



# Coinbase x402

## Security Review

Cantina Managed review by:

**Sujith Somraaj**, Lead Security Researcher

**Red-Swan**, Security Researcher

February 19, 2026

# Contents

|          |   |          |
|----------|---|----------|
| <b>1</b> | <b>Introduction</b>   | <b>2</b> |
| 1.1      | About Cantina . . . . .   | 2        |
| 1.2      | Disclaimer . . . . .  | 2        |
| 1.3      | Risk assessment . . . . .   | 2        |
| 1.3.1    | Severity Classification . . . . .   | 2        |
| <b>2</b> | <b>Security Review Summary</b>  | <b>3</b> |
| 2.1      | Scope . . . . .   | 3        |
| <b>3</b> | <b>Findings</b>   | <b>4</b> |
| 3.1      | Low Risk . . . . .  | 4        |
| 3.1.1    | Signature griefing/frontrunning in <code>x402UptoPermit2Proxy</code> . . . . .  | 4        |
| 3.1.2    | Unbounded length for <code>extra</code> data in witness struct . . . . .  | 4        |
| 3.1.3    | Absence of zero-amount validation allows no-op settlements that consume nonces . . . . .                                  | 5        |
| 3.2      | Informational . . . . .   | 5        |
| 3.2.1    | Replace <code>tx.origin</code> with <code>msg.sender</code> in constructor to fully support multisig deployment . . . . . | 5        |
| 3.2.2    | Non-standard variable naming . . . . .  | 6        |
| 3.2.3    | Lack of observability for EIP-2612 permit failures . . . . .  | 6        |
| 3.2.4    | Ambiguous parameter naming in <code>_settle()</code> function . . . . .   | 6        |

# 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](https://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 11th to Feb 13th the Cantina team conducted a review of [x402](#) on commit hash [c0a80b76](#). The team identified a total of **7** issues:

### Issues Found

| Severity          | Count    | Fixed    | Acknowledged |
|-------------------|----------|----------|--------------|
| Critical Risk     | 0        | 0        | 0            |
| High Risk         | 0        | 0        | 0            |
| Medium Risk       | 0        | 0        | 0            |
| Low Risk          | 3        | 3        | 0            |
| Gas Optimizations | 0        | 0        | 0            |
| Informational     | 4        | 4        | 0            |
| <b>Total</b>      | <b>7</b> | <b>7</b> | <b>0</b>     |

### 2.1 Scope

The security review had the following components in scope for [x402](#) on commit hash [c0a80b76](#):

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

## 3 Findings

### 3.1 Low Risk

#### 3.1.1 Signature griefing/frontrunning in x402UptoPermit2Proxy

**Severity:** Low Risk

**Context:** x402UptoPermit2Proxy.sol#L34, x402UptoPermit2Proxy.sol#L60

**Description:** In the x402UptoPermit2Proxy, a user signs a permit for a maximum amount (permit.permitted.amount). The facilitator then calls settle() or settleWithPermit() with a specific amount that is less than or equal to the maximum. The Permit2 signature covers the permit and the witness (destination, timestamp), but it does not cover the amount parameter passed to settle().

```
function settle(..., uint256 amount, ...) external nonReentrant {
    if (amount > permit.permitted.amount) revert AmountExceedsPermitted();
    _settle(permit, amount, owner, witness, signature);
}
```

This gives anyone monitoring the mempool (or even a malicious/incompetent facilitator) the opportunity to attempt to settle the payment with a different amount. This could be an amount near zero or for the full permissioned amount. The funds will still reach the correct recipient, but this disrupts the intended payment, and the payer/payee may be forced to settle for an unintended amount.

**Recommendation:** Require that the payer sign over the facilitator's address and they be the caller of this function so that only they have the intended power to choose the final settlement amount.

**Coinbase:** Fixed in commit 373ff0e6e: add facilitator to Witness struct to prevent settlement frontrunning and amount.

**Cantina Managed:** Verified fix. But the fixes introduced a centralization risk for the facilitator.

#### 3.1.2 Unbounded length for extra data in witness struct

**Severity:** Low Risk

**Context:** x402BasePermit2Proxy.sol#L86

**Description:** The Witness struct in x402BasePermit2Proxy.sol defines an extra field as a dynamically-sized bytes type with no upper-bound length validation:

```
struct Witness {
    address to;
    uint256 validAfter;
    bytes extra;
}
```

During settlement, the \_settle() function computes keccak256(witness.extra) as part of the witness hash reconstruction:

```
bytes32 witnessHash =
    keccak256(abi.encode(WITNESS_TYPEHASH, witness.to, witness.validAfter,
        ↪ keccak256(witness.extra)));
```

The keccak256 precompile costs 30 gas base + 6 gas per 32-byte word. A malicious client can craft a payment payload with an arbitrarily large extra field (e.g., several hundred kilobytes), sign it, and submit it to the facilitator.

Since the facilitator must pass the exact extra value that was signed to reconstruct a valid witness hash, they are forced to:

1. Relay the bloated calldata (16 gas per non-zero byte, 4 gas per zero byte).
2. Pay for the keccak256 computation over the full extra length.

In an automated facilitator system that does not pre-validate payload sizes, a malicious client can grief the facilitator into spending significantly more gas than expected, while the actual payment amount remains

minimal. The facilitator bears the cost because they are the transaction broadcaster per the X402 protocol design.

