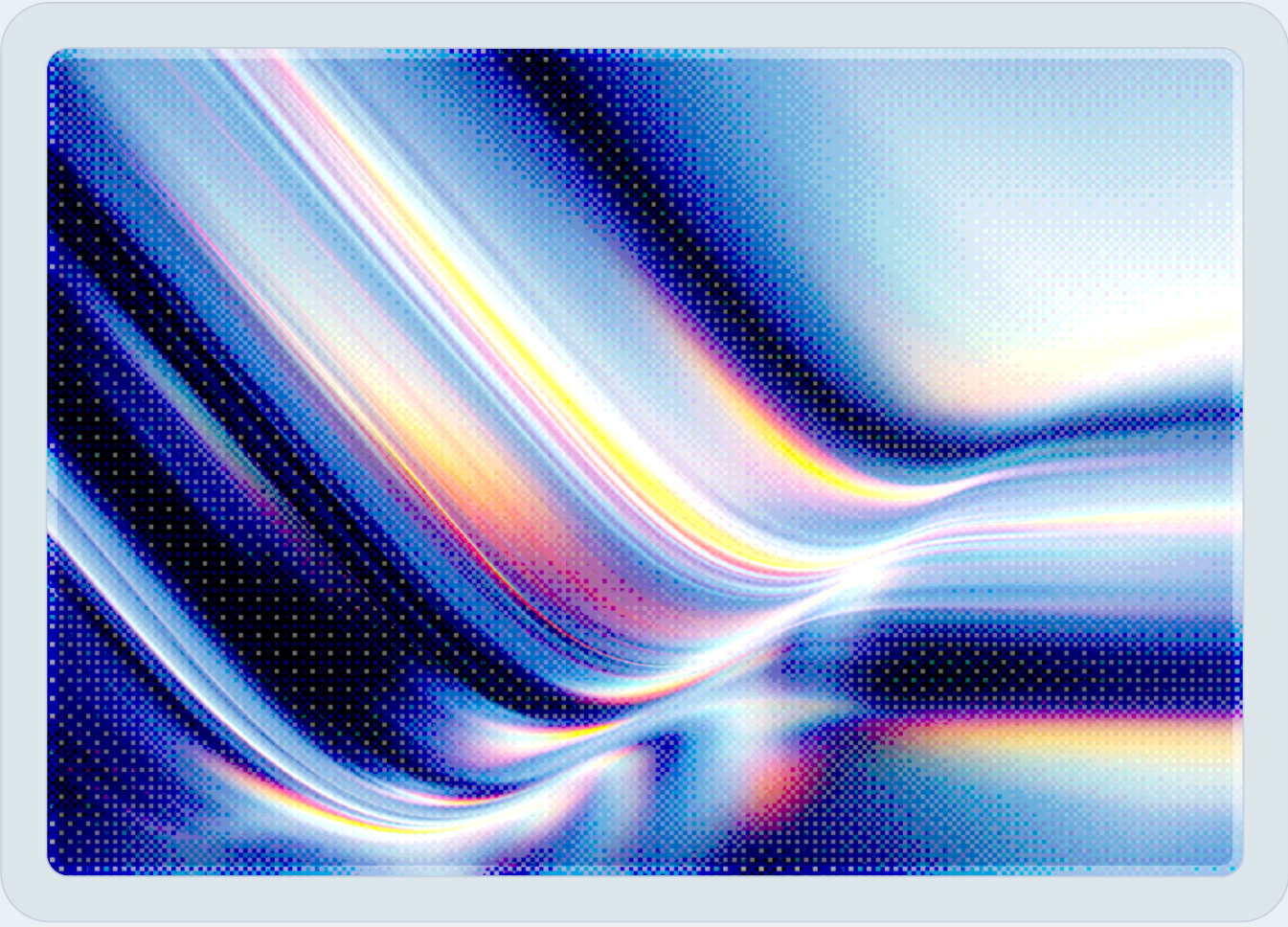
CSAI Foundation | Cloud Security Alliance

nginx2root: AI-Found HTTP/2 RCE and the Container-to-Root Chain

Security Implications and Guidance for Five nginx RCE Flaws and an Unpatched Linux SCTP Bug

