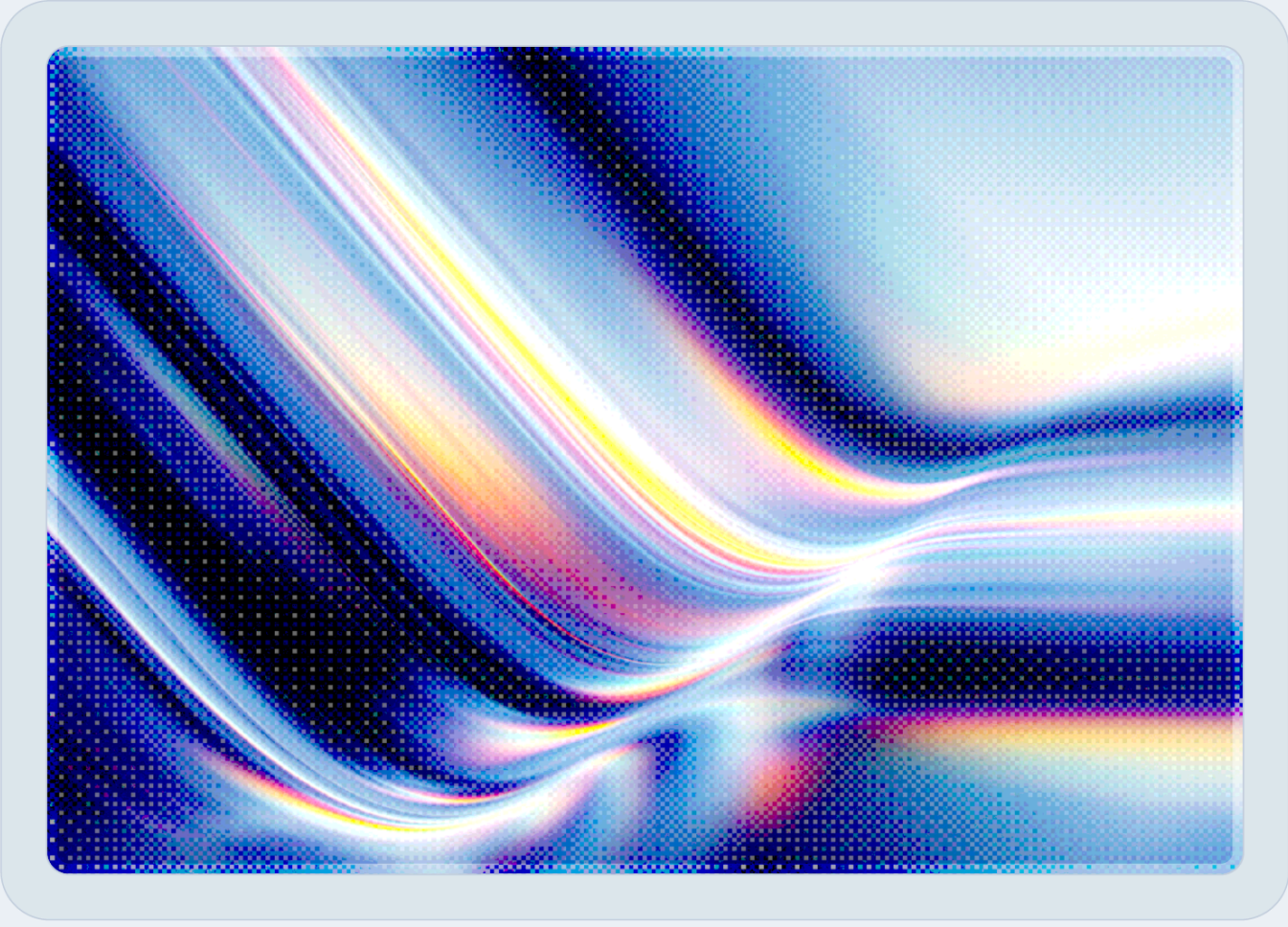
CSAI Foundation | Cloud Security Alliance

