INTELLIGENCE BRIEFING
Security Command Center
TLP:CLEAR
2026-06-23 06:48 UTC

PixelSmash (CVE-2026-8461): FFmpeg MagicYUV Heap Overflow Enables RCE Across Downstream Media Applications

CVE VULNERABILITY | **HIGH** | CVSS 7.5

SCC Item ID	SCC-CVE-2026-0344
Type	CVE Vulnerability
CVE ID	CVE-2026-8461
Severity	HIGH
CVSS Base Score	7.5
EPSS Score	0.0035 (26th percentile)
Affected Products	FFmpeg libavcodec (pre-8.1.2); Jellyfin ≤10.11.9; Kodi; OBS Studio; PhotoPrism; Nextcloud; Emby; GNOME/KDE/XFCE desktop thumbnail generators; potentially Slack, Discord, Telegram, WhatsApp server-side preview pipelines (untested); Plex (mitigated via custom build)
Published	2026-06-22T17:05:01
Discovery Source	Rss

Executive Summary

A heap overflow vulnerability in FFmpeg's MagicYUV video decoder, CVE-2026-8461, enables remote code execution by delivering a maliciously crafted media file to any application that automatically processes video, including Jellyfin, Kodi, OBS Studio, Nextcloud, and desktop thumbnail generators on Linux environments. JFrog researchers demonstrated full zero-click RCE against Jellyfin 10.11.9, meaning no user action is required beyond the file entering a monitored media library. Organizations running self-hosted media servers, file-sharing platforms, or Linux desktop environments face direct compromise risk until affected downstream applications ship and deploy their own builds against FFmpeg 8.1.2.

Technical Analysis

CVE-2026-8461 is a heap out-of-bounds write (CWE-787, CWE-122) in FFmpeg's MagicYUV decoder within libavcodec, patched in FFmpeg 8.1.2 (released June 17). The flaw allows an attacker to trigger arbitrary code execution by supplying a crafted .mkv or similarly-encoded media file to any application invoking the vulnerable

decoder. JFrog's proof-of-concept targets Jellyfin ≤10.11.9 via its automated library scan pipeline, no user interaction required (MITRE ATT&CK techniques T1190 and T1203). Attack surface extends to Kodi, OBS Studio, PhotoPrism, Nextcloud, Emby, GNOME/KDE/XFCE thumbnail daemons. Server-side preview pipelines in Slack, Discord, Telegram, and WhatsApp may be affected if they use vulnerable FFmpeg builds, but this has not been confirmed. CVSS base score is 7.5; the zero-click Jellyfin path suggests real-world severity is higher in automated-processing deployments. EPSS score is 0.00346 (26th percentile) as of scoring date. Downstream projects must independently integrate and ship the FFmpeg 8.1.2 patch; FFmpeg's upstream release does not automatically remediate bundled or statically-linked consumers. No CISA KEV listing as of this report.