2026-11-20

AI-assisted Rapid Research



CSAI

CSA 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 November 18, 2026, F5 shipped an out-of-band nginx release fixing ten vulnerabilities [11]. Five of them are memory-safety flaws that researchers at Xint (Theori) grouped under the name "nginx2root." Each can lead to remote code execution in the nginx worker process, and all five carry a CVSS 4.0 score of 9.2 [2][3]. The most serious, CVE-2026-123007, is an arithmetic error in HTTP/2 flow control that has been in every nginx release since HTTP/2 support arrived in version 1.9.5 in 2015. An unauthenticated remote attacker can reach it against a server that does nothing more than serve static files [1][5][10]. Xint reports that one of its researchers found CVE-2026-123007 by hand, and that a fully automated Xint scan then found the other four "without human guidance" a few days later [1]. Wiz says it found both HTTP/2 flaws with its own AI code-scanning harness [5][10]. That makes nginx2root one of the most prominent cases so far of AI-assisted analysis surfacing critical bugs in widely deployed, heavily reviewed infrastructure code.

The name refers to a second stage. Xint chained the nginx bugs with an unpatched use-after-free in the Linux kernel's SCTP subsystem. In its demonstration, that chain broke out of a Docker container running nginx and gained root on the underlying cloud host [1][5]. When this note was written, the kernel flaw had no CVE and no upstream fix [2][5].

The first priority for security teams is to patch and restart. Organizations should upgrade to nginx 1.31.9 (mainline) or 1.30.7 (stable), or to NGINX Plus 37.0.8.1 or 37.3.0.1, and then fully restart nginx, because a configuration reload keeps the vulnerable binary running [1][2]. Disabling HTTP/2 is only a partial workaround: it closes two of the five flaws, while the HTTP/3, resolver, and mail-proxy bugs each need their own mitigation [5][6]. Nor should teams count on the container boundary. Until a kernel fix ships, the SCTP module should be blocked on hosts that do not need it [4][5].

Kubernetes operators face an additional problem. The community ingress-nginx project was retired in March 2026, so it will not receive an upstream fix, and its default configuration exposes both HTTP/2 flaws [4][7]. On the threat side, no in-the-wild exploitation had been reported, and the CVEs were not in CISA's Known Exploited Vulnerabilities catalog as of November 19 [2]. A public proof-of-concept exists, but it carries significant constraints [3][13], and several teams report private exploits that do not share them.

Background

nginx is one of the most widely deployed web servers and reverse proxies on the internet. It sits at the edge of most cloud architectures: as a TLS terminator, an API gateway, a Kubernetes ingress controller, and a sidecar in container images. Cybersecurity Dive and Equixly put the number of websites running nginx at roughly 225 million [5][8]. Xint's pre-disclosure Shodan survey on November 10 found 56.7 million hosts identifying as nginx. Of the 11.94 million that exposed a version string, 98.9% fell within the affected range [1]. Tenable cites Censys data showing about nine million services running an affected version with HTTP/2 enabled, noting that the figure may include distribution builds that already carry backported fixes [2]. Because the surveys use different methods and both rely on version banners that do not reflect distribution backports, they indicate the scale of exposure rather than measure it precisely. Either way, the affected population is very large.

Several research teams reached the same bugs at about the same time. Xint's timeline lists October 11, 2026, as its discovery date and shows that it reported the flaws through F5's HackerOne program on October 16. The reports reached the nginx team on October 27, and Xint validated patches on November 10 [1]. Rapid7 credits its researcher Stephen Fewer with finding CVE-2026-123007 independently and contacting F5 on November 11 [3]. Wiz reported CVE-2026-123633 on October 21, when F5 told it another team had already filed, and reported the flow-control flaw on November 16 [5][10]. Equixly, HeroDevs, and Cybersecurity Dive also list Ikotas Labs, Metnir, and other researchers as independent reporters, though attributions differ in detail across sources [5][7][8]. HeroDevs additionally credits Palo Alto Networks Unit 42 for CVE-2026-123633 [7], and Cybersecurity Dive adds that F5's internal research contributed as well [8]. Coordinated disclosure and fixed releases followed on November 18 [11].

