LUKSO LSPs #2 Audit Report

Dec 15, 2022

WATCHPUG

Table of Contents

Summary

This report has been prepared for LUKSO LSPs #2 Audit Report smart contract, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

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.

Overview

Project Summary

Project Name	**LUKSO LSPs #2 Audit Report**
Codebase	**https://github.com/lukso-network/lsp-smart-contracts**
Commit	**e7a07d675619f2e35b9bc92c9b43f5d06ff9acd2**
Language	**Solidity**

Audit Summary

Delivery Date	**Dec 15, 2022**
Audit Methodology	**Static Analysis, Manual Review**
Total Isssues	**23**

[WP-M1] LSP-2: `LSP2Utils.isCompactBytesArray(bytes)` Empty array (`[]`) will return `false`

Medium

Issue Description

L270 of `isCompactBytesArray(bytes)` will return `false` when `compactBytesArray.length == 0` . This means the input `compactBytesArray` is an invalid `bytes[CompactBytesArray]` .

https://github.com/lukso-network/lsp-smart-contracts/blob/e7a07d675619f2e35b9bc92c9b43f5d06ff9acd2/contracts/LSP2ERC725YJSONSchema/LSP2Utils.sol#L266-L297

```
266     /**
267      * @dev Verify the validity of the `compactBytesArray` according to LSP2
268      */
269     function isCompactBytesArray(bytes memory compactBytesArray) internal pure returns (bool) {
270         if (compactBytesArray.length == 0) return false;
271         /**
272          * Pointer will always land on these values:
273          *
274          * ↓↓
275          * 03 a00000
276          * 05 fff83a0011
277          * 20 aa0000000000000000000000000000000000000000000000000000000000cafe
278          * 12 bb000000000000000000000000000000beef
279          * 19 cc00000000000000000000000000000000000000000000deed
280          * ↑↑
281          *
282          * The pointer can only land on the length of the following bytes value.
283          */
284         uint256 pointer;
285 
286         /**
287          * Check each length byte and make sure that when you reach the last length byte.
288          * Make sure that the last length describes exactly the last bytes value and you do not get out of bounds.
```

```
             */
            while (pointer < compactBytesArray.length) {
                uint256 elementLength =
uint256(uint8(bytes1(compactBytesArray[pointer])));
                if (elementLength == 0) return false;
                pointer += elementLength + 1;
            }
            if (pointer == compactBytesArray.length) return true;
            return false;
        }
```

Impact

This can be a problem when trying to reset the value of a `bytes[CompactBytesArray]` to empty array.

Such a problem does not exist in the current LSP-6 implementation, where `bytes[CompactBytesArray]` is used.

However, given the fact that LSP-2 is designed to be a general-purpose library, we still consider this a medium severity issue and should be addressed.

Recommendation

Consdier removing L270.

[WP-M2] LSP-2: `LSP2Utils.isCompactBytesArray(bytes)` does not support zero-length elements (`[..., 0x, ...]`)

Medium

Issue Description

`bytes[CompactBytesArray]` should support zero-length elements (`0x`).

However, the current implementation of `LSP2Utils.isCompactBytesArray(bytes)` will return `false` whenever there is a zero-length element.

PoC

When using `isCompactBytesArray()` to verify `0x 00 03 222222` (`[0x, 0x222222]` encoded in `bytes[CompactBytesArray]`)

L292 will return `false` .

https://github.com/lukso-network/lsp-smart-contracts/blob/e7a07d675619f2e35b9bc92c9b43f5d06ff9acd2/contracts/LSP2ERC725YJSONSchema/LSP2Utils.sol#L266-L297

```
266      /**
267       * @dev Verify the validity of the `compactBytesArray` according to LSP2
268       */
269      function isCompactBytesArray(bytes memory compactBytesArray) internal pure
     returns (bool) {
270          if (compactBytesArray.length == 0) return false;
271          /**
272           * Pointer will always land on these values:
273           *
274           * ↓↓
275           * 03 a00000
276           * 05 fff83a0011
277           * 20 aa0000000000000000000000000000000000000000000000000000000000cafe
278           * 12 bb000000000000000000000000000000beef
279           * 19 cc00000000000000000000000000000000000000000000deed
280           * ↑↑
281           *
282             * The pointer can only land on the length of the following bytes value.
283             */
284            uint256 pointer;
285
286            /**
287             * Check each length byte and make sure that when you reach the last length byte.
288             * Make sure that the last length describes exactly the last bytes value and you do not get out of bounds.
289             */
290            while (pointer < compactBytesArray.length) {
291                uint256 elementLength = uint256(uint8(bytes1(compactBytesArray[pointer])));
292                if (elementLength == 0) return false;
293                pointer += elementLength + 1;
294            }
295            if (pointer == compactBytesArray.length) return true;
296            return false;
297        }
```

Impact

Certain use cases will be disabled, such as:

- Using the length of a `CompactBytesArray` to represent the number of available seats, with empty elements for available seats and elements with a value representing an order ID.
- Using empty elements as placeholders to maintain the index of pre-existing elements.

Recommendation

Consider removing L292.

[WP-D3] LSP-16: Consider adding a note about the requirements of `initializeCallData`

Issue Description

When `initializeCallData` includes non-crosschain parameters, the deployed contract will not be recreated at the same address on another network with the same calldata, thus defeating the purpose of `LSP16UniversalFactory`.

Therefore, `initializeCallData` must not include any network-specific parameters, such as a local non-crosschain token contract address.

[WP-I4] LSP-16: Empty calldata (`fallback` , `receive`) for initialize is not differeciated with no initialize call

Informational

Issue Description

https://github.com/lukso-network/lsp-smart-contracts/blob/b3169b44a5df0aca6f001f762df10a127142cda9/contracts/LSP16UniversalFactory/LSP16UniversalFactory.sol#L191-L202

```
function _generateSalt(bytes memory initializeCallData, bytes32 providedSalt)
    internal
    pure
    returns (bytes32)
{
    bool initializable = initializeCallData.length != 0;
    if (initializable) {
        return keccak256(abi.encodePacked(initializable, initializeCallData,
providedSalt));
    } else {
        return keccak256(abi.encodePacked(initializable, providedSalt));
    }
}
```

When the length of `initializeCallData` is `0` , it will not be included in the salt.

