



Sequence V3 - Wallet Contracts

Executive Summary

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

Type	Wallet	Documentation quality	High	<div><div></div></div>
Timeline	2025-07-08 through 2025-08-13	Test quality	High	<div><div></div></div>
Language	Solidity	Total Findings	2	<div><div></div><div>Fixed: 1</div><div>Acknowledged: 1</div></div>
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review	High severity findings ⓘ	1	<div><div></div><div>Fixed: 1</div></div>
Specification	None	Medium severity findings ⓘ	0	
Source Code	<ul style="list-style-type: none">https://github.com/Oxsequence/wallet-contracts-v3 #d115794 	Low severity findings ⓘ	0	
Auditors	<ul style="list-style-type: none">Gereon Mendler Auditing EngineerIbrahim Abouzied Auditing EngineerRabib Islam Senior Auditing Engineer	Undetermined severity findings ⓘ	0	
		Informational findings ⓘ	1	<div><div></div><div>Acknowledged: 1</div></div>

Summary of Findings

We have audited the contracts for v3 of the Sequence Wallet, a modular smart contract wallet system. Architecturally, it is designed around a proxy pattern, with a core innovation in how it manages wallet configurations. Instead of storing settings like signers, thresholds, and extensions directly in storage, the system encodes them into a Merkle tree and stores only the root, referred to as an `imageHash`. This design allows for highly flexible and gas-efficient signature validation, and it enables counterfactual wallet deployment where the address is known before creation.

A key feature of this architecture is the "chained signatures" mechanism, which functions as a state channel for wallet configuration updates. This allows users to authorize a new wallet configuration off-chain by signing it with their current configuration, creating a sequence of valid states. These updates are only settled on-chain when a transaction is performed, making key rotation and permission changes effectively gasless until utilized. To ensure security and prevent replay attacks within this state channel, each configuration is assigned a strictly increasing "checkpoint," guaranteeing that older configurations cannot be reused.

The wallet's functionality is highly extensible through "Sapient Signers," which are smart contract modules that can implement custom and complex validation logic. The provided extensions demonstrate the power of this system, including a `Passkeys` (WebAuthn) signer for modern authentication, a social recovery module, and a comprehensive "Smart Sessions" system. Smart Sessions grant temporary, scoped permissions to dApps through either "Explicit Sessions" (defined in the wallet configuration) or "Implicit Sessions" (authorized off-chain by an identity signer), enabling seamless application interactions without requiring a full signature for every action.

Finally, the system is designed with multi-chain operations in mind, featuring an optional "Checkpointeer" contract. This component acts as a cross-chain oracle to broadcast the latest valid configuration, ensuring state consistency and preventing the use of stale configurations on less active chains.

We have identified one significant vulnerability in the codebase. It relates to the feature whereby explicit sessions may have cumulative usage limits placed on them. This feature can effectively be bypassed through the use of a particular flag that exits the execution loops and prevents increasing usage limits ([SEQ-1](#)).

The documentation is thorough and covers important implementation details for various wallet features. However, it would be good to also include more high-level documentation so as to further orient users and developers around the purpose and features of the wallet.

The test suite is excellent, with all of the repository's original code attaining 100% branch coverage. It doubles as an excellent guide to understanding the intended use of the wallet.

Fix-Review Update 2025-09-09:

All the issues and suggestions have been addressed. Most notably, [SEQ-1](#) has been fixed. The test suite has also been bolstered.

ID	DESCRIPTION	SEVERITY	STATUS
SEQ-1	Cumulative Session Limits Can Be Bypassed, Leading to Asset Drain	• High ⓘ	Fixed
SEQ-2	Rule Changes Immediately Reset Usage Limits for a Permission	• 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.

i

Disclaimer

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

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

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

Methodology

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

Scope

The scope included all files in `src/`, except for Solady contracts, as well as `Guest`, `Estimator`, and `Simulator`.

Files Included

Repo: `https://github.com/0xsequence/wallet-contracts-v3`

Extensions: `sol`, `huff`

Operational Considerations

Custom Hooks

The protocol can define custom hooks that facilitate extensions to the wallet and point to arbitrary code in external implementation contracts. This code is out of scope for this audit.

Checkpoint and Unreachable Configurations

The `checkpointer` contract acts as an oracle for the wallet's latest valid configuration state (`imageHash` and `checkpoint`). The signature validation logic in `BaseSig` strictly enforces that any new transaction must either be valid for the state reported by the `checkpointer` or prove a valid state transition to a configuration with a higher checkpoint.

However, the wallet has no native mechanism to disable or override a faulty checkpointer. If a checkpointer becomes malicious (e.g., its controlling keys are compromised) or unresponsive, it can report a future, unreachable configuration state. Because the user cannot produce a signature valid for this future state, every transaction attempt will fail the `UnusedSnapshot` check, permanently locking the wallet.

Signer Limit

The signature system facilitates theoretically arbitrarily complex configurations, where the total size of signatures are only limited by the inherent EVM gas limitations. The documentation warns that it is not recommended to exceed 200 signers. Our testing indicates, within the most gas-efficient scenarios, signature recovery can take less than 8M gas with 1000-1500 signers. Note that opcode costs can change in the future.

Static Signatures and State Channel Circumvention

Static signatures can be used to pre-approve an address to perform a specific transaction on behalf of the wallet, bypassing normal validation. This allows the wallet owner to delegate actions to other addresses. For example, they can be used to facilitate the use of protocols that require standing orders, such as legacy OpenSea. The validation requires that the `msg.sender` is the pre-approved address, the payload matches the pre-approved payload, and that the static signature has not expired.

However, should the `updateImageHash()` function be pre-approved and called through a static signature, it will not be possible to derive previous configurations from the newest `imageHash`, thereby disrupting the state channel. This is because previous configurations in the state channel are recovered via a linear chain of `BaseSig.recover()` calls, which is bypassed and unused when using static signatures. This could cause a divergence of the configuration between chains.

Static Signatures Persist Through Configuration Changes

Static signatures can be used to pre-approve an address to perform a specific transaction on behalf of the wallet, bypassing normal validation. This allows the wallet owner to delegate actions to other addresses. The signatures are valid until they've reached their expiration timestamp.

Given that their validity is timebound, it is possible for the authority of static signers to persist through configuration changes, even with a full rotation of wallet owners.

While the new owners can manually revoke static signatures, tracking the mapping requires reading the contract's event logs to see which signatures have been granted. Submitting revocations also introduces race conditions between the owners wiping the static signature and the static signer exercising the signature. Additionally, the pattern diverges from common approval patterns, such as `IERC721.approve()`, where the transfer of the token wipes any previous approvals.

Duplicate Signer Leaves

If a signer is duplicated in the merkle tree, i.e. there are two signer leaves pointing to the same address, this signer will be counted multiple times towards the signature threshold.

Implicit Session Blacklist Ordering

Implicit sessions require a blacklist. The blacklist addresses must be sorted or validation may be skipped.

Missing Hooks Will Silently Fail

`Calls._execute()` allows the execution of batched calls. From the batch, an unsuccessful call is treated differently depending on its `call.behaviorOnError`. However, if one of the calls is an attempt to call a missing hook, the call silently fails if no such hook is found. The batch would continue executing under the assumption that the hook succeeded.

Key Actors And Their Capabilities

Regular Signers

Regular signers are the primary controllers of a Sequence smart contract wallet. They are typically EOAs but can also be other smart contracts. The wallet's configuration, defined by a set of signers and a threshold of required signatures, governs its behavior.

Capabilities:

- Transaction Authorization: A sufficient number of regular signers (meeting the required threshold) can authorize any transaction from the wallet.
- Configuration Updates: Regular signers can sign messages to update the wallet's configuration, such as adding or removing other signers or changing the required threshold.
- Full Control: Collectively, regular signers have complete control over the wallet and its assets.

Identity Signer

The Identity Signer is a designated address within the wallet's session configuration. Its primary role is to cryptographically approve off-chain attestations, which are used to authorize implicit sessions.

Capabilities:

- Attestation Verification: The Identity Signer's signature on an attestation is what gives an implicit session key its power. It effectively vouches for the session key, allowing it to sign on behalf of the wallet.
- Central Trust Anchor for Implicit Sessions: It is a critical component for the security of implicit sessions. The `SessionManager` verifies that the attestation presented by an implicit session was indeed signed by the configured Identity Signer.

Implicit Sessions

Implicit sessions are a feature of the wallet's smart sessions system that allows a designated key to sign on behalf of the wallet after presenting an off-chain attestation signed by the wallet's identity signer.

Capabilities:

- Transaction Signing with Attestation: An implicit session key can authorize transactions on behalf of the wallet, but only when accompanied by a valid attestation from the identity signer.
- Subject to Constraints: Implicit sessions are subject to additional security measures, including a mandatory blacklist of disallowed addresses and session signers.

