Incident Handling with Splunk

It is good to understand the following before completing this lesson:

- Splunk overview and basic navigation
- Important Splunk Queries
- Know how to use different functions/values to craft a search query
- How to look for interesting fields

Splunk logs attached in resource section, or you can use following link to download and add it splunk

https://github.com/splunk/botsv1?tab=readme-ov-file

Scenario

A Big corporate organization **Wayne Enterprises** has recently faced a cyber-attack where the attackers broke into their network, found their way to their web server, and have successfully defaced their website **http://www.imreallynotbatman.com**. Their website is now showing the trademark of the attackers with the message **YOUR SITE HAS BEEN DEFACED** as shown below.

They have requested "**US**" to join them as a **Security Analyst** and help them investigate this cyber attack and find the root cause and all the attackers' activities within their network.

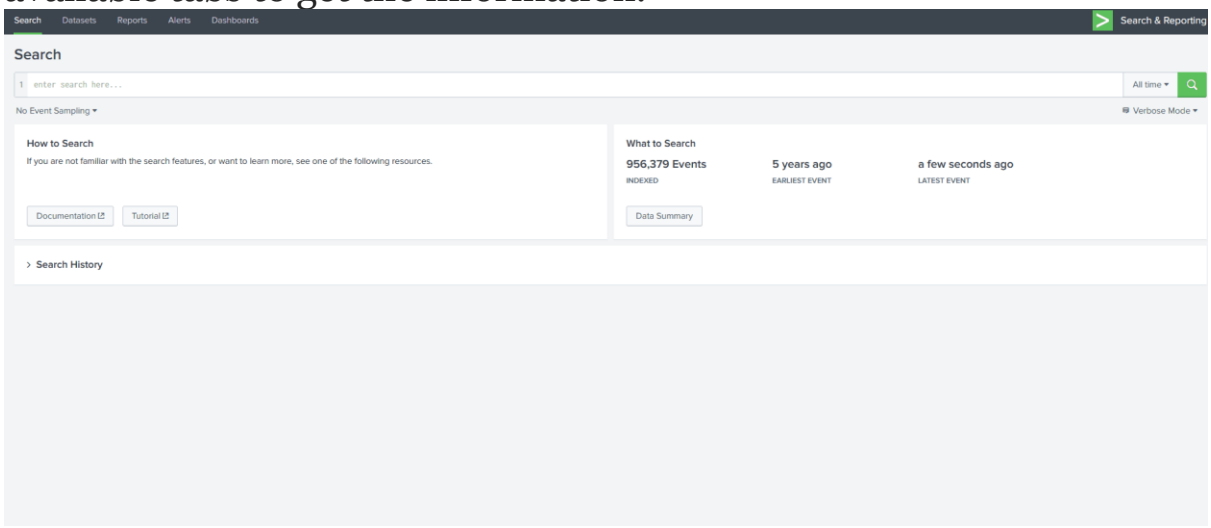
The good thing is, that they have Splunk already in place, so we have got all the event logs related to the attacker's activities captured. We need to explore the records and find how the attack got into their network and what actions they performed.

This Investigation comes under the `Detection and Analysis phase.`

Splunk

During our investigation, we will be using Splunk as our SIEM solution. Logs are being ingested from `webserver/firewall/Suricata/Sysmon` etc. In the data summary tab, we can explore the log sources showing visibility into both network-centric and host-centric activities. To get the complete

picture of the hosts and log sources being monitored in Wayne Enterprise, please click on the **Data summary** and navigate the available tabs to get the information.



Interesting log Sources

Some of the interesting log sources that will help us in our Investigation are:

Log Sources	Details
wineventlog	It contains Windows Event
winRegistry	It contains the logs related to registry creation / modification / deletion etc
XmlWinEventLog	It contains the sysmon event logs. It is a very important log source from an investigation point of view
fortigate_utm	It contains Fortinet Firewall
iis	It contains IIS web server
Nessus:scan	It contains the results from the Nessus vulnerability scanner
Suricata	It contains the details of the alerts from the Suricata IDS. This log source shows which alert was triggered and what caused the alert to get triggered— a very important log source for the Investigation
stream:http	It contains the network flow related to http traffic
stream: DNS	It contains the network flow related to DNS traffic
stream:icmp	It contains the network flow related to icmp traffic

Note: All the event logs that we are going to investigate are present in `index=botsv1`

Now that we know what hosts we have to investigate, what sources and the source types are, **let's connect to the lab and start Investigating**.
Room Machine

Before moving forward, deploy the machine. When you deploy the machine, it will be assigned an IP **Machine IP**: `MACHINE_IP`. The machine will take up to 3-5 minutes to start.

Task 4: Reconnaissance Phase

Reconnaissance Phase

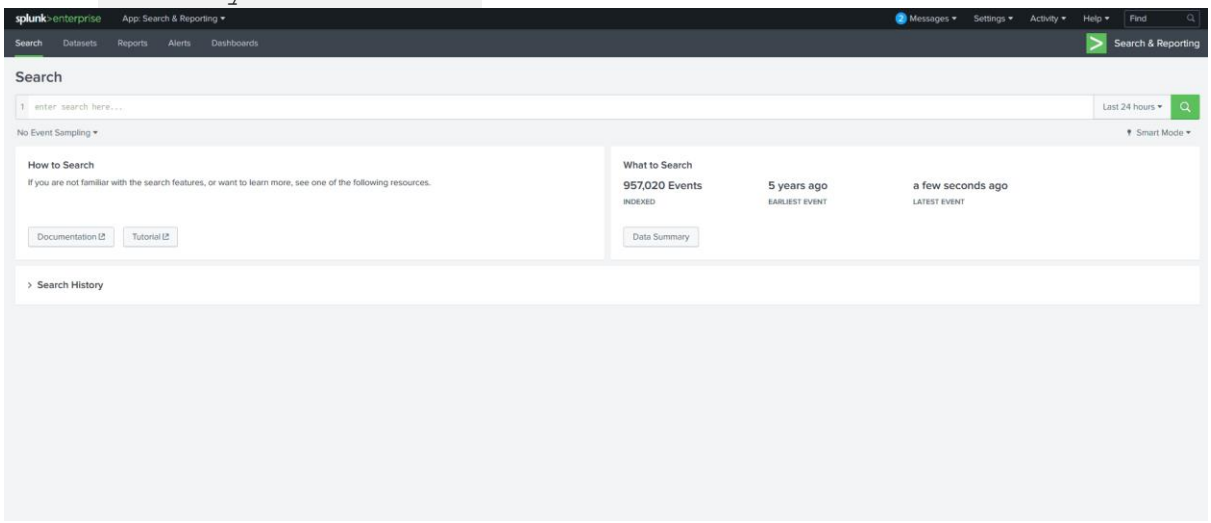
Reconnaissance is an attempt to discover and collect information about a target. It could be knowledge about the system in use, the web application, employees or location, etc.

We will start our analysis by examining any reconnaissance attempt against the webserver `imreallynotbatman.com`. From an analyst perspective, where do we first need to look? If we look at the available log sources, we will find some log sources covering the network traffic, which means all the inbound communication towards our web server will be logged into the log source that contains the web traffic. Let's start by searching for the domain in the search head and see which log source includes the traces of our domain.

Search Query:

```
index=botsv1 imreallynotbatman.com
```

Search Query explanation: We are going to look for the event logs in the index “botsv1” which contains the term `imreallynotbatman.com`



Here we have searched for the term `imreallynotbatman.com` in the index `botsv1`. In the sourcetype field, we saw that the following log sources contain the traces of this search term.

- Suricata
- stream:http
- fortigate_utm
- iis

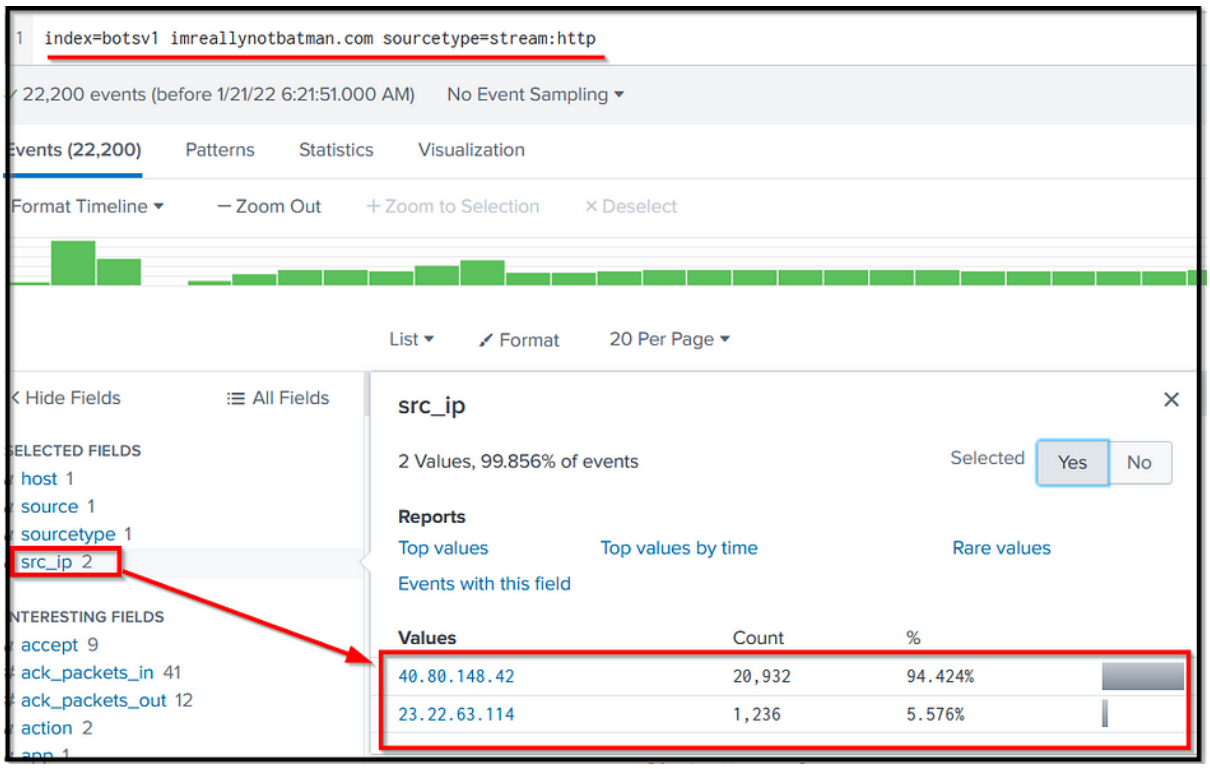
From the name of these log sources, it is clear what each log source may contain. Every analyst may have a different approach to investigating a scenario. Our first task is to identify the IP address attempting to perform reconnaissance activity on our web server. It would be obvious to look at the web traffic coming into the network. We can start looking into any of the logs mentioned above sources.

