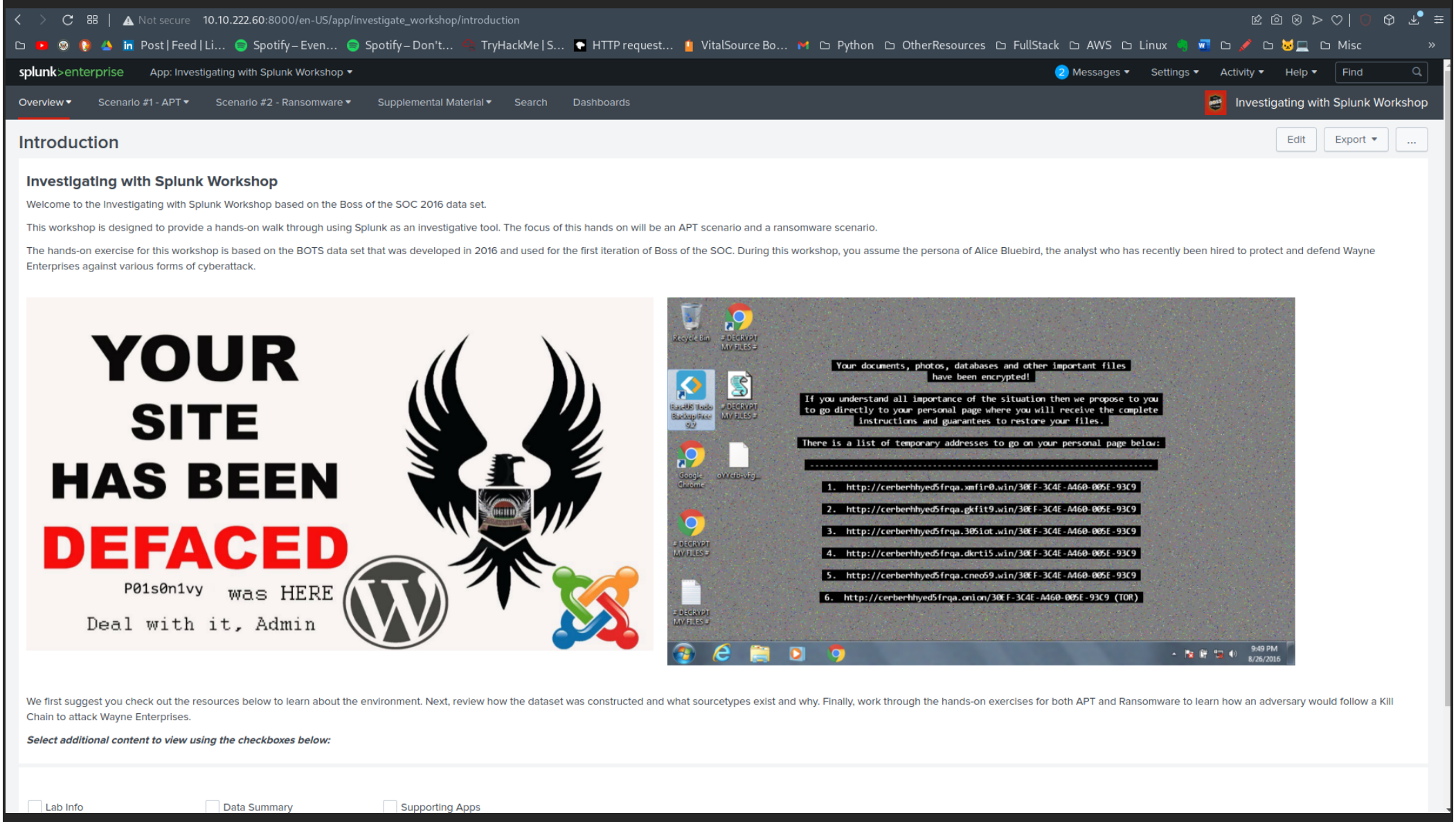
Splunk: 1b (Scenario #1-APT) Walkthrough part 1

