KGeN OFT
Preliminary Audit Report

Fri Aug 22 2025

contact@bitslab.xyz https://twitter.com/scalebit_

ScaleBit

KGeN OFT Preliminary Audit Report

1 Executive Summary

1.1 Project Information

Description	The KGeN OFT project is an enhanced cross-chain token solution based on LayerZero, developed by the KGeN team. It integrates multiple security features, including role-based access control, ERC2771 meta-transaction support, blacklist compliance functionality, an emergency pause mechanism, and token recovery capabilities, aiming to provide secure and reliable token services for decentralized finance
Type	Bridge
Auditors	ScaleBit
Timeline	Tue Aug 19 2025 - Thu Aug 21 2025
Languages	Solidity
Platform	EVM Chains
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	https://github.com/kgen-protocol/smartcontracts
Commits	da051bfc4654a048bf943074ef8568a8e3508da6

1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
ERC2CU3	oft-adapter-aptos-move/contracts/ERC2771Context/ERC2771ContextUpgradable.sol	da256550f3a84e64bff035749507501b928a0f02
KOFT	oft-adapter-aptos-move/contracts/KgenOFT.sol	e742c7127ab409652b41ed739e68e1886e47c56d

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	5	0	0
Informational	0	0	0
Minor	1	0	0
Medium	3	0	0
Major	0	0	0
Critical	0	0	0
Discussion	1	0	0

1.4 ScaleBit Audit Breakdown

ScaleBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow
- Number of rounding errors
- Unchecked External Call
- Unchecked CALL Return Values
- Functionality Checks
- Reentrancy
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic issues
- Gas usage
- Fallback function usage
- tx.origin authentication
- Replay attacks
- Coding style issues

1.5 Methodology

The security team adopted the **"Testing and Automated Analysis"**, **"Code Review"** and **"Formal Verification"** strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

(1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

(2) Code Review

The code scope is illustrated in section 1.2.

(3) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

2 Summary

This report has been commissioned by KGeN to identify any potential issues and vulnerabilities in the source code of the KGeN OFT smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 5 issues of varying severity, listed below.

ID	Title	Severity	Status
KOF-1	Incorrect Handling of ERC20 Transfer Return Values	Medium	Pending
KOF-2	Centralization Risk	Medium	Pending
KOF-3	Single-step Ownership Transfer Can be Dangerous	Medium	Pending
KOF-4	Lack of Pause Checks	Minor	Pending
KOF-5	The Protocol Fee-Payment Logic is Incorrect	Discussion	Pending

3 Participant Process

Here are the relevant actors with their respective abilities within the KGeN OFT Smart Contract :

DEFAULT_ADMIN_ROLE

- `updateFeeVault` : Update the fee vault address.
- `recoverERC20` : Recover ERC20 tokens accidentally sent to the contract.
- `recoverETH` : Recover native ETH accidentally sent to the contract.
- `grantRole` : Grant any role, including PAUSER_ROLE, BLACKLIST_MANAGER_ROLE, FORWARDER_MANAGER_ROLE.

PAUSER_ROLE

- `setPaused` : Pause or unpause the entire contract.
- `setCrossChainPaused` : Pause or unpause only cross-chain operations.

BLACKLIST_MANAGER_ROLE

- `setBlacklistStatus` : Add or remove an account from the blacklist.
- `batchSetBlacklistStatus` : Add or remove multiple accounts from the blacklist.

FORWARDER_MANAGER_ROLE

- `addTrustedForwarder` : Add a new trusted forwarder for meta-transactions.
- `removeTrustedForwarder` : Remove an existing trusted forwarder.
- `updateTrustedForwarder` : Replace an old trusted forwarder with a new one.
- getTrustedForwarders : Retrieve all trusted forwarders.
- `getTrustedForwardersCount` : Retrieve the number of trusted forwarders.

Users

- `transfer` : Transfer tokens to another account.
- `transferFrom` : Transfer tokens from another account using allowance.
- `send` : Send cross-chain message.

- `sendFrom` : Allow users to pay cross-chain fees in two ways when sending cross-chain message.

4 Findings

KOF-1 Incorrect Handling of ERC20 Transfer Return Values

Severity: Medium

Status: Pending

Code Location:

oft-adapter-aptos-move/contracts/KgenOFT.sol#199-203

Descriptions:

The `try/catch` statement in the code only captures executions that `revert`. It does not handle the case where the external call completes successfully but returns `false`.

```
try IERC20(token).transfer(to, amount) {
    emit TokenRecovered(token, to, amount);
} catch {
    revert TokenRecoveryFailed();
}
```

When using the transfer function some older or non-compliant tokens will return `false` instead of `revert`.