XRING: Unpatched XQUIC Flaw Crashes HTTP/3 Servers

A No-CVE Zero-Day in Alibaba's QUIC Library Exposes Global HTTP/3 Deployments

2026-07-11

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- FoxIO researcher Sébastien Féry publicly disclosed a denial-of-service flaw, nicknamed XRING, in XQUIC, the open-source QUIC and HTTP/3 library maintained by Alibaba, on July 8, 2026. A single crafted burst of roughly 260 bytes of fully spec-compliant QPACK traffic — no authentication and no malformed packets required — crashes any vulnerable server process [1][2].
- The defect sits in XQUIC's ring-buffer implementation of the QPACK dynamic table. When a client requests that the table grow, the library computes the size of the data that must be preserved against the new buffer capacity rather than the old one, producing an unsigned integer underflow that drives a memory copy past the end of the allocated buffer [1].
- As of this writing there is no CVE identifier, no CVSS score, and no patched release. Every published XQUIC version through v1.9.4 — covering the library's entire public history since January 2022 — remains vulnerable [1][2].
- FoxIO notified Alibaba on April 7, 2026, through the channel XQUIC's own security policy designates, which commits to a reply within three business days, and followed up four more times through May 9 without receiving any response, prompting disclosure after roughly 90 days of silence [1][2].
- XQUIC is embedded well beyond Alibaba's own infrastructure: it underpins Tengine, Alibaba's Nginx-derived web server that fronts services including Taobao and Alipay, and any third-party product that vendors the library with default QPACK settings inherits the same exposure without necessarily knowing it does [1][2].
- Operators cannot wait for a vendor patch. The available compensating controls — disabling HTTP/3 entirely or setting `SETTINGS_QPACK_MAX_TABLE_CAPACITY` to zero to turn off the QPACK dynamic table — should be applied immediately on any internet-facing service that embeds XQUIC [1][2].

Background

HTTP/3 and its transport layer, QUIC, have become the default target for new web infrastructure over the past several years, prized for eliminating head-of-line blocking and for folding TLS setup into the connection handshake. That performance case has driven rapid adoption across CDNs, cloud load balancers, and hyperscaler edge networks, and it has done so faster than the ecosystem of implementations backing it has had time to mature under adversarial scrutiny. QUIC and HTTP/3 are still young enough that production deployments are widely believed to be concentrated on a small number of shared libraries rather than bespoke implementations, which would mean that a memory-safety defect in any one of those libraries carries a blast radius measured in internet-scale infrastructure rather than in a single vendor's product line.

XQUIC is one of those shared libraries. Alibaba released it as open source in 2022 as a cross-platform implementation of QUIC and HTTP/3, and it has since been folded into Tengine, the Nginx-derived web server that Alibaba uses to front large consumer platforms including Taobao and Alipay [1][2]. Because XQUIC is published on GitHub under a permissive license, it is also available to any third party that wants HTTP/3 support without building a QUIC stack from scratch — a category that, by the nature of open-source redistribution, cannot be fully enumerated from the outside. This illustrates a recurring pattern in concentrated, hard-to-see risk: a component few security teams have heard of, embedded inside products they use every day, whose defects do not show up in the vendor advisory feeds most organizations monitor because no advisory has been issued.

That is the situation FoxIO surfaced on July 8, 2026. Researcher Sébastien Féry found the defect, which he nicknamed XRING, while developing FoxIO's JA4Scan tool, and demonstrated that it could crash an XQUIC-based HTTP/3 server using nothing more than a short sequence of protocol-legal QPACK instructions [1][2]. No exploitation primitive beyond normal client behavior was required, which is what makes this, in our assessment, a live operational risk rather than a theoretical parsing bug: any client capable of opening an HTTP/3 connection can trigger it, and the traffic involved would not be flagged as malformed by any conventional intrusion detection signature tuned to look for protocol violations.