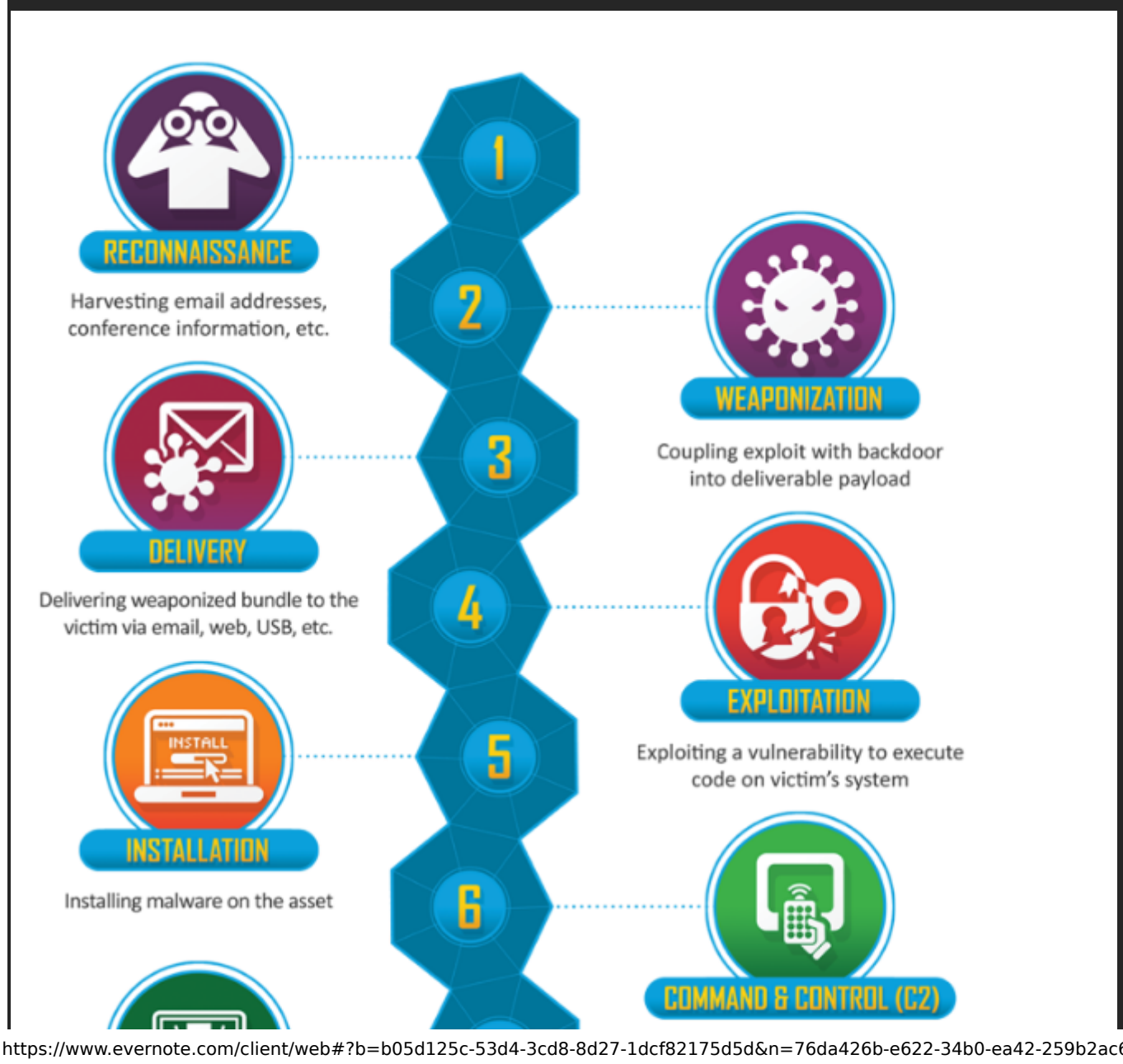
FROM: https://tryhackme.com/room/bpsplunk#

Task 4 Halp, I'm drowning in logs!



Scenario #1 - APT

In this scenario, reports of the below graphic come in from your user community when they visit the Wayne Enterprises website, and some of the reports reference "P01s0n1vy." In case you are unaware, P01s0n1vy is an APT group that has targeted Wayne Enterprises. Your goal, as Alice, is to investigate the defacement, with an eye towards reconstructing the attack via the Lockheed Martin Kill Chain.



GOOD THING I"M ALREADY FAMILIAR WITH THE LOCKHEED MARTIAN KILL CHAIN

LM-White-Paper-Intel-Driven-Defense.pdf 1 MB

During the scenario, we will build both Kill Chain and traffic flow diagrams to help visualize what has happened. While investigating, you may find yourself working backward to reconstruct an attack, but you may also find yourself in the middle of the Kill Chain based on indicators that are identified. If this happens, you must work BOTH backward to reconstruct the past but also forward to find if additional actions have transpired. For simplicity in this exercise, we will build across the Kill Chain using Splunk, starting with reconnaissance the adversary took and build things out as they move through the Kill Chain. That said, there are some phases of the Kill Chain that we will not address with Splunk and we will utilize Open Source Intelligence (OSINT) and other findings to fill in the gaps of the Kill Chain.

