**Cybersecurity 401**

**Module 3 - Security Operations**

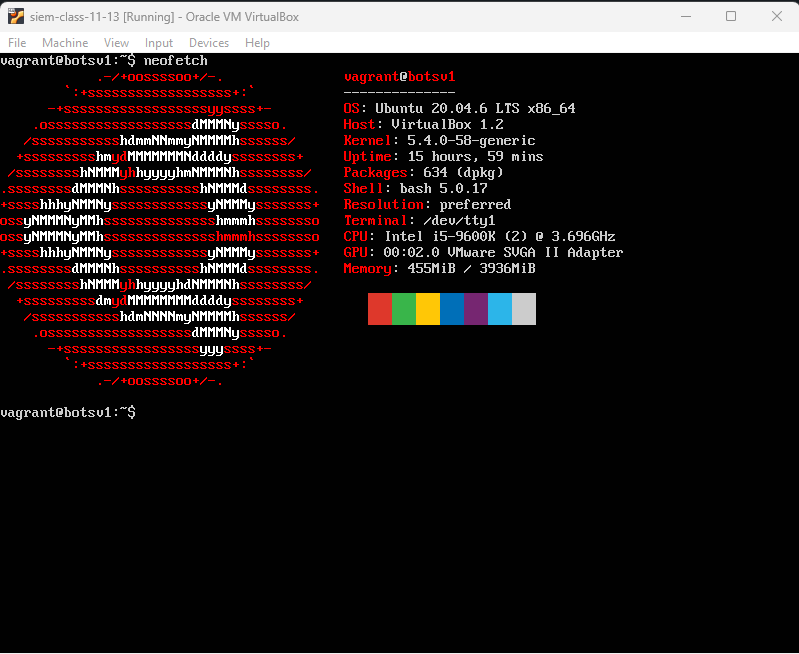
**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

# **Lab 12 - Log Analysis with Splunk**

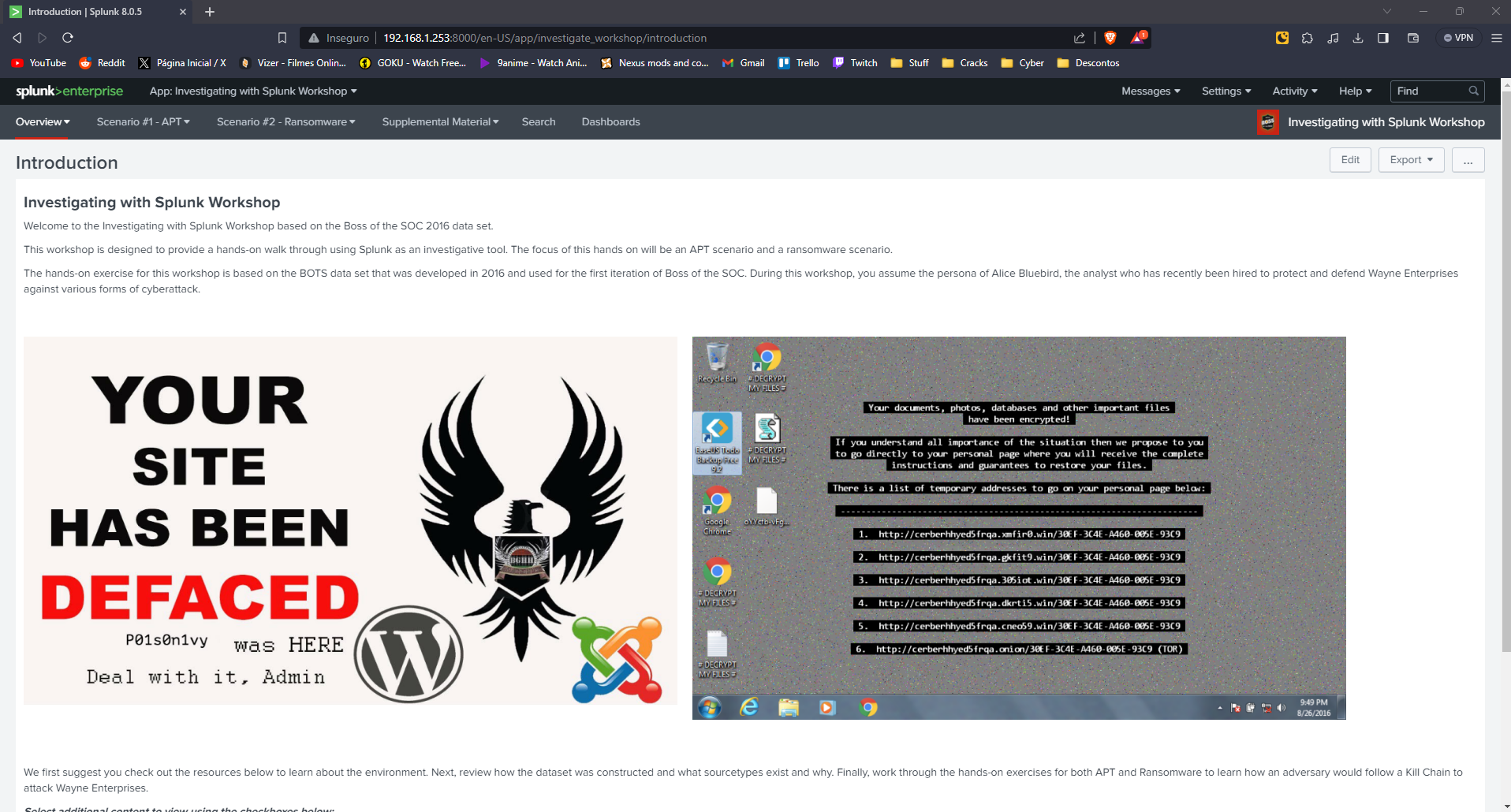
# 

**| Rodrigo Brasil 11/2023 |**

## Part 1: Staging

****

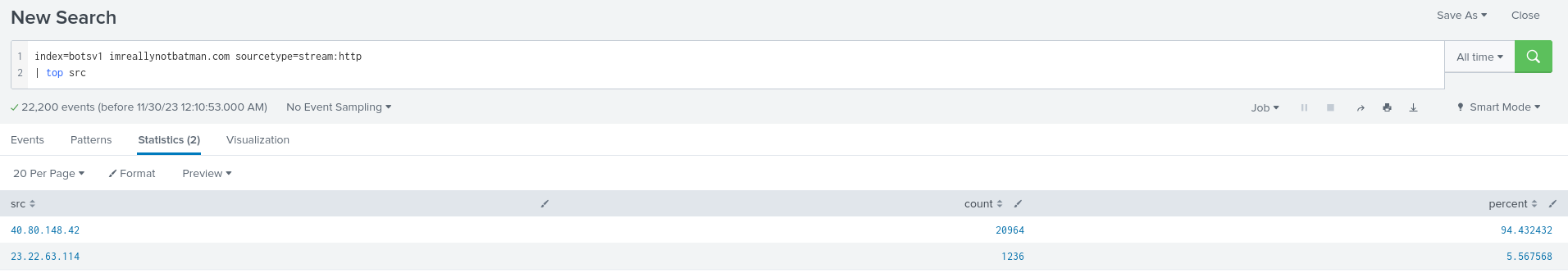
Ubuntu vm running splunk with the IP 192.168.1.253