The AI angle matters for defenders, and so do its limits. Xint attributes four of the five findings to an autonomous scan. Shattered reports a claim by Xint CEO Brian Pak that the platform analyzed about 50,000 lines of nginx code in about 90 minutes, and notes that this is a vendor statement nobody has independently verified [9]. Wiz says it discovered both HTTP/2 flaws with Wiz Atlas, the harness behind its AI code-scanning product, and that its researchers used Gemini 4 Carbon and ChatGPT 5.6 Sol to validate their impact [10]. Xint has not named the model behind its platform, and Wiz has not named the model inside Atlas. Statements that "AI found nginx2root" therefore need care. The most defensible reading is that AI-driven analysis found most of the flaws, human researchers found at least one, and human researchers built the published exploit. The pattern continues the trend CSA described in its June 2026 analysis of the AI-discovered "HTTP/2 Bomb" denial-of-service attack [15]: protocol-handling code in mature web servers is now a productive target for machine-assisted analysis.

Security Analysis

The analysis below proceeds from the nginx flaws themselves, through their practical exploitability, to the kernel chain that turns a worker-level foothold into host compromise, and finally to the platform-specific exposure that complicates remediation. Each stage changes the risk calculation, and defenders who stop at the first stage will underestimate the impact.

The Five nginx Vulnerabilities

The five flaws fall into two classes. CVE-2026-123007 starts with a window-arithmetic error in HTTP/2 flow control. Sources describe it variously as an integer overflow or underflow, and Rapid7 and F5 classify it as CWE-191. It ends in a heap out-of-bounds write [1][3][5]. Rapid7's write-up shows that an attacker can manipulate per-stream window accounting until an unsigned subtraction wraps. The resulting out-of-bounds access lets the attacker read and overwrite memory in the worker process [3]. The other four flaws are use-after-free bugs (CWE-416). Each occurs where nginx frees a request or session object while another code path is still using it [1][4]. The table below brings together the reachability conditions reported across the disclosing researchers and secondary analyses.

CVE	Component	Bug class	Preconditions for reachability	Blocked by disabling HTTP/2?
CVE-2026-123007	HTTP/2 flow control	Integer wrap → out-of-bounds write	HTTP/2 enabled; no authentication or backend required	Yes
CVE-2026-123633	HTTP/2 connection lifecycle	Use-after-free	HTTP/2 with upstream proxying; reportedly low available worker connections	Yes
CVE-2026-123634	DNS resolver	Use-after-free	`resolver` with request-derived `proxy_pass` target plus a subrequest (e.g., `mirror`, SSI)	No

CVE	Component	Bug class	Preconditions for reachability	Blocked by disabling HTTP/2?
CVE-2026-121810	HTTP/3 / QUIC	Use-after-free	HTTP/3 enabled with upstream proxying	No
CVE-2026-113425	Mail proxy	Use-after-free	`proxy on` with `auth_http`; no valid credentials needed	No

Sources: [1], [2], [4], [5], [6]. Sources differ on the earliest affected versions of CVE-2026-123634 and CVE-2026-113425, but agree that both go back to early nginx releases.

The table shows why "turn off HTTP/2" is not a complete answer. Only the first two rows depend on the HTTP/2 module. The HTTP/3, resolver, and mail-proxy flaws stay reachable in configurations that use those features [5]. The resolver flaw deserves attention because each element of its triggering pattern, a dynamically resolved `proxy_pass` target and a subrequest, appears in API gateway and traffic-mirroring setups, so the combination is plausible in those environments. Xint warns that disabling `mirror` alone is not enough, because other subrequest-generating features such as SSI may reach the same code [1]. Equixly reports that in Xint's tests the resolver and the two upstream-proxying flaws required nginx to make outbound connections or DNS lookups that reach the attacker [5]. That is a meaningful exploitation constraint, and it suggests egress filtering may lower practical risk for those three bugs.

