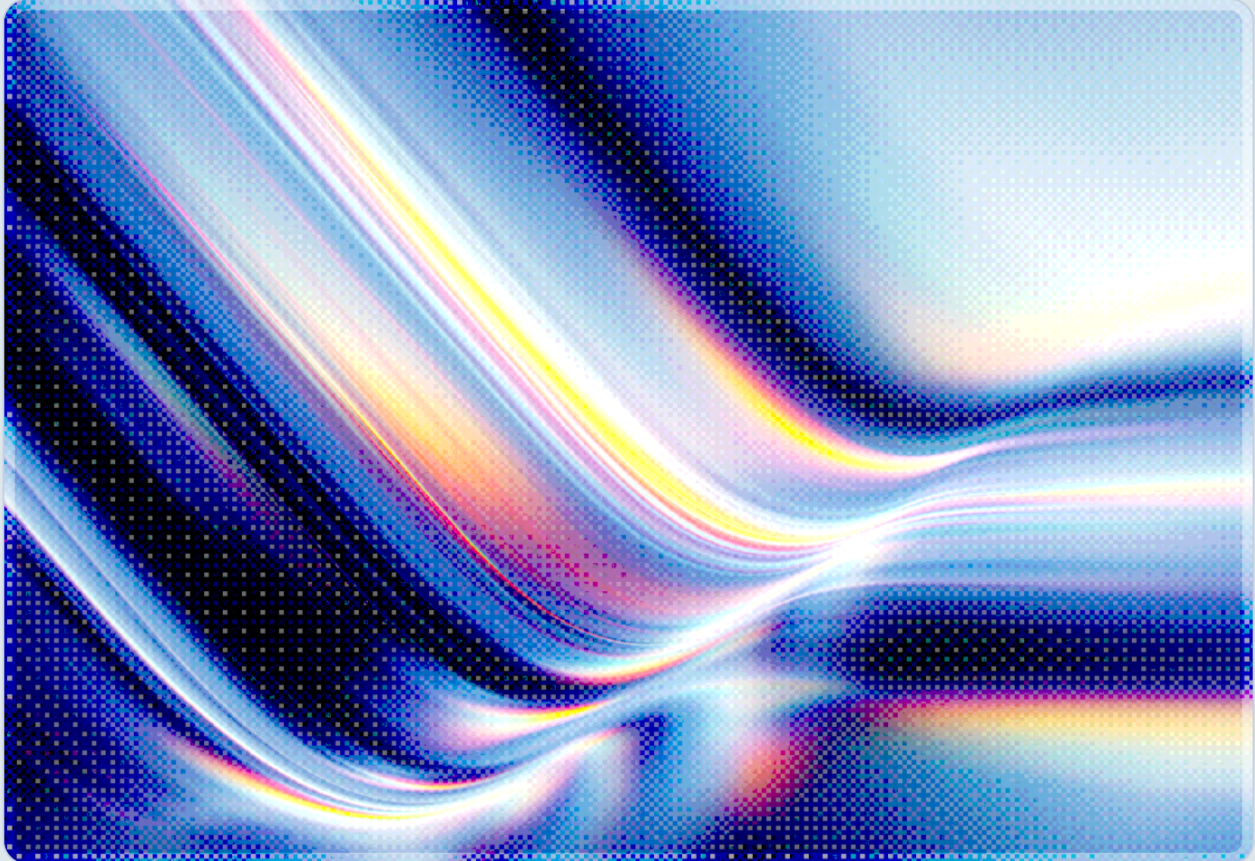


x402 Agent-Payment SDK Flaws: Uncapped Spending and Silent Fixes

Security Implications and Guidance for Autonomous Agent
Payments

2026-10-25

 AI-assisted Rapid Research



CSAI

cloud
security
alliance®

© 2026 Cloud Security Alliance. Some rights reserved.

