Quantstamp

Alchemy Modular Account

Executive Summary

This audit report was prepared by Quantstamp, the leader in blockchain security.

Type	Modular Smart Contract Account
Timeline	2023-11-20 through 2023-12-19
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	ERC-6900: Modular Smart Contract Accounts and Plugins
Source Code	• alchemyplatform/modular-account #0e3fd1e
Auditors	• Nikita Belenkov Auditing Engineer • Shih-Hung Wang Auditing Engineer • Alejandro Padilla Auditing Engineer • Ruben Koch Auditing Engineer

Documentation quality	High
Test quality	High
Total Findings	33 Fixed: 21 Acknowledged: 11 Mitigated: 1
High severity findings	2 Fixed: 2
Medium severity findings	6 Fixed: 3 Acknowledged: 2 Mitigated: 1
Low severity findings	15 Fixed: 12 Acknowledged: 3
Undetermined severity findings	3 Fixed: 1 Acknowledged: 2
Informational findings	7 Fixed: 3 Acknowledged: 4

Summary of Findings

In this audit, we reviewed Alchemy's ERC-6900-compliant Modular Smart Contract Account (MSCA) implementation. That ERC standard is an extension of the ERC-4337 Account Abstraction Infrastructure, further standardizing smart contract accounts and so-called account plugins, which are smart contract interfaces that allow for composable logic within smart contract accounts. It provides further granularity and additional composability for ERC-4337's user operation validation and execution. Similar modularity is also defined for open-ended execution via calls outside of the ERC-4337 context.

The main concerns identified in this audit revolve around an incorrect assumption about execution hooks, leading to possibilities for session key holders to circumvent imposed token spending limits (MSCA-1) and an incorrect storage key derivation for session key data, causing all keys to share the same set of permissions in practice. Furthermore, some problematic side effects in cases where the MSCA is enabled to call itself were uncovered (MSCA-3 and MSCA-4).

The MSCA itself is a fairly abstract implementation that in itself provides little functionality and security. This is intended, as the MSCA mainly gains functionality and access control through installed plugins. However, this leads to multiple possibilities of plugin misconfiguration and requires careful maintenance by owners (MSCA-5, MSCA-6, MSCA-7, MSCA-8, MSCA-9, MSCA-10, MSCA-12, MSCA-17, MSCA-19, MSCA-21).

We also noticed that the pre-execution hook defined in the `SessionKeyPermissionsPlugin` is not idempotent (MSCA-8), as defined as a requirement for execution hooks in the specs. We argue that enforcing idempotency in execution hooks is also challenging and that enforcing non-overlapping execution hooks might be a better design.

This report also includes a finding that has been reported to us by the Alchemy team (MSCA-22).

Overall, we deem the code quality to be high. The test suite features 360 tests, but MSCA-1 and MSCA-2 show that the `SessionKeyPermissionsPlugin` would benefit from more sophisticated tests.

Fix Review Update

All issues have either been fixed, mitigated, or acknowledged. Acknowledged issues are an acceptable risk to the client and some will be addressed in future versions of the ERC. Significant changes have been made to address highlighted issues in the design, such as merging `SessionKeyPlugin` and `SessionKeyPermissionsPlugin`, redesigning hook behavior, and removal of injected and permitted call hooks.

Alchemy has followed good software development practices when introducing fixes, such as code peer review and thorough documentation of changes in the design. The test suite has been suitably updated and adjusted for the introduced changes, which now stands at 417 tests.

ID	DESCRIPTION	SEVERITY	STATUS
MSCA-1	Session Keys with ERC-20 Spend Limit Can Drain MSCA of Limited Tokens	High	Fixed
MSCA-2	Session Keys Share And Overwrite Each Other's Permissions	High	Fixed
MSCA-3	Access Control Bypass on Owner-Only Function Calls	Medium	Fixed
MSCA-4	Self-Calls Are Allowed to Execute Functions without a Runtime Validation Function	Medium	Fixed
MSCA-5	Malicious Plugin Could Prevent Uninstall and Lead to Denial-Of-Service	Medium	Acknowledged
MSCA-6	Other Plugins May Enable Runtime Execution of Session Keys	Medium	Mitigated
MSCA-7	State-Changing Hooks May Affect Validation Logic in Other Hooks	Medium	Acknowledged
MSCA-8	Non-Idempotent Pre-Execution Hook in `SessionKeyPermissionsPlugin`	Medium	Fixed
MSCA-9	A Plugin with an Injected Hook to Itself Cannot Be Uninstalled	Low	Fixed
MSCA-10	Danger Of Faulty Plugin Configurations	Low	Acknowledged
MSCA-11	Session Key ID Can Be Overwritten During Key Rotation	Low	Fixed
MSCA-12	Non-Reverting Execution Hooks In Case Of No `functionId` Match	Low	Fixed
MSCA-13	Separation of Key Storage Between `SessionKeyPermissionsPlugin` and `SessionKeyPlugin` Leads to Complications	Low	Fixed
MSCA-14	Certain Assumptions Are Made in the `PluginStorageLib` that Could Lead to Issues	Low	Fixed
MSCA-15	Missing Validation for Execution Function in `SessionKeyPermissionsPlugin` Hooks	Low	Fixed
MSCA-16	Lack of Function Call Context for Post-Execution Hooks	Low	Fixed
MSCA-17	Potential Risks of Dependency Mismatch	Low	Acknowledged
MSCA-18	Potential Reentrancy Issue During Plugin Uninstallation	Low	Fixed
MSCA-19	Uninstalling the Owner Plugin May Potentially Brick the Account	Low	Acknowledged
MSCA-20	Use of Solidity `transfer()` Function	Low	Fixed
MSCA-21	Plugins May Add the `IPlugin` Interface to MSCA	Low	Fixed
MSCA-22	`SIG_VALIDATION_FAILED` Returned in Incorrect Cases	Low	Fixed
MSCA-23	Ownership Can Be Renounced	Low	Fixed
MSCA-24	Potential EVM Compatibility Issue in Solidity Version >=0.8.21	Informational	Fixed

ID	DESCRIPTION	SEVERITY	STATUS
MSCA-25	Malicious Owner Could Front-Run `updateOwners()` Calls	Informational	Acknowledged
MSCA-26	Incorrect Storage Prefix Constant	Informational	Fixed
MSCA-27	Memory Allocated by `_allocateTempKeyBuffer` Is Fragile and Could Be Overwritten	Informational	Acknowledged
MSCA-28	Extra Care Needs to Be Taken if the Project Is to Be Deployed on ZK-Based Chains	Informational	Acknowledged
MSCA-29	Misuse of Functions in the `CastLib` Library Could Lead to Loss of Information	Informational	Fixed
MSCA-30	Potential Violation of ERC-4337 Storage Access Rules	Informational	Acknowledged
MSCA-31	Different Order of Owners Can Produce Different MSCAs	Undetermined	Fixed
MSCA-32	Plugin Metadata Can Be Misleading to End Users	Undetermined	Acknowledged
MSCA-33	Session Keys Require Explicitly Specifying a Spend Limit for Callable ERC-20 Contract	Undetermined	Acknowledged

Assessment Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

Disclaimer

Only features that are contained within the repositories at the commit hashes specified on the front page of the report are within the scope of the audit and fix review. All features added in future revisions of the code are excluded from consideration in this report.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

1. Code review that includes the following
    1. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
    2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
    3. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
    1. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
    2. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarity, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.

4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Scope

The scope included everything in the `src/` folder that included the Modular Smart Contract Account implementation along with the libraries used for the account. The scope also included 4 Plugins: `TokenReceiverPlugin`, `MultiOwnerPlugin`, `SessionKeyPlugin` and `SessionKeyPermissionsPlugin` along with the base plugin implementation used as a library for the mentioned plugins.

The audit was focused on the Modular Smart Contract Account, that builds on top of the ERC-4337. The actual bundler, entry point, or any other aspects of ERC-4337 were out of scope for this audit and are assumed to be working as expected.

Files Included

`src/*.sol`

Files Excluded

Everything outside the files specified in the included section.

Findings

MSCA-1

Session Keys with ERC-20 Spend Limit Can Drain MSCA of Limited Tokens

High Fixed

Update

The system now always assumes an upper bounded spend limit by increasing the key's `limitUsed` field by X for all `approve(address, X)` calls. Fixed in commits `1da79ebf6df5b680283dd1709b7842b68e26248e` and `96f5feab588c79313f59987f1f4f035c1d4d3a0f`.

Alert

The upper bounded spend limit should be documented for the user, as uncareful use can fully consume a key's spending limit without approving or transferring any additional funds.

File(s) affected: `plugins/session/permissions/SessionKeyPermissionsPlugin.sol`

Description: Part of the permissions that can be set for session keys is an ERC-20 spend limit, with which the owner of an MSCA can grant certain permissions to holders of the session key to spend tokens in possession of the MSCA. Notably, multiple contract calls can be executed within a single `executeWithSessionKey()` user operation through multiple entries in the array of `Call` structs.

In a pre-execution hook, the calldata of each of the calls encoded in the user operation is analyzed for the `transfer()` and `approve()` selector, where the spending limit is adjusted based on the full `amount` parameter for transfers and on the supposed increase in allowance of the given address for approvals.

However, the current implementation of the pre-execution hook incorrectly assumes that no prior calls part of this user operation have adjusted the allowance.

The result of `IERC20(token).allowance(account, spender)` could incorrectly reflect the state at the time of execution of the current `Call` being validated. This enables any malicious holder of a session key with a non-zero ERC-20 token limit for a certain token to fully drain the MSCA's token balance.

Exploit Scenario:

1. The owner of the MSCA issues a session key to a subscription service, enabling it to withdraw 10 USDC per month from the MSCA's USDC balance
2. The session key holder submits an `executeWithSessionKey()` user operation that sets an allowance of `X`, `X` being the key's spending limit, for some `DrainerContract`. That `DrainerContract` has a single function `drain()`, calling `transferFrom()` on the token contract, consuming the entire allowance it holds and sending it to some address.
3. The plugin will correctly mark an increase in the `limitUsed` field of the key and `DrainerContract` will have an allowance of `X` after the successful execution of the user operation.
4. The session key holder issues another `executeWithSessionKey()` user operation, this time encoding multiple `Call` elements:
    1. `DrainerContract.drain()`
    2. `Token.approve(DrainerContract-address, X)`
5. The pre-execution hook will allow the first call to pass (assuming the key is allowed to call `DrainerContract`)
6. As part of the hook, `_getTokenSpendAmount()` is executed, querying the existing allowance from the token contract. As `DrainerContract.drain()` is not executed in the pre-execution context, an allowance of `X` is returned.

7. The function returns `approveAmount - existingAllowance`, resolving to `X - X = 0` for the approval, which will enable all checks to pass.
8. As a result, the session key holder managed to circumvent the spending limit, having drained `X` tokens with an existing allowance of `X` after the user operation execution. These two calls in sequence can be arbitrarily repeated, enabling full draining of the MSCA of all tokens where a non-zero spend limit is set for some session key.

We want to note that this attack would also be possible if the session key only had permission to interact with the limited token contract and not arbitrary addresses. In an alternative attack, the session key holder could approve the MSCA itself in one user operation, only to repeatedly call `Token.transferFrom(MSCA-address, drainingAddress, X)` and `Token.approve(MSCA-address, X)`, achieving the same effect.

Recommendation: Pre-execution hooks validating multiple calls can not assume any state from the blockchain to be unchanged after the first call -- or at least not given the current constraints. There are multiple potential fixes for the issue; however, each has its own tradeoffs:

1. Only allow the `increaseAllowance()` and `decreaseAllowance()` selectors, which are however not part of the official ERC-20 standard, though widely used.
2. Alternatively, always assume an upper bounded spend limit by increasing the key's `limitUsed` field by `X` for all `approve(adddress, X)` calls. Uncareful use of the key could, however, cause a key's spending limit to be reached without any increased allowances.
3. Have a default `ALLOW_LIST` in place for ERC-20 token spend limits that only enables the `approve()` and `transfer()` selector. Carefully document the risks of adding more selectors and whitelisting other external contracts.

We want to note that blacklisting the `transferFrom()` selector is not sufficient. Alternative scenarios of the issue are possible, where the allowance calls are chained with calls to some drainer contract that spends the allowance set as part of the user operation.

MSCA-2

Session Keys Share And Overwrite Each Other's Permissions

High Fixed

Update

`SESSION_KEY_DATA_PREFIX` inside `sessionKeyDataKey` has been removed as recommended. Fixed in commits `7e58e9993255ebf3e238b9330502d77da96249e3` and `27af95a1c8833b827dd249e987bf53a47998aa67`.

File(s) affected: `plugins/session/permissions/SessionKeyPermissionsBase.sol`

Description: The `SessionKeyData` struct of a session key defines the permission access of registered session keys. As this information has to be accessible as part of UserOp validation, it has to comply with the ERC-4337 storage restriction rules. The client developed a `PluginStorageLib` library that follows the `keccak256(A || X) + n` (`A` being the address of the account and `X` being an additional arbitrary key) storage access rule defined in the EIP, which essentially enables access to address-associated storage slots in the plugin's storage. See MSCA-30 for concerns regarding ERC-4337 storage access compliance.

The key derivations to get to the address associated storage slots are defined in `SessionKeyPermissionsBase.sol#L70-L91` and implemented in the underlying `_sessionKeyIdOf()` function.

The implementation of the `SessionKeyData` storage key differs slightly from the specs, resulting in a critical issue for the storage management of `SessionKeyData` structs.

While the `associatedStorageKey` follows the spec with `12 padding zeros || associated address || SESSION_KEY_DATA_PREFIX || batch index`, it is not just appended with the `sessionKeyId` as the specs suggest, but with `bytes32(abi.encodePacked(SESSION_KEY_DATA_PREFIX, SessionKeyId.unwrap(id)))` (`SessionKeyPermissionsBase.sol#L133`). This statement will incorrectly slice off the 4 right-most bytes of the `sessionKeyId`, which turns out to be a `uint256` counter variable, incremented upwards from zero for each account. In the EVM, `uint256` variables are stored in little-endian notation, with the least significant bytes stored in the right-most bytes of the storage slot. Therefore, as a result of incorrectly slicing off the 4 least significant bytes of the key ID, meaning the first 0xffffffff = 4294967295 registered keys within each account share the same storage slot, meaning that they share the same `SessionKeyData` object defining a key's permissions.

As a consequence, the entire access control around session keys is fully compromised for all MSCAs, as practically all keys an individual MSCA would ever issue share the same permissions.

SessionKeyData:

Specs:

`... SESSION_KEY_DATA_PREFIX || batch index || sessionKeyId`

Code:

`... SESSION_KEY_DATA_PREFIX || batch index || **SESSION_KEY_DATA_PREFIX** || sessionKeyId with righ-most 4 bytes sliced off`

Exploit Scenario: Below is a PoC for this issue, which can be run in the `SessionKeyPermissixonsPlugin.t.sol` test file:

```
function test_sessionKeyDataOverwrite() public {
    address requiredPaymaster = address(uint160(0x23456));
    address myKey = address(uint160(0x1122334455));

    vm.startPrank(owner1, owner1);
    // register myKey
```

```
    SessionKeyPermissionsPlugin(address(account1)).registerKey(myKey, bytes32(uint(1)));

    // update required paymaster for sessionKey1
    bytes[] memory updates = new bytes[](1);
    updates[0] = abi.encodeCall(ISessionKeyPermissionsUpdates.setRequiredPaymaster, (requiredPaymaster));
    SessionKeyPermissionsPlugin(address(account1)).updateKeyPermissions(sessionKey1, updates);
    vm.stopPrank();

    // Check the required paymaster for sessionKey1
    address returnedRequiredPaymaster =
        sessionKeyPermissionsPlugin.getRequiredPaymaster(address(account1), sessionKey1);
    assertEq(returnedRequiredPaymaster, requiredPaymaster);

    // bug: the required paymaster for myKey is also modified
    sessionKeyPermissionsPlugin.getRequiredPaymaster(address(account1), myKey);
}
```

Recommendation: As `SESSION_KEY_DATA_PREFIX` is included in `prefixAndBatchIndex` at L129, there is no need to include it again in `sessionKeyDataKey`, but only the session key ID is necessary. Consider removing L133 and modifying L134 as follows:

```
return _toSessionKeyData(PluginStorageLib.associatedStorageLookup(associatedStorageKey,
SessionKeyId.unwrap(id)));
```

MSCA-3 Access Control Bypass on Owner-Only Function Calls

Medium · Fixed

Update

A check has been added in `executeFromPluginExternal()`, which rejects any calls to the MSCA itself. Fixed in `807823b260e5617f81c6784aee639a2fcce549b9`.

Update

The upgrade flow has not been reworked as part of the fixes and will possibly be revisited in future versions.

File(s) affected: `plugins/owner/MultiOwnerPlugin.sol`

Description: The `MultiOwnerPlugin` defines the `runtimeValidationFunction()` function to protect the runtime calls to the native functions of MSCA. This function ensures that the caller is either the MSCA itself or one of the authorized owners. The choice of allowing MSCA to call the owner-only functions seems to be designed to allow the owner to uninstall the owner plugin and install another simultaneously in an `executeBatch()` call.

However, such a design could pose risks to the MSCA and potentially allow an unauthorized party to execute the owner-only functions due to the design of plugin external calls. A plugin can be allowed to perform calls to external contracts, including the MSCA itself, via the `executeFromPluginExternal()` function. If a plugin is allowed to do so, it can, for example, execute `MSCA.upgradeToAndCall()`, to upgrade the account implementation without being an authorized owner.

Note that `SessionKeyPlugin` is allowed to execute calls to any external contracts. While the `SessionKeyPermissionsPlugin` enforces specific permissions on the session keys, if any user with a session key with `ContractAccessControlType` of `NONE` type (i.e., no access control enforced on the called external contracts), they would be able to perform arbitrarily calls to the owner-only functions.

Recommendation: Consider whether the plugins should be allowed to call back the MSCA via `executeFromPluginExternal()`. If not, consider adding a check in `executeFromPluginExternal()` to ensure the target is not the MSCA.

Additionally, consider whether the upgrade flow of the owner plugin or the MSCA can be revised to avoid needing the owner plugin to allow calls from the MSCA. For example, consider defining a native function, `uninstallUpgradeAndInstall()`, to allow the upgrade of the MSCA without relying on grouping multiple calls into an `executeBatch()` call.

MSCA-4 Self-Calls Are Allowed to Execute Functions without a Runtime Validation Function

Medium · Fixed

Update

The conditions have been changed as recommended. Fixed in `ef789cdb3fae133390d0464b36861fbae6cdd8b8`.

File(s) affected: `account/UpgradeableModularAccount.sol`

Description: In numerous instances, the MSCA is able to perform a call on itself ("self-call"). We identified three distinct ways in which self-calls can be achieved:

1. Via the successful call to the `execute()` and `executeBatch()` selectors, which require a passing `preNativeFunction()` check. In the reference implementation, these selectors are protected by the `MultiOwnerPlugin`'s runtime validation, so require the `msg.sender` to provide a valid signature or require the signer of the user operation to be a registered owner.
2. Via the `executeWithPluginExternal()` selector, self-calls can be achieved by all plugins with specifically the MSCA itself listed as a permitted address or with a plugin specifying an activated `permitAnyExternalAddress` flag in its manifest. For example, any holder of a session key registered in an MSCA with access control enabling calling the MSCA itself can do self-calls via `executeWithSessionKey()`, as it calls the `executeWIthPluginExternal()` selector. Such a key would need to be enabled to call the MSCA via `accessControlType=ALLOWLIST` and the MSCA specifically being registered as a callable address, or if the MSCA itself is not being registered in a configured `accesscontrolType=DENYLIST` or if the key can call arbitrary addresses via `accesscontrolType=NONE`.
3. If the MSCA itself were to add the `IPlugin` interface to its `supportedInterface` mapping through a plugin installation (see MSCA-23), self-calls could be achieved through `executeWithPlugin()` calls.

The `_doRuntimeValidation()` function looks up the runtime validation function associated with the given function selector and performs the runtime validation call to the corresponding plugin. While generally, attempted runtime validation fails if no associated runtime validation is found, a special condition at `UpgradeableModularAccount.sol#L617-L625` is evaluated to check if the call should be allowed.

According to the comments at `UpgradeableModularAccount.sol#L626-L630`, the MSCA should prohibit any runtime call with no associated runtime validation function except the cases of self-calls to `installPlugin()` or `upgradeToAndCall()`. However, due to the use of `||` at L622, which seems to be a logical error, any self-calls to the MSCA are allowed, causing a divergence between the code and the comments. As a result, a permitted plugin or owner can execute functions that should not be executed in runtime context due to this logical error in the code.

For example, the `executeWithSessionKey()` function provided by the `SessionKeyPlugin` does not have a runtime validation function, as session keys are intended to be used in user operation context only. However, an owner of the MSCA can execute the `executeWithSessionKey()` function in runtime using self-calls via `executeBatch()` and potentially affect the token or gas usages of an arbitrary session key without obtaining a valid signature of the key. However, note that in the current design, the owner of MSCA can arbitrarily control the spending limits of session keys.