Splunk workshop running on my host machine

## Part 2: BOTSv1

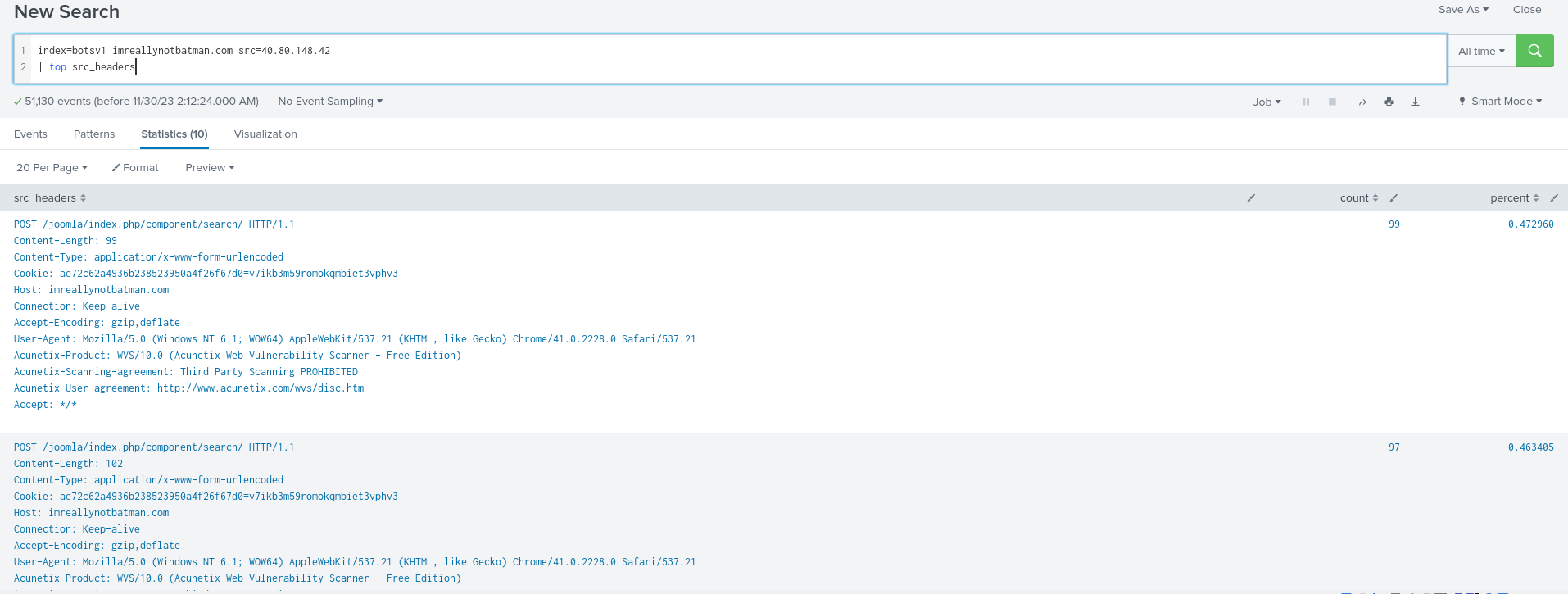
### Reconnaissance



Using the search above i was able to find the IP that accessed the most



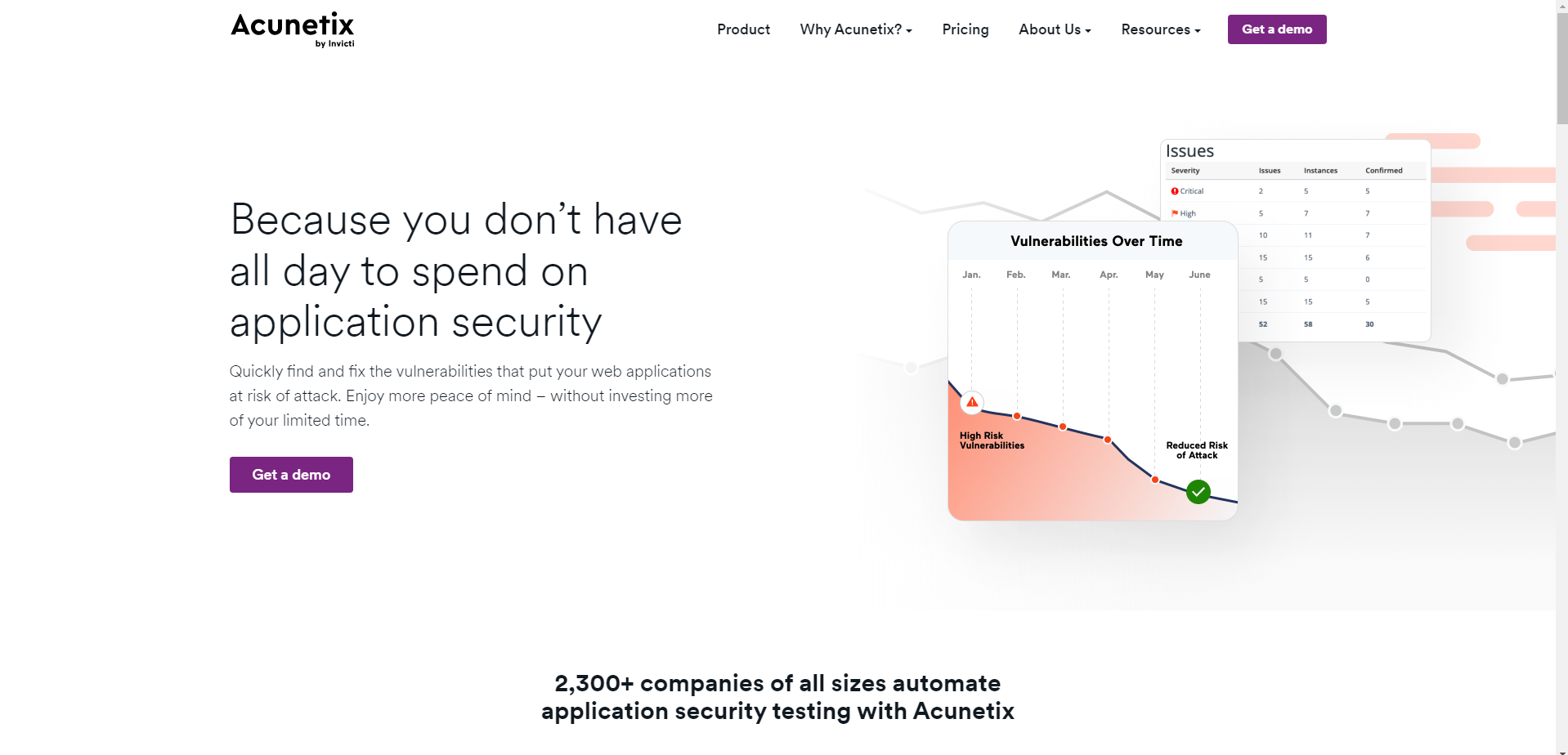
Using suricata and searching signatures we can find that there was a cross-site scripting attack





searching for src\_headers we can see “Acunetix web vulnerability scanner - free edition”

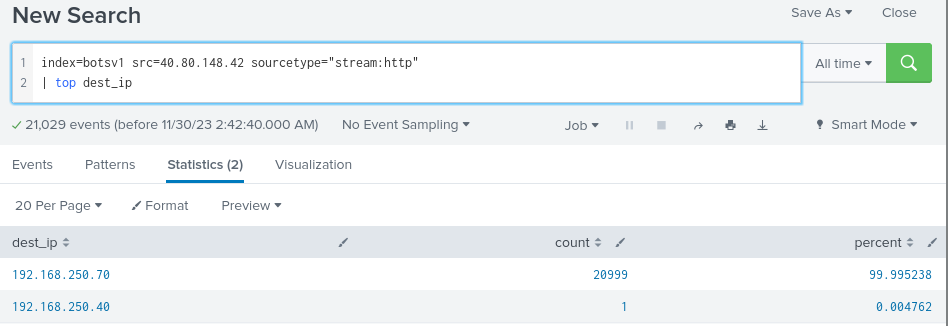
seems like the scan was blocked because third party scanning is prohibited



