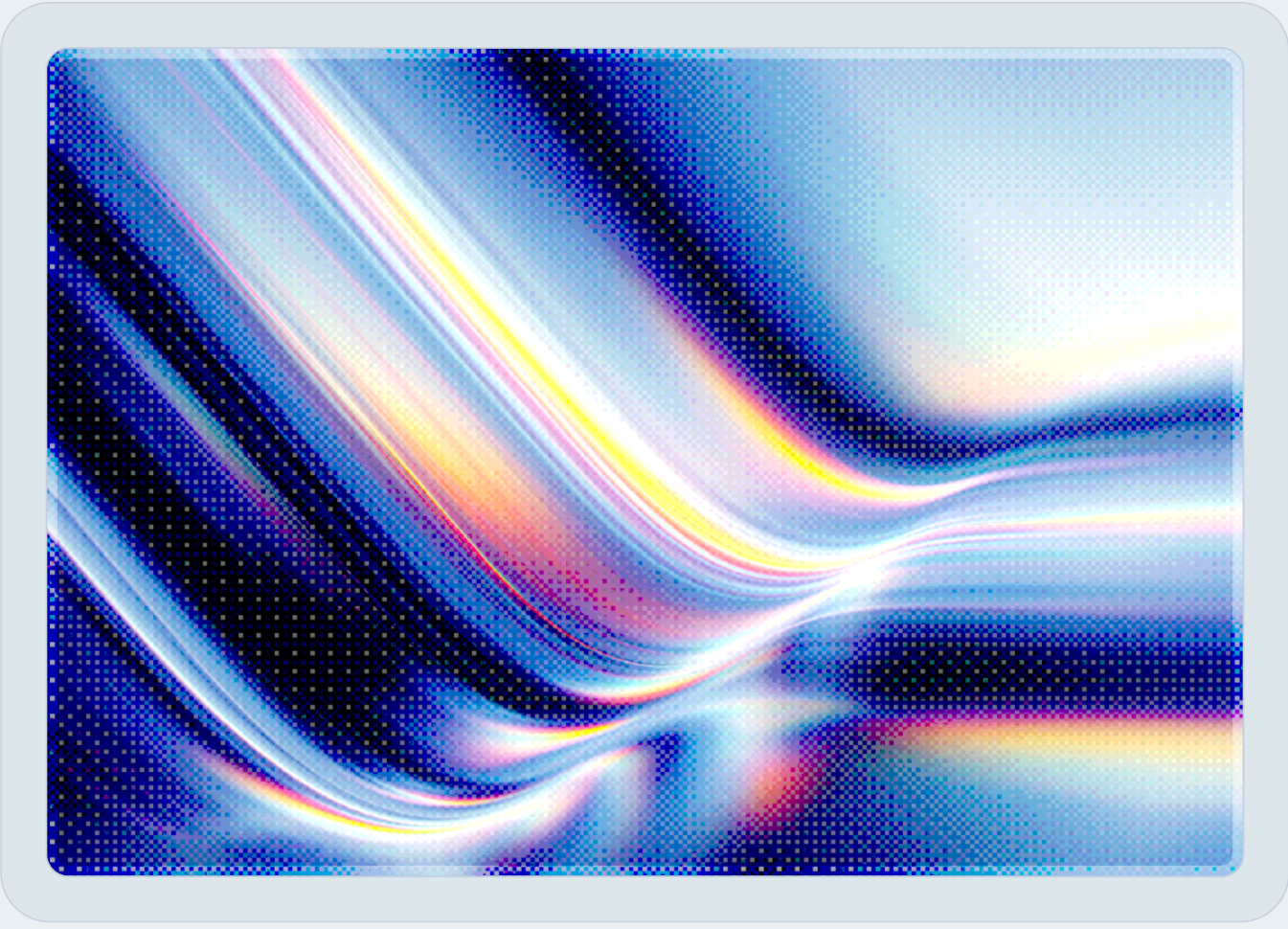
CSAI Foundation | Cloud Security Alliance