Action Checklist

1. Step 1: Containment. Immediately identify all production systems running Jellyfin ≤10.11.9, Emby, Nextcloud, PhotoPrism, OBS Studio, or Kodi, and any Linux desktop environments with automated thumbnail generation enabled (GNOME Tracker, KDE Baloo, Thumbnailer daemons). Disable automated media library scanning and thumbnail generation on unpatched instances until the downstream application ships a build against FFmpeg 8.1.2. Restrict inbound media file ingestion paths at the network perimeter where possible. Reference CIS 4.1 (Establish and Implement Firewall Standards) and network segmentation to limit exposure of media server ports to trusted networks only.
2. Step 2: Detection. Query application and system logs for unexpected process spawning originating from media-processing daemons (e.g., jellyfin, thumbnailer, ffmpeg child processes). Monitor for anomalous outbound network connections from media server hosts. On Linux desktops, review GNOME/KDE thumbnail daemon logs for crash events or unexpected child process creation. Correlate file intake events (new media files added to watched directories) against any subsequent process or network anomalies. No public IOC hashes are confirmed at this time; focus on behavioral indicators aligned with NIST AU-12 (Audit Record Generation) and AC-6 (Least Privilege). Apply file integrity monitoring (FIM) to monitor for unexpected modification of media server executables or configuration files.
3. Step 3: Eradication. Upgrade FFmpeg to 8.1.2 or later on all hosts where it is installed as a shared library. For downstream applications: apply vendor-issued patches as they become available for Jellyfin, Kodi, OBS Studio, PhotoPrism, Nextcloud, and Emby; do not assume the FFmpeg upstream release resolves statically-linked or bundled builds. Verify installed FFmpeg version with 'ffmpeg -version' on each affected host. Remove or quarantine any media files of unknown provenance that entered watched library directories before containment. Reference CIS 7.3 (Perform Automated Operating System Patch Management) and CIS 7.4 (Perform Automated Application Patch Management) to prioritize and track remediation across the asset inventory.
4. Step 4: Recovery. After patching, re-enable automated library scanning and thumbnail generation. Validate the FFmpeg version linked by each downstream application using ldd or equivalent to confirm the patched shared library is in use. Monitor media server process behavior for 72 hours post-remediation for anomalous child process spawning or outbound connections, per NIST AU-12. Confirm no unauthorized changes to media server configuration files or scheduled tasks occurred during the exposure window using file integrity monitoring (FIM).
5. Step 5: Post-Incident. This vulnerability exposed the risk of automated media processing pipelines ingesting untrusted files without sandboxing or input validation controls. Review the software inventory (CIS 2.1) to identify all applications bundling or dynamically linking FFmpeg and establish a tracking process for upstream library CVEs. Evaluate whether media ingestion services should run in isolated containers or sandboxed environments with restricted process execution permissions, aligned with NIST

AC-6 (Least Privilege). Implement least-privilege access controls to restrict the OS-level privileges of media server processes, reducing blast radius for future heap exploitation.

IR / Forensic Enrichment

Triage Priority	IMMEDIATE
Escalation Criteria	Escalate to CISO and legal counsel immediately if forensic evidence (coredumps, auditd execve chains, anomalous outbound connections from the Jellyfin or ffmpeg process) indicates successful RCE exploitation rather than mere exposure, particularly if Nextcloud or Emby instances processed PII or PHI-adjacent media content that may trigger breach notification obligations under HIPAA, GDPR, or applicable state privacy law.
Recovery Notes	After confirming FFmpeg 8.1.2 is linked by all downstream applications via `ldd` validation, re-enable media library scanning incrementally — starting with internally sourced media only — before allowing ingestion from external or user-uploaded sources. Monitor Jellyfin, Nextcloud, and thumbnail daemon process trees continuously for 72 hours using auditd or Sysmon for any execve events descending from media-processing PIDs, as a successful pre-patch exploit may have staged a persistent implant that survives the FFmpeg upgrade. Treat any MagicYUV-encoded file (`MYUV` FourCC) that arrived during the exposure window as potentially weaponized and do not re-ingest it without static analysis in an isolated sandbox.
Forensic Artifacts	Jellyfin application logs at `/var/log/jellyfin/jellyfin*.log` and `~/.local/share/jellyfin/log/` — search for ffmpeg invocations against files with MagicYUV encoding (`MYUV` FourCC) followed by unhandled exceptions, segfaults, or unexpected process exits within the same log session, which would indicate heap overflow triggering in `libavcodec/magicyuv.c` | Linux kernel coredumps in `/var/lib/systemd/coredump/` or `/var/crash/` generated by ffmpeg or thumbnailer processes — GDB backtrace will show stack frames in `magicyuv_decode_frame()` or `magicyuv_decode_slice()` confirming exploitation of the MagicYUV heap overflow path | Filesystem `mtime`/`atime` records and SHA-256 hashes of all media files added to Jellyfin, Nextcloud, Nextcloud External Storage, or GNOME/KDE thumbnail-watched directories during the exposure window — files with `MYUV` FourCC identified via `mediainfo --Inform='Video;%CodecID%'` are priority suspects | Auditd `execve` syscall logs filtered on PPIDs matching jellyfin, gnome-tracker-miner-fs, kde-baloo, or ffmpeg processes — any child process spawning a shell (`/bin/sh`, `/bin/bash`), curl, wget, or a non-media binary from these parents is a high-confidence RCE indicator specific to post-exploitation of CVE-2026-8461 | Network capture (`tcpdump` or `/proc/net/tcp` snapshots) of outbound connections initiated by the Jellyfin or ffmpeg process user account (typically `jellyfin` UID) to non-local, non-CDN destinations — successful RCE via a heap overflow in a media decoder would manifest as a reverse shell or beacon originating from the media server's process network namespace