Security Analysis

The Ring-Buffer Miscalculation

QPACK is the header-compression scheme HTTP/3 uses in place of HPACK, the scheme HTTP/2 relies on. Its defining feature relative to HPACK is a dynamic table that both endpoints maintain to reference previously seen header fields, letting subsequent requests replace repeated header values with short table references instead of retransmitting the full text. Because QUIC streams can arrive out of order, QPACK's dynamic table update messages travel on a dedicated encoder stream, and the receiving endpoint must apply size changes to that table exactly as the protocol specifies or risk desynchronizing the compression state between client and server.

XQUIC stores the bytes backing that dynamic table in a ring buffer: a fixed block of memory in which data wraps from the end back to the beginning once the buffer fills, avoiding the cost of repeatedly reallocating memory as entries are added and evicted. Growing that table is where the defect lives. When a client sends an instruction to increase the dynamic table's capacity, XQUIC must first preserve the table's existing "tail" — the live entries not yet evicted — before resizing the underlying buffer to the new capacity. FoxIO's analysis found that the library computes the amount of tail data to preserve using the new buffer's capacity rather than the old buffer's, and because that computation is performed with an unsigned integer type, a resize request engineered to make the calculation go negative instead wraps around to a very large positive number [1]. The subsequent memory copy operation, instructed to move an enormous number of bytes into a target buffer far smaller than that value, writes past the end of the allocation.

FoxIO characterizes the underlying error as a single incorrect variable on one line of code — the routine references the new buffer's size where it should reference the old one — which is consistent with the kind of subtle off-by-reference defect that unit tests built around common-case table growth would not catch, since it only manifests when growth and eviction timing align in a specific way [1]. FoxIO's testing was conducted on a system running Ubuntu 26.04 with glibc's `_FORTIFY_SOURCE=2` hardening enabled, which caught the malformed copy length and killed the process before further damage occurred; without that hardening in place, the same defect writes past the end of the old buffer rather than stopping cleanly [1]. FoxIO's public disclosure does not claim to have established that the underlying out-of-bounds write is exploitable for arbitrary code execution, but the crash-only outcome should not be assumed to generalize to unhardened or embedded XQUIC builds that lack this compiler and libc protection [1]. That distinction matters for how organizations should prioritize the finding across a fleet that may include both hardened and unhardened deployments, and it is addressed further below.

Trigger Simplicity and Detection Difficulty

The most operationally significant fact about XRING is how little is required to trigger it. FoxIO reports that the crash follows from approximately 260 bytes of ordinary QPACK traffic, requires no authentication, and involves no malformed or out-of-specification packets [1][2]. Every byte an attacker sends is protocol-legal; the defect is purely in how the server's internal state management handles a legal but specific sequence of instructions. This has two consequences for defenders. First, it means the flaw cannot be filtered by any signature-based detection tuned to reject malformed HTTP/3 or QUIC frames, since the traffic in question conforms fully to the wire format. Second, it means the attack requires no privileged position, no prior reconnaissance beyond confirming the target speaks HTTP/3 over XQUIC, and no more infrastructure than a single client capable of opening a QUIC connection — placing it well within reach of any remote, unauthenticated actor.

Scope: Beyond Alibaba's Own Estate

Coverage of XRING has understandably focused on Alibaba's own exposure, since Tengine — carrying the defect into services as large as Taobao and Alipay — is a documented, named instance of the vulnerable code path in production [1][2]. We consider the more consequential exposure for the broader internet to be structural rather than specific to one company. XQUIC is redistributed as an open-source component, and the population of organizations that have vendored it into their own HTTP/3 stacks is not visible from the outside. Every release through the current v1.9.4, spanning the library's full public history back to January 2022, carries the defect, and any deployment that enabled QPACK's dynamic table with its default configuration — which is to say, essentially any deployment that has not specifically hardened this setting — is exposed regardless of whether its operators have ever heard of XQUIC by name [1][2]. This is the recurring shape of software supply chain risk in shared networking libraries: the number of organizations that need to act is larger than the number that will receive a targeted notification, because most of them do not know they are running the affected code.

