Quantstamp

Startale - Account Abstraction

Executive Summary

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

Type	Account Abstraction
Timeline	2025-04-14 through 2025-04-25
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Source Code	• StartaleLabs/scs-aa-account-contracts #be5beb1
Auditors	• Andy Lin Senior Auditing Engineer • Yamen Merhi Auditing Engineer • Nikita Belenkov Senior Auditing Engineer

Documentation quality	Undetermined
Test quality	Medium
Total Findings	15 Fixed: 10 Acknowledged: 3 Mitigated: 2
High severity findings	0
Medium severity findings	2 Fixed: 1 Mitigated: 1
Low severity findings	10 Fixed: 7 Acknowledged: 2 Mitigated: 1
Undetermined severity findings	0
Informational findings	3 Fixed: 2 Acknowledged: 1

Summary of Findings

The Startale Smart Account project implements a modular account abstraction framework adhering to ERC-4337 and ERC-7579 standards, designed for flexibility through installable modules. The codebase is generally well-structured, utilizes modern Solidity practices, and follows established standards. The test quality was assessed as medium, with decent coverage but room for improvement on critical components like the `ModuleManager`.

The audit identified no high severity vulnerabilities, but found two medium severity issues: one concerning incorrect context propagation to hooks during module installation via "Enable Mode" (STAA-1), and another related to potential gas exhaustion during the module cleanup process in redelegation (STAA-2). Several Low and Informational findings were also noted, primarily related to gas optimization, potential edge cases in module interactions, and adherence to best practices. We recommend the Startale team address all identified issues to enhance the system's security, robustness, and gas efficiency before deployment.

Fix Review Update: The team has either fixed or acknowledged all issues and provided tests for their fixes.

ID	DESCRIPTION	SEVERITY	STATUS
STAA-1	**Incorrect Context Sent to Hook when Installing Module via Enable Mode**	Medium	Fixed
STAA-2	**Potential Gas Exhaustion Vulnerability in Redelegation Cleanup**	Medium	Mitigated
STAA-3	**Potential Replay Attack Vector in** `ECDSAValidator.validateSignatureWithData()`	Low	Acknowledged
STAA-4	`AssociatedArrayLib` **Can Risk Not Complying with EIP-7562**	Low	Fixed

ID	DESCRIPTION	SEVERITY	STATUS
STAA-5	Reliance on Undeployed Contracts on Soneium Network	Low	Fixed
STAA-6	Potential Memory Corruption in `withdrawDepositTo()` Assembly	Low	Fixed
STAA-7	Potential Unintended Return Data in `_fallback()` Function	Low	Fixed
STAA-8	Missing Boundary Checks in `DataParserLib` Library	Low	Fixed
STAA-9	Gas Griefing Risk via Calldata Stuffing	Low	Acknowledged
STAA-10	Signature Bypass Vulnerability in `ECDSAValidator.validateSignatureWithData()`	Low	Fixed
STAA-11	Insufficient Protection Against Fallback Handler Overriding Core Functions	Low	Mitigated
STAA-12	Fallback Handlers Persist After Redelegation Leading to Potential Risks	Low	Fixed
STAA-13	Potential Duplicate Entries in ERC7702 `accountStorageBases` Array on Redelegation	Informational	Fixed
STAA-14	Incorrect Error Handling in Module Enable Mode Signature Validation Deviates From ERC-4337 Specification	Informational	Acknowledged
STAA-15	Missing ERC-165 Support	Informational	Fixed

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 includes all contracts code under `src/` aside from the `Greeter` contract.

Files Included

Files: `src/`
Repo: `https://github.com/StartaleLabs/scs-aa-account-contracts`

Files Excluded

Files: `src/Greeter.sol`

Operational Considerations

1. **Module Security:** Account security depends on correctness of installed modules (Validators, Executors, Hooks, etc.), which execute arbitrary logic in critical flows.
2. **Executor Module Trust:** Executors can call arbitrary functions via `executeFromExecutor()`, including `delegatecall()`, which requires trust or enforcement through hooks. However, executors can uninstall the hooks using a carefully crafted `delegatecall()`, so the delegate call targets must be managed very carefully.
3. **Initialization Trust:** Account initialization via `Bootstrap.sol` relies entirely on the integrity of `initData`, which can be compromised by faulty factories or off-chain generators.
4. **Default Validator Security:** The `_DEFAULT_VALIDATOR` address is immutable and must be secure, as it's used initially or as a fallback.
5. **Emergency Uninstall Signer Security:** Relies on a secure EIP-712 signature; compromise of the signing key bypasses normal uninstall and timelock protections.
6. **Factory Ownership/Staking Security:** Factories are `Ownable` and control staking; owner key compromise could lead to loss of staked funds.
7. **UUPS Upgrade Security:** Upgrade paths must be validated and secured; `upgradeToAndCall` assumes hooks/validators prevent malicious upgrades.
8. **Fallback Handler Security:** Fallback handlers can run arbitrary logic; handler installation must be restricted to avoid vulnerabilities.
9. **External Library Security:** Depends on correctness of external libraries (Solady, OpenZeppelin, etc.); vulnerabilities in these can affect the system.
10. **EntryPoint Security & Correctness:** Assumes ERC-4337 EntryPoint behaves as specified for validation, execution, nonce, and deposit logic.
11. **Compiler Security:** Relies on the Solidity compiler version `^0.8.29` being free from critical bugs.
12. **Off-Chain Component Security:** Assumes bundlers, relayers, and UIs behave correctly, especially in signature generation and transaction submission.
13. **ERC7201 Namespace Uniqueness:** Assumes the `startale.account.storage` namespace is unique to prevent storage collision.
14. **SentinelList Library Correctness:** Assumes SentinelListLib handles list operations and pointer logic correctly in all edge cases.
15. **Gas Griefing via Signatures:** Validators must enforce signature length checks to avoid calldata gas griefing.
16. **Gas Griefing via Initialization Data:** Large initData arrays can cause out-of-gas errors; UIs/factories should limit input sizes.
17. **Module onInstall/onUninstall Gas Limits:** Complex install/uninstall logic can risk exceeding gas limits.
18. **Hook Gas Consumption:** Inefficient `preCheck` / `postCheck` hooks can increase gas cost of all core actions.
19. **Nonce Management Complexity:** Relies on correct encoding/decoding of ERC-4337-style nonces for replay protection and validator compatibility.
20. **deInitData Requirements:** Uninstalling modules needs accurate `deInitData`, including correct previous address for list-based modules.
21. **Emergency Uninstall Timelock Period:** Uninstall waits for `_EMERGENCY_TIMELOCK`; period must balance urgency and owner recovery time.
22. **Event Monitoring:** Off-chain systems must monitor events like `ModuleInstalled` or `EmergencyHookUninstallRequest` for operational security.
23. **ERC7702 Redelegation State:** `_onRedelegation` clears module state but not `accountStorageBases`, risking duplicate entries and state conflicts.

Key Actors And Their Capabilities

1. Account Owner (Defined via Validator Module)
    - Definition: The EOA or contract whose signature (or other validation method) is accepted by an installed Validator module (e.g., the owner stored in `ECDSAValidator.smartAccountOwners`). This is the primary controller of the account.
    - Capabilities (Indirect - via signed UserOps/messages):

        - Initiate arbitrary transactions via `execute` or `executeFromExecutor`.
        - Install any module type using `installModule`.
        - Uninstall modules using `uninstallModule` (subject to hook checks).
        - Trigger `emergencyUninstallHook` with a valid EIP-712 signature.
        - Upgrade the implementation using `upgradeToAndCall` (not available for ERC7702).
        - Manage deposit via `addDeposit`, `withdrawDepositTo`.
        - Call internal module functions like `ECDSAValidator.transferOwnership` or `addSafeSender`.
        - Trigger redelegation preparation via `onRedelegation`.
2. Installed Executor Module
    - Definition: A contract installed via `installModule` with `MODULE_TYPE_EXECUTOR`, present in the executors list.
    - Capabilities (Direct):
        - Call `executeFromExecutor` to perform transactions on behalf of the account, subject to hook checks.
3. Installed Hook Module
    - Definition: A contract installed with `MODULE_TYPE_HOOK` stored in `accountStorage.hook`.
    - Capabilities (Direct - via hooked operations):
        - Execute logic before (`preCheck`) and after (`postCheck`) core actions.
        - Block actions by reverting in `preCheck` or `postCheck`.
        - Access `msg.sender`, `msg.value`, and `msg.data` in `preCheck`.
        - Maintain its own state and modify account state (use with caution).
4. Installed PreValidationHook Module (ERC1271 or ERC4337)
    - Definition: A contract installed with `MODULE_TYPE_PREVALIDATION_HOOK_ERC1271` or `_ERC4337`, stored in corresponding pre-validation hook slots.
    - Capabilities (Direct - during validation):
        - Execute logic during `preValidationHookERC1271` or `preValidationHookERC4337`.
        - Modify hash/signature passed to the main Validator module.
5. Installed Fallback Module
    - Definition: A contract installed with `MODULE_TYPE_FALLBACK`, registered for specific selectors in `accountStorage.fallbacks`.
    - Capabilities (Direct - on matching selector):
        - Handle matching calls using `call` (`CALLTYPE_SINGLE`) or `staticcall` (`CALLTYPE_STATIC`).
6. EntryPoint Contract (`_ENTRYPOINT`)
    - Definition: The trusted ERC-4337 EntryPoint contract set in `BaseAccount.sol`.
    - Capabilities (Direct - as `msg.sender` to Account):
        - Call `validateUserOp`, `execute`, `executeUserOp`.
        - Initiate `installModule`, `uninstallModule`.
        - Call `upgradeToAndCall` via `_authorizeUpgrade`.
        - Call `withdrawDepositTo`.
    - Capabilities (External to Account):
        - Manage deposits, nonces, and UserOperation lifecycle.
7. Factory Owner (`Stakeable.owner`)
    - Definition: The owner address set in factory constructors (`StartaleAccountFactory`, `EOAOnboardingFactory`).
    - Capabilities (Direct - on factory contract):
        - Manage factory stake via `addStake`, `unlockStake`, `withdrawStake`.

Findings

STAA-1

Incorrect Context Sent to Hook when Installing Module via Enable Mode

• Medium ⓘ Fixed

Update

The team fixed the issue as recommended in commit `d47a2a3e`, and some tests were also added.

File(s) affected: `src/core/ModuleManager.sol`, `src/StartaleSmartAccount.sol`

Description: When a module is installed via the "Enable Mode" flow (initiated within `validateUserOp()`), the internal `_installModule()` function is ultimately called. Although the specific internal installation functions (`_installValidator()`, `_installExecutor()`, etc.) correctly apply the `withHook` modifier, the context (`msg.sender`, `msg.data`, `msg.value`) passed to the hook's `preCheck()` and `postCheck()` methods corresponds to the original `validateUserOp()` call (where `msg.sender` is the EntryPoint). Hooks specifically designed to intercept and apply policies based on the context of an `installModule()` call (expecting `msg.sender` to be the account/EntryPoint and `msg.data` to contain `installModule()` arguments) will receive incorrect context and may fail to enforce intended security policies or logic when a module is installed via Enable Mode.

Note that the referenced Biconomy Nexus repository also had the issue, and the fix was included in this PR: link.

Recommendation: Modify the `_enableMode()` function in `ModuleManager.sol` to perform a self-call to the *external* `installModule()` function (e.g., `address(this).installModule{value: msg.value}(moduleType, module, moduleInitData)`) instead of directly calling the internal `_installModule()`. This ensures that the `withHook` modifier is triggered within the context of a standard `installModule()` call, providing the hook with the expected `msg.sender` and `msg.data`.

STAA-2

Potential Gas Exhaustion Vulnerability in Redelegation Cleanup

• Medium ⓘ Mitigated

Update

With commits `3bf8109` and `262e02f8`, the team made a design change where, during `_onRedelgation()`, only the account storage is cleared, and `onUninstall()` is not called for the modules. This eliminates the main concern of this issue.