As a result, a special Clone with no `initializeCallData` but an empty call (`contractCreated.call()`) is indistinguishable to a `Create2` without that empty initialize call.

Although it is highly unlikely that a proxy contract would be designed to be initialized with an empty call, it is technically possible and permissible.

Recommendation

We believe it is more consistent and unbiased to allow empty initialize calls, therefore, we recommend you to make the following changes:

1. `_generateSalt()` should differentiate between no initialize call and an initialize call with no `initializeCallData` ;
2. `deployCreate2Proxy()` should be split into two functions: `deployCreate2Proxy()` and `deployCreate2ProxyInit()` ;
3. Line 128, `if (initializeCalldata.length == 0) revert InitializeCalldataRequired();` , should be removed to allow empty initialize calls.

[WP-I5] LSP-8: `LSP8CompatibleERC721` is not compatible with `ERC721`

Informational

Issue Description

This is an extension of [WP-I17] from our previous report.

- `safeTransferFrom()` does not call `receiver.onERC721Received(address,address,uint256,bytes)` which is required by EIP-721's spec, and does not require `receiver.onERC721Received(...) == bytes4(keccak256("onERC721Received(address,address,uint256,byt` ;
- `transferFrom()` also calls `_notifyTokenSender()` , `_notifyTokenReceiver()` ;
- `safeTransferFrom()` also calls `_notifyTokenSender()` .

Recommendation

1. Consider improving the compatibility of `LSP8CompatibleERC721` to ERC721.
2. Or, consider documenting the differences between `LSP8CompatibleERC721` and ERC721.

[WP-L6] LSP-14: `_pendingOwner` should be deleted when `_renounceOwnership` is called for the first time to initialize it

Low

Issue Description

Otherwise, two bad scenarios might occur:

PoC #1:

1. Current owner Bob first calls `transferOwnership()` to set Alice as the `pendingOwner` .
2. Bob changes his mind and calls `_renounceOwnership()` to initiate giving up ownership.
3. Before Bob calls `_renounceOwnership()` for the second time to finalize the renounce, Alice calls `acceptOwnership()` , which is not in line with Bob's expectation.

PoC #2:

Building on PoC 1, after Alice becomes the owner and wants to renounce the ownership, she calls `_renounceOwnership()` .

If the call happens to fall between `confirmationPeriodStart` and `confirmationPeriodEnd` triggered by Bob, the expectation is to initiate the renouncement. However, the actual result is that the renouncement is done immediately and the owner will become `address(0)` right after the call.

https://github.com/lukso-network/lsp-smart-contracts/blob/848026307f597f42a8a769b3a4b4152a0907e2ed/contracts/LSP14Ownable2Step/LSP14Ownable2Step.sol#L141-L162

```
    function _renounceOwnership() internal virtual {
        uint256 currentBlock = block.number;
        uint256 confirmationPeriodStart = _renounceOwnershipStartedAt +
            _RENOUNCE_OWNERSHIP_CONFIRMATION_DELAY;
        uint256 confirmationPeriodEnd = confirmationPeriodStart +
            _RENOUNCE_OWNERSHIP_CONFIRMATION_PERIOD;

        if (currentBlock > confirmationPeriodEnd) {
            _renounceOwnershipStartedAt = currentBlock;
            emit RenounceOwnershipInitiated();
            return;
        }

        if (currentBlock < confirmationPeriodStart) {
            revert NotInRenounceOwnershipInterval(confirmationPeriodStart,
confirmationPeriodEnd);
        }

        _setOwner(address(0));
        delete _renounceOwnershipStartedAt;
        delete _pendingOwner;
        emit OwnershipRenounced();
    }
```

Recommendation

We've come up with 2 possible resolutions for this:

1. Consider introducing a lock to prevent transfer of ownership during the renouncement of ownership and vice-versa:

```
function _renounceOwnership() internal virtual {
    require(_pendingOwner == address(0), "_renounceOwnership is not allowed
now.");
    uint256 currentBlock = block.number;
    uint256 confirmationPeriodStart = _renounceOwnershipStartedAt +
        _RENOUNCE_OWNERSHIP_CONFIRMATION_DELAY;
    uint256 confirmationPeriodEnd = confirmationPeriodStart +
        _RENOUNCE_OWNERSHIP_CONFIRMATION_PERIOD;

    if (currentBlock > confirmationPeriodEnd) {
        _renounceOwnershipStartedAt = currentBlock;
        emit RenounceOwnershipInitiated();
        return;
    }

    if (currentBlock < confirmationPeriodStart) {
        revert NotInRenounceOwnershipInterval(confirmationPeriodStart,
confirmationPeriodEnd);
    }
```

```

        _setOwner(address(0));
        delete _renounceOwnershipStartedAt;
        emit OwnershipRenounced();
    }
```

```
    function _transferOwnership(address newOwner) internal virtual {
        require(_renounceOwnershipStartedAt == 0, "_transferOwnership is not allowed
now.");
        if (newOwner == address(this)) revert CannotTransferOwnershipToSelf();

        _pendingOwner = newOwner;
        address currentOwner = owner();
        emit OwnershipTransferStarted(currentOwner, newOwner);

        _notifyUniversalReceiver(newOwner, _TYPEID_LSP14_OwnershipTransferStarted,
"");
        require(
            currentOwner == owner(),
            "LSP14: newOwner MUST accept ownership in a separate transaction"
        );
    }
```