NGINX Rift: 18-Year-Old Heap Overflow Enables Unauthenticated RCE

CVE-2026-42945 in ngx_http_rewrite_module — AI-Discovered Flaw with CVSS 9.2 Threatens One-Third of the Web

2026-05-14

AI-assisted Rapid Research



CSAI

CSA cloud security alliance®

Key Takeaways

- **CVE-2026-42945 (CVSS 9.2 Critical) is a heap-based buffer overflow** in NGINX's ngx_http_rewrite_module that has existed in the codebase since 2008, affecting every version of NGINX Open Source from 0.6.27 through 1.30.0 and NGINX Plus R32 through R36, along with seven dependent F5 and NGINX product lines [1][4].
- **An unauthenticated attacker can trigger a denial-of-service condition — and, on systems with ASLR disabled, achieve remote code execution —** by sending a single crafted HTTP request to a vulnerable NGINX instance configured with a common rewrite-and-set directive pattern — no credentials, session, or prior access required [1][2].
- **The vulnerability was discovered by an autonomous AI analysis system**, not human code review — depthfirst's system flagged the heap overflow within six hours of ingesting the NGINX source code in April 2026, along with four additional security findings, three of which were confirmed by NGINX as distinct CVEs (CVE-2026-42946, CVE-2026-40701, and CVE-2026-42934) [1].
- **A public proof-of-concept exploit is available** on GitHub as of May 13, 2026, demonstrating full unauthenticated code execution against systems with ASLR disabled [3]. The publicly available exploit means unpatched internet-facing NGINX deployments face active exploitation risk.
- **Patches are available now**: NGINX Open Source 1.30.1 and 1.31.0; NGINX Plus R32 P6 and R36 P4. Organizations that cannot patch immediately must replace unnamed regex captures ($1, $2) with named captures in all affected rewrite directives as an interim workaround [5][6].

Background

NGINX serves as one of the foundational components of the modern internet. As of April 2026, it powers approximately 32.5% of all known web servers globally [8], with a presence spanning enterprises, cloud providers, and content delivery networks. Beyond basic web serving, NGINX is widely deployed as a reverse proxy, load balancer, API gateway, and Kubernetes ingress controller — functions that place it at the perimeter of production infrastructure, directly exposed to untrusted internet traffic. A remotely exploitable vulnerability in NGINX does not affect a niche component; it affects the entry point to a substantial fraction of internet-facing enterprise systems.

CVE-2026-42945, named NGINX Rift by its discoverers, is a heap-based buffer overflow in the ngx_http_rewrite_module — the NGINX component responsible for processing URI rewrite rules that redirect, modify, and transform incoming HTTP requests. The flaw was first introduced in NGINX version 0.6.27, released in 2008, and was not publicly disclosed for approximately 18 years. Its persistence across every subsequent version through 1.30.0 is not the result of an esoteric edge case: the rewrite module is a heavily used feature, and the affected directive combination — rewrite with unnamed captures followed by a set directive — is a documented, commonly recommended pattern in API gateway and reverse proxy configurations [1].

The vulnerability's discovery merits particular attention. In April 2026, the security research firm depthfirst applied its autonomous AI-powered analysis system to the NGINX source code repository. The system required a single onboarding action and six hours of analysis to flag the heap overflow, along with four additional security findings. Three of those findings were confirmed by NGINX as distinct vulnerabilities — CVE-2026-42946 (CVSS 8.3, excessive memory allocation), CVE-2026-40701 (CVSS 6.3, use-after-free in SSL module), and CVE-2026-42934 (CVSS 6.3, out-of-bounds read in charset module) — and a fifth finding reported by depthfirst remained unconfirmed by NGINX at the time of public disclosure [1]. Depthfirst disclosed all five findings to NGINX via a GitHub security advisory on April 21, 2026; NGINX confirmed four of the five issues on April 24, 2026. Coordinated public disclosure followed on May 13, 2026 [1][11].