Explicit Sessions

Explicit sessions are session keys that are explicitly defined within the wallet's configuration. Their permissions are granted counterfactually and can be modified through a configuration update.

Capabilities:

- Batched Call Authorization: Explicit session keys are designed to authorize batches of transactions (calls) via signed payloads.
- Scoped Permissions: The capabilities of an explicit session key are strictly defined by its permission set, which can include:
- Value Limits: A maximum amount of native tokens that can be transferred.
- Deadlines: An expiration time for the session.
- Target and Function Whitelisting: Permissions can be restricted to specific smart contract addresses and even specific function selectors.
- Parameter Rules: Granular rules can be applied to the parameters of a function call, enforcing conditions on values at specific offsets in the calldata.
- Cumulative Limits: Rules can enforce cumulative limits over multiple transactions, such as a daily spending cap.

Static Signers

A static signature is a mechanism to pre-authorize a specific transaction payload within the wallet's configuration. It is not a distinct "signer" in the traditional sense but rather a rule that makes a particular signature perpetually valid.

Capabilities:

- Pre-approval of Transactions: A payload hash matching a hardcoded "subdigest" in the wallet's configuration is automatically considered valid, granting it the maximum possible weight towards meeting the signature threshold. This is useful for pre-approving actions like token approvals.

EntryPoint

The `EntryPoint` is a global, singleton smart contract that is a core component of the ERC-4337 (Account Abstraction) standard. It acts as a trusted intermediary for executing transactions on behalf of smart contract wallets.

Capabilities:

- UserOperation Execution: The `EntryPoint` receives bundles of `UserOperation` s from bundlers and orchestrates their validation and execution.
- Gas Management: It manages the gas payments for transactions, reimbursing bundlers from the wallet's own deposit or from a paymaster's deposit.
- Validation: It calls the `validateUserOp()` function on the smart contract wallet to ensure the `UserOperation` is authorized.

Bundlers

Bundlers are actors in the ERC-4337 ecosystem that monitor an alternative mempool of `UserOperation` s. They are responsible for packaging these operations into a single transaction and submitting it to the `EntryPoint`.

Capabilities:

- Transaction Bundling: Bundlers collect multiple `UserOperation` s and bundle them into a single transaction to be sent to the `EntryPoint` contract.
- Gas Fee Fronting: They pay the initial gas fees required to execute the bundled transactions.
- Reimbursement: After the `EntryPoint` validates and executes the bundle, the bundler is reimbursed for the gas costs, either from the sender's wallet or a paymaster.

Findings

SEQ-1

Cumulative Session Limits Can Be Bypassed, Leading to Asset Drain

• High ⓘ Fixed



Update

Marked as "Fixed" by the client.
Addressed in: `c264ef55e62e826a6f8c8a9438d36cbc4ae2ac89` .
The client provided the following explanation:

Also added re-entrancy guard here: `c938c502bfa80903148b4b0f6194a62de6f40f23`

File(s) affected: `SessionManager.sol`, `Calls.sol`

Description: Session can bypass any configured cumulative limits (e.g., daily spending caps), leading to a complete drain of the assets governed by that limit.

The `SessionManager` is responsible for validating a batch of transactions signed by a session. For explicit sessions, it enforces "cumulative" rules, such as a daily limit on token transfers. To do this, its validation logic requires that the last call in a transaction batch must be a self-call to `incrementUsageLimit()`, which updates the on-chain usage counter. The `SessionManager` validates the entire batch and, if all rules are met and the final `incrementUsageLimit()` call is correct, it approves the transaction for execution.

However, the wallet's main execution module, `Calls`, allows individual calls within a batch to specify different behaviors in case of failure. One of these, `BEHAVIOR_ABORT_ON_ERROR`, causes the entire batch execution to halt immediately if that specific call fails. However, as currently implemented, an attacker can craft a transaction batch where the session limit is consumed, a subsequent call is designed to fail with the `ABORT` behavior, and the final, required `incrementUsageLimit()` call is placed at the end. Because of the structure of the payload, the final call would never be executed, the execution loop would break after the second call, and the usage limits would never be incremented.

Exploit Scenario:

1. Alice configures a session for Bob that can transfer up to 100 USDC per day.
2. Bob designs a payload with three calls:
 1. A valid call to transfer 100 USDC to his own personal wallet. The `behaviorOnError` does not matter for this call.
 2. A call that is guaranteed to fail. `behaviorOnError` is set to `BEHAVIOR_ABORT_ON_ERROR`.
 3. A self-call to `SessionManager.incrementUsageLimit()` to update the onchain usage counter to 100. `behaviorOnError` is set to `BEHAVIOR_REVERT_ON_ERROR`
3. Bob submits the transaction. `signatureValidation` validates the entire payload, passing as the `incrementUsageLimit()` call is properly formed.
4. `Calls` begins executing the batch:
 1. The first call succeeds, sending Bob 100 USDC.
 2. The second call fails. Due to `BEHAVIOR_ABORT_ON_ERROR`, the loop breaks and the third call is not executed.
5. Bob can repeat this until all the USDC is drained from Alice's wallet.

Recommendation: Instead of checking whether the last call is a self-call to `incrementUsageLimit()`, make the check for the first call.

SEQ-2

Rule Changes Immediately Reset Usage Limits for a Permission

• Informational ⓘ Acknowledged

i Update

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

Docs updated to clarify

File(s) affected: `src/extensions/sessions/explicit/PermissionValidator.sol`

Description: An explicit session allows approving signers to execute predefined actions on behalf of the wallet. A session can have fine-grained control over approvals by defining a `Permission` for the action, which denotes a set of `Rules` the call must follow. A `Rule` can be defined to be `cumulative`, performing its validation on a running total value stored in the `ISapient` contract's storage rather than the call's immediate value. These totals are stored in the mapping `limitUsage[wallet][usageHash];`.

However, the `usageHash` is calculated as follows:

```
struct Permission {
    address target;
    ParameterRule[] rules;
}

bytes32 usageHash = keccak256(abi.encode(signer, permission, i)); // i == ruleIndex
```

Any changes to a `Permission` or its rules would change the `usageHash`, thereby resetting the usage limit.

Exploit Scenario:

1. Alice grants Bob an explicit session that allows him to transfer a cumulative 1000 USDC.
2. Bob transfers 500 USDC, increasing his `limitUsage[AlicesWallet][usageHash1] = 500`.
3. Alice decides to be more conservative and reduce the cumulative total to 900 USDC.
4. The new `usageHash` points to an empty storage position, immediately resetting the usage limit to `limitUsage[AlicesWallet][usageHash2] = 0`.
5. Bob is able to transfer an additional 900 USDC before being rate-limited, bringing the total to 1,400 USDC..

Recommendation: Consider redesigning how the `limitUsage` mapping is indexed such that a `Permission`'s totals can be accessed after its modification. Alternatively, document this behavior to inform users.

Auditor Suggestions

S1 Unlocked Pragma

Acknowledged

i Update

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

Not changed

Related Issue(s): [SWC-103](#)

Description: Every Solidity file specifies in the header a version number of the format `pragma solidity (^)0.8.*`. The caret (`^`) before the version number implies an unlocked pragma, meaning that the compiler will use the specified version *and above*, hence the term "unlocked".

Recommendation: For consistency and to prevent unexpected behavior in the future, we recommend to remove the caret to lock the file onto a specific Solidity version.

S2 Use Require Statements with Custom Errors

Acknowledged

i Update

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

Decided not to make changes that aren't security related

Description: Solidity allows the use of custom errors in conjunction with `require` statements. This common pattern can replace `if` statements and contribute to the readability and maintainability of the codebase.

Recommendation: Consider adopting this code pattern.

S3 Unused Import

Acknowledged

i Update

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

Imports not removed

File(s) affected: `src/modules/auth/Stage2Auth.sol` , `src/modules/interfaces/ICheckpointer.sol`

Description: The following files have unused imports:

- `ICheckpointer.sol` : `Payload.sol`
- `Stage2Auth.sol` : `Wallet.sol`

Recommendation: Remove the unused imports.

