# CVE202644212 Patch Guide for PrestaShop 8.2.6 and 9.1.1

CVE202644212 is a critical stored crosssite scripting flaw (CVSS 9.3, CWE79) in the backoffice Customer Service view of PrestaShop. Any visitor can plant the payload through the public Contact Us form and it fires when an employee opens the resulting thread. Stores on the 8.x line below 8.2.6, or on the 9.x line (from 9.0.0alpha.1) below 9.1.1, are affected. The fixes ship in 8.2.6 and 9.1.1 and the security advisory (GHSAw9f3qc75qgx9) states that no workaround exists.If you only have five minutes confirm your version, book the upgrade and tell support staff not to open unreviewed contact form threads in a privileged session until it lands.This page is the operations side of the story. For the mechanics of the bug itself, read our complete CVE202644212 breakdown. Here we focus on what to do, in what order and how to prove the job is finished.




## Why This Security Advisory Deserves a SameWeek Patch

The CVE202644212 vulnerability is unusual because the attacker needs nothing from you. No account, no purchase history, no guessable URL. The contact form is enabled, the Customer service subject is active by default and the only visitor controlled value that reaches a privileged screen is the email address.

Two details make the risk more practical than the score suggests

- The payload sits quietly. Stored XSS does not need the attacker to be online when it fires. A thread submitted on Friday evening can be opened on Monday morning by whoever works the support queue.
- The victim is the most privileged person in the building. Support staff often hold broad backoffice permissions because small teams share roles. A script running in their browser inherits everything their session can do.

The CVSS vector is AVN/ACL/PRN/UIR/SC/CH/IH/AN. The one friction point is user interaction, which is exactly why patch timing matters; you cannot predict when an employee will open the thread, so you cannot count on that friction to protect you.

Hadrian research writeup on stored XSS in the PrestaShop back office via RFC 5321 quoted string email is the most detailed public analysis of the root cause and the fix commits. We lean on it throughout this runbook.

## Affected Versions at a Glance

The CVE record lists two affected ranges. The table below turns them into a decision.

| Your version | Status | Required action |
|---|---|---|
| 8.2.5 or earlier on the 8.x line | Vulnerable | Upgrade to 8.2.6 or later |
| Releases older than 8.0 that the record covers as below 8.2.6 | Vulnerable | Plan a larger upgrade; use the manual hotfix as a bridge only after checking your template paths |
| 9.0.0alpha.1 through 9.1.0 | Vulnerable | Upgrade to 9.1.1 or later |
| 8.2.6 or later | Fixed | Verify, then clean up old threads |
| 9.1.1 or later | Fixed | Verify, then clean up old threads |

Two notes on reading that table. First, the fix appears in two release lines at the same time because maintainers landed the change in both branches together. Second, a version number alone proves the code is patched, not that your store was clean before you patched. We handle that in the cleanup section.

## The First Hour Triage Before You Touch Anything

Resist the urge to start upgrading the moment you read a 9.3. A calm hour of triage prevents the two classic mistakes patching the wrong store and destroying evidence.

## 1. Build an inventory

List every PrestaShop installation you are responsible for, including staging copies, regional storefronts that share a code base and the forgotten shop a former colleague launched for a campaign. Staging servers matter because they often hold real customer messages and are rarely monitored.

## 2. Read the version from the source of truth

The backoffice dashboard shows the version in the footer area and Advanced Parameters holds a configuration information page. For a fleet, script the check against each installation's version constant rather than relying on memory. Record the result in one sheet so you can prove coverage later.

## 3. Decide who may open the Customer Service queue

Until each store is patched, ask staff to avoid opening new, unfamiliar threads from within a fully privileged session. If your team has a readonly role that can triage tickets, use it temporarily. This is a stopgap, not a fix, but it narrows the window.

## 4. Snapshot first

Take a file and database backup before any change. The backup also preserves the thread table as it stood before you touched it, which matters if you later need to establish whether a malicious submission ever existed.The crosssite scripting labs go deeper on stored, reflected and DOM variants.