Security Analysis

Technical Root Cause

The vulnerability resides in the two-pass memory management logic of NGINX's script engine for URI construction. When NGINX processes a rewrite directive, it performs the operation in two sequential stages: a length-calculation pass that determines how large a buffer to allocate, followed by a write pass that copies the constructed URI into the allocated buffer. This design is sound in most cases, but it breaks down under a specific and common configuration pattern.

When a rewrite directive includes a question mark in the replacement string that contains an unnamed PCRE capture group (such as `$1` or `$2`), NGINX's internal flag tracking mechanism permanently sets the `is_args` flag for that processing context. A subsequent `set` directive in the same request-processing chain then computes the required buffer length under the assumption that no URI escaping will be needed. When the write pass actually executes, however, it applies URI escaping to the data being

written — escaping that expands certain characters and produces more output bytes than the length calculation anticipated. The result is a classic heap buffer overflow: the write operation runs past the end of the allocated buffer, overwriting adjacent heap structures with attacker-influenced data [1][4].

The conditions required to trigger this overflow are not unusual. A configuration of the following general form is sufficient:

```
location /api/ {
    rewrite ^/api/(.+)$ /backend/$1? last;
    set $target_param $1;
}
```

This pattern — rewriting a URL with an unnamed capture and a question mark, followed by a `set` directive — is specifically recommended in NGINX documentation for API gateway configurations and appears widely in production systems [1][2]. The attacker does not need to influence the NGINX configuration; they need only to find a server that uses this pattern and send a carefully constructed HTTP request to reach the vulnerable code path.

Exploitation and Impact

An attacker who can send HTTP requests to a vulnerable NGINX server needs no authentication, no valid session, and no prior foothold. A single crafted request is sufficient to trigger the heap overflow in the NGINX worker process [1][2]. Successful exploitation can produce two distinct outcomes depending on the target environment: in all configurations, the worker process crashes, producing a denial of service condition; on systems with ASLR (Address Space Layout Randomization) disabled, depthfirst has demonstrated full unauthenticated remote code execution [3]. Researchers have published a working proof-of-concept exploit on GitHub that confirms this capability [3].

The reliability of remote code execution on ASLR-enabled systems — which represent the majority of modern Linux server deployments — remains an open research question as of this writing. However, the availability of a public exploit, the absence of any authentication requirement, the single-request attack surface, and NGINX's position as a perimeter component make this vulnerability operationally severe regardless of whether reliable ASLR bypass is developed. Worker process crashes are themselves a significant availability risk for high-volume services. Furthermore, industry experience consistently shows that high-severity vulnerabilities with public PoC code tend to be weaponized within hours to days of disclosure [2][10].

Affected Products and Patch Status

The vulnerability affects a broad range of F5 and NGINX products, reflecting NGINX's central role in F5's commercial product portfolio. The table below, derived from published vendor and GitHub security advisories [6][7], summarizes affected version ranges and available remediation.

Product	Affected Versions	Fixed Version
NGINX Open Source	0.6.27 – 1.30.0	1.30.1 (stable) / 1.31.0 (mainline)
NGINX Plus	R32 – R36	R32 P6 / R36 P4
NGINX Instance Manager	2.16.0 – 2.21.1	Vendor advisory [6]
NGINX App Protect WAF	4.9.0 – 4.16.0; 5.1.0 – 5.8.0	Vendor advisory [6]
F5 WAF for NGINX	5.9.0 – 5.12.1	Vendor advisory [6]
F5 App Protect DoS	4.3.0 – 4.7.0; F5 DoS for NGINX 4.8.0	Vendor advisory [6]
NGINX Gateway Fabric	1.3.0 – 1.6.2; 2.0.0 – 2.5.1	Vendor advisory [6]
NGINX Ingress Controller	3.5.0 – 3.7.2; 4.0.0 – 4.0.1; 5.0.0 – 5.4.1	Vendor advisory [6]

Organizations running NGINX in Kubernetes environments should note that NGINX Ingress Controller and NGINX Gateway Fabric are specifically listed as affected, meaning that Kubernetes cluster ingress layers may be directly vulnerable. Container-based NGINX deployments are not inherently protected: the vulnerability is in the application binary, not the host kernel, and any NGINX worker process running the affected configuration is exploitable regardless of whether it runs in a container, a virtual machine, or directly on bare metal.