Recommendation: If the code should follow the comments at L626-L630, consider rewriting the condition at L620-L623 as follows:

```
(
  msg.sig != IPluginManager.installPlugin.selector
    && msg.sig != UUPSUpgradeable.upgradeToAndCall.selector
) || msg.sender != address(this)
```

MSCA-5
Malicious Plugin Could Prevent Uninstall and Lead to Denial-Of-Service

Medium | Acknowledged

> **Update**
> Alchemy has acknowledged this issue and added to the document of risks.

File(s) affected: `account/UpgradeableModularAccount.sol`, `account/PluginManagerInternals.sol`

Description: Plugins can register hooks to offer custom functionality during various stages of a user operation or runtime transaction. However, a malicious plugin can abuse this by introducing hooks that consistently revert, preventing the user from uninstalling the plugin or upgrading the account. As arbitrary selectors can be specified, this could completely brick the account.

This issue outlines the most significant impact of the dangers of the plugin trust model. Other cases are covered in MSCA-10.

Exploit Scenario:

1. Alice, not paying close attention, installs a malicious plugin A on her MSCA.
2. This malicious plugin registers a `preUserOpValidationHook` and a `preRuntimeValidationHook` on the `UpgradeableModularAccount.uninstallPlugin` selector. These hooks always revert, rendering the plugin impossible to uninstall, even with the `forceUninstall` flag set to true.
3. The plugin also registers malicious `preExecHooks` and other validation hooks on crucial selectors, like `UpgradeableModularAccount.execute`, effectively rendering the account unusable.

Recommendation: Consider an emergency mechanism that can be used to uninstall a misbehaving plugin completely. **Note:** This design has to be carefully designed to ensure that it is not possible to abuse this to disable a legitimate plugin, such as the `MultiOwnerPlugin`.

MSCA-6
Other Plugins May Enable Runtime Execution of Session Keys

Medium | Mitigated

> **Update**
> A `preRuntimeValidationHook` has been added to `this.executeWithSessionKey()` will always revert the call. This fixed the issue for the session key plugin but not the general issue. Alchemy has informed us that this would be reviewed for future versions of the MSCA. Mitigated in `2a22fbc187958f99785297a8475aed4f69f0d744`.

File(s) affected: `account/UpgradeableModularAccount.sol`, `plugins/session/SessionKeyPlugin.sol`

Description: As mentioned in the code comments at L245-L247, session keys are only expected to be used in the user operation context. Therefore, `SessionKeyPlugin` does not define the runtime validation function for `executeWithSessionKey()`, with the assumption that any function call in runtime to `executeWithSessionKey()` will revert due to the lack of validation function.

In usual cases, such design should work as expected since `UpgradeableModularAccount` reverts with a `RuntimeValidationFunctionMissing()` error if the validation function corresponding to a runtime call does not exist, except the cases for self-calls to `installPlugin()` or `upgradeToAndCall()`.

However, any other installed plugin can add a runtime validation function to `executeWithSessionKey()`, enabling and allowing session keys in runtime. Also, during the runtime call, the session key validation implemented in `userOpValidationFunction()` will be bypassed and not enforced.

Exploit Scenario:

1. The owner of an MSCA installs `SessionKeyPlugin` and `SessionKeyPermissionsPlugin`.
2. The owner installs Plugin A, which adds a runtime validation function to `executeWithSessionKey()` with a setting of `_RUNTIME_VALIDATION_ALWAYS_ALLOW`. The owner may install Plugin A accidentally or be tricked into installing the malicious plugin.
3. As a result, anyone can use the existing session keys in the MSCA and consume the spend limits since the session key validation is bypassed.

Recommendation: Consider explicitly adding a runtime validation function to `executeWithSessionKey()` and revert any runtime calls without leaving the validation function unset.

Alternatively, define a `_RUNTIME_VALIDATION_ALWAYS_DENY` constant in `FunctionReferenceLib` to implement an always-reverting validation function.

MSCA-7
State-Changing Hooks May Affect Validation Logic in Other Hooks

Medium | Acknowledged

> **Update**
> Alchemy has acknowledged this issue and commented: "This issue is resolved with the fix for MSCA-1. The only left over work is documentation on warning devs to not rely on current states in the hook check as they can change later"

File(s) affected: `account/UpgradeableModularAccount.sol`, `plugins/session/permissions/SessionKeyPermissionsPlugin.sol`

Description: Plugins can define pre- or post-execution hooks to a specific execution function, which allows them to perform additional validations or modify states before or after the execution function call.

However, when a plugin receives a call to its pre-execution hook, it should not assume that the state at the time of the pre-hook call will remain unchanged when the execution function is called. This is because the MSCA implementation allows overlapping hooks, so multiple pre-hooks can be executed after the current pre-hook but before the execution function call. Furthermore, The execution order of the hooks cannot be enforced by the plugins and depends on the account's internal state. Therefore, validation in a pre-hook based on the current state may no longer be valid if the state being checked is changed by another pre-hook before the execution function call.

For example, the `SessionKeyPermissionsPlugin` defines a pre-execution hook, which invokes the `_updateLimitsPreExec()` function to update the ERC-20 token spend limits of a session key based on the `transfer()` or `approve()` value. If the call is `approve()`, this function relies on the current allowance to the spender to calculate the increase in the amount spent. However, such a calculation can be erroneous as it assumes no allowance change will occur after the current pre-hook call.

A similar issue may exist in post-hooks and validation hooks as well. For example, the state at the time of a post-hook call may not be the exact state after the execution function is executed since there may be other post-hooks in between and change the state.

Exploit Scenario: The following scenario demonstrates how the interference between pre-hooks can affect validations based on the current contract state:

1. The owner installs Plugin A, which adds a pre-hook to the `executeWithSessionKey()` function.
2. The owner of an MSCA installs `SessionKeyPlugin` and `SessionKeyPermissionsPlugin`.
3. A session key user sends a user operation to invoke the `executeWithSessionKey()` function. The `calls[]` array specified in the `executeWithSessionKey()` function includes only one call, which is `USDC.approve(spender, 100)`. Suppose that before the call, `USDC.allowance(MSCA, spender)` is 100.
4. The pre-hook from `SessionKeyPermissionsPlugin` is executed before Plugin A's since Plugin A is installed first.
5. During the pre-hook call from `SessionKeyPermissionsPlugin`, the `_updateLimitsPreExec()` function gets the current allowance and the approved amount. No spend limit is consumed since both amounts are 100.

6. During the pre-hook call from Plugin A, the plugin triggered the `spender` to call `USDC.transferFrom(MSCA, 100)`. The allowance is then reduced to 0.
7. During the `executeWithSessionKey()` call, the MSCA approves the `spender` with another 100 USDC.
8. As a result, the session key user successfully transfers 100 USDC from the MSCA to the `spender` without reducing their USDC spend limit. The user can execute the above steps repeatedly to transfer more USDC from the MSCA.

Recommendation: Consider modifying the `_updateLimitsPreExec()` function so that the spent amount calculation does not depend on a current state (which may change later). Also, clarify the potential interference between hooks in the documentation so that plugin developers can be aware of such issues and risks.

MSCA-8
Non-Idempotent Pre-Execution Hook in `SessionKeyPermissionsPlugin`

Medium | Fixed

> **Update**
> Fixed by executing the pre-execution hook once, even if there are multiple post-execution hooks (our first recommendation). This way, pre-execution hooks do not have to be idempotent, and therefore, there is no need to change `_updateLimitsPreExec()` in `SessionKeyPermissionsPlugin`. Fixed in commit `28c7bf1c022d5a71427c58e65e5feefc3640b713`.

File(s) affected: `account/UpgradeableModularAccount.sol`, `plugins/session/permissions/SessionKeyPermissionsPlugin.sol`

Description: According to the MSCA specification, hooks are assumed to be idempotent, so deduplication can be performed when multiple hooks overlap. However, the pre-execution hook defined in `SessionKeyPermissionsPlugin` is not idempotent, specifically the `_updateLimitsPreExec()` function. If the `_updateLimitsPreExec()` function is executed more than once during the `executeWithSessionKey()` call, the spend limits will be reduced multiple times.

Any plugin can add another plugin's hooks to the MSCA without permission from the providing plugin (except in the case of injected hooks). Therefore, a hook can be executed multiple times even though the providing plugin specifies it once in the manifest, which may lead to unexpected results for non-idempotent hooks.

Exploit Scenario:

1. The owner of an MSCA installs `SessionKeyPlugin` and `SessionKeyPermissionsPlugin`.
2. The owner installs Plugin A, which adds a pre- and post-hook pair. The pre-hook is exactly provided by `SessionKeyPermissionsPlugin` (denoted as `H1`), while the post-hook is an arbitrary non-empty hook (denoted as `H2`).
3. The owner installs Plugin B, which adds the same pre-hook from `SessionKeyPermissionsPlugin`, but with an arbitrary non-empty post-hook different than A's (denoted as `H3`).
4. According to the hook deduplication logic, the `executeWithSessionKey()` function ends up having two pre- and post-execution hook pairs: `(H1, H2)` and `(H1, H3)`.
5. As a result, when `executeWithSessionKey()` is called, `H1` is called twice, causing the spend limits to be reduced twice as well.

Recommendation: In general, if a hook is guaranteed to be executed only once in the execution context, it does not have to be idempotent. Therefore, one possible mitigation is to modify the `_doPreHooks()` function implementation in the `UpgradeableModularAccount` contract to remove the repeated executions of a given pre-hook (L719-L35) but only execute it once since the same output of the pre-hook call can be provided to all associated post-hooks as the input. However, post-hooks will still need to be idempotent since this approach cannot remove the duplicated execution of post-hooks.

Another approach is to introduce a way for plugin developers to specify that a hook should be executed only once in the execution context. The MSCA reverts the transaction if this condition is not satisfied.

Alternatively, consider modifying the `_updateLimitsPreExec()` function to make it idempotent. Requiring execution hooks to be read-only would ensure them to be idempotent, which, however, would limit the use case of hooks and may not be applicable in this ERC-20 spending limit scenario.

MSCA-9
A Plugin with an Injected Hook to Itself Cannot Be Uninstalled

Low | Fixed

> **Update**
> The code base for plugin installation has been reordered so that the plugin manifest is only added at the end, resulting in a failed check in case of self-dependencies. Furthermore, the functionality of injected hooks has been fully removed. Fixed in commit `4016fe86d83e47697108661e724ca09b56d8abd4`.

File(s) affected: `account/PluginManagerInternals.sol`

Description: Users can provide additional injected hooks not specified in the plugin manifest during the plugin installation. Injected hooks have the same behavior as permitted call hooks, which are executed when the plugin executes a permitted call to another plugin or an external contract.

However, plugins with injected hooks that execute functions defined by themselves cannot be uninstalled after installation. While adding the injected hooks, the plugin creates a dependency on itself. Therefore, the plugin cannot be uninstalled since the plugin's dependency count will be at least one at the time of the uninstall call, essentially causing a deadlock.

This issue can further impact other plugins that the uninstallable plugin depends on, as they would also become uninstallable.

Recommendation: Consider whether the case of a plugin with injected hooks to itself is allowed. If so, consider not increasing the dependency count of the plugin if the providing plugin of the hook is the plugin being installed.

MSCA-10 Danger Of Faulty Plugin Configurations

Low Acknowledged

Update

These issues have been acknowledged and added to the documentation. Alchemy has addressed each point briefly here: "

1. Users attempting to uninstall the owner plugin must simultaneously install another owner plugin in the same transaction via executeBatch.
2. Users should be careful when installing unverified plugins, and the client should surface to the user the plugin manifest details as much as possible during installation.
3. Upgradable plugins can be dangerous and users and clients should take similar precautions as above.
4. Plugins should be installed from trusted sources.
5. Acknowledged and accepting this security/design tradeoff.
6. Since hooks can no longer be provided as dependencies, this should not occur. In other cases, applying hooks ahead of the selector being installed is actually a desirable feature to increase security during plugin installation.
7. Account upgrades should be done with care. If the account storage layout is changing, plugins must be uninstalled prior to the upgrade to maintain consistency.

"

File(s) affected: `account/UpgradeableModularAccount.sol`

Description: The MSCA's security fully relies on a proper configuration of trusted plugins. Therefore, end users should exercise extreme caution around installing and even uninstalling plugins. Below is a non-exhaustive list of areas of concern that are based on plugin trust assumptions.

1. The MSCA does not provide any native protection of function selectors (though it natively also can not properly spend native assets and tokens). So uninstalling plugins could incorrectly loosen access around certain functionality, for example, if the `MultiOwnerPlugin` were to be uninstalled without any replacement protecting the same selectors to the same degree.
2. A malicious plugin can block access to arbitrary selectors, as outlined in MSCA-5.
3. Furthermore, plugins should always be assured to not be upgradeable, or else unexpected functionality could be introduced via registered validation and hooks.
4. An installation of a malicious or even faulty plugin could cause arbitrary consequences, including loss of funds.
5. Installing plugins that add execution hooks to selectors increases the chances of a validation passing, but reverting user operation execution. While this is a desired design aspect of execution hooks, the Entry Point contract will assure that a Bundler is reimbursed, either by a paymaster or the paying account, for validated user operation, regardless of the success of the execution. Therefore, plugins adding execution hooks to selectors add a level of risk to the account.
6. Uninstalling a plugin providing a certain selector and installing another one using the same selector can cause unforeseen behavior, as other plugins from the prior installation might still have hooks and validation associated with the replaced selector
7. To upgrade the MSCA, one would need to call `upgradeToAndCall()`, and it is up to the user to either uninstall or not the plugins, depending on the severity of the upgrade. So, it is possible not to uninstall plugins and have such an upgrade that either adds functions or changes the function selectors on existing functions. This would lead to an uncertain state, as a malicious plugin can pre-register a hook for a non-existing selector that would be added at an upgrade and hence hijack execution.

Recommendation: Document the dangers and trust assumptions around plugins to end users.

MSCA-11 Session Key ID Can Be Overwritten During Key Rotation

Low Fixed

Update

An assert has been added that checks that the new key has not been used yet. Fixed in commit `bfb9b652ffd94d53b7dc5bfc406ed615e11aa284`.

File(s) affected: `plugins/session/permission/SessionKeyPermissionsPlugin.sol`

Description: The `SessionKeyPermissionsPlugin.rotateKey()` function allows the owner to assign a `keyId` to a new session key. However, it does not ensure that the new session key, which can be already assigned to an existing `keyId`, is not in use. If so, the existing `keyId` will be overwritten, and this action is irreversible.

Recommendation: Consider checking if the `keyId` corresponding to the new session key is 0 before overwriting it.

MSCA-12
Non-Reverting Execution Hooks In Case Of No `functionId` Match

Low | Fixed

> **Update**
> Nonmatching `functionId`s now revert. Fixed in commit `3c1b5b6002be43e42c9c4feba182a83aaa2d139a`

File(s) affected: `plugins/session/permissions/SessionKeyPermissionsPlugin.sol`

Description: The reference implementation provides only one instance of a pre-execution hook in the `SessionKeyPermissionsPlugin`. It implements one `functionId` `PRE_EXECUTION_HOOK_UPDATE_LIMITS` that, if provided in the `functionId` parameter, will lead to the `_updateLimitsPreExec()` hook being executed. Notably, the pre-execution hook does not revert if the provided parameter does not match the `PRE_EXECUTION_HOOK_UPDATE_LIMITS` value and instead returns an empty string.

This is in contrast to the implemented (pre-)userop-validations and (pre-)runtime validations of the codebase that revert in case no implemented `functionId` is provided.

In the current state of EIP-6900, there is no requirement listed for pre-execution hooks to revert in such a scenario; it only states that "To indicate the entire call should revert, the function MUST revert.", which can also be found in the same wording in the comments for the (pre-)runtime validation functions.

A malicious plugin could seem like it integrates with a plugin's (pre-) execution hook but actually does not because it provides a non-matching functionId in its `ManifestFunction` definition.

Recommendation: Consider defining on an EIP level that in case any hook or validation contains at least one defined `functionId`, it should revert in case a non-implemented functionId was provided.

As an additional layer of protection against misconfiguration, consider adjusting the manifest to record for each of the hooks and validations a list of implemented `functionId`s. These would then also be copied over to the MSCA as additional fields in the `PluginData` struct in the `pluginData` mapping. These are then validated against dependencies provided as part of the installation process of plugins, e.g., if plugin B says it will add `(A, 1)` as a pre-exec hook, the MSCA will verify that plugin `A` has `1` in its pre-exec hook's `validFunctionIDs` set.

MSCA-13
Separation of Key Storage Between `SessionKeyPermissionsPlugin` and `SessionKeyPlugin` Leads to Complications

Low | Fixed

> **Update**
> `SessionKeyPlugin` and `SessionKeyPermissionsPlugin` have been merged. Fixed in commit `617677a0703da3107e5980c89162eb510254ae16`.

File(s) affected: `plugins/session/permissions/SessionKeyPermissionsPlugin.sol`, `plugins/session/permissions/SessionKeyPlugin.sol`

Description: `SessionKeyPermissionsPlugin` and `SessionKeyPlugin` are supposed to maintain a database of the keys that are used through the plugin system. These 2 datasets of keys are supposed to match up; otherwise, it leads to inconsistent states between the 2 plugins. When one calls `registerKey()` and hence registers it, the condition of having the same key registered in the `SessionKeyPlugin` is not enforced. So if a user already has a key registered with both plugins, they could pass the user op validation function; one then can register random keys inside `SessionKeyPermissionsPlugin`, which could not be used as they are not registered with the other plugin. A similar but different issue occurs if a user uninstalls the `SessionKeyPermissionsPlugin` and reinstalls it at a later point without removing any keys `SessionKeyPlugin`, all the session keys that were present at the time of uninstallation of the `SessionKeyPermissionsPlugin` would be re-instantiated with the same permissions. This is because in the `onUninstall()` callback, the storage slots associated with the session keys are not reset. However, the `SessionKeyPermissionsPlugin` itself does not have access to an iterateable list of registered session keys from which the storage slots would be derived because that data is stored in the `SessionKeyPlugin` This is a side effect of having 2 coexisting data structures for storing the key values.

Recommendation: There are 2 main approaches to solving this issue:

1. Consider refactoring the `onUninstall()` callback in the `SessionKeyPermissionsPlugin` to clear both the session keys and any other data related to the associated account and enforce `SessionKeyPlugin.isSessionKey()` during the `registerKey()` process to verify the key does indeed exist in the other plugin.
2. If the `SessionKeyPlugin` and the `SessionKeyPermissionsPlugin` were to be merged, the `AssociatedLinkedListSet` of the registered session keys of the associated address could be iterated over if the `SessionKeyPermissionsPlugin` were to be uninstalled from an MSCA, enabling resetting all the storage slots related to registered session keys before the plugin removal. This would also remove the issue of the key state inconsistency

MSCA-14
Certain Assumptions Are Made in the `PluginStorageLib` that Could Lead to Issues

Low | Fixed

Update

The upper bits are now cleared. Fixed in `dec1d76bf1916c32977679ea4161fe28c5fc87ef`.

Update

Other 3 suggestions have been acknowledged with the comment: "Zero values are an acceptable thing to include in the hash input (they don't lead to collisions on their own). The key size enforcement is not needed for safe use. Change the type of the key variable to a bytes32 is not necessary since it is already a single stack slot type as a bytes memory."

File(s) affected: `libraries/PluginStorageLib.sol`

Description: `PluginStorageLib` is a library for allocating and accessing ERC-4337 address-associated storage within plugins. There are multiple assumptions that are used through the library but are not enforced in the library code. In the current state of the codebase, the assumptions are followed, but as more plugins are developed, this cannot be guaranteed. Below is a list of current assumptions that will cause issues if not enforced:

1. The library allows a `keySize` of 256, which corresponds to how many words of memory are needed to look up a specific value. The `keySize` of max 2 is currently used, but the size of the key is not enforced. For example, if an address-associated storage key with `keySize = 2` is allocated, there is nothing stopping storage lookup with that memory allocation with only `input1` and no `input2`. Assure that the `keySize` corresponds to the number of key arguments provided.
2. The `key` variable size is currently unbound, which is not clear as it stores a slot in the memory and hence should be `bytes32,` or it will lead to unexpected outcomes.
3. One should not be allowed to have keys of a value of 0, as this could lead to collisions and unexpected behavior. Make sure that the `input` keys are checked and are not 0 inside the library.
4. Upper bits of assembly input are assumed to be 0s in `allocateAssociatedStorageKey()`, which might not be the case in `addr`, as it is an `address` and hence the upper bits are not guaranteed to be 0. Assure that the input is properly cleared in a similar manner to the `_allocateTempKeyBuffer()` function in `AssociatedLinkedListSetLib`. According to the latest Solidity Inline Assembly documentation at the time of writing:

   "If you access variables of a type that spans less than 256 bits (for example `uint64`, `address`, or `bytes16`), you cannot make any assumptions about bits not part of the encoding of the type. Especially, do not assume them to be zero. To be safe, always clear the data properly before you use it in a context where this is important [...]"

Recommendation: Consider making changes to the library to follow the recommendations above.

MSCA-15
Missing Validation for Execution Function in `SessionKeyPermissionsPlugin` Hooks

Low · Fixed

Update

A selector check has been added as recommended. Fixed in commits `540b0e3884d1f3a315c4f9046c3af25b5d21fd76` and `617677a0703da3107e5980c89162eb510254ae16`.

