Quantstamp

Exactly WebAuthn Owner Plugin

Executive Summary

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

Type	ERC6900 Plugin
Timeline	2024-06-05 through 2024-06-10
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	README.md
Source Code	• exactly/webauthn-owner-plugin #7fcf594
Auditors	• Ruben Koch Senior Auditing Engineer • Shih-Hung Wang Auditing Engineer • Nikita Belenkov Auditing Engineer

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

Summary of Findings

In this audit, we reviewed an ERC-6900 plugin enabling authentication and authorization of critical function selectors for ERC-6900 accounts that would protect flows in the account such as contract upgrades and plugin (un)installations. It offers a variety of possibilities for authentication, including ECDSA signature validation, ERC-1271 contract validation and WebAuthn P256 signature support to e.g. enable Passkeys. The plugin is intended to be equivalent in functionality shared with the `MultiOwnerPlugin`.

We deem the plugin to be fully compatible with ERC-6900 v0.7. The only diverging aspect to the `MultiOwnerPlugin` was the different requirement to the calldata for installations (EXA-8). Some concerns were uncovered in the `updatePublicKeys()` function (EXA-1), but overall we deem the code and test suite to be in a mature state.

As co-authors of the ERC-6900 standard, we are excited about the functionality this plugin brings to the modular account ecosystem.

Update Fix-Review: All issues have been addressed and either fixed, mitigated or acknowledged. The already good test suite was extended by an additional 15 tests.

ID	DESCRIPTION	SEVERITY	STATUS
EXA-1	Incorrect Implementation of `updateOwnersPublicKeys()`	Medium	Fixed
EXA-2	Signature Malleability Is Possible	Low	Acknowledged
EXA-3	`SIG_VALIDATION_FAILED` Returned in Incorrect Cases	Low	Acknowledged
EXA-4	Signatures Are Valid Indefinitely	Informational	Mitigated
EXA-5	Potential Violation of ERC-7562 Rules	Informational	Mitigated
EXA-6	Potential Compatibility Issues with EVM Chains	Informational	Mitigated

ID	DESCRIPTION	SEVERITY	STATUS
EXA-7	Remove Possibility of Adding Malformed Addresses	Informational	Fixed
EXA-8	Owners Are Not Properly Cleared	Informational	Mitigated

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

- src/IWebauthnOwnerPlugin.sol
- src/OwnersLib.sol
- src/WebauthnOwnerPlugin.sol
- src/WebauthnModularAccountFactory.sol

Findings

EXA-1 Incorrect Implementation of `updateOwnersPublicKeys()`

Medium Fixed

> **Update**

Marked as "Fixed" by the client.
Addressed in: `6992ecd` .

File(s) affected: `WebauthnOwnerPlugin.sol`

Description: The `updateOwnersPublicKeys()` function of the `WebauthnOwnerPlugin` contract needs to be adjusted or can be improved as follows:

1. When an owner is removed from the owner array, it is either replaced by a new owner or the last owner in the array. In either case, the `ownerCount` is decreased by one, which shouldn't be the case if a new owner replaces the removed owner. The redundant decrease of the owner array length could cause some owners to be removed unexpectedly.
2. When a new owner replaces a removed owner, the new owner is not validated and, therefore, could contain an invalid public key or be a duplicate of an existing owner in the array.
3. The `for` loop that iterates the `ownersToRemove` array breaks when the size of the owner array is reduced to zero. As a result, some elements in `ownersToRemove` are not used but are still logged in the `OwnerUpdated()` event, which could cause inconsistency between off-chain monitoring tools and the on-chain state.
4. When adding a new owner with `owner.y == 0`, validate that `owner.x <= type(uint160).max`. Although the same check is performed in the `_validateSignature()` function, validating user input as early as possible is best practice.

Recommendation: Consider modifying the `updateOwnersPublicKeys()` function to resolve the above issues.