Per-Action IR Details

Step 1: Containment — Immediately identify all production systems running Jellyfin ≤10.11.9, Emby, Nextcloud, PhotoPrism, OBS Studio, or Kodi, and any Linux desktop environments with automated thumbnail generation enabled (GNOME Tracker, KDE Baloo, Thumbnailer daemons). Disable automated media library scanning and thumbnail generation on unpatched instances until the downstream application ships a build against FFmpeg 8.1.2. Restrict inbound media file ingestion paths at the network perimeter where possible. Reference CIS 4.4 (Implement and Manage a Firewall on Servers) to limit exposure of media server ports.

NIST Phase: Containment

Reference: NIST 800-61r3 §3.3 — Containment Strategy

Controls: CIS 4.4 (Implement and Manage a Firewall on Servers), NIST AC-4 (Information Flow Enforcement)

Compensating: On Linux hosts, immediately stop the Jellyfin, Emby, and thumbnail daemon services: `systemctl stop jellyfin emby-server gnome-tracker-3 kde-baloo`. Block inbound media upload ports (Jellyfin default 8096/8920, Nextcloud 443) at the host firewall using `ufw deny in 8096` or `iptables -I INPUT -p tcp --dport 8096 -j DROP`. For Nextcloud, disable the Files app's background scan via `occ files:scan --unscanned` suspension and comment out the cron job in `/etc/cron.d/nextcloud`. Enumerate FFmpeg-linked applications with `lsof | grep libavcodec` to identify all processes with live exposure.

Evidence: Before disabling scanning daemons or applying firewall rules, capture volatile state: run `ps auxf` to snapshot full process tree showing parent-child relationships of jellyfin, ffmpeg, thumbnailer, and gnome-tracker; capture active network connections with `ss -tulnp` and `netstat -ano`; dump `/proc//net/tcp` and `/proc//maps` to record memory-mapped libraries and confirm libavcodec version in use. If any ffmpeg child process is running at time of discovery, acquire a full memory dump of that process using `gcore ` before killing it — heap overflow exploitation of MagicYUV decoding would leave artifacts in the heap region of that process's address space.

Step 2: Detection — Query application and system logs for unexpected process spawning originating from media-processing daemons (e.g., jellyfin, thumbnailer, ffmpeg child processes). Monitor for anomalous outbound network connections from media server hosts. On Linux desktops, review GNOME/KDE thumbnail daemon logs for crash events or unexpected child process creation. Correlate file intake events (new media files added to watched directories) with any subsequent process or network anomalies. No public IOC hashes are confirmed at this time; focus on behavioral indicators aligned with NIST AU-6 (Audit Record Review, Analysis, and Reporting) and AU-12 (Audit Record Generation). Apply D3-SFA (System File Analysis) to monitor for unexpected modification of media server executables or configuration files.

NIST Phase: Detection Analysis

Reference: NIST 800-61r3 §3.2 — Detection and Analysis

Controls: NIST AU-6 (Audit Record Review, Analysis, And Reporting), NIST AU-12 (Audit Record Generation), NIST AU-2 (Event Logging)

Compensating: Deploy Sysmon on any Windows hosts running OBS Studio or Jellyfin for Windows; use EventID 1 (Process Create) filtered on ParentImage containing `jellyfin.exe` or `ffmpeg.exe` spawning `cmd.exe`, `powershell.exe`, or any non-media binary. On Linux, enable auditd rules: `auditctl -a always,exit -F arch=b64 -S execve -F ppid=$(pgrep jellyfin) -k pixelsmash_spawn` to catch child process execution from the Jellyfin PID tree. Use `inotifywait -m /var/lib/jellyfin/metadata /var/lib/jellyfin/transcodes -e create,modify` to flag new files written by the media server outside expected transcode output. For network anomaly detection, run `tcpdump -i any -w /tmp/mediaserver.pcap host and not port 8096` to capture unexpected outbound connections from the media server process.