The concern regarding the executor remains; however, generally speaking, more trust is required for the executor module.

The following is the original statement from the team:

```
_onRedelegation() is mitigated by design -> by not making onUninstall calls. for the onInstall process
I could cap it with large enough number but without configuring storage in the account if possible.
later this finding should be modified and marked ack / mitigated accordingly.
```

File(s) affected: `src/core/ModuleManager.sol`

Description: The `_tryUninstallValidators()`, `_tryUninstallExecutors()`, `_tryUninstallHook()`, and `_tryUninstallPreValidationHook()` functions are called during `_onRedelegation()` to clean up installed modules. They use `try/catch` or `excessivelySafeCall()` to invoke the `onUninstall()` function of each module, passing `gasleft()`. While this prevents a single failing module from reverting and halting the entire cleanup process, it allows a malicious module to perform a gas griefing attack. A module's `onUninstall()` function can contain an infinite loop or computationally expensive operations that consume all remaining gas (`gasleft()`) before reverting or completing. If multiple modules behave this way, the gas cost of `_onRedelegation()` can become prohibitively high, potentially exceeding block limits and causing a Denial of Service for the redelegation feature.

On a related but distinct note, during the `onInstall()` process for the executor, the executor can call back to `executeFromExecutor()`, which uses the account's gas instead of the executor's. While this is not a DoS factor like the uninstall paths, it can cause the account to pay for unexpected gas costs.

Recommendation: When calling potentially untrusted external code like `onUninstall()` during the bulk cleanup in `_onRedelegation()`, provide a specific amount of gas (a reasonable stipend, e.g., 50,000–100,000 gas) to each call instead of using `gasleft()`. This stipend can be either fixed or configurable via a state variable. Doing so prevents a single malicious module from consuming a disproportionate amount of gas and ensures that the cleanup process can iterate through all modules, even if some misbehave.

Similarly, consider applying a gas limit to the `onInstall()` process.

STAA-3
Potential Replay Attack Vector in `ECDSAValidator.validateSignatureWithData()`

• Low ⓘ Acknowledged

> **Update**
> The team clarified that this is the intended behavior of the function.

File(s) affected: `src/modules/validators/ECDSAValidator.sol`

Description: The `ECDSAValidator.validateSignatureWithData()` function allows validating signatures against a provided `_hash` and owner `_data`. However, it may be susceptible to replay attacks depending on how the caller constructs the `_hash`, since there is no enforcement that the hash follows EIP-712 standards including a `nonce` and chain ID to protect against replay attacks.

Recommendation: Please confirm whether this is the intended behavior. If so, add warning documentation for any contracts that leverage this function. Otherwise, consider wrapping the `_hash` with EIP-712 and verifying the signature accordingly.

STAA-4 `AssociatedArrayLib` Can Risk Not Complying with EIP-7562

• Low ⓘ Fixed

> **Update**
> The team fixed the issue as recommended in commit `83269baf`.

File(s) affected: `src/lib/AssociatedArrayLib.sol`

Description: The current implementation of the AssociatedArrayLib library does not comply with the associated storage limitation specified in EIP 7562. According to the EIP, the array length can be at most 127 due to the calculation of the slot value as `keccak(A||x)+n`, where n is a value in the range 0..128. However, the library does not enforce this limit, potentially leading to storage inefficiencies and exceeding the associated storage constraints.

Recommendation: Update the library to enforce the array length limit of 127 as specified in EIP 7562 to ensure compliance with associated storage constraints and prevent potential storage inefficiencies. This can be achieved by adding appropriate checks and validations in the library functions to restrict the array length to the maximum allowed value of 127.

STAA-5 Reliance on Undeployed Contracts on Soneium Network

• Low ⓘ Fixed

> **Update**
> The missing contracts have now been deployed, as shown in the block explorer sites below:
>
> > Deployed on Soneium mainnet
> https://soneium.blockscout.com/address/0×000000000000D9ECebf3C23529de49815Dac1c4c?tab=contract
>
> > Deployed on Soneium mainnet
> https://soneium.blockscout.com/address/0×000000000000378eDCD5B5B0A24f5342d8C10485?tab=contract

File(s) affected: `src/modules/validators/ECDSAValidator.sol`, `lib/erc7739Validator/src/ERC7739Validator.sol`

Description: The `ECDSAValidator` contract and its dependency `ERC7739Validator` rely on specific, hardcoded contract addresses for parts of their validation logic. These contracts are assumed to be deployed, but they do not exist on the target Soneium network.

1. `ECDSAValidator._erc1271CallerIsSafe()`: This function checks if the `_sender` is the canonical `MulticallerWithSigner` contract at `0x000000000000D9ECebf3C23529de49815Dac1c4c`. This address is considered "safe" because it's known to include the account address in the signed hash. However, this contract is not deployed on Soneium (Link). Consequently, calls originating from this address on Soneium (if it were deployed) would not be recognized as safe by this check, potentially forcing the more complex and gas-intensive nested EIP-712 validation path unnecessarily.
2. `ERC7739Validator._erc1271IsValidSignatureViaRPC()`: This function, inherited by `ECDSAValidator`, attempts to differentiate between on-chain and off-chain (RPC) calls using a heuristic involving `tx.gasprice` and a `staticcall` to a Basefee contract expected at `0x000000000000378eDCD5B5B0A24f5342d8C10485`. This Basefee contract is also not deployed on Soneium (Link). The `staticcall` to a non-existent contract will fail. While the code attempts to handle this, the reliability of the on-chain vs. off-chain detection mechanism is compromised on Soneium, potentially leading to incorrect validation outcomes for signatures intended only for RPC validation.

The absence of these critical contracts on the Soneium network means that specific validation paths designed for efficiency (`_erc1271CallerIsSafe`) or security (`_erc1271IsValidSignatureViaRPC`) will not function as intended. This could lead to increased gas costs and potentially incorrect signature validation results under specific circumstances on this network.

Recommendation: Confirm the deployment status of the `MulticallerWithSigner` (`0x00...1c4c`) and the `Basefee` contract (`0x00...0485`) on the Soneium network. If they are intended to be used, ensure they are deployed correctly.

STAA-6 Potential Memory Corruption in `withdrawDepositTo()` Assembly

• Low Fixed

Update

The team fixed the issue by using a free pointer instead of the scratched memory space in commit `2993a5bc` .

File(s) affected: `src/core/BaseAccount.sol`

Description: The assembly block within the `BaseAccount.withdrawDepositTo()` function potentially corrupts the free memory pointer (slot `0x40`) on the line `mstore(0x34, 0) // Restore the part of the free memory pointer that was overwritten` .

The code uses memory locations `0x00` - `0x54` to prepare calldata for an external call to `entryPoint.withdrawTo()` . Specifically:

- `mstore(0x34, amount)` writes the `amount` to memory, overwriting the range `0x34` - `0x54` , which includes the free memory pointer slot `0x40` .
- After the external call, `mstore(0x34, 0)` is used to clear the memory used by `amount` . However, this also writes zeros to the memory range `0x34` - `0x54` , effectively zeroing out the upper bytes of the free memory pointer slot `0x40` .

While the original free memory pointer value is saved in `freeMemPtr` , it is never restored. The current cleanup logic (`mstore(0x34, 0)`) incorrectly assumes the upper bytes of the free memory pointer are zero or that zeroing them out is safe. This can lead to memory corruption if the free memory pointer has higher-order bytes set, potentially causing issues with subsequent memory allocations within the same transaction context.

Recommendation: Replace the line `mstore(0x34, 0)` with `mstore(0x40, freeMemPtr)` to explicitly restore the free memory pointer to its original value after the external call, ensuring memory integrity.

STAA-7 Potential Unintended Return Data in `_fallback()` Function

• Low Fixed

Update

The team fixed the issue as recommended in commit `2f8d809b` .

File(s) affected: `src/core/ModuleManager.sol`

Description: The `_fallback()` function handles unrecognized selectors. For specific ERC721/ERC1155 `onReceived()` selectors (`0x150b7a02` , `0xf23a6e61` , `0xbc197c81`), it's designed to return the selector itself as a magic value, indicating acceptance. The assembly code achieves this by storing the 4-byte selector (`s`) at memory address `0x20` and then returning 32 bytes starting from address `0x3c` (`return(0x3c, 0x20)`). This return slice (`memory[60:92]`) correctly includes the selector (`memory[60:64]`), but the remaining 28 bytes (`memory[64:92]`) are not explicitly cleared. These bytes might contain remnants of previous operations or, commonly, parts of the free memory pointer (stored at `0x40`). While standard token contracts typically only check the first 4 bytes of the return value, returning potentially "dirty" memory is unconventional and could theoretically cause issues with non-standard or future contract interactions that inspect the full 32 bytes.

Recommendation: Ensure a clean 32-byte return value containing only the required selector. Modify the assembly block to allocate a clean 32-byte segment, store the selector shifted to the most significant bytes (as expected for `bytes4` cast from `bytes32`), and return that segment.

```
// src/core/ModuleManager.sol - inside _fallback() assembly block
    // 0xbc197c81: `onERC1155BatchReceived(address,address,uint256[],uint256[],bytes)`.
  if or(eq(s, 0x150b7a02), or(eq(s, 0xf23a6e61), eq(s, 0xbc197c81))) {
    // Store msg.sig left-aligned in scratch space memory[0:32]
    mstore(0x00, shl(224, s))
    // Return the clean 32-byte word from memory[0:32]
    return(0x00, 0x20)
  }
```

STAA-8 Missing Boundary Checks in `DataParserLib` Library

• Low Fixed

Update

The team fixed the issue as recommended in commit `14e85ad5` .

File(s) affected: `src/lib/DataParserLib.sol`

Description: The assembly code in `parseEnableModeData()` and `parseMultiTypeInitData()` calculates offsets (`*.offset`) and lengths (`*.length`) for various data segments within the input calldata. However, it fails to validate that these segments are actually contained within the bounds of the provided calldata. Specifically, it does not check if `segment.offset + segment.length <= calldatasize()`. Reading beyond the calldata boundary using `calldataload()` returns zeros. If the calling code relies on the parsed data without further validation, this could lead to incorrect behavior, such as processing an empty signature or module data when the calldata was truncated or malformed.

Recommendation: Implement explicit boundary checks within the assembly blocks after calculating the offset and length for each data segment. This ensures that any attempt to parse data segments that extend beyond the actual calldata results in a revert, preventing the processing of potentially invalid or truncated data.

- `parseEnableModeData()`:

```
assembly ("memory-safe") {
  let dataSize := calldatasize() // Get total calldata size
  p := packedData.offset
  // ... other checks and assignments for module, moduleType ...

  // Check if reading moduleInitData length pointer (32 bytes) is within bounds
  if gt(add(add(p, 0x20), 0x20), dataSize) { revert(0, 0) }
  moduleInitData.length := shr(224, calldataload(add(p, 0x20)))
  moduleInitData.offset := add(p, 0x24)

  // Boundary Check: Ensure the calculated moduleInitData segment (offset + length)
  // does not exceed the actual calldata size. Revert if it does.
  if gt(add(moduleInitData.offset, moduleInitData.length), dataSize) { revert(0, 0) }

  p := add(moduleInitData.offset, moduleInitData.length)

  // ... other checks and assignments for enableModeSignature ...
}
// ... check for userOpSignature slice start ...
userOpSignature = packedData[p:];
```

- `parseMultiTypeInitData()`:

```
assembly ("memory-safe") {
  let dataSize := calldatasize() // Get total calldata size
  let offset := initData.offset
  let baseOffset := offset

  // ... check for reading types array offset pointer ...
  let dataPointer := add(baseOffset, calldataload(offset))

  // ... check for reading types array length ...
  types.offset := add(dataPointer, 32)
  types.length := calldataload(dataPointer)

  // Boundary Check: Ensure the calculated types array data segment
  // (offset + length * 32 bytes per element) does not exceed the actual calldata size.
  // Revert if it does.
  if gt(add(types.offset, mul(types.length, 32)), dataSize) { revert(0, 0) }

  offset := add(offset, 32)

  // ... other checks and assignments for initDatas ...
}
```

STAA-9 Gas Griefing Risk via Calldata Stuffing

Low · Acknowledged

> **Update**
> The team add a comment in the test file in commit `175136d2` and acknowledged that they will rely on the handler to validate the data.