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 pose 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

Repo: <https://github.com/0xsequence/wallet-contracts-v3>

- e73...fc8 ./src/Factory.sol
- 4bc...5fc ./src/Stage1Module.sol
- 6b8...481 ./src/Stage2Module.sol
- 71d...6b9 ./src/Wallet.huff
- b8a...0bc ./src/Wallet.sol
- ed7...a70 ./src/extensions/passkeys/Passkeys.sol
- 35e...899 ./src/extensions/recovery/Recovery.sol
- 539...d97 ./src/extensions/sessions/SessionErrors.sol
- 462...b75 ./src/extensions/sessions/SessionManager.sol
- fc9...720 ./src/extensions/sessions/SessionSig.sol
- ec1...66f ./src/extensions/sessions/explicit/ExplicitSessionManager.sol
- 1f5...36a ./src/extensions/sessions/explicit/IExplicitSessionManager.sol
- 010...873 ./src/extensions/sessions/explicit/Permission.sol
- 2d2...560 ./src/extensions/sessions/explicit/PermissionValidator.sol
- 741...90a ./src/extensions/sessions/implicit/Attestation.sol
- 46a...120 ./src/extensions/sessions/implicit/ISignalsImplicitMode.sol
- 0ad...cef ./src/extensions/sessions/implicit/ImplicitSessionManager.sol
- 9e6...30c ./src/modules/Calls.sol
- ef0...00e ./src/modules/ERC4337v07.sol
- f25...a2b ./src/modules/Hooks.sol
- 591...156 ./src/modules/Implementation.sol
- 406...566 ./src/modules/Nonce.sol
- ff7...4e3 ./src/modules/Payload.sol
- 5ee...93e ./src/modules/Storage.sol
- 364...35d ./src/modules/auth/BaseAuth.sol
- a27...412 ./src/modules/auth/BaseSig.sol
- d43...3dc ./src/modules/auth/SelfAuth.sol
- 20f...ce9 ./src/modules/auth/Stage1Auth.sol
- b92...a40 ./src/modules/auth/Stage2Auth.sol
- d50...b13 ./src/modules/interfaces/IAccount.sol
- 022...a6d ./src/modules/interfaces/IAuth.sol
- 00b...3f2 ./src/modules/interfaces/ICheckpointer.sol
- 9d8...72b ./src/modules/interfaces/IDelegatedExtension.sol
- fdd...bb0 ./src/modules/interfaces/IERC1155Receiver.sol
- f9a...e89 ./src/modules/interfaces/IERC1271.sol

- ead...65c ./src/modules/interfaces/IERC223Receiver.sol
- 09a...6db ./src/modules/interfaces/IERC721Receiver.sol
- 29e...bd0 ./src/modules/interfaces/IERC777Receiver.sol
- b62...39c ./src/modules/interfaces/IEntryPoint.sol
- 559...53d ./src/modules/interfaces/IPartialAuth.sol
- 7bf...eda ./src/modules/interfaces/ISapient.sol
- 5f1...fca ./src/utills/LibBytes.sol
- 190...edf ./src/utills/LibOptim.sol

Test Suite Results

Test suite data was obtained using `forge test`. The test suite is excellent and provides insight into the proper functioning of the codebase.

Update: As of commit `0e8f002b5ae5b4975982d1e954bd0457ad4fd96d` in the audited repository, and while using `sequence.js` at commit `478f73e6ddf1fc448f01dc2781afbcf52d5bb67b`, all tests are passing. In addition, the test suite has increased in size overall.

Ran 2 tests **for** test/modules/Hooks.t.sol:MockImplementation

[PASS] testFunction() (gas: 147)

[PASS] testPayableFunction() (gas: 156)

Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 261.43ms (71.14ms CPU time)

Ran 2 tests **for** test/modules/Implementation.t.sol:ImplementationTest

[PASS] test_updateImplementation(address) (runs: 257, μ : 32951, \sim : 33106)

[PASS] test_updateImplementation_revertWhenNotSelf() (gas: 8869)

Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 354.99ms (135.70ms CPU time)

Ran 14 tests **for** test/modules/Hooks.t.sol:HooksTest

[PASS] test_addHook() (gas: 39722)

[PASS] test_addHook_revertWhenHookExists() (gas: 40561)

[PASS] test_addHook_revertWhenNotSelf() (gas: 14260)

[PASS] test_fallback() (gas: 118959)

[PASS] test_fallbackRevertWhenHookNotPayable() (gas: 122207)

[PASS] test_onERC1155BatchReceived(address,address,uint256[],uint256[],bytes) (runs: 257, μ : 19191, \sim : 19166)

[PASS] test_onERC1155Received(address,address,uint256,uint256,bytes) (runs: 257, μ : 14814, \sim : 14808)

[PASS] test_onERC721Received(address,address,uint256,bytes) (runs: 257, μ : 15012, \sim : 15006)

[PASS] test_payableFallback() (gas: 123926)

[PASS] test_receive() (gas: 15208)

[PASS] test_removeHook() (gas: 30344)

[PASS] test_removeHook_revertWhenHookDoesNotExist() (gas: 16953)

[PASS] test_tokenReceived(address,uint256,bytes) (runs: 257, μ : 14858, \sim : 14852)

[PASS] test_tokensReceived(address,address,address,uint256,bytes,bytes) (runs: 257, μ : 14960, \sim : 14955)

Suite result: ok. 14 passed; 0 failed; 0 skipped; finished in 390.48ms (159.06ms CPU time)

Ran 1 test **for** test/extensions/sessions/Attestation.t.sol:AttestationTest

[PASS] test_packAndUnpackAttestation((address,bytes4,bytes32,bytes32,bytes,(string,uint64))) (runs: 257, μ : 27968, \sim : 27908)

Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 390.89ms (171.42ms CPU time)

Ran 5 tests **for** test/utills/Base64.t.sol:Base64Test

[PASS] test_decode(bytes) (runs: 257, μ : 29426, \sim : 25861)

[PASS] test_decode_empty() (gas: 3260)

[PASS] test_encode(bytes) (runs: 257, μ : 48905, \sim : 40389)

[PASS] test_encode_empty() (gas: 3759)

[PASS] test_manual_cases() (gas: 34810)

Suite result: ok. 5 passed; 0 failed; 0 skipped; finished in 434.06ms (246.36ms CPU time)

Ran 3 tests **for** test/modules/Nonce.t.sol:NonceTest

[PASS] test_consumeNonce(uint256,uint256) (runs: 257, μ : 35045, \sim : 35045)

[PASS] test_consumeNonce_revertWhenBadNonce(uint256,uint256,uint256) (runs: 257, μ : 33475, \sim : 33708)

[PASS] test_readNonce(uint256,uint256) (runs: 257, μ : 31535, \sim : 31690)

Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 601.29ms (204.82ms CPU time)

Ran 23 tests **for** test/utills/LibBytes.t.sol:LibBytesTest

[PASS] test_readAddress(bytes,address,bytes) (runs: 257, μ : 10911, \sim : 10868)