Evidence: At this phase, preserve without altering: Jellyfin application logs at `/var/log/jellyfin/jellyfin*.log` and `~/.local/share/jellyfin/log/` — look for ffmpeg invocations against `.mkv` or `.yuv`-encoded files followed by unhandled exceptions or segfaults; GNOME Tracker crash logs in `~/.local/share/recently-used.xbel` and `journalctl -u tracker-miner-fs-3` for segfault entries timestamped within minutes of a new media file arrival; kernel crash dumps in `/var/crash/` or `/var/lib/systemd/coredump/` generated by a heap overflow in libavcodec's MagicYUV decoder (`magicyuv.c`); auditd logs showing execve calls descending from ffmpeg or thumbnailer processes; and `/proc//fd/` directory listings of the media daemon to identify any unexpected file descriptors opened post-decode.

Step 3: Eradication — Upgrade FFmpeg to 8.1.2 or later on all hosts where it is installed as a shared library. For downstream applications: apply vendor-issued patches as they become available for Jellyfin, Kodi, OBS Studio, PhotoPrism, Nextcloud, and Emby — do not assume the FFmpeg upstream release resolves statically-linked or bundled builds. Verify installed FFmpeg version with 'ffmpeg -version' on each affected host. Remove or quarantine any media files of unknown provenance that entered watched library directories before containment. Reference NIST SI-2 is not in the mapped control set; use CIS 7.3 (Perform Automated

Operating System Patch Management) and CIS 7.4 (Perform Automated Application Patch Management) to prioritize and track remediation across the asset inventory.

NIST Phase: Eradication

Reference: NIST 800-61r3 §3.4 — Eradication and Recovery

Controls: CIS 7.3 (Perform Automated Operating System Patch Management), CIS 7.4 (Perform Automated Application Patch Management), CIS 2.2 (Ensure Authorized Software is Currently Supported)

Compensating: Before patching, snapshot the host (VM snapshot or `dd` image of the OS volume) to preserve forensic state. Upgrade FFmpeg via package manager: `apt-get install --only-upgrade ffmpeg` (Debian/Ubuntu) or `dnf upgrade ffmpeg` (RHEL/Fedora), then confirm with `ffmpeg -version | grep -E '8\.[1-9]'`. For statically-linked applications such as Jellyfin Docker images, pull the updated container: `docker pull jellyfin/jellyfin:latest` and verify the embedded FFmpeg version inside the container with `docker run --rm jellyfin/jellyfin:latest ffmpeg -version`. Quarantine suspicious media files by moving them to an offline, write-protected directory: `mv /media/library/ /quarantine/ && chmod 000 /quarantine/`. Use `file` and `mediainfo` on quarantined files to identify MagicYUV-encoded content (codec identifier `MYUV`).

Evidence: Before applying patches or removing files, capture: a SHA-256 hash inventory of all media files that arrived in watched directories during the exposure window (`find /media/library -newer /var/log/jellyfin/jellyfin_.log -exec sha256sum {} \; > /forensics/media_intake_hashes.txt`); a copy of the Jellyfin transcoding cache at `/var/lib/jellyfin/transcodes/` which may contain partially decoded MagicYUV frames reflecting exploit payload structure; `ldd $(which jellyfin)` and `ldd $(which ffmpeg)` output to document which libavcodec version was linked at time of compromise; and any coredumps generated by ffmpeg during the exposure window from `/var/lib/systemd/coredump/` — these are critical for confirming whether exploitation produced a controlled write primitive or only a crash.

Step 4: Recovery — After patching, re-enable automated library scanning and thumbnail generation. Validate the FFmpeg version linked by each downstream application using ldd or equivalent to confirm the patched shared library is in use. Monitor media server process behavior for 72 hours post-remediation for anomalous child process spawning or outbound connections, per NIST AU-6. Confirm no unauthorized changes to media server configuration files or scheduled tasks occurred during the exposure window using D3-SFA (System File Analysis).