File(s) affected: `src/core/ModuleManager.sol`, `src/lib/ExecutionLib.sol`

Description: Several functions pass raw calldata (`msg.data` or module `initData`) to external calls or libraries without validating its length against expectations. Specifically:

- The `withHook` modifier passes `msg.data` to the hook's `preCheck()`.

- The `fallback()` function passes `msg.data` (via `ExecutionLib.get2771CallData()`) to the fallback handler. `get2771CallData()` copies the *entire* `msg.data` to memory.
- The `_installModule()` function passes `initData` to the module's `onInstall()`. If the receiving contract (hook, fallback handler, module) does not validate the length or ignores extra data, an attacker (e.g., a bundler) can append arbitrary bytes to the calldata. This increases the transaction's intrinsic gas cost due to the extra calldata bytes, which is paid by the user or paymaster, without affecting the function's logic. This allows for gas griefing by inflating transaction costs unnecessarily. The use of `get2771CallData()` in the fallback path is particularly notable as it involves memory allocation and copying proportional to the *entire* `msg.data` length.

Recommendation: Mitigate calldata stuffing risks:

1. **Hooks/Fallbacks/Modules:** Implementations of hooks (`preCheck()`), fallback handlers, and modules (`onInstall()`) should be designed to decode only necessary parameters and ignore or revert on excessive trailing data. Document this as a requirement for module developers.
2. **Fallback Path:** Consider modifying `ExecutionLib.get2771CallData()` or the fallback logic to avoid copying the entire `msg.data` if only a prefix is needed by the handler, although this might break compatibility if handlers expect the full data. Alternatively, rely on handlers themselves to be efficient.
3. **Bundler Policies:** Acknowledge reliance on bundlers implementing policies against transactions with excessive calldata padding.

STAA-10
Signature Bypass Vulnerability in `ECDSAValidator.validateSignatureWithData()`

• Low ⓘ Fixed

> **Update**
> The team fixed the issue as recommended in commit `e0b6d6ef`.

File(s) affected: `src/modules/validators/ECDSAValidator.sol`

Description: The `ECDSAValidator` uses an internal `_recoverSigner()` function which relies on Solady's `tryRecoverCalldata()`. This function returns `address(0)` upon signature recovery failure instead of reverting. The `_validateSignatureForOwner()` function compares this recovered address against an expected `_owner`. If the expected `_owner` is `address(0)`, a failed recovery (returning `address(0)`) will incorrectly pass the validation check (`address(0) == address(0)`). While `onInstall()` and `transferOwnership()` prevent setting the stored owner to `address(0)`, the external function `validateSignatureWithData()` accepts the owner address via its `_data` parameter. Calling this function with `_data` specifying `address(0)` as the owner allows any invalid signature `_sig` to pass validation, returning `true`.

Exploit Scenario:
An external system integrates with `ECDSAValidator` using `validateSignatureWithData()` for authorization checks. An attacker calls this system, providing crafted `_data` that decodes to `owner = address(0)` and any invalid signature `_sig`. `validateSignatureWithData()` returns `true`, potentially tricking the external system into granting the attacker unauthorized access or permissions.

Recommendation: Modify the internal `_recoverSigner()` function in `ECDSAValidator.sol` to use Solady's `recoverCalldata()` instead of `tryRecoverCalldata()`. `recoverCalldata()` reverts if the signature is invalid, ensuring that the validation process fails correctly for invalid signatures and preventing the `address(0)` comparison bypass, regardless of the owner address provided via `_data`.

```
// src/modules/validators/ECDSAValidator.sol
function _recoverSigner(bytes32 _hash, bytes calldata _signature) internal view returns (address) {
    // Use recoverCalldata which reverts on invalid signature, preventing address(0) bypass.
    return _hash.recoverCalldata(_signature);
}
```

STAA-11
Insufficient Protection Against Fallback Handler Overriding Core Functions

• Low ⓘ Mitigated

> **Update**
> The team expanded the ban list in commit `24603492` to include the native selectors of ERC-7579, ERC-721, ERC-1155, ERC-1271, and ERC-4337.
>
> Note that the native selectors for the Startale account have not been added to the ban list yet.

File(s) affected: `src/core/ModuleManager.sol`

Description: The `_installFallbackHandler()` function prevents installing handlers for `onInstall(bytes)`, `onUninstall(bytes)`, and `bytes4(0)`. However, it does not block fallback handlers for other critical functions integral to the account's operation and security—such as `execute()`, `executeFromExecutor()`, `validateUserOp()`, `installModule()`, `uninstallModule()`, `isValidSignature()`,

and `upgradeToAndCall()`. Allowing fallback handlers for these selectors enables bypassing core logic, security mechanisms (e.g., hooks, module checks, entry point validation), and access controls, potentially leading to unauthorized actions or fund theft.

If fallback handlers remain allowed for current native function selectors, future upgrades that remove some of these functions could inadvertently trigger the fallback handler when those functions are called, potentially leading to unexpected outcomes.

Recommendation: Expand the list of forbidden selectors in `_installFallbackHandler()` to include all critical functions of the `StartaleSmartAccount` and its inherited contracts, such as:

- All ERC-7579, ERC-721, ERC-1155, ERC-1271 and ERC-4337 native selectors
- All startale account native selectors

Maintain a comprehensive, hardcoded list or mapping of protected function selectors that cannot be overridden by fallback handlers to prevent hijacking of core functionality.

STAA-12
Fallback Handlers Persist After Redelegation Leading to Potential Risks

Low · Fixed

Update
The team addressed the issue in commits `588dc649`, `ec36e93d`, `a7136120f`, `3bf8109a`, and `262e02f8`.

The fallback handler-related storage is cleared during the `_onRedelegation()` function, but `onUninstall()` calls no longer exist. As a consequence, a new change has been introduced: during the `onInstall()` process, the account will call the modules with `onInstall()` only when `data` is provided (i.e., non-empty). This allows the modules to be reinstalled into the account without requiring a separate module call after re-delegation.

File(s) affected: `src/StartaleSmartAccount.sol`, `src/core/ModuleManager.sol`, `src/interfaces/core/IAllStorage.sol`

Description: The `_onRedelegation()` function in `StartaleSmartAccount` is intended to prepare the account for a new implementation by cleaning up modules associated with the previous implementation. While it correctly attempts to uninstall validators, executors, hooks, and pre-validation hooks (which are stored in iterable `SentinelLists` or single storage slots), it does not clear the `fallbacks` mapping located in `AccountStorage` (defined in `src/interfaces/core/IAllStorage.sol` and accessed via `ModuleManager`).

This occurs because Solidity mappings (`mapping(bytes4 => FallbackHandler)`) are not iterable, making it impossible for `_onRedelegation()` to automatically discover and call `onUninstall()` for all registered fallback handlers.

Consequently, when an account is redelegated (upgraded to a new implementation via ERC-7779), the old fallback handlers remain registered in storage. The new implementation might install its own handlers, potentially overwriting some selectors, but any selectors handled only by the old implementation will still point to the old fallback handler modules. This persistence can lead to:

1. Unexpected Behavior: Calls to functions handled by these orphaned fallbacks might execute outdated logic, interact incorrectly with the new state, or revert unexpectedly.
2. Security Risks: If an old fallback handler relied on specific state variables, modules, or assumptions that are changed or invalidated by the redelegation, its continued execution could become insecure.
3. Orphaned Logic / Gas Waste: The account retains potentially irrelevant logic, and calls might route through these old handlers unnecessarily.

Recommendation: Modify the storage structure for fallback handlers to enable iteration. This could involve:

- Maintaining a separate iterable list (e.g., `SentinelListLib.SentinelList` or `AssociatedArrayLib.Bytes32Array`) of registered `bytes4` selectors alongside the mapping.
- Updating `_installFallbackHandler()` and `_uninstallFallbackHandler()` to manage this list.
- Implementing the logic within `_tryUninstallFallbacks()` to iterate through the list, retrieve the handler from the mapping, call `uninstallModule()` for each, and clear the list/mapping. This would fully automate the cleanup during redelegation.

STAA-13
Potential Duplicate Entries in ERC7702 `accountStorageBases` Array on Redelegation

Informational · Fixed

Update
The team fixed the issue as recommended in commit `bc8cfa05`.

File(s) affected: `src/StartaleSmartAccount.sol`, `src/core/ERC7779Adapter.sol`

Description: The `ERC7779Adapter.sol` contract includes functionality related to ERC7702 account redelegation.

During the initialization phase of a `StartaleSmartAccount` (specifically within `_initializeAccount`), if the account is identified as an ERC7702 account (i.e., `_amIERC7702()` returns `true`), the `_addStorageBase(ACCOUNT_STORAGE_LOCATION)` function is called. This appends the account's storage base (`ACCOUNT_STORAGE_LOCATION`) to the `storageBases` array located at the `ERC7779_STORAGE_BASE` slot.

The `onRedelegation` function (invoked via `_onRedelegation` in `StartaleSmartAccount.sol`) is designed to prepare an account for redelegation to a different implementation. It clears various module states (validators, executors, hooks), but does not clear or modify the `storageBases` array.

If an EOA is redelegated multiple times to the same `StartaleSmartAccount` implementation, the initialization logic (including `_addStorageBase`) may be re-executed upon the first user operation after redelegation — potentially appending the same `ACCOUNT_STORAGE_LOCATION` multiple times to `storageBases`.

Here is the potential impact:

1. **Off-chain Tooling:** External tools or wallets querying `accountStorageBases()` may receive arrays with duplicate entries, leading to incorrect assumptions or processing errors about the account's structure or history.
2. **Gas Inefficiency:** Although minor, redundant entries consume unnecessary gas and incrementally bloat storage over time.

Recommendation: Modify `_addStorageBase()` in `ERC7779Adapter.sol` to check for existing entries before appending, which may require looping. Alternatively, document the trade-off if we are okay with some duplication after redelegation and assume that users will not redelegate often.

STAA-14
Incorrect Error Handling in Module Enable Mode Signature Validation Deviates From ERC-4337 Specification

• Informational ⓘ Acknowledged

Update

We had a discussion with the team, and based on the ERC-4337 specification, this falls into a gray area. The original intention of ERC-4337 is to support the standard account consent validation flow, whereas this represents an unexpected flow not covered by the specification. The team decided to keep the current implementation, and we agreed that this should be acceptable.

File(s) affected: `src/core/ModuleManager.sol`, `src/StartaleSmartAccount.sol`

Description: The handling of signature validation failures within the Module Enable Mode flow deviates from the error handling principles outlined in the ERC-4337 specification for the `validateUserOp()` function. This occurs in two related ways:

1. Revert on Signature Failure:
   The `_enableMode()` function in `ModuleManager` calls `_checkEnableModeSignature()` to verify the signature authorizing the module installation. If `_checkEnableModeSignature()` returns `false` (indicating the signature is invalid), `_enableMode()` reverts with `EnableModeSigError()`.
   However, Module Enable Mode is part of the overall `validateUserOp()` execution flow. The ERC-4337 specification requires that `validateUserOp()` should return `SIG_VALIDATION_FAILED (1)` packed in `validationData` for signature mismatches, rather than reverting, to allow for proper gas estimation. Reverting inside `_enableMode()` prevents `validateUserOp()` from returning the appropriate failure code.
2. Masking Validator Reverts:
   The `_checkEnableModeSignature()` function wraps the external call to the validator's `isValidSignatureWithSender()` method within a try-catch block. If the validator call reverts for any reason (e.g., out-of-gas, invalid state access, assertion failure, malformed signature data), the catch block causes `_checkEnableModeSignature()` to return `false`.
   This incorrectly treats potentially critical validator errors as simple signature failures. ERC-4337 mandates that errors other than signature mismatches must cause `validateUserOp()` to revert. Masking these reverts prevents proper error propagation and debugging.

