Quantstamp

Aave ERC-6900 Modules

Executive Summary

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

Type	ERC-6900 Module
Timeline	2026-02-09 through 2026-02-12
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Aave Labs ERC6900 Google DocAccount Diagram_ EOA + SCW.pdf
Source Code	• https://github.com/aave/erc6900 #31bbbce
Auditors	• Ruben Koch Senior Auditing Engineer • Hytham Farah Auditing Engineer • Tim Sigl Auditing Engineer

Documentation quality	High
Test quality	High
Total Findings	8 Fixed: 5 Acknowledged: 3
High severity findings	0
Medium severity findings	2 Fixed: 2
Low severity findings	2 Fixed: 2
Undetermined severity findings	0
Informational findings	4 Fixed: 1 Acknowledged: 3

Summary of Findings

In this audit, we reviewed two modules for ERC-6900v0.8-compliant modular accounts. The `MultiSigValidationModule` is a validation module implementing 2-of-2 multi-signature authentication for ERC-6900 modular accounts. The `AddressBookModule functions as both a validation hook module and an execution module, providing recipient allowlisting for asset transfers. This module implements a security boundary that restricts which addresses can receive native tokens, ERC-20, ERC-721, and ERC-1155 assets from the modular account.

Aave intends to use instances of Alchemy MAv2 with a configuration of modules, including the two audited modules here, as a backbone for the Aave app. The modules carefully restrict and guardrail the overall capabilities of actors in the system, aiming to protect the user in case of private key compromise, yet at the same time restricting the Aave backend in their capabilities. This results in a remarkably fair separation of concerns that keeps the user in control of their funds, yet vastly limits the dangers of a private key compromise.

A few findings have been identified, mainly around ERC-6900-related misconfigurations (AAV-1, AAV-5, AAV-7) and also a circumvention around the enforcement of allowed recipients from the `AddressBookModule` (AAV-2).

Overall the test suite is in great shape, but a few of the findings highlight that the test suite relies to much on `prank()` -calls as the account that directly invoke certain functions instead of being routed through `execute()` with the associated hooks.

Fix-Review Update 2026-02-13:
All findings are successfully fixed or reasonably acknowledged. We value the team's security minded approach and their clear commitment to hardening the system.

Fix-Review Update 2026-04-17:
Aave Labs has requested a follow-up review of PR 10 and PR 11.

PR 10 adds an additional layer of protection that withdrawals remain only self-beneficiary, a check that was previously only enforced on a protocol level. Overall, the change is reasonable as an additional wallet-side security layer; only a minor finding has been identified (AAV-8) that has been fully fixed.

PR 11 adds comprehensive e2e integration tests, properly testing the codebase as intended in operation: Modules installed on a Modular Account V2, with testing UserOperations routed through the EntryPoint contract. We applaud this additional effort and up the Test quality-metric from "Medium" to "High".

ID	DESCRIPTION	SEVERITY	STATUS
AAV-1	`addAllowedRecipients()` **Function Execution Manifest Skips All Validation, Allowing Anyone to Expand Any Account's Allowlist**	Medium	Fixed
AAV-2	`AddressBookModule` **Does Not Check Recipient Parameter in** `StableVault.deposit()`**, Allowing Deposits to an Unauthorized User**	Medium	Fixed
AAV-3	`MultiSigValidationModule` **Breaks ERC-4337 Validation Requirements**	Low	Fixed
AAV-4	**Execution Manifest Configuration Prevents MultiSig from Adding Allowed Recipients**	Informational	Fixed
AAV-5	**Module Uninstallation May Leave Residual Data, Affecting Re-Installations**	Informational	Acknowledged
AAV-6	`RecipientAddressLib` **Does Not Cover All ERC-1155 and ERC-721 Functions**	Informational	Acknowledged
AAV-7	**Reverting Validation Hook Blocks Possibility of Intended Invocation of Execution Module**	Informational	Acknowledged
AAV-8	`SV_EXECUTE_WITHDRAWAL_MIN_LEN` **Is 32 Bytes Too Low for** `executeWithdrawal()` **Calldata**	Low	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

Files Included

Repo: https://github.com/aave/erc6900(31bbbce8488c41172ab10dd63bf137a990d57c06) Files: src/*

Operational Considerations

- **Precise account setup according to specs:** It is very important that account instances are set up exactly as intended (See section "Key Actors and Their Capabilities"), as misconfigurations could circumvent the guardrails put in place
- **Upgradeable/Extendable:** The underlying MAv2 contract is upgradeable. Upgrade functionality is only accessible to the Multisig validation module. We assume that all contract upgrades, as well as additional installations of modules to the existing account, go through a planned, audited process.
- **AddressBook is account-wide:** `_allowedRecipients` is keyed by `msg.sender` (the account), not `entityId`. Recipients added for the user key also apply to the backend session key.

Key Actors And Their Capabilities

After deployment, the account uses three validation entities with distinct scopes to balance safety and usability. A global 2-of-2 multisig (user + backend cosigner) acts as the only unrestricted authority and is expected to manage configuration changes, module updates, and allowlist/address-book updates. Two single-sig entities (user EOA and backend session key) are intentionally restricted by the `Allowlist` and `AddressBook` hooks and can only perform tightly scoped actions via `execute` / `executeBatch`. A temporary bootstrap single-sig entity is installed during deployment and removed immediately after the multisig is installed. `AddressBook` is account-wide, and the backend session key is explicitly prevented from initiating withdrawals (it can only execute them).

Entities

There are separate entities with different degrees of priviliges and concerns:

Multisig Authority (User EOA + Backend Cosigner)

The global 2-of-2 multisig is the sole unrestricted validator that requires a signature from both the user, as well as Aave Labs' backend. It can validate any function including `installValidation()`, `uninstallValidation()`, and account upgrades.