NIST Phase: Recovery

Reference: NIST 800-61r3 §3.5 — Recovery

Controls: NIST AU-6 (Audit Record Review, Analysis, And Reporting), NIST AU-3 (Content Of Audit Records), CIS 8.2 (Collect Audit Logs)

Compensating: Before re-enabling Jellyfin or thumbnail daemons, run integrity verification: `debsums -c ffmpeg libavcodec*` (Debian) or `rpm -Va ffmpeg` (RHEL) to confirm package integrity of the patched libraries. Use `ldd /usr/bin/jellyfin | grep libavcodec` and compare the resolved path against `ffmpeg -version` output to confirm shared library linkage is to 8.1.2 and not a cached older version. During the 72-hour monitoring window, run `inotifywait -m /etc/jellyfin/ /var/lib/jellyfin/config/ -e modify,attrib` to detect unauthorized configuration changes, and tail `journalctl -u jellyfin -f | grep -E '(ffmpeg|child|exec|spawn)'` for anomalous process activity. If auditd is deployed, retain the ruleset from the detection phase throughout the recovery watch period.

Evidence: Before re-enabling scanning services, document the baseline state: record `md5sum /etc/jellyfin/*.xml /var/lib/jellyfin/config/*.json` and store as a signed integrity baseline; capture `crontab -l` for all service accounts (jellyfin, www-data, nextcloud) and `systemctl list-timers` to confirm no persistence mechanisms were written during the exposure window; review `/var/spool/cron/` and `/etc/cron.d/` for entries modified during the compromise window using `find /etc/cron* /var/spool/cron -newer /var/log/jellyfin/ -ls`; and verify no new SUID/SGID binaries were written: `find / -perm /6000 -newer /tmp/containment_marker -ls 2>/dev/null`.

Step 5: Post-Incident — This vulnerability exposed the risk of automated media processing pipelines ingesting untrusted files without sandboxing or input validation controls. Review the software inventory (CIS 2.1) to identify all applications bundling or dynamically linking FFmpeg and establish a tracking process for upstream library CVEs. Evaluate whether media ingestion services should run in isolated containers or

sandboxed environments with restricted process execution permissions, aligned with NIST AC-6 (Least Privilege). Implement D3-UAP (User Account Permissions) to restrict the OS-level privileges of media server processes, reducing blast radius for future heap exploitation.

NIST Phase: Post Incident

Reference: NIST 800-61r3 §4 — Post-Incident Activity

Controls: CIS 2.1 (Establish and Maintain a Software Inventory), CIS 2.2 (Ensure Authorized Software is Currently Supported), NIST AC-6 (Least Privilege), CIS 7.1 (Establish and Maintain a Vulnerability Management Process)

Compensating: Build a persistent FFmpeg dependency map: `grep -r 'ffmpeg\|libavcodec\|libavformat' /var/lib/dpkg/info/*.list /var/lib/rpm/ > /inventory/ffmpeg_dependents.txt` to capture all packages dynamically linking FFmpeg on each host. Subscribe to the FFmpeg security mailing list (ffmpeg-security@ffmpeg.org) and Jellyfin/Nextcloud GitHub release feeds for upstream CVE notifications. For sandboxing without budget, convert Jellyfin to run under a systemd service unit with `NoNewPrivileges=true`, `ProtectSystem=strict`, `PrivateTmp=true`, and `SystemCallFilter=~@mount` to restrict syscall surface exploitable via a future heap overflow. Use `AppArmor` profiles (available in `apparmor-profiles` package) to restrict ffmpeg child process filesystem and network access to only required paths, preventing post-exploitation lateral movement.

Evidence: For the lessons-learned record, preserve: the full timeline of media file ingestion events reconstructed from Jellyfin access logs correlated with filesystem `mtime` on files in the watched library — this establishes the earliest possible exploitation window; any coredumps from libavcodec's MagicYUV decoder (`magicyuv.c` stack frames will appear in `gdb` backtrace of preserved dumps) as proof of vulnerable code path execution; the output of `ldd` and `ffmpeg -version` from each affected host pre-patch to document the exposure surface; and a final `sha256sum` comparison of Jellyfin and ffmpeg binaries post-patch against vendor-published checksums to close the chain of custody on eradication verification.