Let us begin looking at the log source **stream:http**, which contains the http traffic logs, and examine the `src_ip` field from the left panel. **Src_ip** field contains the source IP address it finds in the logs.

Search Query:

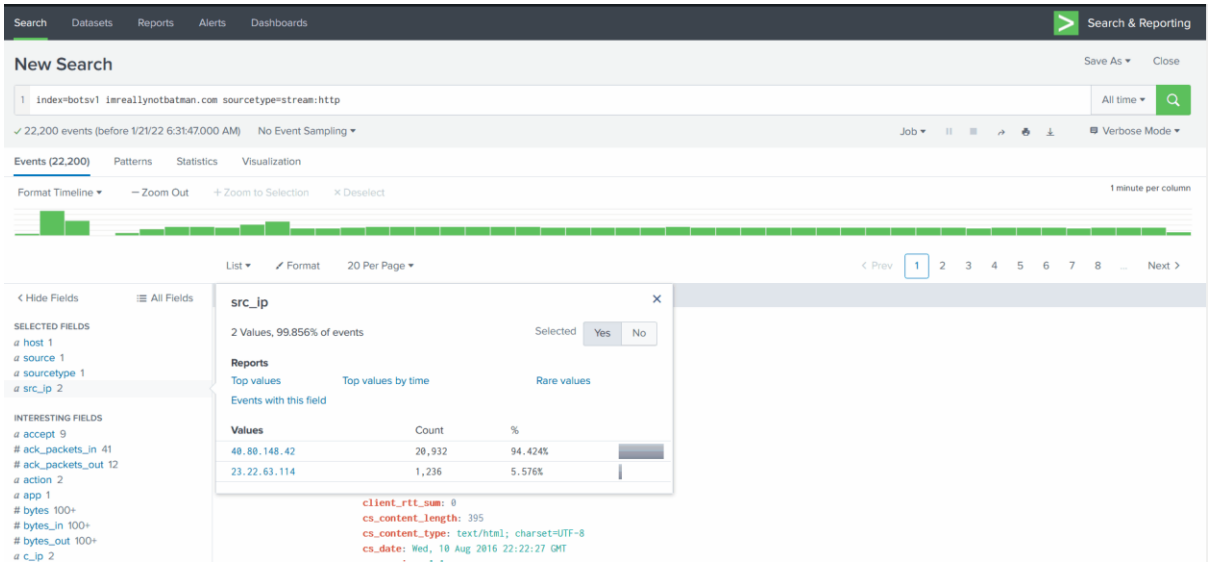
```
index=botsv1 imreallynotbatman.com sourcetype=stream:http
```

Search Query Explanation: This query will only look for the term `imreallynotbatman.com` in the **stream:http** log source.



Note: The important thing to note, if you don't find the field of interest, keep scrolling in the left panel. When you click on a field, it will contain all the values it finds in the logs.
So far, we have found two IPs in the src_ip field `40.80.148.42` and `23.22.63.114`. The first IP seems to contain a high percentage of the logs as compared to the other IP, which could be the answer. If you want to confirm further, click on each IP one by one, it will be added into the search query, and look at the logs, and you will find the answer.

To further confirm our suspicion about the IP address **40.80.148.42**, click on the IP and examine the logs. We can look at the interesting fields like User-Agent, Post request, URIs, etc., to see what kind of traffic is coming from this particular IP.



We have narrowed down the results to only show the logs from the source IP **40.80.148.42**, looked at the fields of interest and found the traces of the domain being probed.

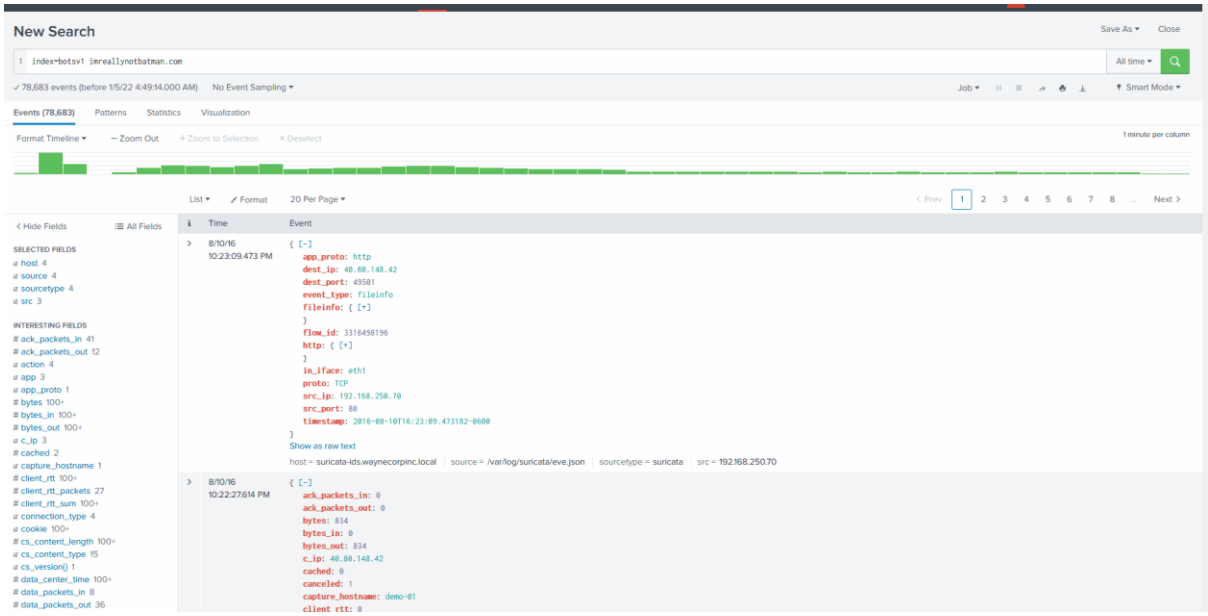
Validate the IP that is scanning

So what do we need to do to validate the scanning attempt? Simple, dig further into the weblogs. Let us narrow down the result, look into the `suricata` logs, and see if any rule is triggered on this communication.

Search Query:

```
index=botsv1 imreallynotbatman.com src=40.80.148.42 sourcetype=suricata
```

Search Query Explanation: This query will show the logs from the suricata log source that are detected/generated from the source IP **40.80.248.42**



We have narrowed our search on the **src IP** and looked at the source type `suricata` to see what Suricata triggered alerts. In the right panel, we could not find the field of our interest, so we clicked on more fields and searched for the fields that contained the signature alerts information, which is an important point to note.

Answer the questions below

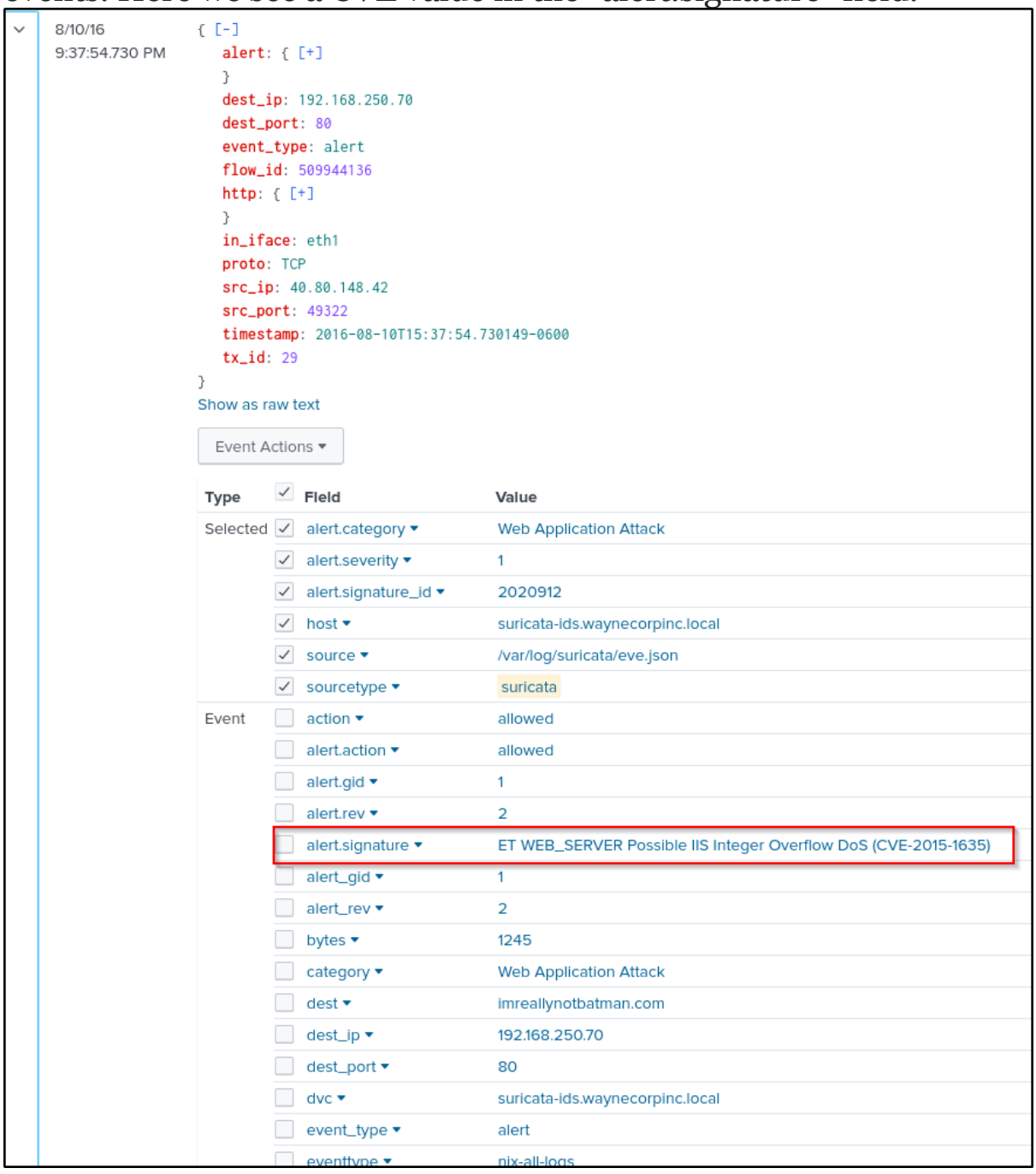
One suricata alert highlighted the CVE value associated with the attack attempt. What is the CVE value?