Responsibility

- Manage account configuration and module lifecycle
- Approve address-book additions/removals and allowlist updates
- Serve as the only unrestricted control plane for the account

Trust Assumption

- If one of the two parties is compromised, the other will refrain from signing malicious UserOperations
- This remains the only global validator module

User EOA (Restricted Single-Sig)

The user's EOA is restricted by what contracts it can interact with, as well as which functions it can invoke via the `Allowlist` + `AddressBook` hooks for approvals, transfers and withdrawal requests. It is limited to very restricted access to the `execute()` and `executeBatch()` functions and cannot modify validations, perform upgrades, or validate signatures (ERC-1271/permit2).

Responsibility

- Perform user-initiated actions within allowlist/address-book constraints
- Request withdrawals when needed (subject to allowlist rules)

Trust Assumption

- The user key may be compromised; hooks are relied upon to limit damage by restricting interactions to whitelisted contracts and functions. **With the outlined allowlist/address-book constraints, private key compromise of the user should not result in loss of funds**, as loosening the constraints requires the signature from Aave Labs backend.

Backend Session Key (Most Restricted Single-Sig)

A backend session key used for automation. Shares `Allowlist` + `AddressBook` constraints with User EOA-entity, but further restricted (cannot request withdrawals, only finalize ones already requested by the user). Limited to `execute()` and `executeBatch()` only. Cannot modify validations, perform upgrades, or validate signatures (ERC-1271/permit2). Access to heavily restricted ERC-20 interactions for e.g. automatic deposits into whitelisted vaults.

Responsibility

- Enables quality-of-life features, tightly guard-railed the validation hooks
- Execute routine actions within strict allowlist/address-book limits
- Execute withdrawals only after a user-authorized request

Trust Assumption

- The same as with User EOA-entity.

Findings

AAV-1

`addAllowedRecipients()` Function Execution Manifest Skips All Validation, Allowing Anyone to Expand Any Account's Allowlist

Medium · Fixed

> **Update**
> Marked as "Fixed" by the client.
> Addressed in: `2a2e9a9e9808ed2a58afc959899a8aaf32019ce8`.

File(s) affected: `src/v0.8/modules/addressbook/AddressBookModule.sol`

Description: The `executionManifest()` function in the `AddressBookModule` contract configures the `addAllowedRecipients()` function with `skipRuntimeValidation: true`.

With `skipRuntimeValidation: true`, the ERC-6900 validation skips all validation for runtime calls to this selector. Any external address can call `account.addAllowedRecipients([attackerAddress])`, and the account forwards the call to the module without checking authorization. Since `msg.sender` in the module is the account itself, the attacker's address is added to the account's allowlist.

Meanwhile, `removeAllowedRecipients()` correctly configures validations with `skipRuntimeValidation: false`.

Exploit Scenario:

1. Account has the `AddressBookModule` execution module installed.
2. An attacker calls `account.addAllowedRecipients([attackerAddress])`. No validation is required. The attacker's address is added to the account's allowlist.
3. If the account is ever compromised, the attacker can now transfer all account assets to `attackerAddress`, as the address book hook will not block it.

The address book module's entire security guarantee to restrict the set of valid recipients is defeated.

Recommendation: Change the manifest configuration for `addAllowedRecipients()` to `skipRuntimeValidation: false`.

AAV-2

`AddressBookModule` Does Not Check Recipient Parameter in `StableVault.deposit()`, Allowing Deposits to an Unauthorized User

Medium · Fixed

> **Update**
> Marked as "Fixed" by the client.
> Addressed in: `0968da6ff3a80a62ba625ea18a6905a6f37000be`.
> The client provided the following explanation:
>
> > After an initial commit to check the beneficiary of a vault deposit, a subsequent change was made to prevent the beneficiary from being address(0). The complete fix is reflected as of commit `0968da6ff3a80a62ba625ea18a6905a6f37000be`.

File(s) affected: `src/v0.8/modules/addressbook/AddressBookModule.sol`, `src/libs/RecipientAddressLib.sol`

Description: `StableVault.deposit(address user, address asset, uint256 amount)` lets the caller set `user` to any address; the `AddressBookModule` does not check this parameter (it only validates recipients for known ERC-20/721/1155 selectors). StableVault's `requestWithdrawal()` and `executeWithdrawal()` already enforce `require(user == msg.sender, OnlyUser())`, so the account can only act for itself there; `deposit()` has no such check. A compromised session key can therefore direct deposit credits to an unauthorized beneficiary. The `AllowlistModule` restricts target and selector only, so it does not close this gap.

Such an attack may be prevented by the backend's strict requirements for a second signature to pass validation, however on-chain validation by the `AddressBookModule` is the more secure option.

Exploit Scenario:

1. User EOA is compromised. The attacker submits a UserOp: `execute(StableVault, 0, deposit(attackerBeneficiary, USDC, amount))`.
2. `AddressBookModule`: `value == 0`, and `deposit()` is not an ERC-20/721/1155 method, so it early-returns with no check.

3. `AllowlistModule` : target StableVault is allowed, selector `deposit()` is allowed, so the call passes.
4. The StableVault credits `attackerBeneficiary` with the deposit. The address book does not restrict where deposit credits flow.
5. The attacker can withdraw the deposit to their own address.

Recommendation: Implement one of the following options:

1. Add `require(user == msg.sender, OnlyUser())` to `StableVault.deposit()` so the account can only deposit for itself.
2. In the `AddressBookModule` (or `RecipientAddressLib`), implement receiver extraction for the `StableVault.deposit()` function and treat the first address parameter as the recipient; allow the account itself as a valid recipient in the address book.

AAV-3

`MultiSigValidationModule` Breaks ERC-4337 Validation Requirements

• Low Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `c1f8182932f711966c18cc243ccb5ed3ba0d2cd8` and `83a72182ae223badc42a02e29f0f652ebfe1e34b`.
The client provided the following explanation:

> We split the fix up into commit `c1f8182932f711966c18cc243ccb5ed3ba0d2cd8` and 83a72182ae223badc42a02e29f0f652ebfe1e34b.