Exploitability in Practice

CVE-2026-123007 is the most exploitable flaw and the best documented. Rapid7 published a full unauthenticated RCE exploit alongside a minimal trigger [3][13]. The exploit has real constraints. It sends about 128 GB of HTTP/2 traffic, largely repeated WINDOW_UPDATE frames, over a single connection. Rapid7 estimates the attack takes about two minutes at 10 Gbit/s and about 17 minutes at 1 Gbit/s; below roughly 300 Mbit/s it exceeds nginx's one-hour connection limit and fails [3]. The public exploit also depends on a specific glibc `memcpy` implementation that is selected at runtime on x86-64 CPUs with AVX-512 support, and an intermediate proxy that re-frames HTTP/2 traffic will likely disrupt the attack

[3]. Tenable notes that code execution generally requires ASLR to be absent or defeated [2]. Wiz and Equixly, by contrast, state that ASLR offers little protection here, because both HTTP/2 flaws allow a memory read before control-flow hijack [5][10].

Those constraints should not be read as safety margins. Wiz reports achieving code execution, including an ASLR bypass, against official nginx 1.31.7 and 1.31.8 builds serving only static content [10]. Xint reports successful exploitation on AWS, GCP, Azure, and bare metal, including over public IP addresses [1][5]. Neither claim can be independently verified, because both organizations are withholding their exploits [2] [5]. Even so, several well-resourced teams report reaching working RCE independently. That suggests the hardware and bandwidth limits of the public exploit reflect one implementation's choices rather than the bug's inherent difficulty. Results for the use-after-free flaws are more mixed. Xint reports worker-level command execution with each against the official `nginx:1.31.8` image [5], whereas Censys reproduced three of them in the lab but did not achieve code execution [2]. Organizations should treat all five as critical while recognizing that public exploitation tooling currently exists only for CVE-2026-123007.

In every case, successful exploitation of the nginx flaws yields code execution with the privileges of the nginx worker process [1][3]. That is typically an unprivileged account such as `nginx` or `www-data`. On a conventional host the attacker gains a foothold in the edge tier, can read TLS private keys and configuration available to the worker, and can tamper with proxied traffic. Even where key files are root-owned, the keys are loaded into worker memory before the master process drops privileges, so a worker-level attacker can typically extract them from memory. The attacker does not, however, have root. The second stage of the chain is what changes this.

The Container-to-Root Chain

Xint's demonstration chains worker-level code execution from CVE-2026-123007 with a separate use-after-free in the Linux kernel's SCTP send path. According to Equixly's account of the Xint findings, an SCTP association can be migrated to a different socket while a send operation is waiting. The waiting operation then frees the association, producing a use-after-free that can become a double free [5]. Aikido reports that the bug was introduced in Linux 4.15 in 2018 [4]. Xint describes the kernel flaw as chainable with any of the five nginx vulnerabilities [1], although Tenable's FAQ discusses it only alongside CVE-2026-123007 [2]. As of November 19, the kernel flaw had no CVE and no upstream patch, and Xint is withholding technical details until a fix is available [2][5].

The demonstration matters because of its starting point. Xint ran nginx inside a Docker container, gained worker-level execution remotely, then used the kernel flaw to obtain root on the underlying host [1][5]. Containers share the host kernel, so a kernel memory-corruption bug reachable from inside a container is in effect a host compromise primitive. Namespaces and capability drops do not remove that risk while

the vulnerable kernel code path stays reachable. CSA made the same structural point in its July 2026 analysis of GhostLock (CVE-2026-43499), a futex use-after-free that also enabled container escape [14]. What is new with nginx2root is the entry point. GhostLock required existing local code execution, whereas nginx2root shows a remote, unauthenticated path from the network edge to that local foothold.