Detection Guidance

Focus on behavioral detection rather than signature-based IOCs, as no confirmed file hashes or network indicators are publicly attributed to active exploitation at this time. Key detection approaches: (1) Monitor process trees on media server hosts, flag any child process spawned by ffmpeg, jellyfin, thumbnailer, or equivalent media daemons that initiates network connections or writes to system directories. (2) On Linux desktops, watch GNOME Tracker (tracker-miner-fs), KDE Baloo, and XFCE Tumbler logs for crash events (SIGSEGV, SIGABRT) correlated with new file additions to watched directories. (3) Enable and review NIST AU-12 (Audit Record Generation) compliant logs for file execution events originating from media library paths. (4) Alert on unexpected outbound TCP/UDP from media server process accounts, particularly to external IPs. (5) Use file integrity monitoring (FIM) or NIST SI-7 (Software, Firmware, and Information Integrity) to baseline and monitor media server binary integrity; a successful heap exploit may stage a secondary payload. (6) Query package managers (dpkg, rpm, apt) for current FFmpeg version across fleet; flag any host below 8.1.2 as unpatched-and-exposed. No confirmed malicious file hashes, C2 domains, or IP indicators are available from the sourced reporting; update detection rules as JFrog releases full technical disclosure.

Indicators of Compromise

Type	Value	Context	Confidence
URL	`https://www.bleepingcomputer.com/news/security/ffmpeg-fixes-pixelsmash-flaw-in-widely-used-video-decoder/`	BleepingComputer coverage of CVE-2026-8461 PixelSmash — T3 source, human validation recommended	LOW
URL	`https://nvd.nist.gov/vuln/detail/CVE-2026-8461`	NVD canonical record for CVE-2026-8461 — T1 source	HIGH

Framework Mappings

MITRE-ATTACK

- **T1190** — Exploit Public-Facing Application
- **T1203** — Exploitation for Client Execution
- **T1059** — Command and Scripting Interpreter
- **T1195.002** — Compromise Software Supply Chain

NIST-800-53R5

- **CA-8** — Penetration Testing
- **RA-5** — Vulnerability Monitoring and Scanning
- **SC-7** — Boundary Protection
- **SI-2** — Flaw Remediation
- **SI-7** — Software, Firmware, and Information Integrity
- **SI-3** — Malicious Code Protection
- **SI-4** — System Monitoring
- **CM-7** — Least Functionality
- **SA-9** — External System Services
- **SR-3** — Supply Chain Controls and Processes
- **SI-16** — Memory Protection

CIS-V8

- **16.10** — Apply Secure Design Principles in Application Architectures
- **7.3** — Perform Automated Operating System Patch Management
- **7.4** — Perform Automated Application Patch Management

ISO-27001-2022

- **A.8.8** — Management of technical vulnerabilities

MITRE ATT&CK Mapping

Technique ID	Technique Name	Tactic
T1190	Exploit Public-Facing Application	Initial-Access
T1203	Exploitation for Client Execution	Execution
T1059	Command and Scripting Interpreter	Execution
T1195.002	Compromise Software Supply Chain	Initial-Access

Sources

Source	URL	Tier
Security News	https://www.bleepingcomputer.com/news/security/ffmpeg-fixes-pixelsm...	T3
CVE-2026-8461 - CVE Record	https://www.cve.org/CVERecord?id=CVE-2026-8461	T3
CVE-2026-33461: Kibana Information Disclosure Vulnerability	https://www.sentinelone.com/vulnerability-database/cve-2026-33461/	T3
Security News - TheHackerWire	https://www.thehackerwire.com/security-news/	T3
CVE-2026-0846 - Red Hat Customer Portal	https://access.redhat.com/security/cve/cve-2026-0846	T3
NVD	https://nvd.nist.gov/vuln/detail/CVE-2026-8461	T1

Generated 2026-06-23 06:48 UTC by TJS Security Command Center