Searching for the name i was able to find the scanner website

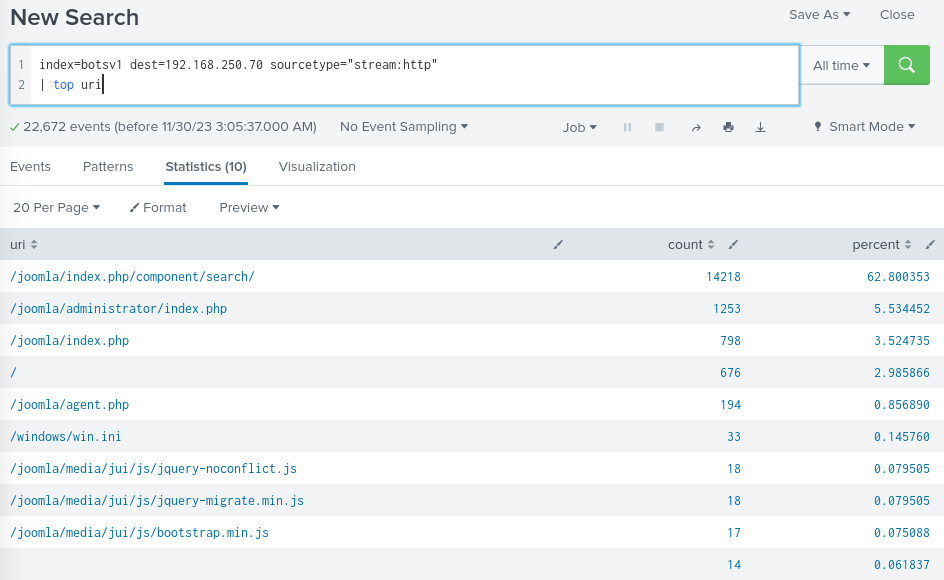
<https://www.acunetix.com/>

which is a legitimate tool



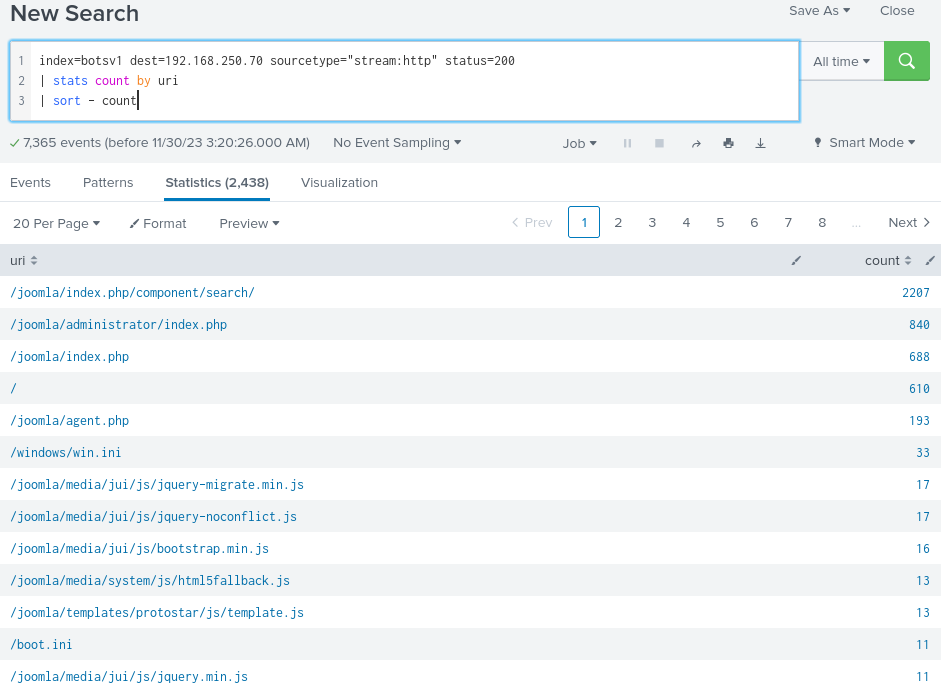
To find out what CMS imreallynotbatman.com is using

we must find its IP which from this search its “192.168.250.70”



now that we know the destination ip and the source type, we should start searching url’s,

joomla show up alot, didn't know what it is so searching on google it shows that joomla is a open source platform to create websites and applications and it is a CMS

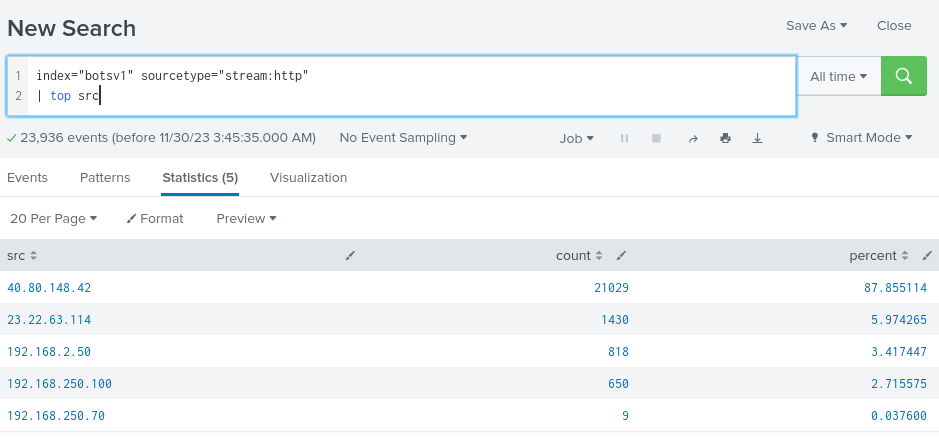


with the search above it will show the amount of uris that have a successful status 200 and sort is count

so the server with the ip 192.168.250.70 running joomla was the targeted server

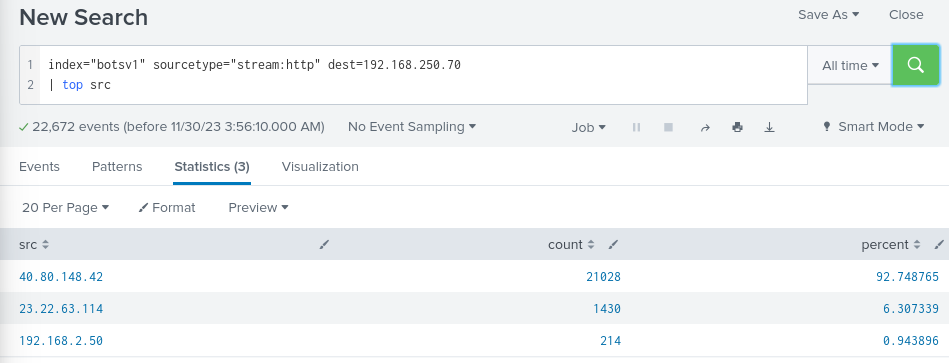
### Exploitation

someone has been attempting a brute force attack against the domain imnotreallybatman.com



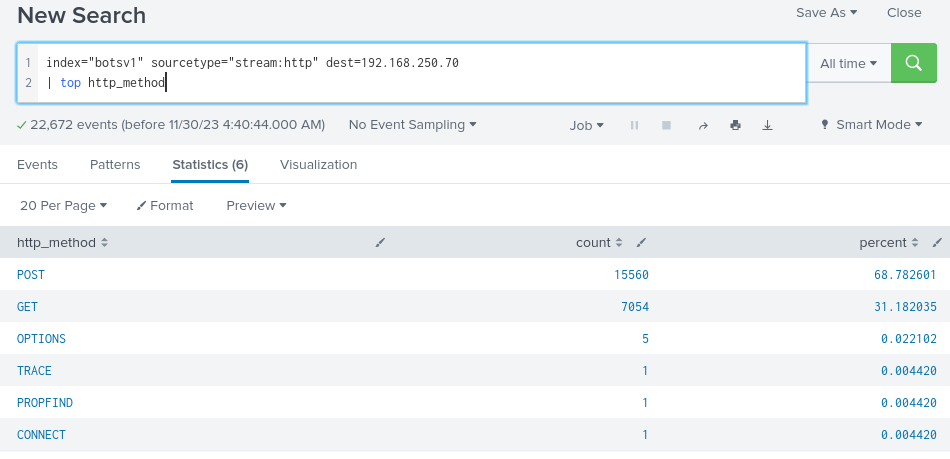
To find out who i started by searching the terms sourcetype.stream:http to look at http traffic and sorting it by source to look at ips

from the sources ips shown i recognized the first and the last IP



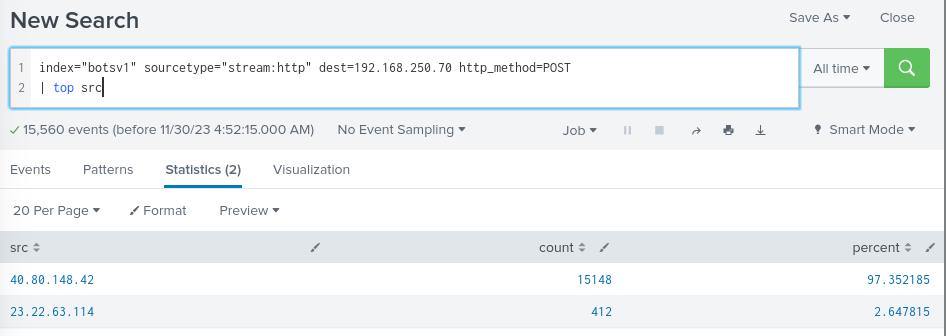
adding the domain IP as the destination IP to the search