EXA-2 Signature Malleability Is Possible

Low · Acknowledged

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

```
Even following the recommendation, there would still be 2 valid signatures for the same data due to
ERC-2098 compact signature support.
  The signature verification implementation is not affected by signature malleability.
```

File(s) affected: `WebauthnOwnerPlugin.sol`

Description: One of the options for signatures for the owner is an ECDSA signature. In the current setup, it is possible to have 2 valid signatures for the same data/ `userOpHash` .

In the context of Ethereum signatures, the `s` value represents a critical part of the signature along with the `r` value. These values are calculated during the signing process using the private key of the sender and are meant to be unique for each signature. However, due to the mathematics of elliptic curve cryptography, it's possible to have two `s` values that are valid for the same `r` and message. This arises from the fact that elliptic curves operate in a cyclic group modulo `n` , where `n` is the order of the curve.

Both `s` and `-s mod n` can be valid for the same `r` and message, even though they are numerically different. The value `v` is used to determine if the expected value is `s` or `-s mod n` .

This functionality of the signature means that there could be 2 valid signatures for the same data. This causes issues in most systems, so it is a common practice to limit the `s` value to the upper quadrant of the elliptic curve so that only one `s` value can be accepted. This is currently not the case with the signature verification logic.

Recommendation: Restrict the s value to the upper quadrant by enforcing a conditional check. by enforcing `uint256(s) > 0x7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5D576E7357A4501DDFE92F46681B20A0` , as can be seen in the OpenZeppelin library.

EXA-3 `SIG_VALIDATION_FAILED` Returned in Incorrect Cases

Low · Acknowledged

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

```
Signature validation is a hot-path, and it would be more expensive to be fully spec-compliant.
  It follows Coinbase's Smart Wallet behavior.
```

File(s) affected: `WebauthnOwnerPlugin.sol`

Description: The ERC-4337 states the following for signature validation:

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

This requirement states that `SIG_VALIDATION_FAILED` should only be returned on a failing signature, i.e. one that does not match the expected value and all the other cases should revert, which is when the signature is malformed or invalid.

Due to the unknown nature of ERC1271 signatures, to return `SIG_VALIDATION_FAILED` for all failing signature validation is a valid approach that was also taken by the Alchemy team. However, the behaviour can be aligned for ECDSA and P256 signature validations. The following errors should revert instead of returning `SIG_VALIDATION_FAILED` :

1. `ECDSA`
    1. Invalid Signature Length
    2. Invalid Signature `S` value
2. `WebAuthnAuth`
    1. Invalid Signature `S` value
    2. Invalid `r` value

Recommendation: Consider adopting a behaviour similar to OpenZeppelin's `SignatureChecker` library, where if `owner.x.code.length == 0` , ECDSA validation is performed and else ERC1271 validation. In case of ECDSA validation, the code could then easily revert for the described cases, e.g. via OpenZeppelin's `SignatureChecker.recover()` function.

For P256 signature validation, revert if `webAuthnAuth.s > _P256_N_DIV_2` (src) or if `r == 0 || r >= FCL_Elliptic_ZZ.n || s == 0 || s >= FCL_Elliptic_ZZ.n` (src).

EXA-4 Signatures Are Valid Indefinitely

Informational Mitigated

Update

The client added an informative comment to the codebase in commit `5e7f737` .
The client provided the following explanation:

```
It follows Alchemy's Multi Owner Plugin and Coinbase's Smart Wallet behavior.
```

File(s) affected: `WebauthnOwnerPlugin.sol`

Description: In the case of an EOA and `webAuthnAuth` signatures, there currently is no expiry time on the validity of the signature, so if the nonce has not been used yet, this signature is valid for the specific `UserOp` for an indefinite amount of time.

Recommendation: Consider adding an expiry time to the signature. Alternatively, document this aspect to end users.

EXA-5 Potential Violation of ERC-7562 Rules

Informational Mitigated

Update

The client added an informative comment to the Readme in commit `80e8dff` .

