Quantstamp

Alchemy - Modular Account V2

Executive Summary

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

Type	Account Abstraction
Timeline	2024-10-17 through 2024-11-05
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Modular Account v2 Audit Scope and Documentation README.md (modular-account) README.md (reference-implementation)
Source Code	• erc6900/reference-implementation #e5f55ab • alchemyplatform/modular-account #14afcd8
Auditors	• Hytham Farah Auditing Engineer • Nikita Belenkov Auditing Engineer • Ruben Koch Senior Auditing Engineer

Documentation quality	High
Test quality	High
Total Findings	23 Fixed: 9 Acknowledged: 14
High severity findings	2 Fixed: 2
Medium severity findings	0
Low severity findings	7 Fixed: 3 Acknowledged: 4
Undetermined severity findings	0
Informational findings	14 Fixed: 4 Acknowledged: 10

Summary of Findings

Quantstamp audited the v2 version of the `modular-account` repository by Alchemy, as well as small parts of the `reference-implementation` repository of ERC-6900 standard.

`modular-account` is designed to provide a flexible, modular system for Ethereum accounts in compliance with ERC-6900 v0.8, enabling users to add, update, and manage various functionalities through modular extensions. The core purpose of this framework is to offer smart contract accounts that can be dynamically customized with specific modules for permissions, validation, and account management, allowing for adaptable, secure account configurations tailored to diverse use cases. Modules operate independently, ensuring modularity, and are supported by foundational libraries that manage function installation, validation logic, and execution handling. This audit also includes critical supporting files from the reference-implementation repository that define constants and libraries used in the former repository (see section "Scope" for a file list of the contracts in the scope of the audit).

Overall the code is well-written and the Alchemy team was responsive and helpful in answering all questions and the test suite for `modular-account` is robust.

Throughout the audit, two high-severity issues were found, one about a lack of validation hooks running on deferred actions (ALC-1) and the other highlighting a potential breach of the `NativeTokenLimitModule` when empty paymaster data is submitted (ALC-2). The rest of the issues are relatively minor and easily addressed.

Fix-Review Update: The issues were either fixed properly or acknowledged with reasonable grounds and a major change was introduced to the nonce system.

ID	DESCRIPTION	SEVERITY	STATUS
ALC-1	**Validation Modules' Validation Can Be Fully Bypassed if Signature Validation Is Also Enabled**	• High	Fixed

ID	DESCRIPTION	SEVERITY	STATUS
ALC-2	`NativeTokenLimitModule` Can Be Bypassed	High	Fixed
ALC-3	Missing Function Selector Check in `isIModuleFunction()`	Low	Fixed
ALC-4	Incorrect Masking of `deadline`	Low	Fixed
ALC-5	Improvements to the Execution Installation Flow	Low	Acknowledged
ALC-6	Sender-Parameter in Certain Cases Will Always Be Entrypoint Rather than `UserOp.sender`	Informational	Acknowledged
ALC-7	Code-Less (Pre-)Signature Validation Modules May Result in Always-Passing Validations	Low	Acknowledged
ALC-8	Forwarding the `entityId` to the (Un-)Installation Callbacks	Low	Acknowledged
ALC-9	Installed Execution Functions Can Collide with Native Functions	Low	Acknowledged
ALC-10	Deferred Actions Allow Usage of Validation Functions in UserOp Context Without `isUserOpValidation` Flag	Informational	Acknowledged
ALC-11	Potential Denial of Service Due to Misconfigured Time Ranges	Low	Fixed
ALC-12	Incomplete List of Native Selectors Returned From `isNativeFunction()`	Informational	Fixed
ALC-13	`getExecutionData()` Incorrectly Marks Native View Functions as Unlockable with Global Validation	Informational	Fixed
ALC-14	Unrelated Deferred Actions Could Be Dropped by Bundler to Apply Unused Gas Penalty to Account	Informational	Acknowledged
ALC-15	Incomplete Module Data Cleanup on Uninstall	Informational	Acknowledged
ALC-16	`executeBatch()` Can Be Used to Bypass Other Selector's Validation Hook Execution	Informational	Acknowledged
ALC-17	Deactivated Fallback Signer in `SemiModularAccountBase` Returned as Global Validation	Informational	Acknowledged
ALC-18	Reduce Possibilities of Execution Reverts for `NativeTokenLimitModule`	Informational	Acknowledged
ALC-19	`NativeTokenLimitModule` Does Not Return `IValidationHookModule` As Supported Interface	Informational	Fixed
ALC-20	Consider adding a `forceUninstallation` Flag that Requires `onInstall()` Callback Success	Informational	Acknowledged
ALC-21	Incorrect Casting in `toSetValue` Functions	Informational	Fixed
ALC-22	Individual Validation Hooks Cannot Be Removed From Configurations	Informational	Acknowledged
ALC-23	Unenforced Off-Chain Checks for Modules and Modular Account	Informational	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 audit primarily focused on the `modular-account` repository, specifically reviewing the core commit:

- `14afcd84859cf468684634eb9a9f8cb787f60b16`.

As the audit progressed, additional features agreed upon were included:

- **Pull Request** #288: Adds functionality to deploy WebAuthn accounts from a factory.
- **Pull Request** #275: Implements a 7702 compatible semi-modular account.

These pull requests were assessed as extensions to the initial scope, reviewed to ensure compatibility and adherence to the security standards of the core audit. Additionally, after fixes were submitted, we reviewed code changes to verify and assess the changes. up to commit:

- `9c1ccda71efb20d4bc62989eb0d93071c28e5d9e`.

The audit also included relevant parts of the `reference-implementation` repository, at commit `e5f55ab1c3ac2f78efa14967e90e95c3e4ace38c`, to address dependencies impacting the `modular-account`'s functionality.

Files Included

1. **Modular Account**

```
src
├── account
│   ├── AccountBase.sol
│   ├── AccountStorage.sol
│   ├── AccountStorageInitializable.sol
```

```
│   ├── ModularAccount.sol
│   ├── ModularAccountBase.sol
│   ├── ModularAccountView.sol
│   ├── ModuleManagerInternals.sol
│   ├── SemiModularAccountBase.sol
│   ├── SemiModularAccountBytecode.sol
│   ├── SemiModularAccountStorageOnly.sol
│   └── TokenReceiver.sol
├── factory
│   └── AccountFactory.sol
├── helpers
│   ├── Constants.sol
│   ├── ExecutionInstallDelegate.sol
│   ├── NativeFunctionDelegate.sol
│   ├── SignatureType.sol
│   └── ValidationResHelpers.sol
├── libraries
│   ├── ERC7739ReplaySafeWrapperLib.sol
│   ├── ExecutionLib.sol
│   ├── KnownSelectorsLib.sol
│   ├── LinkedListSetLib.sol
│   ├── MemManagementLib.sol
│   └── ModuleInstallCommonsLib.sol
└── modules
    ├── ModuleBase.sol
    ├── permissions
    │   ├── AllowlistModule.sol
    │   ├── NativeTokenLimitModule.sol
    │   ├── PaymasterGuardModule.sol
    │   └── TimeRangeModule.sol
    └── validation
        ├── SingleSignerValidationModule.sol
        └── WebAuthnValidationModule.sol
```

2. **Reference Implementation**

```
src
├── helpers
│   ├── Constants.sol
│   └── EmptyCalldataSlice.sol
├── libraries
│   ├── HookConfigLib.sol
│   ├── ModuleEntityLib.sol
│   ├── SparseCalldataSegmentLib.sol
│   └── ValidationConfigLib.sol
└── modules
    ├── ModuleEIP712.sol
    └── ReplaySafeWrapper.sol
```

Operational Considerations

As part of this audit, we have made several assumptions regarding the operational aspects of the protocol. These considerations are essential for understanding the context in which the protocol operates but are not necessarily to be mitigated within the codebase:

- **Upgradability**: The account contracts are designed to be upgradable using the UUPSUpgradeable pattern. We assume that any upgrades will be conducted through well-planned and audited processes. The upgrade authority is trusted to act in the best interest of the users, ensuring that upgrades do not introduce vulnerabilities or inconsistencies.
- **Module Security**: The protocol relies on external modules for validation and execution logic. The security and correctness of the account depend on the proper functioning of these modules. We assume that all installed modules are secure, correctly implemented, installed with the intended hooks and validations, and have undergone thorough security audits.

- **EntryPoint Dependency**: The accounts interact with the EntryPoint contract for user operation validation and execution, adhering to the ERC-4337 standard. We assume that the EntryPoint contract is secure, operates as intended, and any changes to it will not adversely affect the accounts' security.
- **Module Management**: The ability to install and uninstall validation and execution modules allows for flexible account management. We assume that only authorized entities can perform these actions and that they will manage modules carefully to avoid introducing security risks or state inconsistencies.
- **External Contract Interactions**: Through the `execute()` and `executeBatch()` functions, accounts may interact with external contracts. We assume that the contracts being called are trusted and that appropriate measures are in place to handle potential reentrancy or other external risks.
- **Signature Validation**: The accounts support external signature validation modules. We assume that any signature validation logic provided by out-of-scope third-party modules is secure and correctly verifies the authenticity of signatures.
- **Fallback Signer Usage**: In the `SemiModularAccount` implementation, an optional fallback signer can be used for signature validation. We assume that if the fallback signer is enabled, it is securely managed, and its private key is protected against unauthorized access.
- **Gas Management**: Operations involving the accounts may have significant gas costs due to their complexity. We assume that users are aware of gas implications and manage their transactions accordingly to avoid failed transactions or excessive fees.
- **Semi-Modular Accounts**: The repository provides a set of semi-modular accounts, enabling a more gas-efficient deployment and validation. While a default `ModularAccount` can be upgraded to and from a `SemiModularAccountStorageOnly`, a `SemiModularAccountBytecode` should never be upgraded to and from a `SemiModularAccountStorageOnly` or a `ModularAccount` instance.

Key Actors And Their Capabilities

Fallback Signer

Purpose

The Fallback Signer is a user-designated, privileged role that acts as a global validation that is baked into the account that can however be enabled and disabled.

Permissions

- Alternative validation authority for user operations and runtime functions.
- Capable of validating EIP-1271 contract-based signatures.
- Configurable by the user, with the ability to enable or disable its role.

Modified Functions (in SemiModularAccountBase.sol)

1. `updateFallbackSignerData()`: Allows setting or updating the fallback signer and enabling/disabling its functionality.
2. `_execUserOpValidation()`: Extends user operation validation to permit execution through the fallback signer.
3. `_execRuntimeValidation()`: Ensures runtime validation checks can be performed by the fallback signer.
4. `_exec1271Validation()`: Enables fallback signer validation for EIP-1271 contract signatures.
5. `_checkSignature()`: Validates signatures against the fallback signer, supporting both EOAs and contract types.
6. `_getFallbackSigner()`: Retrieves the active fallback signer and reverts if disabled.
7. `_retrieveFallbackSignerUnchecked()`: Provides internal access to the fallback signer without checks.

These functions adapt the standard modular account structure to include fallback-specific validation capabilities.

Findings

ALC-1

Validation Modules' Validation Can Be Fully Bypassed if Signature Validation Is Also Enabled

High Fixed

Update

Deferred actions are now only possible when there are no existing pre-validation hooks, and signature validation. Validation-associated and regular execution hooks are now executed as part of the deferred action. We note that signature validation is invoked instead of the regular user operation validation. This means that in case a validation module has signature validation enabled and technically provides validation for a certain function, the user operation flow can be replaced with the signature validation. This is intended, but this should be communicated to developers, as in case some restrictive logic is being performed in UO validation, which could be circumvented with signature validation flow in deferred action.

Update

Marked as "Fixed" by the client.
Addressed in: `48a77f0bfec4f999bd21299c3a338bc6b704c8f8`.
The client provided the following explanation:

```
Blocked deferred validation for validations that have validation hooks, and made validation-associated
execution hooks run for deferred actions.
```

Description: The modular account system supports deferred actions, enabling operations such as installing validation modules to be delayed until execution. Deferred actions, embedded in the `UserOperation`'s signature field, will likely only be used within the same `UserOperation` for validation installation. However, their design allows for arbitrary use, provided the outer `ValidationConfig` validates the selector.

Currently, deferred action execution relies solely on signature verification through the outer `ValidationConfig`'s `validateSignature()` function. This bypasses critical validation steps, including pre-validation (`preUserOpValidationHook()`) and execution hooks, which enforce security policies like access control. Without these hooks, attackers can bypass the full validation process, enabling unauthorized operations and privilege escalation.

In other words, as soon as a validation module is installed with `isSignatureValidation = true`, all validation-associated hooks and the main validation step that the module also has installed can be skipped by embedding a call to a deferred action and simply validating it via the signature validation path. We argue that this situation is significantly different from the intentional skipping of related hooks when calling `executeBatch()` (ALC-16), as `executeBatch()` is well understood as a powerful selector with very restricted access.

Additionally, it should be noted that due to this issue, any validation module with access to `executeBatch()` and signature validation enables the checks imposed on `_checkExecuteBatchValidationApplicability()` to be bypassed, as the calldata validation is also skipped.

Checking only that a `ValidationConfig` provides a valid signature is insufficient; deferred actions must trigger all associated validation hooks to maintain security.

Exploit Scenario:
Consider an account using a `SessionKeyModule` with an `executeWithSessionKey()` function for restricted operations and a stateless `validateSignature()` function without associated hooks. A user has a session key limited to withdrawing up to **10 USDC per month**, enforced by validation hooks in its `ValidationConfig`.

An attacker with access to this session key could create a `UserOperation` embedding a deferred action to call `executeWithSessionKey()`. The system, relying solely on signature verification, skips the necessary validation hooks. This enables the attacker to bypass restrictions and perform unauthorized actions, such as withdrawing excessive funds or executing restricted operations.