You may download, store, display, view, print, redistribute, and link to this document in its original, unmodified form, provided that attribution to the Cloud Security Alliance is maintained and all trademark and copyright notices remain intact.

This document may not be modified or altered. You may quote portions of the document as permitted by the Fair Use provisions of the United States Copyright Act, provided that attribution is given to the Cloud Security Alliance.

This document may be shared on professional and social media platforms in its original form with attribution.

This document was generated with AI assistance and has not undergone official CSA review and approval processes.

Key Takeaways

On October 22, 2026, the security firm V12 published a review of the x402 agent-payment protocol codebase that reported 25 vulnerabilities, three of them rated critical [1]. The most consequential finding for enterprises running AI agents is that, before the August 2026 fixes, the default x402 payment client enforced no spending limit. The client selected the first payment option a server offered, and the server alone decided the amount, token, recipient, and network. A hostile API could therefore ask an automatically signing agent for the wallet's full balance of a supported token [1][2].

The second issue is how the fixes were released. The x402 Foundation published fixed releases for all three critical flaws on August 18, 2026 [2], and V12 reports that all three had been confirmed fixed by August 20 [1]. As of October 23, however, the Foundation had published no security advisory for them, and its changelogs did not label the changes as security fixes [2]. Because of this, `npm audit` and PyPI vulnerability listings did not flag the last vulnerable releases [2]. Organizations that rely on software composition analysis (SCA) to find vulnerable dependencies would not have been alerted. Neither V12 nor the Foundation has reported any exploitation in the wild [2].

Security teams should act on four points. First, agents that sign x402 payments automatically should run `@x402/fetch`, `@x402/axios`, `@x402/extensions`, and `@x402/svm` at version 2.23.0 or later, or the August 18, 2026 or later Python and Go releases, which also fixed the Sign-In-With-X domain-binding flaw in those clients [2]. Teams should also move `@x402/core` to 2.27.0 or later, which addresses a separate payment bypass fixed in September 2026 [2]. Second, the patched SDK limits each payment to \$1 and to recognized US-dollar stablecoins by default, but setting `spendControls: false` turns these limits off, so teams should search their code for that flag [2]. Third, a clean SCA report is not evidence that x402 dependencies are safe; versions need to be checked manually until advisories exist. Finally, spending authority for agents should be enforced outside the model and outside any single SDK default, with explicit budgets, allowlists, and wallet funding limits acting as independent layers of control.

Background

x402 is an HTTP-based payment protocol that gives practical meaning to the long-dormant **402 Payment Required** status code. When a client requests a paid resource, the server answers with a 402 response that lists acceptable payment requirements. The client signs a payment that meets one of those requirements, usually a stablecoin transfer, and retries the request with the payment attached. A "facilitator" service can verify and settle the payment on the merchant's behalf [1]. The design suits autonomous AI agents well: an agent that reaches a paywalled API, dataset, or tool can pay per request without human involvement or a pre-arranged account.

Coinbase created x402. In 2026 the protocol moved to neutral governance under the x402 Foundation, a Linux Foundation project. The move was announced in early April with participation from payments, cloud, and commerce companies including Adyen, AWS, American Express, Circle, Cloudflare, Google, Mastercard, Microsoft, Shopify, Stripe, and Visa [3][4]. Reporting on the disclosure notes that Coinbase still holds the most senior seat on the Foundation's technical steering committee [2]. Analysts have observed that x402 traffic runs overwhelmingly through USDC settled on Coinbase's Base network via a Coinbase-operated facilitator [5]. Because most deployments appear to use the reference SDKs, a flaw in them is likely to reach a large share of deployed agent-payment integrations.

These SDKs had already received a security advisory. GHSA-qr2g-p6q7-w82m, published March 7, 2026 and updated April 8, 2026, addressed a payment verification flaw affecting resource servers that accept Solana payments through a facilitator running an outdated SDK. The advisory stated that private keys, smart contracts, and user funds were not affected [6]. The earlier advisory shows that the GitHub advisory process was available to the project.