File(s) affected: `WebauthnOwnerPlugin.sol`

Description: When verifying a signature, the `WebAuthn` library attempts to call the RIP-7212 precompile first, and if failed, it falls back to `FreshCryptoLib` . Since the `WebAuthn.verify()` can be called during a user operation validation phase, it has to comply with the ERC-7562 rules.

Rule OP-062 states that the RIP-7212 precompile can be called on networks that accept it. If not, the call to the `0x100` address is not allowed since the address does not have a deployed code and would violate Rule OP-041.

As a result, the account will be incompatible with ERC-4337 on networks that do not support the RIP-7212 precompile, such as Ethereum, at the time of writing.

Recommendation: Consider adding a warning in public-facing documentation to inform users of such limitations.

EXA-6 Potential Compatibility Issues with EVM Chains

Informational Mitigated

Update

The client added an informative comment to the Readme in commit `c9bba08` .

Description: `WebauthnOwnerPlugin` uses the `WebAuthn` library from the `base-org/webauthn-sol` repository as a dependency. As stated in the `webauthn-sol` documentation:

> `FreshCryptoLib uses the ModExp precompile (address(0x05)), which is not supported on some chains, such as Polygon zkEVM. This library will not work on such chains, unless they support the RIP-7212 precompile.`

It should be noted that the `WebauthnOwnerPlugin` may not work as expected on chains that do not support the `ModExp` and RIP-7212 precompile.

Recommendation: Consider adding a warning in public-facing documentation to inform users of such limitations.

EXA-7 Remove Possibility of Adding Malformed Addresses

Informational · Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `0ff0816`.

File(s) affected: `WebauthnOwnerPlugin.sol`

Description: Currently `_validateSignature()` performs the following check: `owner.x > type(uint160).max`. This allows the system to verify that this is a valid address stored.

It is a good practice to not allow for invalid addresses to be added in the first place, so this check should be performed at installation or owner update calls to avoid having invalid addresses altogether.

Recommendation: Consider performing this check before an address is added.

EXA-8 Owners Are Not Properly Cleared

Informational · Mitigated

Update
The client added an informative comment to the codebase in commit `6f8acb2`.

File(s) affected: `WebauthnOwnerPlugin.sol`, `OwnersLib.sol`

Description: When the plugin is uninstalled, the `onUninstall()` function is called, which clears the associated storage with the MSCA address. However, the storage is not fully cleared; only the `_owners.length` is set to 0, while the `PublicKey` array entries remain in the mapping, but become inaccessible due to the length being 0. Similarly, if an owner is removed, its storage entry is not reset, but `_owners.length` is simply decremented. This currently does not cause issues as lookup functions such as `contains()` are bound by the provided size of the array. When the plugin is reinstalled or if additional owners are added after previous removal of owners, these stale, invalid entries are overwritten.

It would be a good idea to fully delete the `PublicKey` array from storage when uninstalling the plugin or deleting individual entries when removing owners to avoid potential issues with future iterations or forks of the codebase where length might not be enforced during lookup.

Recommendation: Consider clearing the `PublicKey` array when the plugin is uninstalled or owners are updated.

Auditor Suggestions

EXA-S-1 Unlocked Pragma

Acknowledged

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

> `Deployers are expected to be able to choose to build the contracts with newer compiler versions without modifying the code.`
> `The caret range enforces backward compatibility. We trust the Solidity team to follow SemVer strictly and not introduce breaking changes in patch versions.`

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

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

EXA-S-2 Incompatibility with ERC4337 V0.7

Mitigated

> **Update**
> The client added an informative comment to the Readme in commit `00e9f35`.

Description: We want to raise awareness of the fact that the plugin is not compatible with the latest version of 4337 at the time of writing, namely v0.7. However, due to the intended compatibility with Alchemy's `MultiOwnerPlugin`, this is desired.