we can see what ips have been passing by it

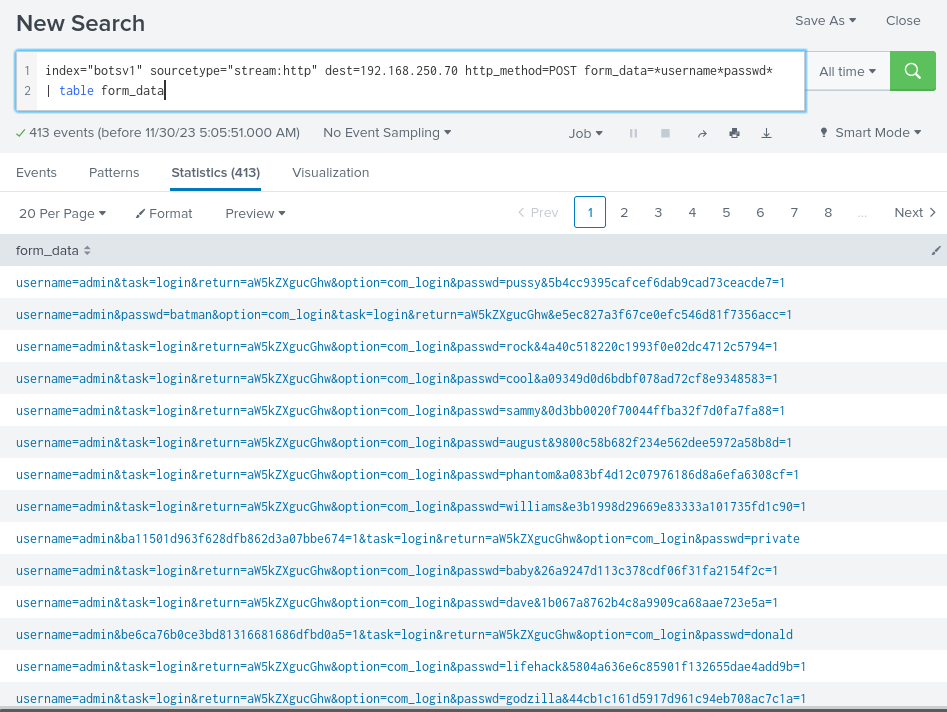


Looking at the method there is an unusual high amount of POST requests compared to GET

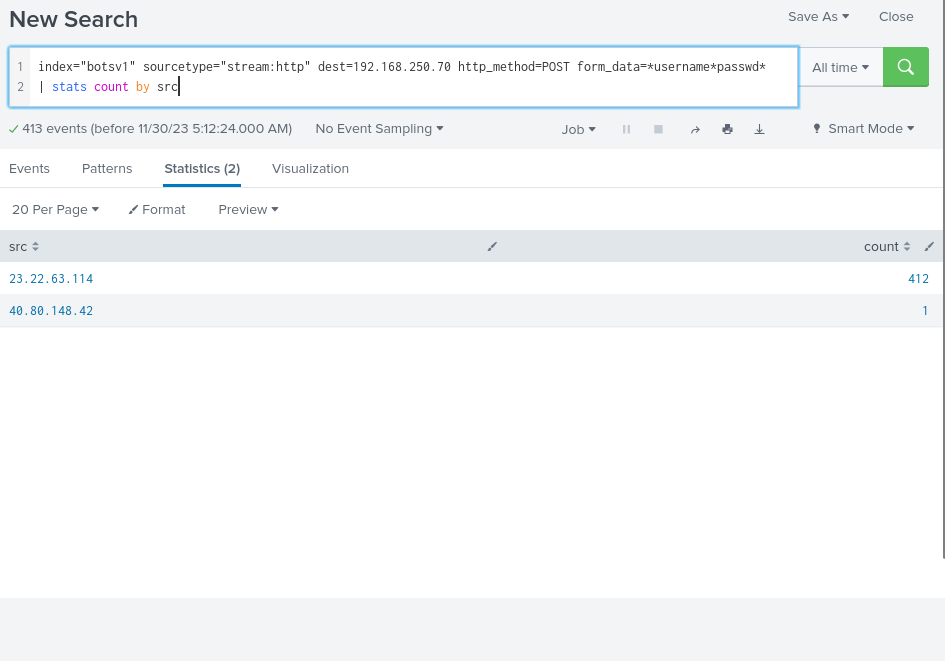
so we can determine that the brute force attack is using a POST method



using the POST method to search we can see only 2 source ip addresses with the majority coming from 40.80.148.42 but I'm not sure if the brute force attack is coming from that IP



Using the table form\_data searching for usernames and passwords

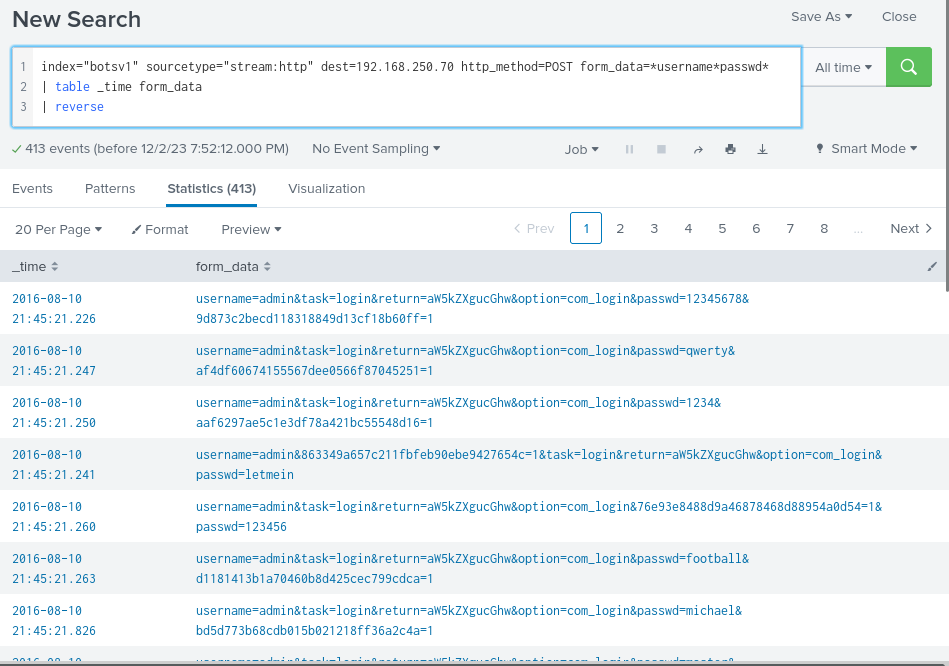


i can use it to count the logins done counting them by source and it shows that actually the 23.22.63.114 has a lot of counts compared to 40.80.148.42