V12's researcher reported 3 critical and 13 high-severity findings to the x402 team on August 12, 2026. According to V12, the three critical issues were fixed by August 20, and the remaining 9 medium- and low-severity findings were submitted on September 9 [1]. V12's write-up describes only the three critical findings in detail, and the status of the other 22 has not been publicly reported [1][7]. V12 says its results come from code review and local testing with the real token encoders, parser, and Sign-In-With-Ethereum (SIWE) library. The Solana finding was not demonstrated with a live on-chain transaction, so its on-chain outcome is inferred from the token program's approval logic [1]. Readers should also know that V12 sells a commercial security agent [1].

Security Analysis

The Three Critical Findings

The three critical flaws affect different parts of the x402 trust model: the paying client, the merchant's facilitator, and the identity extension that lets agents authenticate with their wallets. Each one breaks a different assumption about who controls a transaction, and each was fixed in the August releases. The table below summarizes them as V12 and subsequent reporting described them.

Finding	Component	Root Cause	Who Is Harmed	Fixed In
No default spending limit	Payment clients (TypeScript, Python, Go)	Default client selected <code>accepts[0]</code> with no spending policy; HTTP decoding path skipped the existing schema validator	Agent operator whose wallet signs automatically	<code>@x402/fetch / @x402/axios</code> 2.23.0; August 18, 2026 Python and Go releases
Token approval accepted as payment	Solana <code>exact</code> scheme facilitator (TypeScript only, v1 and v2; Python and Go already checked the instruction type)	Verifier parsed the instruction as <code>TransferChecked</code> without checking the instruction-type byte, so <code>ApproveChecked</code> passed	Merchant and facilitator, who deliver service and pay fees without receiving tokens	<code>@x402/svm</code> 2.23.0 (2.0.0 through 2.22.0 affected)
Cross-site login signature	Sign-In-With-X (SIWx) extension (TypeScript,	Client copied <code>domain</code> and <code>uri</code> from the server's 402 response into a SIWE message and signed it	Agent operator, whose wallet identity can be used to	<code>@x402/extensions</code> 2.23.0; August 18, 2026 Python and Go releases

Finding	Component	Root Cause	Who Is Harmed	Fixed In
	Python, Go clients)	without comparing them to the contacted site	log in elsewhere	

Sources: V12 [1]; The Hacker News [2].

Uncapped Spending: Server-Controlled Authorization

For organizations deploying agents, the spending-limit flaw has the widest consequences, because it reverses the expected direction of authorization. In a typical payment flow, the payer decides how much to pay. In the pre-fix x402 default client, the 402 response from the server effectively decided the payment: the client took the first offered option and passed the server's chosen amount, token, recipient, and network to the signer [1]. V12 notes that a hostile server could first collect a small, legitimate payment to learn the wallet address, check the on-chain balance, and then request a payment equal to that balance [1]. With an automatic signer and no application-level policy, one malicious HTTP response could authorize draining a compatible token balance. The practical limits were the wallet's holdings and any restrictions applied by an external signer [1].

The exposure was conditional. It affected agents that sign payments automatically and whose applications did not add their own spending controls [2]. In CSA's assessment, however, defaults are likely to carry over into quick-start code, prototypes, and agent frameworks that wrap the SDK with little additional code. Although no source reports this attack chain, an agent that can be steered toward attacker-controlled URLs could pay any server it was induced to call, whether through prompt injection in retrieved content, poisoned tool listings, or manipulated search results. The 402 handshake therefore turns ordinary URL-following into a spending decision. Without an independent cap, the agent's exposure can extend to the full balance of any supported token held in a wallet it signs from automatically.

V12 also found that the HTTP decoding path parsed incoming payment requirements without calling the SDK's existing schema validator, and malformed requirements reached the payment selector during testing [1]. Defensive code that existed in the codebase was not on the path that untrusted input actually took. Reviewers of agent tooling should check for this pattern elsewhere: validation logic that is present but not used on the code path that processes untrusted input.

