Base WebAuthn

Security Review

Cantina Managed review by:
Riley Holterhus, Lead Security Researcher
Rappie, Associate Security Researcher

March 7, 2024

Contents

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity	Description
Critical	*Must* fix as soon as possible (if already deployed).
High	Leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
Medium	Global losses <10% or losses to only a subset of users, but still unacceptable.
Low	Losses will be annoying but bearable. Applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.
Gas Optimization	Suggestions around gas saving practices.
Informational	Suggestions around best practices or readability.

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

Base is a secure and low-cost Ethereum layer-2 solution built to scale the userbase on-chain.

From Feb 14th to Feb 19th the Cantina team conducted a review of `WebAuthn.sol` of smart-account on commit hash 2d86e4c4. The team identified a total of **5** issues in the following risk categories:

- Critical Risk: 0
- High Risk: 0
- Medium Risk: 0
- Low Risk: 1
- Gas Optimizations: 0
- Informational: 4

3 Findings

3.1 Low Risk

3.1.1 Add check for `VERIFIER` call return status

Severity: Low Risk

Context: WebAuthn.sol#L147-L151

Description: When the `VERIFIER` precompile is called, the `bool` status value is ignored when interpreting the result of the call. While it makes sense to not solely rely on the status (since a non-existent precompile would return `true`), it would still make sense to ensure the status is `true` if the validation is deemed to have succeeded. While it does not appear possible with the RIP-7212 spec, this would prevent scenarios where a reverting `VERIFIER` call is interpreted as a successful validation due to the returned error message.

Recommendation: Add a check on the status of the low-level call to the `VERIFIER`:

```
(bool success, bytes memory ret) = VERIFIER.staticcall(args);
// staticcall will not revert if address has no code
// check return length in addition to return status
bool valid = success && ret.length > 0;
if (valid) return abi.decode(ret, (uint256)) == 1;
```

Base: Fixed in PR 46.

Cantina Managed: Verified.

3.2 Informational

3.2.1 Typographical error

Severity: Informational

Context: WebAuthn.sol#L29

Description: There is a typographical error in the following comment:

```
/// @dev The r value of secp256r1 signature
uint256 s;
```

Recommendation: Change the comment to reflect the correct variable

```
  /// @dev The r value of secp256r1 signature
  uint256 r;
- /// @dev The r value of secp256r1 signature
+ /// @dev The s value of secp256r1 signature
  uint256 s;
```

Base: Fixed in PR 49.

Cantina Managed: Verified.

3.2.2 Pending comments

Severity: Informational

Context: WebAuthn.sol#L4

Description: There is a pending comment in the `WebAuthn` library code that reads `"TODO check apache license"`.

Recommendation: After following up with any relevant license checks, consider removing this temporary comment in the final version of the code.

Base: Removed the comment in the initial commit (commit 1a0c8d23) that migrated the `WebAuthn` library to its own repository.

Cantina Managed: Verified.

3.2.3 Consider situations where `VERIFIER` constant needs to be changed

Severity: Informational

Context: WebAuthn.sol#L147-L151

Description: The `VERIFIER` address, which is currently set as the constant `0x100`, is intended to represent the address where the RIP-7212 precompile exists, or will exist in the future. Considering the Rollup Improvement Proposal (RIP) process is in its early stages, and considering the diversity of L2s that may use the `WebAuthn` library, there's a possibility that an L2's secp256r1 verification precompile will be located at an address other than `0x100`. In particular, there has been recent discussions regarding L2s that may implement RIPs with slightly different gas pricing from official RIPs. It's anticipated that such L2s will require distinct RIPs and precompile addresses, rendering them incompatible with the current `WebAuthn` library setup.

Recommendation: To prepare for scenarios where it's known that an L2 uses a different precompile address, consider documenting this possibility in the code, for example as follows:

```
/// @dev In rare circumstances, an L2 might deploy its secp256r1 precompile to an address
/// other than 0x100 (most likely due to the L2 deviating from the RIP-7212 gas pricing).
/// The `VERIFIER` constant here may therefore be updated, if this is known ahead of time.
address constant VERIFIER = address(0x100);
```

Base: We do not plan to remediate here given we already fallback to library and there is a lot of alignment around the reserved RIP address.