Update

For the second part of our recommendation, the client responded: "If a plugin is authorized to call executeWithSessionKey, therefore passes the validations and hook checks, then updating limit for the session key is fine. No fix needed here."

File(s) affected: `plugins/session/permissions/SessionKeyPermissionsPlugin.sol`

Description: The pre-execution hook defined by `SessionKeyPermissionsPlugin` updates the spending limit of a session key according to the provided call data. However, it does not validate that the selector (the first 4 bytes of the calldata) is exactly `executeWithSessionKey.selector`.

Since a plugin can reuse another plugin's hook (by specifying it in the manifest or by injected hooks), if a plugin adds the pre-hook from `SessionKeyPermissionsPlugin` to a random execution function, whenever that function is executed, the pre-hook is executed as well. This can cause an issue as the pre-hook is designed to be applied only to the `executeWithSessionKey()` function. Otherwise, the spending limit of a session key may be reduced without the user actually spending any token or gas.

Recommendation: Generally, hooks should validate the function selector in the provided call data to ensure that it is applied on an execution function that the hook recognizes and reverts the transaction if not.

Consider checking whether the function selector is `executeWithSessionKey.selector` at the beginning of the `_updateLimitsPreExec()` function to avoid another plugin triggering the call to the pre-hook either accidentally or intendedly. For the same reason, the `_checkUserOpPermissions()` function should validate the function selector (i.e., `userOp.callData[:4]`) as well.

However, even if the function selector is validated in the pre-execution hook, the plugin still cannot ensure that the hook is not triggered by a random plugin via the `executeFromPlugin()` call. Checking whether the sender supports the `IPlugin` interface does not fully solve the issue since the sender can manipulate the return values of the `supportsInterface()` call. A possible mitigation is to add a flag that indicates whether the call to the hook is triggered via `executeFromPlugin()` or not.

MSCA-16 Lack of Function Call Context for Post-Execution Hooks

Low Fixed

> **Update**
> Hooks as dependencies have been removed outside of the singular user op validation and runtime validation, so the lack of context for post-execution hook dependencies is not an issue anymore. Fixed in commit `f258b31a1364be8779a382f48ab120a19e23fa86`.

File(s) affected: `account/UpgradeableModularAccount.sol`, `interfaces/IPlugin.sol`

Description: Unlike the other hooks, the parameters of post-execution hooks do not include the call data of the execution function but only the `functionId` and the returned data from the pre-execution hook.

Such a design causes an issue: A post-execution hook cannot validate if it is being applied on a recognized execution function selector. The `functionId` parameter is not enough for validation purposes, as any random plugin can add a hook to an arbitrary execution function with the corresponding `functionId`.

Exploit Scenario:

1. Plugin A provides a pre- and post-execution hook for an execution function `foo()`. The post-execution hook performs critical updates related to the MSCA caller and should be called only after `foo()`.
2. The owner of an MSCA installs Plugin A and B. Plugin B adds A's post-execution hook to another execution function, `bar()`.
3. When `bar()` is executed, A's post-execution hook is called, causing a potential error on the Plugin A's side.

Recommendation: Consider whether this is an intended design. If not, consider adding the `sender`, `value`, and `data` values as parameters to the post-execution hooks so that the receiving plugins can perform further validation before state changes.

MSCA-17 Potential Risks of Dependency Mismatch

Low Acknowledged

> **Update**
> Alchemy has left the following comment: "This requires a spec update to fix, so no changes for now. This will be addressed in a future ERC update. "

File(s) affected: `account/PluginManagerInternals.sol`, `plugins/session/SessionKeyPlugin.sol`, `plugins/session/permissions/SessionKeyPermissionsPlugin.sol`

Description: During installation, the dependencies of the being installed plugin are checked against the `dependencyInterfaceIds` in the plugin's manifest. The MSCA ensures that each dependency contract supports the given interface via ERC-165. This approach provides a basic validation of the contracts. However, there are several existing limitations:

1. If the interface of a dependency is provided as `IPlugin` (as in the session key plugins), we can only ensure the dependency contract is a plugin without knowing what type of plugin it is (e.g., `IMultiOwnerPlugin`).
2. The MSCA does not validate that the provided `functionID` is valid for the given dependency.
3. Two plugins can implement the same plugin interface while being different on the function ID order or implementation.

Recommendation: Consider the following mitigations to the issues:

1. Replace `IPlugin` with `IMultiOwnerPlugin` in the `dependencyInterfaceIds` of the `SessionKeyPlugin` and `SessionKeyPermissionsPlugin`.
2. Allow the plugin to specify an optional dependency address in its manifest. If a dependency address is specified, the MSCA should ensure it matches the input parameter to the `installPlugin()` function call.
3. Allow plugins to specify a `validFunctionIDs` field in the manifest. The MSCA should ensure that the provided `functionID` in the dependency is in the `validFunctionIDs` list.

MSCA-18 Potential Reentrancy Issue During Plugin Uninstallation

Low Fixed

> **Update**
> `storage_.pluginData[args.plugin]` is now deleted before the external call, as recommended. Fixed in commit `10330fe1f4721566eb758590a54ab7a0240d4618`.

File(s) affected: `account/PluginManagerInternals.sol`

Description: The `_uninstallPlugin()` function in `PluginManagerInternals` performs a series of actions to remove the storage data corresponding to the plugin being uninstalled. Near the end of the function, the `onHookUnapply()` call is invoked on all providing plugins of the

injected hooks, then the `pluginData` structure is deleted, followed by the `onUninstall()` call to the plugin being uninstalled.

However, this pattern does not follow the best practice of reentrancy protection. During the `onHookUnapply()`, the to-be-installed plugin's `canSpendNativeToken`, `manifestHash`, and `dependencies` still exist in the storage, which may cause inconsistencies if these data are accessed (e.g., being used as a dependency since the manifest hash still exists).

Recommendation: Consider following the Check Effects Interactions (CEI) pattern by modifying the `_uninstallPlugin()` function to cache the injected hook data first, delete the plugin data completely, and then perform the `onHookUnapply()` and `onUninstall()` calls as the last steps of the function.

MSCA-19
Uninstalling the Owner Plugin May Potentially Brick the Account

Low | Acknowledged

> **Update**
> Alchemy acknowledged the issue and commented: "This risk is covered in our documentation. Users attempting to uninstall the owner plugin must simultaneously install another owner plugin in the same transaction via `executeBatch()`."

File(s) affected: `account/UpgradeableModularAccount.sol`

Description: The MSCA specification mentions that uninstalling the owner plugin would allow anyone to take over the account. However, this is not the case according to the current implementation, as the `_doRuntimeValidation()` function only allows self-calls when the runtime validation function of a selector does not exist. The `_doUserOpValidation()` function reverts if the corresponding user operation validation function does not exist.

As a result, an MSCA without the owner plugin will be uncontrollable, and funds will be locked in the account. Technically, an installed plugin with the self-call capability can self-call the MSCA to install or upgrade the implementation, which, however, even performing self-calls requires an initially provided and passing runtime validation in the base-call.

Recommendation: Consider clarifying the expected behavior of an empty MSCA or an MSCA without an owner account in the documentation and modifying the code accordingly.

MSCA-20 Use of Solidity `transfer()` Function

Low | Fixed

> **Update**
> `.transfer()` has been replaced with `.call()`. Fixed in commit `fd097ef7a179e296c3301c61304220ce2705d74e`.

File(s) affected: `factory/MultiOwnerMSCAFactory.sol`, `factory/MultiOwnerTokenReceiverMSCAFactory.sol`

Description: Solidity's `transfer()` function forwards a fixed amount of 2300 gas to the recipient when transferring native tokens to them, which causes several limitations if the recipient is a contract. First, the limited amount of gas prevents the contract from executing more complicated logic (e.g., external calls) in its `fallback()` or `receive()` function. Second, the gas costs of each opcode are subject to change; therefore, it is not guaranteed that currently successful transfers will not fail in the future.

The `withdraw()` functions in the listed contracts use `.transfer()` to transfer native tokens from the factory contracts.

Recommendation: Consider using `.call{value: value_}("")` to transfer native tokens to the recipient.

MSCA-21 Plugins May Add the `IPlugin` Interface to MSCA

Low | Fixed

> **Update**
> `IPlugin` is now not allowed in the `interfaceIds[]` array. Fixed in commit `912340322f7855cbc1d333ddaac2d39c74b4dcc6`.

File(s) affected: `account/PluginManagerInternals.sol`

Description: A plugin can add the `IPlugin` interface in the `interfaceIds[]` array in its manifest. As a result, the `IPlugin` interface will be added to the `supportedInterfaces` of the MSCA. This would cause any future self-calls via `execute()` or `executeBatch()` to be blocked since the MSCA will check whether the call target supports the `IPlugin` interface and, if so, reverts the call. It could also technically enable another possibility of self-calls, as outlined in MSCA-3.

Recommendation: Consider reverting the installation of a plugin if `IPlugin` is included in the `interfaceIds[]` array.

MSCA-22 `SIG_VALIDATION_FAILED` Returned in Incorrect Cases

Low | Fixed

Update

`SIG_VALIDATION_FAILED` is now returned in the correct cases. Fixed in commits `9ea511278c0d85ee86c957b09499ca34d87a35b9` and `436113e0c613f6aca705cd3aefef8c97b678baa5`.

Update

Alchemy noted the following for `MultiOwnerPlugin`: " Since owner can be an ERC-1271 compliant contract, we won't know the format of the signatures. Therefore, any invalid signature are treated as mismatched signatures in the ERC-4337 context."

File(s) affected: `plugins/owner/MultiOwnerPlugin.sol`, `plugins/session/SessionKeyPlugin.sol`

Description: The ERC-4337 specification states the following:

> If the account does not support signature aggregation, it MUST validate the signature is a valid signature of the `userOpHash`, and SHOULD return SIG_VALIDATION_FAILED (and not revert) on signature mismatch. Any other error should revert.

Currently, there are cases in the signature verification process where `SIG_VALIDATION_FAILED` is returned even in the case of a malformed signature, which should not be the case. Currently, in 2 cases, the function does not revert when it should:

1. `MultiOwnerPlugin.userOpValidationFunction()`
2. `SessionKeyPlugin.userOpValidationFunction()`

Exploit Scenario: Fixed in commit `9ea511278c0d85ee86c957b09499ca34d87a35b9`. The issue for `MultiOwnerPlugin` has been acknowledged with the comment: "Since the owner can be an ERC-1271 compliant contract, we won't know the format of the signatures. Therefore, any invalid signature are treated as mismatched signatures in the ERC-4337 context."

Recommendation: Make sure the `SIG_VALIDATION_FAILED` is only returned in the correct cases.

MSCA-23 Ownership Can Be Renounced

Low | Fixed

Update

`Ownable2Step` has been added. Fixed in commits `4d4f76a24242697ff4b4ebfa0f58347da7032224` and `4dc1f454e5970aed6f584f3306d785aa775fe260`.

File(s) affected: `factory/MultiOwnerMSCAFactory.sol`, `factory/MultiOwnerTokenReceiverMSCAFactory.sol`

Description: If the owner renounces their ownership, the following contracts will be left without an owner. Consequently, any function guarded by the `onlyOwner` modifier will no longer be able to be executed, which would remove the possibility of maintaining and withdrawing the factory's stake.

Recommendation: Confirm that this is the intended behavior. If not, override and disable the `renounceOwnership()` function in the affected contracts. For extra security, consider using a two-step process when transferring the ownership of the contract (e.g. `Ownable2Step` from OpenZeppelin).

MSCA-24
Potential EVM Compatibility Issue in Solidity Version >=0.8.21

Informational | Fixed

Update

EVM version has been set to `Paris`. Fixed in `a5baa79ee409f3239afd39285aa674cc1bd42810`.

File(s) affected: `foundry.toml`

Description: The `foundry.toml` configuration file shows that the MSCA code is compiled using version `0.8.21` of the Solidity compiler. By default, this compiler version uses the `Shanghai` version of the EVM. As a result, the contract bytecode may contain `PUSH0` opcodes, which some EVM chains, like Arbitrum and Optimism, don't support. This difference could cause the MSCA's deployment or execution to fail on these chains due to the use of an unsupported opcode.

Recommendation: For chains that don't support `PUSH0`, consider setting the EVM version as `Paris` in the Foundry configuration file. This will prevent the compiler from generating `PUSH0` opcodes in the bytecode. Alternatively, you could use a Solidity version of 0.8.19`.

MSCA-25
Malicious Owner Could Front-Run `updateOwners()` Calls

Informational | Acknowledged

Update

Alchemy has acknowledged this issue and commented: "This is a tradeoff that we're accepting for this iteration, and is covered in our documentation."

File(s) affected: `plugins/owner/MultiOwnerPlugin.sol`

Description: In the `MultiOwnerPlugin`, an MCSA can have multiple owners that each have equal weighting in terms of execution power. This brings advantages of multiple users or entities sharing the account but also complexities such as the one described below. Due to the nature of equal weighting, in the case where one owner's account is compromised or turns malicious, another benign user would attempt to remove the malicious one from the owner list via `updateOwners()`. Theoretically, that would work, but due to the blockchain nature, the malicious entity can monitor the memory pool and front-run the `updateOwners()` with `updateOwners()` of their own, removing the benign owner, hence causing their removal to revert due to lack of permissions.

This leads to a race condition style issue in the plugin.

Recommendation: Consider if this is an issue. If so, it can be mitigated in several ways. One of the approaches could be the addition of a super-admin that solely holds the ability to edit the owner list. Another approach could be to require multiple owners to call `updateOwners()` and only remove users when a sufficient number of owners have voted.

MSCA-26 Incorrect Storage Prefix Constant

Informational | Fixed

Update

`_ASSOCIATED_STORAGE_PREFIX` has been updated to the correct value. Fixed in commit `fe87e379d0d02907d2f3b4ddc403b37cc8a4e7a5`.

File(s) affected: `libraries/AssociatedLinkedListSetLib.sol`

Description: According to the inline documentation, the `_ASSOCIATED_STORAGE_PREFIX` should have a value calculated using `bytes4(keccak256("AssociatedLinkedListSet"))`. This value should be `0xf938c976`, but the current value is `0x9cc6c923`. It is unclear where this value came from.

Recommendation: Consider changing the constant from `0x9cc6c923` to `0xf938c976`.

MSCA-27 Memory Allocated by `_allocateTempKeyBuffer` Is Fragile and Could Be Overwritten

Informational | Acknowledged

Update

Alchemy has acknowledged this issue and added to the document of risks.

File(s) affected: `libraries/AssociatedLinkedListSetLib.sol`

Description: The `AssociatedLinkedListSetLib` hashes a composite key to store and retrieve items linked to an account. Given the frequency of this operation, the library employs the `_allocateTempKeyBuffer()` method to create a memory buffer once and only update it when necessary, all in assembly. However, the `_allocateTempKeyBuffer()` function does not update the free-memory pointer, meaning Solidity remains unaware of any memory changes. This could result in the buffer being mistakenly overwritten. Although the current code seems to avoid erroneous overwriting, the design is quite brittle. Even minor modifications could disrupt memory, potentially breaking the code.

Recommendation: Consider updating the free-memory pointer at position `0x40` in the `_allocateTempKeyBuffer()` method to reflect the data pushed into memory. This will only increase gas consumption marginally, but it will significantly enhance the robustness of the code.

MSCA-28 Extra Care Needs to Be Taken if the Project Is to Be Deployed on ZK-Based Chains

Informational | Acknowledged

Update

Alchemy has acknowledged this issue and commented: "We've documented our chain support. For this iteration, we will support EVM and Polygon zkEVM chains.

1. Based on our factory CREATE2 usage, we should be good for zkSync.
2. MSTORE and MLOAD are more complicated. It would be a larger project to be compatible here, and we will revisit in future iterations."

File(s) affected: `factory/*.sol`

Description: ZK-based EVM, such as zkSync or Scroll, implement their own version of Ethereum Virtual Machine and are EVM compatible but not to the full extent. There are some opcodes that are not supported or some opcodes that behave differently from what happens on the mainnet. If you plan to deploy on a ZK-based chain, it is important to take these into account, particularly the following opcodes in the case of zkSync:

1. `CREATE2`
    1. > To guarantee that `create` / `create2` functions operate correctly, the compiler must be aware of the bytecode of the deployed contract in advance
2. `MSTORE, MLOAD`
    1. > Unlike EVM, where the memory growth is in words, on zkEVM, the memory growth is counted in bytes. For example, if you write `mstore(100, 0)`, the `msize` on zkEVM will be `132`, but on the EVM, it will be `160`.

Recommendation: Take this into account if you plan to deploy on ZK EVM-based chains.

MSCA-29
Misuse of Functions in the `CastLib` Library Could Lead to Loss of Information

Informational · Fixed

> **Update**
> Documentation in the code has been added. Fixed in commit `851c6cff52f1ddb1691bff904371d31a08d195e8`.

File(s) affected: `libraries/CastLib.sol`

Description: The `LinkedListSet` and `AssociatedLinkedListSet` collections return values as a user-defined type `SetValue` when retrieving items. The underlying value of this type is always a `bytes30`, regardless of the actual data stored. To facilitate working with these values, the `CastLib` offers two functions: `toFunctionReferenceArray` and `toAddressArray`. These convert an array of `SetValue` into an array of function references or addresses, respectively. However, these methods do not verify if the underlying value corresponds to `FunctionReferences` (21 bytes long) or `Addresses` (20 bytes). Consequently, if an incorrect type is used, information may be lost and the returned value may be incorrect.

Recommendation: Consider including documentation in the code to inform the function callers that these functions do NOT verify if the given array of `SetValue` matches the type being cast to. The responsibility of ensuring the values are of the correct type and no information is lost falls on the caller. This information could be particularly useful if the code is refactored in the future.

MSCA-30
Potential Violation of ERC-4337 Storage Access Rules

Informational · Acknowledged

> **Update**
> Alchemy has acknowledged this issue and commented: "Based on both ERC-4337 and ERC-7562 wording, the allowed storage size should be bigger than 96 bytes. Will monitor closely to make sure our code is compatible with future changes."

File(s) affected: `libraries/AssociatedLinkedListSetLib.sol`, `libraries/PluginStorageLib.sol`

Description: The plugins rely heavily on the `AssociatedLinkedListSet` or `PluginStorageLib` libraries to calculate the storage slots associated with an MSCA. A slot is calculated as `keccak256(address || batchIndex || input1 || input2)`, while `input2` is optional. Therefore, the data appended to the `address` is 64 or 96 bytes long.

However, according to the latest ERC-4337 specification at the time of writing (commit `8dd085d`), the document does not explicitly specify the length of the appended data can or should be. It is mentioned that slots of type `keccak256(A || X) + n` are to cover mappings with the wallet address as the key, thereby implying that the acceptable range of `X` should at least include 32. (reference)

The draft EIP-7562 specifies that the appended data should be 32 bytes long (see point 5). On the other hand, the Infinitism bundler allows the data lengths between `[0, 480]` bytes. (reference)

Recommendation: In conclusion, currently, the libraries should work fine with the bundlers using the same or similar validation code (or geth custom tracer) as Infinitism's. However, if the 32-byte rule becomes strict, the libraries would technically violate the specification.

On the other hand, in theory, bundlers should be fine to accept the appended data, `X` with an arbitrary-long length, as it should be computationally infeasible to find two tuples `(A, X, n)` with different `A` such that `keccak256(A || X) + n` returns the same value, with the condition of `n` in `[0, 128]`.

MSCA-31
Different Order of Owners Can Produce Different MSCAs

Undetermined · Fixed

Update

The owner array is now ordered. Fixed in commit `b9df2828ec124969e20d8a7728579d7d8795b19e`.

File(s) affected: `factory/MultiOwnerMSCAFactory.sol`, `factory/MultiOwnerTokenReceiverMSCAFactory.sol`

Description: The `MultiOwnerMSCAFactory` uses `CREATE2` to generate the contract. As part of that call, the factory is passing the hash of a salt (provided by the caller) and the encoding of the list of owners. This means that if the owners are given in different orders, the addresses will be different. It is unclear if this is a problem or not

Recommendation: Determine whether it is acceptable to produce separate accounts if the list of owners is given in a different order. Otherwise, consider requiring the owners to be sorted and adding clarifying documentation.

MSCA-32
Plugin Metadata Can Be Misleading to End Users

Undetermined | Acknowledged

Update

Alchemy has acknowledged this issue and added to the document of risks.

Description: According to the code documentation, the `PluginMetadata` struct contains view information intended for front-end clients. However, it is critical to note that this data is not validated. Plugin authors can input any `name`, `version`, `author`, and `permissionDescriptors` they choose. Depending on the usage of this information, malicious developers could potentially mislead MSCA users into installing harmful plugins that resemble legitimate ones.

Recommendation: Account abstraction aims to simplify Ethereum use for non-crypto experts. This implies that some users of MSCAs might lack the expertise to identify safe plugins, potentially making them phishing targets. As such, it is advisable to provide documentation and off-chain mechanisms to assist end-users in selecting and recognizing legitimate plugins.

MSCA-33
Session Keys Require Explicitly Specifying a Spend Limit for Callable ERC-20 Contract

Undetermined | Acknowledged

Update

Alchemy has acknowledged this issue and commented: "Will not pursue, because it is incompatible with the fix for 5.2.1 Spearbit issue. Setting this as default would require every contract added to be "unset" as an ERC-20 in order to call anything other than `transfer` or `approve`."