Suggestion:

Replace the current `try/catch` implementation with the `safeTransfer` function from the `SafeERC20` library.

KOF-2 Centralization Risk

Severity: Medium

Status: Pending

Code Location:

oft-adapter-aptos-move/contracts/KgenOFT.sol#142-145

Descriptions:

A comprehensive centralization risk was identified in the smart contract:

- **System Control**: The administrator can unilaterally pause all or parts of the contract's core functionality through the `setPaused()` and `setCrossChainPaused()` functions.
- **User Asset Control**: The administrator can add or remove any address from the blacklist via the `setBlacklistStatus()` function, thereby freezing or unfreezing user assets.
- **Fund Control**: The administrator can withdraw any tokens or native ETH held within the contract using `recoverERC20()` and `recoverETH()`, and change the fee collection address via `updateFeeVault()`.
- **Ultimate Privilege Control**: As the holder of `DEFAULT_ADMIN_ROLE`, the administrator can grant and revoke any role to other addresses; as the `owner`, the administrator can transfer the ultimate ownership of the contract.
- **Infrastructure Control**: The administrator can manage the list of trusted forwarders for meta-transactions through functions like `addTrustedForwarder()` and `removeTrustedForwarder()`.

Suggestion:

It is recommended that measures be taken to reduce the risk of centralization, such as implementing a multi-signature (Multi-sig) wallet mechanism, optionally combined with a Timelock for increased transparency.

KOF-3 Single-step Ownership Transfer Can be Dangerous

Severity: Medium

Status: Pending

Code Location:

oft-adapter-aptos-move/contracts/KgenOFT.sol#136

Descriptions:

The contract uses a single-step process to transfer ownership.

```
Ownable(_delegate)
```

If the new address is incorrect (e.g., mistyped, set to 0x0, or an unowned address), the contract may permanently lose its owner control, pausing, or managing permissions. Such an error is irreversible and may cause permanent damage to the contract.

Suggestion:

Adopt a two-step transfer mechanism

Refer to OpenZeppelin's Ownable2Step

KOF-4 Lack of Pause Checks

Severity: Minor

Status: Pending

Code Location:

oft-adapter-aptos-move/contracts/KgenOFT.sol#230 245 301 316 346 422

Descriptions:

When the contract is paused, only the `owner` should be able to operate specific functions.The following functions remain callable even when the contract is paused:

- `setBlacklistStatus`
- `batchSetBlacklistStatus`
- `addTrustedForwarder`
- `removeTrustedForwarder`
- `updateTrustedForwarder`
- `_credit`

To ensure that the pause mechanism effectively blocks critical operations in emergency scenarios, these functions should include `whenNotPaused` and `whenCrossChainNotPaused` checks.

Suggestion:

Add the `whenNotPaused` `whenCrossChainNotPaused` modifier to all the above functions

KOF-5 The Protocol Fee-Payment Logic is Incorrect

Severity: Discussion

Status: Pending

Code Location:

oft-adapter-aptos-move/contracts/KgenOFT.sol#407

Descriptions:

According to the documentation,the `sendFrom` function is designed to support a dual-fee mechanism, allowing for two distinct payment models for a cross-chain transaction:

- Protocol Pays (Gas Sponsoring): The user pays a service fee to the protocol in KGEN tokens via the gasFeeAmount parameter.
- User Pays: The user pays the exact LayerZero network gas fee directly by sending native tokens (e.g., ETH) with the transaction via msg.value.

The problem is that if a user wants to perform a cross-chain operation using the protocol to pay the gas fee, the user should only need to provide KGEN to the `FEE_VAULT`. However, in reality, when `sendFrom` collects the gas fee via `transfer` once, in the subsequent operations the `_send` function calls `_lzSend`, and within `_lzSend` the `_payLzToken` function is invoked. The `_payLzToken` function charges the user **again** in the form of LZToken as a gas fee, and requires the user to approve it in advance.

Another issue is that `gasFeeAmount` is not verified, and as long as the transaction executes successfully, it does not affect any subsequent operations. This means the user can pass any value. Moreover, the subsequent fee collection methods, whether `_payNative` or `_payLzToken`, always pull from `msg.sender` —i.e., the user's address—rather than being paid by the protocol.

Suggestion:

Override the `_payLzToken` function to change `msg.sender` to the protocol's designated fee-paying address, and in the `sendFrom` function, validate that the passed

`gasFeeAmount` meets the required gas fee and protocol fee before performing the `transfer` operation.

Appendix 1

Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

Appendix 2

Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.