MiloTruck

LUKSO

Security Review

September 15, 2025

Contents

1 Introduction

1.1 About MiloTruck

MiloTruck is an independent security researcher, primarily working as a Lead Security Researcher at Spearbit and Cantina. Previously, he was part of the team at Renascence Labs and a Lead Auditor at Trust Security.

For private audits or security consulting, please reach out to him on Twitter *@milotruck*.

1.2 Disclaimer

A smart contract security review **can never prove the complete absence of vulnerabilities**. Security reviews are a time, resource and expertise bound effort to find as many vulnerabilities as possible. However, they cannot guarantee the absolute security of the protocol in any way.

2 Risk Classification

Severity Level	**Impact: High**	**Impact: Medium**	**Impact: Low**
Likelihood: High	High	High	Medium
Likelihood: Medium	High	Medium	Low
Likelihood: Low	Medium	Low	Low

2.1 Impact

- High - Funds are **directly** at risk, or a **severe** disruption of the protocol's core functionality.
- Medium - Funds are **indirectly** at risk, or **some** disruption of the protocol's functionality/availability.
- Low - Funds are **not** at risk.

2.2 Likelihood

- High - Highly likely to occur.
- Medium - Might occur under specific conditions.
- Low - Unlikely to occur.

3 Executive Summary

3.1 About LUKSO

LUKSO is the digital base layer for the New Creative Economies. It provides creators and users with future-proof tools and standards to unleash their creative force in an open interoperable ecosystem.

3.2 Overview

Project Name	LUKSO
Project Type	Bridge
Language	Solidity
Repository	lukso-network/lsp-bridge-HypLSP7 lukso-network/lsp-smart-contracts ERC725Alliance/ERC725
Commit Hash	e289c61fc81397cb4ffa39330359d4539ab33124 b71b5322c6b4fff2bed5c5308fe1a901d8bb38df 2f70cb58c40c21d0fa528554ccb773784e93e4a0

3.3 Issues Found

High	0
Medium	2
Low	3
Informational	1

4 Findings

4.1 Medium Risk

4.1.1 Users can specify any arbitrary hook address to be called during cross-chain transfers

Context:

- TokenRouter.sol#L54-L58
- TokenRouter.sol#L75-L81

Description: Hyperlane's `TokenRouter` contract, which is inherited by all four token contracts (i.e. `HypLSP7`, `HypLSP7Collateral`, HypLSP8, `HypLSP8Collateral`), has two `transferRemote()` functions used to initiate a cross-chain transfer. One of the `transferRemote()` functions allows the caller to specify `_hookMetadata` and the `_hook` address to be called:

```
function transferRemote(
    uint32 _destination,
    bytes32 _recipient,
    uint256 _amountOrId,
    bytes calldata _hookMetadata,
    address _hook
) external payable virtual returns (bytes32 messageId) {
```

However, this `_hook` address specified by the user will be called as the post-dispatch hook, instead of the hook set by the contract's owner.

As a result, by calling this `transferRemote()` function, users can bypass any validation performed in the hook contract set by the owner. For example, if the `hook` address was set to `PausableCircuitBreakerHook`, users would still be able to perform cross-chain transfers even while transfers are paused.

Recommendation: In all four token contracts, override the `transferRemote()` function shown above to revert when called. All cross-chain transfers should be performed by calling the other `transferRemote()` function, which does not allow the user to specify the post-dispatch hook to be called.

4.1.2 LSP7 tokens reverting on zero-value-transfers creates DOS risk

Context:

- LSP7DigitalAsset.sol#L694-L701
- LSP7DigitalAssetInitAbstract.sol#L707-L714

Description: In `LSP7DigitalAsset` and `LSP7DigitalAssetInitAbstract`, `_spendAllowance()` was modified to revert whenever the operator's allowance (i.e. `authorizedAmount`) is `0`:

```
if (authorizedAmount == 0 || amountToSpend > authorizedAmount) {
    revert LSP7AmountExceedsAuthorizedAmount(
        tokenOwner,
        authorizedAmount,
        operator,
        amountToSpend
    );
}
```

Note that `_spendAllowance()` is used to reduce the operator's allowance whenever `transfer()` is called by an address which is not the owner.

While the `authorizedAmount == 0` check is not a vulnerability by itself, it creates integration risk for any protocols/contracts that use an LSP7 token. More specifically, since LSP7 tokens now do not allow transfers from addresses which have not granted allowance, any function which performs zero value transfers might unexpectedly revert and be at risk of DOS.

For example, consider a function which is used to deposit two tokens:

```
function deposit(uint256 amountA, uint256 amountB) external {
    // Increase deposit balance
    tokenABalance[msg.sender] += amountA;
    tokenBBalance[msg.sender] += amountB;

    // Transfer tokens in
    tokenA.transfer(msg.sender, address(this), amountA, true, "");
    tokenB.transfer(msg.sender, address(this), amountB, true, "");
}
```

A user might want to call `deposit()` to deposit only `tokenA` and not `tokenB` (i.e. `tokenB = 0`). However, in order for `tokenB.transfer()` to not revert, he must give the contract allowance for `tokenB`.

Recommendation: Consider removing the `authorizedAmount == 0` check if it is not necessary.

Alternatively, document that LSP7 token transfers revert when no allowance is given as a warning for integrators.

