Penetration testing report: Executive Summary

Scope: 34.204.82.211 [ec2-34-204-82-211.compute-1.amazonaws.com], hereafter known as host\*.

As with all vulnerability scoping, we start with ‘nmap’ against the services exposed. We do that using the following command:

root@here# nmap -sC -sV -oA nmap/lateral 34.204.82.211

Doing so yields several interesting ports exposed. Ports like 80, 21, 3306 are interesting in the eyes of attacker as these are likely to contain vulnerable configurations.

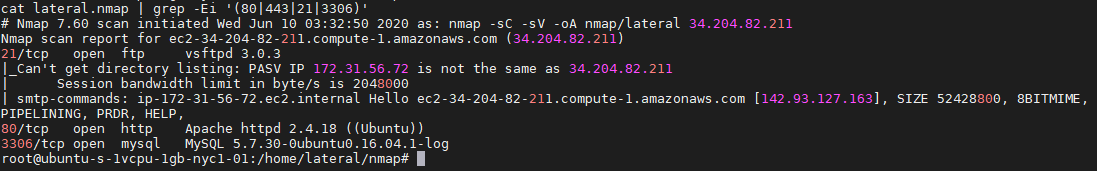


Fig 1: Exposed ports include ftp, http and mysql instances.

We then access the IP address to find a default apache2 page served by the apache process. The page itself does not look vulnerable and do not contain any entry points. We then choose ‘nikto’ as a tool to do a recon against the host to find it uses a wordpress installation. This is found under the header ‘x-pingback’. The header also reveals the wordpress installed under /wordpress directory.

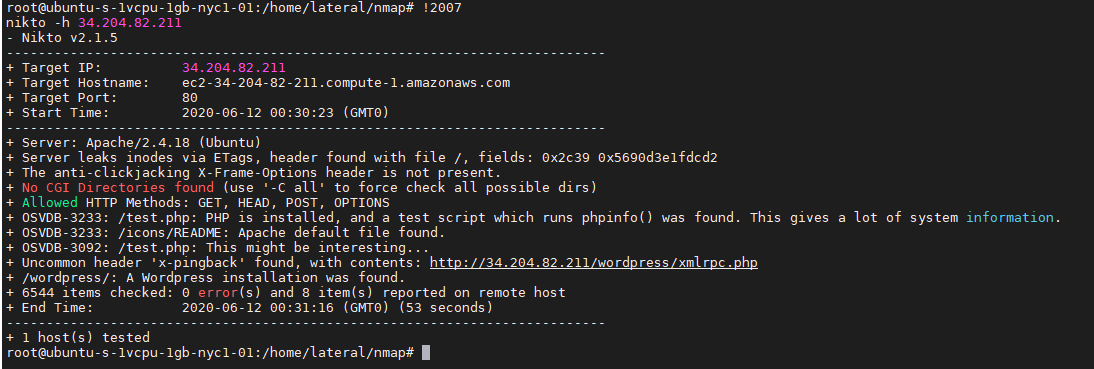


Fig 2: nikto identifies the host running wordpress.

If we notice in the figure 2, we see one xmlrpc.php file exposed. We use burp to change the ingestion method to POST figure out whether we can issue arbitrary XML codes there.