File(s) affected: `src/v0.8/validation/multisig/MultiSigValidationModule.sol`

Description: ERC-4337 defines two relevant requirements for validation:

1. The account MUST validate that the signature is a valid signature of the `userOpHash`, and SHOULD return `SIG_VALIDATION_FAILED` (1) without reverting on signature mismatch. Any other error MUST revert.
2. The validator SHOULD not return early when returning `SIG_VALIDATION_FAILED` (1). Instead, it SHOULD complete the normal flow to enable gas estimation for the validation function.

Parts of the implementation are not fully compliant to these specs:

1) In the `validateUserOp()` function, the module returns `SIG_VALIDATION_FAILED` for all failure cases, including malformed or invalid input. Per the spec, only a signature mismatch (valid ECDSA recovery but signers do not match the stored set) should return the failure code; any other error MUST revert. This means:

1. Wrong signature length in `validateUserOp()` should revert.
2. In `_checkSigs()`, use `ECDSA.recover()` to revert on failures and return `SIG_VALIDATION_FAILED` in the case both recovered signers are the same or they are they are not equal to the stored signers.

2) In the ERC-4337 flow, when validation fails the module returns early with `SIG_VALIDATION_FAILED` instead of completing the normal flow. That can break gas estimation and tooling that expect the full validation path to run before returning `SIG_VALIDATION_FAILED`.

Recommendation: Adjust the flow regarding reverts and `SIG_VALIDATION_FAILED` as outlined above. Furthermore, store the validation result in a boolean variable where currently `SIG_VALIDATION_FAILED` is returned. Only at the end of the validation function, check the boolean variable and return `SIG_VALIDATION_FAILED` or `SIG_VALIDATION_SUCCEEDED`.

AAV-4

Execution Manifest Configuration Prevents MultiSig from Adding Allowed Recipients

• Informational Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `2a2e9a9e9808ed2a58afc959899a8aaf32019ce8`.
The client provided the following explanation:

> This should be moved to an informational finding because the AddressBook can be updated through a UO flow which invoked execute/executeBatch.

Update

The severity of the finding was downgraded after internal discussion.

File(s) affected: `src/v0.8/modules/addressbook/AddressBookModule.sol`

Description: The `AddressBookModule` execution manifest sets `allowGlobalValidation: false` for `addAllowedRecipients()` and `allowGlobalValidation: true` for `removeAllowedRecipients()`.

For global validation modules to add or remove allowed recipients, global validation must be enabled for both execution functions. The account allows a call from a global validator only when the execution function has `allowGlobalValidation: true`. With `allowGlobalValidation: false` on `addAllowedRecipients()`, a global validation module cannot authorize UserOps that call `addAllowedRecipients()`.

The result is an asymmetry: `removeAllowedRecipients()` is callable by a global validation module but `addAllowedRecipients()` is not, even though the audit primer states that the `AddressBookModule` should only be updatable by a global validation module — both operations need global validation access. Therefore, currently, the list of allowed recipients can only be initiated on module installation and then not further expanded.

Recommendation: Set `allowGlobalValidation: true` for `addAllowedRecipients()`. Keep both add and remove operations consistent so a global validator module can manage the address book.

AAV-5
Module Uninstallation May Leave Residual Data, Affecting Re-Installations

Informational · Acknowledged

Update

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

> We plan to always pass data for an uninstallation which would contain at least the entityId. It so happens that the AddressBook module ignores the data param because there is no need to decode the entityId if the AddressBook state is mapped an the account level as opposed to an entityId level.

File(s) affected: `src/v0.8/validation/multisig/MultiSigValidationModule.sol`, `src/v0.8/modules/addressbook/AddressBookModule.sol`

Description: In the Alchemy Modular Account V2 implementation, the account calls a module's `onUninstall()` only when `uninstallData` (or the hook's uninstall data) has `length > 0` (`ModuleInstallCommonsLib.onUninstall()`). Empty uninstall data therefore updates account state but never runs the module's cleanup, so module storage can stay populated.

- `MultiSigValidationModule.onUninstall()` decodes `entityId` from data to clear signers, so callers are likely to pass data. If not data is passed, the signers are not cleared and and re-installation would fail with `AlreadyInitialized`, resulting from that `entityId` not being able to be reused.
- `AddressBookModule.onUninstall()` does not use `data` at all, so callers have no reason to pass it. Uninstalling with empty data is likely; `onUninstall()` is then skipped and `_allowedRecipients` is never cleared. A later reinstall sees the previous allowed recipients still in place next to the new allowed recipients. This however might also be useful, as the `AddressBookModule` only maintains lists on a per-account level, not per-`entityId` level, so a full removal could affect other module installations that still use this hook.

Furthermore, the `AddressBookModule` is implemented to intentionally skip the removal during invoked `onUninstall()` if the `recipients` list is sufficiently large. In that case, a `AllowedAddressesNotRemoved` is emitted that can be tracked. However, if the callback is not invoked all together, then the event will also not be emitted to indicate the residual data.

This is a known and documented aspect of the Modular Account V2 implementation, see ALC-15 in our related audit report, and also in line with the ERC-6900 standard, but we wanted to raise awareness for this in this context as well.

Recommendation: Assure that state is cleared on module uninstallation where needed. Document this behavior and the recovery path (e.g. call `module.onUninstall(abi.encode(...))` via `execute()`.

AAV-6
`RecipientAddressLib` Does Not Cover All ERC-1155 and ERC-721 Functions

Informational · Acknowledged

Update

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

> We do not plan to use any token methods which take in a `bytes` param due to the risk of a griefing attack whereby dead bytes are appended to the calldata.