CVE-2014–6271

The query below is from the section on “Validating the IP that is scanning”. The source type is Suricata from source IP 40.80.148.42.

```
index=botsv1 imreallynotbatman.com sourcetype="suricata"
src_ip="40.80.148.42" CVE
```

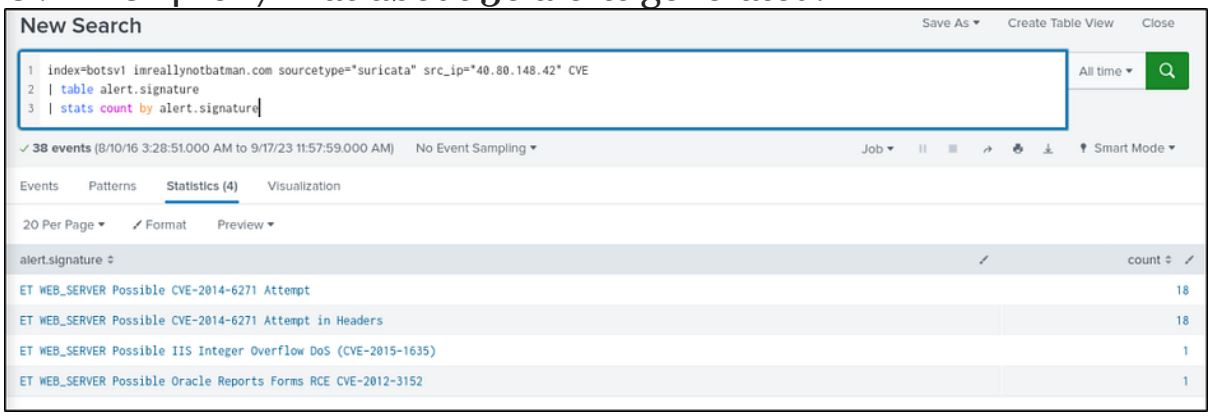
Where in the Field could contain the CVE value? Expand one of the events. Here we see a CVE value in the “alert.signature” field.



Modified the query to display the values in the “alert.signature” and to count the number of times the alert signature was generated.

```
index=botsv1 imreallynotbatman.com sourcetype="suricata"
src_ip="40.80.148.42" CVE
| table alert.signature
| stats count by alert.signature
```

CVE-2014–6271 has about 36 alerts generated.



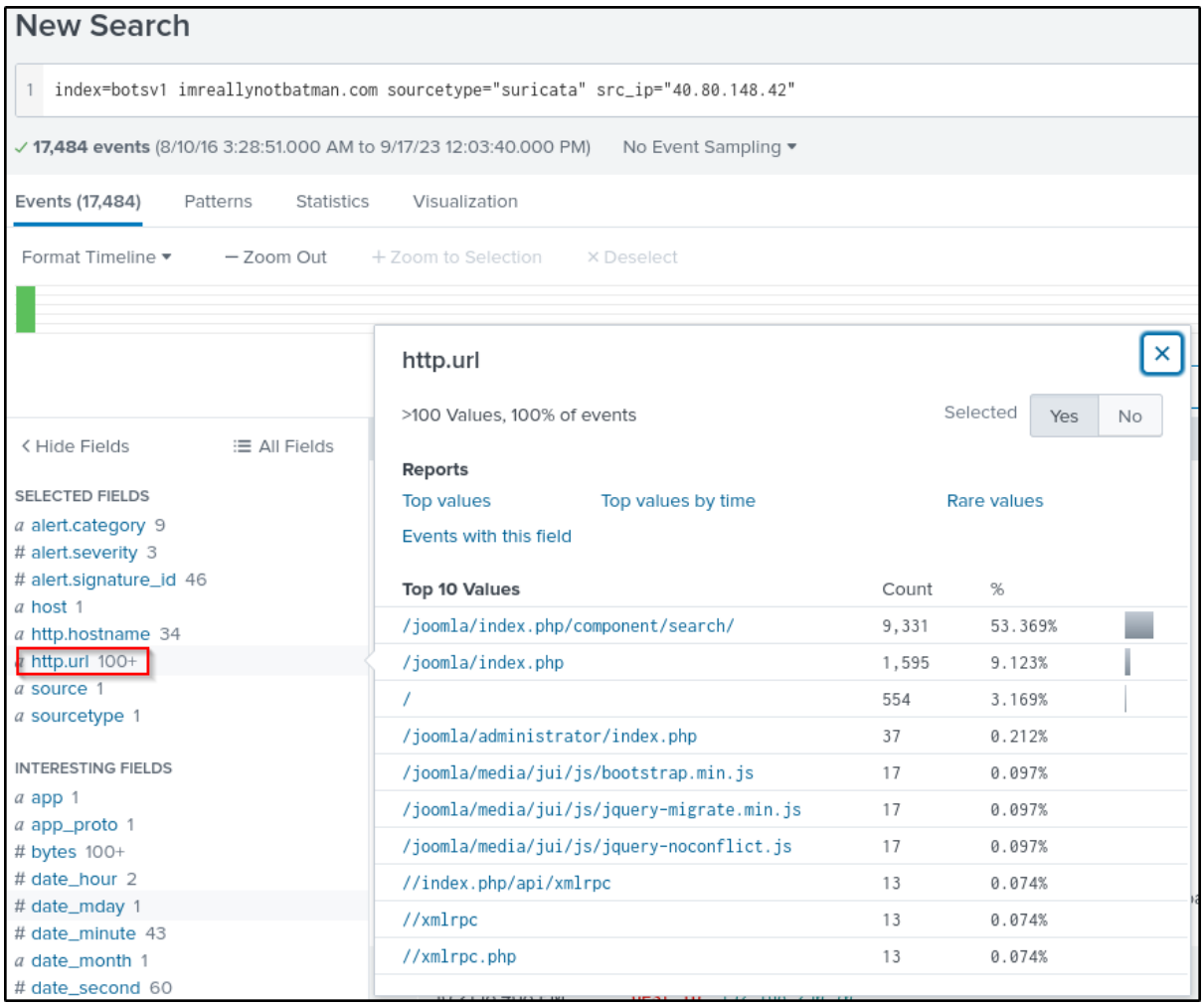
What is the CMS our web server is using?

joomla

Using the query below, select the field http.url to identify subdirectories/urls of the web server.

```
index=botsv1 imreallynotbatman.com sourcetype="suricata"
src_ip="40.80.148.42"
```

Here, “joomla” identified as the CMS of the web server. Joomla is a well known CMS or a Content Management system.



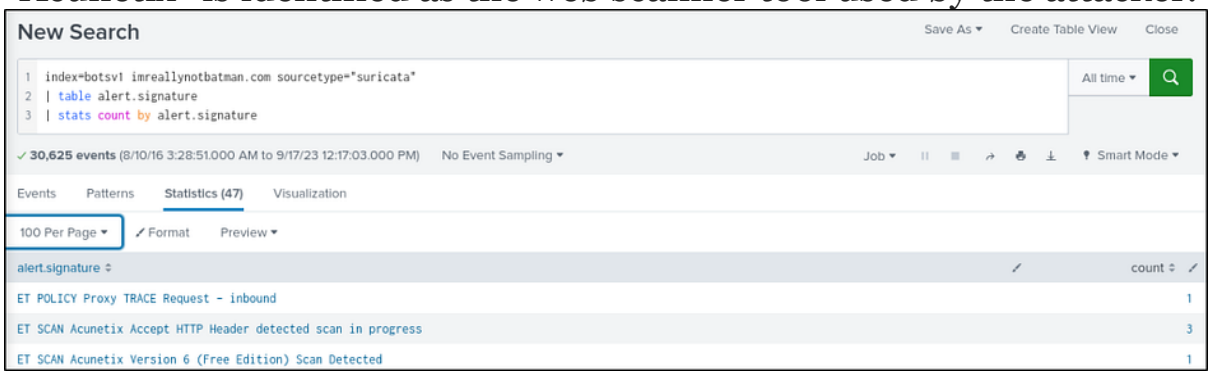
What is the web scanner, the attacker used to perform the scanning attempts?

acunetix

Suricata should have picked this type of event. This query will display the alert signatures in table format and to count the number of times the alert signature was generated.

```
index=botsv1 imreallynotbatman.com sourcetype="suricata" | table
alert.signature
| stats count by alert.signature
```

“Acunetix” is identified as the web scanner tool used by the attacker.



