



Coinbase x402

Security Review

Cantina Managed review by:

Sujith Somraaj, Lead Security Researcher

Red-swan, Security Researcher

March 4, 2026

Contents

| | | |
|----------|--|----------|
| 1 | Introduction | 2 |
| 1.1 | About Cantina | 2 |
| 1.2 | Disclaimer | 2 |
| 1.3 | Risk assessment | 2 |
| 1.3.1 | Severity Classification | 2 |
| 2 | Security Review Summary | 3 |
| 2.1 | Scope | 3 |
| 3 | Findings | 4 |
| 3.1 | Informational | 4 |
| 3.1.1 | Permissionless settlement enables forced payment and cancellation race in x402ExactPermit2Proxy | 4 |

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 level | Impact: High | Impact: Medium | Impact: Low |
|--------------------|--------------|----------------|-------------|
| Likelihood: high | Critical | High | Medium |
| Likelihood: medium | High | Medium | Low |
| Likelihood: low | Medium | Low | Low |

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 are 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, low-cost, builder-friendly Ethereum L2 built to bring the next billion users onchain.

From Feb 27th to Feb 28th the Cantina team conducted a review of [x402](#) on commit hash [79136b97](#). The team identified a total of **1** issues:

Issues Found

| Severity | Count | Fixed | Acknowledged |
|-------------------|----------|----------|--------------|
| Critical Risk | 0 | 0 | 0 |
| High Risk | 0 | 0 | 0 |
| Medium Risk | 0 | 0 | 0 |
| Low Risk | 0 | 0 | 0 |
| Gas Optimizations | 0 | 0 | 0 |
| Informational | 1 | 0 | 1 |
| Total | 1 | 0 | 1 |

2.1 Scope

The security review had the following components in scope for [x402](#) on commit hash [79136b97](#):

```
contracts/evm/src
├─ x402BasePermit2Proxy.sol
├─ x402ExactPermit2Proxy.sol
└─ x402UptoPermit2Proxy.sol
```

3 Findings

3.1 Informational

3.1.1 Permissionless settlement enables forced payment and cancellation race in `x402ExactPermit2Proxy`

Severity: Informational

Context: `x402ExactPermit2Proxy.sol#L51`, `x402ExactPermit2Proxy.sol#L85`

Description: The `settle()` and `settleWithPermit()` functions in `x402ExactPermit2Proxy` impose no restriction on `msg.sender`. Any party that possesses the payment data (permit, witness, signature) can execute a settlement, not just the intended facilitator. This is in direct contrast to `x402UptoPermit2Proxy`, which binds the caller via a `facilitator` field in the signed witness and enforces `msg.sender == witness.facilitator`.

In the `x402` flow, the payer's payment authorization (permit, struct, witness, and signature) is transmitted to the server/facilitator over HTTPS. Any party that obtains this data through a compromised CDN, a leaked access log, or a mempool observer watching for the payer's nonce-invalidation transaction, can call `settle()` directly:

1. Payer signs a payment authorization and transmits it to the facilitator.
2. Payer discovers the merchant is fraudulent and broadcasts a `Permit2.invalidateUnorderedNonces` transaction to `cancel`.
3. Griever observes the cancellation in the mempool, extracts the payment data, and front-runs with `x402ExactPermit2Proxy.settle()`.
4. Settlement succeeds: payer's funds are transferred to `witness.to`.
5. Payer has paid; service may never be rendered.
6. Payer's cancellation transaction executes, but the nonce is already consumed.

Recommendation: Add a `facilitator` field to the `Witness` struct and enforce `msg.sender == witness.facilitator`, mirroring the protection already present in `x402UptoPermit2Proxy`.

Coinbase: This is perfectly acceptable, and just informational.

The facts are:

- Servers may use multiple facilitators, forcing them to race. So even beyond the risk that a CDN may be compromised, or an observer may be watching the mempool, facilitators should just be aware that other facilitators & 3rd parties may have the settlement payloads.
- Clients may revoke authorization, and facilitators should be prepared for transactions to have their authorization revoked.
- Both of those lead to the fact that facilitators should be coded defensively. They need to be prepared for the fact that a settlement transaction may fail for a number of reasons, including but not limited to the fact that authorization may have already been used or cancelled by the time a facilitator attempts settlement. This is expected and within spec.

Per your point about: "5. Payer has paid; service may never be rendered."

This is based on the assumption that a server is fraudulent and simply chose not to return the resource despite being paid. We believe that is an app/ecosystem level concern, not a smart contract or `x402` spec concern. Separately from `x402`, reputation systems such as EIP-8004 are being integrated into the ecosystem. `x402` itself is just a protocol-level primitive for requesting/making/facilitating payments. It's up to the apps themselves to ensure the server they choose to do business with is reputable.

Cantina Managed: Acknowledged.