File(s) affected: `src/libs/RecipientAddressLib.sol`

Description: `RecipientAddressLib` does not recognize some standard ERC-1155 and ERC-721 transfer selectors. This unnecessarily restricts users from using these standard methods. Calls to them pass through `_verifyAllowedTargetOrRecipient()` without recipient check when `value == 0`. If the `AllowlistModule` is ever reconfigured to allow all ERC-721/ERC-1155, `AddressBookModule` protection would not apply to these methods.

Selectors not included:

- ERC-1155 `safeBatchTransferFrom(address,address,uint256[],uint256[],bytes)`.
- ERC-721 `safeTransferFrom(address,address,uint256,bytes)` and `safeTransferFrom(address,address,uint256,bytes)`

Recommendation: Implement the list above in `containsERC1155Methods()` / `getERC1155TokenRecipient()` and `containsERC721Methods()` / `getERC721TokenRecipient()`. Or document that these methods are intentionally uncovered and that the `AllowlistModule` must block them.

AAV-7

Reverting Validation Hook Blocks Possibility of Intended Invocation of Execution Module

Informational · Acknowledged

Update

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

> If we need to access a function that is not execute/executeBatch, we can use the MultiSig validation which is not subject to the AddressBookModule hooks.

File(s) affected: `src/v0.8/modules/addressbook/AddressBookModule.sol`

Description: The `verifyAllowedTargetOrRecipient()` function, intended to be invoked as a validation hook for the two validation modules installed on the account, reverts in case the account action is not `execute()` or `executeBatch()`. This has the unfortunate side effect of blocking calls invoking state changes to the `AddressBookModule` via `account.addAllowedRecipients()` and `account.removeAllowedRecipients()`, which is the intended flow of execution functions installed on to the account via execution modules. Technically, these functions can still be invoked via `execute(AddressBookModule, 0, addAllowedRecipients(addressArray)`, which is e.g. how state changes are done in the `AllowListModule`, but that defies the purpose of exposing the module as an execution module.

The impact is however non-existent in the used signer setup in this particular case, as the validation module that has access to the state changing functionality on the `AddressBookModule` does not have these hooks in place.

Recommendation: Remove the revert at the end of the function in `verifyAllowedTargetOrRecipient()`.

AAV-8

`SV_EXECUTE_WITHDRAWAL_MIN_LEN` Is 32 Bytes Too Low for `executeWithdrawal()` Calldata

Low · Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `515346615fd3120e40420ec8423f5a493b303dfe`.

File(s) affected: `src/libs/RecipientAddressLib.sol`

Description: The constant `SV_EXECUTE_WITHDRAWAL_MIN_LEN` is set to `164`, but the minimum valid ABI encoded calldata for `executeWithdrawal(address,address,uint256,uint256,bytes)` is 196 bytes. The breakdown is: 4 (selector) + 32 (user) + 32 (assetOut) + 32 (minAmountOut) + 32 (iouAmountRay) + 32 (bytes offset) + 32 (bytes length) = 196.

With `SV_EXECUTE_WITHDRAWAL_MIN_LEN = 164`, incorrect calldata between 164 and 195 bytes matching the `executeWithdrawal()` selector would pass the length gate in `getVaultWithdrawalRecipient()` and proceed to parse the user address. While the user address read at offset 36 is still memory safe at that length, the data itself is not valid ABI encoding for the `executeWithdrawal()` function and would never correspond to a legitimate call.

Recommendation: Change the constant to 196 to match the correct minimum ABI encoding. Additionally, the malformed calldata test (`test_verifyAllowedTargetOrRecipient_MalformedExecuteWithdrawal_Reverts()`) should construct data at exactly `MIN_LEN - 1` bytes rather than a short 20 byte payload, so that boundary off by one errors in the constant are actually caught.

Auditor Suggestions

S1 Guard `_addSigner()` Against Overwrite when Both Signer Slots Are Occupied

Fixed

> **Update**
> Marked as "Fixed" by the client.
> Addressed in: `2ab4493aed7e077188e1977c4dfaf74719de9d86`.

File(s) affected: `src/v0.8/validation/multisig/MultiSigValidationModule.sol`

Description: `_addSigner()` fills the first empty slot or else assigns to `signer2`. When both `signer1` and `signer2` are already set and a new signer is passed, it overwrites `signer2` without reverting. This path is not currently reachable via `onInstall()` or `transferSigner()`, but the function is not defensive; any future caller that invokes `_addSigner()` without first removing a signer would silently overwrite `signer2`.

Recommendation: Before assigning, check that at least one slot is free; if both `signer1` and `signer2` are non-zero, revert.

S2

`MultiSigValidationModule.onUninstall()` Emits `SignersRemoved` Even when Entity Was Never Initialized

Fixed

> **Update**
> Marked as "Fixed" by the client.
> Addressed in: `6f12f6e9359ffdaab9e46c649cf2aa9611c15560`.

File(s) affected: `src/v0.8/validation/multisig/MultiSigValidationModule.sol`

Description: The `onUninstall()` function always clears signers and emits `SignersRemoved` for the provided `entityId` without checking whether that `(account, entityId)` pair was ever initialized via `onInstall()`. If it is called for an `entityId` that never had signers set, off-chain indexers may record a phantom uninstallation for an entity that was never active.

Recommendation: Add a check that at least one signer is set before allowing uninstallation; revert when the entity was never initialized so indexers do not see a phantom uninstall:

```
Signers storage signers = signersInEntity[entityId][msg.sender];
if (signers.signer1 == address(0) && signers.signer2 == address(0)) {
    revert NotInitialized(entityId, msg.sender);
}
```

S3 Inaccurate Comment About Behavior of `AddressBookModule`

Fixed

> **Update**
> Marked as "Fixed" by the client.
> Addressed in: `820a3f3bbfe5fbd14d07ef10f353bbe647b8fa87`.
> The client provided the following explanation:
>
> > Indirectly was addressed with commit 820a3f3bbfe5fbd14d07ef10f353bbe647b8fa87.