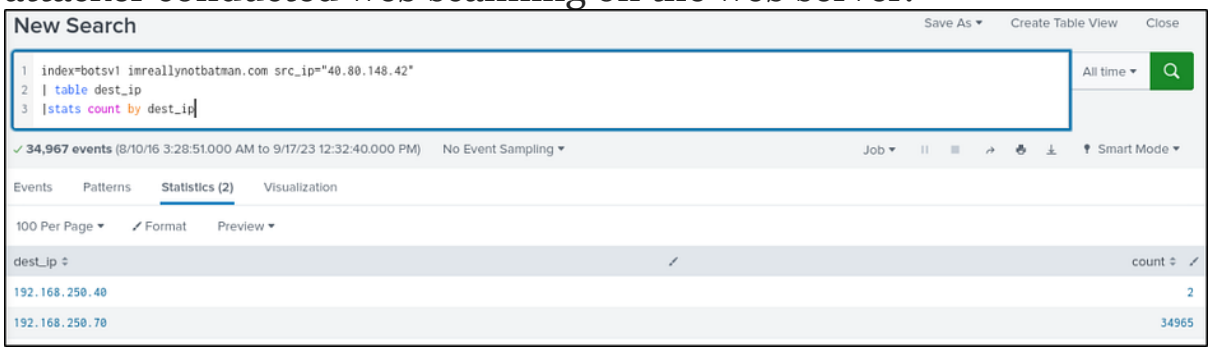
What is the IP address of the server imreallynotbatman.com?

192.168.250.70

The following query would filter only events from the attacker 40.80.148.41 and display the destination IP addresses in table format with the number of times of connection.

```
index=botsv1 imreallynotbatman.com src_ip="40.80.148.42"
| table dest_ip
| stats count by dest_ip
```

Two IP addresses were identified, but 192.168.250.70 is the IP address of “imreallynotbatman.com” because it is known that the attacker conducted web scanning on the web server.



Task 5: Exploitation Phase

Exploitation Phase

The attacker needs to exploit the vulnerability to gain access to the system/server.

In this task, we will look at the potential exploitation attempt from the attacker against our web server and see if the attacker got successful in exploiting or not.

To begin our investigation, let's note the information we have so far:

- We found two IP addresses from the reconnaissance phase with sending requests to our server (`40.80.148.42` and `23.22.63.114`).
- One of the IPs `40.80.148.42` was seen attempting to scan the server with IP **192.168.250.70**.
- The attacker was using the web scanner Acunetix for the scanning attempt.

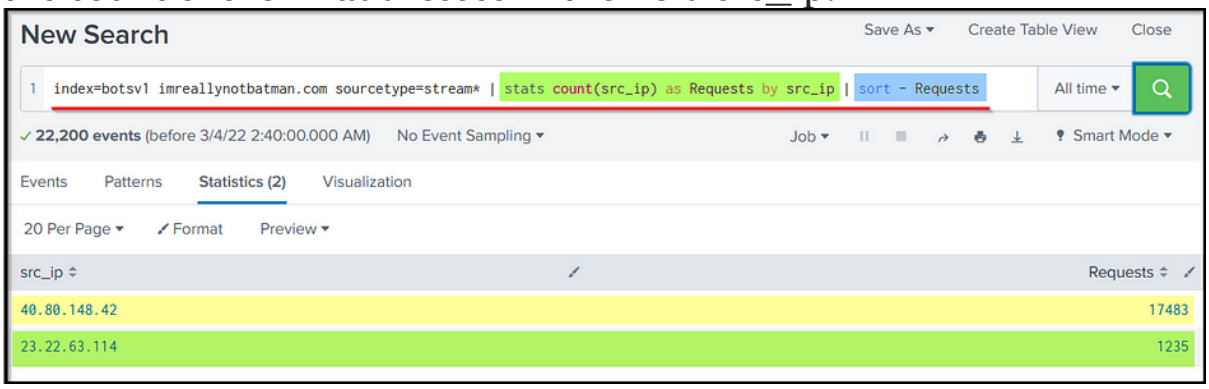
Count

Let's use the following search query to see the number of counts by each source IP against the webserver.

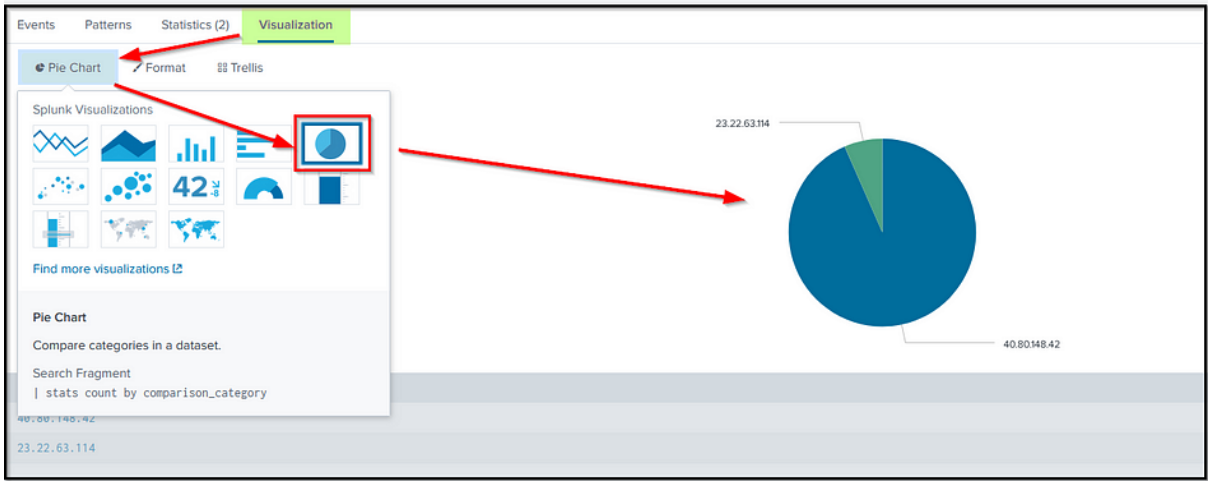
Search Query:

```
index=botsv1 imreallynotbatman.com sourcetype=stream* | stats
count(src_ip) as Requests by src_ip | sort - Requests
```

Query Explanation: This query uses the stats function to display the count of the IP addresses in the field src_ip.



Additionally, we can also create different visualization to show the result. Click on **Visualization → Select Visualization** as shown below.

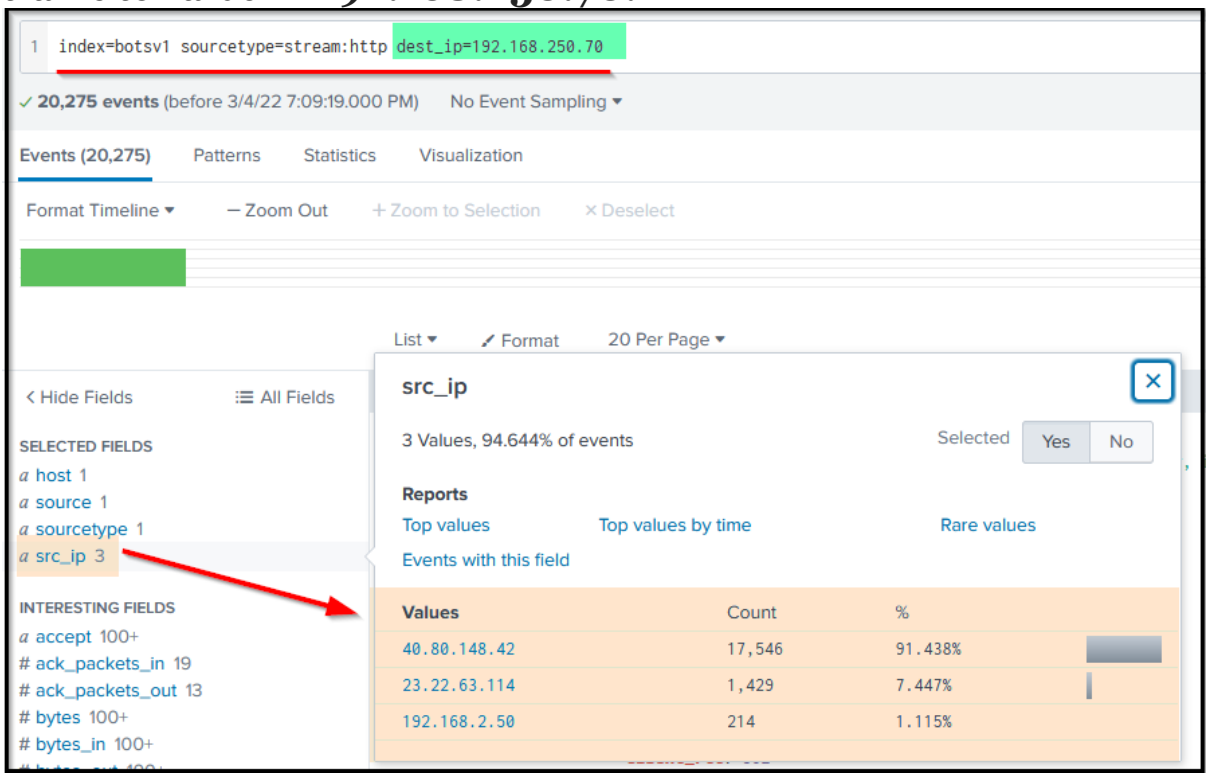


Now we will narrow down the result to show requests sent to our web server, which has the IP `192.168.250.70`

Search Query:

```
index=botsv1 sourcetype=stream:http dest_ip="192.168.250.70"
```