Recommendation: Ensure that deferred actions trigger all associated validation steps, including pre-validation and execution hooks, when executed. This prevents bypassing critical checks and maintains robust security for modular accounts.

ALC-2 `NativeTokenLimitModule` Can Be Bypassed

High · Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `37cd0f18422ece242612660c0f41eb9c4e4552cc` and `d83d463904fe35f6d369a44529e132d8e8640796`

File(s) affected: `NativeTokenLimitModule`, `PaymasterGuardModule`

Description: The `NativeTokenLimitModule` is intended to limit the native tokens some validation can access from the account. In case paymasters provided, the check is regularly skipped, as the paymaster is covering the gas costs, unless it is some special paymaster that has some existing ERC20 token allowance that withdraws e.g. ERC20 tokens equivalent to the gas consumed from the account.

However, the decoded address is missing a check that it is a non-zero length `bytes` array of just zeroes. This enables a bypass in the `paymasters` registry, as in this case `userOp.paymasterAndData.length != 0` as well as `specialPaymasters[address(0x0)][msg.sender] == 0`, if `address(0x0)` is not specifically registered as a `specialPaymaster`, which does not seem intended.

EthInfinitism's EntryPoint internally treats a `paymasterAndData` field of zero length the same way as `paymaster = address(0x0)`, `paymasterVerificationGasLimit = 0` and `paymasterPostOpGasLimit = 0`. The unpacking of the paymasterAndData variable here does not do any reverts on zero-values either.

This enables all UserOperation-validations installed on the account that have this validation hook installed to fully drain the underlying account by simply encoding a 52 byte long `paymasterAndData` field containing only zeroes.

Furthermore, in the `PaymasterGuardModule`, a similar check is performed. If this module were to be installed without the `onInstall()` callback or with an `installData` parameter encoding the zero address, native funds could be drained in the same way.

Exploit Scenario:

An attacker could submit a `UserOperation` with a `paymasterAndData` field of 52 bytes containing only zeroes, causing `payingPaymaster` to resolve to `address(0)`. If `entityId` validation was missed or misconfigured during module installation, the `paymasters[entityId][msg.sender]` mapping could also resolve to `address(0)`, passing the check without a valid registered paymaster. Combining this with the `NativeTokenLimitModule` increases the risk: the attacker could bypass transaction limits set for no-paymaster calls, allowing unrestricted withdrawals and potentially draining account funds.

Recommendation: Add a check in the if-conditions of the modules to check that `paymasterAndData != 0`.

ALC-3 Missing Function Selector Check in `isIModuleFunction()`

Low · Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `c23dec6922d8f9f6e9e4d126b3dea5cd634e488f`.

File(s) affected: `KnownSelectorsLib`

Description: The `KnownSelectorsLib.isIModuleFunction()` is missing a check for `preSignatureValidationHook()`.

Recommendation: Add a check for that selector to the function.

ALC-4 Incorrect Masking of `deadline`

Low | Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `d4d9a23b15d856840a32c718c722ff6be0b12bf4`.

File(s) affected: `ModularAccountBase`

Description: In `ModularAccountBase._computeDeferredValidationInstallTypedDataHash()`, the `deadline` is incorrectly masked to keep the least significant 4 bytes rather 6, truncating the upper 2 bytes. This limits possible deadline encodings to 2^32 - 1 UNIX seconds, some time in the year 2106.

Recommendation: Mask the least 6 significant bytes for the `deadline`.

ALC-5 Improvements to the Execution Installation Flow

Low | Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
There is a workaround for module developers to initialize hooks in the  onInstall()  callback during
execution installation.
```

File(s) affected: `ExecutionInstallDelegate`,

Description: The `installExecution()` function in the `ExecutionInstallDelegate` contract is responsible for setting up both execution functions and execution hooks in a modular account. In case `installData` is provided, the modules to be installed should be invoking `ModuleInstallCommonsLib.onInstall()` on all installed entities, however, for the execution hooks, there is currently no mechanism to do that. The absence of this call is compliant with the standard. However, developers who intend to have stateful execution hooks may not expect this. Hence, it may be beneficial to add an option to add install data to execution hooks. In case `installData` is provided `ModuleInstallCommonsLib.onInstall()` would then perform the interface check and the `onInstall()` call.
Furthermore, the implementation only checks for the `IModule` interface when installing execution functions, which is insufficient for confirming that a module adheres to the necessary `IExecutionModule` functionality.

This oversight can result in the installation of modules that do not support the required interfaces, even if install data is provided.

Recommendation: Ensure that the `installExecution()` function does the following:
1. Consider providing the option for install data for execution hooks, hence invoking the `onInstall()` call for each execution hook if needed.
2. The `IExecutionModule` interface for modules is checked when setting execution functions.

ALC-6 Sender-Parameter in Certain Cases Will Always Be Entrypoint Rather than `UserOp.sender`

Informational | Acknowledged

Update

Marked as "Acknowledged" by the client.

File(s) affected: `ExecutionLib`, `ModularAccountBase`

Description: All execution hooks in UserOperation settings encode the EntryPoint as the `sender`, rather than the `sender` specified in the UserOperation struct.

Regular and validation-associated execution hooks are systematically invoked such that `msg.sender` resolves to the EntryPoint, thereby standardizing the context from which these hooks are triggered. This approach ensures uniform handling of sender identity across different execution paths, reinforcing the design intent to clearly signal the operational context to the module.

This is consistently applied in both direct and deferred validation processes, where the EntryPoint remains the `sender` even during runtime validations and deferred actions like signature validation. Such consistency supports the integrity of operational flows and minimizes confusion regarding the origin of execution hooks.

It is recommended to document this behavior explicitly to help developers understand that this pattern is a feature of system design, aimed at providing clarity and uniform context for execution hooks. Acknowledging this in the official documentation will aid in setting correct expectations and guide proper module implementation.

Recommendation: Add documentation that the `sender` parameter is expected to be EP for UserOp flow

ALC-7
Code-Less (Pre-)Signature Validation Modules May Result in Always-Passing Validations

Low | Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
It’s a very specific edge case, and requires serious user error. It’s also intentional to allow the installation of EOAs as modules for, as an example, specific direct calls. Furthermore, the selfdestruct case, where a faulty module self-destructs, is not likely to occur since the opcode no longer removes code on Ethereum-- with other chains likely to follow suit. Lastly, it's not an expected user flow to install an EOA as a hook, as their only use case is as a direct call validation.
```

File(s) affected: `ExecutionLib`

Description: Within the `ExecutionLib` library, various low-level calls such as `delegatecall()`, `staticcall()`, and `call()` are made without sufficient safeguards, specifically a lack of:

1. `returnsize()` checks
2. `extcodesize()` checks

This oversight is concerning, particularly in the context of EIP-7702, where an Externally Owned Account (EOA) could temporarily function as a module when invoking `installExecution()` or `installValidation()`. This scenario could allow subsequent hook calls to succeed improperly without verification.

A similar risk exists with standard module installations where no `installData` is used for callbacks or interface checks. Additionally, if a module self-destructs after installation, these unchecked calls may continue to pass, leading to potential security vulnerabilities.

Recommendation: Consider always having `exctodesize()` checks or `returndatasize()` checks for external calls.

ALC-8
Forwarding the `entityId` to the (Un-)Installation Callbacks

Low | Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
Acknowledged. We’ll keep the current interface as is for ecosystem alignment and since there may not be a single entity ID to pass in as a parameter (the manifest specifies the IDs to use, and there may be multiple different module functions installed at once).
```

File(s) affected: All Modules

Description: In the current module setup, most validation and validation-hook modules expect the `entityId` to be encoded within the `installData` passed to `onInstall()` and `uninstallData` passed to `onUninstall()`. This `entityId` links specific configurations and data within the module, allowing for stateful validation based on that `entityId`. This parameter is especially crucial for validation-associated hooks, while for regular execution hooks, it is typically less significant. Execution hooks are generally more stateless, and they currently do not involve an `onInstall()` callback for their installation, reducing the need for entityId encoding in these cases.

Encoding the `entityId` in this way introduces potential inconsistencies, as the value in the `validationConfig` may not necessarily match the `entityId` provided in `installData` during `installValidation()` calls. If a module is installed multiple times with different `entityId` values, there is a risk of unintended interference. For example, if module A with `entityId` A is uninstalled but the `uninstallData` includes `entityId` B—an identifier for another installation of the same module—this could cause unintended deregistration on the module or account side, effectively "bricking" both modules by removing important data in both instances.

While this risk is minor and mitigated by the fact that data fields are user-signed and validated, reducing the likelihood of intentional misuse, it still introduces a notable risk surface that could lead to user errors affecting account integrity.

Recommendation: Consider adding an `entityId` parameter to the `onInstall()` callback for modules. Perhaps this change can also be minimized to just validation (hook-) modules. The `entityId` could also be prepended into the `(un-)installData` on the account level before performing the callback to ensure consistency.

ALC-9
Installed Execution Functions Can Collide with Native Functions

Low | Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
Acknowledged.
```

File(s) affected: `ExecutionInstallDelegate`

Description: The ERC-6900 standard currently clearly states that installations of execution selectors that are clashing with ERC-4337 and ERC-6900 selectors that are native to the account should revert:

```
An execution function selector MUST not conflict with native ERC-4337 and ERC-6900 functions.
```

While it is checked for 4337 selectors, the native check is not performed. The assumption is that these selectors will not be invoked, as the native selectors will always be invoked instead. There is an extreme technical edge case, where a collision-clashed selector would suddenly become available in case of an account upgrade, where the native function was removed. Mainly, we want to point out the slight deviation from the standard.

Recommendation: We mainly want to raise awareness for this slight deviation.

ALC-10
Deferred Actions Allow Usage of Validation Functions in UserOp Context Without `isUserOpValidation` Flag

Informational | Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
Acknowledged. This is by design. Deferred actions allow the usage of validation functions in the user operation context without the  isUserOpValidation  flag. The signature validation function does not have to be a user operation validation function, as it is not used to sign over a user operation (only the deferred action), and is just verified in the user operation context. The validation function handling the user operation signature should have the  isUserOpValidation  flag set to true.
```

File(s) affected: `ModularAccountBase`

Description: Deferred actions enable actions on behalf of the account to be delayed right up until their execution is needed. This means that there are 2 calls that are bounded together during the `_validateUserOp()` function. As these 2 calls can have different validation functions, the outer user op validation function is verfiried to be correct with the correct `isUserOpValidation` validation flag. Whilst for the inner deferred action call, the validation function is a signature verification one, hence `isSignatureValidation` flag is validated. This is however not fully correct as the deferred action call is also done in the user op context, hence the signature validation function should also have the `isUserOpValidation` set correctly and verified.

Recommendation: If a validation function is intended to be used in the deferred action contest it should have both `isSignatureValidation` and `isUserOpValidation` flags set and those flags should be verified.

ALC-11
Potential Denial of Service Due to Misconfigured Time Ranges

Low Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `4e7e07308e9719f877690a269bf90624cde486ec`, `0bdb481ec41b4ca43fe0a48bb4a9e0ddfb951a95`, and `9c1ccda71efb20d4bc62989eb0d93071c28e5d9e`

Update

The fix ensures the validation `validUntil <= validAfter` is correctly implemented. Additionally, in this commit, `validUntil == 0` is now considered to mean valid indefinitely.

File(s) affected: `TimeRangeModule`

Description: The module allows account owners to set `validUntil` and `validAfter` timestamps for their operations. However, there is no validation to ensure that `validAfter` is less than or equal to `validUntil`. If an account owner inadvertently sets `validAfter` to a value greater than `validUntil`, the condition in `preRuntimeValidationHook()` will always revert.

Recommendation: Implement validation in the `setTimeRange()` function to ensure that `validAfter` is less than or equal to `validUntil`. If the condition is not met, the function should revert with an informative error message.

ALC-12
Incomplete List of Native Selectors Returned From `isNativeFunction()`

Informational Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `357cb8c1df4795503eb2db8ba72f2898c4d20407`, `3ba95477445835dba38306640218b04f124ce4b9` and `bf5feb9065ecbbf22a74866249dc196c91f85a33`.

The client provided the following explanation:

> `Added missing selectors, including account implementation specific native selectors as described.`

File(s) affected: `NativeFunctionDelegate`, `SemiModularAccountStorageOnly`, `ModularBase`

Description: `NativeFunctionDelegate.isNativeFunction()` does not include all native selectors of the `ModularBase` contract. Crucial selectors are missing such as:

- `invalidateDeferredValidationInstallNonce()`,
- `performCreate()` and
- `isValidSignature()`

are missing, and also less crucial ones, such as:

- `IERC721Receiver.onERC721Received.selector`
- `IERC1155Receiver.onERC1155Received.selector`
- `IERC1155Receiver.onERC1155BatchReceived.selector` and
- `IAccountExecute.executeUserOp.selector`

Furthermore, `SemiModularAccountStorageOnly` also exposes an `initialize()` function and `ModularAccount` exposes an `initializeWithValidation()` that should technically instead be exposed by an overridden `_isNativeFunction()`.

Recommendation: Consider implementing those function selectors to return `true` for checks for native selectors.

ALC-13
`getExecutionData()` Incorrectly Marks Native View Functions as Unlockable with Global Validation

Informational Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `5384fa9567a924082287fbcd4686d6fe8ad9efb8`.

File(s) affected: `ModularAccountView`, `IModularAccountView (reference-implementation)`,

Description: In the `getExecutionData()` function, all selectors that return `true` for `_isNativeFunction()` are marked as unlockable by global validation (`allowGlobalValidation = true`). However, this logic is flawed because `_isNativeFunction()` also returns `true` for many view functions, which are permissionless and not intended to be unlocked for global validation.

Recommendation: Update the logic to ensure that only native selectors that meet the condition `_globalValidationAllowed() == true` are returned with `module = address(this)` and `allowGlobalValidation = true`. Additionally, consider adding a separate condition for native selectors where `_globalValidationAllowed() == false`, returning `module = address(this)` and `skipRuntimeValidation = true` for those cases.

ALC-14
Unrelated Deferred Actions Could Be Dropped by Bundler to Apply Unused Gas Penalty to Account

Informational | Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
Implicitly fixed by the mitigation in  bf9e53acb20d58efdd483e8ebf41ded8c96be43b
```

File(s) affected: `ModularAccountBase`

Description: If some UserOperation were to be expected to include some independent deferred action that is not necessary for a passing validation phase, a bundler could drop that deferred action and making the account pay the 10% penalty for the unused gas that was intended to be consumed by the deferred action. However, as there is limited upside here and as deferred actions are most likely always needed to be included for such a UserOperation, this is only informational.