File(s) affected: `src/v0.8/modules/addressbook/AddressBookModule.sol`

Description: In `AddressBookModule.sol#L43`, it is stated that for token transfers "If the recipient's address is not specified (== address(0)) within the calldata, allow bypass.". This is not the reality of the code, which we argue is good, as it would essentially enable the user to always burn their funds, even with an empty `recipient` list.

Recommendation: Remove the misleading comment.

Definitions

- **High severity** – High-severity issues usually put a large number of users' sensitive information at risk, or are reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.

- **Medium severity** – Medium-severity issues tend to put a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or are reasonably likely to lead to moderate financial impact.
- **Low severity** – The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances.
- **Informational** – The issue does not pose an immediate risk, but is relevant to security best practices or Defence in Depth.
- **Undetermined** – The impact of the issue is uncertain.
- **Fixed** – Adjusted program implementation, requirements or constraints to eliminate the risk.
- **Mitigated** – Implemented actions to minimize the impact or likelihood of the risk.
- **Acknowledged** – The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).

Appendix

File Signatures

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

Files

- `347...664 ./src/v0.8/BaseModule.sol`
- `9ca...9b1 ./src/v0.8/validation/multisig/IMultiSigValidationModule.sol`
- `2a6...fac ./src/v0.8/validation/multisig/MultiSigValidationModule.sol`
- `b9f...3e2 ./src/v0.8/modules/addressbook/AddressBookModule.sol`
- `fe7...c09 ./src/v0.8/modules/addressbook/IAddressBookModule.sol`
- `57c...d7b ./src/common/Structs.sol`
- `8e5...8c4 ./src/common/Constants.sol`
- `fed...cef ./src/libs/CastLib.sol`
- `916...00e ./src/libs/RecipientAddressLib.sol`

Tests

- `a5b...b8c ./test/v0.8/validation/MultiSigValidationModule.t.sol`
- `235...257 ./test/v0.8/modules/AddressBookModule.t.sol`
- `eef...54a ./test/util/TestUtils.sol`
- `6d6...341 ./test/util/TestLiquidityPool.sol`
- `a25...0c7 ./test/util/TestERC721.sol`
- `d42...7b5 ./test/util/TestERC1155.sol`

Test Suite Results

Tests were obtained with `forge test`.

Update Fix-Review:

The test suite was expanded slightly to 117 tests compared to the 109 tests before.

Update Followup-Review:

The test suite was significantly expanded to 155 passing tests. The added integration tests test the modules in very realistic production settings.

```
Ran 29 tests for test/v0.8/validation/MultiSigValidationModule.t.sol:MultiSigValidationModuleTest
[PASS] test_EmptySigners_Getters() (gas: 17041)
[PASS] test_getSigners_NoInstallation() (gas: 17525)
[PASS] test_moduleId() (gas: 11156)
[PASS] test_onInstall_CannotReinstallWithCompletelyDifferentSigners() (gas: 78661)
[PASS] test_onInstall_RevertsOnBadLength() (gas: 14815)
[PASS] test_onInstall_RevertsOnDuplicateSigner() (gas: 41293)
[PASS] test_onInstall_RevertsOnZeroSigner() (gas: 49908)
[PASS] test_onInstall_SetsTwoSigners_AndGetters() (gas: 71856)
```

```
[PASS] test_onUninstall_ClearsSigners() (gas: 56536)
[PASS] test_onUninstall_RevertsWhenNotInitialized() (gas: 15362)
[PASS] test_supportsInterface() (gas: 12609)
[PASS] test_transferSigner_ReplacesOneSigner_AndValidatesNewPair() (gas: 113179)
[PASS] test_transferSigner_RevertIfPreviousNotFound() (gas: 71560)
[PASS] test_transferSigner_UnauthorizedCaller() (gas: 74447)
[PASS] test_validateRuntime_AlwaysRevertsNotAuthorized() (gas: 9832)
[PASS] test_validateSignature_EmptyDigest() (gas: 84148)
[PASS] test_validateSignature_Fails_ComprehensiveLengthChecks() (gas: 94545)
[PASS] test_validateSignature_Fails_WrongLen_Duplicate_Unknown() (gas: 100468)
[PASS] test_validateSignature_ReplayAttackPrevention() (gas: 149807)
[PASS] test_validateSignature_Success() (gas: 536)
[PASS] test_validateSignature_Success_Runtime() (gas: 83176)
[PASS] test_validateUserOp_EntityIdIsolation() (gas: 152924)
[PASS] test_validateUserOp_Fails_DuplicateSigner() (gas: 80366)
[PASS] test_validateUserOp_Fails_UnknownSigner() (gas: 83079)
[PASS] test_validateUserOp_Fails_WrongLength() (gas: 72773)
[PASS] test_validateUserOp_Fails_WrongSignatureLength() (gas: 88244)
[PASS] test_validateUserOp_SignatureOrderIndependence() (gas: 97122)
[PASS] test_validateUserOp_Success_TwoDistinctEOA() (gas: 83501)
[PASS] test_validateUserOp_ZeroAddressAttack() (gas: 87536)
Suite result: ok. 29 passed; 0 failed; 0 skipped; finished in 26.47ms (28.19ms CPU time)

Ran 105 tests for test/v0.8/modules/AddressBookModule.t.sol:AddressBookModuleTest
[PASS] test_addAllowedRecipients_DuplicateRecipient() (gas: 66104)
[PASS] test_addAllowedRecipients_EmptyArray() (gas: 22715)
[PASS] test_addAllowedRecipients_Success() (gas: 125784)
[PASS] test_addZeroAddressRecipient() (gas: 14289)
[PASS] test_executionManifest() (gas: 18757)
[PASS] test_fuzz_addAllowedRecipients(address[]) (runs: 1024, μ: 1463972, ~: 1169035)
[PASS] test_fuzz_verifyAllowedTargetOrRecipient_NativeTransfer(address,uint256) (runs: 1024, μ: 64259, ~:
64259)
[PASS] test_getAllowedRecipients_MultipleAccounts() (gas: 125040)
[PASS] test_moduleId() (gas: 12727)
[PASS] test_onInstall_WithRecipients() (gas: 96743)
[PASS] test_onInstall_WithoutRecipients() (gas: 18429)
[PASS] test_onUninstall_FewRecipients() (gas: 101494)
[PASS] test_preRuntimeValidationHook_Success() (gas: 73079)
[PASS] test_preSignatureValidationHook_Unsupported() (gas: 12086)
[PASS] test_preUserOpValidationHook_ExecuteBatch_EmptyBatch() (gas: 15703)
[PASS] test_preUserOpValidationHook_ExecuteBatch_MixedTokenAndNative() (gas: 117857)
[PASS] test_preUserOpValidationHook_ExecuteBatch_MultipleTokenTypesZeroRecipient() (gas: 98781)
[PASS] test_preUserOpValidationHook_ExecuteBatch_NativeTransferToZeroAddress() (gas: 76259)
[PASS] test_preUserOpValidationHook_ExecuteBatch_NativeTransfers() (gas: 100567)
[PASS] test_preUserOpValidationHook_ExecuteBatch_NonTokenMethod() (gas: 83528)
[PASS] test_preUserOpValidationHook_ExecuteBatch_Success() (gas: 168892)
[PASS] test_preUserOpValidationHook_ExecuteBatch_Success_TokenCall_OtherCalls() (gas: 111023)
[PASS] test_preUserOpValidationHook_ExecuteBatch_TokenTransferZeroRecipient() (gas: 82912)
[PASS] test_preUserOpValidationHook_ExecuteBatch_UnauthorizedRecipient() (gas: 87359)
[PASS] test_preUserOpValidationHook_Success() (gas: 75846)
[PASS] test_preUserOpValidationHook_UnexpectedDataPassed() (gas: 13720)
[PASS] test_preUserOpValidationHook_VaultDeposit_NonSelfBeneficiary_Unauthorized() (gas: 24232)
[PASS] test_preUserOpValidationHook_VaultDeposit_SelfBeneficiary_Success() (gas: 22123)
[PASS] test_preUserOpValidationHook_VaultExecuteWithdrawal_NonSelfBeneficiary_Unauthorized() (gas: 26705)
[PASS] test_preUserOpValidationHook_VaultExecuteWithdrawal_SelfBeneficiary_Success() (gas: 24209)
[PASS] test_preUserOpValidationHook_VaultRequestWithdrawal_NonSelfBeneficiary_Unauthorized() (gas: 24318)
[PASS] test_preUserOpValidationHook_VaultRequestWithdrawal_SelfBeneficiary_Success() (gas: 19035)
[PASS] test_removeAllowedRecipients_NonExistentRecipient() (gas: 20013)
[PASS] test_removeAllowedRecipients_Success() (gas: 80752)
[PASS] test_supportsInterface() (gas: 15293)
[PASS] test_verifyAllowedTargetOrRecipient_BatchExecute_Allowed() (gas: 106065)
[PASS] test_verifyAllowedTargetOrRecipient_BatchExecute_OneUnauthorized() (gas: 84447)
[PASS] test_verifyAllowedTargetOrRecipient_ERC1155SafeBatchTransferFrom_Unchecked() (gas: 22177)
[PASS] test_verifyAllowedTargetOrRecipient_ERC1155SafeBatchTransferFrom_Unchecked_Batch() (gas: 24356)
[PASS] test_verifyAllowedTargetOrRecipient_ERC1155SafeTransferFrom_ZeroRecipient() (gas: 24363)
[PASS] test_verifyAllowedTargetOrRecipient_ERC1155SafeTransferFrom_ZeroRecipient_Batch() (gas: 27400)
[PASS] test_verifyAllowedTargetOrRecipient_ERC1155SetApprovalForAll_ZeroRecipient() (gas: 22928)
[PASS] test_verifyAllowedTargetOrRecipient_ERC1155SetApprovalForAll_ZeroRecipient_Batch() (gas: 25355)
[PASS] test_verifyAllowedTargetOrRecipient_ERC1155_Methods_Allowed() (gas: 83124)
[PASS] test_verifyAllowedTargetOrRecipient_ERC20Approve_ZeroRecipient() (gas: 21684)
[PASS] test_verifyAllowedTargetOrRecipient_ERC20Approve_ZeroRecipient_Batch() (gas: 26448)
[PASS] test_verifyAllowedTargetOrRecipient_ERC20TransferFrom_ZeroRecipient() (gas: 22636)
```

```
[PASS] test_verifyAllowedTargetOrRecipient_ERC20TransferFrom_ZeroRecipient_Batch() (gas: 25081)
[PASS] test_verifyAllowedTargetOrRecipient_ERC20Transfer_ZeroRecipient() (gas: 21870)
[PASS] test_verifyAllowedTargetOrRecipient_ERC20Transfer_ZeroRecipient_Batch() (gas: 25996)
[PASS] test_verifyAllowedTargetOrRecipient_ERC20_Methods_Allowed() (gas: 98700)
[PASS] test_verifyAllowedTargetOrRecipient_ERC20_Methods_Unauthorized() (gas: 75814)
[PASS] test_verifyAllowedTargetOrRecipient_ERC20_Methods_Unchecked() (gas: 19391)
[PASS] test_verifyAllowedTargetOrRecipient_ERC721Approve_ZeroRecipient() (gas: 24293)
[PASS] test_verifyAllowedTargetOrRecipient_ERC721Approve_ZeroRecipient_Batch() (gas: 24864)
[PASS] test_verifyAllowedTargetOrRecipient_ERC721SafeTransferFrom_ZeroRecipient() (gas: 23400)