FINDING THE IP SCANNING YOUR WEB SERVER (1 of 2)

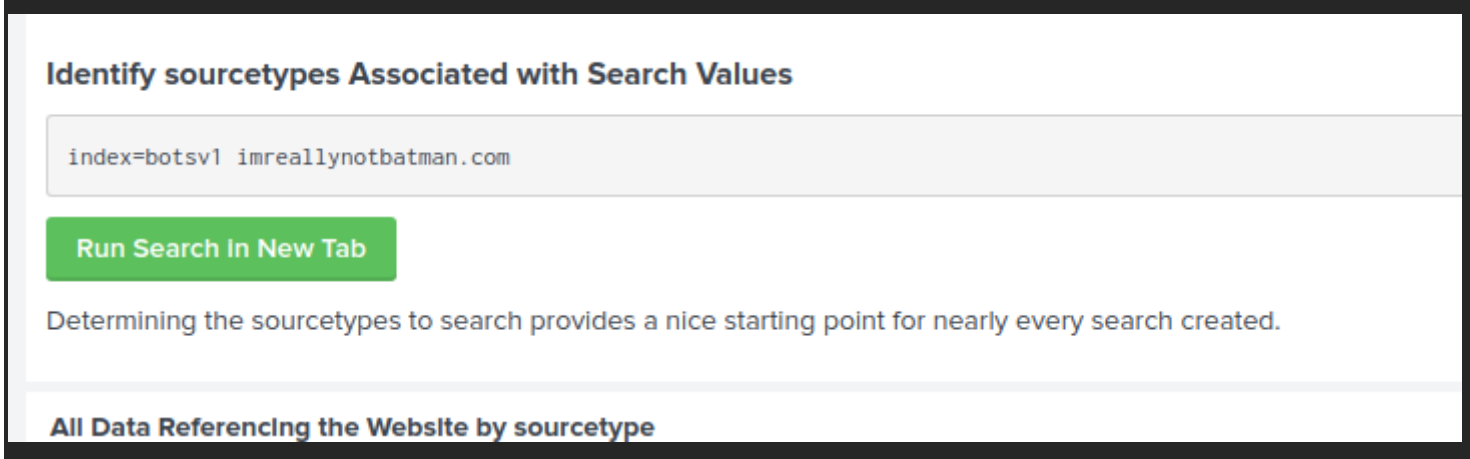
What is the likely IP address of someone from the P01s0n1vy group scanning imreallynotbatman.com for web application vulnerabilities?

Background

During an investigation, there are a number of questions that need to be answered. Some questions will build off of earlier questions, so it is important to capture and record this information as we go so it can be referenced later. Pro tip: have a notepad and pencil at the ready!

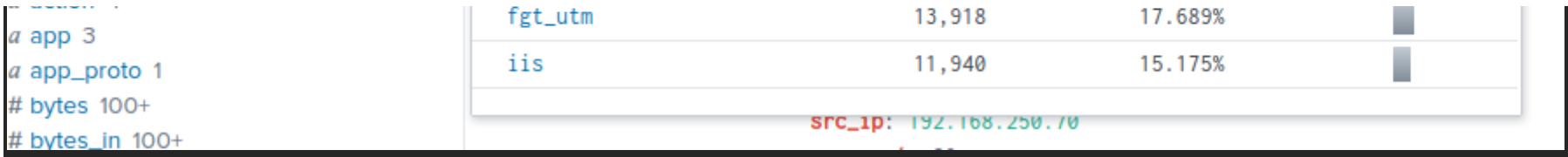
In this example, we're trying to understand the different sourcetypes that are scanning the website imreallynotbatman.com. From there we can drill down and further iterate on the search to find the IP address that is performing the scanning.

But before we dive in, this is an important Public Service Announcement! As we proceed, you will see SPL syntax in grey boxes and below that you will see a green button that will run the search in a new tab. For those familiar with Splunk, you may notice result sets in our dashboard do not exactly reflect the SPL syntax shown. The reason for this is because we are trying to simulate what you would see if you clicked the green search button and clicked on Selected Fields or Interesting Fields, as you can see in the image below. To simulate this result in the dashboard, we use a *stats count* command and return the top values. All right on with the show!



Determining the sourcetypes to search provides a nice starting point for nearly every search created.

sourcetype



I see 4, count them 4, source types.