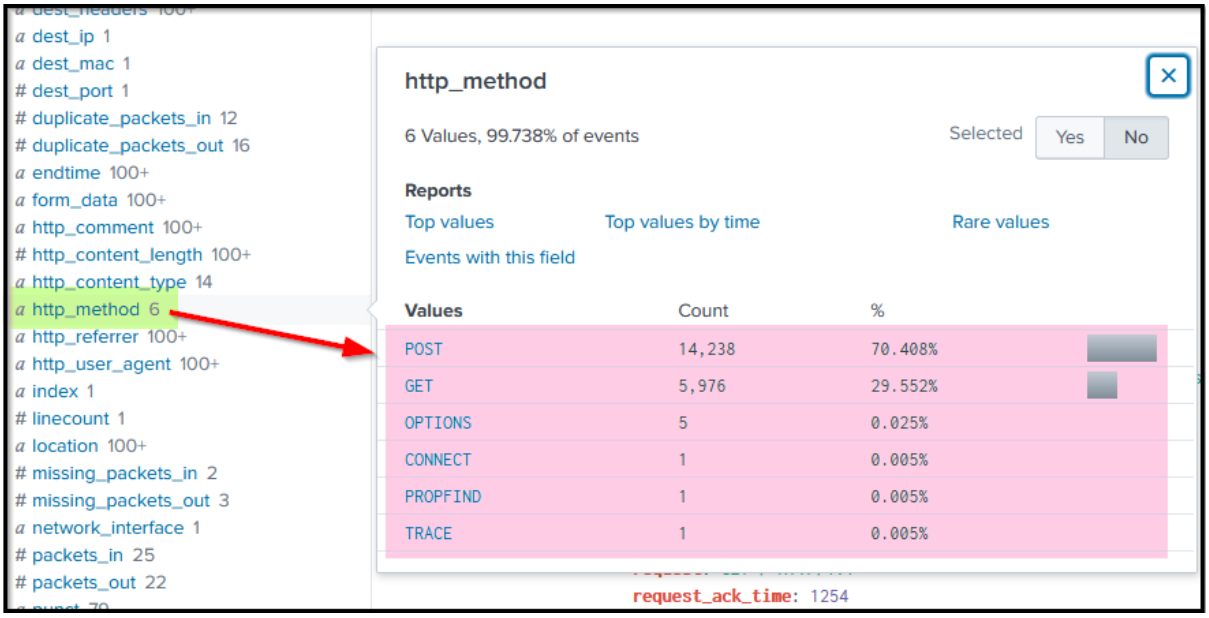
Query Explanation: This query will look for all the inbound traffic towards IP **192.168.250.70.**



The result in the **src_ip** field shows three IP addresses (1 local IP and two remote IPs) that originated the HTTP traffic towards our webserver.

Another interesting field, **http_method** will give us information about the HTTP Methods observed during these HTTP communications.

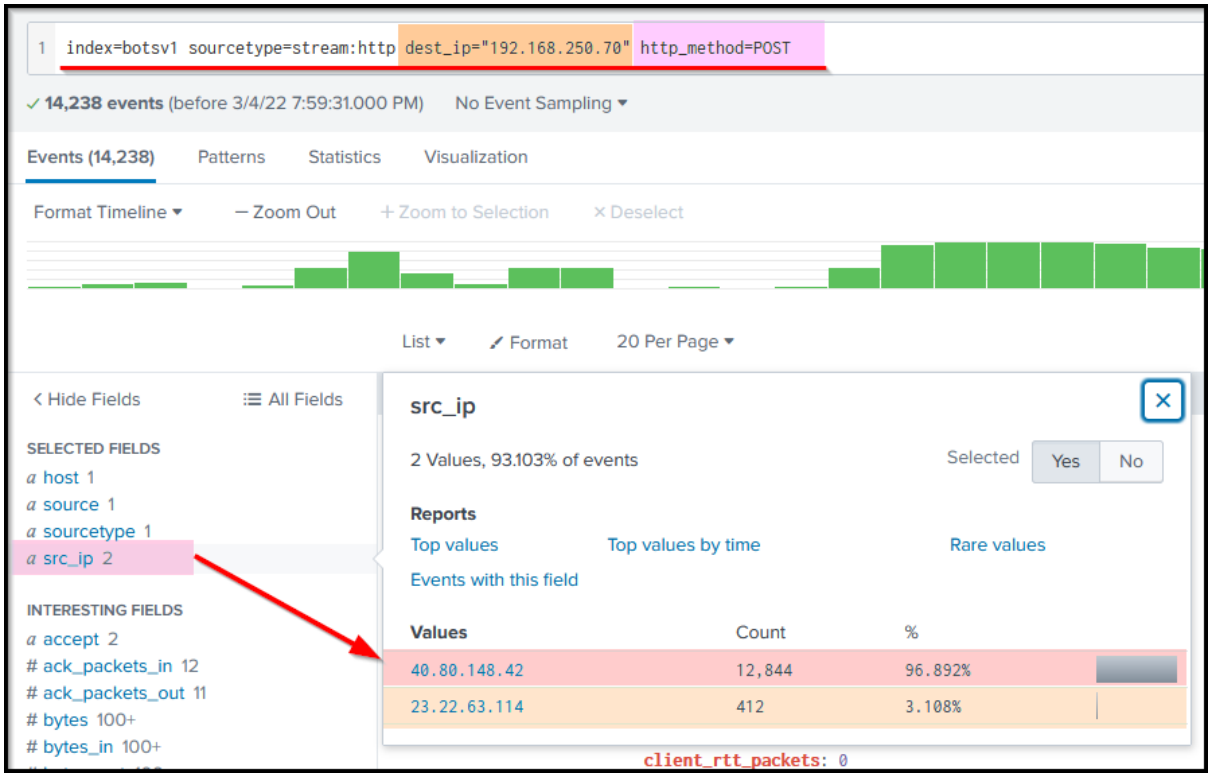
We observed most of the requests coming to our server through the POST request, as shown below.



To see what kind of traffic is coming through the POST requests, we will narrow down on the field `http_method=POST` as shown below:

Search Query:

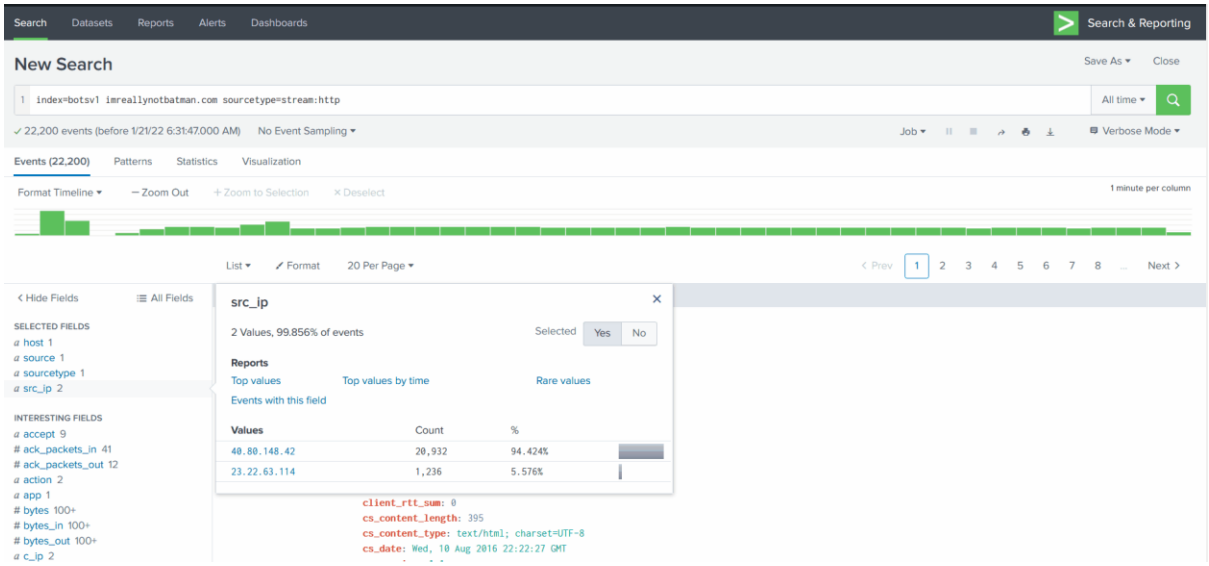
```
index=botsv1 sourcetype=stream:http dest_ip="192.168.250.70"
http_method=POST
```



The result in the **src_ip** field shows two IP addresses sending all the POST requests to our server.

Interesting fields: In the left panel, we can find some interesting fields containing valuable information. Some of the fields are:

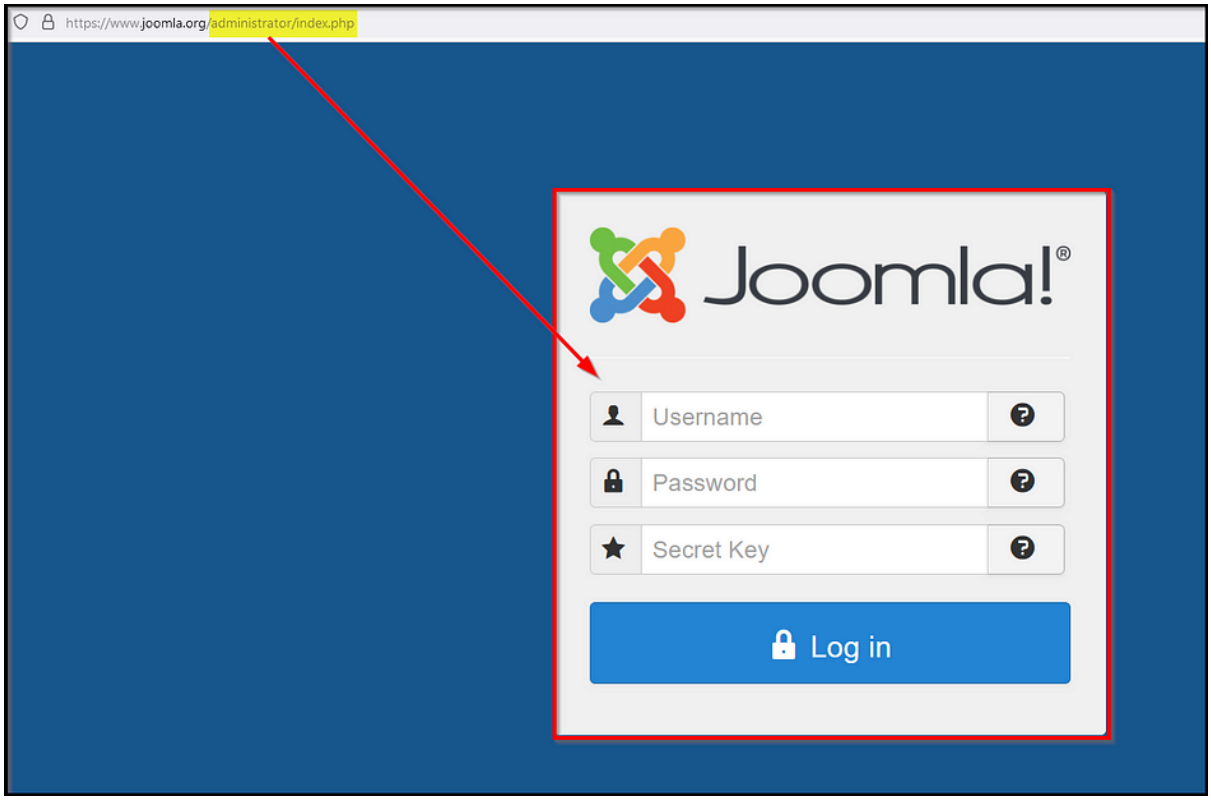
- src_ip
- form_data
- http_user_agent
- uri



The term Joomla is associated with the webserver found in a couple of fields like **uri, uri_path, http_referrer**, etc. This means our webserver is using Joomla CMS (Content Management Service) in the backend.