LUKSO: Acknowledged. Documented this issue in PR 1013 as we do not want to allow zero value transfers from non-operators as this can lead to phishing attacks.

MiloTruck: Acknowledged.

4.2 Low Risk

4.2.1 Minting tokens in the `initialize()` function of `HypLSP7/HypLSP8` introduces additional risks

Context:

- HypLSP7.sol#L66-L67
- HypLSP8.sol#L65-L67

Description: The `initialize()` function of `HypLSP7/HypLSP8` mints tokens to `msg.sender`, which is the deployer address:

```
// mints initial supply to deployer
LSP7DigitalAssetInitAbstract._mint({ to: msg.sender, amount: _totalSupply, force: true, data: "" });
```

```
for (uint256 i = 0; i < _mintAmount; i++) {
    _mint(msg.sender, bytes32(i), true, "");
}
```

This is meant for **new** LSP7/LSP8 tokens which exist as `HypERC20` on Ethereum and `HypLSP7` on LUKSO. In other words, there is no separate token contract deployed on either chain.

However, nothing ensures `initialize()` does not mint tokens in `HypLSP7/HypLSP8` contracts deployed for **existing** tokens. Should this occur, cross-chain accounting for that token will become incorrect:

- For LSP7, the total supply of `HypLSP7` will be greater than the actual amount of tokens bridged to LUKSO. This means more `HypLSP7` tokens are in circulation than the actual bridged amount, which could result in insolvency.
- For LSP8, NFTs with token IDs from `0` to `_mintAmount - 1` cannot be bridged to LUKSO as their token IDs have already been minted to the deployer.

Additionally, for `HypLSP8.initialize()`, minting LSP8 tokens in a for-loop is extremely gas intensive and could potentially run out-of-gas.

Recommendation: Consider if there is a need for LSP7/LSP8 tokens which exist only as warp routes and remove minting in both `initialize()` functions if such functionality is not needed.

Otherwise, include a warning that `_totalSupply`/`_mintAmount` in their respective LSP7/LSP8 `initialize()` functions must be `0` for existing tokens.

4.2.2 Constructors and `initialize()` functions in ERC725 are declared `payable`

Context:

- ERC725.sol#L24-L26
- ERC725Init.sol#L33-L35

Description: In all ERC725 smart contracts, constructors and `initialize()` functions are declared as `payable`, even though the contracts do not handle LYX.

However, this is a dangerous pattern as it allows the contract deployer or the caller of `initialize()` to send LYX even when the contract is not meant to receive LYX. Additionally, declaring them as `payable` does not save *that* much gas to be worth the additional risk.

Recommendation: Consider removing `payable` from the constructors and `initialize()` functions in ERC725 smart contracts.

LUKSO: Acknowledged. The constructors and `initialize()` functions are declared `payable` to allow the contracts to be funded with native assets on deployment. Added warnings in the relevant `ERC725` contracts to document this issue in commit 0062905.

MiloTruck: Acknowledged.

4.2.3 `LSP7DigitalAssetInitAbstract` wrongly inherits `LSP17Extendable`

Context: LSP7DigitalAssetInitAbstract.sol#L77-L81

Description: `LSP7DigitalAssetInitAbstract` inherits `LSP17Extendable`:

```
abstract contract LSP7DigitalAssetInitAbstract is
    ILSP7DigitalAsset,
    LSP4DigitalAssetMetadataInitAbstract,
    LSP17Extendable
{
```

However, it should inherit `LSP17ExtendableInitAbstract` instead as it is an initializable contract.

Recommendation: For `LSP7DigitalAssetInitAbstract`, use `LSP17ExtendableInitAbstract` in-place of `LSP17Extendable`.

LUKSO: Fixed in PR 1008. This issue resulted in incorrectly returning `false` when checking if the contract supports the `ERC725Y` interface using `supportsInterface(ERC725Y interface ID)`.

MiloTruck: Verified, `LSP7DigitalAssetInitAbstract` now correctly inherits `LSP17ExtendableInitAbstract`.

4.3 Informational

4.3.1 Minor improvements to code and comments

Context: See below.

Description:

1. ERC725Y.sol#L86, ERC725Y.sol#L106, HypLSP7.sol#L58 - Typo: "overriden" -> "overridden"
2. HypLSP7Collateral.sol#L38, HypLSP8Collateral.sol#L32 - Can be declared `external` instead of `public`.
3. HypLSP7.sol#L34-L35 - Typo, "modifed" should be "modified". Additionally, the natspec formatting seems to be messed up.
4. HypLSP8Collateral.sol#L37, HypLSP8Collateral.sol#L45 - Casting `wrappedToken` to `ILSP8` is redundant as `wrappedToken` is already an `ILSP8`.
5. CircuitBreakerAdapter.sol#L22-L26 - Can be simplified to:

```
- if (!hasRole(CIRCUIT_BREAKER_ROLE, msg.sender)) {
-     if (owner() != msg.sender) {
+ if (!hasRole(CIRCUIT_BREAKER_ROLE, msg.sender) && owner() != msg.sender)) {
          revert NotCircuitBreakerOrOwner();
      }
  }
```

LUKSO: All issues have been addressed as follows:

1. Fixed in commit b80c572.
2. We keep it `public` to be consistent with Hyperlane's contracts.
3. Fixed in commit b904f57.
4. Fixed in commit 043c349.
5. Fixed in commit a27ec51.

MiloTruck: Verified.