The three related vulnerabilities identified in the same depthfirst audit — CVE-2026-42946, CVE-2026-40701, and CVE-2026-42934 — are addressed in the same patched releases. Organizations applying the NGINX Rift patches will simultaneously receive fixes for these additional findings.

Recommendations

Immediate Actions

The priority response is straightforward: upgrade NGINX. NGINX Open Source users should upgrade to version 1.30.1 or 1.31.0, both of which are available from the official NGINX repository at nginx.org [5][9]. NGINX Plus subscribers should apply patch release R32 P6 or R36 P4 and consult the F5 security advisory K000161019 for detailed upgrade instructions for dependent products including NGINX Instance Manager, NGINX App Protect WAF, and NGINX Ingress Controller [5][6]. Organizations should treat this as a P1 upgrade — the presence of a public exploit and the zero-authentication attack surface means that unpatched public-facing NGINX instances should be considered actively at risk from the moment this advisory became public on May 13, 2026.

For organizations that cannot immediately apply patches due to change management processes or operational constraints, an interim workaround is available: replace all unnamed PCRE captures (`$1`, `$2`, etc.) in affected `rewrite` directives with named captures (e.g., `(?P<name>pattern)` with `$name`) [1][5]. This eliminates the specific code path that triggers the buffer overflow. This workaround requires reviewing every NGINX configuration file across the environment for the affected directive pattern, a process that should be completed within hours, not days, given the publicly available exploit code.

Inventory of NGINX instances is a prerequisite to both actions. Containerized deployments — particularly NGINX-based sidecar proxies, Kubernetes ingress controllers, and service mesh components — may not appear in traditional asset inventories but are equally exploitable. Security teams should query container registries and Kubernetes clusters for NGINX images and chart versions alongside traditional host-based asset scanning.

Short-Term Mitigations

While patches are being applied, additional defensive layers can reduce exposure. Web application firewalls positioned upstream of NGINX may be able to detect and block crafted requests that target the vulnerable code path, though WAF rules for newly disclosed vulnerabilities are not immediately available from all vendors and should not be relied upon as a primary mitigation. Network segmentation that limits internet-accessible NGINX instances to only the ports and protocols necessary for their function reduces the attack surface for opportunistic scanning. Organizations should enable NGINX access

logging and monitor for unusual HTTP request patterns — specifically for requests that target URI paths handled by rewrite-and-set directive combinations — during the window before patches can be fully deployed.

Rate limiting at the load balancer or upstream network edge provides a modest additional layer of protection against automated scanning and exploit tooling. While a single request is sufficient to trigger the overflow, many attackers probe at scale before targeting specific systems, meaning that detection of scanning activity may provide a narrow response window before active exploitation attempts are made. Early detection of that probing provides actionable response time for defenders.

Strategic Considerations

NGINX Rift illustrates a broader challenge for organizations that depend on widely deployed open-source infrastructure: a single well-crafted automated analysis identified, in six hours, a critical vulnerability that had not been publicly reported after 18 years of human code review, security audits, and community scrutiny. This is not a critique of NGINX's development practices — it reflects a fundamental asymmetry between the scale of open-source codebases and the capacity of human review processes to exhaustively analyze memory safety properties. The implication is that organizations should not treat the age of a dependency as a proxy for its security. Long-lived code, particularly code written before memory-safe languages became practical alternatives for infrastructure software, warrants periodic adversarial review regardless of its maturity.

The discovery also demonstrates the operational value of AI-assisted vulnerability analysis as a proactive defensive capability, a theme with growing relevance to enterprise security programs. As autonomous analysis systems become more widely accessible, organizations should evaluate whether applying these tools to their own first-party code and critical third-party dependencies — before adversaries do — represents a viable component of their vulnerability management strategy.