EXA-S-3 Additional Permission Descriptor for `updateOwnersPublicKeys()`

Acknowledged

> **Update**
> Marked as "Acknowledged" by the client.
> The client provided the following explanation:
>
> `We consider Webauthn public keys to be first-class citizens and no different than Ethereum addresses.`

Description: The additional capabilities of the `updateOwnersPublicKeys()` function, which are a superset of the ones from the `updateOwners()` function by also enabling the addition of WebAuthn public keys as owners, should be distinctly emphasized within a separate permission descriptor in the plugin's metadata.

EXA-S-4 Gas Improvements

Fixed

> **Update**
> Marked as "Fixed" by the client.
> Addressed in: `bb8c521`, `65f1885`, `74aa53e`. A few instances of array length caching in for loops are still missing.

Description:

- `array.length` in for-loops can be cached in a local variable to save gas.
- `WebAutnOwnerPlugin._onInstall()` can use `ownerCount` as the index of the for-loop.
- `WebAutnOwnerPlugin._onInstall()` can omit the declaration and usage of the `keys` variable by setting the public keys in this manner: `owners.publicKeys[ownerCount] = initialOwners[ownerCount];`.

EXA-S-5 Incorrect Usage of Library Functions Can Lead to Out-of-Bounds Reading on Owner Array

Mitigated

> **Update**
> The client added informative comments to the codebase in commit `ee3450b`.

Description: `OwnersLib` library introduces functions such as `find()` and `contains()` to allow searching of the array of owners. For the mentioned functions the current length of owners is passed in as a parameter, which is used as the limitation on the search space. Currently, the only thing limiting the reading outside of the allowed owner list is that length parameter, it is important to document this as if the default max size length is passed in, this would allow reading previously deleted values.

Recommendation: Document this behaviour and its potential risks in the library.

EXA-S-6 `onInstall()` Function Requires Slightly Different Calldata than `MultiOwnerPlugin`

Mitigated

> **Update**
> The client added an informative comment to the codebase in commit `85e82f4`.

File(s) affected: `WebAuthnOwnerPlugin.sol`

Description: The plugin is designed to be as equivalent as possible to Alchemy's `MultiOwnerPlugin`. Given that, we want to point out that the required calldata during installation of this plugin does differ from the `MultiOwnerPlugin`, because here, `PublicKey[]` is the expected encoding, while in `MultiOwnerPlugin` it is `address[]`.

Recommendation: We mainly want to raise awareness about this minor divergence. This would only be a problem if e.g. deployment scripts were expected to be used interchangeably between the two plugins.

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

- `123...b07 ./src/WebauthnModularAccountFactory.sol`
- `596...b0c ./src/WebauthnOwnerPlugin.sol`
- `372...94c ./src/OwnersLib.sol`
- `37a...746 ./src/IWebauthnOwnerPlugin.sol`

Tests

- `0cd...c85 ./test/WebauthnPluginIntegration.t.sol`
- `386...aa6 ./test/WebauthnModularAccountFactory.t.sol`
- `132...369 ./test/WebauthnOwnerPlugin.t.sol`
- `3d3...b88 ./test/utils/TestLib.sol`

Toolset

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

Setup

Tool Setup:
- Slither v0.10.0

Steps taken to run the tools:
1. Install the Slither tool: `pip3 install slither-analyzer`
2. Run Slither from the project directory: `slither .`

Automated Analysis

Slither

Slither identified 8 results. Non-false positive findings have been included in the report.

Test Suite Results

Test output was obtained with `forge test`.