FINDING SOURCE ADDRESSES

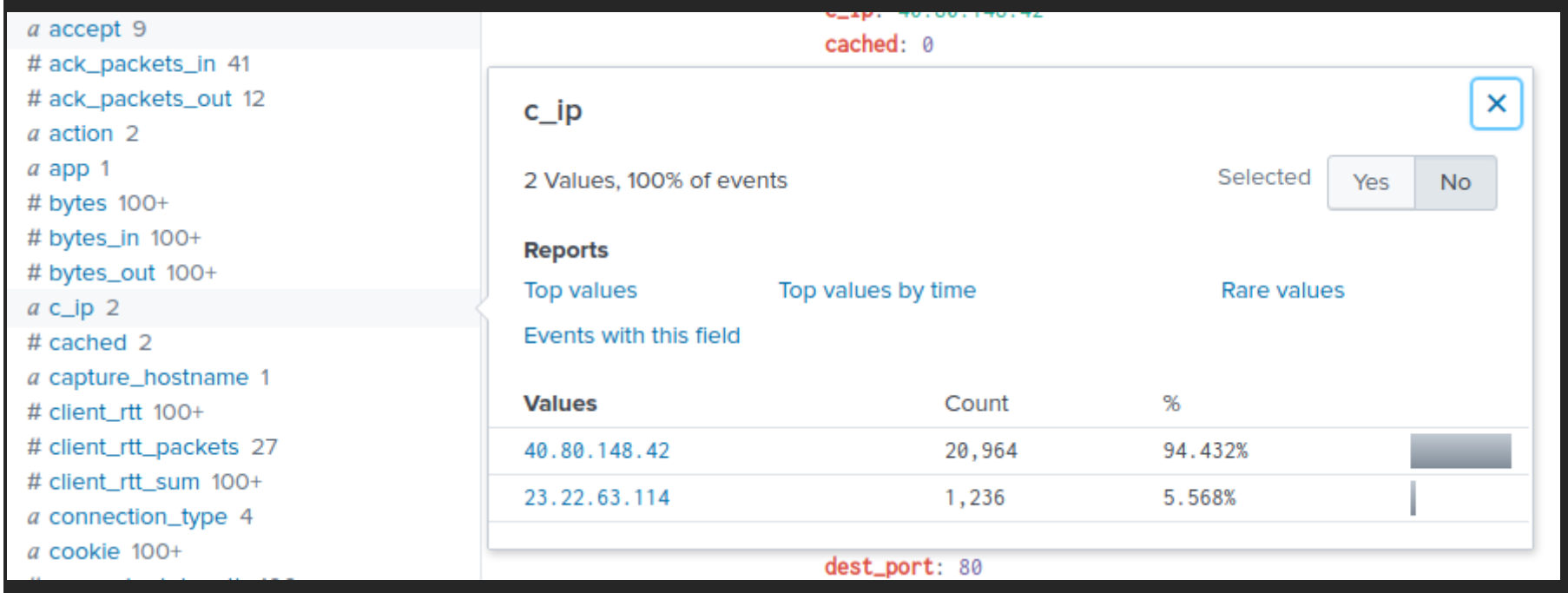
src_ip

Selecting a sourcetype and Searching for Source Address

Because the question asked about scanning the web site for vulnerabilities, it makes sense to drill down on the sourcetype of stream:http to see what kind of web data is being seen on the wire. Stream is a free app for Splunk that collects wire data and can focus on a number of different protocols including smtp, tcp, ip, http and so on.

Once we have narrowed our search to stream:http data, we can again drill down and see what source IP addresses are associated with our domain. At this point, only two source IPs are seen in the http traffic. One of them is associated with 95% of the traffic, so a reasonable assumption would be that this address is the one of greatest interest.

###LOOKS LIKE COUNT BY SOURCE is c_ip; I AMMOST MISSED THIS!!!



That said, we should validate this hypothesis and since we have other sourcetypes available to us, perhaps we should use them.

2 of 2

What is the likely IP address of someone from the P01s0n1vy group scanning imreallynotbatman.com for web application vulnerabilities?

Background

While we have an answer, it is important during an investigation to validate our findings. Different data sets provide different perspectives to the same question, so finding corroboration within our data sets is important.

If we take the IP address we found with our stream data and change our sourcetype to Suricata (an IDS tool and more), we can see a number of logged events coming from that source IP address. Because Suricata has IDS signatures associated with it, we can scroll through the interesting fields on the left side of the screen and discover additional fields to look within. We can also scroll to the bottom of the list and we will see that we have 58 more fields.