File(s) affected: `plugins/session/permissions/SessionKeyPermissionsPlugin.sol`, `plugins/session/permissions/SessionKeyPermissionsLoupe.sol`

Description: If a session key's access control is configured to interact with an ERC-20 token contract, it could consume the MSCA's entire balance and allowances if a spending limit is not also specified along with the access. This is because contract calls are only checked for token transfers and allowances if `tokenContractData.isERC20WithSpendLimit` has specifically been set to `true`, which is only set if the spending limit is explicitly set to a value smaller than `type(uint256).max`. Not performing this explicit update, but enabling the session key to interact with the contract, will therefore act as an unlimited spend limit for that token.

Recommendation: `tokenContractData.isERC20WithSpendLimit` should be aligned with how `sessionKeyData.nativeTokenSpendLimitBypassed` behaves to make the user experience much more intuitive.

Definitions

- **High severity** – High-severity issues usually put a large number of users' sensitive information at risk, or are reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
- **Medium severity** – Medium-severity issues tend to put a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or are reasonably likely to lead to moderate financial impact.
- **Low severity** – The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
- **Informational** – The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
- **Undetermined** – The impact of the issue is uncertain.

- **Fixed** – Adjusted program implementation, requirements or constraints to eliminate the risk.
- **Mitigated** – Implemented actions to minimize the impact or likelihood of the risk.
- **Acknowledged** – The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).

Adherence to Best Practices

1. Fixed In `UpgradeableModularAccount.validateUserOp()`, the explicit check for `userOp.callData.length < 4` is redundant with the follow-up call to `_selectorFromCallData()`.
2. Fixed The gas-optimized call `unchecked {++i;}` as part of `for` loops would become unnecessary with an upgrade to Solidity 0.8.22.
3. Fixed The `UpgradeableModularAccount.executeFromPlugin()` and `executeFromPluginExternal()` functions slightly differ in their logic around the execution of post-only hooks. The former makes the call to `_doCachedPostHooks()` for the post-only hooks conditional depending on the `hasPostOnlyExecHooks` or `hasPostOnlyPermittedCallHooks` flag, assumingly for gas savings. Consider unifying the logic and adding the same check also for the `executeFromPluginExternal()` function. A `hasPreExecHooks` check is also missing, but we assumed that this is because the situation where there is no pre-execution hook for external contract calls is extremely unlikely.
4. Fixed `SessionKeyPermissionsPlugin.rotateKey()` currently recalculates the same `SessionKeyId` storage key three times for the update. Consider adding internal functions to the `SessionKeyPermissionsBase` contract that take the storage key as a parameter instead of recalculating it for the operations.
5. Acknowledged The `if` condition in `SessionKeyPlugin.sol#L170` should use the `isSessionKey()` function (which in turn should be set to `public` visibility).
6. Acknowledged The `BasePlugin.staysInitialized()` modifier remains unused, consider removing it.
7. Acknowledged `Address.isContract()` has been removed in the openzeppelin-contracts 5.0 release. Consider replacing it with `address(this).code.length > 0`.
8. Acknowledged The `UUPSUpgradeable` contract makes heavy use of assembly to enhance gas efficiency. Assembly blocks in this contract are marked with the `/// @solidity memory-safe-assembly` comment, indicating that these blocks adhere to Solidity's memory safety guidelines. However, it is important to note that future Solidity releases will no longer support this annotation, as specified in the official documentation. Given that the MSCA is using a modern version of the Solidity compiler (`^0.8.21`), consider annotating the assembly with the memory-safe dialect string (`assembly ("memory-safe")`).
9. Fixed Consider modifying `_doUserOpValidation()` so that it does not use `_EMPTY_FUNCTION_REFERENCE` directly to check if the function reference is not set but instead via a getter function similar to `isEmptyOrMagicValue()`.
10. Acknowledged Consider renaming the `key` variable in PluginStorageLib to e.g., `associatedStorageBuffer` for better code readability.
11. Acknowledged L529 of `UpgradeableModularAccount.sol` has commented out code.

Toolset

The notes below outline the setup and steps performed in the process of this audit.

Setup

Tool Setup:

- Slither v0.10.0

Steps taken to run the tools:

1. Install the Slither tool: `pip3 install slither-analyzer`
2. Run Slither from the project directory: `slither .`

Automated Analysis

Slither

138 results were found across the codebase. Non-false positive findings have been included in the report.

Test Suite Results

Tests were executed with `forge test`.There are a total of 360 tests of which all pass.