[PASS] test_readAddress_outOfBounds(bytes) (runs: 256, μ : 9330, \sim : 9331)


```
[PASS] test_readBytes32(bytes,bytes32,bytes) (runs: 257, μ: 10871, ~: 10795)
[PASS] test_readBytes32_outOfBounds(bytes) (runs: 256, μ: 9713, ~: 9713)
[PASS] test_readBytes4(bytes,bytes4,bytes) (runs: 257, μ: 10813, ~: 10786)
[PASS] test_readBytes4_outOfBounds(bytes) (runs: 256, μ: 9485, ~: 9486)
[PASS] test_readFirstUint8(bytes) (runs: 257, μ: 9969, ~: 9967)
[PASS] test_readFirstUint8_emptyData() (gas: 6392)
[PASS] test_readRSVCompact(bytes,bytes32,uint256,bytes) (runs: 257, μ: 12746, ~: 12673)
[PASS] test_readRSVCompact_outOfBounds(bytes) (runs: 257, μ: 9394, ~: 9395)
[PASS] test_readUint16(bytes,uint16,bytes) (runs: 257, μ: 11242, ~: 11210)
[PASS] test_readUint160(bytes,uint160,bytes) (runs: 257, μ: 11115, ~: 11059)
[PASS] test_readUint160_outOfBounds(bytes) (runs: 256, μ: 9418, ~: 9419)
[PASS] test_readUint16_outOfBounds(bytes) (runs: 256, μ: 9153, ~: 9152)
[PASS] test_readUint24(bytes,uint24,bytes) (runs: 257, μ: 11330, ~: 11298)
[PASS] test_readUint24_outOfBounds(bytes) (runs: 256, μ: 9462, ~: 9463)
[PASS] test_readUint256(bytes,uint256,bytes) (runs: 257, μ: 11369, ~: 11298)
[PASS] test_readUint256_outOfBounds(bytes) (runs: 256, μ: 9493, ~: 9493)
[PASS] test_readUint64(bytes,uint64,bytes) (runs: 257, μ: 11380, ~: 11342)
[PASS] test_readUint64_outOfBounds(bytes) (runs: 256, μ: 9572, ~: 9573)
[PASS] test_readUint8(bytes,uint8,bytes) (runs: 257, μ: 10759, ~: 10729)
[PASS] test_readUint8_outOfBounds(bytes) (runs: 257, μ: 9814, ~: 9812)
[PASS] test_readUintX(bytes,uint256,uint256,bytes) (runs: 257, μ: 14831, ~: 14843)
Suite result: ok. 23 passed; 0 failed; 0 skipped; finished in 1.04s (822.31ms CPU time)
```

```
Ran 8 tests for test/extensions/sessions/implicit/ImplicitSessionManager.t.sol:ImplicitSessionManagerTest
[PASS] test_blacklist_unsortedSkipsBinarySearch(address[]) (runs: 257, μ: 10443191, ~: 7100330)
[PASS] test_blacklistedAddressNotAllowed(uint256,(address,bytes4,bytes32,bytes32,bytes,(string,uint64)),address[]) (runs: 257, μ: 6642781, ~: 5753094)
[PASS] test_blacklistedSessionSignerNotAllowed(uint256,(address,bytes4,bytes32,bytes32,bytes,(string,uint64)),address[]) (runs: 257, μ: 5793480, ~: 4060708)
[PASS] test_delegateCallNotAllowed((address,bytes4,bytes32,bytes32,bytes,(string,uint64))) (runs: 257, μ: 20503, ~: 20493)
[PASS] test_invalidImplicitResult((address,bytes4,bytes32,bytes32,bytes,(string,uint64))) (runs: 257, μ: 26842, ~: 26825)
[PASS] test_nonZeroValueNotAllowed((address,bytes4,bytes32,bytes32,bytes,(string,uint64)),uint256) (runs: 257, μ: 21521, ~: 21513)
[PASS] test_validImplicitCall((address,bytes4,bytes32,bytes32,bytes,(string,uint64)),address[]) (runs: 257, μ: 206030, ~: 194195)
[PASS] test_validImplicitCall_invalidSessionSigner((address,bytes4,bytes32,bytes32,bytes,(string,uint64))) (runs: 257, μ: 21187, ~: 21172)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 20.95s (20.93s CPU time)
```

```
Ran 2 tests for test/integrations/modules/ERC4337v07/ERC4337Entrypoint.t.sol:IntegrationERC4337v07Test
[PASS] test_Entrypoint_happypath(address) (runs: 257, μ: 269974, ~: 269971)
[PASS] test_Entrypoint_initcode(address) (runs: 257, μ: 259836, ~: 259826)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 72.58s (71.79s CPU time)
```

```
Ran 9 tests for test/modules/ERC4337v07.t.sol:ERC4337v07Test
[PASS] test_executeUserOp_executes_payload() (gas: 401299)
[PASS] test_executeUserOp_protected_from_reentry() (gas: 247261)
[PASS] test_executeUserOp_reverts_if_disabled(bytes) (runs: 257, μ: 6674910, ~: 6674905)
[PASS] test_executeUserOp_reverts_invalidEntryPoint(bytes,address) (runs: 257, μ: 39883, ~: 39878)
[PASS] test_validateUserOp_depositsMissingFunds(bytes32,uint256) (runs: 257, μ: 62141, ~: 62141)
[PASS] test_validateUserOp_returns_one_on_invalidSignature(bytes32) (runs: 257, μ: 24755, ~: 24755)
[PASS] test_validateUserOp_returns_zero_on_validSignature(bytes32) (runs: 257, μ: 65478, ~: 65478)
[PASS] test_validateUserOp_reverts_if_disabled((address,uint256,bytes,bytes,bytes32,uint256,bytes32,bytes,bytes),bytes32,uint256) (runs: 257, μ: 6653633, ~: 6653637)
[PASS] test_validateUserOp_reverts_invalidEntryPoint((address,uint256,bytes,bytes,bytes32,uint256,bytes32,bytes,bytes),bytes32,uint256,address) (runs: 257, μ: 18775, ~: 18768)
Suite result: ok. 9 passed; 0 failed; 0 skipped; finished in 20.26s (20.21s CPU time)
```

```
Ran 20 tests for
test/extensions/sessions/explicit/ExplicitSessionManager.t.sol:ExplicitSessionManagerTest
[PASS] test_incrementUsageLimit((bytes32,uint256)[]) (runs: 257, μ: 181748, ~: 184625)
[PASS] test_incrementUsageLimit_decrement((bytes32,uint256)) (runs: 257, μ: 38277, ~: 38277)
[PASS] test_incrementUsageLimit_twice((bytes32,uint256)[]) (runs: 257, μ: 213573, ~: 213838)
[PASS] test_validateExplicitCall(uint64,address,bytes4,bytes) (runs: 257, μ: 30241, ~: 30192)
[PASS] test_validateExplicitCall_DelegateCall(uint64) (runs: 257, μ: 28482, ~: 28490)
[PASS] test_validateExplicitCall_InvalidPermission(uint64) (runs: 257, μ: 29812, ~: 29832)
[PASS] test_validateExplicitCall_InvalidSelfCall_Value(uint64) (runs: 257, μ: 26997, ~: 27009)
[PASS] test_validateExplicitCall_InvalidSessionSigner(address,uint64) (runs: 257, μ: 26916, ~: 26924)
```

```
[PASS] test_validateExplicitCall_MissingPermission(uint64) (runs: 257, μ: 25663, ~: 25679)
[PASS] test_validateExplicitCall_MultipleValues(uint64,uint256,uint256,uint256,uint256) (runs: 257, μ: 110011, ~: 110063)
[PASS] test_validateExplicitCall_SessionExpired(uint64,uint64,uint64) (runs: 257, μ: 33141, ~: 32937)
[PASS] test_validateExplicitCall_ValueLimitExceeded(uint64) (runs: 257, μ: 30217, ~: 30223)
[PASS] test_validateExplicitCall_invalidChainId(uint64,uint256,address,bytes4,bytes) (runs: 257, μ: 33518, ~: 33482)
[PASS] test_validateLimitUsageIncrement_InvalidBehaviorOnError() (gas: 17597)
[PASS] test_validateLimitUsageIncrement_InvalidCall() (gas: 17148)
[PASS] test_validateLimitUsageIncrement_InvalidCallData() (gas: 20327)
[PASS] test_validateLimitUsageIncrement_OnlyFallbackAllowed() (gas: 17629)
[PASS] test_validateLimitUsageIncrement_RuleAndValue((bytes32,uint256),uint256) (runs: 257, μ: 23017, ~: 23011)
[PASS] test_validateLimitUsageIncrement_rule((bytes32,uint256)) (runs: 257, μ: 25292, ~: 25289)
[PASS] test_validateLimitUsageIncrement_value(uint256) (runs: 257, μ: 20011, ~: 20010)
Suite result: ok. 20 passed; 0 failed; 0 skipped; finished in 1.12s (1.12s CPU time)
```

Ran 3 tests **for** test/Factory.t.sol:FactoryTest

```
[PASS] test_deploy(address,bytes32) (runs: 257, μ: 69890, ~: 69968)
[PASS] test_deployForwardValue(address,bytes32,uint256) (runs: 257, μ: 77086, ~: 77216)
[PASS] test_deployTwice(address,bytes32) (runs: 257, μ: 1040431500, ~: 1040431302)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 22.56ms (22.37ms CPU time)
```

Ran 9 tests **for** test/extensions/sessions/explicit/PermissionValidator.t.sol:PermissionValidatorTest

```
[PASS] test_LimitUsageUpdated(address,bytes32,uint256) (runs: 257, μ: 35529, ~: 35762)
[PASS] test_validatePermission_Cumulative(uint256,uint256) (runs: 257, μ: 59803, ~: 60843)
[PASS] test_validatePermission_Equal(bytes4) (runs: 257, μ: 21430, ~: 21430)
[PASS] test_validatePermission_GreaterThanOrEqual(uint256,uint256) (runs: 257, μ: 15569, ~: 15522)
[PASS] test_validatePermission_NotEqual(uint256,uint256) (runs: 257, μ: 15804, ~: 15804)
[PASS] test_validatePermission_NotEqual_fail(uint256) (runs: 257, μ: 15444, ~: 15444)
[PASS]
test_validatePermission_OverflowCalldata_Zeroed((address,uint256,bytes,uint256,bool,bool,uint256),uint256,
, (bytes32,uint256)[]) (runs: 257, μ: 171826, ~: 170182)
[PASS] test_validatePermission_WithMaskAndOffset(bytes,bytes32,uint256,bytes) (runs: 257, μ: 26441, ~: 26382)
[PASS] test_validatePermission_WrongTarget(address,(address,uint256,bytes,uint256,bool,bool,uint256),
(bytes32,uint256)[]) (runs: 257, μ: 105525, ~: 106739)
Suite result: ok. 9 passed; 0 failed; 0 skipped; finished in 430.90ms (430.62ms CPU time)
```