Kill Chain Phase

- Reconnaissance

Sourcetypes

- suricata, stream*

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

suricata Signatures on 40.80.148.42

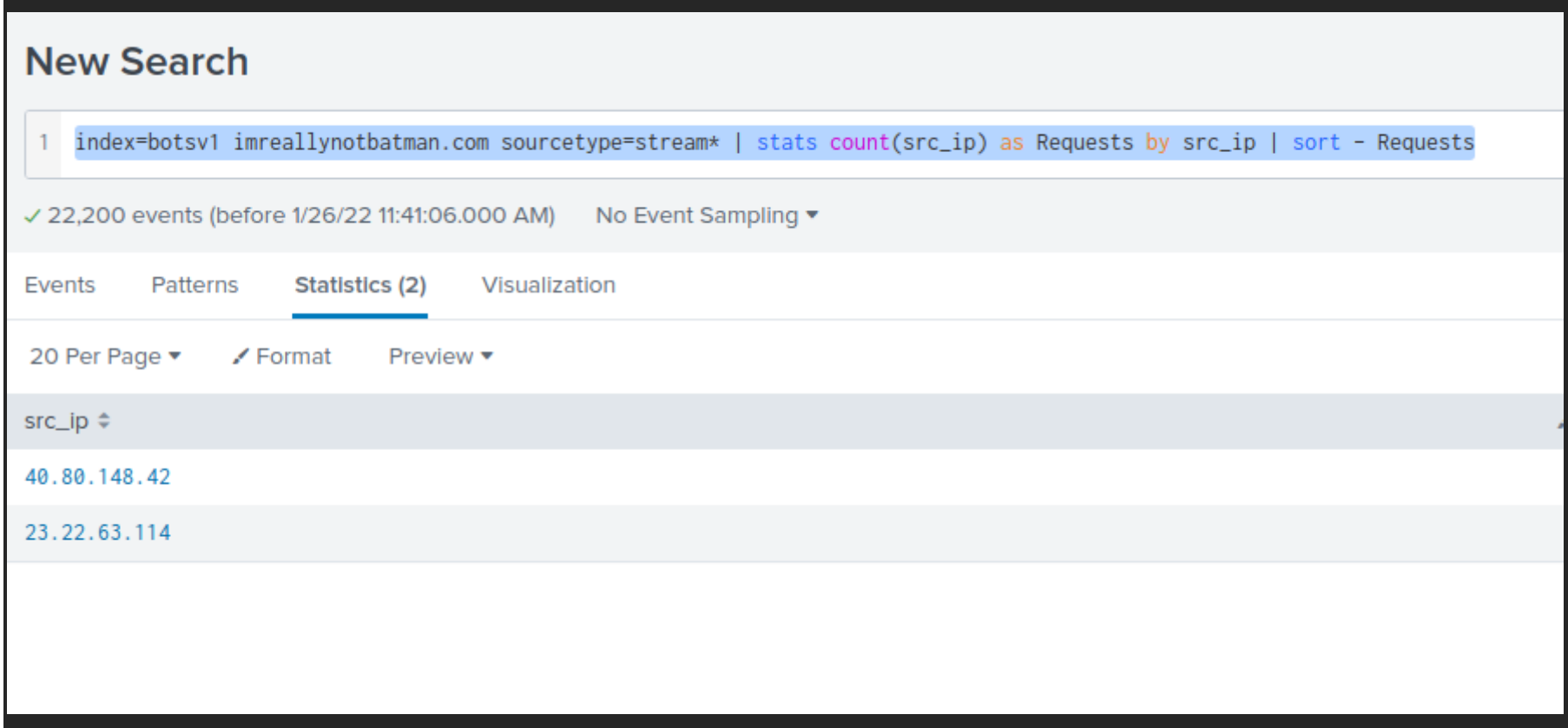
i couldnt figure out how to show the signature like this :((((

signature	count	percentage
ET WEB_SERVER Script tag in URI, Possible Cross Site Scripting Attempt	103	21.78
ET WEB_SERVER Onmouseover= in URI - Likely Cross Site Scripting Attempt	48	10.15
ET WEB_SERVER Possible XXE SYSTEM ENTITY in POST BODY.	41	8.67
SURICATA HTTP Host header invalid	35	7.40
ET WEB_SERVER Possible SQL Injection Attempt SELECT FROM	33	6.98
ET WEB_SERVER SQL Injection Select Sleep Time Delay	32	6.77
ET WEB_SERVER Possible CVE-2014-6271 Attempt	18	3.81
ET WEB_SERVER Possible CVE-2014-6271 Attempt in Headers	18	3.81
ET WEB_SERVER PHP tags in HTTP POST	13	2.75
GPL WEB_SERVER global.asa access	12	2.54

« prev 1 2 3 4 5 next »

Another Option: Search Stream sourcetype and Count the src_ip

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



Identifying The Web Vulnerability Scanner

What company created the web vulnerability scanner used by P01s0n1vy?

Background

As we move to the next question, it is also important to remember that when hunting or handling incidents, results may lead to additional questions and searches may build on one another. Outputs from previous questions may provide you with hints.

Since we know from answering the prior question that web vulnerability scans originated from the particular source IP 40.80.148.42, we can search for traffic from 40.80.148.42 and examine the src_headers for clues related to the tool being used for the scan.

