Mauricio Zaragoza

Professor Webster

IST 590

17 May 2018

**Vulnerabilities using WP-CVE**

**Denial of Service (DoS):**

One of the first vulnerability I have come across was discovered by Barak Tawily. Barak Tawily is a security researcher from Israel and has identified what appears to be a Denial of Service vulnerability in WordPress CMS platform. It is said that the vulnerability was discovered in a built-in script in WordPress CMS within the load-script.php. The vulnerability (CVE-2018-6389) is not only effective but very simple. Here is what was happening, the vulnerability was allowing users to take down several WordPress websites with a single machine. The crazy part is, this DoS was possible without using a large amount of bandwidth. The load-scripts.php was a file that was designed for administrator users only. The script was meant to help improve the website performance and allow for the website to load at a faster rate by combining files into just one request (Kumar, 2018). Unfortunately, due to the company denying the vulnerability, the vulnerability was left unpatched. Therefore, the vulnerability was able to affect multiple versions of WordPress, including version 4.9.2 and older.

Because the authors failed to keep any authentication, any user was able to have access. Ultimately, attackers can use DoS by using the large list of registered .js files. This will construct multiple requests to load every file multiple times. The exploit allows any user to force load-scripts.php to call all possible JavaScript files into one and go by bypassing their names into the URL <https://your-wordpress-site.com/wp-admin/load-scripts.php?c=1&load=editor,common,user-profile,media-widgets,media-gallery> (Kumar, 2018 ). Although the vulnerability continues unpatched, researchers have a simple bash script that was able to fix the issue.

**Cross-Site Scripting (XSS)**

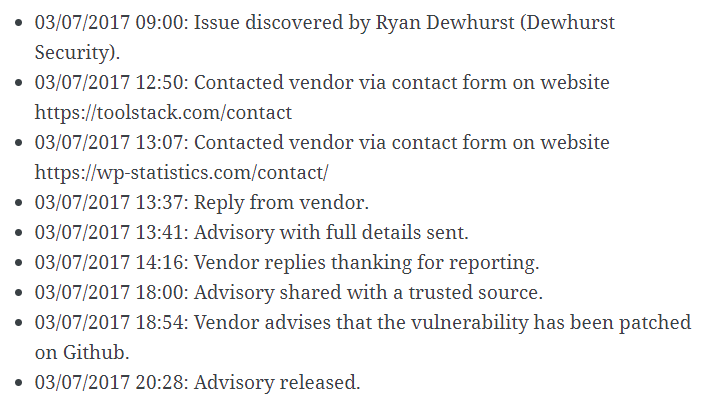
Apparently, there has been an identified XSS vulnerability that was discovered. It appears that versions 12.0.8.1 and older were affected where affected by this vulnerability in WP statistics. This vulnerability was found in WP statistics WordPress Plugin, which is connected to roughly 300,000 websites (Hurst, 2017). The vulnerability consisted of an authentication XSS which the GET parameter on the ‘wps\_visitors\_page’ was being output to a page without any validation or output encoded. Overall, this lead to an Authenticated Reflected Cross-Site Scripting (XSS). This would then allow for an attacker to compromise WordPress application by tricking any authenticated administrator user to click on a link that was specially crafted (Hurst, 2017). On the other hand, there were other XSS vulnerabilities that were identified but were able to stay protected due to Cross-Site Request Forgery (CSRF). Finally, the vulnerability was able to be patched within hours of being discovered and updating to version 12.0.9. The vendor was able to use $\_get variable through WordPress’ esc\_attr() function. This was able to fix the vulnerability in version 12.0.9.

**Technical:**

The following was provided by Dew Hurst Security blog:

* Source: On line 28 of the includes/log/last-visitor.php file, the $\_GET[‘ip’] is placed within the $\_get variable.
* Sink: On line 74 of the includes/log/last-visitor.php file, the $\_get variable is output in the PHP echo() function.

**Timeline:**



**SQL Injection**

The next vulnerability was due to a SQL injection. The vulnerability was first identified by Anthony Ferra yet was not fully fixed but instead ‘partially’ fixed. The SQL injection vulnerability on WordPress which was identified in September 2017. According to Ferra, after identifying the vulnerability, Ferra contacted the WordPress team of the issue. Although they had sent out a ‘fix’ it was of little use as it would only narrow down to a subset of the potential exploits. The WP team should have fixed the root issue. As a consequence, this team would face bigger issues. Within days of the release of the patch, Ferra found a new vulnerability. Ferra would continue to communicate with the WP team, but the team had failed to act on the vulnerability for several weeks. With the updated version 4.8.2, the ‘fix’ did not really fix anything. Instead, the vulnerability broke a ton of third-party code as well as sites. The amount of code was reported to be roughly 1.2 million lines of code (Ferra, 2017).