Recommendation: We mainly want to raise awareness for this possibility.

ALC-15 Incomplete Module Data Cleanup on Uninstall

Informational | Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
Accounts should be aware of this caveat, we don't want to force an account to call  onUninstall()  as
that may lead to issues, the account needs to be able to uninstall without calling the module. Modules
that require such a cleanup should communicate this, such that data is provided to call  onUninstall() .
```

File(s) affected: `AllowlistModule`, `NativeTokenLimitModule`, `PaymasterGuardModule`, `TimeRangeModule`, `SingleSignerValidationModule`, `WebAuthnValidationModule`

Description: The `onUninstall()` functions in various modules may leave behind residual state data linked to `msg.sender` after uninstallation, especially since the external call is only performed if `onInstallData` is specified. While this incomplete cleanup is acknowledged in the documentation, it could pose security risks or lead to unexpected behavior in the case where the module is reinstalled with the same entityId. The danger is that previously set permissions or data may be unintentionally reused.

Below is an overview of potential residual data for each module:

- `AllowlistModule`
  - `allowlist` entries
  - `selector` mappings
- `NativeTokenLimitModule`
  - `limits`
  - `specialPaymasters`
- `PaymasterGuardModule`
  - `paymasters`
- `TimeRangeModule`
  - `timeRanges`

- `SingleSignerValidationModule`
  - `signers`
- `WebAuthnValidationModule`
  - `signers`

Exploit Scenario:
If, for example, the `AllowlistModule` is uninstalled without fully clearing all allowlist entries, and then later reinstalled, the old allowlist data for `entityId` might remain. This could enable unauthorized access to functions or entities that were previously allowed but should no longer be accessible, leading to potential privilege escalation.

Recommendation: Implement a mechanism that fully clears the allowlist state for a given `entityId` during uninstallation, regardless of the provided `AllowlistInit` data.

ALC-16

`executeBatch()` Can Be Used to Bypass Other Selector's Validation Hook Execution

Informational | Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
We will commit to clearly describing the implications of global validations and modules allowed to
validate  executeBatch() .
```

File(s) affected: `ModularAccountBase`

Description: With ERC-6900 v0.8, self-calls to the account are strictly reserved to indicate authorization to invoke some selector. Therefore, the ways in which the account can call itself are very important. The `executeBatch()` function is such a case. In case it performs a self-call, that call's selector is analyzed and has to also be able to be validated by the current validation. However, the actual validation, including the validation associated execution hooks, is then not performed, if invoked via a self-call, only the hooks for the outer `executeBatch()` call is performed.

This is known and documented, but we wanted to emphasize it in the report, as well. If some entity has access to `executeBatch()`, but also to some by hooks heavily protected execution function, the hooks of that other selector can be bypassed by performing an `executeBatch()` self-call encoding that other execution function.

Recommendation: We mainly want to raise awareness for this. This fact should be clearly outlined in user documentation. Access for validation modules to `executeBatch()` should be treated with extreme care.

ALC-17

Deactivated Fallback Signer in `SemiModularAccountBase` Returned as Global Validation

Informational | Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
We will document this, but it does not seem worth the gas tradeoff of an additional (even if warm)
storage read.
```

File(s) affected: `SemiModularAccountBase`

Description: For semi-modular accounts, In case the fallback signer has been disabled through calls to `updateFallbackSignerData()`, the default validation is still returned as global validation in the `_isValidationGlobal()` function.

The impact is minimal and does not enable a passing validation with the disabled fallback signer, because the `_getFallbackSigner()` call ultimately attempting to fetch the signer leads to a revert. However, as in another issue we recommend `getExecutionData()` to contain a global validation check, it might make sense to adjust the function to return false for disabled fallback signers.

Recommendation: Adjust the `_isValidationGlobal()` function to return false for disabled fallback signers.

ALC-18

Reduce Possibilities of Execution Reverts for `NativeTokenLimitModule`

Informational | Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

> `Removing preExecutionHook() in NativeTokenLimitModule will require adding value spend logic in preUserOperationValidationHook(), as well as implementing preRuntimeValidationHook() and postExecutionHook(). Even though the recommendation would decrease the likelihood of execution reverts for user operations, it adds more complexity to set up the hooks. Therefore, we will keep as it is.`

File(s) affected: `NativeTokenLimitModule`

Description: The `NativeTokenLimitModule` provides a pre-execution hook that can use up the allowance for native token transfers under certain execution functions. As native transfers are rolled back in a reverting execution phase, it is a check that is correctly performed not as part of the validation phase, but in the execution phase, so that the spending limit is not changed in case a user operation, containing a native token spend, reverts in execution.

However, this enables the possibility of predictable reverts in the execution phase, as the details of the native token spent remain fully unchecked in the validation phase. The possibility of execution reverts due to exceeded limits could be reduced, at least if only a single UserOp of that account is being bundled, with this slightly more restrictive approach:

- Add a pre-validation hook to the module that reverts in case the checks currently performed in the `preExecutionHook()` are failing. Remove the current `preExecutionHook()` module and replace it with a validation associated post-only execution hook that performs the state change on the `limit` value.
- Place the `preUserOpValidationHook()` that is doing gas limit checks on the account into a separate `GasLimitModule`.

While this requires the configuration of two separate native token limit values (gas limit and token transfer limit), we deem this to be an improvement to the design.

Recommendation: Consider implementing these changes.

ALC-19

`NativeTokenLimitModule` Does Not Return `IValidationHookModule` As Supported Interface

Informational | Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `06cd4996d20fbb6f02bd11402cbe89f5285544b5`.
The client provided the following explanation:

> `We've added the missing interface ID.`

File(s) affected: `NativeTokenLimitModule`

Description: The `NativeTokenLimitModule.supportsInterface()` does not return `true` for `interfaceId == type(IValidationHookModule).interfaceId`, even though it provides validation hooks.

Recommendation: Return true for `interfaceId == type(IValidationHookModule).interfaceId`.

ALC-20

Consider adding a `forceUninstallation` Flag that Requires `onInstall()` Callback Success

Informational | Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

> `If needed, an account can directly call the onUninstall() function on a module after the fact. We suspect this to be an edge case and functionally similar to having an additional flag.`

File(s) affected: `ModuleInstallCommonsLib`

Description: In our opinion, it is not optimal that based on user input issues, the `onUninstall()` callback to the module might not succeed, but cannot be re-attempted due to the modules any longer being registered on the account-side. While it is very reasonable to have a `try-catch` flow around the `onUninstall()` callback in case the module is blocking uninstallation, this should perhaps be specifically intended by a user, e.g. with a `forceUninstallation` flag indicating that a reverting module-callback is acceptable.

Recommendation: Consider adding a flag value with which a user can indicate if an uninstallation process should be forced or not.

ALC-21 Incorrect Casting in `toSetValue` Functions

Informational | Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `4027ce4ce1da07dd9416f53535e0cfe72fb1c887`.
The client provided the following explanation:

```
We've addressed the issue and properly casted to bytes31.
```

File(s) affected: `AccountStorage`, `LinkedListSetLib`

Description: The `LinkedListSetLib` library from the reference implementation has been adjusted to work with an underlying `SetValue` of type `bytes31` instead of `bytes30`. However, the castings in the `AccountStorage.toSetValue()` function have not been adjusted. Importantly, no information gets lost due to the erroneous casting, as the wrapped values are always less than 25 bytes long and also of type bytes, so only unused padding is omitted.

Recommendation: Adjust the castings from `bytes30` to `bytes31`. This should also be done for the test cases.

ALC-22 Individual Validation Hooks Cannot Be Removed From Configurations

Informational | Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
Ack. We've investigated this feature previously, and while it would be a UX benefit, the added
complexity, and likely gas bump, from its implementation lead us not to implement it. We'll continue to
keep this in mind and see if the need arises in a future version.
```

File(s) affected: `ExecutionInstallDelegate`, `ModuleInstallCommonsLib`

Description: While it is possible to remove installed execution hooks from existing execution configurations individually, the same is not possible for validation hooks. Validation configurations are always removed in their entirety, with all associated hooks. This is in contrast with the possible hook extension of existing validation or execution configs.

As additional validation hooks can be added, perhaps by another party that is authorized to do so, but not removed, this possibly creates a minor UX inconvenience, as an validation installation has to be fully removed and re-installed without one hook to mimic a single hook deinstallation.

Recommendation: Consider if the removal of individual validation hooks is worth supporting.

ALC-23 Unenforced Off-Chain Checks for Modules and Modular Account

Informational | Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

```
Ack. No fixes planned. Aware of the risks and will document and communicate with users. Specifically
for case number 2, there are interesting use cases enabled by allowing  skipRuntimeValidation == true ,
```

> so wanted to keep it as it is and communciate the risks with users.

File(s) affected: `All Files`

Description: While technically not vulnerabilities, multiple risks in global validation and hook management create risks of unauthorized access, incomplete uninstallation, and potential account bricking:

1. Global validation modules inherently have access to sensitive native functions (e.g., `executeBatch()`, `upgradeToAndCall()`). As these functions have high privileges, end users must exercise caution when assigning validation modules globally.
2. Pre-runtime validation hooks do not necessarily need to be addresses with code and would pass for all parameters, as a low level `call()` is made without any `codesize` checks or return data checks
3. Execution hooks may not be fully removed during uninstallation, leaving residual hooks on fully removed selectors.
4. The last validation function, given sufficient privileges, can be uninstalled, which could effectively brick the account.

Recommendation: Document these risks clearly.

Auditor Suggestions

S1 Support for `0` Value for `validUntil` Encoding in `TimeRangeModule`

Fixed

> **Update**
> Marked as "Fixed" by the client.
> Addressed in: `0bdb481ec41b4ca43fe0a48bb4a9e0ddfb951a95`.

File(s) affected: `TimeRangeModule`

Description: The TimeRangeModule currently inadequately supports the `0` value for the `validUntil` field, which as specified be EIP-4337, resembles a never expiring time window. While this is properly supported for the UserOperation flow, the equivalent runtime flow would cause reverts, as a value of `0` for the registered `TimeRange` is guaranteed to be greater than `block.timestamp`, therefore always reverting.

Recommendation: Adjust the `preRuntimeValidationHook()` for the special case of `timeRange.validAfter = 0`.

S2 Remarks on Comments / Documentation / Function Names

Fixed

> **Update**
> Marked as "Fixed" by the client.
> Addressed in: `7a14e6c86d1ee42ed91ba84589821ae89e5c72f6`.

File(s) affected: `NativeFunctionDelegate`, `ModularAccountBase`, `NativeTokenLimitModule`, `MemManagementLib`, `ExecutionLib`

Description: We note the following:

- The code contains multiple instances to a native `performCreate2()` function, which however no longer is present.
- `ModularAccountBase.sol#L692` contains a comment that indicates that the call to `doPreHooks()` contains a call to just the execution hooks associated with that selector. This is however inaccurate, as it might also contain validation hooks in case of direct call validation.
- A comment in the `NativeFunctionDelegate` contract states that the contract is delegatecalled into, which is however not the case; a regular, external call is performed.
- The `LinkedListSetLib.sol` file has been adapted from the reference implementation to work with an underlying `SetValue` of type `bytes31` instead of `bytes30`, but some comments in `MemManagementLib.sol` still suggest the `bytes30` type.
- The comment for the `ModularAccount.initializeWithValidation()` function mentions that the call will install the given `validationConfig` as a global validation. This is however not the case and depends on the flag encoding in the `ValidationConfig` bytes.
- The comment on the `ModularAccountBase.wrapNativeFunction()` modifier is outdated, as `invalidateDeferredValidationInstallNonce()` is not mentioned, but the deprecated `performCreate2()` function is.
- The comment in `ExecutionLib.sol#L357` incorrectly states that the length of the authorization field is added. However, this is not the case, but instead the `account` field is again accounted for for the absolute offset.
- The comment block in `ModularAccountBase._computeDeferredValidationInstallTypedDataHash()` mentions that the `_INSTALL_VALIDATION_TYPEHASH` is being encoded in the struct hash, this is however not the case, it is the `_DEFERRED_ACTION_TYPEHASH`.
- `_computeDeferredValidationInstallTypedDataHash()` seems a bit misnamed, as a deferred action is not necessarily a install action.
- The comment in `ModularAccountBase._checkExecuteBatchValidationApplicability()` incorrectly suggests in the commented Solidity code that the assembly code is still doing bytes slicing with `callData[4:]`, which has however already been done

in a prior step.

Recommendation: Consider correcting these code comments.

S3 Gas Optimizations

Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `e7ea249492fd9cccd88e1f8843fcb92495e47983`.

File(s) affected: `AllowListModule`, `NativeTokenLimitModule`, `ModularAccountBase`, `SemiModularAccountBase`, `ModuleManagerInternals`

Description: We note the following:

- In for loops, array length can be cached, as well as having the integer incremented via `++i` instead of `i++` for minor gas improvements (AllowListModule, NativeTokenLimitModule)
- Instead of deploying a new instance of the `ExecutionInstallDelegate` contract in the constructor of `ModularAccountBase`, consider using a global singleton contract that is stored as a constant variable.
- In `ModularAccountBase.executeBatch()` it would be cheaper to have a separate flow for `needReturnData == false`, so that the condition is not rechecked for each iteration.
- In `ModularAccountBase._doRuntimeValidation()` and `_isValidSignature()`, `bytes calldata currentAuthSegment;` can be allocated once outside the for loop.
- `ModularAccountBase.isValidSignature()` can be marked as `external`.
- `SemiModularAccountBase._hashStructReplaySafeHash()` should be imported from the `ReplaySafeWrapper` file.
- In `ModuleManagerInternals._installValidation()`, declare the `moduleEntity` variable and then use it as a key instead of calling `validationConfig.moduleEntity()` twice.