**Recommendation:** Consider enforcing a maximum length for `witness.extra` in the `_settle()` function.

**Coinbase:** Fixed in commit [0ff7cffbd](#). Removed the extra bytes field from Witness entirely, along with its keccak256 hashing, since it served no on-chain purpose and allowed arbitrarily large calldata to grief facilitators.

**Cantina Managed:** Verified fix.

### 3.1.3 Absence of zero-amount validation allows no-op settlements that consume nonces

**Severity:** Low Risk

**Context:** [x402BasePermit2Proxy.sol#L141](#)

**Description:** Neither `x402BasePermit2Proxy._settle()` nor the derived contracts (`x402ExactPermit2Proxy`, `x402UptoPermit2Proxy`) validate that the settlement amount is non-zero. This has two distinct exploitation paths:

1. Exact variant - Client griebs Facilitator: A client signs a permit with `permitted.amount = 0` and submits it as payment. An automated facilitator that does not pre-validate the amount off-chain may grant the resource, settle on-chain, and receive zero tokens. The transaction succeeds, a `Settled()` event is emitted, and the facilitator bears the gas cost for a no-op transfer.
2. Upto variant - Facilitator griebs Client: A facilitator receives a valid signature from a client for amount X, but calls `settle()` with `amount = 0`. The `Permit2` nonce is permanently consumed, invalidating the client's signature. No tokens are transferred, yet the client cannot reuse that signature to pay another facilitator. The facilitator effectively burns the client's payment authorization without settling.

In both cases, the `Settled()` event is emitted, creating misleading on-chain records that suggest a successful payment occurred.

**Recommendation:** Consider adding a zero-amount check in `_settle()` to reject economically meaningless settlements.

**Coinbase:** Fixed in commit [04b844da6](#). Added `InvalidAmount` error and `amount == 0` check in `_settle()` to block economically meaningless settlements that waste gas and consume `Permit2` nonces.

**Cantina Managed:** Verified fix.

## 3.2 Informational

### 3.2.1 Replace `tx.origin` with `msg.sender` in constructor to fully support multisig deployment

**Severity:** Informational

**Context:** [x402BasePermit2Proxy.sol#L112](#)

**Description:** The `x402BasePermit2Proxy` contract uses `tx.origin` in the constructor to identify the `_deployer`, who is the only address authorized to call `initialize()`. This is problematic because the deployment strategy involves deploying and initializing it from a Multisig. `tx.origin` in this context will be the EOA that initiated the transaction, not the Multisig contract itself. Consequently, the Multisig will not be the authorized `_deployer`.

**Recommendation:** Remove the reference to `tx.origin` in the constructor, and bundle the deployment and initialization into a single multi-call transaction to prevent front-running.

**Coinbase:** Fixed in commits [7e154ff](#) and [0042512](#). For deployment reasons we've removed `initialize` and moved the `permit2` assignment to the constructor.

**Cantina Managed:** Verified fix.

### 3.2.2 Non-standard variable naming

**Severity:** Informational

**Context:** `x402BasePermit2Proxy.sol#L26`

**Description:** The variable `PERMIT2` is capitalized like a constant but it is a state variable set during `initialize()`. Per the [Solidity style guide](#), it should use lowercase (e.g., `permit2`).

**Recommendation:** Consider making `PERMIT2` lowercase.

**Coinbase:** Fixed in commit `2ce095d`.

**Cantina Managed:** Verified fix.

### 3.2.3 Lack of observability for EIP-2612 permit failures

**Severity:** Informational

**Context:** `x402BasePermit2Proxy.sol#L190`

**Description:** The `_executePermit()` function in `x402BasePermit2Proxy.sol` wraps the EIP-2612 `permit()` call in a try/catch that silently discards all errors. This design is intentional - if the `Permit2` allowance already exists, the EIP-2612 permit is unnecessary and its failure is harmless. The subsequent `_settle()` call will succeed regardless. However, this creates two observability problems:

1. Obscured root cause on revert: When the EIP-2612 permit fails and no prior `Permit2` allowance exists, the transaction reverts inside `permitWitnessTransferFrom()` with a generic `Permit2 INSUFFICIENT_ALLOWANCE` error. The actual root cause - an invalid EIP-2612 signature, an expired permit deadline, or a non-compliant token - is lost. Facilitators and off-chain monitoring systems cannot distinguish between "token lacks `Permit2` allowance" and "EIP-2612 permit was submitted but rejected."
2. Silent success ambiguity: When the transaction succeeds, there is no way to determine on-chain whether the EIP-2612 permit was actually executed or silently skipped. The `SettledWithPermit()` event is emitted in both cases, making it indistinguishable whether the allowance was freshly granted via EIP-2612 or pre-existed. This hinders debugging, monitoring, and accounting.

**Recommendation:** Emit a distinct event on permit failure to preserve the error context without reverting the transaction.

**Coinbase:** Added more granular error checking in commit `9e95d29`.

**Cantina Managed:** Verified fix. However, if the token is malicious, a gas-greiving attack on the facilitator can occur via a return data bomb. Facilitators should consider this risk before relaying signed user transactions.

### 3.2.4 Ambiguous parameter naming in `_settle()` function

**Severity:** Informational

**Context:** `x402BasePermit2Proxy.sol#L143`

**Description:** In `x402BasePermit2Proxy`, the parameter name `amount` in `_settle` could be confused with the `permitted.amount` in the `permit` struct.

**Recommendation:** Consider renaming the parameter to a name more accurate to its role in the system.

**Coinbase:** Fixed in commit `5cddb02`.

**Cantina Managed:** Verified fix.