1. The initialization of `transferOwnership` and `renounceOwnership` will cancel each other:

```
    function _renounceOwnership() internal virtual {
        uint256 currentBlock = block.number;
        uint256 confirmationPeriodStart = _renounceOwnershipStartedAt +
            _RENOUNCE_OWNERSHIP_CONFIRMATION_DELAY;
        uint256 confirmationPeriodEnd = confirmationPeriodStart +
            _RENOUNCE_OWNERSHIP_CONFIRMATION_PERIOD;

        if (currentBlock > confirmationPeriodEnd) {
            _renounceOwnershipStartedAt = currentBlock;
            // cancel _transferOwnership if any
            delete _pendingOwner;
            emit RenounceOwnershipInitiated();
            return;
        }

        if (currentBlock < confirmationPeriodStart) {
            revert NotInRenounceOwnershipInterval(confirmationPeriodStart,
    confirmationPeriodEnd);
        }

        _setOwner(address(0));
        delete _renounceOwnershipStartedAt;
        emit OwnershipRenounced();
    }
```

```
function _transferOwnership(address newOwner) internal virtual {
    if (newOwner == address(this)) revert CannotTransferOwnershipToSelf();

    _pendingOwner = newOwner;
    // cancel _renounceOwnership if any
    delete _renounceOwnershipStartedAt;
    address currentOwner = owner();
    emit OwnershipTransferStarted(currentOwner, newOwner);

    _notifyUniversalReceiver(newOwner, _TYPEID_LSP14_OwnershipTransferStarted,
"");
    require(
        currentOwner == owner(),
        "LSP14: newOwner MUST accept ownership in a separate transaction"
    );
}
```

[WP-D7] LSP-2: Consider adding a note regarding the max element length in `bytes[CompactBytesArray]`

Issue Description

The maximum length of each element is 255, because `uint8` is used to store the length of each element and the maximum value of `uint8` is 255.

See:
https://github.com/lukso-network/LIPs/blob/56a264e9832b65d856b9650075248b4e32f43912/LSPs/LSP-2-ERC725YJSONSchema.md#bytescompactbytesarray

This should be explicitly documented to avoid misapplication.

[WP-I8] LSP-1: `LSP1UniversalReceiverDelegateUP` can be used for spamming

Informational

Issue Description

Due to the permissionless and open design of `LSP1UniversalReceiverDelegateUP`, there is a potential for malicious or spam assets to be transferred to the target UP, thus cluttering the list of received assets.

For instance:

- Malicious NFTs with Trojan viruses embedded in the images;
- Scam tokens.

Recommendation

1. Adding a filter of blocklist on the frontend level to block out malicious tokens/NFTs;
2. Providing a helper dapp to quickly remove a set of undesired tokens/NFTs.

[WP-D9] LSP-2: Missing detailed documentation regarding the ecoding of `bytesN[CompactBytesArray]`

Issue Description

Based on the context, the encoding of `bytesN[CompactBytesArray]` must be different from `bytes[CompactBytesArray]` and `bytesN[]` .

However, the detailed documentation on the encoding of `bytesN[CompactBytesArray]` is missing.

[WP-D10] LSP-9: The obscure `UPER_SETDATA` permission granted to `universalReceiverDelegate` should be explicitly documented

Issue Description

The `universalReceiverDelegate` of LSP9Vault will be granted a temporary `_reentrantDelegate` role with unlimited `setData()` permission during the incoming `universalReceiver()` call.

If an LSP-6 is the owner of the LSP9Vault, then such a permission will be equivalent to the `SUPER_SETDATA` permission.

Plus, `universalReceiver()` is a public function that can be called by anyone at any time.

This allows a malicious or compromised `universalReceiverDelegate` to do many things that are considered dangerous, eg:

1. Adding or updating a `extension` via LSP-17;
2. Changing the permission via LSP-6 (if the KM is the owner).

https://github.com/lukso-network/lsp-smart-contracts/blob/28c00a0abafcb682ba3a3f7cdc6eab9b09dfb713/contracts/LSP9Vault/LSP9VaultCore.sol#L229-L280

```
function universalReceiver(bytes32 typeId, bytes calldata receivedData)
    public
    payable
    virtual
    returns (bytes memory returnedValues)
{
    if (msg.value != 0) emit ValueReceived(msg.sender, msg.value);
    bytes memory lsp1DelegateValue =
_getData(_LSP1_UNIVERSAL_RECEIVER_DELEGATE_KEY);
    bytes memory resultDefaultDelegate;

    if (lsp1DelegateValue.length >= 20) {
        address universalReceiverDelegate = address(bytes20(lsp1DelegateValue));

        if (universalReceiverDelegate.supportsERC165Interface(_INTERFACEID_LSP1))
{
            _reentrantDelegate = universalReceiverDelegate;
            resultDefaultDelegate = universalReceiverDelegate
                .callUniversalReceiverWithCallerInfos(
                    typeId,
                    receivedData,
                    msg.sender,
                    msg.value
                );
        }
    }

@@ 254,260 @@

    if (lsp1TypeIdDelegateValue.length >= 20) {
        address universalReceiverDelegate =
address(bytes20(lsp1TypeIdDelegateValue));

        if (universalReceiverDelegate.supportsERC165Interface(_INTERFACEID_LSP1))
{
            _reentrantDelegate = universalReceiverDelegate;
            resultTypeIdDelegate = universalReceiverDelegate
                .callUniversalReceiverWithCallerInfos(
                    typeId,
                    receivedData,
                    msg.sender,
                    msg.value
                );
        }
    }

    delete _reentrantDelegate;
    returnedValues = abi.encode(resultDefaultDelegate, resultTypeIdDelegate);
    emit UniversalReceiver(msg.sender, msg.value, typeId, receivedData,
returnedValues);
}
```

https://github.com/lukso-network/lsp-smart-contracts/blob/
28c00a0abafcb682ba3a3f7cdc6eab9b09dfb713/contracts/LSP9Vault/LSP9VaultCore.sol#
L192-L194

```
function setData(bytes32 dataKey, bytes memory dataValue) public virtual override
onlyAllowed {
    _setData(dataKey, dataValue);
}
```