Recommendation: Consider implementing these changes.

S4 Ownership Can Be Renounced

Fixed

Update

The `renounceOwnership()` function now reverts on calls.

Update

Marked as "Fixed" by the client.
Addressed in: `ed5c3e89695e4bb4b2b1ff358e072479808a54cd`.

File(s) affected: `AccountFactory`

Description: If the owner renounces their ownership, all ownable contracts will be left without an owner. Consequently, any function guarded by the `onlyOwner` modifier will no longer be able to be executed.

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).

S5 Critical Role Transfer Not Following Two-Step Pattern

Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `b796c55b840cbd1280b7e899a3db1ec833258c30`.

File(s) affected: `AccountFactory`

Description: The owner of the contracts can call `transferOwnership()` to transfer the ownership to a new address. If an uncontrollable address is accidentally provided as the new owner address then the contract will no longer have an active owner, and functions with the `onlyOwner` modifier can no longer be executed.

Recommendation: Consider using OpenZeppelin's `Ownable2Step` contract to adopt a two-step ownership pattern in which the new owner must accept their position before the transfer is complete.

S6

`uninstallExecution()` Should Verify that Hook or a Function Was Present Before

Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `559a8593845e315d28cea7b6d448cdb66871a82c` .

File(s) affected: `ExecutionInstallDelegate`

Description: When calling `uninstallExecution()` the uninstallation of the hooks is performed based on a user-provided `ExecutionManifest` . However, it remains unchecked whether an execution hook or an execution function was actually present before. This might lead to inconsistent state in case any incorrect codings are being done.

Recommendation: In `ExecutionInstallDelegate._removeExecHooks()` , consider checking that the return value is not equal to `false` . Furthermore, consider checking in `_removeExecutionFunction` that `_executionStorage.module != address(0x0)` before zero-ing out the storage.

S7

`getExecutionData()` Returns Execution Hooks for Functions that Will Never Execute Them

Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `3ba95477445835dba38306640218b04f124ce4b9` and `95c8b99ea8ebafed742a2b478818a6bf91762b71` .

File(s) affected: `ModularAccountView`

Description: The `getExecutionData()` function returns the properties of the execution function along with execution hooks that are installed on that function. But we have native functions like `executeUserOp()` and `executeWithRuntimeValidation()` that have no `wrapNativeFunction` modifier meaning that they will never run the hooks even if they are installed. It could be misleading that the `getExecutionData()` function would return the hooks for these functions and hence users could assume that they would be executed, but that would actually never happen.

Recommendation: Consider not returning execution hooks for functions that will never execute them.

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).

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition

or potential vulnerability that was not within the scope of the review.

Files

- `0aa...bf0 ./ReplaySafeWrapper.sol`
- `372...bd2 ./ModuleEIP712.sol`
- `8b5...2d7 ./ValidationConfigLib.sol`
- `0e7...976 ./SparseCalldataSegmentLib.sol`
- `13b...3d2 ./ModuleEntityLib.sol`
- `1b7...298 ./HookConfigLib.sol`
- `de1...d6a ./Constants.sol`
- `d86...ba9 ./src/factory/AccountFactory.sol`
- `e73...064 ./src/helpers/SignatureType.sol`
- `5e2...3ce ./src/helpers/ValidationResHelpers.sol`
- `de1...d6a ./src/helpers/Constants.sol`
- `89e...fd9 ./src/helpers/NativeFunctionDelegate.sol`
- `9c1...abc ./src/helpers/ExecutionInstallDelegate.sol`
- `f77...13d ./src/modules/ModuleBase.sol`
- `8fa...166 ./src/modules/validation/WebAuthnValidationModule.sol`
- `861...781 ./src/modules/validation/SingleSignerValidationModule.sol`
- `ed5...413 ./src/modules/permissions/NativeTokenLimitModule.sol`
- `7e3...de6 ./src/modules/permissions/TimeRangeModule.sol`
- `f20...d09 ./src/modules/permissions/PaymasterGuardModule.sol`
- `06a...673 ./src/modules/permissions/AllowlistModule.sol`
- `9a3...93f ./src/account/SemiModularAccountBase.sol`
- `4ba...660 ./src/account/TokenReceiver.sol`
- `290...f07 ./src/account/SemiModularAccountStorageOnly.sol`
- `598...b01 ./src/account/SemiModularAccountBytecode.sol`
- `7d2...643 ./src/account/ModuleManagerInternals.sol`
- `822...438 ./src/account/ModularAccount.sol`
- `b88...c9c ./src/account/ModularAccountBase.sol`
- `317...b91 ./src/account/ModularAccountView.sol`
- `a50...e6a ./src/account/AccountStorageInitializable.sol`
- `695...3ad ./src/account/AccountStorage.sol`
- `14a...27b ./src/account/AccountBase.sol`
- `bbf...7bc ./src/libraries/ExecutionLib.sol`
- `362...c22 ./src/libraries/ModuleInstallCommonsLib.sol`
- `3b3...f1c ./src/libraries/KnownSelectorsLib.sol`
- `03b...d87 ./src/libraries/ERC7739ReplaySafeWrapperLib.sol`
- `698...9d3 ./src/libraries/MemManagementLib.sol`
- `a31...025 ./src/libraries/LinkedListSetLib.sol`

Tests

- `03b...a64 ./AccountFactory.t.sol`
- `aac...054 ./ValidationResHelpers.t.sol`
- `7a8...a85 ./WebAuthnValidationModule.t.sol`
- `c94...18f ./AllowlistERC20TokenLimit.t.sol`
- `9c0...c5a ./PaymasterGuardModule.t.sol`
- `8f2...45a ./AllowlistModule.t.sol`
- `479...20a ./TimeRangeModule.t.sol`
- `c78...adc ./NativeTokenLimitModule.t.sol`
- `403...f18 ./ReplaceModule.t.sol`
- `238...ff0 ./ValidationAssocHooks.t.sol`
- `062...693 ./ValidationIntersection.t.sol`
- `2c3...d50 ./PHCallBuffers.t.sol`
- `d64...ce3 ./DeferredValidation.t.sol`

- 1ed...a50 ./RTCallBuffer.t.sol
- 0bb...d60 ./ModularAccount.t.sol
- 167...351 ./UpgradeToSma.t.sol
- bc4...fdb ./SigCallBuffer.t.sol
- a48...315 ./SelfCallAuthorization.t.sol
- ac2...f77 ./ExecutionInstallDelegate.t.sol
- 8b1...889 ./SemiModularAccountDirectCall.t.sol
- fc0...127 ./AccountExecHooks.t.sol
- b73...ba2 ./PerHookData.t.sol
- 7a0...1b7 ./AccountReturnData.t.sol
- 873...85f ./SMASpecific.t.sol
- cbe...33f ./UOCallBuffer.t.sol
- 62b...41e ./PermittedCallPermissions.t.sol
- 2f3...da8 ./GlobalValidationTest.t.sol
- 0e0...a5e ./DeferredAction.t.sol
- 57d...cd3 ./DirectCallsFromModule.t.sol
- 188...499 ./TokenReceiver.t.sol
- ced...71b ./PostHookData.t.sol
- 417...c17 ./HookOrdering.t.sol
- 329...f1e ./MultiValidation.t.sol
- fb1...bc6 ./ModularAccountView.t.sol
- 714...fba ./ValidateSetup.t.sol
- 1d1...769 ./TestConstants.sol
- be4...313 ./ModuleSignatureUtils.sol
- 318...7cc ./StorageAccesses.sol
- c0e...6cd ./OptimizedTest.sol
- 2d3...f0a ./CustomValidationTestBase.sol
- 0ae...73a ./AccountTestBase.sol
- f40...967 ./AccountStorage.t.sol
- 59f...4c3 ./LinkedListSetLib.t.sol
- 810...61f ./KnownSelectorsLib.t.sol
- ab9...e56 ./SparseCalldataSegmentLib.t.sol
- 657...382 ./LinkedListSetLibInvariants.t.sol
- e17...e88 ./LinkedListSetHandler.sol
- c84...ba9 ./MockRevertingConstructor.sol
- a57...9ad ./MockERC1155.sol
- 25f...fc6 ./MockERC20.sol
- a42...3d8 ./MockDecoder.sol
- 96a...638 ./MockTokenPaymaster.sol
- 0f1...dfb ./Counter.sol
- 40d...bae ./MockDiamondStorageContract.sol
- faf...b5a ./MockInterface.sol
- b40...952 ./MockERC721.sol
- 340...64b ./MockDeployment.sol
- 39a...231 ./ValidationModuleMocks.sol
- 8da...c63 ./MockCountModule.sol
- 296...0a3 ./MockModule.sol
- 676...7b6 ./ComprehensiveModule.sol
- 3c1...361 ./PermittedCallMocks.sol
- 8bd...015 ./HookOrderCheckerModule.sol
- f9f...eb8 ./MockAccessControlHookModule.sol
- 131...dff ./MockSMADirectFallbackModule.sol
- 4f7...ae9 ./DirectCallModule.sol
- 88d...874 ./MockExecutionInstallationModule.sol
- 940...82e ./ReturnDataModuleMocks.sol

- b9c...c76 ./test/modules/TokenReceiverModule.t.sol
- e52...625 ./test/modules/validation/SingleSignerValidationModule.t.sol
- e2f...11f ./test/modules/permissions/AllowlistModule.t.sol
- 0e5...c9a ./test/modules/permissions/NativeTokenLimitModule.t.sol
- 1be...79e ./test/modules/permissions/ERC20TokenLimitModule.t.sol
- c1a...b92 ./test/account/ReferenceModularAccount.t.sol
- e60...468 ./test/account/ReplaceModule.t.sol
- 0a9...f05 ./test/account/ValidationIntersection.t.sol
- 794...a8e ./test/account/AccountStorage.t.sol
- fb5...567 ./test/account/SelfCallAuthorization.t.sol
- fc0...19a ./test/account/AccountExecHooks.t.sol
- 476...63e ./test/account/PerHookData.t.sol
- 865...568 ./test/account/AccountReturnData.t.sol
- 729...f4b ./test/account/PermittedCallPermissions.t.sol
- 3c4...674 ./test/account/AccountFactory.t.sol
- bbf...f6c ./test/account/GlobalValidationTest.t.sol
- dfc...e1a ./test/account/DirectCallsFromModule.t.sol
- 600...4de ./test/account/MultiValidation.t.sol
- 79d...946 ./test/account/ModularAccountView.t.sol
- e6a...8f4 ./test/comparison/CompareSimpleAccount.t.sol
- 1e0...ec0 ./test/script/Deploy.s.t.sol
- 532...44d ./test/script/DeployAllowlistModule.s.t.sol
- 192...e1a ./test/utils/TestConstants.sol
- 73a...c16 ./test/utils/ModuleSignatureUtils.sol
- 579...7d6 ./test/utils/OptimizedTest.sol
- 321...4c4 ./test/utils/CustomValidationTestBase.sol
- d86...dd4 ./test/utils/AccountTestBase.sol
- e6c...ca3 ./test/libraries/ValidationConfigLib.t.sol
- cc3...d58 ./test/libraries/ModuleEntityLib.t.sol
- a71...a51 ./test/libraries/KnowSelectorsLib.t.sol
- 19c...363 ./test/libraries/ModuleStorageLib.t.sol
- 02c...795 ./test/libraries/HookConfigLib.t.sol
- 23e...9c1 ./test/libraries/SparseCalldataSegmentLib.t.sol
- d01...b20 ./test/mocks/MockERC1155.sol
- 122...726 ./test/mocks/ContractOwner.sol
- d9d...a5c ./test/mocks/MockModule.sol
- a78...891 ./test/mocks/MockERC20.sol
- cc1...aaa ./test/mocks/SingleSignerFactoryFixture.sol
- 06f...cdd ./test/mocks/Counter.sol
- 5a0...2bd ./test/mocks/MockDiamondStorageContract.sol
- 6b2...1a5 ./test/mocks/MockERC721.sol
- bc1...977 ./test/mocks/Counter.t.sol
- ccb...a2f ./test/mocks/modules/ValidationModuleMocks.sol
- baf...2b9 ./test/mocks/modules/ComprehensiveModule.sol
- 281...1ec ./test/mocks/modules/PermittedCallMocks.sol
- 40a...3aa ./test/mocks/modules/MockAccessControlHookModule.sol
- e58...d10 ./test/mocks/modules/DirectCallModule.sol
- 757...79f ./test/mocks/modules/ReturnDataModuleMocks.sol

Test Suite Results

Test suite was comprehensive and robust with 29 test suits for the `modular-account` repo.

Ran using `forge test` in both cases.