Update Fix-Review: The already thorough test suite has been extended by another 15 tests.

```
Ran 4 tests for test/WebauthnPluginIntegration.t.sol:MultiOwnerPluginIntegration
[PASS] test_ownerPlugin_successInstallation() (gas: 39135)
[PASS] test_runtimeValidation_alwaysAllow_isValidSignature() (gas: 104093)
[PASS] test_runtimeValidation_ownerOrSelf_standardExecute() (gas: 145057)
[PASS] test_userOpValidation_owner_standardExecute() (gas: 347849)
Suite result: ok. 4 passed; 0 failed; 0 skipped; finished in 22.49ms (9.73ms CPU time)

Ran 19 tests for test/WebauthnModularAccountFactory.t.sol:WebauthnModularAccountFactoryTest
[PASS] test_2StepOwnershipTransfer() (gas: 87593)
[PASS] test_addStake() (gas: 106153)
[PASS] test_addressMatch() (gas: 801814)
[PASS] test_badOwnersArray() (gas: 18409)
[PASS] test_constructor_failWithZeroAddress() (gas: 1740649)
[PASS] test_deploy() (gas: 810381)
[PASS] test_deployCollision() (gas: 826021)
[PASS] test_deployWithDuplicateOwners() (gas: 737357)
[PASS] test_deployWithUnsortedOwners() (gas: 737379)
[PASS] test_deploy_PasskeyOwner() (gas: 797192)
[PASS] test_deployedAccountHasCorrectPlugins() (gas: 801483)
[PASS] test_getAddressWithMaxOwnersAndDeploy() (gas: 2704053)
[PASS] test_getAddressWithTooManyOwners() (gas: 205692)
[PASS] test_getAddressWithUnsortedOwners() (gas: 11342)
[PASS] test_renounceOwnership() (gas: 10459)
[PASS] test_unlockStake() (gas: 147882)
[PASS] test_withdraw() (gas: 128078)
[PASS] test_withdrawStake() (gas: 210351)
[PASS] test_withdraw_token() (gas: 995155)
Suite result: ok. 19 passed; 0 failed; 0 skipped; finished in 520.44ms (18.90ms CPU time)

Ran 32 tests for test/WebauthnOwnerPlugin.t.sol:MultiOwnerPluginTest
[PASS] testFuzz_isValidSignature_ContractOwner(bytes32) (runs: 256, μ: 110067, ~: 110067)
[PASS] testFuzz_isValidSignature_ContractOwnerWithEOAOwner(bytes32) (runs: 256, μ: 120381, ~: 120381)
[PASS] testFuzz_isValidSignature_EOAOwner(string,bytes32) (runs: 256, μ: 130741, ~: 130734)
[PASS] testFuzz_isValidSignature_PasskeyOwner(bytes32) (runs: 256, μ: 365788, ~: 365955)
[PASS] testFuzz_runtimeValidationFunction_BadFunctionId(uint8) (runs: 256, μ: 9747, ~: 9747)
[PASS] testFuzz_userOpValidationFunction_BadFunctionId(uint8) (runs: 256, μ: 10744, ~: 10744)
[PASS]
testFuzz_userOpValidationFunction_ContractOwner((address,uint256,bytes,bytes,uint256,uint256,uint256,uint
256,uint256,bytes,bytes)) (runs: 256, μ: 130910, ~: 130910)
[PASS]
testFuzz_userOpValidationFunction_ContractOwnerWithEOAOwner((address,uint256,bytes,bytes,uint256,uint256,
uint256,uint256,uint256,bytes,bytes)) (runs: 256, μ: 144525, ~: 144521)
[PASS] testFuzz_userOpValidationFunction_EOAOwner(string,
(address,uint256,bytes,bytes,uint256,uint256,uint256,uint256,uint256,bytes,bytes)) (runs: 256, μ: 138772,
~: 138764)
[PASS]
testFuzz_userOpValidationFunction_PasskeyOwner((address,uint256,bytes,bytes,uint256,uint256,uint256,uint2
56,uint256,bytes,bytes)) (runs: 256, μ: 373522, ~: 373753)
[PASS] test_eip712Domain() (gas: 35438)
[PASS] test_isValidSignature_failMalformedAddress() (gas: 15544)
[PASS] test_isValidSignature_failWithOutOfBounds() (gas: 12319)
[PASS] test_multiOwnerPlugin_sentinelIsNotOwner() (gas: 19805)
[PASS] test_onInstall_failWithEmptyOwners() (gas: 36299)
[PASS] test_onInstall_failWithInvalidAddress() (gas: 38500)
[PASS] test_onInstall_failWithLimitExceeded() (gas: 1714289)
[PASS] test_onInstall_success() (gas: 94741)
[PASS] test_onUninstall_success() (gas: 84935)
```

```
[PASS] test_ownerIndexOf_failWithNotExist() (gas: 28310)
[PASS] test_pluginInitializeGuards() (gas: 163933)
[PASS] test_pluginManifest() (gas: 38695)
[PASS] test_pluginMetadata_success() (gas: 16954)
[PASS] test_runtimeValidationFunction_OwnerOrSelf() (gas: 26703)
[PASS] test_updateOwnersPublicKeys_failWithInvalidAddress() (gas: 55729)
[PASS] test_updateOwners_addAndRemove() (gas: 179888)
[PASS] test_updateOwners_failWithDuplicatedAddresses() (gas: 85516)
[PASS] test_updateOwners_failWithEmptyOwners() (gas: 70726)
[PASS] test_updateOwners_failWithLimitExceeded() (gas: 1924832)
[PASS] test_updateOwners_failWithNotExist() (gas: 58523)
[PASS] test_updateOwners_failWithZeroAddressOwner() (gas: 56428)
[PASS] test_updateOwners_success() (gas: 119812)
Suite result: ok. 32 passed; 0 failed; 0 skipped; finished in 520.46ms (2.20s CPU time)

Ran 3 test suites in 538.40ms (1.06s CPU time): 55 tests passed, 0 failed, 0 skipped (55 total tests)
```

