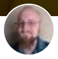

Ram Gall

May 3, 2021

SQL Injection Vulnerability Patched in CleanTalk AntiSpam Plugin

On March 4, 2021, the Wordfence Threat Intelligence team initiated responsible disclosure for a Time-Based Blind SQL Injection vulnerability discovered in Spam protection, AntiSpam, FireWall by CleanTalk, a WordPress plugin installed on over 100,000 sites. This vulnerability could be used to extract sensitive information from a site's database, including user emails and password hashes, all without logging into the site.

We initially reached out to the plugin's developer on March 4, 2021 and sent over the full disclosure on March 5, 2021. A patched version of the plugin, 5.153.4, was released on March 10, 2021.

Wordfence Premium users received firewall rules protecting against this vulnerability on March 4, 2021. Sites still running the free version of Wordfence received the same protection on April 3, 2021.

Description: Unauthenticated Time-Based Blind SQL Injection
Affected Plugin: Spam protection, AntiSpam, FireWall by CleanTalk
Plugin Slug: cleantalk-spam-protect
Affected Versions: < 5.153.4
CVE ID: CVE-2021-24295
CVSS Score: 7.5 (High)
CVSS Vector: CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N
Fully Patched Version: 5.153.4

The CleanTalk WordPress plugin has a number of uses, but one of its primary purposes is to protect sites against spam comments. Part of how it does this is by maintaining a blocklist and tracking the behavior of different IP addresses, including the user-agent string that browsers send to identify themselves.

Unfortunately, the `update_log` function in `lib/Cleantalk/ApbctWP/Firewall/SFW.php` which was used to insert records of these requests into the database failed to use a prepared SQL statement.

```php
public function update_log( $ip, $status ) {

    $id   = md5( $ip . $this->module_name );
    $time = time();

    $query = "INSERT INTO " . $this->db__table__logs . "
    SET
        id = '$id',
        ip = '$ip',
        status = '$status',
        all_entries = 1,
        blocked_entries = " . ( strpos( $status, 'DENY' ) !== false ? 1 : 0 ) . ",
        entries_timestamp = '" . $time . "',
        ua_name = '" . sanitize_text_field( Server::get('HTTP_USER_AGENT') ) . "'
    ON DUPLICATE KEY
    UPDATE
        status = '$status',
        all_entries = all_entries + 1,
        blocked_entries = blocked_entries" . ( strpos( $status, 'DENY' ) !== false ? ' + 1' : '' ) . ",
        entries_timestamp = '" . intval( $time ) . "',
        ua_name = '" . sanitize_text_field( Server::get('HTTP_USER_AGENT') ) . "'";

    $this->db->execute( $query );
}
```

There were a number of features to the plugin code that made it more difficult to successfully perform a SQL injection attack.

By design, the `update_log` function should only have been executed a single time for each visitor IP address. However, it was possible to manipulate the cookies set by the plugin, sending an initial request to obtain a `ct_sfw_pass_key` cookie and then manually setting a separate `ct_sfw_passed` cookie and disallowing it from being reset.

Additionally, the vulnerable SQL query used `INSERT` rather than `SELECT`. Since data was not being inserted into a sensitive table, the `INSERT` query could not be used by an attacker to exploit the site by changing values in the database, and this also made it difficult to retrieve any sensitive data from the database.

Finally, the SQL statement used the `sanitize_text_field` function in an attempt to prevent SQL injection, and the User-Agent was included in the query within single quotes.

Despite these obstacles, we were able to craft a Proof of Concept capable of extracting data from anywhere in the database by sending requests containing SQL commands in the User-Agent request header. This exploit could be used by unauthenticated visitors to steal user email addresses, password hashes, and other sensitive information.

Prepared Statements are Crucial

We were able to successfully exploit the vulnerability in CleanTalk via the Time-Based Blind SQL Injection technique, which sends requests that "guess" at the content of a database table and instructs the database to delay the response or "sleep" if the guess is correct. For example, a request might ask the database if the first letter of the admin user's email address starts with the letter "c", and instruct it to delay the response by 5 seconds if this is true, and then try guessing the next letters in sequence. There are a number of other SQL injection techniques that can work around many forms of traditional input sanitization depending on the exact construction of the vulnerable query.

This is why it is essential to "prepare" any database queries before actually sending them to the database. Prepared statements isolate each query parameter and are by far the most effective defense against SQL Injection. Fortunately, WordPress offers an incredibly easy way to do this, by using the `$wpdb->prepare()` function. If you develop WordPress plugins, themes, or any other software that interacts with a database, regularly using prepared statements will ensure your software will be far more secure.

Timeline

firewall rules to Wordfence Premium customers and initiate contact with the plugin developers.
March 5, 2021 – We send over the full disclosure to the plugin developers.
March 10, 2021 – A patched version of the plugin is released.
April 3, 2021 – Sites still using the free version of Wordfence receive protection against this vulnerability.

Conclusion

In today's post, we covered a SQL Injection vulnerability in the Spam protection, AntiSpam, FireWall by CleanTalk plugin which could be used to extract sensitive information from a site's database and explained why using prepared statements is a critical best practice for plugin developers.

This vulnerability was patched in version 5.153.4, and we strongly recommend updating to the latest version of the plugin, 5.156 as of this writing, immediately.

Wordfence Premium users received firewall rules protecting against these vulnerabilities on March 4, 2021, while those still using the free version of Wordfence received the same protection on April 3, 2021.

If you know a friend or colleague who is using this plugin on their site, we highly recommend forwarding this advisory to them to help keep their sites protected as this vulnerability allows a breach of any confidential data stored in a site's database.

Special Thanks to Threat Analyst Chloe Chamberland for her instrumental role in developing the Proof of Concept exploit for this vulnerability.