[PASS] test_verifyAllowedTargetOrRecipient_ERC721SafeTransferFrom_ZeroRecipient_Batch() (gas: 26760)
[PASS] test_verifyAllowedTargetOrRecipient_ERC721SetApprovalForAll_ZeroRecipient() (gas: 23555)
[PASS] test_verifyAllowedTargetOrRecipient_ERC721SetApprovalForAll_ZeroRecipient_Batch() (gas: 24937)
[PASS] test_verifyAllowedTargetOrRecipient_ERC721Transfer_ZeroRecipient() (gas: 24064)
[PASS] test_verifyAllowedTargetOrRecipient_ERC721Transfer_ZeroRecipient_Batch() (gas: 24652)
[PASS] test_verifyAllowedTargetOrRecipient_ERC721_Methods_Allowed() (gas: 93950)
[PASS] test_verifyAllowedTargetOrRecipient_ExtraLongERC20Transfer_DirectCall() (gas: 86151)
[PASS] test_verifyAllowedTargetOrRecipient_MalformedERC1155SafeTransferFrom_Reverts() (gas: 23268)
[PASS] test_verifyAllowedTargetOrRecipient_MalformedERC1155SetApprovalForAll_Reverts() (gas: 23016)
[PASS] test_verifyAllowedTargetOrRecipient_MalformedERC20Approve_Reverts() (gas: 23097)
[PASS] test_verifyAllowedTargetOrRecipient_MalformedERC20DecreaseAllowance_Reverts() (gas: 22594)
[PASS] test_verifyAllowedTargetOrRecipient_MalformedERC20IncreaseAllowance_Reverts() (gas: 23743)
[PASS] test_verifyAllowedTargetOrRecipient_MalformedERC20TransferFrom_Reverts() (gas: 21863)
[PASS] test_verifyAllowedTargetOrRecipient_MalformedERC20Transfer_DirectCall() (gas: 84140)
[PASS] test_verifyAllowedTargetOrRecipient_MalformedERC20Transfer_Reverts() (gas: 22873)
[PASS] test_verifyAllowedTargetOrRecipient_MalformedERC721Approve_Reverts() (gas: 22393)
[PASS] test_verifyAllowedTargetOrRecipient_MalformedERC721SafeTransferFrom_Reverts() (gas: 23374)
[PASS] test_verifyAllowedTargetOrRecipient_MalformedERC721SetApprovalForAll_Reverts() (gas: 23742)
[PASS] test_verifyAllowedTargetOrRecipient_MalformedERC721TransferFrom_Reverts() (gas: 22424)
[PASS] test_verifyAllowedTargetOrRecipient_MalformedExecuteWithdrawal_Reverts() (gas: 20096)
[PASS] test_verifyAllowedTargetOrRecipient_MalformedRequestWithdrawal_Reverts() (gas: 19425)
[PASS] test_verifyAllowedTargetOrRecipient_MalformedVaultDeposit_Reverts() (gas: 18917)
[PASS] test_verifyAllowedTargetOrRecipient_NativeTransferBatchToZeroAddress() (gas: 74027)
[PASS] test_verifyAllowedTargetOrRecipient_NativeTransferBatchWithData() (gas: 73483)
[PASS] test_verifyAllowedTargetOrRecipient_NativeTransferBatch_Allowed() (gas: 99430)
[PASS] test_verifyAllowedTargetOrRecipient_NativeTransferBatch_Unauthorized() (gas: 78387)
[PASS] test_verifyAllowedTargetOrRecipient_NativeTransferToZeroAddress() (gas: 14700)
[PASS] test_verifyAllowedTargetOrRecipient_NativeTransferWithData() (gas: 68233)
[PASS] test_verifyAllowedTargetOrRecipient_NativeTransfer_Allowed() (gas: 67430)
[PASS] test_verifyAllowedTargetOrRecipient_NativeTransfer_Unauthorized() (gas: 20316)
[PASS] test_verifyAllowedTargetOrRecipient_NonTokenMethod() (gas: 70690)
[PASS] test_verifyAllowedTargetOrRecipient_TokenContractWithoutCode() (gas: 23825)
[PASS] test_verifyAllowedTargetOrRecipient_UnsupportedSelector() (gas: 16411)
[PASS] test_verifyAllowedTargetOrRecipient_VaultDeposit_Batch_Allowed() (gas: 81081)
[PASS] test_verifyAllowedTargetOrRecipient_VaultDeposit_Batch_OneUnauthorized() (gas: 31645)
[PASS] test_verifyAllowedTargetOrRecipient_VaultDeposit_NonSelfBeneficiary_Unauthorized(address) (runs:
1024, μ: 21106, ~: 21106)
[PASS] test_verifyAllowedTargetOrRecipient_VaultDeposit_SelfBeneficiary_Allowed() (gas: 20317)
[PASS] test_verifyAllowedTargetOrRecipient_VaultDeposit_ZeroAddressBeneficiary_Unauthorized() (gas:
21849)
[PASS] test_verifyAllowedTargetOrRecipient_VaultExecuteWithdrawal_ArbitraryBytesPayload_Allowed() (gas:
23764)
[PASS] test_verifyAllowedTargetOrRecipient_VaultExecuteWithdrawal_EmptyBytes_AtMinLen_Allowed() (gas:
21737)
[PASS]
test_verifyAllowedTargetOrRecipient_VaultExecuteWithdrawal_NonSelfBeneficiary_Unauthorized(address)
(runs: 1024, μ: 23200, ~: 23200)
[PASS] test_verifyAllowedTargetOrRecipient_VaultExecuteWithdrawal_SelfBeneficiary_Allowed() (gas: 20169)
[PASS] test_verifyAllowedTargetOrRecipient_VaultExecuteWithdrawal_ZeroAddressBeneficiary_Unauthorized()
(gas: 23300)
[PASS]
test_verifyAllowedTargetOrRecipient_VaultRequestWithdrawal_NonSelfBeneficiary_Unauthorized(address)
(runs: 1024, μ: 20350, ~: 20350)
[PASS] test_verifyAllowedTargetOrRecipient_VaultRequestWithdrawal_SelfBeneficiary_Allowed() (gas: 18882)
[PASS] test_verifyAllowedTargetOrRecipient_VaultRequestWithdrawal_ZeroAddressBeneficiary_Unauthorized()
(gas: 20358)
[PASS] test_verifyAllowedTargetOrRecipient_VaultWithdrawal_Batch_Allowed() (gas: 87452)
[PASS] test_verifyAllowedTargetOrRecipient_VaultWithdrawal_Batch_OneUnauthorized() (gas: 31390)
[PASS] test_verifyAllowedTargetOrRecipient_ZeroValueTokenCall() (gas: 74155)
Suite result: ok. 105 passed; 0 failed; 0 skipped; finished in 782.78ms (869.75ms CPU time)

Ran 10 tests for
test/v0.8/integration/AddressBookModuleSingleSigFork.t.sol:AddressBookModuleSingleSigFork
```

```
[PASS] test_addAllowedRecipients_throughAccount_updatesAllowlist() (gas: 124006)
[PASS] test_install_withInitialRecipients_storesAllowlist() (gas: 18273)
[PASS] test_removeAllowedRecipients_throughAccount_blocksSubsequentTransfer() (gas: 78787)
[PASS] test_runtime_executeBatch_allAllowed_succeeds() (gas: 103153)
[PASS] test_runtime_executeBatch_oneUnauthorized_reverts() (gas: 66947)
[PASS] test_runtime_execute_allowedRecipient_succeeds() (gas: 75903)
[PASS] test_runtime_execute_unauthorizedRecipient_reverts() (gas: 49313)
[PASS] test_userOp_executeBatch_allAllowed_succeeds() (gas: 211284)
[PASS] test_userOp_execute_allowedRecipient_succeeds() (gas: 197858)
[PASS] test_userOp_execute_unauthorizedRecipient_fails() (gas: 77617)
Suite result: ok. 10 passed; 0 failed; 0 skipped; finished in 2.87s (2.69s CPU time)

Ran 11 tests for test/v0.8/integration/AddressBookModuleMultiSigFork.t.sol:AddressBookModuleMultiSigFork
[PASS] test_install_recordsBothSigners() (gas: 21951)
[PASS] test_runtime_onMultiSigEntity_neverAuthorized() (gas: 49700)
[PASS] test_userOp_addAllowedRecipients_updatesAllowlist() (gas: 314928)
[PASS] test_userOp_duplicateSigners_rejected() (gas: 84009)
[PASS] test_userOp_executeBatch_oneUnauthorized_rejectedByHook() (gas: 93784)
[PASS] test_userOp_executeBatch_twoOfTwo_allAllowed_succeeds() (gas: 219740)
[PASS] test_userOp_execute_twoOfTwo_allowedRecipient_succeeds() (gas: 207673)
[PASS] test_userOp_execute_unauthorizedRecipient_rejectedByHook() (gas: 80771)
[PASS] test_userOp_onlyOneSignature_rejectedForLength() (gas: 75624)
[PASS] test_userOp_removeAllowedRecipients_blocksSubsequentTransfer() (gas: 224507)
[PASS] test_userOp_strangerInPlaceOfCosigner_rejected() (gas: 120516)
Suite result: ok. 11 passed; 0 failed; 0 skipped; finished in 2.87s (7.23s CPU time)

Ran 4 test suites in 2.87s (6.55s CPU time): 155 tests passed, 0 failed, 0 skipped (155 total tests)
```