CSA Resource Alignment

The NGINX Rift disclosure intersects with several active areas of CSA research and framework development.

CSA's AI Controls Matrix (AICM) and its predecessor, the Cloud Controls Matrix (CCM), address vulnerability management as a foundational cloud security discipline. Specifically, CCM domain TVM (Threat and Vulnerability Management) covers the processes for identifying, assessing, and remediating vulnerabilities in cloud-hosted infrastructure, including the requirement for timely patching of publicly

disclosed critical vulnerabilities. NGINX Rift serves as a concrete test case for organizational TVM maturity: the combination of a CVSS 9.2 score, publicly available exploit code, and zero authentication requirement creates an unambiguous P1 scenario that TVM processes must handle at speed.

The discovery mechanism — an autonomous AI system completing in six hours what human review had not accomplished in 18 years — is directly relevant to CSA's AI Safety Initiative's work on AI-assisted security operations. The MAESTRO framework (Multi-Agent Extensible Security Threat Reference and Operations), CSA's threat modeling methodology for agentic AI systems, provides a lens for thinking about how autonomous analysis agents should be governed, authorized, and monitored when applied to vulnerability research. Organizations integrating AI-assisted code analysis into their security programs should consider how MAESTRO's trust boundary and capability governance concepts apply to tools that have the potential to identify exploitable vulnerabilities in production systems.

CSA's Zero Trust guidance is also relevant to the post-exploitation phase of the threat model. NGINX typically operates as a perimeter gateway, and successful code execution within a worker process provides an attacker with a foothold in the infrastructure network. Zero Trust principles — particularly the requirement that no network path should grant implicit trust based on network location alone — govern the blast radius of such a compromise. Organizations that have implemented Zero Trust segmentation, mutual TLS between NGINX and upstream services, and runtime process isolation for web server workloads are likely better positioned to contain an NGINX worker compromise than those that rely on perimeter security alone.

Finally, CSA's STAR (Security Trust Assurance and Risk) program provides the assessment framework through which cloud service providers and managed infrastructure vendors can demonstrate their patch responsiveness for vulnerabilities of this severity. Organizations using managed NGINX offerings — whether through cloud provider load balancers, managed Kubernetes ingress services, or CDN providers — should verify patch timelines with their vendors against the May 13, 2026 disclosure date and document vendor-supplied evidence of remediation within their STAR-aligned audit records.

References

[1] depthfirst. "NGINX Rift: Achieving NGINX Remote Code Execution via an 18-Year-Old Vulnerability." depthfirst Research, May 2026.

[2] The Hacker News. "18-Year-Old NGINX Rewrite Module Flaw Enables Unauthenticated RCE." The Hacker News, May 2026.

[3] depthfirst Disclosures. "Nginx-Rift: Exploit for CVE-2026-42945." GitHub, May 2026.

[4] NIST National Vulnerability Database. "CVE-2026-42945 Detail." NVD, May 2026.

[5] NGINX. "NGINX Security Advisories." nginx.org, May 2026.

[6] F5 Networks. "K000161019: NGINX ngx_http_rewrite_module Heap Buffer Overflow (CVE-2026-42945)." F5 Support, May 2026.

[7] GitHub Security Advisories. "NGINX Plus and NGINX Open Source Vulnerability in ngx_http_rewrite_module (GHSA-gcgv-v5gf-c543)." GitHub Advisory Database, May 2026.

[8] W3Techs. "Usage Statistics and Market Share of Nginx, April 2026." W3Techs, April 2026.

[9] AlmaLinux. "NGINX Rift (CVE-2026-42945): Patched nginx Available in Testing." AlmaLinux Blog, May 13, 2026.

[10] CybersecurityNews. "Critical 18-Year-Old NGINX Vulnerability Enables Remote Code Execution Attacks." CybersecurityNews, May 2026.

[11] oss-security. "NGINX ngx_http_rewrite_module Vulnerability CVE-2026-42945." oss-security Mailing List, May 2026.