Both signature mismatches and unexpected validator errors during the Module Enable Mode signature check lead to a revert in `_enableMode()`, contradicting the ERC-4337 requirement to return `SIG_VALIDATION_FAILED (1)` for signature failures and to propagate reverts for validator errors.

Recommendation:

1. Remove the try-catch in `_checkEnableModeSignature()`: Modify `_checkEnableModeSignature()` in `ModuleManager` to remove the try-catch block around the call to `IValidator(validator).isValidSignatureWithSender()`. Alternatively, if off-chain components rely on specific error types to detect validator failures, wrap and rethrow the caught error using a custom revert reason.
2. Handle signature failures in `validateUserOp()` during the `_enableMode()` path: Update the `validateUserOp()` function in `StartaleSmartAccount` so that if `_enableMode()` returns a signature mismatch (rather than reverting), it instead returns `SIG_VALIDATION_FAILED (1)` packed in `validationData`.

STAA-15 Missing ERC-165 Support

• Informational ⓘ Fixed

Update

The team fixed the issue in commits `69a0663` and `7f7cebdd`.

File(s) affected: `src/StartaleSmartAccount.sol`

Description: The smart account implementation includes support for receiving ERC-721 and ERC-1155 tokens in the fallback function, as evidenced by the code in `ModuleManager._fallback()` which handles `onERC721Received()`, `onERC1155Received()`, and `onERC1155BatchReceived()` function selectors. However, the contract does not implement the required `supportsInterface()` function as specified by ERC-165, which is a prerequisite for both ERC-721 and ERC-1155 standards.

In the current implementation, when a token contract attempts to verify if the recipient supports the required interfaces via `supportsInterface(bytes4)`, the call will be routed to the fallback function. Without a proper fallback handler for this selector, the transaction will revert with a `MissingFallbackHandler` error, potentially causing token transfers to fail.

This issue affects the ability of the smart account to reliably receive ERC-721 and ERC-1155 tokens, as compliant token implementations will check for interface support before transferring tokens to the account.

Recommendation: Consider implementing the function in one of the following ways:

1. Native Implementation: Add a native `supportsInterface` function to the `StartaleSmartAccount` contract that explicitly declares support for ERC-721 and ERC-1155 receiver interfaces:
2. Dynamic Interface Registry: Implement a registry system, where supported interfaces can be tracked when fallback handlers are added. This would allow the account to dynamically register support for new interfaces when fallback handlers are installed, without requiring contract upgrades.
3. Fallback Handler: Include a fallback handler for the `supportsInterface` selector that returns `true` for the ERC-721 and ERC-1155 receiver, or other interfaces during account initialization.

Auditor Suggestions

S1 General Code Improvement and Cleanup

Fixed

Update

The team fixed most of the suggestions in commit `50a921d2`.

The client provided the following explanation:

```
fixed except "Remove the unused cd parameter and add a comment clarifying that the function processes
msg.data."
```

File(s) affected: `src/core/ExecutionHelper.sol`, `src/modules/validators/ECDSAValidator.sol`, `src/lib/ExecutionLib.sol`, `src/StartaleSmartAccount.sol`, `src/types/Constants.sol`, `src/core/BaseAccount.sol`, `src/lib/EnumerableMap4337.sol`, `src/factory/EOAOnboardingFactory.sol`, `src/lib/BootstrapLib.sol`, `src/lib/ModuleTypeLib.sol`, `src/core/ModuleManager.sol`

Description: Several opportunities exist for code cleanup, readability improvements, and adherence to best practices across the codebase. Applying these suggestions can enhance maintainability and reduce potential confusion.

1. `ExecutionHelper._tryExecuteBatch()`: Emit target address along with calldata and result in the `TryExecuteUnsuccessful` event for better debugging of batch failures.
2. `ECDSAValidator.validateSignatureWithData()`: Add NatSpec comments clarifying the intended use case (stateless validation against a provided owner).
3. `ExecutionLib.sol`: Remove the unused `ERC7579DecodingError` custom error definition.
4. `ExecutionLib.get2771CallData()`: Remove the unused `cd` parameter and add a comment clarifying that the function processes `msg.data`.
5. `StartaleSmartAccount.validateUserOp()`: Refactor the logic to remove the empty `if (op.nonce.isValidateMode())` block.
6. `StartaleSmartAccount.checkERC7739Support()`: Improve readability of the line `(uint256(hash) == (~signature.length / 0xffff) * 0x7739)` by stating it is checking the `hash` equals to `0x7739773977397739773977397739773977397739773977397739773977397739`.
7. `StartaleSmartAccount.executeUserOp()`: Remove the unused `innerCallRet` variable declaration.
8. `Constants.sol`: Add a comment explaining the origin and purpose of the `ACCOUNT_STORAGE_LOCATION` constant.
9. `BaseAccount.payPrefund()`: Refactor the assembly for better readability. Use `pop(call(gas(), caller(), missingAccountFunds, 0, 0, 0, 0)` instead of `pop(call(gas(), caller(), missingAccountFunds, codesize(), 0x00, codesize(), 0x00))` as there is no benefit of using `codesize()` here.
10. `BaseAccount.withdrawDepositTo()`: Correct the comment regarding `0x205c2878000000000000000000000000` to accurately state it's the function selector `withdrawTo(address,uint256)` left-shifted 128 bits, not just padding with zeros until 32 bytes. The left-shifting is the reason why the `call()` can use `0x10`, which then will skip the 128 bits and start from `0x205c2878`.
11. `EnumerableMap4337.sol`: Remove the entire file as the `EnumerableMap` library appears unused in the project.
12. `EOAOnboardingFactory.sol`: Remove the unused import statement for `BootstrapLib`.
13. `BootstrapLib.sol`: Move this file to the `test/` directory as it appears to be used only for testing.
14. `ModuleTypeLib.sol`: Move this file to the `test/` directory as it appears to be used only for testing.

Recommendation: Apply the specific code changes and organizational adjustments listed above to improve code quality, readability, maintainability, and reduce deployment size by removing unused components.

S2 Application Monitoring Can Be Improved by Emitting More Events

Fixed

Update

The team addressed most of the suggestions in commit `5aaf7e62`, except for point 2 (`ModuleManager._tryUninstallXXX()` functions).

File(s) affected: `src/core/ModuleManager.sol`, `src/StartaleSmartAccount.sol`, `src/modules/validators/ECDSAValidator.sol`

Description: Several state-changing operations and external calls across the contracts lack corresponding event emissions, hindering off-chain tracking, monitoring, and debugging:

1. `ModuleManager._enableMode()`: When a module is installed using the "Enable Mode" flow within `validateUserOp()`, the `ModuleInstalled` event is not emitted because the internal installation functions are called directly, bypassing the public `installModule()` which emits the event.
2. `ModuleManager._tryUninstallXXX()` functions (during `_onRedelegation()`): The `_onRedelegation()` function calls internal helper functions (`_tryUninstallValidators()`, `_tryUninstallExecutors()`, etc.) to remove modules. These helpers call `onUninstall()` on the modules but do not emit the standard `ModuleUninstalled` event for each successful removal.
3. `ECDSAValidator.transferOwnership()`: This function modifies the `smartAccountOwners` mapping but does not emit an event reflecting the ownership change. Only `onInstall()` emits `OwnerRegistered`.
4. `ECDSAValidator.onUninstall()`: This function deletes the owner entry from `smartAccountOwners` but does not emit an event indicating the removal.
5. `ModuleManager._uninstallValidator()`, `_uninstallExecutor()`, `_uninstallHook()`, `_uninstallFallbackHandler()`: These functions use `excessivelySafeCall()` to invoke the module's `onUninstall()` method but do not check the return status or emit an event if the external call fails. This can mask issues during uninstallation.

This inconsistent event emission makes it difficult for off-chain services, UIs, and monitoring tools to reliably track the account's module configuration, ownership status, and potential errors during module interactions.

Recommendation: Ensure consistent event emission for all significant state changes and external call outcomes:

1. `ModuleManager._enableMode()`: Ensure `ModuleInstalled` is emitted. Either call the external `installModule()` from `_enableMode()` or explicitly emit `ModuleInstalled(moduleType, module)` in `_enableMode()` after the internal `_installModule()` call succeeds.
2. `ModuleManager._tryUninstallXXX()` functions: Modify these functions used during redelegation cleanup. Inside the logic where each module's `onUninstall()` is successfully called within the try block, emit the corresponding `ModuleUninstalled(moduleTypeId, moduleAddress)` event. This requires passing the `moduleTypeId` to `_tryUninstallPreValidationHook()`.
3. `ECDSAValidator.transferOwnership()`: After updating `smartAccountOwners[msg.sender]`, emit `OwnerRegistered(msg.sender, _newOwner);`.
4. `ECDSAValidator.onUninstall()`: Define a new event `event OwnerRemoved(address indexed account);` and emit it after `delete smartAccountOwners[msg.sender];` using `emit OwnerRemoved(msg.sender);`.
5. `ModuleManager._uninstall*()` functions: Define a new event like `event ExternalCallFailed(address indexed target, bytes callData, bytes returnData);`. In `_uninstallValidator()`, `_uninstallExecutor()`, `_uninstallHook()`, and `_uninstallFallbackHandler()`, check the boolean `success` value returned by `excessivelySafeCall()`. If it's `false`, emit the `ExternalCallFailed` event with relevant details.

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

Repo: `https://github.com/StartaleLabs/scs-aa-account-contracts`

- `ff8...44b ./.env.example`
- `597...4f4 ./.gas-snapshot`
- `1af...fd7 ./.gitattributes`
- `be7...af9 ./.github/workflows/canary.yml`
- `48c...5c3 ./.github/workflows/coverage.yml`
- `22f...f26 ./.github/workflows/release.yml`
- `380...3cb ./.github/workflows/tests.yml`
- `3fd...961 ./.gitignore`
- `8a2...33c ./.gitmodules`
- `441...284 ./.husky/.gitignore`
- `b61...e40 ./.husky/commit-msg`
- `aca...ac9 ./.husky/pre-commit`
- `0a0...5a8 ./.solhint.json`
- `c32...51c ./LICENSE`
- `f22...313 ./README.md`
- `9da...d67 ./arguments.js`
- `651...5fe ./commitlint.config.js`
- `57f...df0 ./foundry.toml`
- `792...78d ./hardhat.config.ts`
- `04b...a18 ./medusa.json`
- `aba...2ee ./natspec-smells.config.js`
- `af4...205 ./package.json`
- `2f4...bc6 ./remappings.txt`
- `3ad...7eb ./script/.solhint.json`
- `39d...85b ./script/Deploy.sol`
- `3ad...7eb ./scripts/foundry/.solhint.json`
- `39d...85b ./scripts/foundry/Deploy.sol`
- `1f4...ac2 ./scripts/foundry/DeployStartaleAccountContracts.sol`
- `e58...04f ./scripts/hardhat/deploy-contracts.ts`
- `3d7...dbb ./src/Greeter.sol`
- `b4e...24f ./src/StartaleSmartAccount.sol`
- `0b6...054 ./src/core/AllStorage.sol`
- `193...58f ./src/core/BaseAccount.sol`
- `1ec...570 ./src/core/ERC7779Adapter.sol`
- `a61...f56 ./src/core/ExecutionHelper.sol`
- `133...937 ./src/core/ModuleManager.sol`
- `6cb...bb4 ./src/factory/EOAOnboardingFactory.sol`
- `3a8...eb8 ./src/factory/StartaleAccountFactory.sol`
- `cc1...e3e ./src/interfaces/.solhint.json`
- `4ff...416 ./src/interfaces/IAccountEventsAndErrors.sol`
- `1c1...dcd ./src/interfaces/IERC4337Account.sol`
- `e77...834 ./src/interfaces/IERC7579Account.sol`
- `101...d7c ./src/interfaces/IERC7579Module.sol`
- `87f...93a ./src/interfaces/IERC7779.sol`
- `074...c9e ./src/interfaces/IGreeter.sol`
- `12a...1fb ./src/interfaces/IStartaleAccountFactory.sol`
- `791...116 ./src/interfaces/IStartaleSmartAccount.sol`
- `034...c90 ./src/interfaces/core/IAccountConfig.sol`