A little search on the internet for the admin login page of the Joomla CMS will show as -> `/joomla/administrator/index.php`

It is important because this uri contains the login page to access the web portal therefore we will be examining the traffic coming into this admin panel for a potential brute-force attack.



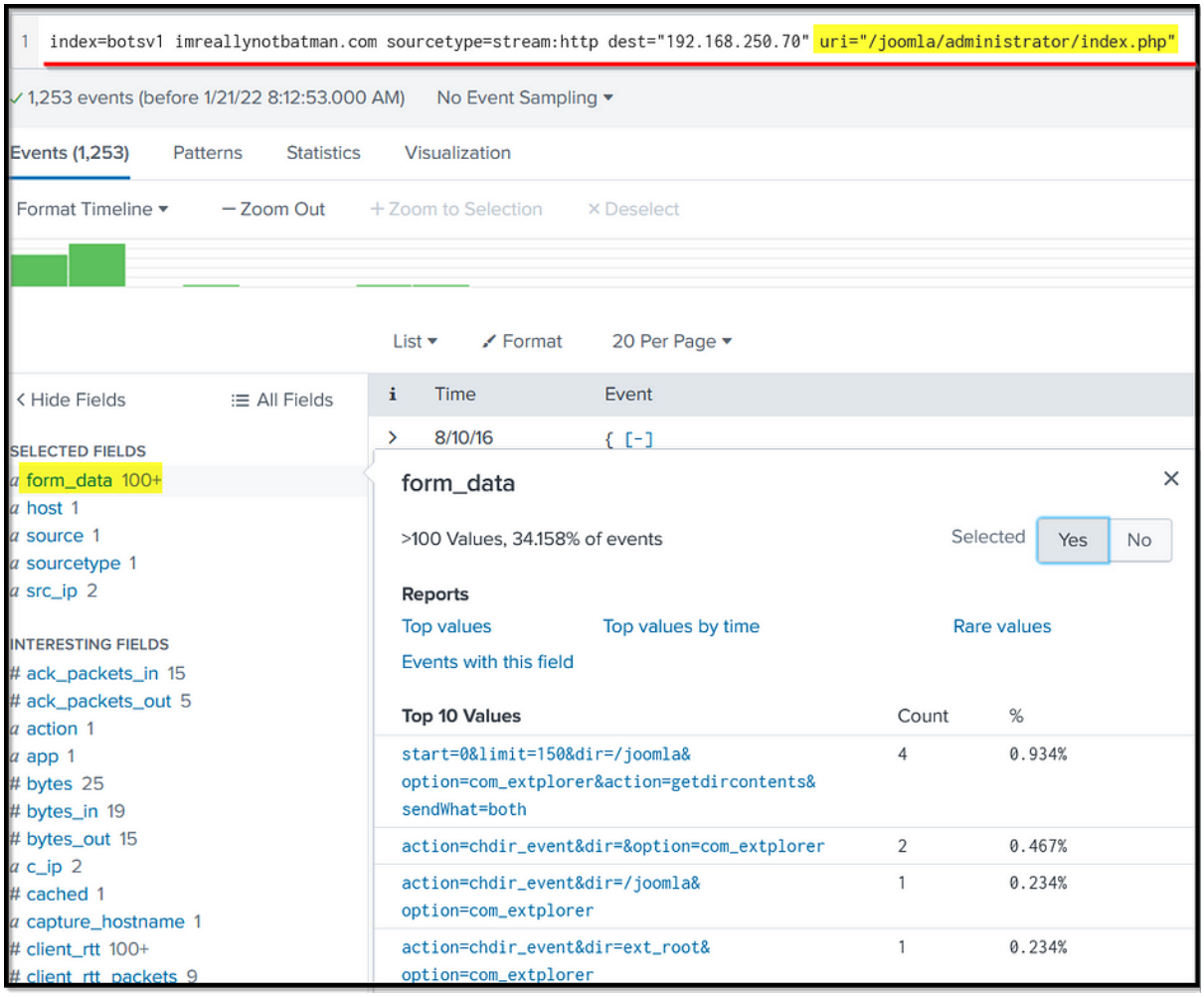
Reference: https://www.joomla.org/administrator/index.php

We can narrow down our search to see the requests sent to the login portal using this information.

Search query:

```
index=botsv1 imreallynotbatman.com sourcetype=stream:http
dest_ip="192.168.250.70"  uri="/joomla/administrator/index.php"
```

Query Explanation: We are going to add `uri="/joomla/administrator/index.php"` in the search query to show the traffic coming into this URI.

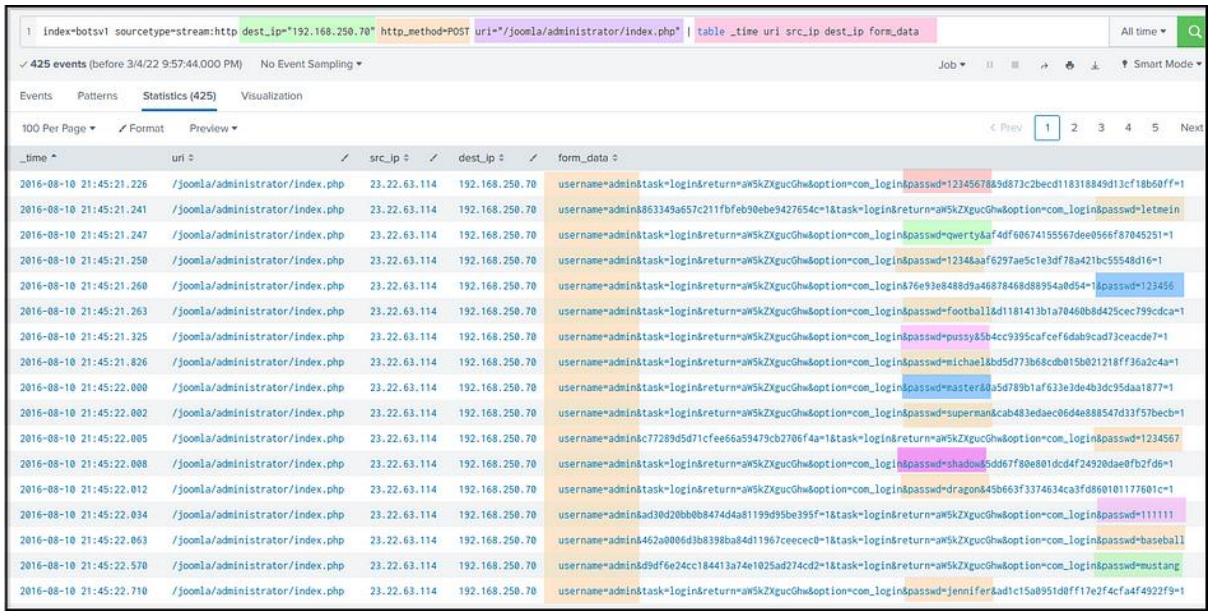


`form_data` The field contains the requests sent through the form on the admin panel page, which has a login page. We suspect the attacker may have tried multiple credentials in an attempt to gain access to the admin panel. To confirm, we will dig deep into the values contained within the form_data field, as shown below:

Search Query:

```
index=botsv1 sourcetype=stream:http dest_ip="192.168.250.70"
http_method=POST uri="/joomla/administrator/index.php" | table _time uri
src_ip dest_ip form_data
```

Query Explanation: We will add this -> `| table _time uri src dest_ip form_data` to create a table containing important fields as shown below:



If we keep looking at the results, we will find two interesting fields `username` that includes the single username `admin` in all the events and another field `passwd` that contains multiple passwords in it, which shows the attacker from the IP `23.22.63.114` Was trying to guess the password by brute-forcing and attempting numerous passwords.

The time elapsed between multiple events also suggests that the attacker was using an automated tool as various attempts were observed in a short time.

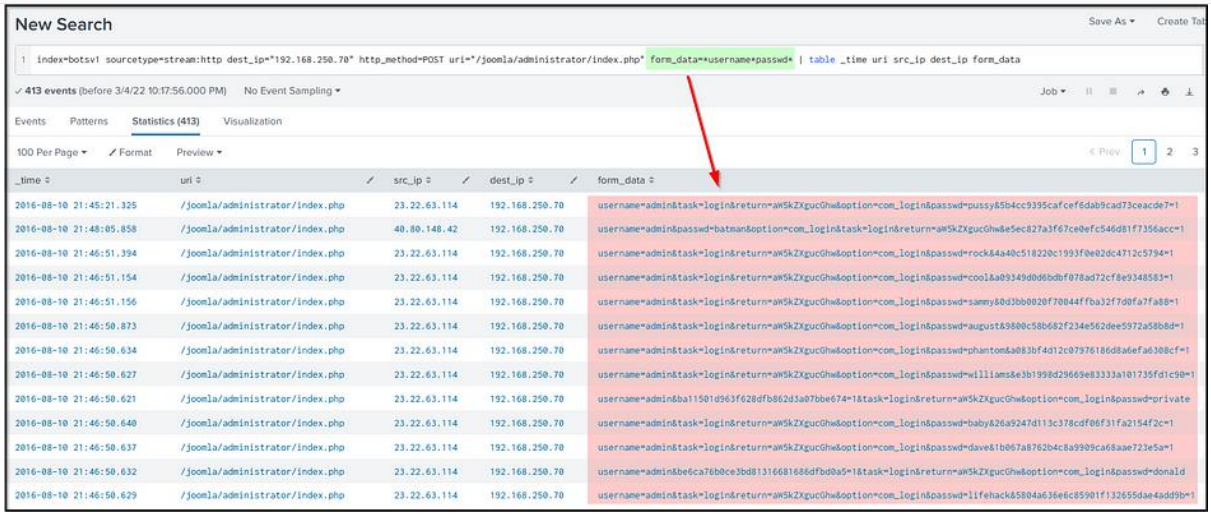
Extracting Username and Passwd Fields using Regex

Looking into the logs, we see that these fields are not parsed properly. Let us use **Regex** in the search to extract only these two fields and their values from the logs and display them.