Fix-Review Update: The test suite expanded from 256 to 284 tests with several key additions, including tests for proper handling of insufficient return data during signature validation; behavior when validation functions return incomplete data; the creation and addressing of WebAuthn-based accounts; and the correct identification and handling of ERC-4337 function selectors.

```
% forge test
```

Modular Account

```
Compiling 212 files with Solc 0.8.26
Solc 0.8.26 finished in 9.93s
Compiler run successful!
Analysing contracts...
Running tests...

Ran 3 tests for test/libraries/AccountStorage.t.sol:AccountStorageTest
[PASS] test_accountStorage_revertOnBadDisableInitializers() (gas: 36643)
[PASS] test_storageSlotImpl() (gas: 8136)
[PASS] test_storageSlotProxy() (gas: 36217)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 347.67µs (128.62µs CPU time)

Ran 7 tests for test/factory/AccountFactory.t.sol:AccountFactoryTest
[PASS] test_createAccount() (gas: 397828)
[PASS] test_createAccountAndGetAddress() (gas: 408982)
[PASS] test_createWebAuthnAccount() (gas: 201803)
[PASS] test_createWebAuthnAccountAndGetAddress() (gas: 207235)
[PASS] test_multipleDeploy() (gas: 402084)
[PASS] test_multipleDeployWebAuthn() (gas: 203319)
[PASS] test_withdraw() (gas: 937535)
Suite result: ok. 7 passed; 0 failed; 0 skipped; finished in 4.53ms (1.77ms CPU time)

Ran 4 tests for test/account/GlobalValidationTest.t.sol:GlobalValidationTest
[PASS] test_globalValidation_failsOnSelectorApplicability() (gas: 695925)
[PASS] test_globalValidation_runtime_simple() (gas: 624162)
[PASS] test_globalValidation_runtime_updateFallbackSignerData() (gas: 281792)
[PASS] test_globalValidation_userOp_simple() (gas: 931959)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 6.83ms (4.19ms CPU time)

Ran 2 tests for test/account/ReplaceModule.t.sol:UpgradeModuleTest
[PASS] test_upgradeModuleExecutionFunction() (gas: 5500098)
[PASS] test_upgradeModuleValidationFunction() (gas: 8200765)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 7.13ms (4.73ms CPU time)

Ran 10 tests for test/modules/PaymasterGuardModule.t.sol:PaymasterGuardModuleTest
[PASS] test_install_i() (gas: 328260)
[PASS] test_onInstall() (gas: 377442)
[PASS] test_onUninstall() (gas: 334134)
[PASS] test_preRuntimeValidationHook_success() (gas: 324745)
[PASS] test_preUserOpValidationHook_fail() (gas: 405144)
[PASS] test_preUserOpValidationHook_failWithInvalidData() (gas: 397005)
[PASS] test_preUserOpValidationHook_failWithValidationData() (gas: 414435)
[PASS] test_preUserOpValidationHook_success() (gas: 398602)
[PASS] test_userOp_fail_i() (gas: 416180)
[PASS] test_userOp_success_i() (gas: 437668)
Suite result: ok. 10 passed; 0 failed; 0 skipped; finished in 10.37ms (6.77ms CPU time)

Ran 5 tests for test/account/AccountReturnData.t.sol:AccountReturnDataTest
[PASS] test_returnData_authorized_exec() (gas: 2726662)
[PASS] test_returnData_execFromModule_fallback() (gas: 2683748)
[PASS] test_returnData_executeBatch() (gas: 2734268)
```

```
[PASS] test_returnData_fallback() (gas: 2667157)
[PASS] test_returnData_singular_execute() (gas: 2711271)
Suite result: ok. 5 passed; 0 failed; 0 skipped; finished in 11.20ms (7.83ms CPU time)

Ran 7 tests for test/modules/AllowlistERC20TokenLimit.t.sol:AllowlistERC20TokenLimitTest
[PASS] test_install() (gas: 1227094)
[PASS] test_runtime_executeBatchLimit() (gas: 1501325)
[PASS] test_runtime_executeLimit() (gas: 1386140)
[PASS] test_userOp_executeBatchLimit() (gas: 1432560)
[PASS] test_userOp_executeBatch_approveAndTransferLimit() (gas: 1474470)
[PASS] test_userOp_executeBatch_approveAndTransferLimit_fail() (gas: 1556254)
[PASS] test_userOp_executeLimit() (gas: 1377808)
Suite result: ok. 7 passed; 0 failed; 0 skipped; finished in 15.59ms (12.99ms CPU time)

Ran 1 test for test/utils/ValidateSetup.t.sol:ValidateSetupTest
[PASS] test_deployedEntryPoint() (gas: 96898)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 555.38µs (440.96µs CPU time)

Ran 9 tests for test/account/AccountExecHooks.t.sol:AccountExecHooksTest
[PASS] test_execHookPair_install() (gas: 2995918)
[PASS] test_execHookPair_run() (gas: 6077991)
[PASS] test_execHookPair_uninstall() (gas: 6087712)
[PASS] test_postOnlyExecHook_install() (gas: 2956209)
[PASS] test_postOnlyExecHook_run() (gas: 5969235)
[PASS] test_postOnlyExecHook_uninstall() (gas: 6008097)
[PASS] test_preExecHook_install() (gas: 2956208)
[PASS] test_preExecHook_run() (gas: 5973353)
[PASS] test_preExecHook_uninstall() (gas: 6008120)
Suite result: ok. 9 passed; 0 failed; 0 skipped; finished in 23.75ms (20.13ms CPU time)

Ran 5 tests for test/account/DeferredAction.t.sol:DeferredActionTest
[PASS] test_deferredAction_approveERC20InInitcode() (gas: 3537902)
[PASS] test_deferredAction_privilegeEscalationPrevented_executeBatch() (gas: 101658)
[PASS] test_deferredAction_privilegeEscalationPrevented_executeSingle() (gas: 94347)
[PASS] test_deferredAction_validationApplicabilityCheck() (gas: 139505)
[PASS] test_deferredAction_validationAssociatedExecHooks() (gas: 4381743)
Suite result: ok. 5 passed; 0 failed; 0 skipped; finished in 12.62ms (8.06ms CPU time)

Ran 2 tests for test/account/PermittedCallPermissions.t.sol:PermittedCallPermissionsTest
[PASS] test_permittedCall_Allowed() (gas: 1683928)
[PASS] test_permittedCall_NotAllowed() (gas: 1684046)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 4.55ms (1.54ms CPU time)

Ran 9 tests for test/account/DeferredValidation.t.sol:DeferredValidationTest
[PASS] test_deferredValidation_deployed() (gas: 2376748)
[PASS] test_deferredValidation_deployedWithValidationAssociatedExecHooks() (gas: 4166098)
[PASS] test_deferredValidation_initCode() (gas: 2634659)
[PASS] test_fail_deferredValidation_invalidDeferredValidationSig() (gas: 2240189)
[PASS] test_fail_deferredValidation_invalidSig() (gas: 2025850)
[PASS] test_fail_deferredValidation_nonceInvalidated() (gas: 2242982)
[PASS] test_fail_deferredValidation_nonceUsed() (gas: 2478301)
[PASS] test_fail_deferredValidation_pastDeadline() (gas: 2242333)
[PASS] test_fail_deferredValidation_withValidationHooks() (gas: 2138216)
Suite result: ok. 9 passed; 0 failed; 0 skipped; finished in 31.63ms (29.02ms CPU time)

Ran 19 tests for test/account/PerHookData.t.sol:PerHookDataTest
[PASS] test_fail1271AccessControl_badSigData() (gas: 1694236)
[PASS] test_fail1271AccessControl_noSigData() (gas: 1686472)
[PASS] test_failAccessControl_badIndexProvided_runtime() (gas: 1706128)
[PASS] test_failAccessControl_badIndexProvided_userOp() (gas: 1805609)
[PASS] test_failAccessControl_badSigData_runtime() (gas: 1703171)
[PASS] test_failAccessControl_badSigData_userOp() (gas: 1805592)
[PASS] test_failAccessControl_badTarget_runtime() (gas: 1707511)
```

```
[PASS] test_failAccessControl_badTarget_userOp() (gas: 1803327)
[PASS] test_failAccessControl_indexOutOfOrder_runtime() (gas: 1848503)
[PASS] test_failAccessControl_indexOutOfOrder_userOp() (gas: 1945901)
[PASS] test_failAccessControl_noSigData_runtime() (gas: 1696165)
[PASS] test_failAccessControl_noSigData_userOp() (gas: 1794395)
[PASS] test_failPerHookData_nonCanonicalEncoding_runtime() (gas: 1686705)
[PASS] test_failPerHookData_nonCanonicalEncoding_userOp() (gas: 1784804)
[PASS] test_pass1271AccessControl() (gas: 1713502)
[PASS] test_passAccessControl_runtime() (gas: 1775248)
[PASS] test_passAccessControl_twoHooks_runtime() (gas: 1929108)
[PASS] test_passAccessControl_twoHooks_userOp() (gas: 2225557)
[PASS] test_passAccessControl_userOp() (gas: 1777396)
Suite result: ok. 19 passed; 0 failed; 0 skipped; finished in 27.80ms (22.42ms CPU time)

Ran 1 test for test/account/ExecutionInstallDelegate.t.sol:ExecutionInstallDelegateTest
[PASS] test_fail_directCall_delegateCallOnly() (gas: 1706561)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 2.33ms (114.08µs CPU time)

Ran 12 tests for test/account/SelfCallAuthorization.t.sol:SelfCallAuthorizationTest
[PASS] test_batchAction_allowed_execUserOp() (gas: 2191255)
[PASS] test_batchAction_allowed_runtime() (gas: 1905278)
[PASS] test_batchAction_allowed_userOp() (gas: 2169447)
[PASS] test_recursiveDepthCapped_execUserOp() (gas: 1890195)
[PASS] test_recursiveDepthCapped_runtime() (gas: 1803665)
[PASS] test_recursiveDepthCapped_userOp() (gas: 1885359)
[PASS] test_selfCallFails_execUserOp() (gas: 1774795)
[PASS] test_selfCallFails_runtime() (gas: 1676565)
[PASS] test_selfCallFails_userOp() (gas: 1772613)
[PASS] test_selfCallPrivilegeEscalation_prevented_execUserOp() (gas: 1908050)
[PASS] test_selfCallPrivilegeEscalation_prevented_runtime() (gas: 1723333)
[PASS] test_selfCallPrivilegeEscalation_prevented_userOp() (gas: 1901392)
Suite result: ok. 12 passed; 0 failed; 0 skipped; finished in 35.51ms (32.19ms CPU time)

Ran 15 tests for test/account/HookOrdering.t.sol:HookOrderingTest
[PASS] test_hookOrder_directCall_accountNativeFunction_noAssoc() (gas: 3506641)
[PASS] test_hookOrder_directCall_accountNativeFunction_withAssoc() (gas: 4290179)
[PASS] test_hookOrder_directCall_moduleExecFunction() (gas: 4290158)
[PASS] test_hookOrder_directCall_moduleExecFunction_noAssoc() (gas: 3506621)
[PASS] test_hookOrder_runtime_accountNativeFunction_noAssoc() (gas: 3586116)
[PASS] test_hookOrder_runtime_accountNativeFunction_regular() (gas: 4370949)
[PASS] test_hookOrder_runtime_moduleExecFunction() (gas: 4364778)
[PASS] test_hookOrder_runtime_moduleExecFunction_noAssoc() (gas: 3579885)
[PASS] test_hookOrder_signatureValidation() (gas: 3263983)
[PASS] test_hookOrder_userOp_accountNativeFunction_noAssoc_execUO() (gas: 3867426)
[PASS] test_hookOrder_userOp_accountNativeFunction_noAssoc_regular() (gas: 3850811)
[PASS] test_hookOrder_userOp_accountNativeFunction_withAssoc() (gas: 4654574)
[PASS] test_hookOrder_userOp_moduleExecFunction_noAssoc_execUO() (gas: 3860132)
[PASS] test_hookOrder_userOp_moduleExecFunction_noAssoc_regular() (gas: 3845308)
[PASS] test_hookOrder_userOp_moduleExecFunction_withAssoc() (gas: 4647142)
Suite result: ok. 15 passed; 0 failed; 0 skipped; finished in 46.35ms (43.95ms CPU time)

Ran 2 tests for test/libraries/KnownSelectorsLib.t.sol:KnownSelectorsLibTest
[PASS] test_isERC4337Function() (gas: 7288)
[PASS] test_isIModuleFunction() (gas: 12595)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 184.67µs (75.13µs CPU time)

Ran 17 tests for test/libraries/LinkedListSetLib.t.sol:LinkedListSetLibTest
[PASS] test_add_contains() (gas: 48948)
[PASS] test_add_remove_add() (gas: 54620)
[PASS] test_add_remove_add_empty() (gas: 54638)
[PASS] test_clear() (gas: 37105)
[PASS] test_empty() (gas: 8388)
[PASS] test_getAll() (gas: 74775)
```

```
[PASS] test_getAll2() (gas: 99349)
[PASS] test_getAll_empty() (gas: 5784)
[PASS] test_isSentinel() (gas: 5129)
[PASS] test_remove() (gas: 36789)
[PASS] test_remove_empty() (gas: 5709)
[PASS] test_remove_nonexistent() (gas: 52679)
[PASS] test_remove_nonexistent_empty() (gas: 5664)
[PASS] test_remove_nonexistent_empty2() (gas: 52700)
[PASS] test_tryRemoveKnown1() (gas: 37492)
[PASS] test_tryRemoveKnown2() (gas: 63621)
[PASS] test_tryRemoveKnown_invalid1() (gas: 75119)
Suite result: ok. 17 passed; 0 failed; 0 skipped; finished in 742.21µs (610.58µs CPU time)

Ran 4 tests for test/account/SemiModularAccountDirectCall.t.sol:SemiModularAccountDirectCallTest
[PASS] test_Flow_smaDirectCall_installedHooksUninstalled() (gas: 241794)
[PASS] test_fail_smaDirectCall_disabledFallbackSigner() (gas: 67928)
[PASS] test_fail_smaDirectCall_notFallbackSigner() (gas: 34714)
[PASS] test_smaDirectCall() (gas: 36611)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 10.08ms (7.02ms CPU time)

Ran 3 tests for test/account/ValidationAssocHooks.t.sol:ValidationAssocHooksTest
[PASS] test_revertOnMissingExecuteUserOp() (gas: 263309289)
[PASS] test_validationAssocHooks_maxExecHooks() (gas: 275269137)
[PASS] test_validationAssocHooks_maxValidationHooks() (gas: 275221430)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 125.65ms (100.56ms CPU time)

Ran 10 tests for test/account/ValidationIntersection.t.sol:ValidationIntersectionTest
[PASS] testFuzz_validationIntersect_single(uint256) (runs: 500, µ: 103813, ~: 104052)
[PASS] test_validationIntersect_authorizerAndTimeRange() (gas: 3609018)
[PASS] test_validationIntersect_authorizer_sigfail_hook() (gas: 3564174)
[PASS] test_validationIntersect_authorizer_sigfail_validationFunction() (gas: 3564302)
[PASS] test_validationIntersect_multiplePreValidationHooksIntersect() (gas: 3618284)
[PASS] test_validationIntersect_multiplePreValidationHooksSigFail() (gas: 3578095)
[PASS] test_validationIntersect_revert_unexpectedAuthorizer() (gas: 3503399)
[PASS] test_validationIntersect_timeBounds_intersect_1() (gas: 3604429)
[PASS] test_validationIntersect_timeBounds_intersect_2() (gas: 3604429)
[PASS] test_validationIntersect_validAuthorizer() (gas: 3569154)