- `02f...061 ./src/interfaces/core/IAllStorage.sol`
- `ac6...9a8 ./src/interfaces/core/IBaseAccount.sol`
- `084...71b ./src/interfaces/core/IExecutionHelper.sol`
- `b24...8e2 ./src/interfaces/core/IModuleManager.sol`
- `67c...164 ./src/interfaces/core/IModuleManagerEventsAndErrors.sol`
- `839...ec7 ./src/lib/AssociatedArrayLib.sol`
- `738...e07 ./src/lib/DataParserLib.sol`
- `47a...f72 ./src/lib/EnumerableSet4337.sol`
- `3b3...d1c ./src/lib/ExecutionLib.sol`
- `28f...9f1 ./src/lib/Initializable.sol`
- `218...65e ./src/lib/ModeLib.sol`
- `654...bfe ./src/lib/NonceLib.sol`
- `5c2...189 ./src/lib/ProxyLib.sol`
- `711...615 ./src/modules/validators/ECDSAValidator.sol`
- `5ce...bbe ./src/types/Constants.sol`
- `16f...904 ./src/types/Structs.sol`
- `e8b...246 ./src/types/Types.sol`
- `967...5be ./src/utils/AccountProxy.sol`
- `f1c...57b ./src/utils/Bootstrap.sol`
- `7d0...a31 ./src/utils/Stakeable.sol`
- `242...907 ./test/.solhint.json`
- `749...592 ./test/foundry/integration/IntegrationBase.sol`
- `74d...fd3 ./test/foundry/invariants/PROPERTIES.md`
- `0b9...a62 ./test/foundry/invariants/fuzz/FuzzTest.t.sol`
- `adc...34e ./test/foundry/invariants/fuzz/handlers/guided/Greeter.t.sol`
- `4eb...062 ./test/foundry/invariants/fuzz/handlers/unguided/Greeter.t.sol`
- `62e...948 ./test/foundry/invariants/fuzz/properties/Greeter.t.sol`
- `390...5a9 ./test/foundry/invariants/fuzz/setup/Greeter.t.sol`
- `cba...126 ./test/foundry/invariants/symbolic/Greeter.t.sol`
- `c88...b10 ./test/foundry/mocks/Counter.sol`
- `b22...568 ./test/foundry/mocks/EmittingHook.sol`
- `7b9...8c3 ./test/foundry/mocks/Exposed7702SmartAccount.sol`
- `cc2...717 ./test/foundry/mocks/MockAccountLocker.sol`
- `f4a...bcb ./test/foundry/mocks/MockDelegateTarget.sol`
- `8e7...fb8 ./test/foundry/mocks/MockERC7739PreValidationHook.sol`
- `129...a67 ./test/foundry/mocks/MockERC7779.sol`
- `06f...546 ./test/foundry/mocks/MockExecutor.sol`
- `29e...844 ./test/foundry/mocks/MockHandler.sol`
- `1c2...a0f ./test/foundry/mocks/MockHook.sol`
- `6e0...43e ./test/foundry/mocks/MockInvalidModule.sol`
- `334...3ad ./test/foundry/mocks/MockMultiModule.sol`
- `1a8...d77 ./test/foundry/mocks/MockNFT.sol`
- `2ff...08d ./test/foundry/mocks/MockPaymaster.sol`
- `570...cd1 ./test/foundry/mocks/MockPreValidationHook.sol`
- `98c...3cf ./test/foundry/mocks/MockPreValidationHookMultiplexer.sol`
- `a07...e1e ./test/foundry/mocks/MockResourceLockPreValidationHook.sol`
- `c9a...9c3 ./test/foundry/mocks/MockSafe1271Caller.sol`
- `8b9...1b9 ./test/foundry/mocks/MockSimpleValidator.sol`
- `37b...e7c ./test/foundry/mocks/MockTarget.sol`
- `8cb...4cd ./test/foundry/mocks/MockToken.sol`
- `81e...3fe ./test/foundry/mocks/MockTransferer.sol`
- `ce8...6f1 ./test/foundry/mocks/MockValidator.sol`
- `60f...e7f ./test/foundry/mocks/MockValidator7739.sol`
- `f63...345 ./test/foundry/mocks/TokenWithPermit.sol`
- `9da...d33 ./test/foundry/shared/TestExecutionBase.t.sol`

- `3b7...ec1 ./test/foundry/shared/TestModuleManagerBase.t.sol`
- `ce6...949 ./test/foundry/unit/concrete/4337account/Test4337_AddDeposit.t.sol`
- `3cc...dd3 ./test/foundry/unit/concrete/4337account/Test4337_EntryPoint.t.sol`
- `ce6...949 ./test/foundry/unit/concrete/4337account/Test4337_GetDeposit.t.sol`
- `e87...2c3 ./test/foundry/unit/concrete/4337account/Test4337_NonceRelated.t.sol`
- `bc1...f29 ./test/foundry/unit/concrete/4337account/Test4337_OnlyEntryPoint.t.sol`
- `d30...b8a ./test/foundry/unit/concrete/4337account/Test4337_OnlyEntryPointOrSelf.t.sol`
- `d41...439 ./test/foundry/unit/concrete/4337account/Test4337_PayPrefund.t.sol`
- `1b6...f34 ./test/foundry/unit/concrete/4337account/Test4337_ValidateUserOp.t.sol`
- `f42...889 ./test/foundry/unit/concrete/4337account/Test4337_WithdrawDepositTo.t.sol`
- `794...464 ./test/foundry/unit/concrete/config/Test_AccountConfig.t.sol`
- `3b8...a2c ./test/foundry/unit/concrete/config/Test_AccountConfig_SupportsMode.t.sol`
- `dad...733 ./test/foundry/unit/concrete/eip7702/TestEIP7702.t.sol`
- `496...4c7 ./test/foundry/unit/concrete/erc1271/Test1271_IsValidSignature.t.sol`
- `821...a9a ./test/foundry/unit/concrete/erc1271/Test1271_WithMockProtocol.t.sol`
- `e9e...174 ./test/foundry/unit/concrete/execution/TestExecution_ExecuteBatch.t.sol`
- `97f...846 ./test/foundry/unit/concrete/execution/TestExecution_ExecuteDelegateCall.t.sol`
- `918...06e ./test/foundry/unit/concrete/execution/TestExecution_ExecuteFromExecutor.t.sol`
- `47d...3d1 ./test/foundry/unit/concrete/execution/TestExecution_ExecuteSingle.t.sol`
- `9bd...024 ./test/foundry/unit/concrete/execution/TestExecution_ExecuteUserOp.t.sol`
- `101...114 ./test/foundry/unit/concrete/execution/TestExecution_TryExecuteBatch.t.sol`
- `5a7...72b ./test/foundry/unit/concrete/execution/TestExecution_TryExecuteSingle.t.sol`
- `f1c...cc2 ./test/foundry/unit/concrete/executionlib/TestExecutionLib.t.sol`
- `84f...9e9 ./test/foundry/unit/concrete/factory/TestEOAOnboardingFactory_Deployments.t.sol`
- `ca8...1e4 ./test/foundry/unit/concrete/factory/TestStartaleAccountFactory_Deployments.t.sol`
- `9f3...25a ./test/foundry/unit/concrete/fallback/TestFallbackFunc.t.sol`
- `172...482 ./test/foundry/unit/concrete/hook/TestHook_EmergencyUninstall.t.sol`
- `a6d...0fe ./test/foundry/unit/concrete/modelib/TestModeLib.t.sol`
- `29c...881 ./test/foundry/unit/concrete/modulemanager/TestModuleManager.Hooking.t.sol`
- `9e4...58f ./test/foundry/unit/concrete/modulemanager/TestModuleManager_EnableMode.t.sol`
- `0dd...43f ./test/foundry/unit/concrete/modulemanager/TestModuleManager_FallbackHandler.t.sol`
- `01b...c8b ./test/foundry/unit/concrete/modulemanager/TestModuleManager_InstallModule.t.sol`
- `aab...113 ./test/foundry/unit/concrete/modulemanager/TestModuleManager_SupportsModule.t.sol`
- `50a...fae ./test/foundry/unit/concrete/modulemanager/TestModuleManager_UninstallModule.t.sol`
- `6a6...96a ./test/foundry/unit/concrete/modules/TestAccount_WithECDSAValidator.t.sol`
- `a5c...da6 ./test/foundry/unit/concrete/utils/TestStakeable.t.sol`
- `00f...d0f ./test/foundry/unit/fuzz/TestAccountFactory_Deployment.t.sol`
- `5a2...dde ./test/foundry/unit/fuzz/erc7779/TestFuzz_ERC7779Adapter.t.sol`
- `01c...b75 ./test/foundry/utils/BootstrapLib.sol`
- `cf7...426 ./test/foundry/utils/CalculateSelectors.t.sol`
- `1ab...435 ./test/foundry/utils/CheatCodes.sol`
- `3d0...5ea ./test/foundry/utils/EventsAndErrors.sol`
- `7e7...75b ./test/foundry/utils/ModuleTypeLib.sol`
- `87b...985 ./test/foundry/utils/Structs.sol`
- `265...624 ./test/foundry/utils/TestBase.sol`
- `859...a6b ./test/foundry/utils/TestHelper.sol`
- `fa1...e0f ./tsconfig.json`
- `62e...8d4 ./yarn.lock`