# Upgrade Paths 8.x to 8.2.6 and 9.x to 9.1.1

Both fixes are small in code terms, but you should still treat the upgrade like any production release.For the 8.x line move to 8.2.6. If you run containers, the official image tags for 8.2.5 and 8.2.6 both exist, which makes it easy to reproduce the vulnerable and patched behavior side by side in a test environment.For the 9.x line move to 9.1.1. Stores still on 9.0.x are inside the affected range, so they need a minor upgrade rather than a patchlevel one. Check thirdparty module compatibility before you start, because module vendors do not always certify new minor versions on the day they ship.

A sensible sequence for both lines

1. Clone production into a staging environment with a recent database copy.
2. Run your normal upgrade procedure, whether that is an upgrade module, a command line flow, or a rebuilt container.

3. Clear the template and application caches. Smarty caches compiled templates, so skipping this step can leave a stale, unpatched copy of the vulnerable view serving requests.
4. Smoketest the Customer Service screen, order management and the contact form.
5. Repeat on production in a low traffic window and keep the rollback plan visible.

If your theme or custom modules override backoffice templates, check for overrides of the customer threads view. An override that copies the old markup can quietly reintroduce the unescaped output even after the core file is fixed.

## If You Cannot Upgrade Today The Manual Hotfix

Sometimes a module incompatibility or a change freeze blocks the upgrade. The published analysis describes two small edits that replicate the official fix. Treat them as a bridge, record them in your change log and expect an upgrade to overwrite them.A fixed server template does not protect a custom storefront script that writes URL data into the page, which is the territory of DOMbased XSS.

### Edit 1 escape the email in the thread view

In the Customer Service thread template (under the backoffice theme, at `controllers/customer_threads/helpers/view/view.tpl`), the email is printed twice without escaping once inside a heading and once as the value of a hidden input. Add the HTML escape modifier to bothOn production installations the admin folder usually carries a randomized name rather than the development folder name, so locate the file accordingly. Clear the cache afterward.

### Edit 2 tighten the email validator

In the validation class, the email check calls the Symfony Email constraint in loose mode. Changing the mode to strict makes the validator reject addresses that raise RFC warnings and quoted string local parts always raise one. The side effect is that a vanishingly rare legitimate address using a quoted local part will be refused. For a retail contact form, that trade is almost always acceptable.

### A WAF rule as a third layer

A web application firewall rule can block contact form submissions where the email field contains a URLencoded double quote followed by attribute style keywords. Treat this as detection and friction, not protection. Encoding tricks and alternative attribute names mean a rule alone will not hold, which is why the advisory is blunt about upgrading.

# CleanUp After Patching Old Threads Still Matter

Patching closes the door. It does not tell you whether someone already walked through it.

## Hunt for suspicious stored emails




Legitimate customers almost never submit a quoted string address. The thread table stores the sender email, so a readonly query that looks for values starting with a double quote, or containing spaces, finds the odd ones quickly. Adjust the table name to your database prefix and run it against a replica if you canReview every hit by reading the raw value in the database client, not by opening the thread in the back office.

## Decide what a hit means

- Hit exists, no employee ever opens the thread, delete or archive the row, document it and move on. The attacker's payload never ran.
- Hit exists and the thread was opened before the patch assumed the script may have executed. Invalidate sessions, force password resets for every employee who could have opened it and review recent backoffice activity such as new employees, changed permissions, installed modules and edited payment or shipping settings.
- No hits, you still have a clean bill for this vector, but keep the patched validator and template as the real control.

### Why existing rows are no longer dangerous

After the template fix, stored values are escaped when rendered, so an old malicious row displays as inert text. That is good, but it also means the thread will look harmless in the interface. Do not rely on the screen to tell you whether a row was hostile. The database does. The same trustboundary thinking that explains this XSS also explains SQL injection.