Fix Review Update

The test suite has been further improved. Total test cases stand at 417, of which all pass.

```
Running 2 tests for test/invariant/LLSLRepro.t.sol:LLSLReproTest
[PASS] test_repro_1() (gas: 101492)
[PASS] test_repro_2() (gas: 103744)
Test result: ok. 2 passed; 0 failed; 0 skipped; finished in 3.55ms

Running 2 tests for test/helpers/KnownSelectors.t.sol:KnownSelectorsTest
[PASS] test_isErc4337Function() (gas: 356)
[PASS] test_isNativeFunction() (gas: 132)
Test result: ok. 2 passed; 0 failed; 0 skipped; finished in 3.74ms

Running 5 tests for test/account/AccountReturnData.t.sol:AccountReturnDataTest
[PASS] test_returnData_execFromPlugin_execute() (gas: 46127)
[PASS] test_returnData_execFromPlugin_fallback() (gas: 36796)
[PASS] test_returnData_executeBatch() (gas: 41718)
[PASS] test_returnData_fallback() (gas: 19632)
[PASS] test_returnData_singular_execute() (gas: 34052)
Test result: ok. 5 passed; 0 failed; 0 skipped; finished in 4.82ms

Running 4 tests for test/comparison/CompareSimpleAccount.t.sol:CompareSimpleAccountTest
[PASS] test_SimpleAccount_deploy_basicSend() (gas: 269036)
[PASS] test_SimpleAccount_deploy_empty() (gas: 259670)
[PASS] test_SimpleAccount_postDeploy_basicSend() (gas: 111081)
[PASS] test_SimpleAccount_postDeploy_contractInteraction() (gas: 115650)
Test result: ok. 4 passed; 0 failed; 0 skipped; finished in 10.69ms

Running 30 tests for
test/account/AccountPermittedCallHooks.t.sol:UpgradeableModularAccountPermittedCallHooksTest
[PASS] test_overlappingPermittedCallHookPairsOnPost_install() (gas: 2146778)
[PASS] test_overlappingPermittedCallHookPairsOnPost_run() (gas: 2191896)
[PASS] test_overlappingPermittedCallHookPairsOnPost_uninstall() (gas: 2066824)
[PASS] test_overlappingPermittedCallHookPairsOnPreWithNullPost_install() (gas: 2019173)
[PASS] test_overlappingPermittedCallHookPairsOnPreWithNullPost_run() (gas: 2047742)
[PASS] test_overlappingPermittedCallHookPairsOnPreWithNullPost_uninstall() (gas: 1977038)
[PASS] test_overlappingPermittedCallHookPairsOnPreWithNullPre_install() (gas: 2063982)
[PASS] test_overlappingPermittedCallHookPairsOnPreWithNullPre_run() (gas: 2095224)
[PASS] test_overlappingPermittedCallHookPairsOnPreWithNullPre_uninstall() (gas: 1981031)
[PASS] test_overlappingPermittedCallHookPairsOnPre_install() (gas: 2103173)
[PASS] test_overlappingPermittedCallHookPairsOnPre_run() (gas: 2143255)
[PASS] test_overlappingPermittedCallHookPairsOnPre_uninstall() (gas: 2041681)
[PASS] test_overlappingPermittedCallHookPairs_install() (gas: 2080710)
[PASS] test_overlappingPermittedCallHookPairs_run() (gas: 2108619)
[PASS] test_overlappingPermittedCallHookPairs_uninstall() (gas: 2038131)
[PASS] test_overlappingPostPermittedCallHooks_install() (gas: 1913757)
[PASS] test_overlappingPostPermittedCallHooks_run() (gas: 1929021)
[PASS] test_overlappingPostPermittedCallHooks_uninstall() (gas: 1868814)
[PASS] test_overlappingPrePermittedCallHooks_install() (gas: 1914801)
[PASS] test_overlappingPrePermittedCallHooks_run() (gas: 1931790)
[PASS] test_overlappingPrePermittedCallHooks_uninstall() (gas: 1868474)
[PASS] test_permittedCallHookPair_install() (gas: 1879104)
[PASS] test_permittedCallHookPair_run() (gas: 1915770)
[PASS] test_permittedCallHookPair_uninstall() (gas: 1829039)
[PASS] test_postOnlyPermittedCallHook_install() (gas: 1773099)
[PASS] test_postOnlyPermittedCallHook_run() (gas: 1790701)
[PASS] test_postOnlyPermittedCallHook_uninstall() (gas: 1721165)
[PASS] test_prePermittedCallHook_install() (gas: 1773434)
[PASS] test_prePermittedCallHook_run() (gas: 1793851)
[PASS] test_prePermittedCallHook_uninstall() (gas: 1721183)
Test result: ok. 30 passed; 0 failed; 0 skipped; finished in 32.55ms

Running 2 tests for test/upgrade/LightAccountToMSCA.t.sol:LightAccountToMSCATest
[PASS] test_upgrade() (gas: 873948)
[PASS] test_verifySetup() (gas: 30571)
Test result: ok. 2 passed; 0 failed; 0 skipped; finished in 3.06ms

Running 3 tests for test/libraries/CountableLinkedListSetLib.t.sol:CountableLinkedListSetLibTest
[PASS] test_getCount() (gas: 45950)
[PASS] test_tryDecrement() (gas: 509227)
[PASS] test_tryIncrement() (gas: 359271)
Test result: ok. 3 passed; 0 failed; 0 skipped; finished in 6.16ms
```

```
Running 4 tests for test/libraries/AccountStorage.t.sol:AccountStorageTest
[PASS] testFuzz_permittedCallKey(address,bytes4) (runs: 500, μ: 657, ~: 657)
[PASS] test_storageSlotErc7201Formula() (gas: 531)
[PASS] test_storageSlotImpl() (gas: 7064)
[PASS] test_storageSlotProxy() (gas: 32060)
Test result: ok. 4 passed; 0 failed; 0 skipped; finished in 22.12ms

Running 24 tests for test/libraries/LinkedListSetLib.t.sol:LinkedListSetLibTest
[PASS] test_add_contains() (gas: 45221)
[PASS] test_add_remove_add() (gas: 49861)
[PASS] test_add_remove_add_empty() (gas: 49736)
[PASS] test_clear() (gas: 33100)
[PASS] test_empty() (gas: 4909)
[PASS] test_getAll() (gas: 68811)
[PASS] test_getAll2() (gas: 91888)
[PASS] test_getAll_empty() (gas: 2540)
[PASS] test_isSentinel() (gas: 305)
[PASS] test_remove() (gas: 33044)
[PASS] test_remove_empty() (gas: 2896)
[PASS] test_remove_nonexistent() (gas: 47433)
[PASS] test_remove_nonexistent_empty() (gas: 3008)
[PASS] test_remove_nonexistent_empty2() (gas: 47577)
[PASS] test_tryRemoveKnown1() (gas: 33135)
[PASS] test_tryRemoveKnown2() (gas: 53836)
[PASS] test_tryRemoveKnown_invalid1() (gas: 68399)
[PASS] test_userFlags_basic() (gas: 68567)
[PASS] test_userFlags_fail_does_not_contain() (gas: 70006)
[PASS] test_userFlags_flagsDisabled() (gas: 46468)
[PASS] test_userFlags_flagsEnabled() (gas: 46500)
[PASS] test_userFlags_getAll() (gas: 70289)
[PASS] test_userFlags_tryDisable() (gas: 45894)
[PASS] test_userFlags_tryEnable() (gas: 46277)
Test result: ok. 24 passed; 0 failed; 0 skipped; finished in 2.24ms

Running 2 tests for test/mocks/Counter.t.sol:CounterTest
[PASS] testIncrement() (gas: 28530)
[PASS] testSetNumber(uint256) (runs: 500, μ: 27383, ~: 28418)
Test result: ok. 2 passed; 0 failed; 0 skipped; finished in 26.36ms

Running 17 tests for test/libraries/AssociatedLinkedListSetLib.t.sol:AssociatedLinkedListSetLibTest
[PASS] test_add_contains() (gas: 47515)
[PASS] test_add_remove_add() (gas: 52184)
[PASS] test_add_remove_add_empty() (gas: 52060)
[PASS] test_clear() (gas: 35487)
[PASS] test_empty() (gas: 7322)
[PASS] test_getAll() (gas: 72178)
[PASS] test_getAll2() (gas: 95736)
[PASS] test_getAll_empty() (gas: 5061)
[PASS] test_no_address_collision() (gas: 50143)
[PASS] test_remove() (gas: 35110)
[PASS] test_remove_empty() (gas: 4989)
[PASS] test_remove_nonexistent() (gas: 50196)
[PASS] test_remove_nonexistent_empty() (gas: 5101)
[PASS] test_remove_nonexistent_empty2() (gas: 50296)
[PASS] test_tryRemoveKnown1() (gas: 35368)
[PASS] test_tryRemoveKnown2() (gas: 58549)
[PASS] test_tryRemoveKnown_invalid1() (gas: 71528)
Test result: ok. 17 passed; 0 failed; 0 skipped; finished in 677.46µs

Running 14 tests for test/account/ExecuteFromPluginPermissions.t.sol:ExecuteFromPluginPermissionsTest
[PASS] test_executeFromPluginAllowed() (gas: 52810)
[PASS] test_executeFromPluginExternal_Allowed_AllSelectors() (gas: 98093)
[PASS] test_executeFromPluginExternal_Allowed_AnyContract() (gas: 171417)
[PASS] test_executeFromPluginExternal_Allowed_IndividualSelectors() (gas: 85801)
[PASS] test_executeFromPluginExternal_Allowed_NativeTokenSpending() (gas: 122460)
[PASS] test_executeFromPluginExternal_NotAllowed_AllSelectors() (gas: 86811)
[PASS] test_executeFromPluginExternal_NotAllowed_NativeTokenSpending() (gas: 61657)
[PASS] test_executeFromPluginExternal_NotAlowed_IndividualSelectors() (gas: 107219)
[PASS] test_executeFromPluginExternal_PermittedCallHooks() (gas: 108348)
[PASS] test_executeFromPluginNotAllowed() (gas: 31726)
[PASS] test_executeFromPluginUnrecognizedFunction() (gas: 52671)
[PASS] test_executeFromPlugin_ExecutionHooks() (gas: 59803)
```

```
[PASS] test_executeFromPlugin_PermittedCallHooks() (gas: 98169)
[PASS] test_getPermissionsTestAddresses() (gas: 14500)
Test result: ok. 14 passed; 0 failed; 0 skipped; finished in 10.53ms

Running 14 tests for
test/plugin/session/permissions/SessionKeyNativeTokenSpendLimits.t.sol:SessionKeyNativeTokenSpendLimitsTe
st
[PASS] testFuzz_sessionKeyNativeTokenSpendLimits_setLimits(uint256,uint48,uint48) (runs: 500, μ: 98016,
~: 101394)
[PASS] test_sessionKeyNativeTokenSpendLimits_basic_multi() (gas: 415771)
[PASS] test_sessionKeyNativeTokenSpendLimits_basic_refreshInterval_takeMaxStartTime() (gas: 426596)
[PASS] test_sessionKeyNativeTokenSpendLimits_basic_single() (gas: 338387)
[PASS] test_sessionKeyNativeTokenSpendLimits_enforceLimit_none() (gas: 415941)
[PASS] test_sessionKeyNativeTokenSpendLimits_exceedLimit_multi() (gas: 190457)
[PASS] test_sessionKeyNativeTokenSpendLimits_exceedLimit_single() (gas: 338463)
[PASS] test_sessionKeyNativeTokenSpendLimits_multiUserOpBundle_check_noInterval() (gas: 395613)
[PASS] test_sessionKeyNativeTokenSpendLimits_overflowBatch() (gas: 170097)
[PASS] test_sessionKeyNativeTokenSpendLimits_overflowState() (gas: 332466)
[PASS] test_sessionKeyNativeTokenSpendLimits_refreshInterval_exceedNewInterval() (gas: 363543)
[PASS] test_sessionKeyNativeTokenSpendLimits_refreshInterval_multi() (gas: 519466)
[PASS] test_sessionKeyNativeTokenSpendLimits_refreshInterval_single() (gas: 461299)
[PASS] test_sessionKeyNativeTokenSpendLimits_validateSetUp() (gas: 35451)
Test result: ok. 14 passed; 0 failed; 0 skipped; finished in 109.75ms

Running 22 tests for
test/plugin/session/permissions/SessionKeyERC20SpendLimits.t.sol:SessionKeyERC20SpendLimitsTest
[PASS] testFuzz_sessionKeyERC20SpendLimits_setLimits(address,uint256,uint48,uint48) (runs: 500, μ:
118728, ~: 123088)
[PASS] test_executeWithSessionKey_approveOnlyCountsIncrease() (gas: 379804)
[PASS] test_executeWithSessionKey_failWithExceedLimit_multipleSpendFunctions() (gas: 356151)
[PASS] test_executeWithSessionKey_failWithExceedLimit_multipleTokens() (gas: 439871)
[PASS] test_executeWithSessionKey_failWithExceedLimit_multipleTransfer() (gas: 346425)
[PASS] test_executeWithSessionKey_failWithExceedLimit_overflow() (gas: 327337)
[PASS] test_executeWithSessionKey_refreshInterval_failWithSomeTokenLimit() (gas: 859173)
[PASS] test_executeWithSessionKey_refreshInterval_multipleApprove() (gas: 653492)
[PASS] test_executeWithSessionKey_refreshInterval_multipleSpendFunctions() (gas: 656236)
[PASS] test_executeWithSessionKey_refreshInterval_multipleTransfer() (gas: 627703)
[PASS] test_executeWithSessionKey_refreshInterval_singleTransfer() (gas: 585048)
[PASS] test_executeWithSessionKey_success_multipleApprove() (gas: 446546)
[PASS] test_executeWithSessionKey_success_multipleSpendFunctions() (gas: 467271)
[PASS] test_executeWithSessionKey_success_multipleTokens() (gas: 597096)
[PASS] test_executeWithSessionKey_success_multipleTransfer() (gas: 443527)
[PASS] test_sessionKeyERC20Limits_refreshInterval_failWithExceedNextLimit() (gas: 485022)
[PASS] test_sessionKeyERC20SpendLimits_basic_single() (gas: 377453)
[PASS] test_sessionKeyERC20SpendLimits_enforceLimit_none_basic() (gas: 401677)
[PASS] test_sessionKeyERC20SpendLimits_enforceLimit_none_batch() (gas: 357815)
[PASS] test_sessionKeyERC20SpendLimits_exceedLimit_single() (gas: 464280)
[PASS] test_sessionKeyERC20SpendLimits_tokenAddressZeroFails() (gas: 38945)
[PASS] test_sessionKeyERC20SpendLimits_validateSetUp() (gas: 35757)
Test result: ok. 22 passed; 0 failed; 0 skipped; finished in 173.92ms

Running 15 tests for
test/account/AccountPreValidationHooks.t.sol:UpgradeableModularAccountPreValidationHooksTest
[PASS] testFuzz_preRuntimeValidationHooks_revert(bytes) (runs: 500, μ: 1749120, ~: 1743332)
[PASS] test_overlappingPreRuntimeValidationHook_install() (gas: 3262630)
[PASS] test_overlappingPreRuntimeValidationHook_uninstall() (gas: 3189581)
[PASS] test_overlappingPreRuntimeValidationHooks_run() (gas: 3275979)
[PASS] test_overlappingPreUserOpValidationHooks_install() (gas: 3584570)
[PASS] test_overlappingPreUserOpValidationHooks_run() (gas: 3695598)
[PASS] test_overlappingPreUserOpValidationHooks_uninstall() (gas: 3533185)
[PASS] test_preRuntimeValidationHooks_install() (gas: 3151993)
[PASS] test_preRuntimeValidationHooks_revertAlwaysDeny() (gas: 1674534)
[PASS] test_preRuntimeValidationHooks_run() (gas: 3172528)
[PASS] test_preRuntimeValidationHooks_uninstall() (gas: 3121705)
[PASS] test_preUserOpValidationHooks_install() (gas: 3473709)
[PASS] test_preUserOpValidationHooks_revertAlwaysDeny() (gas: 2010671)
[PASS] test_preUserOpValidationHooks_run() (gas: 3598518)
[PASS] test_preUserOpValidationHooks_uninstall() (gas: 3466927)
Test result: ok. 15 passed; 0 failed; 0 skipped; finished in 532.14ms

Running 19 tests for test/account/UpgradeableModularAccount.t.sol:UpgradeableModularAccountTest
[PASS] testFuzz_runtime_revert(bytes) (runs: 500, μ: 34351, ~: 34221)
```

```
[PASS] test_basicUserOp() (gas: 964449)
[PASS] test_batchExecute() (gas: 166622)
[PASS] test_contractInteraction() (gas: 149041)
[PASS] test_debug_upgradeableModularAccount_storageAccesses() (gas: 176158)
[PASS] test_deployAccount() (gas: 809334)
[PASS] test_initialize_revertArrayLengthMismatch() (gas: 164637)
[PASS] test_postDeploy_ethSend() (gas: 149058)
[PASS] test_runtime_batchExecute() (gas: 63340)
[PASS] test_runtime_contractInteraction() (gas: 841616)
[PASS] test_runtime_debug_upgradeableModularAccount_storageAccesses() (gas: 46417)
[PASS] test_runtime_revertPluginCall() (gas: 849510)
[PASS] test_runtime_standardExecuteEthSend() (gas: 865884)
[PASS] test_standardExecuteEthSend() (gas: 976406)
[PASS] test_validateUserOp_revertFunctionMissing() (gas: 1312543)
[PASS] test_validateUserOp_revertNotFromEntryPoint() (gas: 30680)
[PASS] test_validateUserOp_revertUnrecognizedFunction() (gas: 30058)
[PASS] test_view_entryPoint() (gas: 815413)
[PASS] test_view_getNonce() (gas: 949522)
Test result: ok. 19 passed; 0 failed; 0 skipped; finished in 79.79ms

Running 26 tests for
test/account/UpgradeableModularAccountPluginManager.t.sol:UpgradeableModularAccountPluginManagerTest
[PASS] test_deployAccount() (gas: 809491)
[PASS] test_forceOnHookUnapply() (gas: 2032326)
[PASS] test_forceOnUninstall() (gas: 672939)
[PASS] test_injectHooks() (gas: 2775831)
[PASS] test_injectHooksApplyGoodCalldata() (gas: 2661512)
[PASS] test_injectHooksBadUninstallDependency() (gas: 2757680)
[PASS] test_injectHooksMissingPlugin() (gas: 1322854)
[PASS] test_injectHooksUnapplyBadCalldata() (gas: 2769001)
[PASS] test_injectHooksUnapplyGoodCalldata() (gas: 2630634)
[PASS] test_injectHooksUninstall() (gas: 2625188)
[PASS] test_installPlugin() (gas: 911868)
[PASS] test_installPlugin_ExecuteFromPlugin_PermittedExecSelectorNotInstalled() (gas: 1290873)
[PASS] test_installPlugin_alreadyInstalled() (gas: 409940)
[PASS] test_installPlugin_failWithNativeFunctionSelector() (gas: 1261301)
[PASS] test_installPlugin_failWtihErc4337FunctionSelector() (gas: 1261716)
[PASS] test_installPlugin_interfaceNotSupported() (gas: 62783)
[PASS] test_installPlugin_invalidManifest() (gas: 89426)
[PASS] test_installPlugin_missingDependency() (gas: 1838445)
[PASS] test_noNonSelfInstallAfterUninstall() (gas: 1040166)
[PASS] test_onHookUnapplyGasLimit() (gas: 2107122)
[PASS] test_onUninstallGasLimit() (gas: 863064)
[PASS] test_uninstallAndInstallInBatch() (gas: 1839282)
[PASS] test_uninstallPlugin_default() (gas: 1471109)
[PASS] test_uninstallPlugin_invalidManifestFails() (gas: 1789661)
[PASS] test_uninstallPlugin_manifestHasChanged() (gas: 4027874)
[PASS] test_uninstallPlugin_manifestParameter() (gas: 1467742)
Test result: ok. 26 passed; 0 failed; 0 skipped; finished in 26.39ms

Running 10 tests for test/account/ValidationIntersection.t.sol:ValidationIntersectionTest
[PASS] testFuzz_validationIntersect_single(uint256) (runs: 500, μ: 87621, ~: 88417)
[PASS] test_validationIntersect_authorizerAndTimeRange() (gas: 117070)
[PASS] test_validationIntersect_authorizer_sigfail_hook() (gas: 95077)
[PASS] test_validationIntersect_authorizer_sigfail_validationFunction() (gas: 95508)
[PASS] test_validationIntersect_multiplePreValidationHooksIntersect() (gas: 121832)
[PASS] test_validationIntersect_multiplePreValidationHooksSigFail() (gas: 101580)
[PASS] test_validationIntersect_revert_unexpectedAuthorizer() (gas: 64803)
[PASS] test_validationIntersect_timeBounds_intersect_1() (gas: 115133)
[PASS] test_validationIntersect_timeBounds_intersect_2() (gas: 115290)
[PASS] test_validationIntersect_validAuthorizer() (gas: 97244)
Test result: ok. 10 passed; 0 failed; 0 skipped; finished in 74.79ms

Running 8 tests for test/account/ManifestValidity.t.sol:ManifestValidityTest
[PASS] test_ManifestValidity_invalid_HookAlwaysDeny_PostExecHook() (gas: 630821)
[PASS] test_ManifestValidity_invalid_HookAlwaysDeny_RuntimeValidationFunction() (gas: 618338)
[PASS] test_ManifestValidity_invalid_HookAlwaysDeny_UserOpValidation() (gas: 616414)
[PASS] test_ManifestValidity_invalid_ValidationAlwaysAllow_PostExecHook() (gas: 631354)
[PASS] test_ManifestValidity_invalid_ValidationAlwaysAllow_PreExecHook() (gas: 630965)
[PASS] test_ManifestValidity_invalid_ValidationAlwaysAllow_PreRuntimeValidationHook() (gas: 662675)
[PASS] test_ManifestValidity_invalid_ValidationAlwaysAllow_PreUserOpValidationHook() (gas: 659525)
[PASS] test_ManifestValidity_invalid_ValidationAlwaysAllow_UserOpValidationFunction() (gas: 615937)
```

```
Test result: ok. 8 passed; 0 failed; 0 skipped; finished in 3.85ms

Running 8 tests for test/factory/MultiOwnerMSCAFactoryTest.t.sol:MultiOwnerMSCAFactoryTest
[PASS] test_addStake() (gas: 81063)
[PASS] test_addressMatch() (gas: 841988)
[PASS] test_deploy() (gas: 849078)
[PASS] test_deployCollision() (gas: 842462)
[PASS] test_deployedAccountHasCorrectPlugins() (gas: 839909)
[PASS] test_unlockStake() (gas: 87857)
[PASS] test_withdraw() (gas: 81003)
[PASS] test_withdrawStake() (gas: 78966)
Test result: ok. 8 passed; 0 failed; 0 skipped; finished in 3.75ms

Running 14 tests for
test/plugin/session/permissions/SessionKeyPermissionsPlugin.t.sol:SessionKeyPermissionsPluginTest
[PASS] testFuzz_sessionKeyPermissions_checkRequiredPaymaster(address,address) (runs: 500, μ: 174757, ~:
174983)
[PASS] testFuzz_sessionKeyPermissions_setRequiredPaymaster(address) (runs: 500, μ: 86445, ~: 86624)
[PASS] testFuzz_sessionKeyTimeRange(uint48,uint48) (runs: 500, μ: 119357, ~: 119440)
[PASS] test_rotateKey_basic() (gas: 403708)
[PASS] test_rotateKey_permissionsTransfer() (gas: 103750)
[PASS] test_sessionKeyPerms_requiredPaymaster_partialAddressFails() (gas: 131760)
[PASS] test_sessionKeyPerms_updatePermissions_invalidUpdates() (gas: 64425)
[PASS] test_sessionPerms_contractAllowList() (gas: 304896)
[PASS] test_sessionPerms_contractDefaultAllowList() (gas: 360439)
[PASS] test_sessionPerms_contractDenyList() (gas: 377734)
[PASS] test_sessionPerms_duplicateRegister() (gas: 36509)
[PASS] test_sessionPerms_selectorAllowList() (gas: 330957)
[PASS] test_sessionPerms_selectorDenyList() (gas: 398570)
[PASS] test_sessionPerms_validateSetUp() (gas: 15671)
Test result: ok. 14 passed; 0 failed; 0 skipped; finished in 730.48ms

Running 2 tests for test/upgrade/MSCAToMSCA.t.sol:MSCAToMSCATest
[PASS] test_sameStorageSlot_reinstallUpgradeToAndCall() (gas: 223412)
[PASS] test_sameStorageSlot_upgradeToAndCall() (gas: 125050)
Test result: ok. 2 passed; 0 failed; 0 skipped; finished in 2.98ms

Running 8 tests for test/plugin/TokenReceiverPlugin.t.sol:TokenReceiverPluginTest
[PASS] test_failERC1155Transfer() (gas: 184731)
[PASS] test_failERC721Transfer() (gas: 65933)
[PASS] test_failERC777Transfer() (gas: 23149)
[PASS] test_failIntrospection() (gas: 20073)
[PASS] test_passERC1155Transfer() (gas: 590620)
[PASS] test_passERC721Transfer() (gas: 456312)
[PASS] test_passERC777Transfer() (gas: 437740)
[PASS] test_passIntrospection() (gas: 404582)
Test result: ok. 8 passed; 0 failed; 0 skipped; finished in 8.58ms

Running 9 tests for
test/factory/MultiOwnerTokenReceiverFactoryTest.t.sol:MultiOwnerTokenReceiverMSCAFactoryTest
[PASS] test_addStake() (gas: 81063)
[PASS] test_addressMatch() (gas: 1181394)
[PASS] test_deploy() (gas: 1188415)
[PASS] test_deployCollision() (gas: 1181845)
[PASS] test_deployedAccountHasCorrectPlugins() (gas: 1182387)
[PASS] test_receiveTokens() (gas: 1464288)
[PASS] test_unlockStake() (gas: 87857)
[PASS] test_withdraw() (gas: 81033)
[PASS] test_withdrawStake() (gas: 79019)
Test result: ok. 9 passed; 0 failed; 0 skipped; finished in 5.46ms

Running 4 tests for test/plugin/owner/MultiOwnerPluginIntegration.t.sol:MultiOwnerPluginIntegration
[PASS] test_ownerPlugin_successInstallation() (gas: 59086)
[PASS] test_runtimeValidation_alwaysAllow_isValidSignature() (gas: 78067)
[PASS] test_runtimeValidation_ownerOrSelf_standardExecute() (gas: 84388)
[PASS] test_userOpValidation_owner_standardExecute() (gas: 224116)
Test result: ok. 4 passed; 0 failed; 0 skipped; finished in 5.12ms

Running 2 tests for test/libraries/PluginStorageLib.t.sol:PluginStorageLibTest
[PASS] testFuzz_storagePointer(address,uint256,bytes32,uint256[32]) (runs: 500, μ: 732622, ~: 755706)
[PASS] test_storagePointer() (gas: 52865)
Test result: ok. 2 passed; 0 failed; 0 skipped; finished in 147.55ms
```

```
Running 3 tests for test/invariant/LinkedListSetLibInvariants.t.sol:LinkedListSetLibInvariantsTest