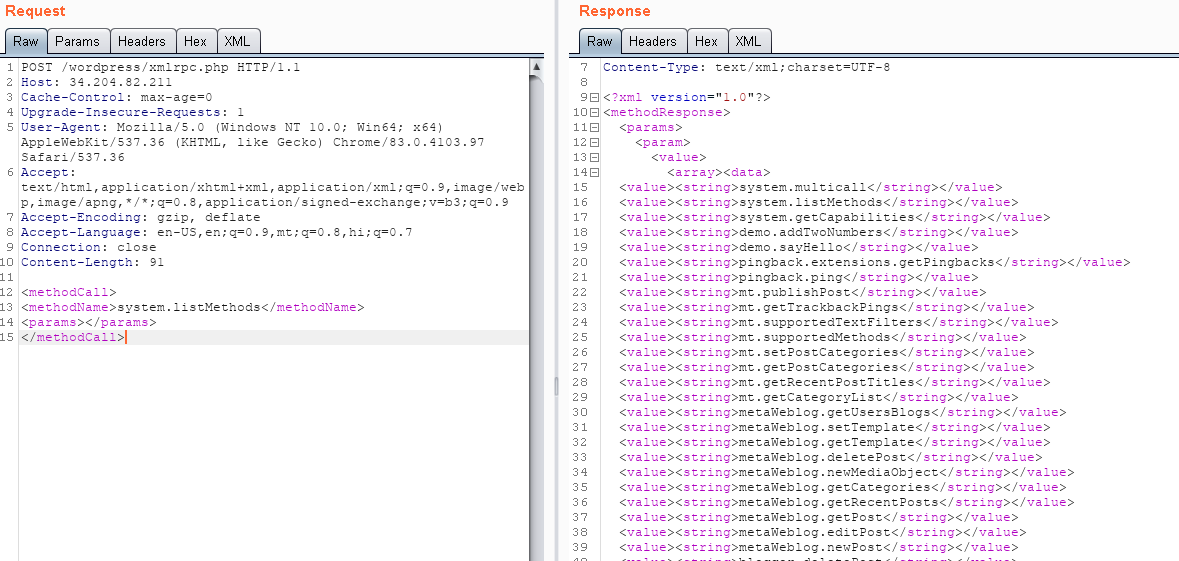


Fig 3: Changing the method to POST and passing all methods to xmlrpc.php.

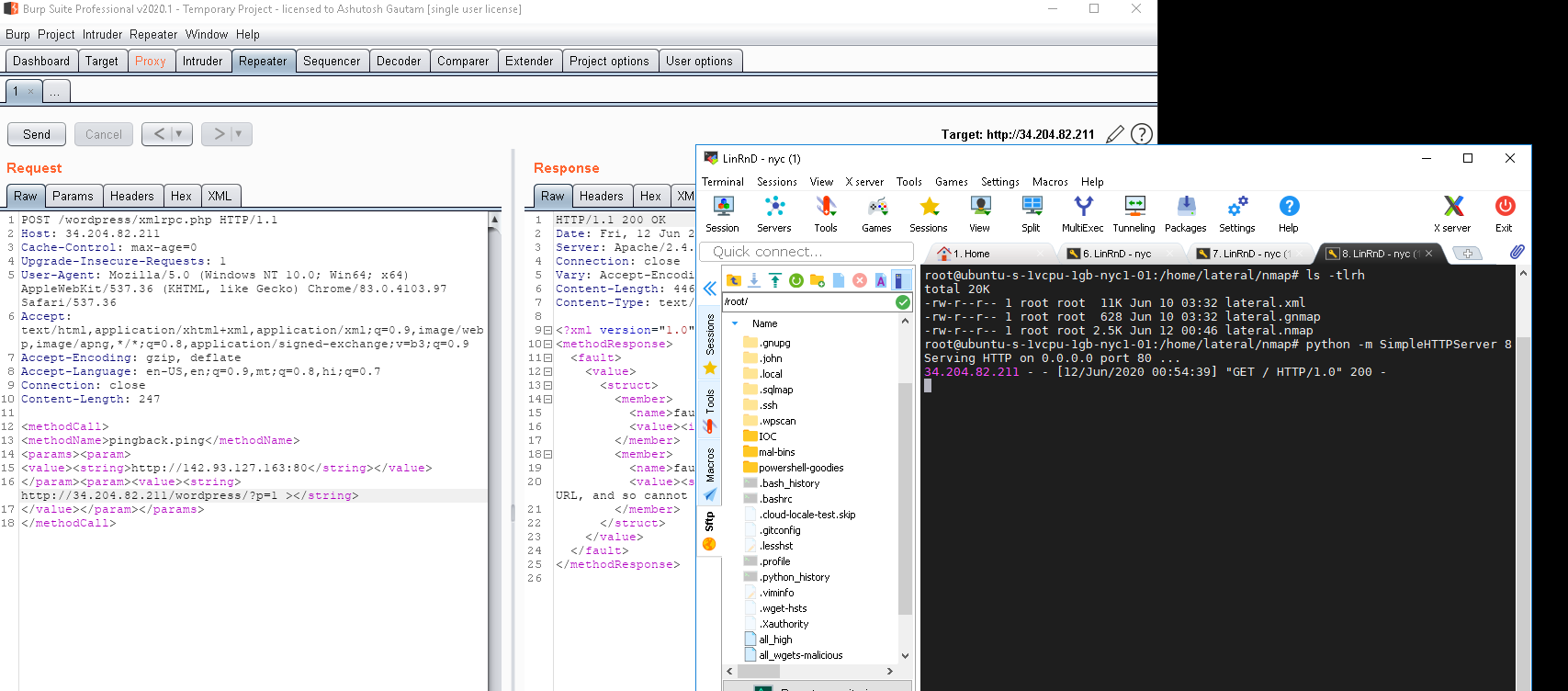
While the method do not necessarily represent a vulnerability in the application, these are known for being a conduit to launch other attacks such as RCE(in few cases) and DoS. These also are critical as if left exposed can be used by attackers to perform DoS attacks on behalf of ourselves – this works with pingback.ping method. 