Test Suite Results

The test data is generated by running `forge test`.

```
Ran 6 tests for
test/foundry/unit/concrete/erc1271/Test1271_IsValidSignature.t.sol:TestERC1271Account_IsValidSignature
[PASS] test_ERC7739SupportDetectionRequest() (gas: 590565)
[PASS] test_isValidSignature_EIP712Sign_K1Validator_Success() (gas: 70599)
[PASS] test_isValidSignature_EIP712Sign_K1Validator_Wrong1271Signer_Fail() (gas: 1024174498)
[PASS] test_isValidSignature_ERC6492Unwrapping() (gas: 74324)
[PASS] test_isValidSignature_NoERC6492Unwrapping() (gas: 47836)
[PASS] test_isValidSignature_PersonalSign_K1Validator_Success() (gas: 49269)
Suite result: ok. 6 passed; 0 failed; 0 skipped; finished in 178.43ms (15.87ms CPU time)

Ran 6 tests for
test/foundry/unit/concrete/4337account/Test4337_ValidateUserOp.t.sol:TestERC4337Account_ValidateUserOp
[PASS] testPayPrefund_WithSufficientFunds() (gas: 78161)
[PASS] test_RevertWhen_InvalidNonce() (gas: 101328)
[PASS] test_ValidateUserOp_InsufficientFunds() (gas: 68784)
[PASS] test_ValidateUserOp_InvalidSignature() (gas: 62410)
[PASS] test_ValidateUserOp_InvalidSignatureFormat() (gas: 46967)
[PASS] test_ValidateUserOp_ValidOperation() (gas: 62748)
Suite result: ok. 6 passed; 0 failed; 0 skipped; finished in 178.50ms (16.67ms CPU time)

Ran 8 tests for
test/foundry/unit/concrete/4337account/Test4337_GetDeposit.t.sol:TestERC4337Account_AddDeposit
[PASS] test_AddDeposit_BatchDepositViaHandleOps() (gas: 205444)
[PASS] test_AddDeposit_DepositViaHandleOps() (gas: 131521)
[PASS] test_AddDeposit_EventEmitted() (gas: 53531)
[PASS] test_AddDeposit_RevertIf_WrongEntryPoint() (gas: 36529)
[PASS] test_AddDeposit_Success() (gas: 43291)
[PASS] test_AddDeposit_Try_BatchDepositViaHandleOps() (gas: 206626)
[PASS] test_AddDeposit_Try_DepositViaHandleOps() (gas: 131770)
[PASS] test_RevertIf_AddDeposit_NoValue() (gas: 16859)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 181.66ms (20.02ms CPU time)

Ran 26 tests for
test/foundry/unit/concrete/execution/TestExecution_ExecuteFromExecutor.t.sol:TestExecution_ExecuteFromExe
cutor
[PASS] test_ExecBatchFromExecutor_Success() (gas: 69668)
[PASS] test_ExecuteBatchEmpty_Success() (gas: 27629)
[PASS] test_ExecuteBatch_MixedOutcomes_Success() (gas: 63196)
[PASS] test_ExecuteDelegateCallFromExecutor_Success() (gas: 76919)
[PASS] test_ExecuteERC20ApproveAndTransferBatch_Success() (gas: 98103)
[PASS] test_ExecuteERC20TransferExecutor_Success() (gas: 64948)
[PASS] test_ExecuteERC20TransferFromExecutor_Success() (gas: 64992)
[PASS] test_ExecuteSingleFromExecutor_Success() (gas: 143188)
[PASS] test_ExecuteSingleValueTransfer_Success() (gas: 71177)
[PASS] test_RevertIf_ExecuteFromExecutor_UnsupportedCallType() (gas: 30064)
[PASS] test_RevertIf_ExecuteFromExecutor_UnsupportedExecType_Batch() (gas: 33156)
[PASS] test_RevertIf_ExecuteFromExecutor_UnsupportedExecType_Single() (gas: 31711)
[PASS] test_RevertIf_UnauthorizedExecutor() (gas: 775095)
[PASS] test_RevertIf_ZeroValueTransferInBatch() (gas: 44387)
[PASS] test_TryExecuteBatchViaAccount_AllFail() (gas: 61396)
[PASS] test_TryExecuteBatchViaAccount_MixedOutcomes() (gas: 76653)
[PASS] test_TryExecuteBatchViaAccount_Success() (gas: 74393)
[PASS] test_TryExecuteBatchViaAccount_ValueTransfer() (gas: 76251)
[PASS] test_TryExecuteBatch_EmptyCallData() (gas: 46128)
[PASS] test_TryExecuteBatch_InsufficientGas() (gas: 23021)
[PASS] test_TryExecuteBatch_MultipleFailures() (gas: 83124)
[PASS] test_TryExecuteBatch_SingleFailure() (gas: 72033)
[PASS] test_TryExecuteBatch_SingleFailure_WithPrank() (gas: 64363)
[PASS] test_TryExecuteViaAccount_Revert() (gas: 41984)
[PASS] test_TryExecuteViaAccount_Success() (gas: 58804)
[PASS] test_TryExecuteViaAccount_ValueTransfer() (gas: 72887)
Suite result: ok. 26 passed; 0 failed; 0 skipped; finished in 182.98ms (20.98ms CPU time)

Ran 3 tests for
test/foundry/unit/concrete/4337account/Test4337_NonceRelated.t.sol:TestERC4337Account_Nonce
[PASS] test_InitialNonce() (gas: 27379)
[PASS] test_NonceIncrementAfterOperation() (gas: 136826)
[PASS] test_NonceIncrementedEvenOnFailedOperation() (gas: 126064)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 5.32ms (1.55ms CPU time)
```

```
Ran 8 tests for
test/foundry/unit/concrete/4337account/Test4337_WithdrawDepositTo.t.sol:TestERC4337Account_WithdrawDeposi
tTo
[PASS] test_RevertIf_WithdrawDepositTo_ContractAddress() (gas: 36035)
[PASS] test_RevertIf_WithdrawDepositTo_ExceedsAvailable() (gas: 20133)
[PASS] test_RevertIf_WithdrawDepositTo_InsufficientGas() (gas: 8939)
[PASS] test_RevertIf_WithdrawDepositTo_UnauthorizedAddress() (gas: 20091)
[PASS] test_WithdrawDepositTo_AuthorizedAddress() (gas: 122100)
[PASS] test_WithdrawDepositTo_Self() (gas: 117716)
[PASS] test_WithdrawDepositTo_Success() (gas: 145183)
[PASS] test_WithdrawDepositTo_ZeroAmount() (gas: 106734)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 8.79ms (4.89ms CPU time)

Ran 10 tests for
test/foundry/unit/concrete/execution/TestExecution_ExecuteSingle.t.sol:TestExecution_ExecuteSingle
[PASS] test_ExecuteSingle_ApproveAndTransferFrom() (gas: 171481)
[PASS] test_ExecuteSingle_Empty_Success() (gas: 100911)
[PASS] test_ExecuteSingle_Success() (gas: 130699)
[PASS] test_ExecuteSingle_TokenTransfer() (gas: 139836)
[PASS] test_ExecuteSingle_ValueTransfer() (gas: 144220)
[PASS] test_RevertIf_ExecuteSingle_Failure() (gas: 119790)
[PASS] test_RevertIf_ExecuteSingle_ZeroAddress() (gas: 110353)
[PASS] test_RevertIf_SingleExecutionWithUnsupportedExecType() (gas: 117140)
[PASS] test_SingleExecution_RevertOnUnsupportedCallType_FromAccount() (gas: 32897)
[PASS] test_SingleExecution_RevertOnUnsupportedCallType_FromEntryPoint() (gas: 33139)
Suite result: ok. 10 passed; 0 failed; 0 skipped; finished in 8.74ms (4.86ms CPU time)

Ran 3 tests for
test/foundry/unit/concrete/erc1271/Test1271_WithMockProtocol.t.sol:TestERC1271Account_MockProtocol
[PASS] test_RevertWhen_SignatureIsInvalidDueToWrongAllowance() (gas: 1008177552)
[PASS] test_RevertWhen_SignatureIsInvalidDueToWrongSigner() (gas: 1008177566)
[PASS] test_isValidSignature_EIP712Sign_Success() (gas: 127003)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 9.72ms (1.66ms CPU time)

Ran 4 tests for
test/foundry/unit/concrete/4337account/Test4337_OnlyEntryPoint.t.sol:TestERC4337Account_OnlyEntryPoint
[PASS] test_RevertIf_InvalidUserOpSignature() (gas: 62674)
[PASS] test_RevertIf_UserOpFromNonEntryPoint() (gas: 42271)
[PASS] test_ValidUserOpFromEntryPoint() (gas: 69452)
[PASS] test_ValidateUserOp_InvalidSignature() (gas: 62970)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 7.30ms (3.74ms CPU time)

Ran 5 tests for
test/foundry/unit/concrete/execution/TestExecution_ExecuteUserOp.t.sol:TestExecution_ExecuteUserOp
[PASS] test_ExecuteUserOp_EmptyCalldata() (gas: 109403)
[PASS] test_ExecuteUserOp_ShouldExecute() (gas: 133092)
[PASS] test_ExecuteUserOp_ZeroAddress() (gas: 116530)
[PASS] test_RevertIf_ExecuteUserOp_InvalidSignature() (gas: 75024)
[PASS] test_SetUpState() (gas: 10919)
Suite result: ok. 5 passed; 0 failed; 0 skipped; finished in 6.72ms (2.56ms CPU time)

Ran 8 tests for
test/foundry/unit/concrete/4337account/Test4337_AddDeposit.t.sol:TestERC4337Account_AddDeposit
[PASS] test_AddDeposit_BatchDepositViaHandleOps() (gas: 205444)
[PASS] test_AddDeposit_DepositViaHandleOps() (gas: 131521)
[PASS] test_AddDeposit_EventEmitted() (gas: 53531)
[PASS] test_AddDeposit_RevertIf_WrongEntryPoint() (gas: 36529)
[PASS] test_AddDeposit_Success() (gas: 43291)
[PASS] test_AddDeposit_Try_BatchDepositViaHandleOps() (gas: 206626)
[PASS] test_AddDeposit_Try_DepositViaHandleOps() (gas: 131770)
[PASS] test_RevertIf_AddDeposit_NoValue() (gas: 16859)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 6.72ms (2.98ms CPU time)

Ran 12 tests for
test/foundry/unit/concrete/4337account/Test4337_OnlyEntryPointOrSelf.t.sol:TestERC4337Account_OnlyEntryPo
intOrSelf
[PASS] test_ExecuteUserOp_Valid_FromEntryPoint() (gas: 100538)
[PASS] test_InstallModuleFromEntryPoint_Success() (gas: 57188)
[PASS] test_InstallModuleFromSelf_Success() (gas: 54945)
[PASS] test_InstallModuleWithUserOpsFromEntryPoint_Success() (gas: 140634)
[PASS] test_RevertIf_ExecuteUserOp_FromNonEntryPoint() (gas: 44745)
[PASS] test_RevertIf_InstallModuleFromUnauthorized() (gas: 19696)
```

```
[PASS] test_RevertIf_UninstallModuleFromNonEntryPointOrSelf() (gas: 19562)
[PASS] test_RevertIf_WithdrawDeposit_FromUnauthorizedAddress() (gas: 18957)
[PASS] test_WithdrawDepositFromSelf_Success() (gas: 35907)
[PASS] test_WithdrawDepositViaExecutor() (gas: 95290)
[PASS] test_WithdrawDeposit_ToAuthorizedAddress() (gas: 37732)
[PASS] test_WithdrawDeposit_ToAuthorizedAddress_WithUserOps() (gas: 116437)
Suite result: ok. 12 passed; 0 failed; 0 skipped; finished in 7.24ms (2.64ms CPU time)

Ran 8 tests for
test/foundry/unit/concrete/4337account/Test4337_EntryPoint.t.sol:TestERC4337Account_AddDeposit
[PASS] test_AddDeposit_BatchDepositViaHandleOps() (gas: 205444)
[PASS] test_AddDeposit_DepositViaHandleOps() (gas: 131521)
[PASS] test_AddDeposit_EventEmitted() (gas: 53531)
[PASS] test_AddDeposit_RevertIf_WrongEntryPoint() (gas: 36529)
[PASS] test_AddDeposit_Success() (gas: 43291)
[PASS] test_AddDeposit_Try_BatchDepositViaHandleOps() (gas: 206626)
[PASS] test_AddDeposit_Try_DepositViaHandleOps() (gas: 131770)
[PASS] test_RevertIf_AddDeposit_NoValue() (gas: 16859)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 6.55ms (2.92ms CPU time)

Ran 9 tests for
test/foundry/unit/concrete/execution/TestExecution_TryExecuteBatch.t.sol:TestExecution_TryExecuteBatch
[PASS] test_TryExecuteBatch_ApproveAndTransfer_SeparateOps() (gas: 234990)
[PASS] test_TryExecuteBatch_ApproveAndTransfer_SingleOp() (gas: 162477)
[PASS] test_TryExecuteBatch_Empty() (gas: 115059)
[PASS] test_TryExecuteBatch_RevertIf_HandleFailure() (gas: 143960)
[PASS] test_TryExecuteBatch_RevertIf_HandleFailure_WithPrank() (gas: 62748)
[PASS] test_TryExecuteBatch_RevertIf_HandleMultipleFailures() (gas: 140824)
[PASS] test_TryExecuteBatch_Success() (gas: 138130)
[PASS] test_TryExecuteBatch_TokenTransfers() (gas: 173431)
[PASS] test_TryExecuteBatch_ValueTransfer() (gas: 164398)
Suite result: ok. 9 passed; 0 failed; 0 skipped; finished in 10.66ms (6.46ms CPU time)

Ran 1 test for
test/foundry/unit/concrete/4337account/Test4337_PayPrefund.t.sol:TestERC4337Account_PayPrefund
[PASS] testPayPrefund_WithSufficientFunds() (gas: 77978)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 5.33ms (623.88µs CPU time)

Ran 5 tests for
test/foundry/unit/concrete/modulemanager/TestModuleManager.Hooking.t.sol:TestModuleManager_HookModule
[PASS] test_GetActiveHook_Success() (gas: 143948)
[PASS] test_HookTriggeredOnModuleInstallation() (gas: 260924)
[PASS] test_InstallHookModule_Success() (gas: 141812)
[PASS] test_RevertIf_ReinstallHookModule() (gas: 463504)
[PASS] test_UninstallHookModule_Success() (gas: 212656)
Suite result: ok. 5 passed; 0 failed; 0 skipped; finished in 9.30ms (5.56ms CPU time)

Ran 2 tests for
test/foundry/unit/concrete/modulemanager/TestModuleManager_EnableMode.t.sol:TestModuleManager_EnableMode
[PASS] test_encodeDecodeBatch_Success() (gas: 4443)
[PASS] test_encodeDecodeSingle_Success() (gas: 4285)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 97.58µs (33.29µs CPU time)

Ran 8 tests for
test/foundry/unit/concrete/execution/TestExecution_TryExecuteSingle.t.sol:TestExecution_TryExecuteSingle
[PASS] test_RevertIf_TryExecuteSingle_Fails() (gas: 112930)
[PASS] test_TryExecuteDelegateCall_EmitTryDelegateCallUnsuccessful() (gas: 116932)
[PASS] test_TryExecuteSingle_ApproveAndTransferFrom() (gas: 171341)
[PASS] test_TryExecuteSingle_EmitTryExecuteUnsuccessful() (gas: 116775)
[PASS] test_TryExecuteSingle_Empty() (gas: 100898)
[PASS] test_TryExecuteSingle_Success() (gas: 130570)
[PASS] test_TryExecuteSingle_TokenTransfer() (gas: 140337)
[PASS] test_TryExecuteSingle_ValueTransfer() (gas: 144647)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 11.43ms (7.01ms CPU time)

Ran 10 tests for
test/foundry/unit/concrete/modulemanager/TestModuleManager_UninstallModule.t.sol:TestModuleManager_Uninst
allModule
[PASS] test_ExecutorModuleUninstallation_Success() (gas: 962764)
[PASS] test_ModuleInstallation_Success() (gas: 148485)
[PASS] test_ModuleUninstallation_Success() (gas: 1194360)
[PASS] test_NewModuleUninstallation_Success() (gas: 1062293)
```

```
[PASS] test_RevertIf_IncorrectModuleTypeUninstallation() (gas: 131755)
[PASS] test_RevertIf_IncorrectPrevModuleData() (gas: 222004)
[PASS] test_RevertIf_UninstallingNonExistentFallbackHandler() (gas: 123527)
[PASS] test_RevertIf_UninstallingNonExistentModule() (gas: 132464)
[PASS] test_SuccessfulUninstallationOfExecutorModule() (gas: 934263)
[PASS] test_SuccessfulUninstallationOfFallbackHandler() (gas: 185376)
Suite result: ok. 10 passed; 0 failed; 0 skipped; finished in 23.51ms (17.62ms CPU time)

Ran 10 tests for test/foundry/unit/concrete/utils/TestStakeable.t.sol:TestStakeable
[PASS] test_AddStake_RevertIf_InvalidEPAddress() (gas: 20597)
[PASS] test_AddStake_RevertIf_NotOwner() (gas: 19512)