https://github.com/lukso-network/lsp-smart-contracts/blob/28c00a0abafcb682ba3a3f7cdc6eab9b09dfb713/contracts/LSP9Vault/LSP9VaultCore.sol#L204-L217

```
function setData(bytes32[] memory dataKeys, bytes[] memory dataValues)
    public
    virtual
    override
    onlyAllowed
{
    if (dataKeys.length != dataValues.length) {
        revert ERC725Y_DataKeysValuesLengthMismatch(dataKeys.length,
dataValues.length);
    }

    for (uint256 i = 0; i < dataKeys.length; i = GasLib.uncheckedIncrement(i)) {
        _setData(dataKeys[i], dataValues[i]);
    }
}
```

https://github.com/lukso-network/lsp-smart-contracts/blob/28c00a0abafcb682ba3a3f7cdc6eab9b09dfb713/contracts/LSP9Vault/LSP9VaultCore.sol#L64-L75

```
    /**
     * @dev Modifier restricting the call to the owner of the contract and the
UniversalReceiverDelegate
     */
    modifier onlyAllowed() {
        if (msg.sender != owner()) {
            require(
                msg.sender == _reentrantDelegate,
                "Only Owner or reentered Universal Receiver Delegate allowed"
            );
            }
            _;
        }
```

Recommendation

Considering the high impact of this implied `SUPER_SETDATA` permission to `universalReceiverDelegate`, it should either be restricted in some way or explicitly documented in the documentation.

[WP-L11] LSP-6: `SuperTransferValue` permission can be used to initiate calls

Low

Issue Description

`SuperTransferValue` permission should differentiate between:

A, transfers of native tokens to EOA, or contracts with no code execution in the `recieve()` / `fallback()` function; B, contracts that do have code execution in the `recieve()` / `fallback()` function.

https://github.com/lukso-network/lsp-smart-contracts/blob/b3169b44a5df0aca6f001f762df10a127142cda9/contracts/LSP6KeyManager/LSP6KeyManagerCore.sol#L828-L830

```
// Skip if caller has SUPER permission for value transfers
if (hasSuperTransferValue && !isCallDataPresent && value != 0) return;
```

Recommendation

We believe that when the target is a smart contract (`.code.length > 0`), it should require the `CALL` permission, or even the `SUPERCALL` permission.

[WP-L12] LSP-6: `_verifyCanExecute` may unexpectedly allow function calls

Low

Issue Description

`isCallDataPresent` and `containsFunctionCall` are determined based on the length of the payload.

One can bypass this by using `receive()` / `fallback()` to execute code on the target contract.

https://github.com/lukso-network/lsp-smart-contracts/blob/b3169b44a5df0aca6f001f762df10a127142cda9/contracts/LSP6KeyManager/LSP6KeyManagerCore.sol#L842-L845

```
bool containsFunctionCall = payload.length >= 168;
bytes4 selector;
if (containsFunctionCall) selector = bytes4(payload[164:168]);
```

https://github.com/lukso-network/lsp-smart-contracts/blob/b3169b44a5df0aca6f001f762df10a127142cda9/contracts/LSP6KeyManager/LSP6KeyManagerCore.sol#L805

```
bool isCallDataPresent = payload.length > 164;
```

[WP-L13] LSP-6: `AddressPermissions:AllowedCalls:<address>` should use `bytes28[]` or `bytes28[CompactBytesArray]` rather than `bytes[CompactBytesArray]`

Low

Issue Description

Per the docs and the implementation:

> The compact bytes array MUST be constructed in this format according to [LSP2-ERC725YJSONSchema]:

> <1c> <bytes4 allowedInterfaceId> <bytes20 allowedAddress> <bytes4 allowedFunction> 1c: 1c in decimals is 28, which is the sum of bytes length of the elements stored in the array.

Thus, we believe `bytes28[]` or `bytes28[CompactBytesArray]` would be more suitable for this.

https://github.com/lukso-network/lsp-smart-contracts/blob/e7a07d675619f2e35b9bc92c9b43f5d06ff9acd2/contracts/LSP6KeyManager/LSP6KeyManagerCore.sol#L838-L877

```
function _verifyAllowedCall(address from, bytes calldata payload) internal view {
    // CHECK for ALLOWED CALLS
    address to = address(bytes20(payload[48:68]));

    bool containsFunctionCall = payload.length >= 168;
    bytes4 selector;
    if (containsFunctionCall) selector = bytes4(payload[164:168]);

    bytes memory allowedCalls = ERC725Y(target).getAllowedCallsFor(from);
    uint256 allowedCallsLength = allowedCalls.length;

    if (allowedCallsLength == 0 || !LSP2Utils.isCompactBytesArray(allowedCalls)) {
        revert NoCallsAllowed(from);
    }

    bool isAllowedStandard;
    bool isAllowedAddress;
    bool isAllowedFunction;

    for (uint256 ii = 0; ii < allowedCallsLength; ii += 29) {

        bytes memory chunk = BytesLib.slice(allowedCalls, ii + 1, 28);

        if (bytes28(chunk) == 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffff) {
            revert InvalidWhitelistedCall(from);
        }

        bytes4 allowedStandard = bytes4(chunk);
        address allowedAddress = address(bytes20(bytes28(chunk) << 32));
        bytes4 allowedFunction = bytes4(bytes28(chunk) << 192);

        isAllowedStandard = allowedStandard == 0xffffffff || to.supportsERC165Interface(allowedStandard);
        isAllowedAddress = allowedAddress == 0xFFfFfFffFFfffFFfFFfFFFFFffFFFffffFfFFFfF || to == allowedAddress;
        isAllowedFunction = allowedFunction == 0xffffffff || containsFunctionCall && (selector == allowedFunction);

        if (isAllowedStandard && isAllowedAddress && isAllowedFunction) return;
    }

    revert NotAllowedCall(from, to, selector);
}
```

Recommendation

Consider using `bytes28[]` or `bytes28[CompactBytesArray]` instead.