Code Coverage

Tests were obtained with `forge test`, requiring a manual change in the `remappings.txt` to `forge-std/=lib/forge-std/` instead of `forge-std/=node_modules/forge-std/`.

Update Fix-Review:

Even though more tests were added the test coverage stayed nearly identical.

Update Followup-Review:

The added tests as part of PR 11 increase the coverage metrics throughout most contracts.

File	% Lines	% Statements	% Branches	% Funcs
src/libs/CastLib.sol	100.00% (10/10)	100.00% (11/11)	100.00% (0/0)	100.00% (2/2)
src/libs/RecipientAddressLib.sol	97.44% (76/78)	98.04% (100/102)	95.83% (23/24)	100.00% (11/11)
src/v0.8/BaseModule.sol	64.71% (11/17)	62.50% (10/16)	50.00% (1/2)	100.00% (4/4)
src/v0.8/modules/addressbook/AddressBookModule.sol	99.03% (102/103)	99.08% (108/109)	95.45% (21/22)	100.00% (16/16)
src/v0.8/validation/multisig/MultiSigValidationModule.sol	97.10% (67/69)	97.47% (77/79)	81.25% (13/16)	100.00% (12/12)
Total	96.03% (266/277)	96.53% (306/317)	90.62% (58/64)	100.00% (45/45)

Changelog

- 2026-02-12 - Initial report
- 2026-02-16 - Final report
- 2026-04-17 - Updated final report

About Quantstamp

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

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

To date, Quantstamp has performed more than 1300 audits and secured over $500 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, Solana, Canton, Polymarket, PancakeSwap, Virtuals Protocol, Wayfinder, Lido, TrustWallet, Alchemy, 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, Solana, Polygon, TON, Cardano, Binance Smart Chain, Avalanche, Arbitrum, Flow
- DeFi: Lido, Ethena, Compound, Curve, Venus, PancakeSwap, Polymarket
- Infra/Staking: TrustWallet, Alchemy, Liquid Collective, Kiln, Galxe, API3, ssv.network, Luganodes
- Immersive: Virtuals Protocol, Wayfinder, OpenSea, Square Enix, Parallel, Camp, Decentraland
- Institutional: Visa, Circle, Revolut, Anthropic, OpenAI, Canton, Republic, Arkham, Sequoia

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.

© 2026 – Quantstamp, Inc.