Ran 16 tests **for** test/modules/Payload.t.sol:PayloadTest

```
[PASS] test_fromConfigUpdate(bytes32) (runs: 257, μ: 15579, ~: 15579)
[PASS] test_fromDigest(bytes32) (runs: 257, μ: 15532, ~: 15532)
[PASS] test_fromMessage(bytes) (runs: 257, μ: 17040, ~: 16767)
[PASS] test_fromPackedCalls((address,uint256,bytes,uint256,bool,bool,uint256)[],uint256,uint256) (runs: 257, μ: 4154250, ~: 3692792)
[PASS] test_fromPackedCalls_2bytes((address,uint256,bytes,uint256,bool,bool,uint256)) (runs: 257, μ: 6555372, ~: 6068422)
[PASS] test_fromPackedCalls_self((address,uint256,bytes,uint256,bool,bool,uint256)) (runs: 257, μ: 73029, ~: 72441)
[PASS] test_hashFor_invalidPayload(uint8) (runs: 257, μ: 15482, ~: 15418)
[PASS] test_hashFor_kindConfigUpdate(bytes32,address[],address) (runs: 257, μ: 161617, ~: 162133)
[PASS] test_hashFor_kindMessage(bytes,address[],address) (runs: 257, μ: 156646, ~: 154986)
[PASS] test_hashFor_kindMessage_as_digest(bytes,address[],address) (runs: 257, μ: 158432, ~: 163478)
[PASS] test_hashFor_kindTransactions((address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,address[],address) (runs: 257, μ: 2298262, ~: 2192097)
[PASS]
test_hashFor_kindTransactions(address,uint256,bytes,uint256,bool,bool,uint256,uint256,uint256,address[],a
ddress) (runs: 257, μ: 170722, ~: 165249)
[PASS] test_hash_kindConfigUpdate(bytes32,address[]) (runs: 257, μ: 189833, ~: 181217)
[PASS] test_hash_kindMessage(bytes,address[]) (runs: 257, μ: 194306, ~: 196517)
[PASS] test_hash_kindMessage_as_digest(bytes,address[]) (runs: 257, μ: 189218, ~: 192571)
[PASS]
test_hash_kindTransactions(address,uint256,bytes,uint256,bool,bool,uint256,uint256,uint256,address[])
(runs: 257, μ: 216279, ~: 225101)
Suite result: ok. 16 passed; 0 failed; 0 skipped; finished in 409.81s (409.59s CPU time)
```

Ran 3 tests **for** test/extensions/sessions/Permission.t.sol:PermissionTest

```
[PASS] test_fail_packRulesLengthExceedsMax((address,(bool,uint8,bytes32,uint256,bytes32)[])) (runs: 257,
μ: 687847, ~: 697936)
[PASS] test_packAndRead((address,(bool,uint8,bytes32,uint256,bytes32)[])) (runs: 257, μ: 8257623, ~: 5266982)
[PASS] test_packAndReadAtPointer((address,(bool,uint8,bytes32,uint256,bytes32)[]),bytes,bytes) (runs:
```

257, μ : 7914486, ~: 4226560)

Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 7.36s (7.36s CPU time)

Ran 2 tests for

test/integrations/extensions/sessions/SessionUsingERC4337.t.sol:IntegrationSessionUsing4337

[PASS] test_ExplicitSession_ERC4337_InvalidPayloadKind(address) (runs: 257, μ : 257795, ~: 257790)

[PASS] test_ImplicitSession_ERC4337_InvalidPayloadKind(address) (runs: 257, μ : 255288, ~: 255282)

Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 483.45s (483.45s CPU time)

Ran 3 tests for test/Wallet.t.sol:WalletTest

[PASS] test_doNotForwardWithValue(bytes32,uint256) (runs: 257, μ : 789060, ~: 789060)

[PASS] test_forward(bytes32,bytes,bytes) (runs: 257, μ : 892025, ~: 871694)

[PASS] test_forwardValueWithData(bytes32,bytes,bytes,uint256) (runs: 257, μ : 900932, ~: 879334)

Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 69.98ms (69.78ms CPU time)

Ran 1 test for

test/integrations/extensions/sessions/SessionLimitIncrementTest.t.sol:IntegrationSessionLimitIncrementTest

[PASS] test_SessionLimitIncrement_DoSSigner(string) (runs: 257, μ : 376317, ~: 376361)

Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 274.30s (274.30s CPU time)

Ran 7 tests for test/Guest.t.sol:GuestTest

[PASS] test_callFailsWithAbortBehavior((bool,(address,uint256,bytes,uint256,bool,bool,uint256)[],uint160,uint56),uint256) (runs: 257, μ : 2998915, ~: 2838146)

[PASS] test_callFailsWithIgnoreBehavior((bool,(address,uint256,bytes,uint256,bool,bool,uint256)[],uint160,uint56),uint256) (runs: 257, μ : 3463923, ~: 3002748)

[PASS] test_callFailsWithRevertBehavior((bool,(address,uint256,bytes,uint256,bool,bool,uint256)[],uint160,uint56),uint256) (runs: 257, μ : 3367642, ~: 2845961)

[PASS] test_delegateCallNotAllowed((bool,(address,uint256,bytes,uint256,bool,bool,uint256)[],uint160,uint56),uint256) (runs: 257, μ : 2611215, ~: 2414654)

[PASS] test_fallback((bool,(address,uint256,bytes,uint256,bool,bool,uint256)[],uint160,uint56)) (runs: 256, μ : 111599, ~: 118322)

[PASS] test_forwardPayment(uint256,uint256) (runs: 257, μ : 149337, ~: 150000)

[PASS] test_notEnoughGas((bool,(address,uint256,bytes,uint256,bool,bool,uint256)[],uint160,uint56),uint256) (runs: 257, μ : 3169459, ~: 3037266)

Suite result: ok. 7 passed; 0 failed; 0 skipped; finished in 613.45s (171.01s CPU time)

Ran 2 tests for test/extensions/sessions/SessionCalls.t.sol:SessionCallsTest

[PASS] test_fuzzForSkippingIncrementCall((address,uint256,bytes,uint256,bool,bool,uint256),(address,uint256,bytes,uint256,bool,bool,uint256),bool,bool) (runs: 257, μ : 428951, ~: 423064)

[PASS] test_fuzzForSkippingIncrementCall2((address,uint256,bytes,uint256,bool,bool,uint256),(address,uint256,bytes,uint256,bool,bool,uint256),(address,uint256,bytes,uint256,bool,bool,uint256),bool,bool,bool) (runs: 257, μ : 471742, ~: 470847)

Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 628.84s (628.83s CPU time)

Ran 7 tests for test/extensions/passkeys/Passkeys.t.sol:PasskeysTest

[PASS]

test_decodeSignature_abi((bool,bytes32,bytes32,bytes32,bytes32,bytes,bytes,bytes32,bool,uint256,uint256)) (runs: 257, μ : 248003, ~: 250441)

[PASS]

test_decodeSignature_packed((bool,bytes32,bytes32,bytes32,bytes32,bytes,bytes,bytes32,bool,uint256,uint256)) (runs: 257, μ : 289590, ~: 295696)

[PASS]

test_recoverSapientSignatureCompact_invalidSignature((uint256,bytes32,bool,bytes32,bool,uint256,bytes32,uint32),uint256) (runs: 257, μ : 388606, ~: 386260)

[PASS]

test_recoverSapientSignatureCompact_valid((uint256,bytes32,bool,bytes32,bool,uint256,bytes32,uint32)) (runs: 257, μ : 380456, ~: 380763)

[PASS] test_rootForPasskey_credentialId(bool,bytes32,bytes32,uint256) (runs: 257, μ : 150782, ~: 151069)

[PASS] test_rootForPasskey_metadataHash(bool,bytes32,bytes32,bytes32) (runs: 257, μ : 46673, ~: 46673)

[PASS] test_rootForPasskey_noMetadata(bool,bytes32,bytes32) (runs: 257, μ : 44423, ~: 44423)

Suite result: ok. 7 passed; 0 failed; 0 skipped; finished in 726.52s (177.48s CPU time)

Ran 3 tests for

test/integrations/extensions/sessions/SessionDenialOfService.t.sol:IntegrationSessionDenialOfServiceTest

[PASS] test_DOS_Wallet_SignerRace(uint160) (runs: 257, μ : 382703, ~: 382568)

[PASS] test_DOS_Wallet_SpaceBlockedHighRange(uint160) (runs: 257, μ : 300471, ~: 300630)

[PASS] test_DOS_Wallet_StaticSignatureRace(uint160) (runs: 257, μ : 389907, ~: 389762)

Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 1040.11s (1040.11s CPU time)