[PASS] test_AddStake_Success() (gas: 66985)
[PASS] test_DeployStakeable() (gas: 356043)
[PASS] test_UnlockStake_RevertIf_InvalidEPAddress() (gas: 13762)
[PASS] test_UnlockStake_RevertIf_NotOwner() (gas: 12617)
[PASS] test_UnlockStake_Success() (gas: 65628)
[PASS] test_WithdrawStake_RevertIf_InvalidEPAddress() (gas: 14046)
[PASS] test_WithdrawStake_RevertIf_NotOwner() (gas: 13212)
[PASS] test_WithdrawStake_Success() (gas: 84950)
Suite result: ok. 10 passed; 0 failed; 0 skipped; finished in 5.77ms (872.54µs CPU time)

Ran 16 tests for test/foundry/unit/concrete/fallback/TestFallbackFunc.t.sol:TestFallbackFunction
[PASS] test_FallbackFunction_AuthorizedEntryPoint() (gas: 147079)
[PASS] test_FallbackFunction_AuthorizedExecutorModule() (gas: 234820)
[PASS] test_FallbackFunction_AuthorizedSelf() (gas: 146690)
[PASS] test_FallbackFunction_UnauthorizedEntity() (gas: 146086)
[PASS] test_FallbackHandlerInsufficientGas() (gas: 145033)
[PASS] test_FallbackHandlerInvalidCallType() (gas: 123472)
[PASS] test_FallbackHandlerInvalidFunctionSelector() (gas: 146785)
[PASS] test_FallbackHandlerMissingHandler() (gas: 17864)
[PASS] test_FallbackHandlerSingleCall_Revert() (gas: 147991)
[PASS] test_FallbackHandlerSingleCall_Success() (gas: 147013)
[PASS] test_FallbackHandlerStateChange_SingleCall() (gas: 169785)
[PASS] test_FallbackHandlerStateChange_StaticCall() (gas: 1024177350)
[PASS] test_FallbackHandlerStaticCall_Revert() (gas: 147780)
[PASS] test_FallbackHandlerStaticCall_Success() (gas: 147284)
[PASS] test_SetFallbackHandler_Success() (gas: 144865)
[PASS] test_receive_transfer() (gas: 102304)
Suite result: ok. 16 passed; 0 failed; 0 skipped; finished in 21.43ms (16.15ms CPU time)

Ran 13 tests for
test/foundry/unit/concrete/factory/TestStartaleAccountFactory_Deployments.t.sol:TestStartaleAccountFactor
y_Deployments
[PASS] test_ComputeAccountAddress_ManualComparison() (gas: 24556)
[PASS] test_Constructor_RevertIf_EntryPointIsZero() (gas: 76117)
[PASS] test_Constructor_RevertIf_ImplementationIsZero() (gas: 60769)
[PASS] test_DeployAccount_CannotReinitialize() (gas: 219977)
[PASS] test_DeployAccount_CreateAccount() (gas: 221321)
[PASS] test_DeployAccount_CreateAccount_SameAddress() (gas: 225634)
[PASS] test_DeployAccount_DifferentIndexes() (gas: 402548)
[PASS] test_DeployAccount_HandleOps_Success() (gas: 357149)
[PASS] test_DeployAccount_InvalidValidatorModule() (gas: 143411)
[PASS] test_RevertIf_DeployAccount_InsufficientGas() (gas: 19163)
[PASS] test_RevertIf_HandleOps_AccountExists() (gas: 363833)
[PASS] test_createArrayConfig_MultipleModules_DeployAccount() (gas: 275725)
[PASS] test_initScoped_WithHook_DeployAccount() (gas: 254827)
Suite result: ok. 13 passed; 0 failed; 0 skipped; finished in 9.05ms (3.92ms CPU time)

Ran 1 test for test/foundry/unit/concrete/config/Test_AccountConfig.t.sol:TestAccountConfig_AccountId
[PASS] test_WhenCheckingTheAccountID() (gas: 10083)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 1.83ms (41.67µs CPU time)

Ran 3 tests for
test/foundry/unit/concrete/config/Test_AccountConfig_SupportsMode.t.sol:TestAccountConfig_SupportsExecuti
onMode
[PASS] test_RevertIf_UnsupportedExecutionMode() (gas: 14731)
[PASS] test_SupportsBatchExecutionMode_Success() (gas: 14507)
[PASS] test_SupportsSingleExecutionMode_Success() (gas: 14596)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 4.74ms (82.79µs CPU time)

Ran 17 tests for
test/foundry/unit/concrete/modulemanager/TestModuleManager_InstallModule.t.sol:TestModuleManager_InstallM
```

```
odule
[PASS] test_InstallExecutorModule_Success() (gas: 142542)
[PASS] test_InstallFallbackHandler_WithCustomData() (gas: 145165)
[PASS] test_InstallHookModule_Success() (gas: 140079)
[PASS] test_InstallModule_MultiTypeInstall() (gas: 334175)
[PASS] test_InstallModule_Success() (gas: 150646)
[PASS] test_InstallModule_TrySuccess() (gas: 150060)
[PASS] test_InstallPreValidationHooks_Success() (gas: 508880)
[PASS] test_InstallValidatorModule_Success() (gas: 144826)
[PASS] test_RevertIf_IncompatibleExecutorModule() (gas: 984224)
[PASS] test_RevertIf_IncompatibleModuleAsExecutor() (gas: 122149)
[PASS] test_RevertIf_IncompatibleValidatorModule() (gas: 867641)
[PASS] test_RevertIf_InvalidModuleAddress() (gas: 116247)
[PASS] test_RevertIf_InvalidModuleTypeId() (gas: 983419)
[PASS] test_RevertIf_InvalidModuleWithInvalidTypeId() (gas: 216603)
[PASS] test_RevertIf_ModuleAlreadyInstalled() (gas: 219201)
[PASS] test_RevertIf_ReinstallFallbackHandler() (gas: 204434)
[PASS] test_RevertIf_ReinstallHookModule() (gas: 230985)
Suite result: ok. 17 passed; 0 failed; 0 skipped; finished in 20.38ms (14.99ms CPU time)

Ran 6 tests for
test/foundry/unit/concrete/modulemanager/TestModuleManager_SupportsModule.t.sol:TestModuleManager_Support
sModule
[PASS] test_SupportsModuleExecutor_Success() (gas: 14746)
[PASS] test_SupportsModuleFallback_Success() (gas: 14838)
[PASS] test_SupportsModuleHook_Success() (gas: 14773)
[PASS] test_SupportsModuleMultiType_Success() (gas: 14630)
[PASS] test_SupportsModuleValidator_Success() (gas: 14280)
[PASS] test_SupportsModule_FailsForUnsupportedModule() (gas: 14974)
Suite result: ok. 6 passed; 0 failed; 0 skipped; finished in 4.41ms (144.21µs CPU time)

Ran 12 tests for
test/foundry/unit/concrete/hook/TestHook_EmergencyUninstall.t.sol:TestHook_EmergencyUninstall
[PASS] test_EmergencyUninstallHook_1271_DirectCall_Fail_WrongSigner() (gas: 413250)
[PASS] test_EmergencyUninstallHook_1271_DirectCall_Success() (gas: 434463)
[PASS] test_EmergencyUninstallHook_4337_DirectCall_Fail_WrongSigner() (gas: 411162)
[PASS] test_EmergencyUninstallHook_4337_DirectCall_Success() (gas: 433069)
[PASS] test_EmergencyUninstallHook_DirectCall_Fail_WrongSigner() (gas: 204076)
[PASS] test_EmergencyUninstallHook_DirectCall_Success() (gas: 218758)
[PASS] test_EmergencyUninstallHook_Fail_AfterInitiated() (gas: 371029)
[PASS] test_EmergencyUninstallHook_Initiate_Success() (gas: 282233)
[PASS] test_EmergencyUninstallHook_PreValidation1271_Uninstall() (gas: 439144)
[PASS] test_EmergencyUninstallHook_PreValidation4337_Uninstall() (gas: 437714)
[PASS] test_EmergencyUninstallHook_Success_LongAfterInitiated() (gas: 330642)
[PASS] test_EmergencyUninstallHook_Success_Reset_SuperLongAfterInitiated() (gas: 368551)
Suite result: ok. 12 passed; 0 failed; 0 skipped; finished in 22.94ms (17.68ms CPU time)

Ran 2 tests for test/foundry/unit/concrete/modelib/TestModeLib.t.sol:ModeLibTest
[PASS] test_encodeDecodeBatch_Success() (gas: 4443)
[PASS] test_encodeDecodeSingle_Success() (gas: 4285)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 121.25µs (32.92µs CPU time)

Ran 10 tests for test/foundry/unit/concrete/eip7702/TestEIP7702.t.sol:TestEIP7702
[PASS] test_amIERC7702_success() (gas: 5367583)
[PASS] test_delegateCall() (gas: 192255)
[PASS] test_delegateCall_fromExecutor() (gas: 434045)
[PASS] test_execBatch() (gas: 218436)
[PASS] test_execBatchFromExecutor() (gas: 402445)
[PASS] test_execSingle() (gas: 179241)
[PASS] test_execSingleFromExecutor() (gas: 395165)
[PASS] test_initializeAndExecSingle() (gas: 385819)
[PASS] test_onRedelegation() (gas: 671211)
[PASS] test_transfer_to_eip7702_account() (gas: 103529)
Suite result: ok. 10 passed; 0 failed; 0 skipped; finished in 11.37ms (7.48ms CPU time)

Ran 11 tests for
test/foundry/unit/concrete/factory/TestEOAOnboardingFactory_Deployments.t.sol:TestEOAOnboardingFactory_De
ployments
[PASS] test_ComputeAccountAddress() (gas: 217397)
[PASS] test_ComputeAccountAddress_MatchesManualComputation() (gas: 20667)
[PASS] test_ConstructorInitializesFactory() (gas: 3379331)
[PASS] test_ConstructorInitializesWithRegistryAddressZero() (gas: 3379107)
```

```
[PASS] test_Constructor_RevertIf_BootstrapperIsZero() (gas: 65055)
[PASS] test_Constructor_RevertIf_FactoryOwnerIsZero() (gas: 45287)
[PASS] test_Constructor_RevertIf_ImplementationIsZero() (gas: 65537)
[PASS] test_Constructor_RevertIf_K1ValidatorIsZero() (gas: 65187)
[PASS] test_CreateAccount_DifferentIndexes() (gas: 406109)
[PASS] test_CreateAccount_SameOwnerAndIndex() (gas: 231894)
[PASS] test_DeployAccount_EOAOnboardingFactory_CreateAccount() (gas: 221852)
Suite result: ok. 11 passed; 0 failed; 0 skipped; finished in 5.57ms (1.11ms CPU time)

Ran 1 test for test/foundry/unit/concrete/executionlib/TestExecutionLib.t.sol:TestExecutionLib
[PASS] test_encode_decode(address,uint256,bytes) (runs: 1000, μ: 6483, ~: 6456)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 29.16ms (29.08ms CPU time)

Ran 8 tests for
test/foundry/unit/concrete/execution/TestExecution_ExecuteBatch.t.sol:TestExecution_ExecuteBatch
[PASS] test_ExecuteBatch_ApproveAndTransfer_SeparateOps() (gas: 234774)
[PASS] test_ExecuteBatch_ApproveAndTransfer_SingleOp() (gas: 161572)
[PASS] test_ExecuteBatch_Empty_Success() (gas: 115221)
[PASS] test_ExecuteBatch_Success() (gas: 134503)
[PASS] test_ExecuteBatch_TokenTransfers() (gas: 172251)
[PASS] test_ExecuteBatch_ValueTransfer() (gas: 162909)
[PASS] test_RevertIf_BatchExecutionWithDefaultExecTypeAndOneActionReverts() (gas: 143930)
[PASS] test_RevertIf_BatchExecutionWithUnsupportedExecType() (gas: 119281)
Suite result: ok. 8 passed; 0 failed; 0 skipped; finished in 10.13ms (6.18ms CPU time)

Ran 2 tests for
test/foundry/unit/concrete/execution/TestExecution_ExecuteDelegateCall.t.sol:TestExecution_ExecuteDelegat
eCall
[PASS] test_ExecuteDelegateCall_Success() (gas: 163135)
[PASS] test_TryExecuteDelegateCall_Success() (gas: 163687)
Suite result: ok. 2 passed; 0 failed; 0 skipped; finished in 6.35ms (1.67ms CPU time)

Ran 25 tests for
test/foundry/unit/concrete/modules/TestAccount_WithECDSAValidator.t.sol:TestAccountWithECDSAValidator
[PASS] test_IsInitialized() (gas: 13424)
[PASS] test_IsModuleType() (gas: 10677)
[PASS] test_IsValidSignatureWithSender_Failure() (gas: 57956)
[PASS] test_IsValidSignatureWithSender_Inverted_S_Value_Fails() (gas: 76178)
[PASS] test_IsValidSignatureWithSender_SafeCaller_Success() (gas: 403000)
[PASS] test_IsValidSignatureWithSender_Success() (gas: 45883)
[PASS] test_IsValidSignatureWithSender_ValidSignature() (gas: 44783)
[PASS] test_Name() (gas: 10102)
[PASS] test_OnInstall_Success() (gas: 40180)
[PASS] test_OnUninstall_Success() (gas: 143506)
[PASS] test_RevertWhen_OnInstall_NoOwnerProvided() (gas: 9362)
[PASS] test_RevertWhen_TransferOwnership_ToZeroAddress() (gas: 12356)
[PASS] test_SetUpState() (gas: 18473)
[PASS] test_TransferOwnership_Success() (gas: 21148)
[PASS] test_ValidateUserOp_Failure() (gas: 138748)
[PASS] test_ValidateUserOp_Inverted_S_Value_Fails_because_of_nonce() (gas: 271630)
[PASS] test_ValidateUserOp_Success() (gas: 126505)
[PASS] test_ValidateUserOp_ValidSignature() (gas: 126619)
[PASS] test_ValidateUserOp_toEthSignedMessageHash_Success() (gas: 136902)
[PASS] test_Version() (gas: 9860)
[PASS] test_addSafeSender_Success() (gas: 18176)
[PASS] test_fillSafeSenders_Success() (gas: 149084)
[PASS] test_isSafeSender_Success() (gas: 79205)
[PASS] test_removeSafeSender_Success() (gas: 21474)
[PASS] test_returns_AccountAddress_as_owner_if_owner_not_set_for_Account() (gas: 11301)
Suite result: ok. 25 passed; 0 failed; 0 skipped; finished in 11.95ms (6.92ms CPU time)

Ran 1 test for test/foundry/unit/fuzz/erc7779/TestFuzz_ERC7779Adapter.t.sol:TestFuzz_ERC7779Adapter
[PASS] test_Fuzz_ERC7779Adapter_AddStorageBases(uint256) (runs: 1000, μ: 539032, ~: 217568)
Suite result: ok. 1 passed; 0 failed; 0 skipped; finished in 335.65ms (335.56ms CPU time)

Ran 3 tests for
test/foundry/unit/fuzz/TestAccountFactory_Deployment.t.sol:TestFuzz_AccountFactory_Deployment
[PASS] testFuzz_CreateAccountWithLargeIndex(uint256) (runs: 1000, μ: 219917, ~: 219917)
[PASS] testFuzz_CreateAccountWithRandomData(uint256) (runs: 1000, μ: 219744, ~: 219744)
[PASS] testFuzz_RepeatedAccountCreation(uint256) (runs: 1000, μ: 225033, ~: 225033)
Suite result: ok. 3 passed; 0 failed; 0 skipped; finished in 451.12ms (623.26ms CPU time)
```

```
Ran 13 tests for
test/foundry/unit/concrete/modulemanager/TestModuleManager_FallbackHandler.t.sol:TestModuleManager_Fallba
ckHandler
[PASS] test_ComplexReturnData() (gas: 207584)
[PASS] test_GenericFallbackHandlerTriggered() (gas: 13759)
[PASS] test_GetFallbackHandlerBySelector() (gas: 18157)
[PASS] test_HandleOpsTriggersGenericFallback(bool) (runs: 1000, μ: 149371, ~: 149371)
[PASS] test_ReturnBytes_and_Hook_fallback() (gas: 203433)
[PASS] test_RevertIf_FunctionSelectorAlreadyUsed() (gas: 555858)
[PASS] test_RevertIf_FunctionSelectorNotUsed() (gas: 550187)
[PASS] test_RevertIf_FunctionSelectorNotUsedByThisHandler() (gas: 148163)
[PASS] test_RevertIf_InstallForbiddenOnInstallSelector() (gas: 153655)
[PASS] test_RevertIf_InstallForbiddenOnUninstallSelector() (gas: 153461)
[PASS] test_RevertIf_UninstallNonInstalledFallbackHandler() (gas: 146255)
[PASS] test_UninstallFallbackHandler_Success() (gas: 148139)
[PASS] test_onTokenReceived_Success() (gas: 219690)
Suite result: ok. 13 passed; 0 failed; 0 skipped; finished in 687.92ms (688.96ms CPU time)

Ran 37 test suites in 946.36ms (2.49s CPU time): 296 tests passed, 0 failed, 0 skipped (296 total tests)
```