Suite result: ok. 10 passed; 0 failed; 0 skipped; finished in 152.86ms (145.69ms CPU time)

Ran 8 tests for test/libraries/ValidationLocatorLib.t.sol:ValidationLocatorLibTest
[PASS] testFuzz_configToLookupKey(bytes24,bool,bool,bool) (runs: 500, µ: 5026, ~: 5026)
[PASS] testFuzz_loadFromNonce_directCall(address,bool,bool) (runs: 500, µ: 4491, ~: 4491)
[PASS] testFuzz_loadFromNonce_regular(uint32,bool,bool) (runs: 500, µ: 4437, ~: 4436)
[PASS] testFuzz_loadFromSignature_directCall(address,bool,bool,bytes) (runs: 500, µ: 11588, ~: 11336)
[PASS] testFuzz_loadFromSignature_regular(uint32,bool,bool,bytes) (runs: 500, µ: 11524, ~: 11300)
[PASS] testFuzz_moduleEntityToLookupKey(bytes24) (runs: 500, µ: 4170, ~: 4170)
[PASS] testFuzz_validationLookupKey_directCall(address,bool,bool) (runs: 500, µ: 4493, ~: 4493)
[PASS] testFuzz_validationLookupKey_regular(uint32,bool,bool) (runs: 500, µ: 4359, ~: 4359)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 127.52ms (127.37ms CPU time)

Ran 2 tests for test/helpers/ValidationResHelpers.t.sol:ValidationResHelpersTest
[PASS] test_coalescePreValidation() (gas: 16202)
[PASS] test_coalesceValidation() (gas: 19580)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 258.75µs (142.33µs CPU time)

Ran 8 tests for test/account/DirectCallsFromModule.t.sol:DirectCallsFromModuleTest
[PASS] testFuzz_directCallFromModuleCallback(bool) (runs: 500, µ: 187316, ~: 209692)
[PASS] testFuzz_directCallFromModulePrank(bool) (runs: 500, µ: 175647, ~: 198024)
[PASS] testFuzz_directCallFromModuleSequence(bool) (runs: 500, µ: 162552, ~: 179091)
[PASS] testFuzz_fail_directCallModuleUninstalled(bool) (runs: 500, µ: 126482, ~: 142816)
[PASS] test_directCallFromModuleSequence_runHooks() (gas: 151562)
[PASS] test_directCallsFromEOA() (gas: 1158778)
[PASS] test_fail_directCallModuleCallOtherSelector() (gas: 1252337)
```

```
[PASS] test_fail_directCallModuleNotInstalled() (gas: 988505)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 707.66ms (705.38ms CPU time)

Ran 25 tests for test/account/ModularAccount.t.sol:ModularAccountTest
[PASS] test_accountId() (gas: 972192)
[PASS] test_assertCallerEntryPoint() (gas: 999671)
[PASS] test_basicUserOp_withInitCode() (gas: 1588855)
[PASS] test_batchExecute() (gas: 1361292)
[PASS] test_contractInteraction() (gas: 1328116)
[PASS] test_debug_ModularAccount_storageAccesses() (gas: 1337662)
[PASS] test_deployAccount() (gas: 1284334)
[PASS] test_installExecution() (gas: 1126170)
[PASS] test_installExecution_PermittedCallSelectorNotInstalled() (gas: 3003322)
[PASS] test_installExecution_alreadyInstalled() (gas: 1101881)
[PASS] test_installExecution_interfaceNotSupported() (gas: 1000958)
[PASS] test_isValidSignature() (gas: 1025062)
[PASS] test_modularAccountBase_supportsInterface() (gas: 1074092)
[PASS] test_performCreate_create() (gas: 13723225)
[PASS] test_performCreate_create2() (gas: 1071785885)
[PASS] test_postDeploy_ethSend() (gas: 1332521)
[PASS] test_revertOnNoInstalledModuleExecFunction() (gas: 979491)
[PASS] test_rtValidationWithValue() (gas: 3232998)
[PASS] test_signatureValidationFlag_enforce() (gas: 1139182)
[PASS] test_standardExecuteEthSend_withInitcode() (gas: 1635154)
[PASS] test_transferOwnership() (gas: 1045428)
[PASS] test_uninstallExecution_default() (gas: 3748877)
[PASS] test_upgradeToAndCall() (gas: 13261852)
[PASS] test_userOpValidationFlag_enforce() (gas: 1303020)
[PASS] test_validationRevertsOnShortCalldata() (gas: 1082177)
Suite result: ok. 25 passed; 0 failed; 0 skipped; finished in 42.39ms (37.52ms CPU time)

Ran 7 tests for test/account/ModularAccountView.t.sol:ModularAccountViewTest
[PASS] test_moduleView_getExecutionData_module() (gas: 1867628)
[PASS] test_moduleView_getExecutionData_nativeGlobalValidationAllowed() (gas: 2029291)
[PASS] test_moduleView_getExecutionData_nativeGlobalValidationAllowedSMA() (gas: 1844825)
[PASS] test_moduleView_getExecutionData_nativeUnwrapped() (gas: 4494703)
[PASS] test_moduleView_getExecutionData_nativeUnwrappedMA() (gas: 4285198)
[PASS] test_moduleView_getExecutionData_nativeUnwrappedSMA() (gas: 4285376)
[PASS] test_moduleView_getValidationData() (gas: 1873096)
Suite result: ok. 7 passed; 0 failed; 0 skipped; finished in 13.74ms (9.41ms CPU time)

Ran 3 tests for test/account/MultiValidation.t.sol:MultiValidationTest
[PASS] test_overlappingValidationInstall() (gas: 1636132)
[PASS] test_runtimeValidation_specify() (gas: 4781487)
[PASS] test_userOpValidation_specify() (gas: 5076320)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 8.11ms (5.63ms CPU time)

Ran 17 tests for test/modules/NativeTokenLimitModule.t.sol:NativeTokenLimitModuleTest
[PASS] test_deleteSingleSessionKey() (gas: 1265917)
[PASS] test_runtime_executeBatchLimit() (gas: 574640)
[PASS] test_runtime_executeLimit() (gas: 503180)
[PASS] test_runtime_performCreate2Limit() (gas: 507703)
[PASS] test_runtime_performCreateLimit() (gas: 507748)
[PASS] test_userOp_combinedExecBatchLimit_success() (gas: 823204)
[PASS] test_userOp_combinedExecLimit_failExec() (gas: 678096)
[PASS] test_userOp_combinedExecLimit_success() (gas: 746716)
[PASS] test_userOp_executeBatchLimit() (gas: 546356)
[PASS] test_userOp_executeLimit() (gas: 524228)
[PASS] test_userOp_failsWithValidationData() (gas: 879264)
[PASS] test_userOp_gasLimit() (gas: 491028)
[PASS] test_userOp_invalidPaymaster() (gas: 476731)
[PASS] test_userOp_paymaster() (gas: 446801)
[PASS] test_userOp_performCreate2Limit() (gas: 483527)
```

```
[PASS] test_userOp_performCreateLimit() (gas: 483575)
[PASS] test_userOp_specialPaymaster() (gas: 506145)
Suite result: ok. 17 passed; 0 failed; 0 skipped; finished in 21.35ms (18.42ms CPU time)

Ran 12 tests for test/account/PHCallBuffers.t.sol:PHCallBufferTest
[PASS] test_preExecHooksRun_execUO() (gas: 9120268)
[PASS] test_preExecHooksWithRtValidation_freshBuffer_regularCallData() (gas: 4441286)
[PASS] test_preExecHooksWithRtValidation_freshBuffer_unalignedCallData() (gas: 4442134)
[PASS] test_preExecHooksWithRtValidation_reuseConvertedRTBuffer_regularCallData() (gas: 10936611)
[PASS] test_preExecHooksWithRtValidation_reuseConvertedRTBuffer_unalignedCallData() (gas: 10938401)
[PASS] test_preExecHooksWithRtValidation_reusePRTOnlyBuffer_regularCalldata() (gas: 5533540)
[PASS] test_preExecHooksWithRtValidation_reusePRTOnlyBuffer_unalignedCalldata() (gas: 11062106)
[PASS] test_preExecHooksWithRtValidation_reuseRTOnlyBuffer_regularCallData() (gas: 8753436)
[PASS] test_preExecHooksWithRtValidation_reuseRTOnlyBuffer_unalignedCallData() (gas: 8755198)
[PASS] test_preExecHooks_EPCall_regularCallData() (gas: 6689267)
[PASS] test_preExecHooks_EPCall_unalignedCallData() (gas: 6691168)
[PASS] test_preExecHooks_directCallValidation_withPRTHooks() (gas: 10923310)
Suite result: ok. 12 passed; 0 failed; 0 skipped; finished in 33.15ms (30.48ms CPU time)

Ran 2 tests for test/invariant/LinkedListSetLibInvariants.t.sol:LinkedListSetLibInvariantsTest
[PASS] invariant_getAllEquivalence() (runs: 500, calls: 5000, reverts: 0)
[PASS] invariant_shouldContain() (runs: 500, calls: 5000, reverts: 0)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 822.23ms (821.90ms CPU time)

Ran 4 tests for test/account/TokenReceiver.t.sol:TokenReceiverTest
[PASS] test_receiveERC1155() (gas: 3595256)
[PASS] test_receiveERC1155Batch() (gas: 3600360)
[PASS] test_receiveERC721() (gas: 3582253)
[PASS] test_supportedInterfaces() (gas: 3537585)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 15.94ms (2.73ms CPU time)

Ran 3 tests for test/account/UpgradeToSma.t.sol:UpgradeToSmaTest
[PASS] test_fail_upgradeToAndCall_initializedMaToSmaStorage() (gas: 35499)
[PASS] test_upgradeToAndCall_LaToSmaStorage() (gas: 3183290)
[PASS] test_upgradeToAndCall_MaToSmaStorage() (gas: 386423)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 12.23ms (9.39ms CPU time)

Ran 9 tests for test/modules/AllowlistModule.t.sol:AllowlistModuleTest
[PASS] testFuzz_allowlistHook_runtime_batch(uint256) (runs: 500, μ: 1186066, ~: 1159722)
[PASS] testFuzz_allowlistHook_runtime_single(uint256) (runs: 500, μ: 1157599, ~: 1122159)
[PASS] testFuzz_allowlistHook_userOp_batch(uint256) (runs: 500, μ: 1253290, ~: 1215776)
[PASS] testFuzz_allowlistHook_userOp_single(uint256) (runs: 500, μ: 1216506, ~: 1199690)
[PASS] test_checkAllowlistCalldata_execute() (gas: 202114)
[PASS] test_nativeTokenTransfer_success() (gas: 155259)
[PASS] test_onInstall() (gas: 165539)
[PASS] test_onUninstall() (gas: 158585)
[PASS] test_revertsOnUnnecessaryValidationData() (gas: 631674)
Suite result: ok. 9 passed; 0 failed; 0 skipped; finished in 1.62s (3.63s CPU time)

Ran 13 tests for test/modules/TimeRangeModule.t.sol:TimeRangeModuleTest
[PASS] testFuzz_timeRangeModule_userOp_fail(uint48,uint48) (runs: 500, μ: 353278, ~: 354591)
[PASS] testFuzz_timeRangeModule_userOp_success(uint48,uint48) (runs: 500, μ: 361318, ~: 362631)
[PASS] testFuzz_timeRangeModule_userOp_validUntil_0() (gas: 358617)
[PASS] test_timeRangeModule_install() (gas: 1598694)
[PASS] test_timeRangeModule_moduleId() (gas: 10650)
[PASS] test_timeRangeModule_runtime_after() (gas: 1603489)
[PASS] test_timeRangeModule_runtime_before() (gas: 1603515)
[PASS] test_timeRangeModule_runtime_during() (gas: 1628979)
[PASS] test_timeRangeModule_runtime_success_validUntil_0() (gas: 1626327)
[PASS] test_timeRangeModule_setBadTime() (gas: 1573794)
[PASS] test_timeRangeModule_setGoodTime() (gas: 1594553)
[PASS] test_timeRangeModule_uninstall() (gas: 4020224)
[PASS] test_timeRangeModule_userOp_fails_extraValidationData() (gas: 1627897)
```

```
Suite result: ok. 13 passed; 0 failed; 0 skipped; finished in 997.26ms (998.31ms CPU time)

Ran 5 tests for test/modules/WebAuthnValidationModule.t.sol:WebAuthnValidationModuleTest
[PASS] testFuzz_fail_isValidSignature(bytes32,uint256,uint256) (runs: 500, μ: 290552, ~: 302361)
[PASS] testFuzz_pass_isValidSignature(bytes32) (runs: 500, μ: 287046, ~: 287328)
[PASS] testFuzz_uoValidation_shouldFail(uint256,uint256) (runs: 500, μ: 301532, ~: 317481)
[PASS] test_isValidSignature() (gas: 289834)
[PASS] test_uoValidation() (gas: 3481021)
Suite result: ok. 5 passed; 0 failed; 0 skipped; finished in 2.74s (4.15s CPU time)

Ran 2 tests for test/libraries/SparseCalldataSegmentLib.t.sol:SparseCalldataSegmentLibTest
[PASS] testFuzz_sparseCalldataSegmentLib_encodeDecode_simple(bytes[]) (runs: 500, μ: 3329098, ~: 2093742)
[PASS] testFuzz_sparseCalldataSegmentLib_encodeDecode_withIndex(bytes[],uint256) (runs: 500, μ: 3577422,
~: 2498059)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 6.21s (10.09s CPU time)

Ran 4 tests for test/account/PostHookData.t.sol:PostHookDataTest
[PASS] testFuzz_randomizedValAssocExecHooks_passDataCorrectly_runtime((uint8,bytes[256],uint8[256]))
(runs: 500, μ: 75491335, ~: 28784486)
[PASS] testFuzz_randomizedValAssocExecHooks_passDataCorrectly_userOp((uint8,bytes[256],uint8[256]))
(runs: 500, μ: 75812426, ~: 28965422)
[PASS] test_randomizedValAssocExecHooks_passDataCorrectly_runtime_example() (gas: 4353431)
[PASS] test_valAssocExecHooks_passDataCorrectly_userOp_example() (gas: 4480903)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 7.54s (10.29s CPU time)

Ran 1 test for test/account/SMASpecific.t.sol:SMASpecificTest
[PASS] testFuzz_fallbackValidation_hooksFlow(uint32,uint32,bool[254]) (runs: 500, μ: 6806948, ~: 7100381)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 7.66s (7.65s CPU time)

Ran 4 tests for test/account/SigCallBuffer.t.sol:SigCallBufferTest
[PASS] testFuzz_sigCallBuffer(bytes32,(uint8,bytes[256],bytes)) (runs: 500, μ: 147757839, ~: 53288852)
[PASS] test_sigCallBuffer_noData() (gas: 10793639)
[PASS] test_sigCallBuffer_shortReturnData() (gas: 5352405)
[PASS] test_sigCallBuffer_withData() (gas: 10859612)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 13.48s (13.48s CPU time)

Ran 3 tests for test/account/RTCallBuffer.t.sol:RTCallBufferTest
[PASS] testFuzz_multipleRTCalls(bytes[5],bytes) (runs: 500, μ: 13104250, ~: 13103811)
[PASS] testFuzz_variableLengthRTCalls(uint8,bytes[256],bytes) (runs: 500, μ: 144791753, ~: 55904158)
[PASS] test_multipleRTCalls() (gas: 13036762)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 14.95s (13.55s CPU time)

Ran 5 tests for test/account/UOCallBuffer.t.sol:UOCallBufferTest
[PASS] testFuzz_multipleUOCalls(bytes[5],bytes) (runs: 500, μ: 13246946, ~: 13246602)
[PASS] testFuzz_variableLengthUOCalls(uint8,bytes[256],bytes) (runs: 500, μ: 156453610, ~: 58570064)
[PASS] test_multipleUOCalls() (gas: 13180634)
[PASS] test_uoCallBuffer_shortReturnData_preValidationHook() (gas: 12851867)
[PASS] test_uoCallBuffer_shortReturnData_validationFunction() (gas: 51794)
Suite result: ok. 5 passed; 0 failed; 0 skipped; finished in 14.91s (15.17s CPU time)

Ran 41 test suites in 16.02s (100.35s CPU time): 284 tests passed, 0 failed, 0 skipped (284 total tests)
```