[WP-I14] LSP-17: The abstract contract `LSP17Extension` should include access control to avoid misapplication

Informational

Issue Description

The concrete contracts that inherit `LSP17Extension` will most certainly have some privileges on the LSP-0.

Thus, it must include some sort of access control to avoid unpermissioned calls.

https://github.com/lukso-network/LIPs/blob/56a264e9832b65d856b9650075248b4e32f43912/LSPs/LSP-17-ContractExtension.md#security-considerations

https://github.com/lukso-network/lsp-smart-contracts/blob/e7a07d675619f2e35b9bc92c9b43f5d06ff9acd2/contracts/LSP17ContractExtension/LSP17Extension.sol#L14-L45

```
abstract contract LSP17Extension is ERC165 {
    // solhint-disable

    /**
     * @dev See {IERC165-supportsInterface}.
     */
    function supportsInterface(bytes4 interfaceId) public view virtual override returns (bool) {
        return interfaceId == _INTERFACEID_LSP17_EXTENSION ||
super.supportsInterface(interfaceId);
    }

    /**
     * @dev Returns the original msg.data passed to the extendable contract
     * without the appended msg.sender and msg.value
     */
    function _extendableMsgData() internal view virtual returns (bytes calldata) {
        return msg.data[:msg.data.length - 52];
    }

    /**
```

```
     * @dev Returns the original msg.sender calling the extendable contract
     */
    function _extendableMsgSender() internal view virtual returns (address) {
        return address(bytes20(msg.data[msg.data.length - 52:msg.data.length -
32]));
    }

    /**
     * @dev Returns the original msg.value sent to the extendable contract
     */
    function _extendableMsgValue() internal view virtual returns (uint256) {
        return uint256(bytes32(msg.data[msg.data.length - 32:]));
    }
}
```

Recommendation

The abstract contract should provide a framework for access control:

```
/**
 * @title Implementation of the extension logic according to
LSP17ContractExtension
 * @dev To be inherited to provide context of the msg variable related to the
extendable contract
 */
abstract contract LSP17Extension is ERC165 {

    /// @custom:oz-upgrades-unsafe-allow state-variable-immutable
    address private immutable _trustedCaller;

    /// @custom:oz-upgrades-unsafe-allow constructor
    constructor(address trustedCaller) {
        _trustedCaller = trustedCaller;
    }

    function isTrustedCaller(address caller) public view virtual returns (bool) {
        return caller == _trustedCaller;
    }

    /**
     * @dev See {IERC165-supportsInterface}.
     */
    function supportsInterface(bytes4 interfaceId) public view virtual override
returns (bool) {
        return interfaceId == _INTERFACEID_LSP17_EXTENSION ||
super.supportsInterface(interfaceId);
    }

    /**
     * @dev Returns the original msg.data passed to the extendable contract
     * without the appended msg.sender and msg.value
     */
    function _extendableMsgData() internal view virtual returns (bytes calldata) {
        if (isTrustedCaller(msg.sender)) {
            return msg.data[:msg.data.length - 52];
        } else {
            return msg.data;
        }
    }

    /**
     * @dev Returns the original msg.sender calling the extendable contract
     */
    function _extendableMsgSender() internal view virtual returns (address) {
        if (isTrustedCaller(msg.sender)) {
            return address(bytes20(msg.data[msg.data.length - 52:msg.data.length -
32]));
        } else {
            return msg.sender;
        }
    }

    /**
     * @dev Returns the original msg.value sent to the extendable contract
     */
    function _extendableMsgValue() internal view virtual returns (uint256) {
        if (isTrustedCaller(msg.sender)) {
            return uint256(bytes32(msg.data[msg.data.length - 32:]));
        } else {
            return msg.value;
        }
    }
}
```

[WP-I15] LSP-17: Unconventional behavior: when called with an unsupported `msg.sig` , `fallback()` does not revert

Informational

Issue Description

A regular smart contract will revert when called with an unsupported `msg.sig` .

Therefore, for the caller, a common way to confirm a successful call is: `.code.length > 0` && the contract call did not revert (such as OpenZeppelin's implementation).

Lukso breaks this assumption (no revert when encountering an unsupported `msg.sig`), which leads to misjudgment of the above method.

For example, OpenZeppelin's does not revert as expected.

Similar to LSP-17, the Diamond proxy states explicitly in the spec that execution should revert when an unsupported function call is encountered.

https://github.com/lukso-network/lsp-smart-contracts/blob/28c00a0abafcb682ba3a3f7cdc6eab9b09dfb713/contracts/LSP9Vault/LSP9VaultCore.sol#L121-L124

```
    fallback() external payable virtual {
        if (msg.value != 0) emit ValueReceived(msg.sender, msg.value);
        _fallbackLSP17Extendable();
    }
```

https://github.com/lukso-network/lsp-smart-contracts/blob/28c00a0abafcb682ba3a3f7cdc6eab9b09dfb713/contracts/LSP0ERC725Account/LSP0ERC725AccountCore.sol#L109-L112

```
    fallback() external payable virtual {
        if (msg.value != 0) emit ValueReceived(msg.sender, msg.value);
        _fallbackLSP17Extendable();
    }
```

https://github.com/lukso-network/lsp-smart-contracts/blob/28c00a0abafcb682ba3a3f7cdc6eab9b09dfb713/contracts/LSP17ContractExtension/LSP17Extendable.sol#L58-L94

```
    function _fallbackLSP17Extendable() internal virtual {
        if (msg.data.length < 4) return;
        // If there is a function selector
        address extension = _getExtension(msg.sig);

        // if no extension was found, return
        if (extension == address(0)) return;

@@ 66,93 @@
    }
```

Recommendation

Consider reverting when in the case of an unsupported `msg.sig` .