A No-CVE Zero-Day, and What That Means for Risk Management

XRING is, as of this writing, unassigned a CVE identifier and carries no CVSS score [1][2]. For organizations whose vulnerability management programs are built around ingesting CVE feeds and prioritizing by CVSS threshold, that absence means the flaw will not appear in a standard feed, will not trip an automated severity gate, and will not surface in a scanner that keys its findings to the National Vulnerability Database. This is not a new problem in the abstract — security teams have long known that CVE assignment lags disclosure — but 2026 has seen that lag become more pronounced as an industry-wide phenomenon. Independent reporting this year has documented that advisory-review pipelines at major clearinghouses, including the GitHub Advisory Database, are straining under record submission

volume, with review backlogs stretching to weeks even for reports that are ultimately validated [3]. XRING's current state — a technically detailed, researcher-confirmed defect with public reproduction steps, sitting outside the CVE system entirely because the vendor has not engaged — is one illustration of a pattern organizations should expect to see more often: real, exploitable risk that predates, and may never receive, the identifier their tooling is built to watch for.

The disclosure timeline itself is worth holding up against Alibaba's own published security policy for XQUIC, which commits to an initial response within three business days. FoxIO's report went unanswered from April 7 through May 9 despite four follow-up attempts, a gap of roughly a month beyond the vendor's own stated window before the researcher moved to public disclosure at the 90-day mark [1][2]. Organizations that maintain their own coordinated-disclosure programs for internally developed or acquired software should treat this as a cautionary data point: a stated SLA that goes unenforced from the inside erodes researcher trust, and plausibly compounds a separate, documented pressure on the disclosure ecosystem this year — advisory pipelines straining under record submission volume [3] — by giving researchers less reason to wait on vendors who do not meet their own timelines.

Precedent: This Is Not the First HTTP/3 Memory-Safety Bug

XRING is not an isolated case of a QUIC or HTTP/3 implementation failing under crafted-but-legal input. In February 2024, NGINX disclosed CVE-2024-24989, a null-pointer dereference in its own HTTP/3 QUIC module that allowed undisclosed requests to crash worker processes, carrying a CVSS v3.1 base score of 7.5 [4]. That defect was patched in NGINX 1.25.4 within roughly the same window as its disclosure, standing in sharp contrast to XRING's unresolved status many weeks past initial vendor notification. The comparison underscores that memory-safety bugs in HTTP/3 header-compression and connection-state handling are a recurring class of defect across independent implementations, not a one-off coding error specific to XQUIC — suggesting that vendor responsiveness, more than the underlying bug class, may be the variable organizations have the most leverage to demand from their suppliers.

Recommendations

Immediate Actions

Organizations operating any HTTP/3-capable server should first determine whether XQUIC is present in their stack, directly or vendored inside a third-party product, since the library's redistribution makes this a non-obvious question that may require checking with CDN, load-balancer, and web-server vendors

directly rather than assuming the answer from a product name alone. Where XQUIC is confirmed or suspected, apply one of the two available compensating controls without waiting for a vendor patch: set `SETTINGS_QPACK_MAX_TABLE_CAPACITY` to zero to disable the QPACK dynamic table, which removes the specific growth-and-resize code path the defect depends on, or disable HTTP/3 support outright and fall back to HTTP/2 until a fixed release is available [1][2]. Given that no patch exists and no CVE tracks the issue, security teams should not expect a scanner alert to surface this finding and should instead treat manual verification of XQUIC exposure as a standalone task for this disclosure cycle.

Short-Term Mitigations

Because the trigger traffic is fully protocol-compliant, signature-based detection is not a reliable control; organizations should instead monitor for the operational symptom — unexpected worker or server process restarts correlated with HTTP/3 connections — as an indirect indicator that the defect is being triggered, whether accidentally or by a probing actor. Any organization that maintains its own fork or vendored copy of XQUIC should track the upstream GitHub repository directly for a fix, since the absence of a CVE means standard advisory feeds will not carry that signal. Organizations that operate their own bug bounty or coordinated-disclosure program should also use this incident as a prompt to audit whether their own stated response SLAs are actually being met in practice, given how directly Alibaba's unmet three-day commitment shaped the timeline to public disclosure here.