## Verifying That the Fix Works

Verification should be boring and repeatable. On a staging copy, submit a contact form entry with an inert quoted string address, such as a quoted label at a reserved test domain. On a patched store, the form should reject it with an invalid email message. On a vulnerable store, the form accepts it and stores it, which is how the public scanning approach identifies exposed versions.Because a positive result creates a real thread, do this only on systems you own or are authorized to test and delete the test thread afterward. For production, prefer version evidence plus a code comparison of the template and validator over firing probes at a live shop.Finally, view the Customer Service thread page source on staging and confirm that the email appears with encoded entities rather than raw quote characters.

## Reading the Score: How Urgent Is It Really?

A 9.3 sounds like an emergency and for exposed back offices it should be treated as one. It helps to separate the score from the realworld signals so you can explain priority to nontechnical stakeholders.

At the time of writing, public trackers show a low exploitation prediction value and the CISA enrichment lists exploitation as none, automation as not possible and technical impact as total. Read that as no confirmed mass exploitation, but if someone does target you, the consequence is complete control of the vulnerable component. Check those sources again before each decision, because they change.Compare that profile with an authentication bypass such as CVE202641940 in cPanel, where an attacker can act the moment they find a vulnerable host. CVE202644212 needs a human to open a thread, which slightly lowers its automation potential and shifts the priority from tonight to this week. That ordering is a judgment call, not a rule and any store that handles a high volume of messages or sensitive customer data should tighten it.

## Beyond the Patch Hardening the Back Office

A patched store is safer, but this bug class tends to come back in other fields. Use the incident to harden the surrounding area.

- Limit roles. Give support staff only the permissions they need. An XSS in a privileged session is far less useful when the session cannot install modules or change payment

settings. Reviewing role boundaries is also practice for spotting IDOR and broken access control issues, which often surface alongside rendering bugs.
- Shorten sessions and protect cookies. Prefer HttpOnly and Secure flags and short lifetimes for privileged users. If any of your integrations or headless front ends use bearer tokens, the JWT security guide explains how to keep a stolen token from becoming a long lived foothold.
- Treat admin templates as in scope. Include backoffice views and custom module templates in every code review. Teams that only audit the storefront miss exactly this kind of flaw.
- Review custom queries. While you are in the database, check that custom modules use parameterized queries.

If your team wants a safe place to practice finding and fixing flaws like this one, handson web application hacking labs let developers and testers work through injection and rendering bugs without touching a real store.

## Conclusion

CVE202644212 is a small bug with a large blast radius. Two missing escape modifiers and a permissive validator turned a public contact form into a route into the back office. The remediation is an equally compact upgrade to 8.2.6 or 9.1.1, clear caches, check template overrides and hunt through old threads for anything unusual. Where an upgrade is blocked, the twoedit hotfix buys time and a clean evidence trail turns a stressful week into a closed ticket.The bigger value comes from treating the patch as a prompt to look harder at everything customers can type into and employees can later see. If you want your team to build that instinct on safe targets, explore the challenges on AppSecMaster, then start practicing.

## Frequently Asked Questions (FAQs)

## Which PrestaShop versions are vulnerable to CVE202644212?

Versions below 8.2.6 on the 8.x line and versions from 9.0.0alpha.1 up to but not including 9.1.1 are affected. The CVE record expresses the first range as below 8.2.6, so older releases are inside it as well.

### Is there a workaround if I cannot upgrade right away?

The advisory lists no official workaround. A manual hotfix that escapes the email in the thread view and switches the email validator to strict mode replicates the fix in the interim, but upgrading remains the correct remediation.

## Does patching remove malicious threads that were already submitted?

No. The fix escapes output and rejects new quotedstring emails, but rows already in the database stay there. Query the thread table, review suspicious entries in the database client and delete or archive them.

## Do I need to reset passwords after patching?

Only if there is a reason to believe a poisoned thread was opened before the patch. If a suspicious row exists and an employee viewed it, invalidate their sessions, reset credentials and review recent backoffice changes.

## How can I confirm the patch is working?

On staging, submit an inert quote string email through the contact form. A patched store rejects it with an invalid email message. You can also inspect the thread page source and confirm the email is HTML encoded.