Payment Bypass and Identity Relay

Where the spending flaw harms agents, the Solana facilitator flaw harms merchants. The `ApproveChecked` and `TransferChecked` instructions in the SPL token program share a data layout and differ only in their first byte (`0x0d` versus `0x0c`). Because the verifier did not check that byte, a transaction that only approved a delegate, and moved no tokens, passed verification with `isValid: true` . The settlement path also skipped its post-settlement payment check [1]. V12 called this "a complete payment bypass" [2]. The attacker needed a token account but no balance, and the facilitator paid the transaction fee. Repeated use could therefore drain both the merchant's paid resources and the facilitator's fee budget, although V12 inferred this outcome rather than demonstrating it on-chain [1].

The SIWx flaw shows a familiar weakness in wallet-based authentication. The automatic wrapper filled the SIWE message's `domain` and `uri` fields from the 402 response instead of from the site the client had actually contacted [1]. A malicious data server could obtain a fresh login nonce from a target site, place that challenge in its own 402 response, collect the agent's signature, and replay it to the target. The target's own checks would pass, because the signed message correctly named the target [1]. The attacker could therefore authenticate as the agent's wallet identity on any site whose SIWE challenge it can relay within the nonce's validity window, regardless of the price of the original request. The domain-binding check that SIWE depends on is only effective if the client performs it before signing. The August 18 releases make the TypeScript, Python, and Go clients refuse to sign unless the site in the message matches; the SIWx specification was also changed, and code that calls `createSIWxPayload` directly must now pass the 402 response URL [2].

Silent Fixes and the Limits of Automated Dependency Scanning

The way the fixes were disclosed is a separate risk from the flaws themselves. According to reporting on October 23, the Foundation's changelogs listed the August changes without identifying them as security fixes, and no advisory was published for any of the three critical issues [2]. With no advisory filed, `npm audit` and PyPI's vulnerability listings showed the last vulnerable releases as clean [2]. Advisory-driven tools such as GitHub Dependabot work the same way, because they match dependencies against advisory databases. An organization that had pinned a pre-August version and relied on automated scanning to prompt upgrades had no security signal from its tooling for roughly two months, until V12's publication.

This report does not assume the Foundation's reasons, and the sources reviewed do not include a response from the Foundation. Maintainers sometimes delay advisories to give downstream users time to upgrade, or because the fix also changes default behavior. The 2.23.0 change does introduce new defaults, a \$1 per-payment cap and a stablecoin allowlist, that applications can disable with

`spendControls: false` [2]. Teams that upgraded for unrelated reasons may have noticed changed behavior without understanding its security purpose, and could have disabled the new limits to restore the old behavior without realizing they were removing a security control. Silent fixes in payment-capable agent infrastructure therefore leave users with incomplete information on two counts: they do not know they are exposed, and they do not know why the new defaults matter.

Version Reference

The table below consolidates the version information available as of October 23, 2026. Public reporting gives the patched Python and Go releases only by their August 18 release date [2]. PyPI's release history shows `x402` 2.20.0 as the only Python release published that day [14], although CSA has not independently confirmed its patch contents. No specific fixed version could be determined for the Go module, so teams should confirm both against the Foundation's release history.

Package	Vulnerable	Fixed	Latest Reported (Oct 23, 2026)
<code>@x402/fetch</code> , <code>@x402/axios</code>	Before 2.23.0	2.23.0	2.28.0
<code>@x402/extensions</code> (SIWx)	Before 2.23.0	2.23.0	2.28.0
<code>@x402/svm</code> (Solana facilitator)	2.0.0 through 2.22.0	2.23.0	2.28.0
<code>@x402/core</code> (separate payment bypass)	Before 2.27.0	2.27.0 (September 2026)	—
Python <code>x402</code> client (spending limit, SIWx)	Releases before August 18, 2026	August 18, 2026 release (2.20.0 per PyPI)	2.25.0
Go module <code>github.com/coinbase/x402/go</code>	Releases before	August 18, 2026 release	v2.28.0

Package	Vulnerable	Fixed	Latest Reported (Oct 23, 2026)
(spending limit, SIWx)	August 18, 2026		

Sources: *The Hacker News* [2]; *PyPI release history* [14]. The Go fixed version number is not specified in public reporting.