[PASS] invariant_flagValidity() (runs: 500, calls: 5000, reverts: 0)
[PASS] invariant_getAllEquivalence() (runs: 500, calls: 5000, reverts: 0)
[PASS] invariant_shouldContain() (runs: 500, calls: 5000, reverts: 0)
Test result: ok. 3 passed; 0 failed; 0 skipped; finished in 1.55s

Running 10 tests for
test/plugin/session/SessionKeyPluginWithMultiOwner.t.sol:SessionKeyPluginWithMultiOwnerTest
[PASS] testFuzz_sessionKey_invalidFunctionId(uint8,
(address,uint256,bytes,bytes,uint256,uint256,uint256,uint256,uint256,bytes,bytes)) (runs: 500, μ: 22521,
~: 22347)
[PASS] testFuzz_sessionKey_userOpValidation_invalid(uint8,uint64) (runs: 500, μ: 283586, ~: 266158)
[PASS] testFuzz_sessionKey_userOpValidation_valid(uint16) (runs: 500, μ: 297255, ~: 290394)
[PASS] test_sessionKey_addAndRemoveKeys() (gas: 103357)
[PASS] test_sessionKey_addKeySuccess() (gas: 89305)
[PASS] test_sessionKey_doesNotContainSentinelValue() (gas: 85476)
[PASS] test_sessionKey_getPredecessor_address() (gas: 108772)
[PASS] test_sessionKey_getPredecessor_missing() (gas: 86419)
[PASS] test_sessionKey_getPredecessor_sentinel() (gas: 109508)
[PASS] test_sessionKey_useSessionKey() (gas: 230081)
Test result: ok. 10 passed; 0 failed; 0 skipped; finished in 1.57s

Running 14 tests for test/plugin/owner/MultiOwnerPlugin.t.sol:MultiOwnerPluginTest
[PASS] testFuzz_isValidSignature_ContractOwner(bytes32) (runs: 500, μ: 63061, ~: 63061)
[PASS] testFuzz_isValidSignature_EOAOwner(string,bytes32) (runs: 500, μ: 67223, ~: 67150)
[PASS]
testFuzz_userOpValidationFunction_ContractOwner((address,uint256,bytes,bytes,uint256,uint256,uint256,uint
256,uint256,bytes,bytes)) (runs: 500, μ: 88223, ~: 88173)
[PASS] testFuzz_userOpValidationFunction_EOAOwner(string,
(address,uint256,bytes,bytes,uint256,uint256,uint256,uint256,uint256,bytes,bytes)) (runs: 500, μ: 77234,
~: 77252)
[PASS] test_eip712Domain() (gas: 28153)
[PASS] test_multiOwnerPlugin_sentinelIsNotOwner() (gas: 8395)
[PASS] test_onInstall_success() (gas: 68603)
[PASS] test_onUninstall_success() (gas: 24198)
[PASS] test_pluginInitializeGuards() (gas: 49455)
[PASS] test_pluginManifest() (gas: 41048)
[PASS] test_runtimeValidationFunction_OwnerOrSelf() (gas: 19956)
[PASS] test_updateOwners_failWithEmptyOwners() (gas: 30553)
[PASS] test_updateOwners_failWithNotExist() (gas: 17843)
[PASS] test_updateOwners_success() (gas: 66370)
Test result: ok. 14 passed; 0 failed; 0 skipped; finished in 1.21s

Running 34 tests for test/account/AccountExecHooks.t.sol:UpgradeableModularAccountExecHooksTest
[PASS] testFuzz_preExecHook_revertData(bytes) (runs: 500, μ: 1770032, ~: 1764244)
[PASS] test_execHookPair_install() (gas: 1812830)
[PASS] test_execHookPair_run() (gas: 1848823)
[PASS] test_execHookPair_uninstall() (gas: 1781349)
[PASS] test_execHooksWithPostOnlyForNativeFunction_install() (gas: 3471320)
[PASS] test_execHooksWithPostOnlyForNativeFunction_run() (gas: 3509384)
[PASS] test_overlappingExecHookPairsOnPostWithNullPre_install() (gas: 3449086)
[PASS] test_overlappingExecHookPairsOnPostWithNullPre_run() (gas: 3480260)
[PASS] test_overlappingExecHookPairsOnPostWithNullPre_uninstall() (gas: 3360391)
[PASS] test_overlappingExecHookPairsOnPost_install() (gas: 3533038)
[PASS] test_overlappingExecHookPairsOnPost_run() (gas: 3576508)
[PASS] test_overlappingExecHookPairsOnPost_uninstall() (gas: 3448796)
[PASS] test_overlappingExecHookPairsOnPreWithNullPost_install() (gas: 3405949)
[PASS] test_overlappingExecHookPairsOnPreWithNullPost_run() (gas: 3432624)
[PASS] test_overlappingExecHookPairsOnPreWithNullPost_uninstall() (gas: 3358147)
[PASS] test_overlappingExecHookPairsOnPre_install() (gas: 3488110)
[PASS] test_overlappingExecHookPairsOnPre_run() (gas: 3528635)
[PASS] test_overlappingExecHookPairsOnPre_uninstall() (gas: 3422990)
[PASS] test_overlappingExecHookPairs_install() (gas: 3537046)
[PASS] test_overlappingExecHookPairs_run() (gas: 3564667)
[PASS] test_overlappingExecHookPairs_uninstall() (gas: 3472232)
[PASS] test_overlappingPostExecHooks_install() (gas: 3299729)
[PASS] test_overlappingPostExecHooks_run() (gas: 3313752)
[PASS] test_overlappingPostExecHooks_uninstall() (gas: 3237137)
[PASS] test_overlappingPreExecHooks_install() (gas: 3300707)
[PASS] test_overlappingPreExecHooks_run() (gas: 3316023)
[PASS] test_overlappingPreExecHooks_uninstall() (gas: 3239602)
```

```
[PASS] test_postOnlyExecHook_install() (gas: 1706967)
[PASS] test_postOnlyExecHook_run() (gas: 1722237)
[PASS] test_postOnlyExecHook_uninstall() (gas: 1673727)
[PASS] test_preExecHook_install() (gas: 1707522)
[PASS] test_preExecHook_revertAlwaysDeny() (gas: 1695168)
[PASS] test_preExecHook_run() (gas: 1726299)
[PASS] test_preExecHook_uninstall() (gas: 1673107)
Test result: ok. 34 passed; 0 failed; 0 skipped; finished in 1.80s

Running 3 tests for
test/invariant/AssociatedLinkedListSetLibInvariants.t.sol:AssociatedLinkedListSetLibInvariantsTest
[PASS] invariant_flagValidity() (runs: 500, calls: 5000, reverts: 0)
[PASS] invariant_getAllEquivalence() (runs: 500, calls: 5000, reverts: 0)
[PASS] invariant_shouldContain() (runs: 500, calls: 5000, reverts: 0)
Test result: ok. 3 passed; 0 failed; 0 skipped; finished in 1.89s

Running 8 tests for test/account/AccountLoupe.t.sol:AccountLoupeTest
[PASS] test_pluginLoupe_getExecutionFunctionConfig_native() (gas: 55490)
[PASS] test_pluginLoupe_getExecutionFunctionConfig_plugin() (gas: 40208)
[PASS] test_pluginLoupe_getExecutionHooks() (gas: 28044)
[PASS] test_pluginLoupe_getExecutionHooks_overlapping() (gas: 2408070)
[PASS] test_pluginLoupe_getHooks_multiple() (gas: 1853076)
[PASS] test_pluginLoupe_getInstalledPlugins_initial() (gas: 21706)
[PASS] test_pluginLoupe_getPermittedCallHooks() (gas: 27952)
[PASS] test_pluginLoupe_getPreValidationHooks() (gas: 26186)
Test result: ok. 8 passed; 0 failed; 0 skipped; finished in 1.92s

Running 2 tests for test/libraries/FunctionReferenceLib.t.sol:FunctionReferenceLibTest
[PASS] testFuzz_functionReference_operators(bytes21,bytes21) (runs: 500, μ: 367, ~: 338)
[PASS] testFuzz_functionReference_packing(address,uint8) (runs: 500, μ: 523, ~: 523)
Test result: ok. 2 passed; 0 failed; 0 skipped; finished in 1.99s

Running 16 tests for test/plugin/session/permissions/SessionKeyGasLimits.t.sol:SessionKeyGasLimitsTest
[PASS] testFuzz_sessionKeyGasLimits_nolimit(uint256) (runs: 500, μ: 197152, ~: 197061)
[PASS] testFuzz_sessionKeyGasLimits_requireNonceAsAddress(uint192) (runs: 500, μ: 202686, ~: 202686)
[PASS] testFuzz_sessionKeyGasLimits_setLimits(uint256,uint48,uint48) (runs: 500, μ: 100818, ~: 104239)
[PASS] test_sessionKeyGasLimits_enforceLimit_basic_multipleInBundle() (gas: 354557)
[PASS] test_sessionKeyGasLimits_enforceLimit_basic_single() (gas: 306969)
[PASS] test_sessionKeyGasLimits_enforceLimit_none() (gas: 182140)
[PASS] test_sessionKeyGasLimits_exceedLimit_multipleInBundle() (gas: 257405)
[PASS] test_sessionKeyGasLimits_exceedLimit_single() (gas: 202088)
[PASS] test_sessionKeyGasLimits_refreshInterval_inspectValidationData() (gas: 335353)
[PASS] test_sessionKeyGasLimits_refreshInterval_multipleInBundle() (gas: 508766)
[PASS] test_sessionKeyGasLimits_refreshInterval_multipleInBundle_tryExceedFails() (gas: 455789)
[PASS] test_sessionKeyGasLimits_refreshInterval_resetFlagTracking() (gas: 374661)
[PASS] test_sessionKeyGasLimits_refreshInterval_resetFlag_fixWithExtraUO() (gas: 452554)
[PASS] test_sessionKeyGasLimits_refreshInterval_resetFlag_fixWithOwnerReset() (gas: 393086)
[PASS] test_sessionKeyGasLimits_refreshInterval_resetFlag_fixWithPublicReset() (gas: 381003)
[PASS] test_sessionKeyGasLimits_refreshInterval_single() (gas: 405760)
Test result: ok. 16 passed; 0 failed; 0 skipped; finished in 1.68s

Ran 34 test suites: 360 tests passed, 0 failed, 0 skipped (360 total tests)

**Fix Review Update**

Running 3 tests for test/helpers/KnownSelectors.t.sol:KnownSelectorsTest
[PASS] test_isErc4337Function() (gas: 378)
[PASS] test_isIPluginFunction() (gas: 154)
[PASS] test_isNativeFunction() (gas: 132)
Test result: ok. 3 passed; 0 failed; 0 skipped; finished in 11.48ms

Running 24 tests for test/libraries/LinkedListSetLib.t.sol:LinkedListSetLibTest
[PASS] test_add_contains() (gas: 45155)
[PASS] test_add_remove_add() (gas: 49808)
[PASS] test_add_remove_add_empty() (gas: 49684)
[PASS] test_clear() (gas: 32914)
[PASS] test_empty() (gas: 4843)
[PASS] test_getAll() (gas: 68791)
[PASS] test_getAll2() (gas: 91877)
[PASS] test_getAll_empty() (gas: 2565)
[PASS] test_isSentinel() (gas: 305)
[PASS] test_remove() (gas: 33017)
```

```
[PASS] test_remove_empty() (gas: 2896)
[PASS] test_remove_nonexistent() (gas: 47400)
[PASS] test_remove_nonexistent_empty() (gas: 3008)
[PASS] test_remove_nonexistent_empty2() (gas: 47544)
[PASS] test_tryRemoveKnown1() (gas: 33048)
[PASS] test_tryRemoveKnown2() (gas: 53780)
[PASS] test_tryRemoveKnown_invalid1() (gas: 68201)
[PASS] test_userFlags_basic() (gas: 68402)
[PASS] test_userFlags_fail_does_not_contain() (gas: 69940)
[PASS] test_userFlags_flagsDisabled() (gas: 46402)
[PASS] test_userFlags_flagsEnabled() (gas: 46434)
[PASS] test_userFlags_getAll() (gas: 70203)
[PASS] test_userFlags_tryDisable() (gas: 45729)
[PASS] test_userFlags_tryEnable() (gas: 46112)
Test result: ok. 24 passed; 0 failed; 0 skipped; finished in 13.14ms

Running 5 tests for test/account/AccountReturnData.t.sol:AccountReturnDataTest
[PASS] test_returnData_execFromPlugin_execute() (gas: 37080)
[PASS] test_returnData_execFromPlugin_fallback() (gas: 38636)
[PASS] test_returnData_executeBatch() (gas: 38134)
[PASS] test_returnData_fallback() (gas: 20336)
[PASS] test_returnData_singular_execute() (gas: 30680)
Test result: ok. 5 passed; 0 failed; 0 skipped; finished in 13.04ms

Running 3 tests for test/libraries/CountableLinkedListSetLib.t.sol:CountableLinkedListSetLibTest
[PASS] test_getCount() (gas: 45785)
[PASS] test_tryDecrement() (gas: 440530)
[PASS] test_tryIncrement() (gas: 313554)
Test result: ok. 3 passed; 0 failed; 0 skipped; finished in 13.99ms

Running 13 tests for test/account/ExecuteFromPluginPermissions.t.sol:ExecuteFromPluginPermissionsTest
[PASS] test_executeFromPluginAllowed() (gas: 52405)
[PASS] test_executeFromPluginExternal_Allowed_AllSelectors() (gas: 83611)
[PASS] test_executeFromPluginExternal_Allowed_AnyContract() (gas: 149638)
[PASS] test_executeFromPluginExternal_Allowed_AnyContractButSelf() (gas: 102666)
[PASS] test_executeFromPluginExternal_Allowed_IndividualSelectors() (gas: 73790)
[PASS] test_executeFromPluginExternal_Allowed_NativeTokenSpending() (gas: 107041)
[PASS] test_executeFromPluginExternal_NotAllowed_AllSelectors() (gas: 88809)
[PASS] test_executeFromPluginExternal_NotAllowed_NativeTokenSpending() (gas: 62696)
[PASS] test_executeFromPluginExternal_NotAlowed_IndividualSelectors() (gas: 95396)
[PASS] test_executeFromPluginNotAllowed() (gas: 32221)
[PASS] test_executeFromPluginUnrecognizedFunction() (gas: 54848)
[PASS] test_executeFromPlugin_ExecutionHooks() (gas: 59285)
[PASS] test_getPermissionsTestAddresses() (gas: 14284)
Logs:
  counter1 address: 0x5615dEB798BB3E4dFa0139dFa1b3D433Cc23b72f
  counter2 address: 0x2e234DAe75C793f67A35089C9d99245E1C58470b
  counter3 address: 0xF62849F9A0B5Bf2913b396098F7c7019b51A820a
  resultCreatorPlugin address: 0x5991A2dF15A8F6A256D3Ec51E99254Cd3fb576A9

Test result: ok. 13 passed; 0 failed; 0 skipped; finished in 19.96ms

Running 24 tests for test/account/phases/AccountStatePhasesExec.t.sol:AccountStatePhasesUOValidationTest
[PASS] test_ASP_exec_add_postExec_associated_firstElement() (gas: 1875714)
[PASS] test_ASP_exec_add_postExec_associated_notFirstElement() (gas: 1982909)
[PASS] test_ASP_exec_add_postExec_firstElement() (gas: 1771042)
[PASS] test_ASP_exec_add_postExec_notFirstElement() (gas: 1827483)
[PASS] test_ASP_exec_remove_postExec_associated_firstElement() (gas: 1845272)
[PASS] test_ASP_exec_remove_postExec_associated_notFirstElement() (gas: 1951917)
[PASS] test_ASP_exec_remove_postExec_firstElement() (gas: 1750578)
[PASS] test_ASP_exec_remove_postExec_notFirstElement() (gas: 1805947)
[PASS] test_ASP_postExec_add_postExec_associated_notFirstElement() (gas: 1983062)
[PASS] test_ASP_postExec_add_postExec_firstElement() (gas: 1900059)
[PASS] test_ASP_postExec_add_postExec_notFirstElement() (gas: 1826932)
[PASS] test_ASP_postExec_remove_postExec_associated_notFirstElement() (gas: 1951982)
[PASS] test_ASP_postExec_remove_postExec_firstElement() (gas: 1879382)
[PASS] test_ASP_postExec_remove_postExec_notFirstElement() (gas: 1806187)
[PASS] test_ASP_preExec_add_postExec_associated_notFirstElement() (gas: 1931360)
[PASS] test_ASP_preExec_add_postExec_firstElement() (gas: 1847763)
[PASS] test_ASP_preExec_add_postExec_notFirstElement() (gas: 3210482)
[PASS] test_ASP_preExec_add_preExec_notFirstElement() (gas: 1826410)
[PASS] test_ASP_preExec_remove_exec() (gas: 309472)
```

```
[PASS] test_ASP_preExec_remove_postExec_associated_notFirstElement() (gas: 1900427)
[PASS] test_ASP_preExec_remove_postExec_firstElement() (gas: 1828069)
[PASS] test_ASP_preExec_remove_postExec_notFirstElement() (gas: 3137883)
[PASS] test_ASP_preExec_remove_preExec() (gas: 1806544)
[PASS] test_ASP_preExec_replace_exec() (gas: 2042730)
Test result: ok. 24 passed; 0 failed; 0 skipped; finished in 40.79ms

Running 2 tests for test/invariant/LLSLRepro.t.sol:LLSLReproTest
[PASS] test_repro_1() (gas: 101463)
[PASS] test_repro_2() (gas: 102909)
Test result: ok. 2 passed; 0 failed; 0 skipped; finished in 754.21µs

Running 2 tests for test/upgrade/LightAccountToMSCA.t.sol:LightAccountToMSCATest
[PASS] test_upgrade() (gas: 760600)
[PASS] test_verifySetup() (gas: 30117)
Test result: ok. 2 passed; 0 failed; 0 skipped; finished in 3.43ms

Running 15 tests for test/factory/MultiOwnerMSCAFactoryTest.t.sol:MultiOwnerMSCAFactoryTest
[PASS] test_2StepOwnershipTransfer() (gas: 32477)
[PASS] test_addStake() (gas: 80798)
[PASS] test_addressMatch() (gas: 746838)
[PASS] test_badOwnersArray() (gas: 14300)
[PASS] test_deploy() (gas: 751718)
[PASS] test_deployCollision() (gas: 747167)
[PASS] test_deployWithDuplicateOwners() (gas: 10161)
[PASS] test_deployWithUnsortedOwners() (gas: 10403)
[PASS] test_deployedAccountHasCorrectPlugins() (gas: 743599)
[PASS] test_getAddressWithMaxOwnersAndDeploy() (gas: 3405056)
[PASS] test_getAddressWithTooManyOwners() (gas: 257147)
[PASS] test_getAddressWithUnsortedOwners() (gas: 10536)
[PASS] test_unlockStake() (gas: 87497)
[PASS] test_withdraw() (gas: 80967)
[PASS] test_withdrawStake() (gas: 78687)
Test result: ok. 15 passed; 0 failed; 0 skipped; finished in 7.06ms

Running 2 tests for test/helpers/FunctionReferenceLib.t.sol:FunctionReferenceLibTest
[PASS] testFuzz_functionReference_operators(bytes21,bytes21) (runs: 500, µ: 361, ~: 335)
[PASS] testFuzz_functionReference_packing(address,uint8) (runs: 500, µ: 511, ~: 511)
Test result: ok. 2 passed; 0 failed; 0 skipped; finished in 101.27ms

Running 2 tests for test/mocks/Counter.t.sol:CounterTest
[PASS] testIncrement() (gas: 28494)
[PASS] testSetNumber(uint256) (runs: 500, µ: 27426, ~: 28382)
Test result: ok. 2 passed; 0 failed; 0 skipped; finished in 71.84ms

Running 4 tests for test/comparison/CompareSimpleAccount.t.sol:CompareSimpleAccountTest
[PASS] test_SimpleAccount_deploy_basicSend() (gas: 272388)
[PASS] test_SimpleAccount_deploy_empty() (gas: 263061)
[PASS] test_SimpleAccount_postDeploy_basicSend() (gas: 110112)
[PASS] test_SimpleAccount_postDeploy_contractInteraction() (gas: 114627)
Test result: ok. 4 passed; 0 failed; 0 skipped; finished in 4.98ms

Running 30 tests for
test/account/phases/AccountStatePhasesRTValidation.t.sol:AccountStatePhasesRTValidationTest
[PASS] test_ASP_RTValidation_add_postExec_associated_firstElement() (gas: 1937870)
[PASS] test_ASP_RTValidation_add_postExec_associated_notFirstElement() (gas: 1972512)
[PASS] test_ASP_RTValidation_add_postExec_firstElement() (gas: 1824933)
[PASS] test_ASP_RTValidation_add_postExec_notFirstElement() (gas: 1861122)
[PASS] test_ASP_RTValidation_add_preExec_firstElement() (gas: 1825691)
[PASS] test_ASP_RTValidation_add_preExec_notFirstElement() (gas: 1861908)
[PASS] test_ASP_RTValidation_remove_exec() (gas: 356813)
[PASS] test_ASP_RTValidation_remove_postExec_associated_firstElement() (gas: 1881157)
[PASS] test_ASP_RTValidation_remove_postExec_associated_notFirstElement() (gas: 1917200)
[PASS] test_ASP_RTValidation_remove_postExec_firstElement() (gas: 1794398)
[PASS] test_ASP_RTValidation_remove_postExec_notFirstElement() (gas: 1831084)
[PASS] test_ASP_RTValidation_remove_preExec() (gas: 1793887)
[PASS] test_ASP_RTValidation_replace_exec() (gas: 2010309)
[PASS] test_ASP_preRTValidation_add_postExec_associated_firstElement() (gas: 1991779)
[PASS] test_ASP_preRTValidation_add_postExec_associated_notFirstElement() (gas: 2026602)
[PASS] test_ASP_preRTValidation_add_postExec_firstElement() (gas: 1879392)
[PASS] test_ASP_preRTValidation_add_postExec_notFirstElement() (gas: 1914860)
[PASS] test_ASP_preRTValidation_add_preExec_firstElement() (gas: 1879820)
```

```
[PASS] test_ASP_preRTValidation_add_preExec_notFirstElement() (gas: 1915823)
[PASS] test_ASP_preRTValidation_add_preRTValidation() (gas: 1836863)
[PASS] test_ASP_preRTValidation_remove_RTValidation() (gas: 2048261)
[PASS] test_ASP_preRTValidation_remove_exec() (gas: 415414)
[PASS] test_ASP_preRTValidation_remove_postExec_associated_firstElement() (gas: 1935726)
[PASS] test_ASP_preRTValidation_remove_postExec_associated_notFirstElement() (gas: 1971158)
[PASS] test_ASP_preRTValidation_remove_postExec_firstElement() (gas: 1848813)
[PASS] test_ASP_preRTValidation_remove_postExec_notFirstElement() (gas: 1885240)
[PASS] test_ASP_preRTValidation_remove_preExec() (gas: 1848830)
[PASS] test_ASP_preRTValidation_remove_preRTValidation() (gas: 1814039)
[PASS] test_ASP_preRTValidation_replace_RTValidation() (gas: 2231870)
[PASS] test_ASP_preRTValidation_replace_exec() (gas: 2049402)
Test result: ok. 30 passed; 0 failed; 0 skipped; finished in 79.39ms

Running 4 tests for test/plugin/owner/MultiOwnerPluginIntegration.t.sol:MultiOwnerPluginIntegration
[PASS] test_ownerPlugin_successInstallation() (gas: 36850)
[PASS] test_runtimeValidation_alwaysAllow_isValidSignature() (gas: 96974)
[PASS] test_runtimeValidation_ownerOrSelf_standardExecute() (gas: 80562)
[PASS] test_userOpValidation_owner_standardExecute() (gas: 229657)
Test result: ok. 4 passed; 0 failed; 0 skipped; finished in 7.56ms

Running 15 tests for
test/factory/MultiOwnerTokenReceiverFactoryTest.t.sol:MultiOwnerTokenReceiverMSCAFactoryTest
[PASS] test_2StepOwnershipTransfer() (gas: 32556)
[PASS] test_addStake() (gas: 80826)
[PASS] test_addressMatch() (gas: 1082504)
[PASS] test_deploy() (gas: 1087313)
[PASS] test_deployCollision() (gas: 1082840)
[PASS] test_deployWithDuplicateOwners() (gas: 10193)
[PASS] test_deployWithUnsortedOwners() (gas: 10457)
[PASS] test_deployedAccountHasCorrectPlugins() (gas: 1082298)
[PASS] test_getAddressWithMaxOwnersAndDeploy() (gas: 3740274)
[PASS] test_getAddressWithTooManyOwners() (gas: 257160)
[PASS] test_getAddressWithUnsortedOwners() (gas: 10621)
[PASS] test_receiveTokens() (gas: 1365684)
[PASS] test_unlockStake() (gas: 87540)
[PASS] test_withdraw() (gas: 81026)
[PASS] test_withdrawStake() (gas: 78775)
Test result: ok. 15 passed; 0 failed; 0 skipped; finished in 16.19ms

Running 30 tests for
test/account/phases/AccountStatePhasesUOValidation.t.sol:AccountStatePhasesUOValidationTest
[PASS] test_ASP_UOValidation_add_postExec_associated_firstElement() (gas: 2063660)
[PASS] test_ASP_UOValidation_add_postExec_associated_notFirstElement() (gas: 2099110)
[PASS] test_ASP_UOValidation_add_postExec_firstElement() (gas: 1950943)
[PASS] test_ASP_UOValidation_add_postExec_notFirstElement() (gas: 1987855)
[PASS] test_ASP_UOValidation_add_preExec_firstElement() (gas: 1952757)
[PASS] test_ASP_UOValidation_add_preExec_notFirstElement() (gas: 1988685)
[PASS] test_ASP_UOValidation_remove_exec() (gas: 403056)
[PASS] test_ASP_UOValidation_remove_postExec_associated_firstElement() (gas: 2008385)
[PASS] test_ASP_UOValidation_remove_postExec_associated_notFirstElement() (gas: 2044444)
[PASS] test_ASP_UOValidation_remove_postExec_firstElement() (gas: 1920834)
[PASS] test_ASP_UOValidation_remove_postExec_notFirstElement() (gas: 1957539)
[PASS] test_ASP_UOValidation_remove_preExec() (gas: 1920455)
[PASS] test_ASP_UOValidation_replace_exec() (gas: 2136987)
[PASS] test_ASP_preUOValidation_add_postExec_associated_firstElement() (gas: 2118579)
[PASS] test_ASP_preUOValidation_add_postExec_associated_notFirstElement() (gas: 2154275)
[PASS] test_ASP_preUOValidation_add_postExec_firstElement() (gas: 2006389)
[PASS] test_ASP_preUOValidation_add_postExec_notFirstElement() (gas: 2042910)
[PASS] test_ASP_preUOValidation_add_preExec_firstElement() (gas: 2007477)
[PASS] test_ASP_preUOValidation_add_preExec_notFirstElement() (gas: 2043982)
[PASS] test_ASP_preUOValidation_add_preUOValidation() (gas: 1964262)
[PASS] test_ASP_preUOValidation_remove_UOValidation() (gas: 451039)
[PASS] test_ASP_preUOValidation_remove_exec() (gas: 450370)
[PASS] test_ASP_preUOValidation_remove_postExec_associated_firstElement() (gas: 2063674)
[PASS] test_ASP_preUOValidation_remove_postExec_associated_notFirstElement() (gas: 2099364)