Code Coverage

The coverage is generated using the command: `forge coverage --no-match-coverage "test|script|Greeter"`. Generally speaking, the coverage is decent but still has room for improvement. Specifically, the team could benefit from increasing coverage for critical contracts such as `ModuleManager`, `ExecutionLib`, `ECDSAValidator`, and `Bootstrap`. We recommend aiming for over 90% branch coverage.

File	% Lines	% Statements	% Branches	% Funcs
src/StartaleSmartAccount.sol	87.65% (142/162)	89.77% (158/176)	66.67% (38/57)	79.17% (19/24)
src/core/AllStorage.sol	100.00% (2/2)	100.00% (1/1)	100.00% (0/0)	100.00% (1/1)
src/core/BaseAccount.sol	93.33% (28/30)	95.45% (21/22)	57.14% (4/7)	87.50% (7/8)
src/core/ERC7779Adapter.sol	100.00% (12/12)	100.00% (9/9)	50.00% (1/2)	100.00% (3/3)
src/core/ExecutionHelper.sol	92.17% (106/115)	91.87% (113/123)	81.82% (27/33)	100.00% (15/15)
src/core/ModuleManager.sol	84.19% (229/272)	78.42% (229/292)	67.74% (63/93)	93.62% (44/47)
src/factory/EOAOnboardingFactory.sol	100.00% (17/17)	100.00% (20/20)	80.00% (4/5)	100.00% (3/3)
src/factory/StartaleAccountFactory.sol	100.00% (12/12)	100.00% (11/11)	71.43% (5/7)	100.00% (3/3)
src/lib/AssociatedArrayLib.sol	28.23% (35/124)	27.78% (30/108)	0.00% (0/7)	28.95% (11/38)
src/lib/BootstrapLib.sol	100.00% (17/17)	93.75% (15/16)	0.00% (0/1)	100.00% (4/4)
src/lib/DataParserLib.sol	43.48% (10/23)	42.86% (9/21)	100.00% (0/0)	50.00% (1/2)
src/lib/EnumerableMap4337.sol	0.00% (0/109)	0.00% (0/105)	0.00% (0/3)	0.00% (0/40)

File	% Lines	% Statements	% Branches	% Funcs
src/lib/EnumerableSet4337.sol	40.96% (34/83)	44.19% (38/86)	80.00% (4/5)	35.71% (10/28)
src/lib/ExecutionLib.sol	90.00% (36/40)	88.24% (30/34)	50.00% (2/4)	100.00% (7/7)
src/lib/Initializable.sol	44.44% (4/9)	42.86% (3/7)	0.00% (0/1)	50.00% (1/2)
src/lib/ModeLib.sol	91.30% (21/23)	92.86% (13/14)	100.00% (0/0)	88.89% (8/9)
src/lib/ModuleTypeLib.sol	0.00% (0/12)	0.00% (0/16)	100.00% (0/0)	0.00% (0/3)
src/lib/NonceLib.sol	50.00% (5/10)	50.00% (3/6)	100.00% (0/0)	50.00% (2/4)
src/lib/ProxyLib.sol	100.00% (10/10)	100.00% (10/10)	75.00% (3/4)	100.00% (2/2)
src/modules/validators/ECDSAValidator.sol	89.06% (57/64)	85.48% (53/62)	50.00% (5/10)	95.00% (19/20)
src/utils/AccountProxy.sol	40.00% (2/5)	50.00% (2/4)	100.00% (0/0)	50.00% (1/2)
src/utils/Bootstrap.sol	55.38% (36/65)	52.78% (38/72)	33.33% (1/3)	72.73% (8/11)
src/utils/Stakeable.sol	100.00% (11/11)	100.00% (7/7)	100.00% (6/6)	100.00% (4/4)
Total	**67.32% (826/1227)**	**66.53% (813/1222)**	**65.73% (163/248)**	**61.79% (173/280)**

Changelog

- 2025-04-25 - Initial report
- 2025-05-12 - Add Missing ERC-165 Support Issue
- 2025-05-15 - Fix review update 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.

Quantstamp