Here is an illustrative example of how a diamond's fallback function might be implemented:

```
// Find facet for function that is called and execute the
// function if a facet is found and return any value.
fallback() external payable {
  // get facet from function selector
  address facet = selectorTofacet[msg.sig];
  require(facet != address(0));
  // Execute external function from facet using delegatecall and return any value.
  assembly {
    // copy function selector and any arguments
    calldatacopy(0, 0, calldatasize())
    // execute function call using the facet
    let result := delegatecall(gas(), facet, 0, calldatasize(), 0, 0)
    // get any return value
    returndatacopy(0, 0, returndatasize())
    // return any return value or error back to the caller
    switch result
      case 0 {revert(0, returndatasize())}
      default {return (0, returndatasize())}
  }
}
```

[WP-D16] LSP-17: Consider explicitly document that `_fallbackLSP17Extendable()` should be called at the end of the `fallback()` function

Issue Description

https://github.com/lukso-network/lsp-smart-contracts/blob/28c00a0abafcb682ba3a3f7cdc6eab9b09dfb713/contracts/LSP17ContractExtension/LSP17Extendable.sol#L58-L94

```solidity
function _fallbackLSP17Extendable() internal virtual {
    if (msg.data.length < 4) return;
    // If there is a function selector
    address extension = _getExtension(msg.sig);

    // if no extension was found, return
    if (extension == address(0)) return;

    // solhint-disable no-inline-assembly
    // if the extension was found, call the extension with the msg.data
    // appended with bytes20(address) and bytes32(msg.value)
    assembly {
        calldatacopy(0, 0, calldatasize())

        // The msg.sender address is shifted to the left by 12 bytes to remove the padding
        // Then the address without padding is stored right after the calldata
        mstore(calldatasize(), shl(96, caller()))

        // The msg.value is stored right after the calldata + msg.sender
        mstore(add(calldatasize(), 20), callvalue())

        // Add 52 bytes for the msg.sender and msg.value appended at the end of the calldata
        let success := call(gas(), extension, 0, 0, add(calldatasize(), 52), 0, 0)

        // Copy the returned data
        returndatacopy(0, 0, returndatasize())

        switch success
```

```
            // call returns 0 on failed calls
            case 0 {
                revert(0, returndatasize())
            }
            default {
                return(0, returndatasize())
            }
        }
    }
```

As `_fallbackLSP17Extendable()` uses assembly `return()` / `revert()` to terminate the call, it cannot be called before other codes in `fallback()` .

Otherwise, the codes after `_fallbackLSP17Extendable()` may never be reached.

[WP-I17] LSP-14: `renounceOwnership()` is considered unnecessary and error-prone for LSP-0 and LSP-9

Informational

Issue Description

`renounceOwnership()` is a function provided by `LSP14Ownable2Step` and `OwnableUnset` .

It can be useful for a general smart contract with a few `onlyOwner` functions that are only used during the initializing period and can be abandoned afterwards to avoid centralization risks.

However, both LSP-0 and LSP-9 rely on the owner for their core features.

We believe there is no use case where `renounceOwnership()` can be helpful.

Furthermore, once `renounceOwnership()` is called, the caller will immediately lose control over the LSP0/LSP9 and the error is irreversible.

https://github.com/lukso-network/lsp-smart-contracts/blob/e7a07d675619f2e35b9bc92c9b43f5d06ff9acd2/contracts/LSP9Vault/LSP9VaultCore.sol#L296-L306

```
    /**
     * @dev Renounce ownership of the contract in a 2-step process
     */
    function renounceOwnership()
        public
        virtual
        override(LSP14Ownable2Step, OwnableUnset)
        onlyOwner
    {
        LSP14Ownable2Step._renounceOwnership();
    }
```

Recommendation

Consider overriding and reverting in `renounceOwnership()` .

[WP-D18] LSP-14: Event name in the implementation does not match the documentation

Issue Description

In the docs

```
event RenounceOwnershipStarted();
```

> MUST be emitted when the process of renouncing ownership of the contract is initiated.

The same event is called `RenounceOwnershipInitiated` in the implementation:

https://github.com/lukso-network/lsp-smart-contracts/blob/e7a07d675619f2e35b9bc92c9b43f5d06ff9acd2/contracts/LSP14Ownable2Step/LSP14Ownable2Step.sol#L35-L38

```
/**
* @dev emitted whenever the `renounceOwnership(..)` 2-step process is started
*/
event RenounceOwnershipInitiated();
```

https://github.com/lukso-network/lsp-smart-contracts/blob/28c00a0abafcb682ba3a3f7cdc6eab9b09dfb713/contracts/LSP14Ownable2Step/LSP14Ownable2Step.sol#L141-L162

```
function _renounceOwnership() internal virtual {
    uint256 currentBlock = block.number;
    uint256 confirmationPeriodStart = _renounceOwnershipStartedAt +
        _RENOUNCE_OWNERSHIP_CONFIRMATION_DELAY;
    uint256 confirmationPeriodEnd = confirmationPeriodStart +
        _RENOUNCE_OWNERSHIP_CONFIRMATION_PERIOD;

    if (currentBlock > confirmationPeriodEnd) {
        _renounceOwnershipStartedAt = currentBlock;
        emit RenounceOwnershipInitiated();
```

```
            return;
        }

        if (currentBlock < confirmationPeriodStart) {
            revert NotInRenounceOwnershipInterval(confirmationPeriodStart,
    confirmationPeriodEnd);
        }

        _setOwner(address(0));
        delete _renounceOwnershipStartedAt;
        delete _pendingOwner;
        emit OwnershipRenounced();
    }
```

Recommendation

Based on the context, we believe the name `RenounceOwnershipStarted()` in the docs is incorrect and should be changed to `RenounceOwnershipInitiated` .

[WP-I19] LSP-14: Using number of block for CONFIRMATION_DELAY will result in different length of time in different networks

Informational

Issue Description

Because different chains have different block time.