None of the disclosing researchers has stated whether default container seccomp profiles block the SCTP code path, so organizations should not assume that their runtime defaults provide this protection. The published interim mitigation is to prevent the SCTP module from loading on hosts that do not need it, by adding `install sctp /bin/false` to a file under `/etc/modprobe.d/` [4][5]. This control has two limits. It does not unload a module that is already loaded, and it has no effect on kernels with SCTP compiled in [4]. Teams should therefore verify the actual state of each host rather than rely on configuration management alone.

Ingress and Platform Exposure

Kubernetes environments face a particular risk. The community ingress-nginx controller was retired in March 2026, so upstream will not ship a fix [4][7]. HeroDevs notes that the controller enables HTTP/2 by default (`use-http2: "true"`) and proxies every request to a backend service, which makes both HTTP/2 flaws reachable in a default deployment [7]. Because an ingress controller terminates TLS and sees all traffic it routes, a compromised controller pod is likely to expose cluster traffic and TLS material, and if the kernel chain is also available, it offers a route to the worker node. HeroDevs released a patched build (v1.15.1-nes-1.15.12) that addresses four of the five CVEs; it marks the mail-proxy flaw as not affected because the build omits the mail module [7]. Phoenix Security reports fixes in F5's NGINX Ingress Controller 5.8.0 and NGINX Gateway Fabric 2.9.0, though Equixly had not independently confirmed coverage for F5's bundled products [5][6]. NGINX Plus customers on the affected R-series releases have no R-series fix and must move to a fixed 37.x release; Tenable lists the affected range as R34 through R36, while Phoenix Security lists R33 through R36 [2][6], so customers on R33 should check F5's advisory directly [11].

Distribution packages add another complication. Debian and Ubuntu backport fixes without changing the upstream version string. Ubuntu 22.04's package, for example, moved from `1.18.0-6ubuntu14` to `1.18.0-6ubuntu14.22` while `nginx -v` still reports 1.18.0 [1][5]. Version-banner scanning therefore cannot distinguish patched distribution builds from unpatched ones. It will overstate exposure on patched hosts, and it offers no assurance about container images pinned to older digests, which must be verified by package version or image rebuild date rather than by the reported nginx version.

Recommendations

The recommendations below are ordered by urgency. Immediate actions address the exploitable flaws directly; short-term mitigations close the gaps that patching and version scanning leave behind; and the strategic considerations address the architectural lessons that will outlast this incident.

Immediate Actions

The first priority is to replace vulnerable nginx binaries on internet-facing systems. Upgrade nginx Open Source to 1.31.9 or 1.30.7 and NGINX Plus to 37.0.8.1 or 37.3.0.1, or apply your distribution's backported package [1][2][11]. Then perform a full restart or a supported live binary upgrade; Xint and Tenable both stress that a configuration reload does not replace the running executable [1][2]. Container images should be rebuilt and redeployed rather than patched in place, and teams should check that image digests in registries and deployment manifests have actually changed [5]. Where patching must wait, disable HTTP/2 on exposed listeners (`http2 off;` on 1.25.1 and later, also removing any deprecated `http2` parameter from `listen` directives, or simply removing `http2` from the `listen` directive on older versions). Then verify the change from outside the network with an ALPN probe [1][5]. Native gRPC proxying will break under this workaround, so plan the change with application owners [1].

At the same time, teams must address the configurations the HTTP/2 workaround does not cover. The following table summarizes the interim mitigation for each remaining exposure.

Exposure	Interim mitigation
CVE-2026-121810 (HTTP/3)	Remove `quic` listeners, set `http3 off;`, send `Alt-Svc: clear`, and consider blocking UDP/443 at the perimeter [1][5].
CVE-2026-123634 (resolver)	Point `proxy_pass` at a static `upstream` group whose `server` entries use literal IP addresses or names resolved at startup (no `resolve` parameter), so that no DNS resolution happens at request time [1][5].
CVE-2026-113425 (mail proxy)	Bind mail listeners to loopback or restrict them to trusted source addresses [1] [4].

Exposure	Interim mitigation
Kernel SCTP flaw	Block the SCTP module through `/etc/modprobe.d/` on container hosts and edge servers that do not use SCTP, and check whether it is already loaded [4][5].