so we could safely assume the brute force attack came from 23.22.63.114

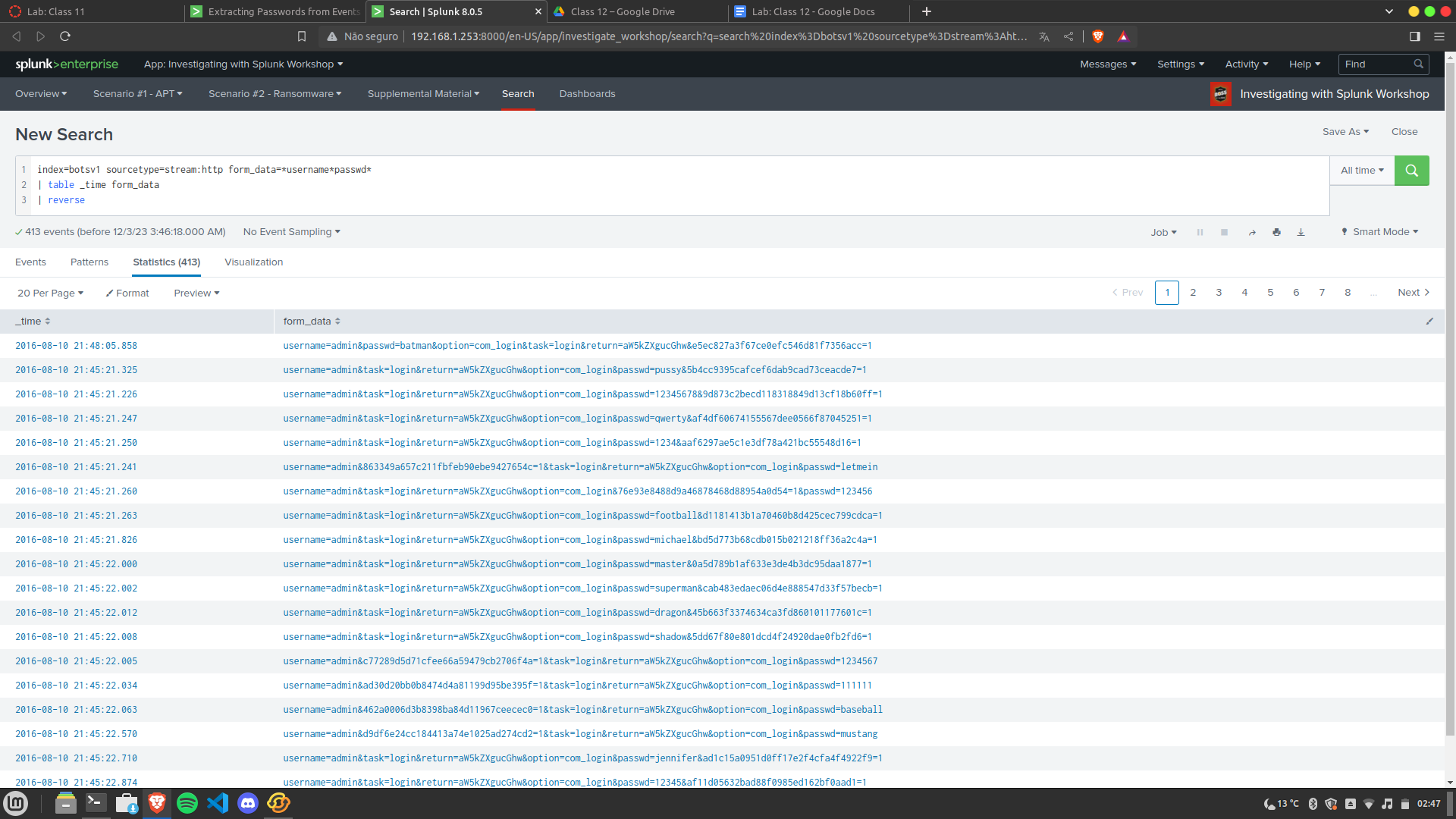
### 

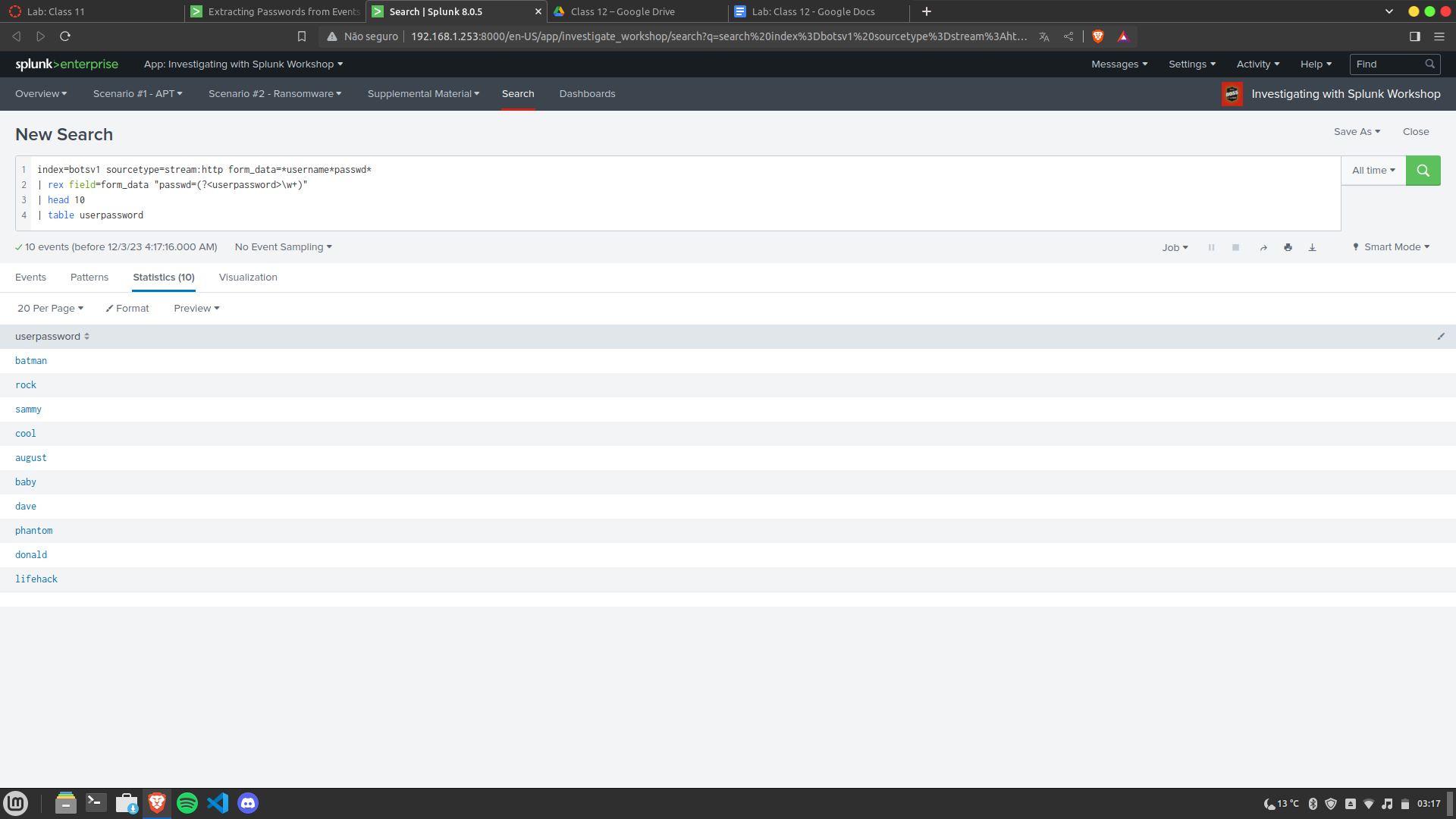
by adding table \_time form\_data to the search it will show the date and time for the form\_data given also in the search which is the name and password