Also, it is not convenient to query CONFIRMATION_DELAY and CONFIRMATION_PERIOD with private visibility.

https://github.com/lukso-network/lsp-smart-contracts/blob/28c00a0abafcb682ba3a3f7cdc6eab9b09dfb713/contracts/LSP14Ownable2Step/LSP14Ownable2Step.sol#L45-L54

```
    /**
     * @dev The number of block that MUST pass before one is able to
     *  confirm renouncing ownership
     */
    uint256 private constant _RENOUNCE_OWNERSHIP_CONFIRMATION_DELAY = 100;

    /**
     * @dev The number of blocks during which one can renounce ownership
     */
    uint256 private constant _RENOUNCE_OWNERSHIP_CONFIRMATION_PERIOD = 100;
```

https://github.com/lukso-network/lsp-smart-contracts/blob/28c00a0abafcb682ba3a3f7cdc6eab9b09dfb713/contracts/LSP14Ownable2Step/LSP14Ownable2Step.sol#L141-L162

```
    function _renounceOwnership() internal virtual {
        uint256 currentBlock = block.number;
        uint256 confirmationPeriodStart = _renounceOwnershipStartedAt +
            _RENOUNCE_OWNERSHIP_CONFIRMATION_DELAY;
        uint256 confirmationPeriodEnd = confirmationPeriodStart +
            _RENOUNCE_OWNERSHIP_CONFIRMATION_PERIOD;

        if (currentBlock > confirmationPeriodEnd) {
            _renounceOwnershipStartedAt = currentBlock;
            emit RenounceOwnershipInitiated();
            return;
        }

        if (currentBlock < confirmationPeriodStart) {
            revert NotInRenounceOwnershipInterval(confirmationPeriodStart,
confirmationPeriodEnd);
        }

        _setOwner(address(0));
        delete _renounceOwnershipStartedAt;
        delete _pendingOwner;
        emit OwnershipRenounced();
    }
```

Recommendation

1. Consider changing to public visibility

2. ⊠⊠⊠⊠⊠⊠⊠⊠⊠⊠

⊠⊠ ⊠⊠ ⊠⊠⊠ number of block ⊠⊠⊠ step duration

or

⊠⊠ immutable ⊠⊠⊠⊠⊠⊠⊠⊠

[WP-G20] LSP-14: Unnecessary SLOAD

Gas

Issue Description

The SLOAD of `_pendingOwner` at L122 is unnecessary, as it must be `msg.sender` .

https://github.com/lukso-network/lsp-smart-contracts/blob/b3169b44a5df0aca6f001f762df10a127142cda9/contracts/LSP14Ownable2Step/LSP14Ownable2Step.sol#L118-L135

```
118  function _acceptOwnership() internal virtual {
119      require(msg.sender == pendingOwner(), "LSP14: caller is not the
     pendingOwner");
120
121      address previousOwner = owner();
122      _setOwner(_pendingOwner);
123      delete _pendingOwner;
124
125      _notifyUniversalReceiver(
126          previousOwner,
127          _TYPEID_LSP14_OwnershipTransferred_SenderNotification,
128          ""
129      );
130      _notifyUniversalReceiver(
131          msg.sender,
132          _TYPEID_LSP14_OwnershipTransferred_RecipientNotification,
133          ""
134      );
135  }
```

Recommendation

Change to:

```
118  function _acceptOwnership() internal virtual {
119      require(msg.sender == pendingOwner(), "LSP14: caller is not the
     pendingOwner");
120
```

```
    address previousOwner = owner();
    _setOwner(msg.sender);
    delete _pendingOwner;

    _notifyUniversalReceiver(
        previousOwner,
        _TYPEID_LSP14_OwnershipTransferred_SenderNotification,
        ""
    );
    _notifyUniversalReceiver(
        msg.sender,
        _TYPEID_LSP14_OwnershipTransferred_RecipientNotification,
        ""
    );
}
```

[WP-G21] LSP-1: Check the `notifierMapValue` first can save gas

Gas

Issue Description

When `bytes12(notifierMapValue) != bytes12(0)` , the rather expensive external call of `ILSP7DigitalAsset(notifier).balanceOf(msg.sender)` is unnecessary.

Moving L61-62 down to after L66 can save gas from the unnecessary external call in such case.

https://github.com/lukso-network/lsp-smart-contracts/blob/b3169b44a5df0aca6f001f762df10a127142cda9/contracts/LSP1UniversalReceiver/LSP1UniversalReceiverDelegateVault/LSP1UniversalReceiverDelegateVault.sol#L38-L85

```
38  function universalReceiver(
39      bytes32 typeId,
40      bytes memory /* data */
41  ) public payable virtual returns (bytes memory result) {
42      if (msg.value != 0) revert NativeTokensNotAccepted();
43      // This contract acts like a UniversalReceiverDelegate of a Vault where we append the
44      // address and the value, sent to the universalReceiver function of the LSP9, to the msg.data
45      // Check https://github.com/lukso-network/LIPs/blob/main/LSPs/LSP-9-Vault.md#universalreceiver
46      address notifier = address(bytes20(msg.data[msg.data.length - 52:]));
47
48      (bool invalid, bytes10 mapPrefix, bytes4 interfaceID, bool isReceiving) = LSP1Utils
49          .getTransferDetails(typeId);
50
51      if (invalid || interfaceID == _INTERFACEID_LSP9) return "LSP1: typeId out of scope";
52
53      // solhint-disable avoid-tx-origin
54      if (notifier == tx.origin) revert CannotRegisterEOAsAsAssets(notifier);
55
56      bytes32 notifierMapKey = LSP2Utils.generateMappingKey(mapPrefix, bytes20(notifier));
57      bytes memory notifierMapValue = IERC725Y(msg.sender).getData(notifierMapKey);
```

```

    if (isReceiving) {
        // if the amount sent is 0, then do not update the keys
        uint256 balance = ILSP7DigitalAsset(notifier).balanceOf(msg.sender);
        if (balance == 0) return "LSP1: balance not updated";

        // if the map value is already set, then do nothing
        if (bytes12(notifierMapValue) != bytes12(0))
            return "URD: asset received is already registered";

        (bytes32[] memory receiverDataKeys, bytes[] memory receiverDataValues) = LSP5Utils
            .generateReceivedAssetKeys(msg.sender, notifier, notifierMapKey, interfaceID);

        IERC725Y(msg.sender).setData(receiverDataKeys, receiverDataValues);
    } else {
        // if there is no map value for the asset to remove, then do nothing
        if (bytes12(notifierMapValue) == bytes12(0))
            return "LSP1: asset sent is not registered";
        // if it's a token transfer (LSP7/LSP8)
        uint256 balance = ILSP7DigitalAsset(notifier).balanceOf(msg.sender);
        if (balance != 0) return "LSP1: full balance is not sent";

        (bytes32[] memory senderDataKeys, bytes[] memory senderDataValues) = LSP5Utils
            .generateSentAssetKeys(msg.sender, notifierMapKey, notifierMapValue);

        IERC725Y(msg.sender).setData(senderDataKeys, senderDataValues);
    }
}
```