The resolver mitigation warrants particular care. Recent open-source nginx releases support a `resolve` parameter on `upstream` server entries that performs periodic runtime DNS resolution, so an operator who moves to an `upstream` group but keeps that parameter may not have removed the resolver code path at all.

Short-Term Mitigations

Over the following weeks, organizations should inventory every nginx instance that version scanning may miss. That includes nginx embedded in ingress controllers, API gateways, appliance images, and third-party containers. Community ingress-nginx deployments should be migrated to a maintained controller or moved to a supported patched build, and until then operators should set `use-http2: "false"` [7]. The three upstream-dependent use-after-free flaws reportedly required outbound connectivity to attacker infrastructure in Xint's tests [5]. Restricting egress from edge proxies to known backends and resolvers is therefore a worthwhile layer of protection that will pay off beyond this incident.

Detection deserves investment as well, because the HTTP/2 control frames used in the attack (WINDOW_UPDATE, RST_STREAM, PING) are not logged as requests [1]. Xint suggests watching for long-lived connections whose received byte counts keep climbing, adding `$connection` and `$request_length` to log formats, and alerting on `execve` activity by the nginx worker user and on worker segfaults or core dumps [1]. Rapid7's public exploit sends on the order of 128 GB over a single connection [3], so per-connection traffic anomalies are a practical signal for that exploit and its derivatives. Because Wiz and Xint report working exploits whose techniques have not been published, defenders should not assume every exploit of CVE-2026-123007 will show the same volume, and should pair traffic thresholds with the host-level signals above. For containerized nginx, consider custom seccomp profiles that deny SCTP socket creation, and sandboxed runtimes such as gVisor or Kata Containers for internet-facing workloads. Both reduce the kernel attack surface reachable from a compromised container.

Strategic Considerations

nginx2root fits a pattern CSA has tracked through 2026, in which AI-assisted analysis is contributing to a sharp rise in critical findings in mature infrastructure code. Xint notes that the nginx security advisory list recorded 26 entries in 2026 through November 17, not counting nginx2root, compared with two in all of 2025 [1][12]. Not all of that increase is attributable to AI tooling, since this incident alone involved human discoverers and F5's internal research, but the volume itself places pressure on downstream patch pipelines. CSA's analysis of AI-accelerated vulnerability discovery and patch debt argued that this trend would compress the window between disclosure and weaponization and expose organizations whose patch processes assume human-paced discovery [16]. The collision on CVE-2026-123007 supports that argument: Xint, Wiz, Rapid7, and Ikotas Labs each reached the same decade-old bug within weeks of one another [5][7]. It is reasonable to infer that capable adversaries using similar tools could find the same class of flaw, and that the time available to defenders after disclosure will keep shrinking.

Two architectural lessons follow. First, the edge tier should be treated as a high-churn, rapidly patchable component. That means immutable images, automated rebuilds on upstream releases, and restart-capable rollouts, not long-lived hand-maintained hosts. Second, container isolation should be valued for what it actually provides. A shared kernel with a reachable memory-corruption bug cannot be relied on as a tenant boundary. Workloads exposed directly to untrusted network input therefore warrant stronger isolation, such as microVMs, sandboxed runtimes, or dedicated nodes, in proportion to what a host compromise would expose. Organizations should also expect more such chains. Pairing a remote application flaw with a local kernel escalation is a plausible next target for automated vulnerability research, particularly if tooling begins to explore application and kernel layers together.

CSA Resource Alignment

CSA's July 2026 research note **GhostLock: 15-Year-Old Linux Kernel Flaw Grants Root, Escapes Containers** [14] is the closest prior analysis of nginx2root's second stage. It examined how a long-lived kernel use-after-free turns local code execution in a container into root on the host. It recommended prioritizing multi-tenant systems and treating container boundaries as insufficient against kernel bugs. nginx2root supplies the remote entry point GhostLock lacked. Organizations that adopted the GhostLock guidance on stronger isolation for exposed workloads are better placed to absorb the unpatched SCTP flaw, and the same note's emphasis on kernel patch velocity will matter once a fix ships.