Kill Chain Phase

- Reconnaissance

- Reconnaissance

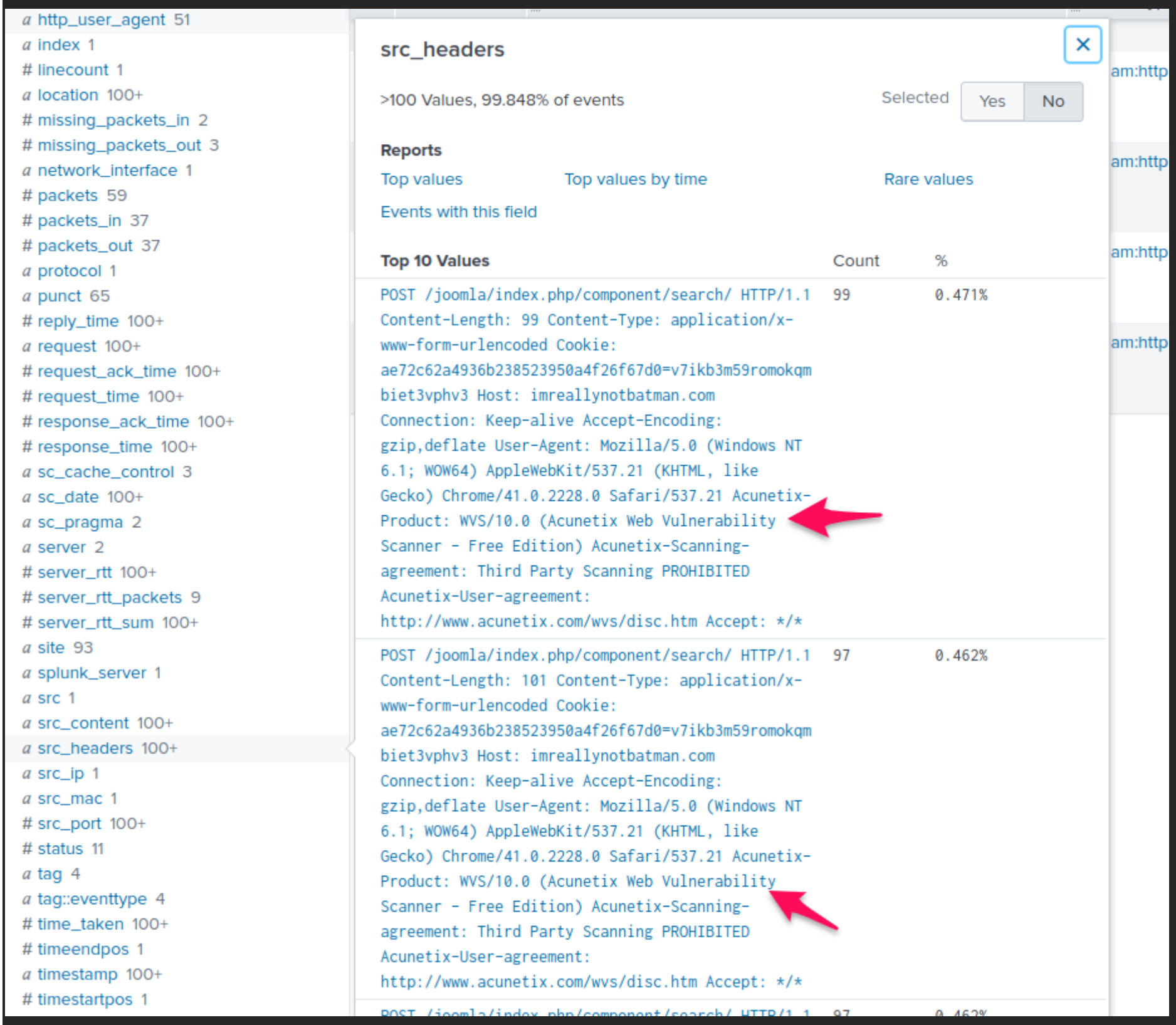
Sourcetypes

- stream:http

Looking at src_headers

```
index=botsv1 src=40.80.148.42 sourcetype=stream:http
```

If we zoom in on the source headers, we can see information that says (Acunetix Web Vulnerability Scanner - Free Edition), so our antenna is up based on that clue.