Strategic Considerations

XRING is a useful forcing function for revisiting software composition visibility specifically for networking and transport-layer libraries — a category that, anecdotally, receives less software bill of materials scrutiny than application-level dependencies despite sitting directly on the internet-facing attack surface. Organizations should treat the inability to quickly answer "do we run XQUIC, anywhere, including inside vendor products" as a gap worth closing independent of this specific disclosure. More broadly, security leaders should recalibrate any vulnerability management process that gates prioritization strictly on CVE assignment or CVSS score, since 2026's advisory-pipeline strain means an increasing share of credible, actively reproducible defects will sit outside that system for extended periods, as XRING currently does.

CSA Resource Alignment

XRING's combination of an unpatched transport-layer defect, an embedded open-source dependency, and no CVE tracking connects to several areas of CSA's published guidance.

The CSA AI Controls Matrix (AICM v1.1), which extends and supersedes the Cloud Controls Matrix, is the most directly applicable framework for the vulnerability management dimension of this disclosure [5]. Its Threat and Vulnerability Management domain speaks to exactly the scenario XRING presents: a confirmed defect for which no vendor-supplied fix or CVE tracking exists, requiring organizations to apply compensating controls and to track remediation independently of standard advisory feeds. Its Application and Interface Security domain covers the underlying input-handling failure — an unsigned integer underflow driven by trusting client-supplied resize parameters — that a rigorous secure development lifecycle for protocol-parsing code is intended to catch before release.

CSA's broader guidance on software transparency and digital supply chain security connects to this disclosure as well, though CSA's existing published material in this area has focused more on application-layer dependency tracking than on transport-layer or networking-infrastructure libraries specifically [6]. XQUIC's redistribution as an embedded library inside Tengine and an unknown population of third-party HTTP/3 stacks nonetheless illustrates the same visibility gap that software bill of materials practices exist to close: organizations that cannot enumerate which of their internet-facing services depend on XQUIC cannot assess their exposure to this disclosure — a gap this disclosure helps illustrate even where CSA's existing guidance has not yet addressed transport-layer libraries as a distinct category.

Finally, CSA's Zero Trust guidance and its published work on network-hiding approaches to zero trust network infrastructure bear on the mitigation side of this disclosure [7][8]. An HTTP/3 endpoint that is reachable only from an allowlisted set of clients, or that is fronted by a network-hiding layer that limits which sources can complete a connection at all, narrows the population of actors capable of sending the 260-byte trigger sequence in the first place. Organizations that have already adopted software-defined perimeter or equivalent zero-trust network access controls for their internet-facing services have a meaningful head start in reducing exposure to XRING while a permanent fix remains unavailable; those that have not should treat this disclosure as one more argument for reducing the population of unauthenticated clients that can reach protocol-parsing code directly.

References

[1] The Hacker News. "Unpatched XRING Flaw in XQUIC Lets Remote Clients Crash HTTP/3 Servers." The Hacker News, July 10, 2026.

[2] Recon Bee. "Unpatched XRING Flaw in XQUIC Lets Remote Clients Crash HTTP/3 Servers." Recon Bee, July 2026.

[3] Help Net Security. "Vulnerability reports are arriving faster than GitHub can review them." Help Net Security, June 30, 2026.

[4] NVD. "CVE-2024-24989 Detail." NIST National Vulnerability Database, February 2024.

[5] Cloud Security Alliance. "AI Controls Matrix (AICM) v1.1." Cloud Security Alliance, June 22, 2026.

[6] Cloud Security Alliance. "Software Transparency: Securing the Digital Supply Chain." Cloud Security Alliance, accessed July 2026.

[7] Cloud Security Alliance. "Zero Trust Working Group." Cloud Security Alliance, accessed July 2026.

[8] Cloud Security Alliance. "Stealth Mode SDP for Zero Trust Network Infrastructure." Cloud Security Alliance, accessed July 2026.