HTTP/2 Bomb: AI-Discovered DoS Hits Every Major Web Server [15], published in June 2026, documented an AI-assisted discovery of a denial-of-service technique against HTTP/2 implementations, including nginx. Together with nginx2root, it shows that HTTP/2 protocol-handling code, long considered

mature, is now a productive target for machine-assisted analysis. Six months later, nginx2root shows the same class of code yielding remote code execution rather than only denial of service.

CSA's **AI-Accelerated Vulnerability Discovery and the Patch Debt Crisis** [16] provides the strategic framing for the recommendations above. Its core argument is that AI-driven discovery raises disclosure volume and compresses exploitation windows faster than enterprise patch capacity grows. nginx2root is a concrete instance: a ten-CVE out-of-band release, an unpatched kernel dependency, and a retired ingress project that will never receive an upstream fix.

For control mapping, the **AI Controls Matrix (AICM) v1.1** [17] covers the relevant requirements in its Threat and Vulnerability Management (TVM) domain, including timely remediation, vulnerability identification, and prioritization. It covers secure configuration and isolation of infrastructure in its Infrastructure Security (I&S) domain. Organizations can use these controls to set and assess remediation SLAs for edge components and container hosts that reflect machine-speed discovery.

References

[1] Xint. "NGINX2Root: New Remote Code Execution Vulnerabilities in nginx Require Urgent Patching." Xint Blog, November 18, 2026.

[2] Tenable Research. "NGINX2Root (CVE-2026-123007): Frequently Asked Questions About Five Critical NGINX Vulnerabilities." Tenable Blog, November 18, 2026 (updated November 19, 2026).

[3] Rapid7. "CVE-2026-123007: Critical NGINX HTTP/2 Integer Underflow Leads to RCE (Fixed)." Rapid7 Blog, November 18, 2026 (updated November 20, 2026).

[4] Aikido Security. "nginx2root: Critical nginx RCE Vulnerabilities and How to Fix Them." Aikido Blog, November 18, 2026.

[5] Equixly. "nginx2root: Disabling HTTP/2 Stops Two of Five nginx RCE Flaws." Equixly Blog, November 18, 2026.

[6] Phoenix Security. "NGINX2Root (CVE-2026-123007): Unauthenticated HTTP/2 RCE." Phoenix Security, November 2026.

[7] HeroDevs. "nginx2root: Patch the Ingress NGINX HTTP/2 RCE Flaws." HeroDevs Blog, November 19, 2026.

[8] Jones, David. "Critical Remote Code Execution Flaws Found in Nginx Web Server." Cybersecurity Dive, November 19, 2026.

[9] Vogel, Heinrich. "NGINX HTTP/2 Vulnerability: 50,000 Lines in 90 Minutes." Shattered, November 2026 (updated November 20, 2026).

[10] Wiz Research. "Unauthenticated Remote Code Execution in NGINX: CVE-2026-123633 and CVE-2026-123007." Wiz Blog, November 18, 2026.

[11] F5. "K000164058: NGINX Security Advisory Index for the November 18, 2026 Release." F5 MyF5 Knowledge Base, November 18, 2026.

[12] nginx.org. "nginx Security Advisories." nginx.org, accessed November 20, 2026.

[13] Fewer, Stephen (Rapid7). "CVE-2026-123007 Proof-of-Concept Exploit." GitHub, November 2026.

[14] Cloud Security Alliance. "GhostLock: 15-Year-Old Linux Kernel Flaw Grants Root, Escapes Containers." CSA AI Safety Initiative, July 14, 2026.

[15] Cloud Security Alliance. "HTTP/2 Bomb: AI-Discovered DoS Hits Every Major Web Server." CSA AI Safety Initiative, June 4, 2026.

[16] Cloud Security Alliance. "AI-Accelerated Vulnerability Discovery and the Patch Debt Crisis." CSA AI Safety Initiative, June 17, 2026.

[17] Cloud Security Alliance. "AI Controls Matrix (AICM) v1.1." Cloud Security Alliance, 2025.