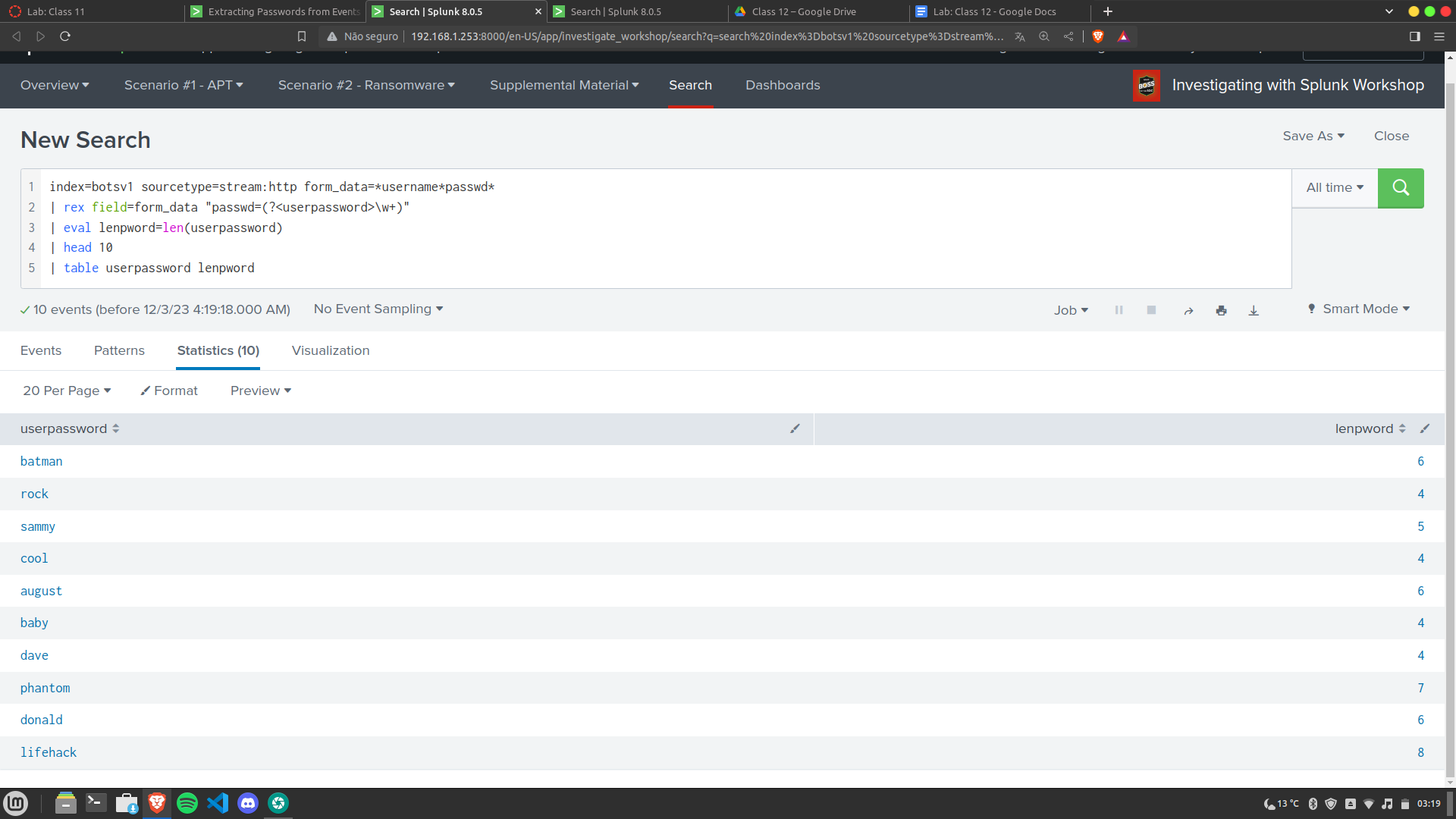


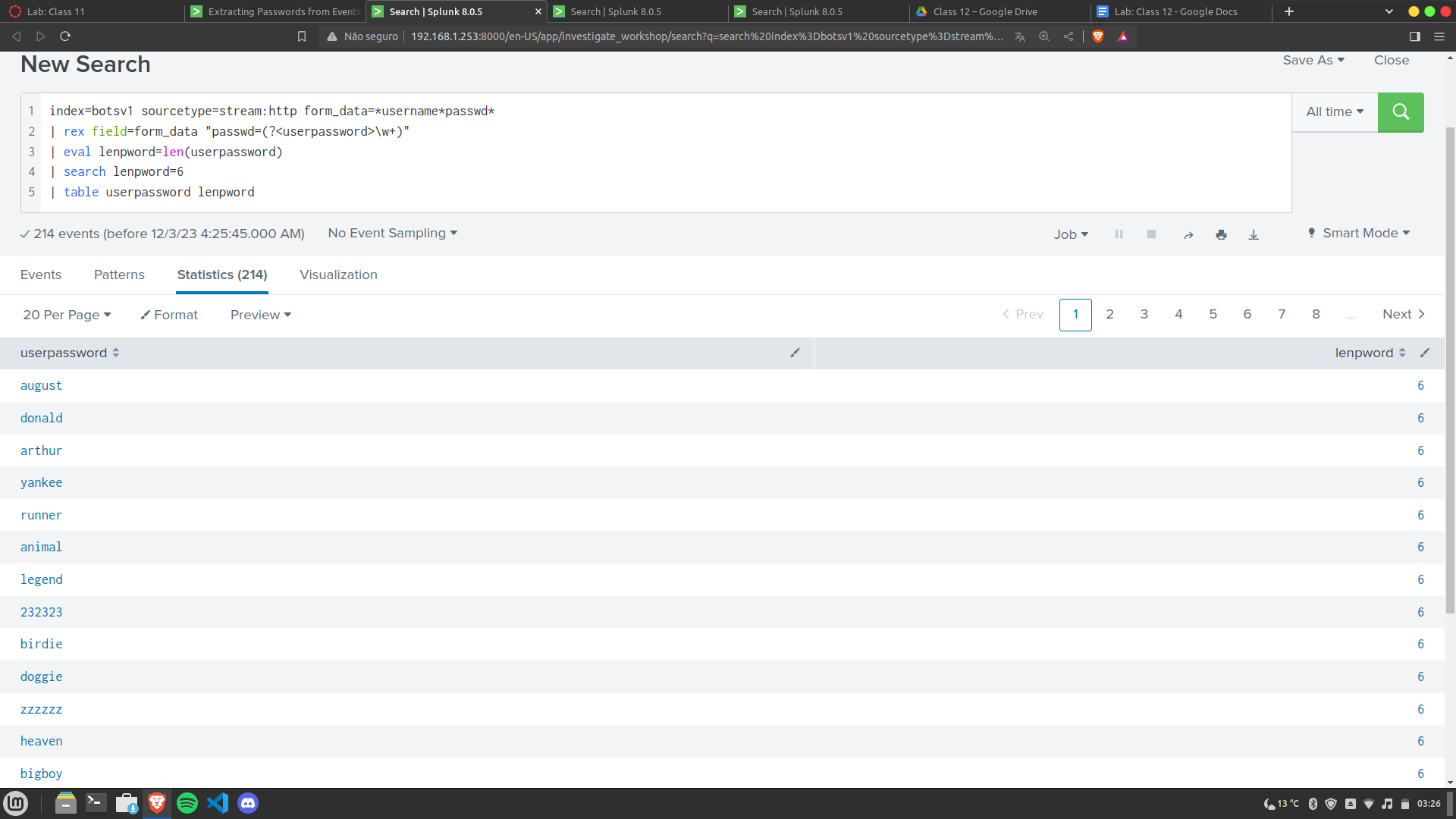
Adding the | reverse to the search bar it shows from the oldest to the newest results