Reference Implementation

```
[⠒] Compiling...
[⠒] Compiling 166 files with Solc 0.8.26
[⠢] Solc 0.8.26 finished in 8.73s
Compiler run successful!
Analysing contracts...
Running tests...
```

```
Ran 2 tests for test/account/AccountStorage.t.sol:AccountStorageTest
[PASS] test_storageSlotImpl() (gas: 7430)
[PASS] test_storageSlotProxy() (gas: 35046)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 8.64ms (2.07ms CPU time)

Ran 3 tests for test/account/AccountFactory.t.sol:AccountFactoryTest
[PASS] test_createAccount() (gas: 201772)
[PASS] test_createAccountAndGetAddress() (gas: 201835)
[PASS] test_multipleDeploy() (gas: 202085)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 13.54ms (2.64ms CPU time)

Ran 3 tests for test/account/ModularAccountView.t.sol:ModularAccountViewTest
[PASS] test_moduleView_getExecutionData_module() (gas: 33898)
[PASS] test_moduleView_getExecutionData_native() (gas: 35178)
[PASS] test_moduleView_getValidationData() (gas: 39777)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 14.34ms (3.79ms CPU time)

Ran 4 tests for test/comparison/CompareSimpleAccount.t.sol:CompareSimpleAccountTest
[PASS] test_SimpleAccount_deploy_basicSend() (gas: 313581)
[PASS] test_SimpleAccount_deploy_empty() (gas: 302385)
[PASS] test_SimpleAccount_postDeploy_basicSend() (gas: 128928)
[PASS] test_SimpleAccount_postDeploy_contractInteraction() (gas: 133635)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 20.74ms (12.39ms CPU time)

Ran 6 tests for test/modules/permissions/ERC20TokenLimitModule.t.sol:ERC20TokenLimitModuleTest
[PASS] test_runtime_executeBatchLimit() (gas: 180387)
[PASS] test_runtime_executeLimit() (gas: 117350)
[PASS] test_userOp_executeBatchLimit() (gas: 144032)
[PASS] test_userOp_executeBatch_approveAndTransferLimit() (gas: 164966)
[PASS] test_userOp_executeBatch_approveAndTransferLimit_fail() (gas: 204424)
[PASS] test_userOp_executeLimit() (gas: 111378)
Suite result: ok. 6 passed; 0 failed; 0 skipped; finished in 28.06ms (16.90ms CPU time)

Ran 5 tests for test/account/AccountReturnData.t.sol:AccountReturnDataTest
[PASS] test_returnData_authorized_exec() (gas: 56848)
[PASS] test_returnData_execFromModule_fallback() (gas: 30844)
[PASS] test_returnData_executeBatch() (gas: 66154)
[PASS] test_returnData_fallback() (gas: 19926)
[PASS] test_returnData_singular_execute() (gas: 52574)
Suite result: ok. 5 passed; 0 failed; 0 skipped; finished in 19.49ms (7.95ms CPU time)

Ran 2 tests for test/account/GlobalValidationTest.t.sol:GlobalValidationTest
[PASS] test_globalValidation_runtime_simple() (gas: 249530)
[PASS] test_globalValidation_userOp_simple() (gas: 407722)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 8.06ms (4.81ms CPU time)

Ran 3 tests for test/account/MultiValidation.t.sol:MultiValidationTest
[PASS] test_overlappingValidationInstall() (gas: 118272)
[PASS] test_runtimeValidation_specify() (gas: 168453)
[PASS] test_userOpValidation_specify() (gas: 338251)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 11.62ms (8.55ms CPU time)

Ran 1 test for test/script/DeployAllowlistModule.s.t.sol:DeployAllowlistModuleTest
[PASS] test_deployAllowlistModuleScript_run() (gas: 3261634)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 53.05ms (45.60ms CPU time)

Ran 18 tests for test/account/ReferenceModularAccount.t.sol:ReferenceModularAccountTest
[PASS] test_accountId() (gas: 16620)
[PASS] test_basicUserOp_withInitCode() (gas: 404763)
[PASS] test_batchExecute() (gas: 232019)
[PASS] test_contractInteraction() (gas: 209081)
[PASS] test_debug_ReferenceModularAccount_storageAccesses() (gas: 212910)
```

```
[PASS] test_deployAccount() (gas: 205620)
[PASS] test_installExecution() (gas: 212798)
[PASS] test_installExecution_PermittedCallSelectorNotInstalled() (gas: 1002418)
[PASS] test_installExecution_alreadyInstalled() (gas: 195046)
[PASS] test_installExecution_interfaceNotSupported() (gas: 26972)
[PASS] test_isValidSignature() (gas: 48738)
[PASS] test_postDeploy_ethSend() (gas: 208796)
[PASS] test_signatureValidationFlag_enforce() (gas: 104139)
[PASS] test_standardExecuteEthSend_withInitcode() (gas: 433039)
[PASS] test_transferOwnership() (gas: 46155)
[PASS] test_uninstallExecution_default() (gas: 1385973)
[PASS] test_upgradeToAndCall() (gas: 7420967)
[PASS] test_userOpValidationFlag_enforce() (gas: 188782)
Suite result: ok. 18 passed; 0 failed; 0 skipped; finished in 53.50ms (43.96ms CPU time)

Ran 2 tests for test/script/Deploy.s.t.sol:DeployTest
[PASS] test_deployScript_addStake() (gas: 42052264)
[PASS] test_deployScript_run() (gas: 20128203)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 64.55ms (81.30ms CPU time)

Ran 2 tests for test/account/ReplaceModule.t.sol:UpgradeModuleTest
[PASS] test_upgradeModuleExecutionFunction() (gas: 2682238)
[PASS] test_upgradeModuleValidationFunction() (gas: 3852724)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 13.93ms (11.02ms CPU time)

Ran 8 tests for test/modules/permissions/NativeTokenLimitModule.t.sol:NativeTokenLimitModuleTest
[PASS] test_runtime_executeBatchLimit() (gas: 163472)
[PASS] test_runtime_executeLimit() (gas: 115268)
[PASS] test_userOp_combinedExecBatchLimit_success() (gas: 292537)
[PASS] test_userOp_combinedExecLimit_failExec() (gas: 200687)
[PASS] test_userOp_combinedExecLimit_success() (gas: 238700)
[PASS] test_userOp_executeBatchLimit() (gas: 143030)
[PASS] test_userOp_executeLimit() (gas: 130973)
[PASS] test_userOp_gasLimit() (gas: 115205)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 40.92ms (35.36ms CPU time)

Ran 2 tests for test/mocks/Counter.t.sol:CounterTest
[PASS] testIncrement() (gas: 28677)
[PASS] testSetNumber(uint256) (runs: 500, μ: 28931, ~: 29170)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 115.05ms (108.57ms CPU time)

Ran 12 tests for test/account/SelfCallAuthorization.t.sol:SelfCallAuthorizationTest
[PASS] test_batchAction_allowed_execUserOp() (gas: 356169)
[PASS] test_batchAction_allowed_runtime() (gas: 207425)
[PASS] test_batchAction_allowed_userOp() (gas: 343199)
[PASS] test_recursiveDepthCapped_execUserOp() (gas: 164423)
[PASS] test_recursiveDepthCapped_runtime() (gas: 127747)
[PASS] test_recursiveDepthCapped_userOp() (gas: 161986)
[PASS] test_selfCallFails_execUserOp() (gas: 73322)
[PASS] test_selfCallFails_runtime() (gas: 30080)
[PASS] test_selfCallFails_userOp() (gas: 72249)
[PASS] test_selfCallPrivilegeEscalation_prevented_execUserOp() (gas: 142560)
[PASS] test_selfCallPrivilegeEscalation_prevented_runtime() (gas: 60113)
[PASS] test_selfCallPrivilegeEscalation_prevented_userOp() (gas: 139239)
Suite result: ok. 12 passed; 0 failed; 0 skipped; finished in 68.21ms (63.76ms CPU time)

Ran 19 tests for test/account/PerHookData.t.sol:PerHookDataTest
[PASS] test_fail1271AccessControl_badSigData() (gas: 39681)
[PASS] test_fail1271AccessControl_noSigData() (gas: 35967)
[PASS] test_failAccessControl_badIndexProvided_runtime() (gas: 57089)
[PASS] test_failAccessControl_badIndexProvided_userOp() (gas: 104153)
[PASS] test_failAccessControl_badSigData_runtime() (gas: 56766)
[PASS] test_failAccessControl_badSigData_userOp() (gas: 98451)
```

```
[PASS] test_failAccessControl_badTarget_runtime() (gas: 58823)
[PASS] test_failAccessControl_badTarget_userOp() (gas: 98076)
[PASS] test_failAccessControl_indexOutOfOrder_runtime() (gas: 134740)
[PASS] test_failAccessControl_indexOutOfOrder_userOp() (gas: 179726)
[PASS] test_failAccessControl_noSigData_runtime() (gas: 53096)
[PASS] test_failAccessControl_noSigData_userOp() (gas: 94771)
[PASS] test_failPerHookData_nonCanonicalEncoding_runtime() (gas: 42738)
[PASS] test_failPerHookData_nonCanonicalEncoding_userOp() (gas: 87829)
[PASS] test_pass1271AccessControl() (gas: 60428)
[PASS] test_passAccessControl_runtime() (gas: 97022)
[PASS] test_passAccessControl_twoHooks_runtime() (gas: 182784)
[PASS] test_passAccessControl_twoHooks_userOp() (gas: 332416)
[PASS] test_passAccessControl_userOp() (gas: 246825)
Suite result: ok. 19 passed; 0 failed; 0 skipped; finished in 62.92ms (50.78ms CPU time)

Ran 2 tests for test/account/PermittedCallPermissions.t.sol:PermittedCallPermissionsTest
[PASS] test_permittedCall_Allowed() (gas: 31819)
[PASS] test_permittedCall_NotAllowed() (gas: 33427)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 27.00ms (16.97ms CPU time)

Ran 2 tests for test/libraries/ModuleEntityLib.t.sol:ModuleEntityLibTest
[PASS] testFuzz_moduleEntity_operators(bytes24,bytes24) (runs: 500, μ: 1581, ~: 1578)
[PASS] testFuzz_moduleEntity_packing(address,uint32) (runs: 500, μ: 987, ~: 987)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 210.26ms (210.18ms CPU time)

Ran 2 tests for test/libraries/HookConfigLib.t.sol:HookConfigLibTest
[PASS] testFuzz_hookConfig_packingModuleEntity(bytes24,bool,bool,bool) (runs: 500, μ: 1755, ~: 2001)
[PASS] testFuzz_hookConfig_packingUnderlying(address,uint32,bool,bool,bool) (runs: 500, μ: 2337, ~: 2574)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 238.98ms (238.89ms CPU time)

Ran 3 tests for test/libraries/KnowSelectorsLib.t.sol:KnownSelectorsLibTest
[PASS] test_isErc4337Function() (gas: 486)
[PASS] test_isIModuleFunction() (gas: 598)
[PASS] test_isNativeFunction() (gas: 514)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 429.21µs (332.00µs CPU time)

Ran 2 tests for test/libraries/ValidationConfigLib.t.sol:ValidationConfigLibTest
[PASS] testFuzz_validationConfig_packingModuleEntity(bytes24,bool,bool,bool) (runs: 500, μ: 4433, ~:
4433)
[PASS] testFuzz_validationConfig_packingUnderlying(address,uint32,bool,bool,bool) (runs: 500, μ: 4701, ~:
4702)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 489.42ms (489.31ms CPU time)

Ran 2 tests for test/libraries/ModuleStorageLib.t.sol:ModuleStorageLibTest
[PASS] testFuzz_storagePointer(address,uint256,bytes32,uint256[32]) (runs: 500, μ: 785250, ~: 793847)
[PASS] test_storagePointer() (gas: 55733)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 1.58s (1.58s CPU time)

Ran 6 tests for test/modules/TokenReceiverModule.t.sol:TokenReceiverModuleTest
[PASS] test_failERC1155Transfer() (gas: 201363)
[PASS] test_failERC721Transfer() (gas: 57278)
[PASS] test_failIntrospection() (gas: 17917)
[PASS] test_passERC1155Transfer() (gas: 408950)
[PASS] test_passERC721Transfer() (gas: 229647)
[PASS] test_passIntrospection() (gas: 182876)
Suite result: ok. 6 passed; 0 failed; 0 skipped; finished in 30.73ms (23.80ms CPU time)

Ran 10 tests for test/account/ValidationIntersection.t.sol:ValidationIntersectionTest
[PASS] testFuzz_validationIntersect_single(uint256) (runs: 500, μ: 114886, ~: 115125)
[PASS] test_validationIntersect_authorizerAndTimeRange() (gas: 148001)
[PASS] test_validationIntersect_authorizer_sigfail_hook() (gas: 125486)
[PASS] test_validationIntersect_authorizer_sigfail_validationFunction() (gas: 125560)
[PASS] test_validationIntersect_multiplePreValidationHooksIntersect() (gas: 155874)
```

```
[PASS] test_validationIntersect_multiplePreValidationHooksSigFail() (gas: 135769)
[PASS] test_validationIntersect_revert_unexpectedAuthorizer() (gas: 90487)
[PASS] test_validationIntersect_timeBounds_intersect_1() (gas: 145635)
[PASS] test_validationIntersect_timeBounds_intersect_2() (gas: 145657)
[PASS] test_validationIntersect_validAuthorizer() (gas: 127917)
Suite result: ok. 10 passed; 0 failed; 0 skipped; finished in 1.16s (1.15s CPU time)

Ran 5 tests for
test/modules/validation/SingleSignerValidationModule.t.sol:SingleSignerValidationModuleTest
[PASS] testFuzz_isValidSignatureForContractOwner(bytes32) (runs: 500, μ: 38923, ~: 38923)
[PASS] testFuzz_isValidSignatureForEOAOwner(string,bytes32) (runs: 500, μ: 52636, ~: 52718)
[PASS] test_runtimeValidate() (gas: 87232)
[PASS] test_runtime_with2SameValidationInstalled() (gas: 155325)
[PASS] test_userOpValidation() (gas: 233811)
Suite result: ok. 5 passed; 0 failed; 0 skipped; finished in 1.94s (1.94s CPU time)

Ran 7 tests for test/account/DirectCallsFromModule.t.sol:DirectCallsFromModuleTest
[PASS] testFuzz_Fail_DirectCallModuleUninstalled(bool) (runs: 500, μ: 146920, ~: 164695)
[PASS] testFuzz_Flow_DirectCallFromModuleSequence(bool) (runs: 500, μ: 186205, ~: 204197)
[PASS] testFuzz_Pass_DirectCallFromModuleCallback(bool) (runs: 500, μ: 237562, ~: 261036)
[PASS] testFuzz_Pass_DirectCallFromModulePrank(bool) (runs: 500, μ: 206629, ~: 230103)
[PASS] test_Fail_DirectCallModuleCallOtherSelector() (gas: 189261)
[PASS] test_Fail_DirectCallModuleNotInstalled() (gas: 29172)
[PASS] test_directCallsFromEOA() (gas: 120843)
Suite result: ok. 7 passed; 0 failed; 0 skipped; finished in 5.21s (6.41s CPU time)

Ran 4 tests for test/modules/permissions/AllowlistModule.t.sol:AllowlistModuleTest
[PASS] testFuzz_allowlistHook_runtime_batch(uint256) (runs: 500, μ: 1095887, ~: 1062693)
[PASS] testFuzz_allowlistHook_runtime_single(uint256) (runs: 500, μ: 1058524, ~: 1026577)
[PASS] testFuzz_allowlistHook_userOp_batch(uint256) (runs: 500, μ: 1157254, ~: 1119243)
[PASS] testFuzz_allowlistHook_userOp_single(uint256) (runs: 500, μ: 1111184, ~: 1075762)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 5.26s (15.91s CPU time)

Ran 9 tests for test/account/AccountExecHooks.t.sol:AccountExecHooksTest
[PASS] test_execHookPair_install() (gas: 1391273)
[PASS] test_execHookPair_run() (gas: 1425341)
[PASS] test_execHookPair_uninstall() (gas: 1383245)
[PASS] test_postOnlyExecHook_install() (gas: 1371441)
[PASS] test_postOnlyExecHook_run() (gas: 1388966)
[PASS] test_postOnlyExecHook_uninstall() (gas: 1363348)
[PASS] test_preExecHook_install() (gas: 1371418)
[PASS] test_preExecHook_run() (gas: 1392559)
[PASS] test_preExecHook_uninstall() (gas: 1363349)
Suite result: ok. 9 passed; 0 failed; 0 skipped; finished in 5.27s (29.46ms CPU time)

Ran 2 tests for test/libraries/SparseCalldataSegmentLib.t.sol:SparseCalldataSegmentLibTest
[PASS] testFuzz_sparseCalldataSegmentLib_encodeDecode_simple(bytes[]) (runs: 500, μ: 4310902, ~: 3034788)
[PASS] testFuzz_sparseCalldataSegmentLib_encodeDecode_withIndex(bytes[],uint256) (runs: 500, μ: 4458359,
~: 3304992)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 33.78s (61.55s CPU time)

Ran 29 test suites in 33.82s (55.78s CPU time): 148 tests passed, 0 failed, 0 skipped (148 total tests)
```