On 20 October 2017, WP says they are working on fixing the bug and will fix the meta.php. Through out the next few days, WP would be discussing the issue and considering using a “comment marker” to indicate if a string has been through WPDB::prepare() before. Later, WP wanted to push-back \_real\_escape side of the issue and es\_sql() was passed (Ferra, 2017).

The upgrade to 4.8.3 would update any plugins that override $wpdb. Also, it was recommended that all should upgrade to wp-dp.php for all clients. Anthony Ferra also suggests some firewall rules to be implemented such as blocking % and sprintf() values. Finally, Ferra recommended to get rid of all the prepare mechanism, which will return a sting SQL query.

**CSRF**

In 2018, there were more vulnerabilities associated with WordPress. This vulnerability would allow for a user to conduct cross-site request forgery attacks. This allows users to consume large amounts of memory on the system that is being targeted also allowing users to bypass security controls. The problem was that the software did not filter HTML code properly from user-supplied input before it could be display the input. Unfortunately, a remote user can cause arbitrary scripting code to be executed by the target user's browser (Nylen, 2018). The code will originate from the site running the WordPress software and will run in the security context of that site. As a result, the code will be able to access the target user's cookies (including authentication cookies), if any, associated with the site, access data recently submitted by the target user via web form to the site, or take actions on the site acting as the target user (Nylen, 2018).

A remote user can create a specially crafted HTML page or URL that, when loaded by the target authenticated user, will take actions on the target interface acting as the target user via the Press. The vulnerability [CVE-2017-6819] can be exploited to consume excessive server resources. These vulnerabilities were reported by Chris Andre Dale, Yorick Koster, Simon P. Briggs, Daniel Chatfield, xuliang, Marc Montpas, Delta, and Sipke Mellema

**SQL injection**

This vulnerability is a SQL injection. The vulnerability allows a user’s interface for assigning taxonomy terms regardless if the user has permissions to use the interface or not. WP\_Query does not properly validate user-supplied input ([Campbell](https://profiles.wordpress.org/aaroncampbell), 2017). Users can supply a specially crafted parameter value to execute SQL commands on the underlying database. Thus, the post list table does not properly filter HTML code from user supplied input before displaying the input. So, by causing arbitrary code, users can use this to execute the targets browser. The code originates from the site running WP software and will run the security context of that specific site. By doing this, the code will be able to access the cookies of desired target. The problem with this is the attacker will be able to target authentication cookies as well, if any available. Also, the vulnerability allows for a user to remotely exploit a flaw in REST API endpoint to gain privileges on the system that is being targeted. This vulnerability can exploit sensitive information from the intended target. Finaly, several SQL commands will be able to be executed remotely on the underlying database. The solution to this vulnerability was issuing a patch that would be available with 4.7.2.

* The user interface for assigning taxonomy terms in Press This is shown to users who do not have permissions to use it. Reported by David Herrera of Alley Interactive.
* WP\_Query is vulnerable to a SQL injection (SQLi) when passing unsafe data. WordPress core is not directly vulnerable to this issue, but we’ve added hardening to prevent plugins and themes from accidentally causing a vulnerability. Reported by Mo Jangda.
* WP\_Query is vulnerable to a SQL injection (SQLi) when passing unsafe data. WordPress core is not directly vulnerable to this issue, but we’ve added hardening to prevent plugins and themes from accidentally causing a vulnerability. Reported by Mo Jangda.
* WP\_Query is vulnerable to a SQL injection (SQLi) when passing unsafe data. WordPress core is not directly vulnerable to this issue, but we’ve added hardening to prevent plugins and themes from accidentally causing a vulnerability. Reported by Mo Jangda ([Campbell](https://profiles.wordpress.org/aaroncampbell), 2017).

**Cited Sources**

[Aaron D. Campbell](https://profiles.wordpress.org/aaroncampbell), SecurityTracker, <https://www.securitytracker.com/id/1037731>, Feb 10, 2018, Web, May 17, 2018

Anthony Ferra, ircmaxell’s blog, <https://blog.ircmaxell.com/2017/10/disclosure-wordpress-wpdb-sql-injection-technical.html>, Oct 27, 2018, web, May 17, 2018

Dew Hurst, Dew Herst Security Blog, <http://blog.dewhurstsecurity.com/2017/07/03/wp-statistics-authenticated-xss-vulnerability.html>, July 03, 2018, web, May 16, 2018

James Nylen, WordPress.ORG, <https://wordpress.org/news/2017/03/wordpress-4-7-3-security-and-maintenance-release/>, March 06, 2018, web, May 17, 2018

Mohit Kumar ,The hacker news, <https://thehackernews.com/2018/02/wordpress-dos-exploit.html>, Feb 05, 2018, web, May 16, 2018