[PASS] test_ASP_preUOValidation_remove_postExec_firstElement() (gas: 1976299)
[PASS] test_ASP_preUOValidation_remove_postExec_notFirstElement() (gas: 2012635)
[PASS] test_ASP_preUOValidation_remove_preExec() (gas: 1976624)
[PASS] test_ASP_preUOValidation_remove_preUOValidation() (gas: 1946183)
[PASS] test_ASP_preUOValidation_replace_UOValidation() (gas: 2253383)
[PASS] test_ASP_preUOValidation_replace_exec() (gas: 2177520)
```

```
Test result: ok. 30 passed; 0 failed; 0 skipped; finished in 68.12ms

Running 19 tests for test/account/UpgradeableModularAccount.t.sol:UpgradeableModularAccountTest
[PASS] testFuzz_runtime_revert(bytes) (runs: 500, μ: 33972, ~: 33827)
[PASS] test_basicUserOp() (gas: 866735)
[PASS] test_batchExecute() (gas: 162052)
[PASS] test_contractInteraction() (gas: 144580)
[PASS] test_debug_upgradeableModularAccount_storageAccesses() (gas: 166082)
Logs:
  read: 0x360894a13ba1a3210667c828492db98dca3e2076cc3735a920a3ca505d382bbc val:
0x000000000000000000000000c7183455a4c133ae270771860664b6b7ec320bb1
  read: 0x7439823b81eb7fa62221b59ceaad33e97f63c5f36069f86d0d08ae0908f689b8 val:
0x0000000000000000000000002e234dae75c793f67a35089c9d99245e1c58470b01
  read: 0x360894a13ba1a3210667c828492db98dca3e2076cc3735a920a3ca505d382bbc val:
0x000000000000000000000000c7183455a4c133ae270771860664b6b7ec320bb1
  read: 0x7439823b81eb7fa62221b59ceaad33e97f63c5f36069f86d0d08ae0908f689b7 val:
0x0000000000000000000000000000000000000000000000000000000000000000

[PASS] test_deployAccount() (gas: 712355)
[PASS] test_initialize_revertArrayLengthMismatch() (gas: 167533)
[PASS] test_postDeploy_ethSend() (gas: 144600)
[PASS] test_runtime_batchExecute() (gas: 59777)
[PASS] test_runtime_contractInteraction() (gas: 741241)
[PASS] test_runtime_debug_upgradeableModularAccount_storageAccesses() (gas: 42926)
[PASS] test_runtime_revertPluginCall() (gas: 751211)
[PASS] test_runtime_standardExecuteEthSend() (gas: 765434)
[PASS] test_standardExecuteEthSend() (gas: 874862)
[PASS] test_validateUserOp_revertFunctionMissing() (gas: 1302585)
[PASS] test_validateUserOp_revertNotFromEntryPoint() (gas: 30297)
[PASS] test_validateUserOp_revertUnrecognizedFunction() (gas: 29480)
[PASS] test_view_entryPoint() (gas: 718307)
[PASS] test_view_getNonce() (gas: 847939)
Test result: ok. 19 passed; 0 failed; 0 skipped; finished in 144.62ms

Running 23 tests for
test/plugin/session/permissions/SessionKeyERC20SpendLimits.t.sol:SessionKeyERC20SpendLimitsTest
[PASS] testFuzz_sessionKeyERC20SpendLimits_setLimits(address,uint256,uint48,uint48) (runs: 500, μ:
118002, ~: 122681)
[PASS] test_executeWithSessionKey_approveOnlyCountsIncrease() (gas: 350706)
[PASS] test_executeWithSessionKey_failWithExceedLimit_multipleSpendFunctions() (gas: 332388)
[PASS] test_executeWithSessionKey_failWithExceedLimit_multipleTokens() (gas: 418209)
[PASS] test_executeWithSessionKey_failWithExceedLimit_multipleTransfer() (gas: 327150)
[PASS] test_executeWithSessionKey_failWithExceedLimit_overflow() (gas: 308088)
[PASS] test_executeWithSessionKey_refreshInterval_failWithSomeTokenLimit() (gas: 795010)
[PASS] test_executeWithSessionKey_refreshInterval_multipleApprove() (gas: 588536)
[PASS] test_executeWithSessionKey_refreshInterval_multipleSpendFunctions() (gas: 590569)
[PASS] test_executeWithSessionKey_refreshInterval_multipleTransfer() (gas: 566594)
[PASS] test_executeWithSessionKey_refreshInterval_singleTransfer() (gas: 529059)
[PASS] test_executeWithSessionKey_success_multipleApprove() (gas: 407733)
[PASS] test_executeWithSessionKey_success_multipleSpendFunctions() (gas: 430986)
[PASS] test_executeWithSessionKey_success_multipleTokens() (gas: 560360)
[PASS] test_executeWithSessionKey_success_multipleTransfer() (gas: 408454)
[PASS] test_sessionKeyERC20Limits_refreshInterval_failWithExceedNextLimit() (gas: 446202)
[PASS] test_sessionKeyERC20SpendLimits_basic_single() (gas: 349715)
[PASS] test_sessionKeyERC20SpendLimits_enforceLimit_none_basic() (gas: 359841)
[PASS] test_sessionKeyERC20SpendLimits_enforceLimit_none_batch() (gas: 321861)
[PASS] test_sessionKeyERC20SpendLimits_exceedLimit_single() (gas: 425496)
[PASS] test_sessionKeyERC20SpendLimits_tokenAddressZeroFails() (gas: 39642)
[PASS] test_sessionKeyERC20SpendLimits_unknownSelectorFails() (gas: 237343)
[PASS] test_sessionKeyERC20SpendLimits_validateSetUp() (gas: 19192)
Test result: ok. 23 passed; 0 failed; 0 skipped; finished in 245.42ms

Running 4 tests for test/account/AccountStorage.t.sol:AccountStorageTest
[PASS] testFuzz_permittedCallKey(address,bytes4) (runs: 500, μ: 645, ~: 645)
[PASS] test_storageSlotErc7201Formula() (gas: 486)
[PASS] test_storageSlotImpl() (gas: 6965)
[PASS] test_storageSlotProxy() (gas: 31967)
Test result: ok. 4 passed; 0 failed; 0 skipped; finished in 44.63ms

Running 20 tests for
test/account/UpgradeableModularAccountPluginManager.t.sol:UpgradeableModularAccountPluginManagerTest
[PASS] test_deployAccount() (gas: 712487)
```

```
[PASS] test_forceOnUninstall() (gas: 653128)
[PASS] test_installPlugin_ExecuteFromPlugin_PermittedExecSelectorNotInstalled() (gas: 1281401)
[PASS] test_installPlugin_alreadyInstalled() (gas: 399058)
[PASS] test_installPlugin_failIfAddingIPluginInterfaceId() (gas: 1229919)
[PASS] test_installPlugin_failWithIPluginFunctionSelector() (gas: 1232706)
[PASS] test_installPlugin_failWithNativeFunctionSelector() (gas: 1230672)
[PASS] test_installPlugin_failWtihErc4337FunctionSelector() (gas: 1233509)
[PASS] test_installPlugin_interfaceNotSupported() (gas: 61231)
[PASS] test_installPlugin_invalidManifest() (gas: 85857)
[PASS] test_installPlugin_missingDependency() (gas: 2507249)
[PASS] test_installPlugin_standard() (gas: 1147680)
[PASS] test_noNonSelfInstallAfterUninstall() (gas: 1019230)
[PASS] test_onUninstallGasLimit() (gas: 844586)
[PASS] test_uninstallAndInstallInBatch() (gas: 1658721)
[PASS] test_uninstallAndInstallInBatch_failwithOtherCalls() (gas: 225344)
[PASS] test_uninstallPlugin_default() (gas: 1322501)
[PASS] test_uninstallPlugin_invalidManifestFails() (gas: 1630630)
[PASS] test_uninstallPlugin_manifestHasChanged() (gas: 4120042)
[PASS] test_uninstallPlugin_manifestParameter() (gas: 1320399)
Test result: ok. 20 passed; 0 failed; 0 skipped; finished in 19.26ms

Running 17 tests for test/libraries/AssociatedLinkedListSetLib.t.sol:AssociatedLinkedListSetLibTest
[PASS] test_add_contains() (gas: 47488)
[PASS] test_add_remove_add() (gas: 52127)
[PASS] test_add_remove_add_empty() (gas: 52002)
[PASS] test_clear() (gas: 35277)
[PASS] test_empty() (gas: 7286)
[PASS] test_getAll() (gas: 71312)
[PASS] test_getAll2() (gas: 94614)
[PASS] test_getAll_empty() (gas: 4793)
[PASS] test_no_address_collision() (gas: 50116)
[PASS] test_remove() (gas: 35074)
[PASS] test_remove_empty() (gas: 4983)
[PASS] test_remove_nonexistent() (gas: 50151)
[PASS] test_remove_nonexistent_empty() (gas: 5095)
[PASS] test_remove_nonexistent_empty2() (gas: 50251)
[PASS] test_tryRemoveKnown1() (gas: 35294)
[PASS] test_tryRemoveKnown2() (gas: 57256)
[PASS] test_tryRemoveKnown_invalid1() (gas: 71468)
Test result: ok. 17 passed; 0 failed; 0 skipped; finished in 1.40ms

Running 8 tests for test/account/AccountLoupe.t.sol:AccountLoupeTest
[PASS] test_pluginLoupe_getExecutionFunctionConfig_native() (gas: 54160)
[PASS] test_pluginLoupe_getExecutionFunctionConfig_plugin() (gas: 39491)
[PASS] test_pluginLoupe_getExecutionHooks() (gas: 27806)
[PASS] test_pluginLoupe_getExecutionHooks_overlapping() (gas: 2394661)
[PASS] test_pluginLoupe_getHooks_multiple() (gas: 1412911)
[PASS] test_pluginLoupe_getInstalledPlugins_initial() (gas: 21478)
[PASS] test_pluginLoupe_getPreValidationHooks() (gas: 25986)
[PASS] test_trace_comprehensivePlugin() (gas: 40529)
Test result: ok. 8 passed; 0 failed; 0 skipped; finished in 5.58ms

Running 10 tests for test/account/ValidationIntersection.t.sol:ValidationIntersectionTest
[PASS] testFuzz_validationIntersect_single(uint256) (runs: 500, μ: 86897, ~: 87733)
[PASS] test_validationIntersect_authorizerAndTimeRange() (gas: 116167)
[PASS] test_validationIntersect_authorizer_sigfail_hook() (gas: 94245)
[PASS] test_validationIntersect_authorizer_sigfail_validationFunction() (gas: 94665)
[PASS] test_validationIntersect_multiplePreValidationHooksIntersect() (gas: 120899)
[PASS] test_validationIntersect_multiplePreValidationHooksSigFail() (gas: 100682)
[PASS] test_validationIntersect_revert_unexpectedAuthorizer() (gas: 64073)
[PASS] test_validationIntersect_timeBounds_intersect_1() (gas: 114266)
[PASS] test_validationIntersect_timeBounds_intersect_2() (gas: 114423)
[PASS] test_validationIntersect_validAuthorizer() (gas: 96344)
Test result: ok. 10 passed; 0 failed; 0 skipped; finished in 156.02ms

Running 2 tests for test/libraries/PluginStorageLib.t.sol:PluginStorageLibTest
[PASS] testFuzz_storagePointer(address,uint256,bytes32,uint256[32]) (runs: 500, μ: 726127, ~: 731620)
[PASS] test_storagePointer() (gas: 52757)
Test result: ok. 2 passed; 0 failed; 0 skipped; finished in 333.25ms

Running 8 tests for test/plugin/TokenReceiverPlugin.t.sol:TokenReceiverPluginTest
[PASS] test_failERC1155Transfer() (gas: 185741)
```

```
[PASS] test_failERC721Transfer() (gas: 66633)
[PASS] test_failERC777Transfer() (gas: 23876)
[PASS] test_failIntrospection() (gas: 19962)
[PASS] test_passERC1155Transfer() (gas: 581235)
[PASS] test_passERC721Transfer() (gas: 447388)
[PASS] test_passERC777Transfer() (gas: 428713)
[PASS] test_passIntrospection() (gas: 394961)
Test result: ok. 8 passed; 0 failed; 0 skipped; finished in 9.65ms

Running 2 tests for test/upgrade/MSCAToMSCA.t.sol:MSCAToMSCATest
[PASS] test_sameStorageSlot_reinstallUpgradeToAndCall() (gas: 193130)
[PASS] test_sameStorageSlot_upgradeToAndCall() (gas: 112087)
Test result: ok. 2 passed; 0 failed; 0 skipped; finished in 3.86ms

Running 8 tests for test/account/ManifestValidity.t.sol:ManifestValidityTest
[PASS] test_ManifestValidity_invalid_HookAlwaysDeny_PostExecHook() (gas: 590238)
[PASS] test_ManifestValidity_invalid_HookAlwaysDeny_RuntimeValidationFunction() (gas: 576489)
[PASS] test_ManifestValidity_invalid_HookAlwaysDeny_UserOpValidation() (gas: 576181)
[PASS] test_ManifestValidity_invalid_ValidationAlwaysAllow_PostExecHook() (gas: 589554)
[PASS] test_ManifestValidity_invalid_ValidationAlwaysAllow_PreExecHook() (gas: 589197)
[PASS] test_ManifestValidity_invalid_ValidationAlwaysAllow_PreRuntimeValidationHook() (gas: 620857)
[PASS] test_ManifestValidity_invalid_ValidationAlwaysAllow_PreUserOpValidationHook() (gas: 621093)
[PASS] test_ManifestValidity_invalid_ValidationAlwaysAllow_UserOpValidationFunction() (gas: 575929)
Test result: ok. 8 passed; 0 failed; 0 skipped; finished in 4.83ms

Running 14 tests for
test/plugin/session/permissions/SessionKeyNativeTokenSpendLimits.t.sol:SessionKeyNativeTokenSpendLimitsTe
st
[PASS] testFuzz_sessionKeyNativeTokenSpendLimits_setLimits(uint256,uint48,uint48) (runs: 500, μ: 96882,
~: 101046)
[PASS] test_sessionKeyNativeTokenSpendLimits_basic_multi() (gas: 380480)
[PASS] test_sessionKeyNativeTokenSpendLimits_basic_refreshInterval_takeMaxStartTime() (gas: 385274)
[PASS] test_sessionKeyNativeTokenSpendLimits_basic_single() (gas: 305061)
[PASS] test_sessionKeyNativeTokenSpendLimits_enforceLimit_none() (gas: 362661)
[PASS] test_sessionKeyNativeTokenSpendLimits_exceedLimit_multi() (gas: 185572)
[PASS] test_sessionKeyNativeTokenSpendLimits_exceedLimit_single() (gas: 305153)
[PASS] test_sessionKeyNativeTokenSpendLimits_multiUserOpBundle_check_noInterval() (gas: 357276)
[PASS] test_sessionKeyNativeTokenSpendLimits_overflowBatch() (gas: 130458)
[PASS] test_sessionKeyNativeTokenSpendLimits_overflowState() (gas: 299427)
[PASS] test_sessionKeyNativeTokenSpendLimits_refreshInterval_exceedNewInterval() (gas: 329879)
[PASS] test_sessionKeyNativeTokenSpendLimits_refreshInterval_multi() (gas: 470230)
[PASS] test_sessionKeyNativeTokenSpendLimits_refreshInterval_single() (gas: 413958)
[PASS] test_sessionKeyNativeTokenSpendLimits_validateSetUp() (gas: 18791)
Test result: ok. 14 passed; 0 failed; 0 skipped; finished in 234.56ms

Running 15 tests for
test/account/AccountPreValidationHooks.t.sol:UpgradeableModularAccountPreValidationHooksTest
[PASS] testFuzz_preRuntimeValidationHooks_revert(bytes) (runs: 500, μ: 1733259, ~: 1727115)
[PASS] test_overlappingPreRuntimeValidationHookAlwaysDeny_install() (gas: 3068289)
[PASS] test_overlappingPreRuntimeValidationHookAlwaysDeny_revert() (gas: 3074578)
[PASS] test_overlappingPreRuntimeValidationHookAlwaysDeny_uninstallPlugin1() (gas: 3068344)
[PASS] test_overlappingPreRuntimeValidationHookAlwaysDeny_uninstallPlugin2() (gas: 3084436)
[PASS] test_overlappingPreRuntimeValidationHook_invalid() (gas: 3143988)
[PASS] test_overlappingPreUserOpValidationHookAlwaysDeny_install() (gas: 3389040)
[PASS] test_preRuntimeValidationHooks_install() (gas: 3129895)
[PASS] test_preRuntimeValidationHooks_revertAlwaysDeny() (gas: 1661420)
[PASS] test_preRuntimeValidationHooks_run() (gas: 3151007)
[PASS] test_preRuntimeValidationHooks_uninstall() (gas: 3088237)
[PASS] test_preUserOpValidationHooks_install() (gas: 3450322)
[PASS] test_preUserOpValidationHooks_revertAlwaysDeny() (gas: 1996789)
[PASS] test_preUserOpValidationHooks_run() (gas: 3574359)
[PASS] test_preUserOpValidationHooks_uninstall() (gas: 3430060)
Test result: ok. 15 passed; 0 failed; 0 skipped; finished in 786.66ms

Running 16 tests for test/account/AccountExecHooks.t.sol:UpgradeableModularAccountExecHooksTest
[PASS] testFuzz_preExecHook_revertData(bytes) (runs: 500, μ: 1748772, ~: 1742768)
[PASS] test_execHookPair_install() (gas: 1797365)
[PASS] test_execHookPair_run() (gas: 1822011)
[PASS] test_execHookPair_uninstall() (gas: 1735898)
[PASS] test_overlappingPreExecHookAlwaysDeny_install() (gas: 3100841)
[PASS] test_overlappingPreExecHookAlwaysDeny_revert() (gas: 3108892)
[PASS] test_overlappingPreExecHookAlwaysDeny_uninstallPlugin1() (gas: 3102161)
```

```
[PASS] test_overlappingPreExecHookAlwaysDeny_uninstallPlugin2() (gas: 3121727)
[PASS] test_overlappingPreExecHook_invalid() (gas: 3176860)
[PASS] test_postOnlyExecHook_install() (gas: 1692033)
[PASS] test_postOnlyExecHook_run() (gas: 1705694)
[PASS] test_postOnlyExecHook_uninstall() (gas: 1648655)
[PASS] test_preExecHook_install() (gas: 1692327)
[PASS] test_preExecHook_revertAlwaysDeny() (gas: 1679427)
[PASS] test_preExecHook_run() (gas: 1707518)
[PASS] test_preExecHook_uninstall() (gas: 1648100)
Test result: ok. 16 passed; 0 failed; 0 skipped; finished in 811.16ms

Running 16 tests for test/plugin/session/permissions/SessionKeyGasLimits.t.sol:SessionKeyGasLimitsTest
[PASS] testFuzz_sessionKeyGasLimits_nolimit(uint256) (runs: 500, μ: 169231, ~: 169141)
[PASS] testFuzz_sessionKeyGasLimits_requireNonceAsAddress(uint192) (runs: 500, μ: 194980, ~: 194980)
[PASS] testFuzz_sessionKeyGasLimits_setLimits(uint256,uint48,uint48) (runs: 500, μ: 100685, ~: 103909)
[PASS] test_sessionKeyGasLimits_enforceLimit_basic_multipleInBundle() (gas: 313234)
[PASS] test_sessionKeyGasLimits_enforceLimit_basic_single() (gas: 273548)
[PASS] test_sessionKeyGasLimits_enforceLimit_none() (gas: 154682)
[PASS] test_sessionKeyGasLimits_exceedLimit_multipleInBundle() (gas: 243600)
[PASS] test_sessionKeyGasLimits_exceedLimit_single() (gas: 174627)
[PASS] test_sessionKeyGasLimits_refreshInterval_inspectValidationData() (gas: 302476)
[PASS] test_sessionKeyGasLimits_refreshInterval_multipleInBundle() (gas: 443868)
[PASS] test_sessionKeyGasLimits_refreshInterval_multipleInBundle_tryExceedFails() (gas: 407241)
[PASS] test_sessionKeyGasLimits_refreshInterval_resetFlagTracking() (gas: 335488)
[PASS] test_sessionKeyGasLimits_refreshInterval_resetFlag_fixWithExtraUO() (gas: 400206)
[PASS] test_sessionKeyGasLimits_refreshInterval_resetFlag_fixWithOwnerReset() (gas: 354068)
[PASS] test_sessionKeyGasLimits_refreshInterval_resetFlag_fixWithPublicReset() (gas: 341332)
[PASS] test_sessionKeyGasLimits_refreshInterval_single() (gas: 359458)
Test result: ok. 16 passed; 0 failed; 0 skipped; finished in 2.48s

Running 16 tests for
test/plugin/session/permissions/SessionKeyPermissions.t.sol:SessionKeyPermissionsTest
[PASS] testFuzz_initialSessionKeysWithPermissions(uint256) (runs: 500, μ: 623922, ~: 693990)
[PASS] testFuzz_sessionKeyPermissions_checkRequiredPaymaster(address,address) (runs: 500, μ: 163857, ~:
164038)
[PASS] testFuzz_sessionKeyPermissions_setRequiredPaymaster(address) (runs: 500, μ: 87252, ~: 87430)
[PASS] testFuzz_sessionKeyTimeRange(uint48,uint48) (runs: 500, μ: 110393, ~: 110517)
[PASS] test_rotateKey_basic() (gas: 361421)
[PASS] test_rotateKey_permissionsTransfer() (gas: 135798)
[PASS] test_sessionKeyPerms_independentKeyStorage() (gas: 147209)
[PASS] test_sessionKeyPerms_reinstallResets() (gas: 338831)
[PASS] test_sessionKeyPerms_requiredPaymaster_partialAddressFails() (gas: 132500)
[PASS] test_sessionKeyPerms_updatePermissions_invalidUpdates() (gas: 66328)
[PASS] test_sessionPerms_contractAllowList() (gas: 271755)
[PASS] test_sessionPerms_contractDefaultAllowList() (gas: 323023)
[PASS] test_sessionPerms_contractDenyList() (gas: 340148)
[PASS] test_sessionPerms_selectorAllowList() (gas: 298155)
[PASS] test_sessionPerms_selectorDenyList() (gas: 361324)
[PASS] test_sessionPerms_validateSetUp() (gas: 15543)
Test result: ok. 16 passed; 0 failed; 0 skipped; finished in 2.43s

Running 3 tests for test/invariant/LinkedListSetLibInvariants.t.sol:LinkedListSetLibInvariantsTest
[PASS] invariant_flagValidity() (runs: 500, calls: 5000, reverts: 0)
[PASS] invariant_getAllEquivalence() (runs: 500, calls: 5000, reverts: 0)
Logs:
  Bound Result 0

[PASS] invariant_shouldContain() (runs: 500, calls: 5000, reverts: 0)
Test result: ok. 3 passed; 0 failed; 0 skipped; finished in 3.14s

Running 17 tests for
test/plugin/session/SessionKeyPluginWithMultiOwner.t.sol:SessionKeyPluginWithMultiOwnerTest
[PASS] testFuzz_sessionKey_addKeysDuringInstall(uint8) (runs: 500, μ: 743303, ~: 665464)
[PASS] testFuzz_sessionKey_invalidFunctionId(uint8,
(address,uint256,bytes,bytes,uint256,uint256,uint256,uint256,uint256,bytes,bytes)) (runs: 500, μ: 22635,
~: 22465)
[PASS] testFuzz_sessionKey_userOpValidation_invalidSig(uint8,uint64) (runs: 500, μ: 803286, ~: 587552)
[PASS] testFuzz_sessionKey_userOpValidation_mismathcedSig(uint8,uint64) (runs: 500, μ: 795787, ~: 688302)
[PASS] testFuzz_sessionKey_userOpValidation_valid(uint16) (runs: 500, μ: 883932, ~: 880244)
[PASS] test_sessionKey_addAndRemoveKeys() (gas: 175372)
[PASS] test_sessionKey_addKeyFailure() (gas: 162899)
[PASS] test_sessionKey_addKeySuccess() (gas: 134109)
```

```
[PASS] test_sessionKey_doesNotContainSentinelValue() (gas: 132047)
[PASS] test_sessionKey_getPredecessor_address() (gas: 165309)
[PASS] test_sessionKey_getPredecessor_missing() (gas: 132158)
[PASS] test_sessionKey_getPredecessor_sentinel() (gas: 165321)
[PASS] test_sessionKey_rotate_existing() (gas: 199118)
[PASS] test_sessionKey_rotate_invalid() (gas: 142309)
[PASS] test_sessionKey_rotate_valid() (gas: 159154)
[PASS] test_sessionKey_useSessionKey() (gas: 317662)
[PASS] test_sessionKey_useSessionKey_failInRuntime() (gas: 138891)
Test result: ok. 17 passed; 0 failed; 0 skipped; finished in 3.12s

Running 3 tests for
test/invariant/AssociatedLinkedListSetLibInvariants.t.sol:AssociatedLinkedListSetLibInvariantsTest
[PASS] invariant_flagValidity() (runs: 500, calls: 5000, reverts: 0)
[PASS] invariant_getAllEquivalence() (runs: 500, calls: 5000, reverts: 0)
[PASS] invariant_shouldContain() (runs: 500, calls: 5000, reverts: 0)
Test result: ok. 3 passed; 0 failed; 0 skipped; finished in 3.79s

Running 18 tests for test/plugin/owner/MultiOwnerPlugin.t.sol:MultiOwnerPluginTest
[PASS] testFuzz_isValidSignature_ContractOwner(bytes32) (runs: 500, μ: 61941, ~: 61941)
[PASS] testFuzz_isValidSignature_ContractOwnerWithEOAOwner(bytes32) (runs: 500, μ: 77403, ~: 77403)
[PASS] testFuzz_isValidSignature_EOAOwner(string,bytes32) (runs: 500, μ: 81311, ~: 81238)
[PASS]
testFuzz_userOpValidationFunction_ContractOwner((address,uint256,bytes,bytes,uint256,uint256,uint256,uint
256,uint256,bytes,bytes)) (runs: 500, μ: 84734, ~: 84714)
[PASS]
testFuzz_userOpValidationFunction_ContractOwnerWithEOAOwner((address,uint256,bytes,bytes,uint256,uint256,
uint256,uint256,uint256,bytes,bytes)) (runs: 500, μ: 103758, ~: 103704)
[PASS] testFuzz_userOpValidationFunction_EOAOwner(string,
(address,uint256,bytes,bytes,uint256,uint256,uint256,uint256,uint256,bytes,bytes)) (runs: 500, μ: 90458,
~: 90373)
[PASS] test_eip712Domain() (gas: 24723)
[PASS] test_multiOwnerPlugin_sentinelIsNotOwner() (gas: 10802)
[PASS] test_onInstall_success() (gas: 66256)
[PASS] test_onUninstall_success() (gas: 34587)
[PASS] test_pluginInitializeGuards() (gas: 57381)
[PASS] test_pluginManifest() (gas: 35717)
[PASS] test_runtimeValidationFunction_OwnerOrSelf() (gas: 19632)
[PASS] test_updateOwners_failWithDuplicatedAddresses() (gas: 42008)
[PASS] test_updateOwners_failWithEmptyOwners() (gas: 30396)
[PASS] test_updateOwners_failWithNotExist() (gas: 17467)
[PASS] test_updateOwners_failWithZeroAddressOwner() (gas: 13723)
[PASS] test_updateOwners_success() (gas: 60909)
Test result: ok. 18 passed; 0 failed; 0 skipped; finished in 3.68s

Ran 36 test suites: 417 tests passed, 0 failed, 0 skipped (417 total tests)
```