Recommendations

Immediate Actions

Organizations should first inventory every service, agent, and agent framework that depends on any x402 package, including transitive dependencies pulled in by agent toolkits. Each instance should be upgraded to the fixed versions above or later, including `@x402/core` 2.27.0 or later. Because advisory-driven scanners will not flag the vulnerable releases, this inventory has to compare version numbers directly rather than rely on SCA output. Teams should also search their codebases and configuration for `spendControls: false` and for custom payment selectors that bypass the new policy layer [2]. Where an upgrade cannot happen right away, The Hacker News reports that teams can register a payment policy that rejects options above a fixed amount, using `registerPolicy` in TypeScript and Go or `register_policy` in Python [2].

Teams should also reduce the value at risk. Wallets that agents can sign from should hold only the working balance the agent needs for a short operating period. Facilitator operators accepting Solana payments should confirm they run `@x402/svm` 2.23.0 or later and should reconcile recent settlements against actual token receipts to detect approval-only transactions [1][2].

Short-Term Mitigations

Over the next several weeks, teams should move spending authority from SDK defaults to explicit, organization-owned policy. V12 recommends a small set of checks that must run before the client signs, and CSA extends several of them to fit enterprise agent deployments. The table below separates V12's guidance from CSA's additions.

Control	V12 Recommendation [1]	CSA Addition
Spending budgets	Explicit per-payment and total budgets	Scope cumulative budgets per agent and per time window
Allowlists	Permitted tokens, networks, and recipients	Where feasible, also allowlist merchant domains
Input validation	Schema validation of every incoming 402 payment requirement before it reaches the selector	–
SIWx domain binding	Check that the message's domain and uri match the origin the client actually contacted	–

These controls work best in a component the agent's reasoning cannot override, such as a signing proxy, policy engine, or custodial wallet service with its own limits. In-process configuration that a prompt-injected agent could influence is less reliable. Teams should log every 402 challenge and every signed payment with the requesting URL, amount, and recipient, and should alert on payments to new recipients or on spending rates that exceed a baseline. Agent-initiated payments should also be added to existing fraud and treasury monitoring rather than treated as an engineering-only telemetry stream.

Strategic Considerations

In CSA's view, the x402 disclosure shows that agent payment rails need the same disclosure discipline as other financial infrastructure. Organizations that rely on x402 should ask the x402 Foundation and its member companies to publish advisories, and request CVE identifiers, for the three August fixes and for any of the remaining 22 findings that have security impact. They should also ask the Foundation to commit to a published security policy that labels security fixes as such. Procurement and vendor-risk teams evaluating agent platforms that embed x402 should ask how the vendor tracks upstream fixes when advisories are missing.

More broadly, the case argues for treating every agent payment capability as a delegated financial authority. That authority needs an accountable owner, a defined limit, and a revocation path. A default that lets the counterparty set the price is not compatible with that model, regardless of which protocol or SDK carries the transaction. As agentic commerce protocols multiply, CSA considers the main defense to be an architectural rule: the payer's policy decides what is paid, and the payee's request is only a proposal.

CSA Resource Alignment

CSA's research note *AI Agent Identity Sprawl: The Enterprise Authorization Crisis* [8] describes agent identity sprawl as a largely unaudited risk surface and argues that authorization for agents is often broader than any human reviewer would approve. The x402 spending flaw is a concrete instance of that problem: an agent identity held unbounded financial authority, and the remote party, not the enterprise, effectively decided how it was used. The SIWx relay flaw extends the same concern to the agent's wallet identity, which could be used to authenticate to sites the operator never intended. The controls that note recommends, including scoped credentials, least privilege, and auditable authorization decisions, apply directly to agent wallets.