### 

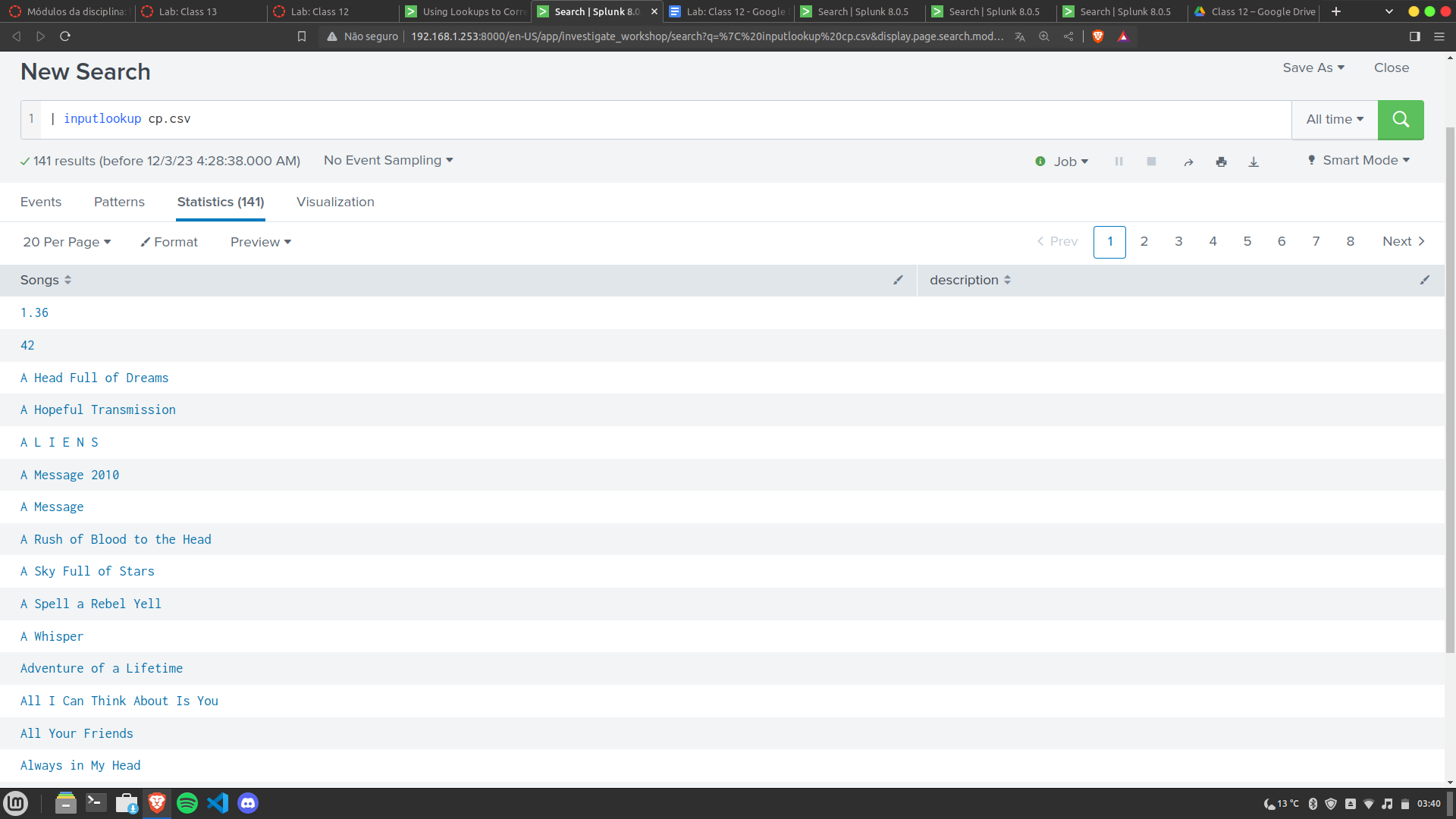


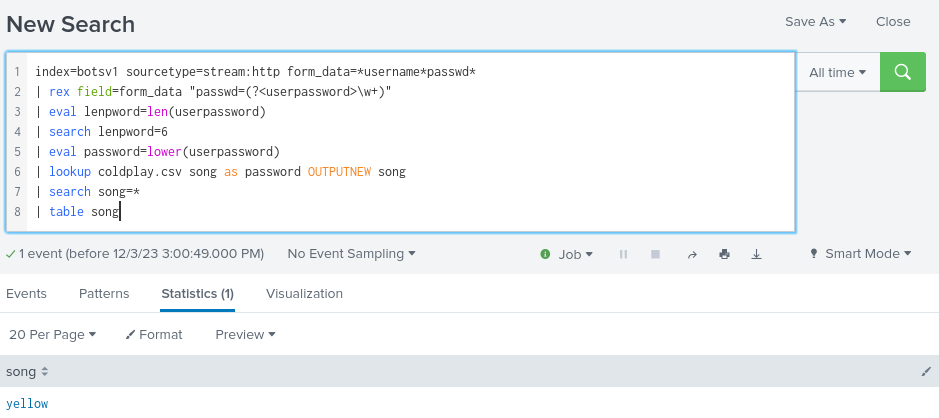






#### **Using Lookups to Correlate Events (Part 2 of 2)**





Here is our initial search that returns all passwords with a length of 6 from our events.

In lookups, case matters, so we will convert those extracted passwords to lower case using the eval command and lower function.

The lookup command compares the lookup value, in this case song from the coldplay.csv file, to the password value from the events. If we get a hit, output the song.

Search for any of the results that have a song value

Return the values for the field song in a table

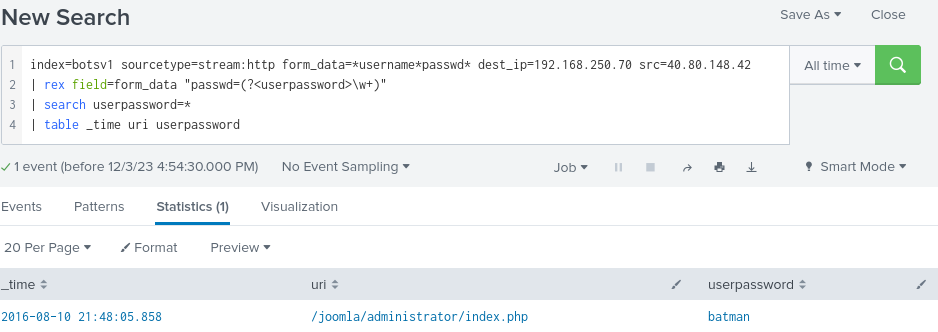
#### **Identifying the Password Used To Gain Access**

#### 

Once that extraction is in place, we want to see userpassword values that are used more than once. It would stand to reason that if we are brute forcing a web site, we are going to run through a password list but we are only going to retry a password that we get a positive hit on. By sorting on count, we see that the admin password of batman was used twice so we can deduce that this is the password for the admin account.

#### 

By adding the values function to our stats command, we can see that the brute force attack came from 23.22.63.114 but it appears that the actual penetration with the correct password came from 40.80.148.42. That is good information to add to our notes as well because we saw other activity from that 40.80.148.42 address previously.

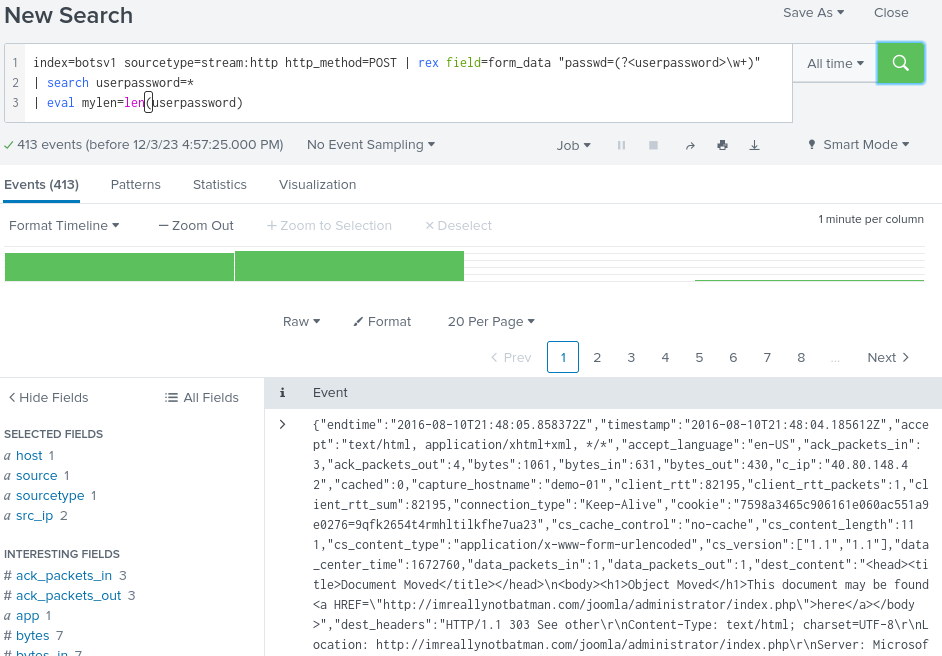


The next logical step is to get an idea of the time and the URI that is associated with the actual penetration. To do this, we add the source address that we identified to our search.