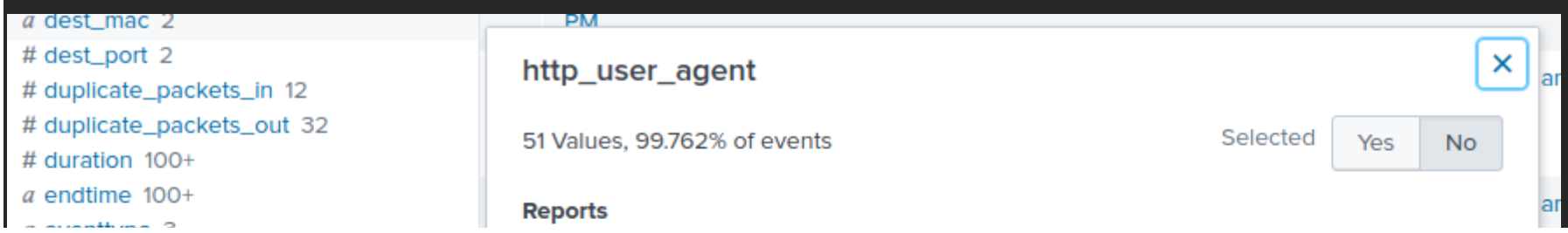
Looking at http_user_agent strings

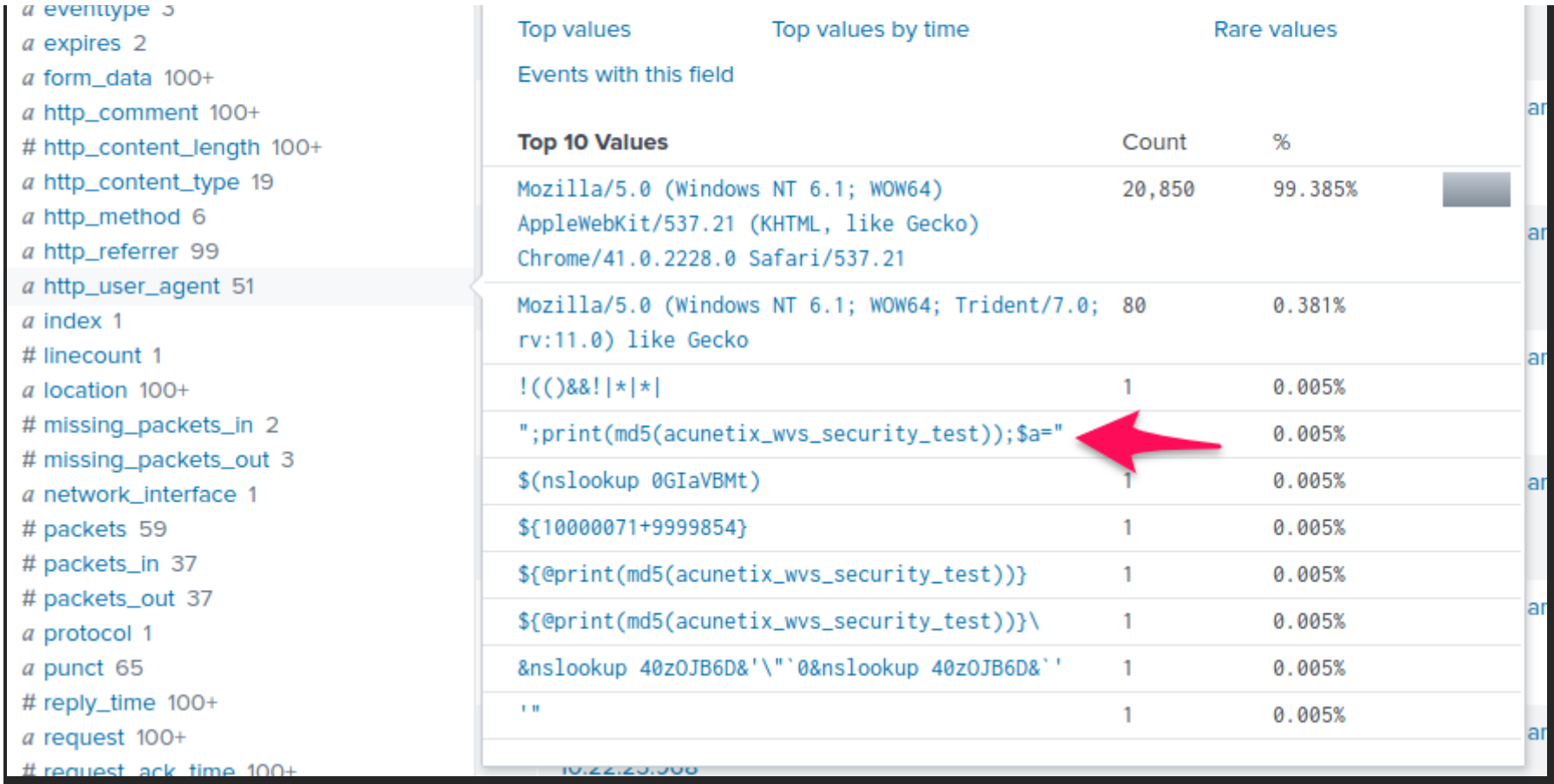
```
index=botsv1 src=40.80.148.42 sourcetype=stream:http
```

We can take this a step further and look in other fields of the HTTP wire data. If we look at http_user_agent, we see some anomalous/unusual agent strings and in a few of them we again see Acunetix.

Unfortunately, I may not be well versed in web application vulnerability scanners. What's an analyst to do?

Examine http_user_agent on [40.80.148.42](40.80.148.42)





Research on the Internet

If we see something in our logs and we don't know what it is, Google it!

Google for Acunetix

As you can see from our search, Acunetix is in fact a web vulnerability scanner so we can confirm our earlier findings.

Determining Which Web Server is the Target

What content management system is imreallynotbatman.com likely using?

Background

This question is similar to the previous questions in the sense that we are trying to get our bearings and figure out the systems that are operating within the environment. Now we want to know what is running on our website because IT and Dev may not have told us how our production applications are built. Once we understand the underlying techgnology, we can also understand the vulnerabilities that exist on our systems. Then, we can eradicate the threat, and also patch to prevent these vulnerabilities from being exploited in the future.

