target may be the ULN allowing for some replayability concerns. Furthermore, functions like `quorumChangeAdmin` within the VerifierNetwork only namespace the hash by the VerifierNetwork address (due to the target being included in the hash). Such namespacing with only the contract address is however often insufficient, especially on multi-chain deployments where contracts often have the same address.
Recommendation	Consider deploying all contracts at different addresses to decrease replayability in general. Though it is typically nice UX to deterministically deploy all contracts at the same address, it could be considered more secure if all contracts are deployed on globally unique addresses. Next, consider adding a namespace `bytes32` to `_getEthSignedMessageHash`, ideally relying on EIP-712. One can for example rely on OpenZeppelin's implementation of building it here: https://github.com/OpenZeppelin/openzeppelin-contracts/blob/0a25c1940ca220686588c4af3ec526f725fe2582/contracts/utils/cryptography/EIP712.sol#L70 Gas cost is minimal as this is an immutable `bytes32` value added to the keccak operation.

Resolution	✓ RESOLVED The client has added a configurable identifier for within the `VerifierNetwork` contract. It is crucial that signers never enlist on verifiers with the same identifier. It should also be noted that there is no explicit namespacing on the function which is called within the VerifierNetwork implementation. This mean that there is no way to specify that a signature must be executed via `execute` versus `quorumChangeAdmin`. To the best of our knowledge this is not exploitable yet due to the abi encoding of the quorum admin being very restrictive, but if the MultiSig dependency is used within other contexts, such replayability might pose more severe issues. Paladin has indicated this to the client and recommended them to change this already in anticipation.

Issue #08	Signatures are malleable as low-level `ecrecover` is used
Severity	● LOW SEVERITY
Description	The codebase uses the low-level `ecrecover` precompile to validate signatures. However, this precompile lacks many validations which are typically recommended in ECDSA validation implementations. Such validations can prevent things like malleable signatures and the zero-address being marked as a valid address by accident (the latter is already checked within MultiSig however). Note that as signature malleability does not seem like a strict concern for LayerZero, we understand if this issue is acknowledged. However, in our opinion, using a library which addresses common issues with the low-level `ecrecover` is beneficial from the perspective of third-party reviewers who will trust the known implementations more. However, we would understand if the client opts against it from a gas usage perspective.
Recommendation	Consider using ECDSA from OpenZeppelin to rely on the best practices for using the ecrecover precompile instead of trying to address these yourself. It should be noted that the `tryRecover` function should be used in case reversion is something the client wishes to avoid. The client could also rely on OZ's implementation of `splitSignature` within one of the `tryRecover` functions to reduce the line count of the MultiSig a bit if desired.
Resolution	✓ RESOLVED ECDSA by OpenZeppelin is now used.

Issue #09	Gas optimizations
Severity	● INFORMATIONAL

Description	Line 36
	`require(signerSize >= quorum, "MultiSig: committee size < threshold");`
	
	Although the gas saved is minimal (and this is a non-critical path), it might make sense to write an if-else statement for the `_active` boolean and only execute this requirement when the signer is made inactive. The reason why this might be beneficial is mainly due to readability, as it better communicates that this check is only relevant when the signer size is decreased. `signerSize` could also be cached more cleanly to save further gas as its fetched twice from storage (though again since this is a non-critical path, gas optimizations matter very little).
	
	Line 54
	`for (uint i = 0; i < quorum; i++) {`
	
	Consider caching quorum to save significant gas.

Recommendation	Consider implementing the gas optimizations mentioned above.
Resolution	● ACKNOWLEDGED

Issue #10	Typographical issues
Severity	● INFORMATIONAL

Description	Line 10
	`event UpdateSigner(address _signer, bool _active);`
	The `_signer` parameter can be indexed.
	
	Line 25
	`signers[signer] = true;`
	It might be more consistent to emit an `UpdateSigner` event here.
	
	Line 29
	`quorum = _quorum;`
	It might be more consistent to emit an `UpdateQuorum` event here.
	
	A final consideration to make is that `verifySignatures` which are in-flight will revert if the quorum variable was reduced before the verify transaction confirms due to an inequality check at line 47. We are of the opinion that this is fine as its nicer to strictly enforce input size than to address this edge case but still include this simply to make the client aware of this edge-case and to ensure that all components can graciously handle it and retry with the new quorum size.

Recommendation	Consider fixing the typographical errors.
Resolution	● ACKNOWLEDGED

PALADIN

BLOCKCHAIN SECURITY