#### **Finding the Average Length of the Passwords During the Brute Force Attack**

#### 

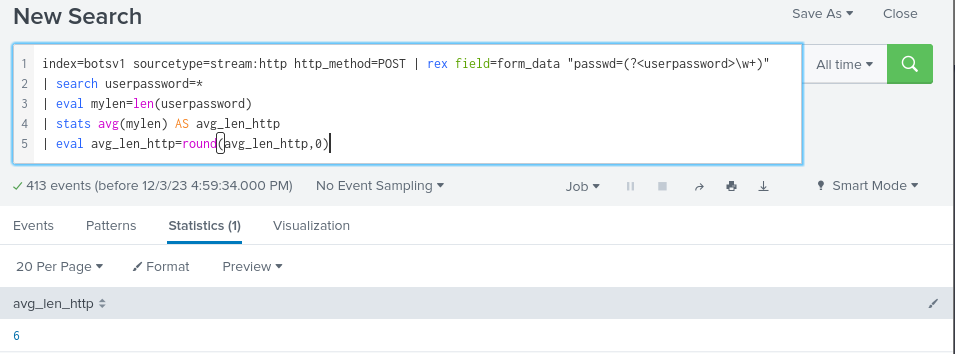
Once we have the userpassword list of values, all we really need to do is figure out the average length. While we have password strings, we need to convert them to a numeric first.



We did this earlier when we were looking for 6 character Coldplay songs and passwords. We use the eval command with the len function to calculate the userpassword length.

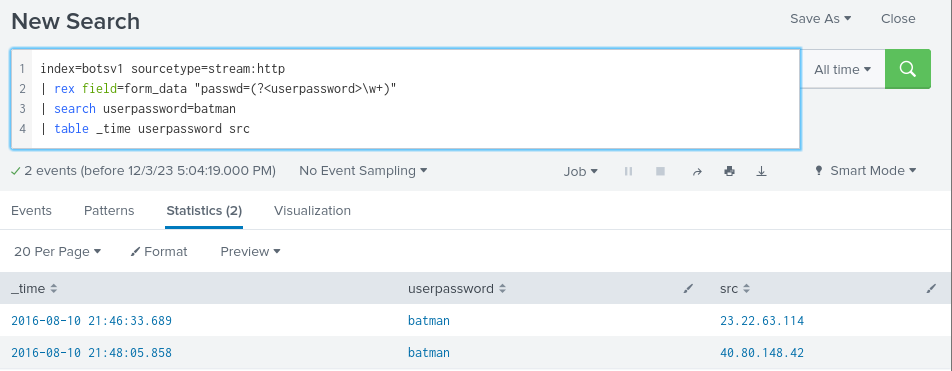


The next step is to calculate an average for these lengths. We have used the stats command with count and values but another function we can use is average. By using AS after the stats calculation we can rename the output of that calculation without using the rename command. It’s just an efficient way to change the name of the calculated value.

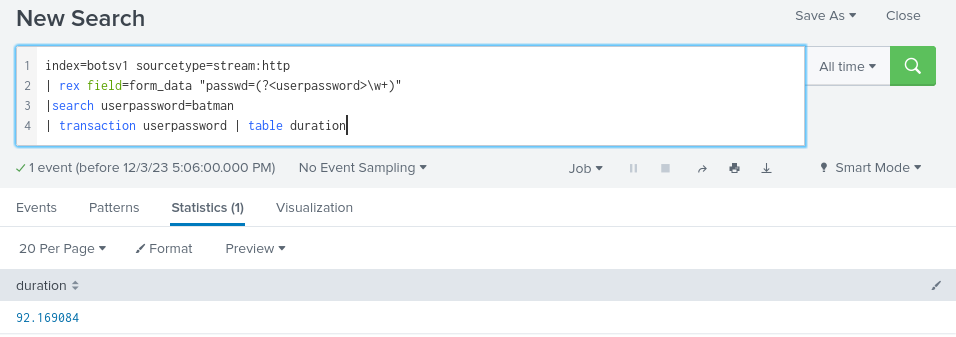


Finally, because we want the answer rounded off to the integer, we can add another eval command with the round function to specify the field we want to round. The 0 represents the number of decimal points we are rounding to and avg\_len\_http is the name of the field we are outputting the value to, in this case over-writng the decimal containing contents of avg\_len\_http with our rounded value.

#### **Determining The Elapsed Time Between Events**



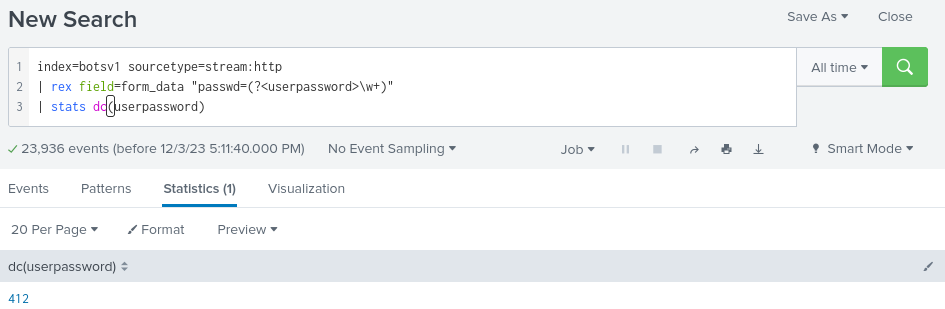
We know the password of interest is "batman" because we saw it being used twice, once during the scan and again from a different IP address to login. Based on that, we should look for the delta in time between these batman login events. We can visualize this by using the table command and grabbing the \_time and the src. At this point, we could pull out our fingers and toes and start calculating the number of seconds between these two events but let’s use Splunk to do this and then we have a search we can use in the future as well.

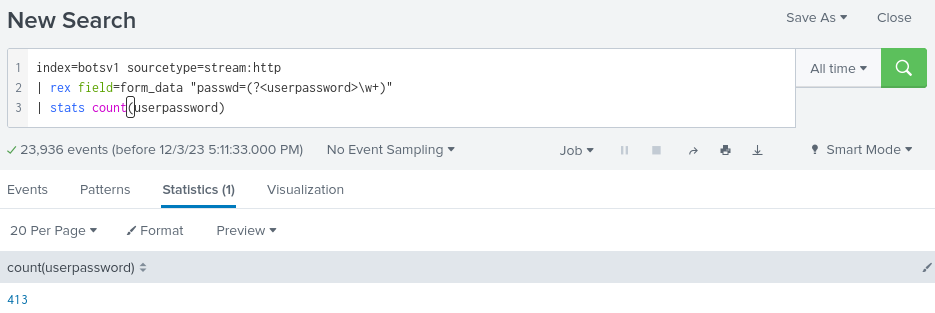


The transaction command gives us an easy way to group events together based on one or more fields and returns a field called duration that calculates the difference between the first and last event in a transaction. A transaction is defined by the common value or values specified. If we wanted to calculate the length of the vulnerability scan we could do that using the source address, but if we put the source address field into this search with userpassword, we would get two distinct transactions because the login came in from two different source addresses.

The question asked us to round the answer to two decimal places. We could do this with eval and round like we previously did, but if the results are appearing within a dashboard panel, it can also be set there.

#### **Identifying the Number of Unique Passwords Attempted During the Brute Force Attack**





With the use of the distinct count (dc) function within the stats command, this becomes very easy. Distinct count will look for every unique value within a field and count them. As you can see we had 413 total events that met our password criteria. If you stop and think about it, we saw the password batman twice, so it stands to reason that we would have 412 unique passwords attempted, but distinct count provides a way for us to calculate it and reuse that search.

## Part 3: Reporting

* What kind of attack was taking place on Wayne Enterprises systems?
  + website defacement attack and brute force attack
* Describe your findings for these stages of the kill chain:
  + Reconnaissance: For reconnaissance i was able to find what server that was being targeted, the IP was “192.168.250.70” and it was accessed 20000+ times. The domain used a CMS called Joomla which is an open source platform used to make web sites and software. Also The IP was scanned multiple times with a tool called “Acunetix web vulnerability scanner - free edition” and was victim of a cross-site scripting attack.
  + Exploitation: Found out there was 15000+ POST made from the IP “40.80.148.42” which made it seem that indicated that the server was being victim of a **brute force attack**, which it was. Found out that every password was introduced (412 to be exact) but only one was introduced twice which was “batman” once from the IP mentioned above and another time but from a different IP “23.22.63.114”. Also found out that the average passwords used on the server are 6 characters long which is a big no no for security.
* How could the kill chain have been disrupted to prevent the attacker from progressing?
  + First introduce longer passwords 8 characters long **minimum**. Account Lockout Policies, lock a user account after multiple login attempts, Multi Factor Authentication (MFA) and a firewall configured to block IP’s accessing multiple times in a short period of time.