Cantina Managed: Acknowledged.

3.2.4 Invalid signature can be checked twice

Severity: Informational

Context: WebAuthn.sol#L147-L153

Description: The `WebAuthn` library supports secp256r1 verification through two methods. Initially, the `verify()` function tries to use the `VERIFIER` precompile address, and if that precompile is unavailable, it falls back to using the `FCL_ecdsa` library. This is implemented as follows:

```
// try the RIP-7212 precompile address
(, bytes memory ret) = VERIFIER.staticcall(args);
// staticcall will not revert if address has no code
// check return length
bool valid = ret.length > 0;
if (valid) return abi.decode(ret, (uint256)) == 1;

return FCL_ecdsa.ecdsa_verify(messageHash, webAuthnAuth.r, webAuthnAuth.s, x, y);
```

Notice that the `VERIFIER` result is ignored if the precompile does not return any data. In the current RIP-7212 spec, an invalid signature will return no data, and will not error. This means that even if the precompile does exist, invalid signatures will still fall back to the `FCL_ecdsa` library, which implies the signature verification is computed twice.

This may be unexpected, as it introduces a path to using the `FCL_ecdsa` library on L2s that *do* support the RIP-7212 precompile. This behavior could also lead to wasted gas, but this is less of a concern since invalid signatures should be caught before being executed on-chain anyway.

Recommendation: Consider documenting this behavior, so that it's known that the `FCL_ecdsa` logic can be reached even on L2s with the RIP-7212 precompile.

Alternatively, consider adopting an implementation that permanently switches to using the `VERIFIER` precompile once it's known to exist. This can be accomplished in a trustless manner (e.g. by "proving" the precompile exists by validating an example signature), or by eventually deploying an entirely new `WebAuthn` library for users to use in an upgrade.

Base: Addressed in PR 46 by documenting this behavior.

Cantina Managed: Verified.

4 Appendix

4.1 Non-conformant `clientDataJSON` serializations

Context: WebAuthn.sol#L110-L123

Description: The Base team raised this change during the audit. This change has been added here for tracking purposes.

The WebAuthn specification provides a detailed procedure for verifying an authentication assertion. A critical part of this process involves checking certain attributes within the `clientDataJSON` input. The specification provides a specific JSON structure for this input, so that verification can be done without the need for full JSON parsing (which is especially useful for on-chain verification, where full parsing would be expensive).

Unfortunately, some clients do not conform to the specification. For example, Mozilla does not follow the requirement to place the `type` field first, and instead places the `challenge` field first. The `WebAuthn` library will break with these non-conforming clients, as the `clientDataJSON` used on-chain will not match what was signed off-chain.

Recommendation: To conform to clients that don't follow the WebAuthn specification, Base has decided to change their verification algorithm. Now, the ordering of fields is not assumed, and the locations of each relevant field is provided as user input.

Base: As described above, changed in PR 6.

Cantina Managed: Verified. Following the recent implementation changes, there was a discussion that the `challenge` and `type` fields are no longer guaranteed to be found in the top-level of the `clientDataJSON` structure. It's now especially important for users to avoid signing data where there are additional `challenge` and `type` fields in unexpected locations (e.g. within other sub-fields) of the `clientDataJSON`, as these values can be interpreted as valid when they wouldn't have before. The implication is that users need to remain cautious as always when they sign any data.

4.2 Additional Testing

Description: We have created additional tests to check the behaviour of `verify`. These tests leverage test vectors automatically created by WebAuthn authentication assertions using a headless browser with a virtual authenticator device. Random authentication assertions are created using `ffi` with fuzzed values in the generator script.

The code can be found here:

- Foundry tests
- Python scripts

Result: No additional security issues have been identified.

Recommendation: Incorporate these tests into the existing test suite.