Ran 8 tests for test/modules/Calls.t.sol:CallsTest

[PASS] test_abort_on_error((address,uint256,bytes,uint256,bool,bool,uint256),bytes) (runs: 257, μ :


```
210303, ~: 202556)
[PASS] test_empty_payload_consumes_nonce(uint256,uint256,bytes) (runs: 257, μ: 183242, ~: 171880)
[PASS] test_error_flag_behavior((address,uint256,bytes,uint256,bool,bool,uint256),
(address,uint256,bytes,uint256,bool,bool,uint256),bytes) (runs: 257, μ: 246595, ~: 249621)
[PASS] test_execute(bytes32,(bool,(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint160,uint56),bytes) (runs: 256, μ: 346646, ~: 349269)
[PASS] test_invalid_signature((bool,(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint160,uint56),bytes,bytes) (runs: 257, μ: 2781231, ~: 2426147)
[PASS] test_not_enough_gas((address,uint256,bytes,uint256,bool,bool,uint256),uint256,bytes) (runs: 257,
μ: 203472, ~: 193569)
[PASS] test_revert_on_error((address,uint256,bytes,uint256,bool,bool,uint256),bytes) (runs: 257, μ:
212039, ~: 202937)
[PASS] test_self_execute((bool,(address,uint256,bytes,uint256,bool,bool,uint256)[],uint160,uint56))
(runs: 256, μ: 228479, ~: 217590)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 1134.17s (193.84s CPU time)
```

```
Ran 1 test for test/integrations/extensions/sessions/SessionSelfCall.t.sol:IntegrationSessionSelfCall
[PASS] test_ExplicitSession_SelfCall(bytes32) (runs: 257, μ: 340292, ~: 340292)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 45.53s (45.53s CPU time)
```

```
Ran 13 tests for test/extensions/recovery/Recovery.t.sol:RecoveryTest
[PASS] test_queue_payload(uint256,address,(uint8,bool,(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),uint64) (runs: 257, μ: 1202929, ~: 1118214)
[PASS] test_queue_payload_already_queued_fail(uint256,address,(uint8,bool,
(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),uint256,uint256) (runs: 257, μ: 1631418, ~: 1464409)
[PASS] test_queue_payload_ecdsa_with_code(uint256,address,(uint8,bool,
(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),uint64,bytes) (runs: 257, μ: 1160757, ~: 1058561)
[PASS] test_queue_payload_erc1271(address,bytes,address,(uint8,bool,
(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),uint64,bytes) (runs: 257, μ: 1194606, ~: 1076457)
[PASS] test_queue_payload_erc1271_invalid_signature_fail(address,bytes,address,(uint8,bool,
(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),uint64,bytes,bytes4) (runs: 257, μ: 1186576, ~:
1124063)
[PASS] test_queue_payload_erc1271_revert_fail(address,bytes,address,(uint8,bool,
(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),uint64,bytes,bytes) (runs: 257, μ: 810221, ~: 762502)
[PASS] test_queue_payload_invalid_signature_fail_has_code(uint256,address,(uint8,bool,
(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),uint64,address,bytes) (runs: 257, μ: 1118923, ~:
1057900)
[PASS] test_queue_payload_invalid_signature_fail_no_code(uint256,address,(uint8,bool,
(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),uint64,address) (runs: 257, μ: 1101654, ~: 985875)
[PASS] test_recoverBranch((address,uint24,uint64)[],address,bytes32) (runs: 257, μ: 11604246, ~: 5372583)
[PASS] test_recover_fail_invalid_signature_flag(address,bytes32,uint8,bytes) (runs: 257, μ: 10604, ~:
10599)
[PASS] test_recover_sapient_signature_compact((uint256,address,(uint8,bool,
(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),uint256,uint256,uint256,uint256,
(address,uint24,uint64)[],(address,uint24,uint64)[])) (runs: 257, μ: 27572933, ~: 16303047)
[PASS] test_recover_sapient_signature_compact_fail_deltaTime((uint256,address,(uint8,bool,
(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),uint256,uint256,uint256,uint256,
(address,uint24,uint64)[],(address,uint24,uint64)[])) (runs: 257, μ: 28757104, ~: 19416207)
[PASS] test_recover_sapient_signature_compact_fail_minTimestamp((uint256,address,(uint8,bool,
(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),uint256,uint256,uint256,uint256,
(address,uint24,uint64)[],(address,uint24,uint64)[])) (runs: 257, μ: 31473477, ~: 21166726)
Suite result: ok. 13 passed; 0 failed; 0 skipped; finished in 796.67s (777.38s CPU time)
```

```
Ran 3 tests for
test/integrations/extensions/sessions/SessionSignatureAbuse.t.sol:IntegrationSessionSignatureAbuseTest
[PASS] test_CrossChain_ReplayProtection_RepeatableCall() (gas: 348531)
[PASS] test_PermissionIndex_OutOfRange_reverts_MissingPermission((address,
(bool,uint8,bytes32,uint256,bytes32)[])[],uint8) (runs: 257, μ: 98559805, ~: 80499148)
[PASS] test_SessionSigner_ZeroAddress_reverts_InvalidSessionSigner(uint8,bytes32) (runs: 257, μ: 309842,
~: 309842)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 480.89s (480.89s CPU time)
```

```
Ran 16 tests for test/extensions/sessions/SessionManager.t.sol:SessionManagerTest
[PASS] testBehaviorAbortOnErrorCallsRevert(address,bytes) (runs: 256, μ: 155122, ~: 155013)
[PASS] testExplicitSessionBehaviorAbortOnErrorReverts(address,bytes) (runs: 256, μ: 132933, ~: 132830)
[PASS] testExplicitSessionBehaviorIgnoreErrorAllowed(address,bytes) (runs: 256, μ: 140254, ~: 140134)
[PASS] testExplicitSessionOnlyFallbackAllowed(address,bytes) (runs: 256, μ: 141040, ~: 140949)
[PASS] testIncrementCallOnlyFallbackReverts(address,bytes) (runs: 256, μ: 169189, ~: 169053)
[PASS] testIncrementOverMultipleCalls_Invalid(address,address,uint256,uint256,uint256,uint256) (runs: 257, μ: 259119, ~: 257796)
[PASS] testIncrementOverMultipleCalls_Valid(address,address,uint256,uint256,uint256,uint256) (runs: 257, μ: 271055, ~: 271329)
[PASS] testIncrementReentrancy() (gas: 8183237)
[PASS] testInvalidCallsLengthReverts(bytes) (runs: 257, μ: 10957, ~: 10904)
[PASS] testInvalidDelegateCallReverts((address,bytes4,bytes32,bytes32,bytes,(string,uint64)),bytes,address) (runs: 257, μ: 160458, ~: 160419)
[PASS] testInvalidPayloadKindReverts() (gas: 14231)
[PASS] testInvalidSpaceReverts(uint160,bytes) (runs: 257, μ: 15031, ~: 15152)
[PASS] testOnlyFallbackCallsAllowed() (gas: 165946)
[PASS] testValidExplicitSessionSignature(bytes4,uint256,uint256,address,address,bool) (runs: 257, μ: 272113, ~: 271691)
[PASS] testValidImplicitSessionSignature((address,bytes4,bytes32,bytes32,bytes,(string,uint64))) (runs: 257, μ: 171929, ~: 171954)
[PASS] testValidLinearExecution(address,bytes) (runs: 256, μ: 140711, ~: 140593)
Suite result: ok. 16 passed; 0 failed; 0 skipped; finished in 1172.78s (1183.49s CPU time)
```

```
Ran 3 tests for test/integrations/extensions/sessions/SessionValueForwarding.t.sol:IntegrationSessionValueForwardingTest
[PASS] test_ValueForwarding_Explicit_Limited(uint256,uint160,address,uint256,uint256) (runs: 257, μ: 802981, ~: 802905)
[PASS] test_ValueForwarding_Explicit_OverMultipleCalls(uint256,uint160,address,uint256,uint256) (runs: 257, μ: 1105424, ~: 1106633)
[PASS] test_ValueForwarding_NoRequiredIncrementAfterIncrement(uint256,uint160,address,uint256,uint256) (runs: 257, μ: 697088, ~: 697194)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 671.40s (1221.89s CPU time)
```

```
Ran 18 tests for test/Stage1Module.t.sol:TestStage1Module
[PASS] test_1271_single_signer(uint16,uint56,uint8,uint256,bytes32,bool) (runs: 257, μ: 174184, ~: 174170)
[PASS] test_fails_on_low_weight(uint16,uint56,uint8,uint256,bytes32,bool) (runs: 257, μ: 135158, ~: 135224)
[PASS] test_forbid_reentrancy(uint16,uint56,uint8,uint256,(uint8,bool,(address,uint256,bytes,uint256,bool,bool,uint256)[],uint256,uint256,bytes,bytes32,bytes32,address[]),bool,bool) (runs: 257, μ: 3737384, ~: 3495914)
[PASS] test_invalid_is_valid_signature((bytes32,uint16,uint56,uint8,uint256,uint256,bool)) (runs: 257, μ: 161175, ~: 161114)
[PASS] test_receiveETH_stage1() (gas: 79875)
[PASS] test_receiveETH_stage2((uint256,uint256,uint16,uint16,uint56)) (runs: 257, μ: 275929, ~: 275919)
[PASS] test_recover_partial_signature(((uint8,bool,(address,uint256,bytes,uint256,bool,bool,uint256)[],uint256,uint256,bytes,bytes32,bytes32,address[]),uint16,uint56,uint8,uint256,bool,bool,bool)) (runs: 257, μ: 797842, ~: 689794)
[PASS] test_recover_sapient_as_if_nested(((uint8,bool,(address,uint256,bytes,uint256,bool,bool,uint256)[],uint256,uint256,bytes,bytes32,bytes32,address[]),uint16,uint56,uint8,uint256,address)) (runs: 257, μ: 851883, ~: 783232)
[PASS] test_recover_sapient_as_if_nested_wrong_signature_fail(((uint8,bool,(address,uint256,bytes,uint256,bool,bool,uint256)[],uint256,uint256,bytes,bytes32,bytes32,address[]),uint16,uint56,uint8,uint256,address),uint56) (runs: 257, μ: 1291909, ~: 1272848)
[PASS] test_reverts_invalid_static_signature_expired(bytes32,bytes32,uint256,uint256) (runs: 257, μ: 125933, ~: 125929)
[PASS] test_reverts_set_static_signature_not_self(bytes32,bytes32,address,uint96,address) (runs: 257, μ: 85530, ~: 85530)
[PASS] test_reverts_update_to_zero_image_hash(uint16,uint56,uint8,uint256,bool) (runs: 257, μ: 225952, ~: 226035)
[PASS] test_send_many_transactions(uint256,uint8,uint16,uint256,uint256,bool) (runs: 257, μ: 782855, ~: 760479)
[PASS] test_static_signature_any_address(bytes32,bytes32,uint256,uint256,address) (runs: 257, μ: 122382, ~: 122368)
[PASS] test_static_signature_specific_address(bytes32,bytes32,uint256,uint256,address,address) (runs: 257, μ: 125237, ~: 125191)
[PASS] test_update_config((uint16,uint56,uint8,uint256,bool,uint16,uint56,uint8,uint256,bytes32)) (runs: 257, μ: 300869, ~: 300998)
[PASS] test_update_image_hash_then_zero((uint16,uint56,uint8,uint256,uint16,uint56,uint8,uint256,bool)) (runs: 257, μ: 341186, ~: 341291)
```



```
[PASS] test_update_image_hash_twice((uint16,uint56,uint8,uint256,uint16,uint56,uint8,uint256,bool))
(runs: 257, μ: 358129, ~: 358225)
Suite result: ok. 18 passed; 0 failed; 0 skipped; finished in 754.17s (2229.33s CPU time)
```