Recommendation

Change to:

```
if (isReceiving) {
    if (bytes12(notifierMapValue) != bytes12(0))
        return "URD: asset received is already registered";

    uint256 balance = ILSP7DigitalAsset(notifier).balanceOf(msg.sender);
    if (balance == 0) return "LSP1: balance not updated";
```

[WP-N22] LSP-6: Inconsistent `InvalidCompactByteArrayLengthElement` check

Issue Description

Unlike in `_verifyAllowedERC725YSingleKey()` , the `length > 32` check is not performed for the error `InvalidCompactByteArrayLengthElement` in `_verifyAllowedERC725YDataKeys()` .

https://github.com/lukso-network/lsp-smart-contracts/blob/28c00a0abafcb682ba3a3f7cdc6eab9b09dfb713/contracts/LSP6KeyManager/LSP6KeyManagerCore.sol#L599-L678

```
function _verifyAllowedERC725YSingleKey(address from, bytes32 inputKey, bytes memory allowedERC725YDataKeysCompacted) internal pure {
@@ 600,626 @@
    while (pointer < allowedERC725YDataKeysCompacted.length) {
        /**
            * save the length of the following allowed key
            * which is saved in `AllowedERC725YDataKeys[pointer]`
            */
        uint256 length = uint256(uint8(bytes1(allowedERC725YDataKeysCompacted[pointer])));

        /**
        * the length of the following key must be under 33 bytes
        */
        if (length > 32) revert InvalidCompactByteArrayLengthElement(length);

@@ 639,674 @@
    }

    revert NotAllowedERC725YDataKey(from, inputKey);
}
```

https://github.com/lukso-network/lsp-smart-contracts/blob/28c00a0abafcb682ba3a3f7cdc6eab9b09dfb713/contracts/LSP6KeyManager/LSP6KeyManagerCore.sol#L685-L784

```
function _verifyAllowedERC725YDataKeys(address from, bytes32[] memory inputKeys,
bytes memory allowedERC725YDataKeysCompacted) internal pure {
@@ 686,714 @@
    while (allowedKeysFound < inputKeys.length && pointer <
allowedERC725YDataKeysCompacted.length) {
        /**
            * save the length of the following allowed key
            * which is saved in `AllowedERC725YDataKeys[pointer]`
            */
        uint256 length =
uint256(uint8(bytes1(allowedERC725YDataKeysCompacted[pointer])));

        /*
            * transform the allowed key situated from `pointer + 1` until `pointer
+ 1 + length` to a bytes32 value
            * E.g. 0xfff83a ->
0xfff83a0000000000000000000000000000000000000000000000000000000000
            */
        bytes32 allowedKey = bytes32(allowedERC725YDataKeysCompacted.slice(
            pointer + 1,
            length
        ));

@@ 731,775 @@
    }
    // ...
}
```

[WP-D23] LSP-6: The corresponding `valueContent` of `valueType` `bytes[CompactBytesArray]` should be `bytes[]` instead of `Bytes`

Issue Description

According to the LSP-2-ERC725YJSONSchema.mdvaluecontent document:

`valuecontent`

> Describes how to interpret the content of the returned ***decoded*** value.

Combined with the description of `AddressPermissions:AllowedCalls:<address>` and `AddressPermissions:AllowedERC725YDataKeys:<address>` in LSP-6-KeyManager.md, and their `valueType` , their `valueContent` should be `bytes[]` instead of `Bytes` .

https://github.com/lukso-network/LIPs/blob/56a264e9832b65d856b9650075248b4e32f43912/LSPs/LSP-6-KeyManager.md#implementation

```
[
    {
        "name": "AddressPermissions[]",
        "key":
"0xdf30dba06db6a30e65354d9a64c609861f089545ca58c6b4dbe31a5f338cb0e3",
        "keyType": "Array",
        "valueType": "address",
        "valueContent": "Address"
    },
    {
        "name": "AddressPermissions:Permissions:<address>",
        "key": "0x4b80742de2bf82acb3630000<address>",
        "keyType": "MappingWithGrouping",
        "valueType": "bytes32",
        "valueContent": "BitArray"
    },
    {
        "name": "AddressPermissions:AllowedCalls:<address>",
        "key": "0x4b80742de2bf393a64c70000<address>",
        "keyType": "MappingWithGrouping",
        "valueType": "bytes[CompactBytesArray]",
        "valueContent": "Bytes"
    },
    {
        "name": "AddressPermissions:AllowedERC725YDataKeys:<address>",
        "key": "0x4b80742de2bf866c29110000<address>",
        "keyType": "MappingWithGrouping",
        "valueType": "bytes[CompactBytesArray]",
        "valueContent": "Bytes"
    }
]
```

Appendix

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by WatchPug; however, WatchPug does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication.

Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of 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 sole risk. Smart Contract technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.