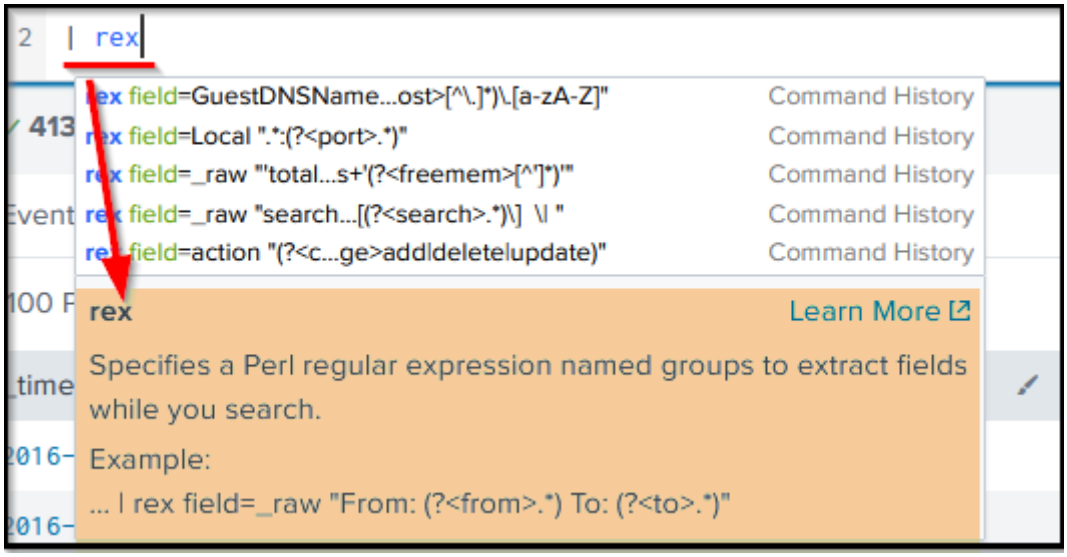
We can display only the logs that contain the **username** and **passwd** values in the form_data field by adding `form_data=*username*passwd*` in the above search.

Search Query:

```
index=botsv1 sourcetype=stream:http dest_ip="192.168.250.70"
http_method=POST uri="/joomla/administrator/index.php"
form_data=*username*passwd* | table _time uri src_ip dest_ip form_data
```



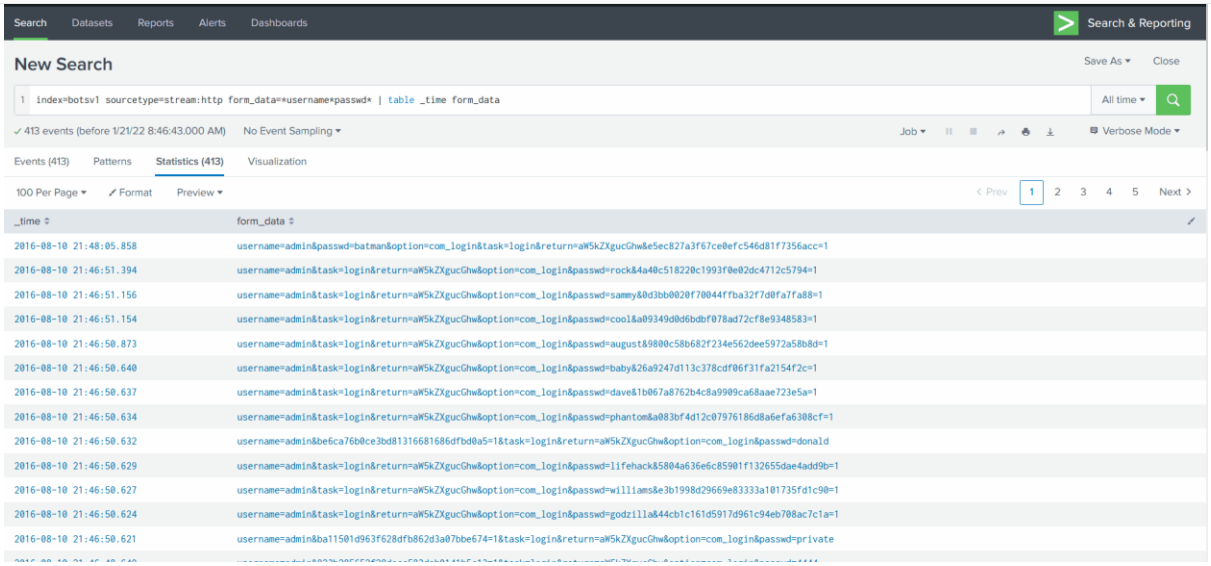
It's time to use Regex **(regular expressions)** to extract all the password values found against the field passwd in the logs. To do so, Splunk has a function called rex. If we type it in the search head, it will show detail and an example of how to use it to extract the values.



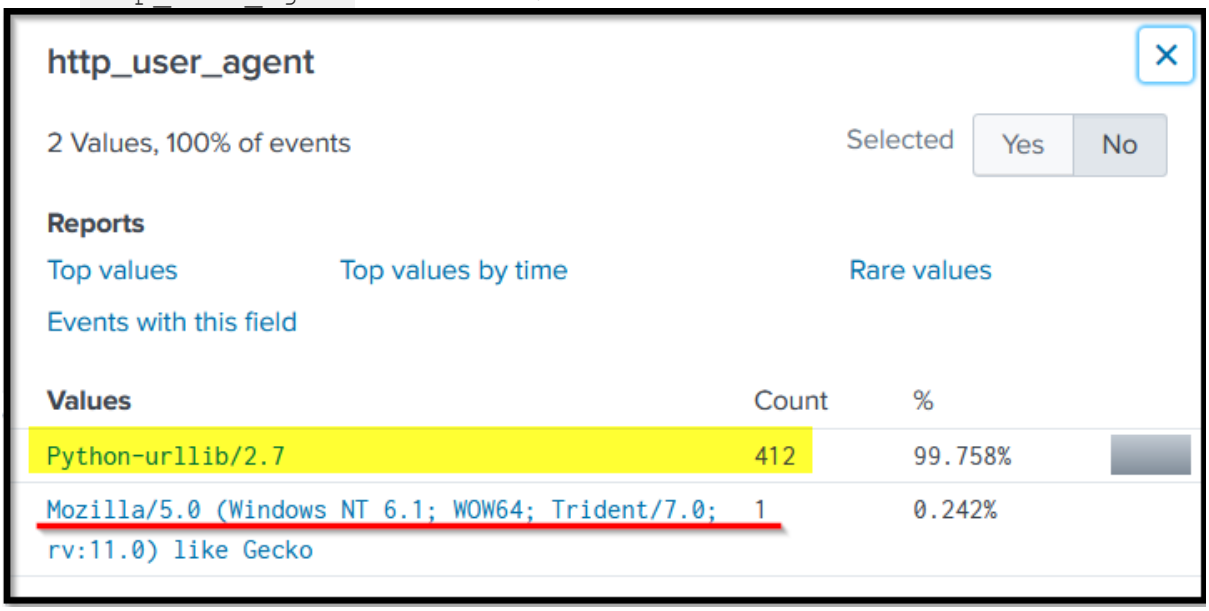
Now, let's use Regex. **rex field=form_data "passwd=(?<creds>\w+)"** To extract the **passwd** values only. This will pick the **form_data** field and extract all the values found with the field. **creds**.

Search Query:

```
index=botsv1 sourcetype=stream:http dest_ip="192.168.250.70"
http_method=POST form_data=*username*passwd*
|rex field=form_data "passwd=(?<creds>\w+)"
|table src_ip creds
```



We have extracted the passwords being used against the username admin on the admin panel of the webserver. If we examine the fields in the logs, we will find two values against the field`http_user_agent` as shown below:

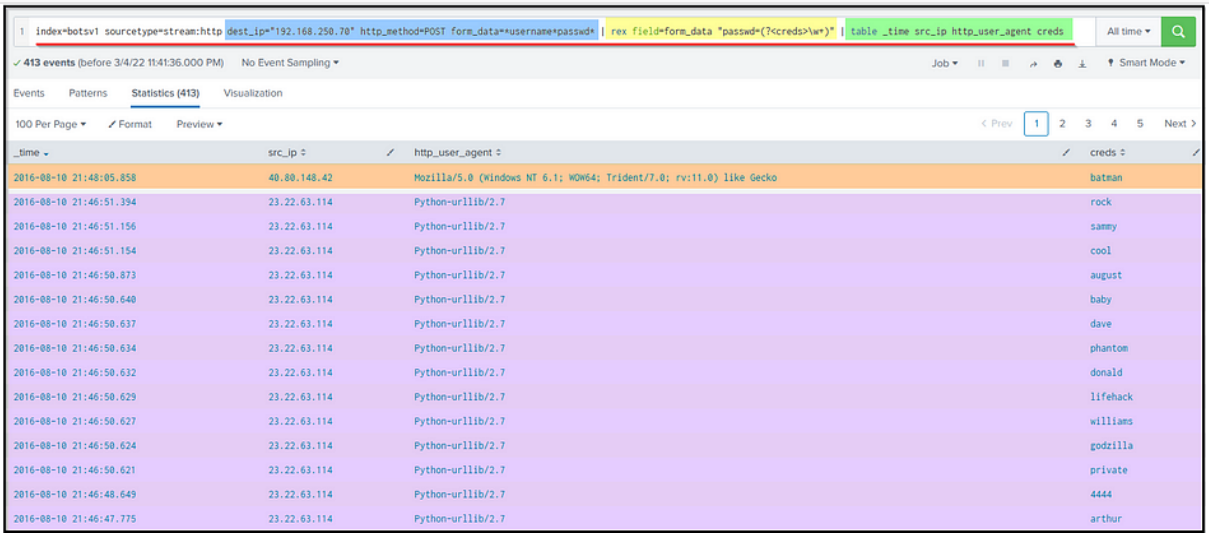


The first value clearly shows attacker used a python script to automate the brute force attack against our server. But one request came from a Mozilla browser. WHY? To find the answer to this query, let's slightly change to the about search query and add `http_user_agent` a field in the search head.