Ran 3 tests for

```
test/integrations/extensions/recovery/RecoveryDenialOfService.t.sol:IntegrationRecoveryDenialOfService
[PASS] test_Recovery_CallableByAnyone(uint160,uint64,uint24,address,address) (runs: 257, μ: 343152, ~: 343097)
[PASS] test_Recovery_DenialOfService(uint160,uint64,uint24) (runs: 257, μ: 395533, ~: 395478)
[PASS] test_Recovery_Reexecution(uint160,uint64,uint24) (runs: 257, μ: 368538, ~: 368477)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 1176.70s (634.28s CPU time)
```

Ran 27 tests for test/modules/BaseSig.t.sol:BaseSigTest

```
[PASS] test_checkpoint_current_snapshot(((uint8,bool,(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),address,uint56,uint256,uint8,uint8,bytes)) (runs:
256, μ: 186106, ~: 188096)
[PASS] test_checkpoint_different_image_hash_fail(((uint8,bool,
(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),address,bytes,uint56,uint256,uint8,uint8,bytes32))
(runs: 257, μ: 743609, ~: 673387)
[PASS] test_checkpoint_disabled(((uint8,bool,(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),address,uint56,uint56,uint256,uint8,uint8,bytes))
(runs: 256, μ: 191539, ~: 191925)
[PASS] test_checkpoint_disabled_old_checkpoint(((uint8,bool,
(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),address,uint56,uint56,uint256,uint8,uint8,bytes,bytes
32,uint56)) (runs: 256, μ: 187407, ~: 185262)
[PASS] test_checkpoint_disabled_with_chain(((uint8,bool,
(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),address,bytes,uint256,uint256,uint56,uint56,uint56))
(runs: 256, μ: 254310, ~: 254511)
[PASS] test_checkpoint_higher_checkpoint_fail(((uint8,bool,
(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),address,bytes,uint56,uint56,uint256,uint8,uint8))
(runs: 257, μ: 722414, ~: 643824)
[PASS] test_checkpoint_migrate_from_no_checkpoint(((uint8,bool,
(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),address,uint56,uint56,uint256,uint256,uint8,uint8,byt
es)) (runs: 256, μ: 252286, ~: 255946)
[PASS] test_checkpoint_migrate_from_snapshot_to_snapshot(((uint8,bool,
(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),address,address,uint56,uint56,uint256,uint8,uint8,byt
es)) (runs: 256, μ: 261222, ~: 263720)
[PASS] test_checkpoint_migrate_to_no_checkpoint(((uint8,bool,
(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),address,uint56,uint56,uint256,uint256,uint8,uint8,byt
es)) (runs: 256, μ: 250538, ~: 250203)
[PASS] test_checkpoint_old_checkpoint_with_chain(((uint8,bool,
(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),address,bytes,uint256,uint256,uint56,uint56,uint56,by
tes32)) (runs: 256, μ: 256823, ~: 256168)
[PASS] test_checkpoint_past_with_chain(((uint8,bool,(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),address,bytes,uint256,uint256,uint56,uint56)) (runs:
256, μ: 254474, ~: 252133)
[PASS] test_checkpoint_unused_snapshot_chain_fail(((uint8,bool,
(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),address,bytes,uint256,uint256,uint56,uint56,uint56,by
tes32)) (runs: 256, μ: 236522, ~: 233716)
[PASS] test_recover_anyAddressSubdigest(((uint8,bool,(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),address,uint8)[],(address,uint8)[],uint16,uint56))
(runs: 256, μ: 19852376, ~: 13492071)
[PASS] test_recover_chained_low_weight_fail(((uint8,bool,
(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),uint256,uint256,uint256,uint256,uint256)) (runs: 257,
μ: 808958, ~: 714933)
[PASS] test_recover_chained_signature_single_case((uint8,bool,
(address,uint256,bytes,uint256,bool,bool,uint256)[],uint256,uint256,bytes,bytes32,bytes32,address[]))
(runs: 257, μ: 893815, ~: 789272)
[PASS] test_recover_chained_wrong_checkpoint_order_fail(((uint8,bool,
(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),uint256,uint256,uint256,uint256,uint256,uint256))
(runs: 257, μ: 797804, ~: 726241)
```

```
[PASS] test_recover_invalid_signature_flag((uint8,bool,(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),uint8,uint8,uint8) (runs: 257, μ: 508424, ~: 442647)
[PASS] test_recover_nested_config(((address,uint8)[],(address,uint8)[],(address,uint8)[],(address,uint8)
[],(uint8,bool,(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),uint16,uint56,uint16,uint8,uint8,uint256,bool))
(runs: 256, μ: 74391516, ~: 67716094)
[PASS] test_recover_one_1271_invalid_signature_bad_return_fail(((address,uint8)[],(address,uint8)[],
(uint8,bool,(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),uint16,uint56,uint8,address,bytes,bytes4)) (runs:
257, μ: 22268067, ~: 13984708)
[PASS] test_recover_one_1271_invalid_signature_revert_fail(((address,uint8)[],(address,uint8)[],
(uint8,bool,(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),uint16,uint56,uint8,address,bytes,bytes)) (runs: 257,
μ: 23241956, ~: 13638380)
[PASS] test_recover_one_1271_signer(((address,uint8)[],(address,uint8)[],(uint8,bool,
(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),uint16,uint56,uint8,address,bytes)) (runs: 257, μ:
22259722, ~: 15203400)
[PASS] test_recover_one_sapient_signer(((address,uint8)[],(address,uint8)[],(uint8,bool,
(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),uint16,uint56,uint8,address,bytes,bytes32,bool))
(runs: 257, μ: 24912941, ~: 15589167)
[PASS] test_recover_one_signer(((address,uint8)[],(address,uint8)[],(uint8,bool,
(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]),uint16,uint56,uint8,uint256,bool)) (runs: 257, μ:
22512434, ~: 14988067)
[PASS] test_recover_random_config_unsigned(uint256,uint256) (runs: 257, μ: 121640, ~: 96329)
[PASS] test_recover_random_config_unsigned_skewed_left(uint256) (runs: 257, μ: 225682, ~: 225223)
[PASS] test_recover_random_config_unsigned_skewed_right(uint256) (runs: 257, μ: 192991, ~: 192143)
[PASS] test_recover_subdigest(((uint8,bool,(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]), (address,uint8)[], (address,uint8)[],uint16,uint56))
(runs: 257, μ: 22193501, ~: 14139539)
Suite result: ok. 27 passed; 0 failed; 0 skipped; finished in 1177.90s (5370.04s CPU time)
```