Fig 4: Pingback.ping method calling external VPS at port 80.

As of this moment, we have identified only 1 potential vulnerability that comes out of the xmlrpc.php’s exposure. We now progress to the ‘nmap’ result that we have performed earlier in the test.

Revisiting nmap:

Nmap shows that we have port 21 open. This is of interest to us. We try to log on to the ftp service by being an anonymous user.

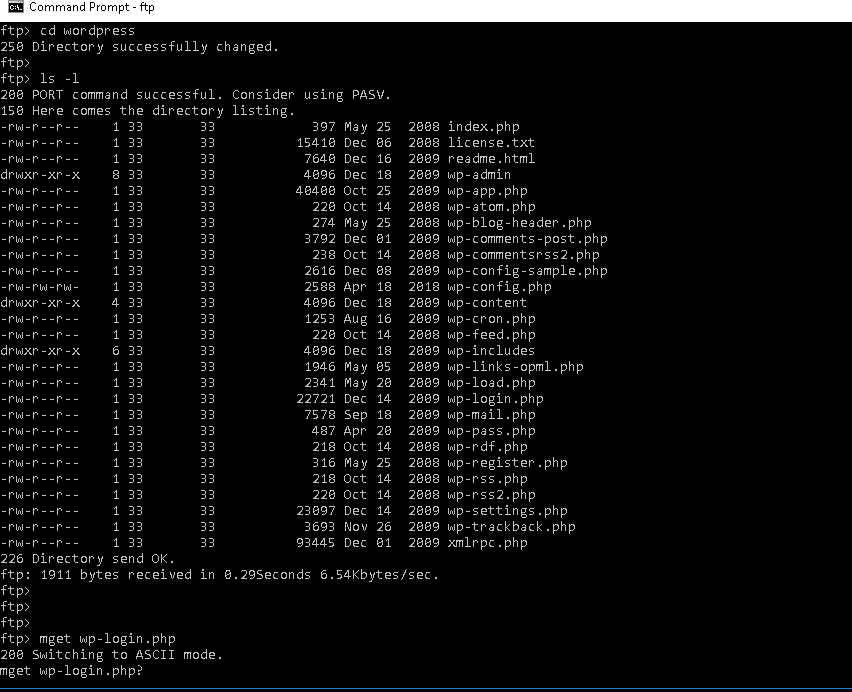


Fig 5: ftp permits the anonymous login. This is critical.

Now that we have access to the server, we can move onto the directory that’s of interest to us. These normally are /var/www/html/wordpress/\*. These contain several files that are of interest to us. One of such file is wp-config.php.

Upon perusing wp-config.php file, we see the mysqld serving the connection using root/root combo.

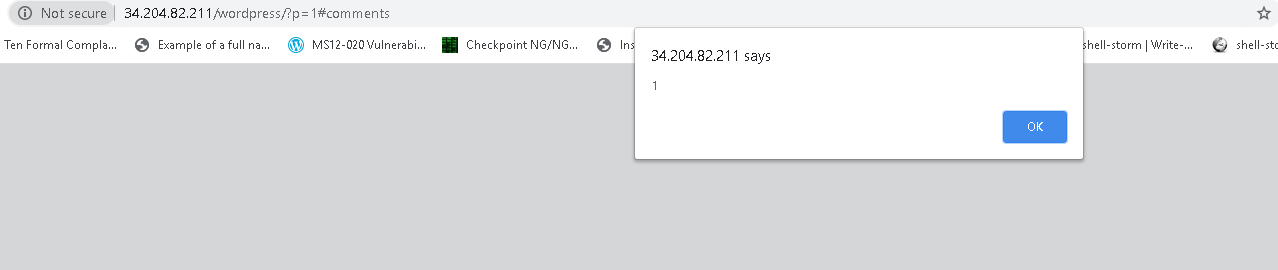
We then log on to the mysql instance using the credentials.



Fig 6: Using INSERT against the DB wordpress to create new users having administrator privilege.

By this time, the server has been compromised. We then use the user ‘abyss1’, and password ‘welcome1’ to log in to the wordpress instance.

Finally, after logging into the instance we release the comment we made prior to the attack containing XSS payload.



Final thoughts:

* We now have persistent XSS across the comments section.

**Replication**: Once we are admin, we release the comment that was marked for approval. In reverse, we can create a new comment putting the XSS trigger.

* We can further peruse this exploit into uploading a PHP reverse shell, exploit the www-apache user into giving us a root – a RCE in essence.