Let's create a table to display key fields and values by appending -> `| table _time src_ip uri http_user_agent creds` in the search query as shown below.

Search Query:

```
index=botsv1 sourcetype=stream:http dest_ip="192.168.250.70"
http_method=POST form_data=*username*passwd* | rex field=form_data
"passwd=(?<creds>\w+)" |table _time src_ip uri http_user_agent creds
```



This result clearly shows a continuous brute-force attack attempt from an IP **23.22.63.114** and 1 password attempt **batman** from IP **40.80.148.42** using the Mozilla browser.

Answer the questions below

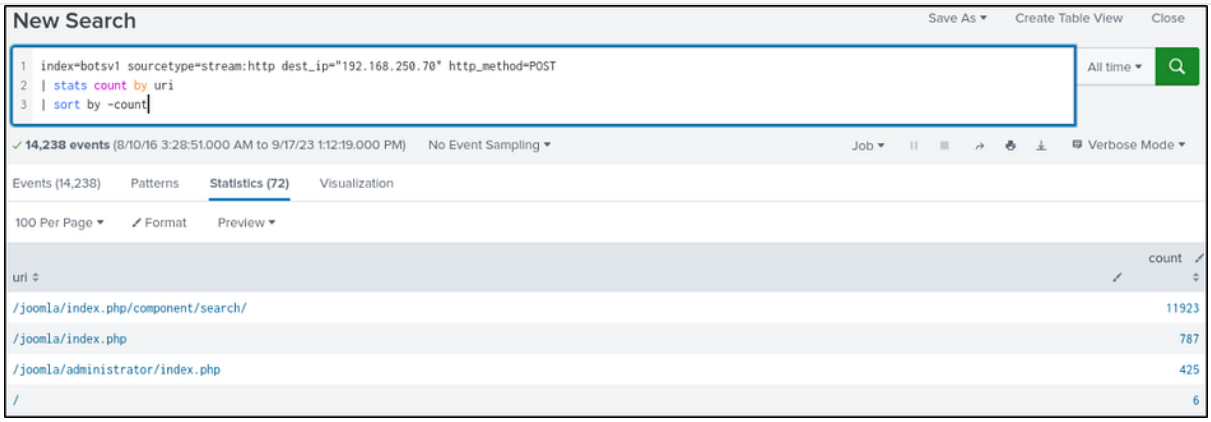
What was the URI which got multiple brute force attempts?

Answer: /joomla/administrator/index.php

From the queries in this task, It was determined that "/joomla/administrator/index.php" was being brute-forced as it is the administrator login page of the CMS.

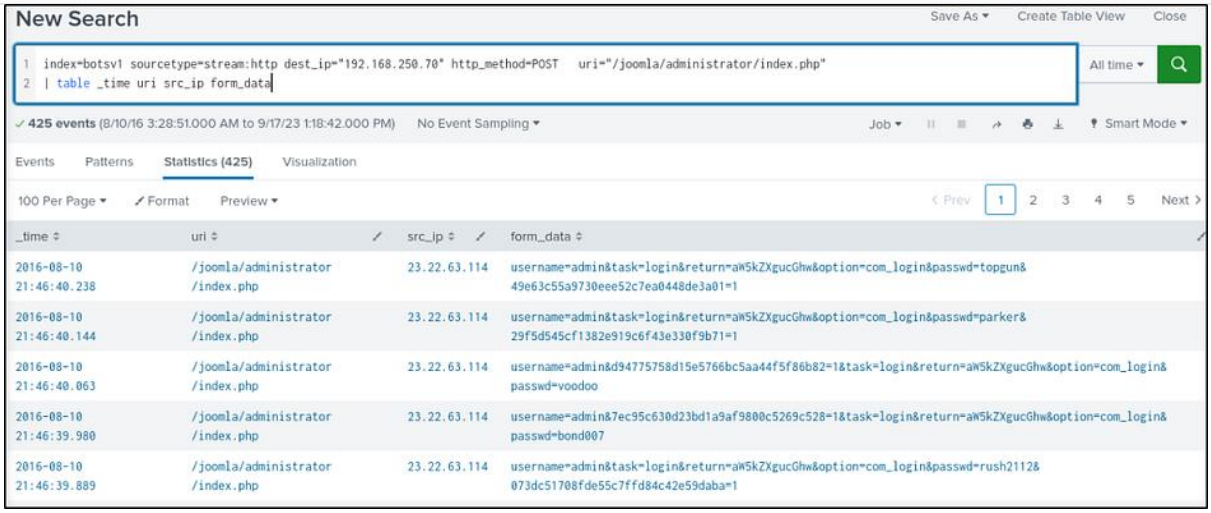
If assuming that CMS is not known, the following query will provide the numbers of the visited URI of the web server.

```
index=botsv1 sourcetype=stream:http dest_ip="192.168.250.70"
http_method=POST
| stats count by uri
| sort by -count
```



From there analyse the top most URI's. Upon further analysis the URI "/joomla/administrator/index.php" is found to be the under brute-force attack

```
index=botsv1 sourcetype=stream:http dest_ip="192.168.250.70"
http_method=POST   uri="/joomla/administrator/index.php"
| table _time uri src_ip form_data
```



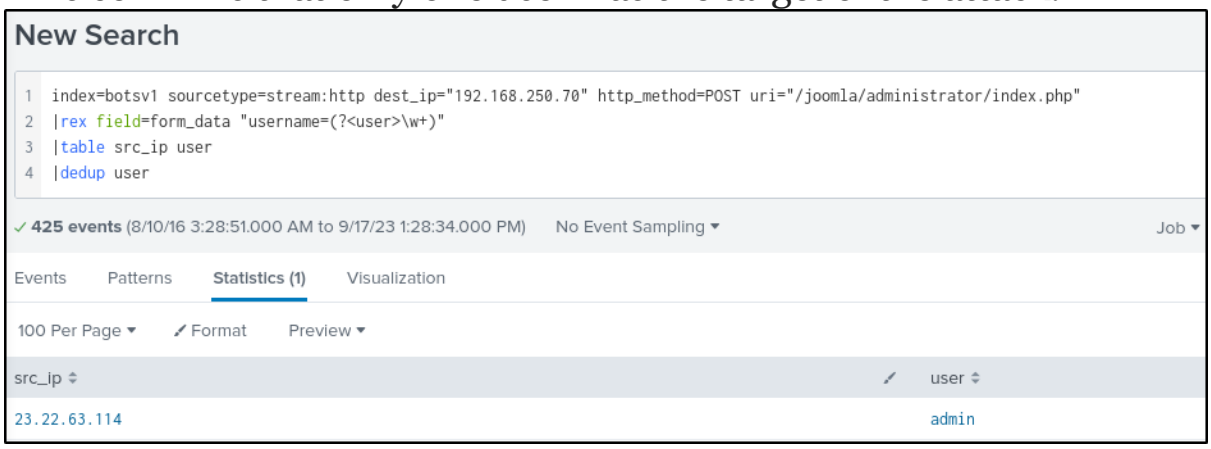
Against which username was the brute force attempt made?

Answer: admin

From the task, it was determined that the username being targeted is “admin”. The following query will identify all possible targeted usernames in the attack assuming there could be other usernames being targeted. Instead of password, the “rex” command will extract the username values in the “form_data” and “dedup” will remove any duplicate values.

```
index=botsv1 sourcetype=stream:http dest_ip="192.168.250.70"
http_method=POST uri="/joomla/administrator/index.php"
|rex field=form_data "username=(?<user>\w+)"
|table src_ip user
|dedup user
```

This confirms that only one user was the target of the attack.



What was the correct password for admin access to the content management system running imreallynotbatman.com?

Answer: batman

It was identified in the task that “batman” is the correct password because the attacker from 40.80.148.42 used a Mozilla browser to connect. Another way to determine is to count the number of times the password was used. If it was used more than once, then it can be assumed that it is the correct password.

```
index=botsv1 sourcetype=stream:http dest_ip="192.168.250.70"
http_method=POST uri=/joomla/Administrator/index.php
| rex field=form_data
"username=(?<username>admin).*passwd=(?<password>\w+)"
|stats count by password
| sort -count
```

“batman” was indeed used twice.

I’ll dig further to determine if it was successfully used as a credential to log in.

```
index=botsv1 sourcetype=stream:http dest_ip="192.168.250.70"
http_method=POST uri=/joomla/Administrator/index.php status=200
|rex field=form_data
"username=(?<username>admin).*passwd=(?<password>\batman)"
|stats count by src_ip
```

Here, it is determined that the attacker was able to log in 12 times using the credentials, as indicated by the HTTP status of 200.

How many unique passwords were attempted in the brute force attempt?

Answer: 412

"40.80.148.42" was identified as having logged in using the credentials, but it was not a participant in the brute-force attempt. The query will count all attacks from "`23.22.63.114`" and remove all duplicate passwords. Afterward, it will calculate the number of unique passwords that were used.

```
index=botsv1 sourcetype=stream:http src_ip=23.22.63.114
dest_ip="192.168.250.70" http_method=POST
uri_path="/joomla/administrator/index.php"
| rex field=form_data "passwd=(?<password>\w+)"
| dedup password
| sort count
```

What IP address is likely attempting a brute force password attack against imreallynotbatman.com?

Answer: 23.22.63.114

After finding the correct password, which IP did the attacker use to log in to the admin panel?

Answer: 40.80.148.42