ChainLight

Rhinestone WebAuthnValidator Security Audit

：Rhinestone WebAuthnValidator

Jan 19, 2026

Revision 1.0

ChainLight@Theori

Table of Contents

Executive Summary

Beginning on December 27, 2025, ChainLight conducted a 3-day security audit of the Rhinestone Smart Contract. The audit focused on whether removing the account-bound linkage could allow signatures produced by the same passkey to be reused (replayed) across multiple smart accounts.

The audit revealed a total of 1 issue, categorized by severity as follows:

- Total: 1
- Low: 1

Since the identified issue was acknowledged without further code changes, the final commit remains identical to the one listed in the Audit Target.

Audit Overview

Scope

Name	Rhinestone WebAuthnValidator Security Audit
Target / Version	• https://github.com/rhinestonewtf/core-modules/pull/31: `b3c9904b75f9ea241129bd299ca22941de28108b`
Application Type	Smart contracts
Lang. / Platforms	Smart contracts [Solidity]

Code Revision

N/A

Severity Categories

Severity	Description
Critical	The attack cost is low (not requiring much time or effort to succeed in the actual attack), and the vulnerability causes a high-impact issue. (e.g., Effect on service availability, Attacker taking financial gain)
High	An attacker can succeed in an attack which clearly causes problems in the service's operation. Even when the attack cost is high, the severity of the issue is considered "high" if the impact of the attack is remarkably high.
Medium	An attacker may perform an unintended action in the service, and the action may impact service operation. However, there are some restrictions for the actual attack to succeed.
Low	An attacker can perform an unintended action in the service, but the action does not cause significant impact or the success rate of the attack is remarkably low.
Informational	Any informational findings that do not directly impact the user or the protocol.
Note	Neutral information about the target that is not directly related to the project's safety and security.

Status Categories

Status	Description
Reported	ChainLight reported the issue to the client.
WIP	The client is working on the patch.
Patched	The client fully resolved the issue by patching the root cause.
Mitigated	The client resolved the issue by reducing the risk to an acceptable level by introducing mitigations.
Acknowledged	The client acknowledged the potential risk, but they will resolve it later.
Won't Fix	The client acknowledged the potential risk, but they decided to accept the risk.

Finding Breakdown by Severity

Category	Count	Findings
Critical	0	• N/A
High	0	• N/A
Medium	0	• N/A
Low	1	• `WebAuthn-2512-01`
Informational	0	• N/A
Note	0	• N/A

Findings

Summary

#	ID	Title	Severity	Status
1	`WebAuthn-2512-01`	Signature replay risk in `isValidSignatureWithSender()` and `validateSignatureWithData()`	Low	Acknowledged

#1 `WebAuthn-2512-01` Signature replay risk in `isValidSignatureWithSender()` and `validateSignatureWithData()`

ID	Summary	Severity
`WebAuthn-2512-01`	If `isValidSignatureWithSender()` and `validateSignatureWithData()` compute the signature hash without binding it to the smart account address, signatures produced by the same passkey may be replayed across different accounts. Applying ERC-7739-style defensive rehashing can prevent cross-account signature reuse.	Low

Description

In `WebAuthnValidator`, `validateUserOp()` verifies signatures over `userOpHash`, which is derived in a way that includes the account address, preventing signature reuse across different accounts.

However, for `isValidSignatureWithSender()` and `validateSignatureWithData()`, there is no guarantee that the message hash being validated is bound to the account address. In such cases, a signature produced by a passkey could be reused by another smart account that uses the same passkey.

For example, in Permit2's `permitWitnessTransferFrom()`, the contract calls `IERC1271(claimedSigner).isValidSignature(hash, signature)`, where `hash` does not include the account address. If multiple smart accounts share the same passkey, a Permit2 signature may be replayed, allowing funds to be authorized and spent from a different account than originally intended.

To mitigate this risk, a defensive rehash that incorporates the account address should be performed before signature verification.

Impact

Low

If the same passkey and `WebAuthnValidator` module are used across multiple smart accounts, signatures verified via `isValidSignatureWithSender()` and `validateSignatureWithData()` may be replayed across accounts when the validated hash is not account-bound. This can lead to unintended authorization and potential double-spend of funds in integrations that rely on ERC-1271 signature validation (e.g., Permit2). However, since the system can apply EIP-7739-style defensive rehashing via the prevalidation hook, this replay risk can be mitigated in practice. Considering this additional defense, the issue is assessed as Low severity.

Recommendation

Apply ERC-7739 to `isValidSignatureWithSender()` and `validateSignatureWithData()` by defensively rehashing the input hash with the smart account address before verification, ensuring signatures are not valid across different accounts.

Remediation

Acknowledged

The team is already aware of this issue, and performs EIP-7739 protection via the prevalidation hook.

Revision History

Version	Date	Description
1.0	Jan 19, 2026	Initial version
1.1	Jan 20, 2026	Added post-patch commits

Theori, Inc. (“We”) is acting solely for the client and is not responsible to any other party. Deliverables are valid for and should be used solely in connection with the purpose for which they were prepared as set out in our engagement agreement. You should not refer to or use our name or advice for any other purpose. The information (where appropriate) has not been verified. No representation or warranty is given as to accuracy, completeness or correctness of information in the Deliverables, any document, or any other information made available. Deliverables are for the internal use of the client and may not be used or relied upon by any person or entity other than the client. Deliverables are confidential and are not to be provided, without our authorization (preferably written), to entities or representatives of entities (including employees) that are not the client, including affiliates or representatives of affiliates of the client.