Ran 20 tests for test/extensions/sessions/SessionSig.t.sol:SessionSigTest

```
[PASS] testAttestationOptimisation((address,bytes4,bytes32,bytes32,bytes,(string,uint64)),
(address,bytes4,bytes32,bytes32,bytes,(string,uint64))) (runs: 257, μ: 278669, ~: 276553)
[PASS] testConfiguration_duplicateBlacklistNodes(address[5],uint8) (runs: 257, μ: 27596, ~: 25432)
[PASS] testConfiguration_duplicateBlacklistNodes_inBranch(address[5],uint8) (runs: 257, μ: 30727, ~:
29104)
[PASS] testConfiguration_duplicateIdentityNodes(address,address) (runs: 257, μ: 15817, ~: 15817)
[PASS] testConfiguration_duplicateIdentityNodes_inBranch(address) (runs: 257, μ: 19225, ~: 19225)
[PASS] testConfiguration_invalidNode(uint8) (runs: 257, μ: 17864, ~: 18015)
[PASS] testConfiguration_largeBlacklist(address[]) (runs: 257, μ: 18203295, ~: 9891092)
[PASS] testEmptyPermissionsStructSize_direct(address,uint256,uint256,uint64) (runs: 257, μ: 22398, ~:
22398)
[PASS] testHashCallCollision(uint256,(uint8,bool,(address,uint256,bytes,uint256,bool,bool,uint256)
[],uint256,uint256,bytes,bytes32,bytes32,address[]), (uint8,bool,
(address,uint256,bytes,uint256,bool,bool,uint256)[],uint256,uint256,bytes,bytes32,bytes32,address[]))
(runs: 257, μ: 4384267, ~: 4482155)
[PASS] testLargeTopology(address[],uint256,bool,address[]) (runs: 256, μ: 9223810, ~: 7263370)
[PASS] testMultipleExplicitSignatures(bool) (runs: 257, μ: 263842, ~: 263967)
[PASS] testMultipleImplicitSignatures((address,bytes4,bytes32,bytes32,bytes,(string,uint64))) (runs: 257,
μ: 251143, ~: 248954)
[PASS] testRecover_invalidAttestationIndex((address,bytes4,bytes32,bytes32,bytes,
(string,uint64)),uint256,uint256) (runs: 257, μ: 158849, ~: 150869)
[PASS] testRecover_invalidBlacklist_requiredForImplicitSigner((address,bytes4,bytes32,bytes32,bytes,
(string,uint64))) (runs: 257, μ: 103648, ~: 103529)
[PASS] testRecover_invalidIdentitySigner_noMatchAttestationSigner((address,bytes4,bytes32,bytes32,bytes,
(string,uint64))) (runs: 257, μ: 146708, ~: 145971)
[PASS] testRecover_invalidIdentitySigner_noSignersEncoded((address,bytes4,bytes32,bytes32,bytes,
(string,uint64))) (runs: 257, μ: 132942, ~: 132189)
[PASS] testRecover_invalidIdentitySigner_unset() (gas: 58692)
[PASS] testRecover_invalidSessionSigner(bool) (runs: 257, μ: 111783, ~: 111755)
[PASS] testSingleExplicitSignature(bool) (runs: 257, μ: 167945, ~: 167939)
[PASS] testSingleImplicitSignature((address,bytes4,bytes32,bytes32,bytes,(string,uint64))) (runs: 257, μ:
175179, ~: 174200)
Suite result: ok. 20 passed; 0 failed; 0 skipped; finished in 802.79s (1009.59s CPU time)
```

Ran 32 test suites in 1220.03s (12515.74s CPU time): 257 tests passed, 0 failed, 0 skipped (257 total tests)

Code Coverage

Coverage data was obtained using `forge coverage --no-match-coverage "(script|test)"`. All original contracts are achieving 100% branch coverage.

Update: As confirmed by the client, coverage data cannot be obtained as of the updated commit due to encountering a "stack too deep" error. The coverage data below is from the initially audited commit.

File	% Lines	% Statements	% Branches	% Funcs
src/Estimator.sol	0.00% (0/39)	0.00% (0/46)	0.00% (0/9)	0.00% (0/3)
src/Factory.sol	100.00% (5/5)	100.00% (5/5)	100.00% (1/1)	100.00% (1/1)
src/Guest.sol	100.00% (30/30)	100.00% (36/36)	100.00% (7/7)	100.00% (2/2)
src/Simulator.sol	0.00% (0/39)	0.00% (0/46)	0.00% (0/8)	0.00% (0/1)
src/Stage1Module.sol	100.00% (2/2)	100.00% (2/2)	100.00% (0/0)	100.00% (1/1)
src/Stage2Module.sol	100.00% (2/2)	100.00% (2/2)	100.00% (0/0)	100.00% (1/1)
src/extensions/passkeys/Pass keys.sol	100.00% (40/40)	100.00% (47/47)	100.00% (4/4)	100.00% (3/3)
src/extensions/recovery/Recovery.sol	100.00% (71/71)	100.00% (82/82)	100.00% (10/10)	100.00% (8/8)
src/extensions/sessions/SessionManager.sol	100.00% (39/39)	97.92% (47/48)	100.00% (8/8)	100.00% (1/1)
src/extensions/sessions/SessionSig.sol	100.00% (136/136)	100.00% (151/151)	100.00% (19/19)	100.00% (7/7)
src/extensions/sessions/explicit/ExplicitSessionManager.sol	100.00% (60/60)	100.00% (79/79)	100.00% (17/17)	100.00% (3/3)
src/extensions/sessions/explicit/Permission.sol	100.00% (23/23)	100.00% (26/26)	100.00% (1/1)	100.00% (3/3)
src/extensions/sessions/explicit/PermissionValidator.sol	100.00% (50/50)	98.15% (53/54)	100.00% (16/16)	100.00% (3/3)
src/extensions/sessions/implicit/ImplicitSessionManager.sol	100.00% (27/27)	100.00% (29/29)	100.00% (10/10)	100.00% (2/2)
src/modules/Calls.sol	100.00% (40/40)	100.00% (48/48)	100.00% (9/9)	100.00% (3/3)
src/modules/Hooks.sol	100.00% (31/31)	100.00% (26/26)	100.00% (5/5)	100.00% (11/11)
src/modules/Implementation.sol	100.00% (11/11)	100.00% (7/7)	100.00% (0/0)	100.00% (5/5)
src/modules/Nonce.sol	100.00% (12/12)	100.00% (11/11)	100.00% (1/1)	100.00% (3/3)
src/modules/Payload.sol	100.00% (72/72)	100.00% (91/91)	100.00% (20/20)	100.00% (10/10)

File	% Lines	% Statements	% Branches	% Funcs
src/modules/Storage.sol	100.00% (10/10)	100.00% (8/8)	100.00% (0/0)	100.00% (4/4)
src/modules/auth/BaseAuth.sol	100.00% (48/48)	100.00% (50/50)	100.00% (6/6)	100.00% (9/9)
src/modules/auth/BaseSig.sol	100.00% (201/201)	100.00% (239/239)	100.00% (33/33)	100.00% (8/8)
src/modules/auth/SelfAuth.sol	100.00% (3/3)	100.00% (2/2)	100.00% (1/1)	100.00% (1/1)
src/modules/auth/Stage1Auth.sol	100.00% (13/13)	100.00% (12/12)	100.00% (1/1)	100.00% (3/3)
src/modules/auth/Stage2Auth.sol	100.00% (9/9)	100.00% (11/11)	100.00% (1/1)	100.00% (3/3)
src/utls/Base64.sol	98.36% (60/61)	98.25% (56/57)	100.00% (5/5)	100.00% (4/4)
src/utls/LibBytes.sol	100.00% (51/51)	100.00% (39/39)	100.00% (0/0)	100.00% (12/12)
src/utls/LibOptim.sol	100.00% (15/15)	100.00% (11/11)	100.00% (0/0)	100.00% (4/4)
src/utls/P256.sol	32.50% (13/40)	34.29% (12/35)	0.00% (0/4)	20.00% (1/5)
src/utls/WebAuthn.sol	21.55% (25/116)	20.00% (23/115)	7.69% (1/13)	10.00% (1/10)
Total	84.80% (1099/1296)	85.16% (1205/1415)	84.21% (176/209)	87.31% (117/134)

Changelog

- 2025-08-14 - Initial report
- 2025-09-12 - 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' 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 may not be represented as such. No third party is entitled to rely on the report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. Quantstamp does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party, or any open source or third-party software, code, libraries, materials, or information to, called by, referenced by or accessible through the report, its content, or any related services and products, any hyperlinked websites, or any other websites or mobile applications, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third party. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