Code Coverage

In the `reference-implementation` repo, files like `SparseCalldataSegmentLib` and `ValidationConfigLib` demonstrate excellent coverage with 100% across all metrics, including branch and function coverage.

Though generally robust, the `modular-account` repo shows deficiencies in branch coverage for several files: `ModuleManagerInternals.sol`, `AccountFactory.sol`, `SemiModularAccountBase.sol`, and `AllowlistModule.sol`. These files would benefit from enhanced testing around conditional branches to ensure comprehensive validation under diverse scenarios.

This targeted improvement could significantly bolster the reliability and security of the code by covering more potential execution paths. In general, aiming for at least 90% coverage across all metrics—statement, branch, function, and line coverage—is recommended.

Fix-Review Update: The coverage metrics continue to show a strong overall testing regime, with many files like `src/account/AccountBase.sol` and `src/libraries/MemManagementLib.sol` achieving full 100% coverage across all categories. This indicates excellent test thoroughness in these areas. Although there is a slight decrease in coverage for a few files, such as `src/account/ModularAccountBase.sol` where line coverage modestly dropped from 99.62% to 99.09% and branch coverage from 82.95% to 77.59%, the overall high standards of coverage are maintained.

Modular Account

File	% Lines	% Statements	% Branches	% Funcs
src/account/AccountBase.sol	100.00% (12/12)	100.00% (7/7)	100.00% (2/2)	100.00% (4/4)
src/account/AccountStorageInitializable.sol	100.00% (21/21)	100.00% (26/26)	100.00% (5/5)	100.00% (2/2)
src/account/ModularAccount.sol	100.00% (6/6)	100.00% (6/6)	100.00% (0/0)	100.00% (3/3)
src/account/ModularAccountBase.sol	99.09% (326/329)	96.23% (357/371)	77.59% (45/58)	100.00% (36/36)
src/account/ModularAccountView.sol	100.00% (34/34)	100.00% (33/33)	100.00% (3/3)	100.00% (5/5)
src/account/ModuleManagerInternals.sol	94.29% (66/70)	95.24% (80/84)	63.64% (7/11)	100.00% (4/4)
src/account/SemiModularAccount7702.sol	0.00% (0/9)	0.00% (0/6)	0.00% (0/1)	0.00% (0/3)
src/account/SemiModularAccountBase.sol	90.48% (76/84)	91.84% (90/98)	64.71% (11/17)	100.00% (16/16)
src/account/SemiModularAccountBytecode.sol	100.00% (8/8)	100.00% (7/7)	100.00% (1/1)	100.00% (2/2)
src/account/SemiModularAccountStorageOnly.sol	55.56% (5/9)	50.00% (5/10)	100.00% (0/0)	33.33% (1/3)
src/account/TokenReceiver.sol	33.33% (2/6)	33.33% (1/3)	100.00% (0/0)	33.33% (1/3)
src/factory/AccountFactory.sol	79.03% (49/62)	87.10% (54/62)	50.00% (3/6)	62.50% (10/16)
src/helpers/ExecutionInstallDelegate.sol	89.39% (59/66)	89.47% (68/76)	25.00% (2/8)	100.00% (7/7)
src/libraries/ExecutionLib.sol	99.66% (297/298)	98.89% (268/271)	90.91% (30/33)	100.00% (24/24)
src/libraries/KnownSelectorsLib.sol	100.00% (18/18)	100.00% (34/34)	100.00% (0/0)	100.00% (2/2)
src/libraries/LinkedListSetLib.sol	96.55% (56/58)	97.50% (78/80)	83.33% (5/6)	100.00% (8/8)
src/libraries/MemManagementLib.sol	100.00% (66/66)	100.00% (70/70)	100.00% (0/0)	100.00% (12/12)

File	% Lines	% Statements	% Branches	% Funcs
src/libraries/ModuleInstallCommonsLib.sol	64.71% (11/17)	42.11% (8/19)	75.00% (3/4)	100.00% (3/3)
src/libraries/ValidationLocatorLib.sol	97.12% (101/104)	95.70% (89/93)	83.33% (20/24)	100.00% (20/20)
src/modules/ModuleBase.sol	100.00% (16/16)	94.12% (16/17)	100.00% (2/2)	100.00% (3/3)
src/modules/permissions/AllowlistModule.sol	83.81% (88/105)	92.04% (104/113)	86.96% (20/23)	50.00% (9/18)
src/modules/permissions/NativeTokenLimitModule.sol	84.75% (50/59)	90.48% (57/63)	100.00% (13/13)	66.67% (8/12)
src/modules/permissions/PaymasterGuardModule.sol	76.19% (16/21)	78.95% (15/19)	33.33% (1/3)	71.43% (5/7)
src/modules/permissions/TimeRangeModule.sol	96.30% (26/27)	96.30% (26/27)	100.00% (5/5)	87.50% (7/8)
src/modules/validation/SingleSignerValidationModule.sol	82.93% (34/41)	81.58% (31/38)	62.50% (5/8)	90.00% (9/10)
src/modules/validation/WebAuthnValidationModule.sol	72.73% (24/33)	77.78% (21/27)	100.00% (3/3)	60.00% (6/10)

Reference Implementation

File	Statement Coverage	Branch Coverage	Function Coverage	Line Coverage
src/libraries/HookConfigLib.sol	64.71% (11/17)	76.47% (26/34)	100.00% (0/0)	83.33% (10/12)
src/libraries/ModuleEntityLib.sol	87.50% (7/8)	81.82% (18/22)	100.00% (0/0)	83.33% (5/6)
src/libraries/SparseCalldataSegmentLib.sol	100.00% (17/17)	100.00% (23/23)	100.00% (8/8)	100.00% (4/4)
src/libraries/ValidationConfigLib.sol	100.00% (18/18)	100.00% (43/43)	100.00% (0/0)	100.00% (13/13)
src/modules/ModuleEIP712.sol	100.00% (1/1)	100.00% (2/2)	100.00% (0/0)	100.00% (1/1)
src/modules/ReplaySafeWrapper.sol	100.00% (4/4)	100.00% (5/5)	100.00% (0/0)	100.00% (2/2)
Total	92.06% (58/63)	93.62% (117/125)	100.00% (8/8)	94.44% (34/36)

Changelog

- 2024-11-05 - Initial report
- 2024-12-06 - 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

Timeliness of content

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

Notice of confidentiality

This report, including the content, data, and underlying methodologies, are subject to the confidentiality and feedback provisions in your agreement with Quantstamp. These materials are not to be disclosed, extracted, copied, or distributed except to the extent expressly authorized by Quantstamp.

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 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 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 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 and may not be represented as such. No third party is entitled to rely on the report in any any way, including for the purpose of making any decisions to buy or sell a product, 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 or any open source or third-party software, code, libraries, materials, or information to, to, called by, referenced by or accessible through the report, its content, or any related 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 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.