Code Coverage

Coverage was determined with `forge coverage`.

File	% Lines	% Statements	% Branches	% Funcs
src/account/AccountExecutor.sol	94.44% (**17/**18)	94.44% (**17/**18)	87.50% (**7/**8)	100.00% (**11/**11)
src/account/AccountLoupe.sol	100.00% (**41/**41)	100.00% (**47/**47)	75.00% (**3/**4)	100.00% (**6/**6)
src/account/AccountStorageInitializable.sol	0.00% (**0/**5)	0.00% (**0/**5)	0.00% (**0/**4)	0.00% (**0/**1)
src/account/PluginManagerInternals.sol	91.29% (**262/**287)	92.14% (**293/**318)	83.02% (**88/**106)	100.00% (**19/**19)
src/account/UpgradeableMo	100.00%	100.00%	87.18% (**68/**78)	100.00%

File	% Lines	% Statements	% Branches	% Funcs
dularAccount.sol	(**234/**234)	(**280/**280)		(**23/**23)
src/factory/MultiOwnerMSCAFactory.sol	94.74% (**18/**19)	96.15% (**25/**26)	50.00% (**2/**4)	100.00% (**7/**7)
src/factory/MultiOwnerTokenReceiverMSCAFactory.sol	95.24% (**20/**21)	96.43% (**27/**28)	50.00% (**2/**4)	100.00% (**7/**7)
src/helpers/KnownSelectors.sol	100.00% (**20/**20)	100.00% (**46/**46)	100.00% (**0/**0)	100.00% (**2/**2)
src/libraries/AccountStorageV1.sol	100.00% (**2/**2)	100.00% (**4/**4)	100.00% (**0/**0)	100.00% (**2/**2)
src/libraries/AssociatedLinkedListSetLib.sol	0.00% (**0/**111)	0.00% (**0/**221)	0.00% (**0/**26)	0.00% (**0/**22)
src/libraries/CastLib.sol	100.00% (**11/**11)	100.00% (**14/**14)	100.00% (**0/**0)	100.00% (**4/**4)
src/libraries/CountableLinkedListSetLib.sol	0.00% (**0/**19)	0.00% (**0/**28)	0.00% (**0/**10)	0.00% (**0/**3)
src/libraries/FunctionReferenceLib.sol	100.00% (**5/**5)	100.00% (**10/**10)	100.00% (**0/**0)	100.00% (**3/**3)
src/libraries/LinkedListSetLib.sol	0.00% (**0/**79)	0.00% (**0/**153)	0.00% (**0/**22)	0.00% (**0/**18)
src/libraries/PluginStorageLib.sol	100.00% (**4/**4)	100.00% (**4/**4)	100.00% (**0/**0)	100.00% (**3/**3)
src/plugins/BasePlugin.sol	23.08% (**3/**13)	37.50% (**6/**16)	50.00% (**1/**2)	21.43% (**3/**14)
src/plugins/TokenReceiverPlugin.sol	80.00% (**20/**25)	80.77% (**21/**26)	100.00% (**0/**0)	75.00% (**6/**8)
src/plugins/owner/MultiOwnerPlugin.sol	86.87% (**86/**99)	89.68% (**113/**126)	82.14% (**23/**28)	90.48% (**19/**21)
src/plugins/session/SessionKeyPlugin.sol	82.14% (**69/**84)	84.96% (**96/**113)	37.50% (**6/**16)	78.57% (**11/**14)
src/plugins/session/permissions/SessionKeyPermissionsBase.sol	100.00% (**36/**36)	100.00% (**54/**54)	100.00% (**2/**2)	100.00% (**10/**10)
src/plugins/session/permissions/SessionKeyPermissionsLoupe.sol	100.00% (**30/**30)	100.00% (**45/**45)	100.00% (**4/**4)	100.00% (**8/**8)
src/plugins/session/permissions/SessionKeyPermissionsPlugin.sol	95.00% (**247/**260)	95.86% (**301/**314)	94.90% (**93/**98)	93.10% (**27/**29)
test/TestUtils.sol	0.00% (**0/**7)	0.00% (**0/**14)	100.00% (**0/**0)	0.00% (**0/**1)
test/Utils.sol	100.00% (**4/**4)	100.00% (**6/**6)	100.00% (**0/**0)	100.00% (**1/**1)
test/invariant/AssociatedLinkedListSetLibInvariants.t.sol	22.45% (**11/**49)	18.18% (**12/**66)	0.00% (**0/**4)	12.50% (**1/**8)

File	% Lines	% Statements	% Branches	% Funcs
test/invariant/LinkedListSetLibInvariants.t.sol	29.73% (**11/**37)	22.22% (**12/**54)	0.00% (**0/**4)	20.00% (**1/**5)
test/invariant/handlers/AssociatedLinkedListSetHandler.sol	48.57% (**85/**175)	51.04% (**123/**241)	28.95% (**22/**76)	47.06% (**8/**17)
test/invariant/handlers/LinkedListSetHandler.sol	56.35% (**71/**126)	56.71% (**93/**164)	43.59% (**34/**78)	50.00% (**8/**16)
test/mocks/ContractOwner.sol	75.00% (**3/**4)	85.71% (**6/**7)	50.00% (**1/**2)	100.00% (**2/**2)
test/mocks/Counter.sol	100.00% (**2/**2)	100.00% (**2/**2)	100.00% (**0/**0)	100.00% (**2/**2)
test/mocks/MockDiamondStorageContract.sol	100.00% (**0/**0)	100.00% (**0/**0)	100.00% (**0/**0)	100.00% (**1/**1)
test/mocks/MockPlugin.sol	68.75% (**11/**16)	76.19% (**16/**21)	100.00% (**2/**2)	75.00% (**6/**8)
test/mocks/plugins/BadTransferOwnershipPlugin.sol	0.00% (**0/**16)	0.00% (**0/**17)	100.00% (**0/**0)	0.00% (**0/**5)
test/mocks/plugins/BaseTestPlugin.sol	0.00% (**0/**1)	0.00% (**0/**1)	100.00% (**0/**0)	0.00% (**0/**1)
test/mocks/plugins/ChangingManifestPlugin.sol	41.18% (**7/**17)	41.18% (**7/**17)	100.00% (**0/**0)	63.64% (**7/**11)
test/mocks/plugins/ComprehensivePlugin.sol	45.61% (**26/**57)	40.58% (**28/**69)	0.00% (**0/**20)	27.27% (**3/**11)
test/mocks/plugins/ExecFromPluginPermissionsMocks.sol	98.11% (**104/**106)	97.60% (**122/**125)	50.00% (**2/**4)	85.71% (**36/**42)
test/mocks/plugins/ManifestValidityMocks.sol	76.67% (**46/**60)	67.65% (**46/**68)	100.00% (**0/**0)	21.88% (**7/**32)
test/mocks/plugins/ReturnDataPluginMocks.sol	97.14% (**34/**35)	95.65% (**44/**46)	100.00% (**0/**0)	76.92% (**10/**13)
test/mocks/plugins/UninstallErrorsPlugin.sol	58.33% (**7/**12)	58.33% (**7/**12)	100.00% (**2/**2)	85.71% (**6/**7)
test/mocks/plugins/ValidationPluginMocks.sol	94.87% (**37/**39)	95.45% (**42/**44)	66.67% (**4/**6)	69.23% (**9/**13)
test/mocks/tokens/MockERC1155.sol	100.00% (**1/**1)	100.00% (**1/**1)	100.00% (**0/**0)	100.00% (**1/**1)
test/mocks/tokens/MockERC20.sol	100.00% (**1/**1)	100.00% (**1/**1)	100.00% (**0/**0)	100.00% (**1/**1)
test/mocks/tokens/MockERC777.sol	66.67% (**6/**9)	70.00% (**7/**10)	100.00% (**0/**0)	18.18% (**2/**11)
Total	72.46% (**1597/**2204)	68.64% (**1981/**2886)	59.32% (**366/**617)	63

Fix Review Update

File	% Lines	% Statements	% Branches	% Funcs
ext/UUPSUpgradeable.sol	83.33% (**5/**6)	100.00% (**3/**3)	0.00% (**0/**3)	100.00% (**2/**2)
script/Deploy.s.sol	0.00% (**0/**52)	0.00% (**0/**67)	0.00% (**0/**12)	0.00% (**0/**2)
src/account/AccountExecutor.sol	94.44% (**17/**18)	94.44% (**17/**18)	87.50% (**7/**8)	100.00% (**11/**11)
src/account/AccountLoupe.sol	100.00% (**35/**35)	100.00% (**45/**45)	75.00% (**3/**4)	100.00% (**5/**5)
src/account/AccountStorageInitializable.sol	0.00% (**0/**5)	0.00% (**0/**5)	0.00% (**0/**4)	0.00% (**0/**1)
src/account/AccountStorageV1.sol	100.00% (**2/**2)	100.00% (**4/**4)	100.00% (**0/**0)	100.00% (**2/**2)
src/account/PluginManagerInternals.sol	90.05% (**199/**221)	90.77% (**246/**271)	78.57% (**77/**98)	100.00% (**16/**16)
src/account/UpgradeableModularAccount.sol	99.51% (**202/**203)	99.61% (**256/**257)	90.28% (**65/**72)	100.00% (**23/**23)
src/factory/MultiOwnerMSCAFactory.sol	92.86% (**26/**28)	94.29% (**33/**35)	78.57% (**11/**14)	100.00% (**6/**6)
src/factory/MultiOwnerTokenReceiverMSCAFactory.sol	90.00% (**27/**30)	91.89% (**34/**37)	71.43% (**10/**14)	100.00% (**6/**6)
src/helpers/CastLib.sol	100.00% (**10/**10)	100.00% (**14/**14)	100.00% (**0/**0)	100.00% (**4/**4)
src/helpers/FactoryHelpers.sol	100.00% (**7/**7)	100.00% (**10/**10)	100.00% (**2/**2)	100.00% (**2/**2)
src/helpers/FunctionReferenceLib.sol	75.00% (**6/**8)	63.64% (**14/**22)	100.00% (**0/**0)	66.67% (**4/**6)
src/helpers/KnownSelectors.sol	100.00% (**26/**26)	100.00% (**64/**64)	100.00% (**0/**0)	100.00% (**3/**3)
src/libraries/AssociatedLinkedListSetLib.sol	0.00% (**0/**97)	0.00% (**0/**198)	0.00% (**0/**24)	0.00% (**0/**22)
src/libraries/CountableLinkedListSetLib.sol	0.00% (**0/**17)	0.00% (**0/**24)	0.00% (**0/**10)	0.00% (**0/**3)
src/libraries/LinkedListSetLib.sol	0.00% (**0/**78)	0.00% (**0/**152)	0.00% (**0/**22)	0.00% (**0/**18)
src/libraries/PluginStorageLib.sol	100.00% (**5/**5)	100.00% (**5/**5)	100.00% (**0/**0)	100.00% (**3/**3)
src/plugins/BasePlugin.sol	23.08% (**3/**13)	37.50% (**6/**16)	50.00% (**1/**2)	16.67% (**2/**12)
src/plugins/TokenReceiverPlugin.sol	80.00% (**20/**25)	80.77% (**21/**26)	100.00% (**0/**0)	75.00% (**6/**8)
src/plugins/owner/MultiOwnerPlugin.sol	87.78% (**79/**90)	90.98% (**111/**122)	87.50% (**21/**24)	89.47% (**17/**19)
src/plugins/session/SessionK	85.71%	88.97%	85.00%	92.86%

File	% Lines	% Statements	% Branches	% Funcs
eyPlugin.sol	(**90/**105)	(**121/**136)	(**17/**20)	(**13/**14)
src/plugins/session/permissions/SessionKeyPermissions.sol	98.51% (**199/**202)	98.80% (**247/**250)	96.67% (**87/**90)	100.00% (**21/**21)
src/plugins/session/permissions/SessionKeyPermissionsBase.sol	97.30% (**36/**37)	98.21% (**55/**56)	50.00% (**1/**2)	100.00% (**10/**10)
src/plugins/session/permissions/SessionKeyPermissionsLoupe.sol	100.00% (**30/**30)	100.00% (**42/**42)	100.00% (**4/**4)	100.00% (**8/**8)
test/TestUtils.sol	0.00% (**0/**7)	0.00% (**0/**14)	100.00% (**0/**0)	0.00% (**0/**1)
test/Utils.sol	100.00% (**4/**4)	100.00% (**6/**6)	100.00% (**0/**0)	100.00% (**1/**1)
test/account/phases/AccountStatePhases.t.sol	0.00% (**0/**51)	0.00% (**0/**58)	100.00% (**0/**0)	0.00% (**0/**15)
test/invariant/AssociatedLinkedListSetLibInvariants.t.sol	22.45% (**11/**49)	18.18% (**12/**66)	0.00% (**0/**4)	12.50% (**1/**8)
test/invariant/LinkedListSetLibInvariants.t.sol	29.73% (**11/**37)	22.22% (**12/**54)	0.00% (**0/**4)	20.00% (**1/**5)
test/invariant/handlers/AssociatedLinkedListSetHandler.sol	48.57% (**85/**175)	50.62% (**122/**241)	30.26% (**23/**76)	47.06% (**8/**17)
test/invariant/handlers/LinkedListSetHandler.sol	48.41% (**61/**126)	48.78% (**80/**164)	39.74% (**31/**78)	50.00% (**8/**16)
test/mocks/ContractOwner.sol	85.71% (**6/**7)	90.91% (**10/**11)	75.00% (**3/**4)	100.00% (**2/**2)
test/mocks/Counter.sol	100.00% (**2/**2)	100.00% (**2/**2)	100.00% (**0/**0)	100.00% (**2/**2)
test/mocks/MockDiamondStorageContract.sol	100.00% (**0/**0)	100.00% (**0/**0)	100.00% (**0/**0)	100.00% (**1/**1)
test/mocks/MockPlugin.sol	68.75% (**11/**16)	76.19% (**16/**21)	100.00% (**2/**2)	75.00% (**6/**8)
test/mocks/plugins/AccountStateMutatingPlugin.sol	90.79% (**69/**76)	91.03% (**71/**78)	63.64% (**28/**44)	100.00% (**15/**15)
test/mocks/plugins/BadTransferOwnershipPlugin.sol	0.00% (**0/**16)	0.00% (**0/**17)	100.00% (**0/**0)	0.00% (**0/**5)
test/mocks/plugins/BaseTestPlugin.sol	0.00% (**0/**1)	0.00% (**0/**1)	100.00% (**0/**0)	0.00% (**0/**1)
test/mocks/plugins/ChangingManifestPlugin.sol	41.18% (**7/**17)	41.18% (**7/**17)	100.00% (**0/**0)	63.64% (**7/**11)
test/mocks/plugins/ComprehensivePlugin.sol	51.79% (**29/**56)	48.53% (**33/**68)	15.00% (**3/**20)	54.55% (**6/**11)
test/mocks/plugins/ExecFromPluginPermissionsMocks.sol	97.40% (**75/**77)	96.84% (**92/**95)	100.00% (**0/**0)	86.67% (**26/**30)

File	% Lines	% Statements	% Branches	% Funcs
test/mocks/plugins/ManifestValidityMocks.sol	76.67% (**46/**60)	67.65% (**46/**68)	100.00% (**0/**0)	21.88% (**7/**32)
test/mocks/plugins/ReturnDataPluginMocks.sol	97.14% (**34/**35)	95.65% (**44/**46)	100.00% (**0/**0)	76.92% (**10/**13)
test/mocks/plugins/UninstallErrorsPlugin.sol	54.55% (**6/**11)	54.55% (**6/**11)	100.00% (**2/**2)	83.33% (**5/**6)
test/mocks/plugins/ValidationPluginMocks.sol	94.87% (**37/**39)	95.45% (**42/**44)	66.67% (**4/**6)	69.23% (**9/**13)
test/mocks/tokens/MockERC1155.sol	100.00% (**1/**1)	100.00% (**1/**1)	100.00% (**0/**0)	100.00% (**1/**1)
test/mocks/tokens/MockERC20.sol	100.00% (**1/**1)	100.00% (**1/**1)	100.00% (**0/**0)	100.00% (**1/**1)
test/mocks/tokens/MockERC777.sol	66.67% (**6/**9)	70.00% (**7/**10)	100.00% (**0/**0)	18.18% (**2/**11)
Total	68.65% (**1526/**2223)	65.86% (**1958/**2973)	60.09% (**402/**669)	62.47% (**283/**453)

Changelog

- 2023-12-20 - Initial report
- 2024-01-30 - Final report

About Quantstamp

Quantstamp is a global leader in blockchain security. Founded in 2017, Quantstamp's mission is to securely onboard the next billion users to Web3 through its best-in-class Web3 security products and services.

Quantstamp's team consists of cybersecurity experts hailing from globally recognized organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Quantstamp engineers hold PhDs or advanced computer science degrees, with decades of combined experience in formal verification, static analysis, blockchain audits, penetration testing, and original leading-edge research.

To date, Quantstamp has performed more than 500 audits and secured over $200 billion in digital asset risk from hackers. Quantstamp has worked with a diverse range of customers, including startups, category leaders and financial institutions. Brands that Quantstamp has worked with include Ethereum 2.0, Binance, Visa, PayPal, Polygon, Avalanche, Curve, Solana, Compound, Lido, MakerDAO, Arbitrum, OpenSea and the World Economic Forum.

Quantstamp's collaborations and partnerships showcase our commitment to world-class research, development and security. We're honored to work with some of the top names in the industry and proud to secure the future of web3.

Notable Collaborations & Customers:

- Blockchains: Ethereum 2.0, Near, Flow, Avalanche, Solana, Cardano, Binance Smart Chain, Hedera Hashgraph, Tezos
- DeFi: Curve, Compound, Maker, Lido, Polygon, Arbitrum, SushiSwap
- NFT: OpenSea, Parallel, Dapper Labs, Decentraland, Sandbox, Axie Infinity, Illuvium, NBA Top Shot, Zora
- Academic institutions: National University of Singapore, MIT

Links to other websites

You may, through hypertext or other computer links, gain access to web sites operated by persons other than Quantstamp. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites&aspo; owners. You agree that Quantstamp are not responsible for the content or operation of such web sites, and that Quantstamp shall have no liability to you or any other person or entity for the use of third-party web sites. Except as described below, a hyperlink from this web site to another web site does not imply or mean that Quantstamp endorses the content on that web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the report. Quantstamp assumes no responsibility for the use of third-party software on any website and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any output generated by such software.

Disclaimer

The review and this report are provided on an as-is, where-is, and as-available basis. To the fullest extent permitted by law, Quantstamp disclaims 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. You agree that your access and/or use of the report and other results of the review, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. 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. This report is based on the scope of materials and documentation provided for a limited review at the time provided. You acknowledge that Blockchain technology remains under development and is subject to unknown risks and flaws and, as such, the report may not be complete or inclusive of all vulnerabilities. The review is limited to the materials identified in the report and does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. The report does not indicate the endorsement by Quantstamp of any particular project or team, nor guarantee its security, and may not be represented as such. No third party is entitled to rely on the report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. Quantstamp does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party, or 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, or any related services and products, any hyperlinked websites, or any other websites or mobile applications, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third party. 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.

Quantstamp