The silent-fix aspect of this disclosure relates to CSA's *AI Coding Agents: An Unaudited Supply Chain Node* [9], which recommends extending software supply chain controls such as SBOMs, provenance, and dependency governance to agent tooling. The x402 case shows a gap those controls must account for: dependency governance that depends only on advisory feeds can miss upstream security changes, so SBOM-driven inventories need to support direct version-range queries against vendor and researcher disclosures. CSA's related note *Derived Keys, Spent Funds: An AI Model's Unauthorized Wallet Use* [10] covers the complementary risk of an agent spending funds it was never authorized to use. Read together with this note, it indicates that wallet access for AI systems needs controls on both the agent side and the payment-protocol side.

CSA's earlier guidance *Securing LLM Backed Systems: Essential Authorization Practices* [11] states that authorization decisions should be enforced by deterministic systems outside the language model. The recommendation in this note to place spending limits in a signing proxy or policy engine applies that principle to payments. At the control-framework level, the AI Controls Matrix (AICM) v1.1 [12] provides the mapping for audit and assurance: its Identity and Access Management controls cover delegated agent authority, its Threat and Vulnerability Management controls cover remediation of upstream components, and its Supply Chain Management, Transparency, and Accountability controls cover supplier disclosure obligations. Teams threat-modeling agentic payment flows can use MAESTRO [13] to examine trust boundaries between the agent, the tools and APIs it calls, and the payment layer. The 402 handshake is one of those boundaries.

References

- [1] Renwa, V12. "[25 Vulnerabilities in the x402 Protocol: Inside Three Critical Findings](#)." V12 Blog, October 22, 2026.
- [2] Khandelwal, Swati. "[Critical x402 SDK Flaw Could Let Malicious APIs Drain AI Agents' Token Balances](#)." The Hacker News, October 23, 2026.
- [3] PYMNTS. "[Coinbase Partners Linux Foundation to Build Internet-Native Payment Layer](#)." PYMNTS, April 2, 2026.
- [4] Decrypt. "[Coinbase Links Up With Linux Foundation to Launch x402 Foundation](#)." Decrypt, April 2026.
- [5] BIIA. "[x402 Foundation Launches Under Linux Foundation With 40 Members, Including Google and AWS](#)." BIIA, April 9, 2026.
- [6] GitLab Advisory Database. "[GHSA-gr2g-p6g7-w82m: x402 SDK Security Advisory](#)." GitLab, March 7, 2026 (updated April 8, 2026).
- [7] Adyog Pulse. "[x402 Default Let Servers Ask Unprotected AI Agents for a Full-Balance Payment](#)." Adyog, October 23, 2026.
- [8] Cloud Security Alliance. "[AI Agent Identity Sprawl: The Enterprise Authorization Crisis](#)." CSA Lab Space, June 20, 2026.
- [9] Cloud Security Alliance. "[AI Coding Agents: An Unaudited Supply Chain Node](#)." Cloud Security Alliance, July 8, 2026.
- [10] Cloud Security Alliance. "[Derived Keys, Spent Funds: An AI Model's Unauthorized Wallet Use](#)." CSA Lab Space, October 20, 2026.
- [11] Cloud Security Alliance. "[Securing LLM Backed Systems: Essential Authorization Practices](#)." Cloud Security Alliance, 2024.
- [12] Cloud Security Alliance. "[AI Controls Matrix \(AICM\) v1.1](#)." Cloud Security Alliance, 2025.
- [13] Cloud Security Alliance. "[Agentic AI Threat Modeling Framework: MAESTRO](#)." Cloud Security Alliance, February 6, 2025.
- [14] Python Package Index. "[x402: Release History](#)." PyPI, accessed October 25, 2026.