Code Coverage

Code coverage data was obtained with `forge coverage`. The metrics show good to very good results, only the branch coverage in the `WebauthnOwnerPlugin` could be slightly improved.

Update Fix-Review: The coverage metrics increased to an excellent level.

File	% Lines	% Statements	% Branches	% Funcs
script/Deploy.s.sol	100.00% (**6/**6)	100.00% (**6/**6)	50.00% (**1/**2)	100.00% (**1/**1)
src/OwnersLib.sol	100.00% (**33/**33)	100.00% (**66/**66)	100.00% (**8/**8)	100.00% (**12/**12)
src/WebauthnModularAccountFactory.sol	100.00% (**33/**33)	100.00% (**55/**55)	100.00% (**12/**12)	100.00% (**9/**9)
src/WebauthnOwnerPlugin.sol	100.00% (**107/**107)	100.00% (**157/**157)	87.50% (**28/**32)	95.00% (**19/**20)
test/utils/TestLib.sol	100.00% (**1/**1)	100.00% (**2/**2)	100.00% (**0/**0)	100.00% (**1/**1)
Total	100.00% (**180/**180)	100.00% (**286/**286)	90.74% (**49/**54)	97.67% (**42/**43)

Changelog

- 2024-06-10 - Initial report
- 2024-07-08 - Final report

About Quantstamp

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

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

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

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

Notable Collaborations & Customers:

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

Timeliness of content

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

Notice of confidentiality

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

Links to other websites

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

Disclaimer

The review and this report are provided on an as-is, where-is, and as-available basis. To the fullest extent permitted by law, Quantstamp disclaims all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. You agree that your access and/or use of the report and other results of the review, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE. This report is based on the scope of materials and documentation provided for a limited review at the time provided. You acknowledge that Blockchain technology remains under development and is subject to unknown risks and flaws and, as such, the report may not be complete or inclusive of all vulnerabilities. The review is limited to the materials identified in the report and does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. The report does not indicate the endorsement by Quantstamp of any particular project or team, nor guarantee its security, and may not be represented as such. No third party is entitled to rely on the report in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. Quantstamp does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party, or any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, or any related services and products, any hyperlinked websites, or any other websites or mobile applications, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third party. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.