Kill Chain Phase

- Reconnaissance

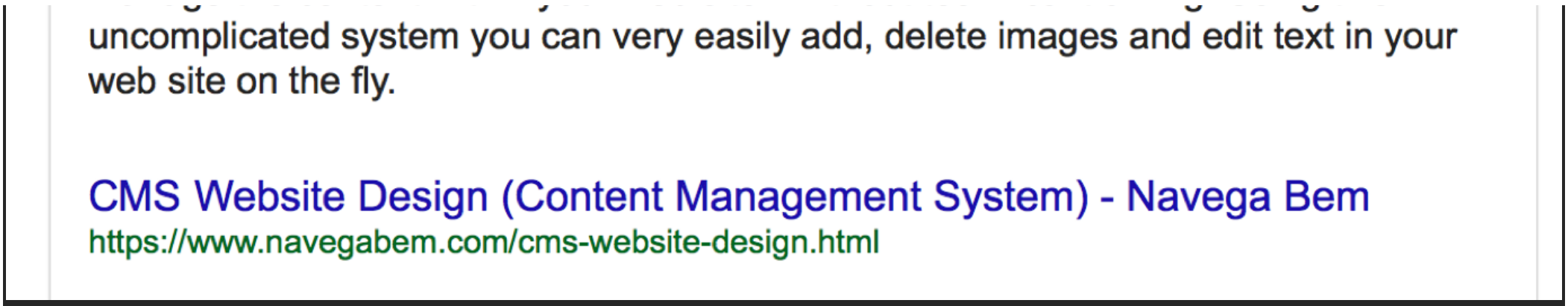
Sourcetypes

- stream:http, iis

Where To Start?

I am a security analyst. I don't know much about web servers. I really don't know anything about content management systems and what they do. So, back to Google to figure out what a CMS system is.

Google for Content Management System

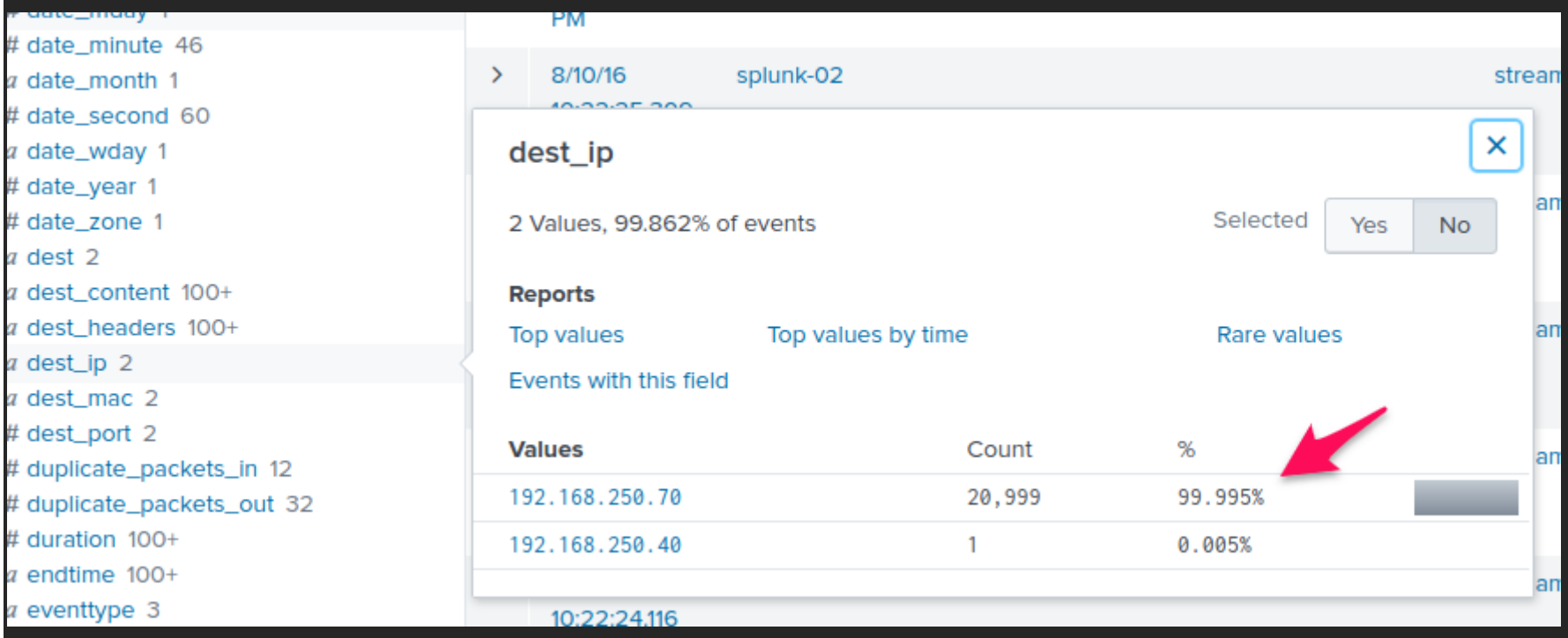


Now that we know a little bit more about web content management systems, we should figure out the kinds of logs needed to dig deeper.

We should also identify the IP address of the victim system.

```
index=botsv1 src=40.80.148.42 sourcetype=stream:http
```

I'm going to guess destination ip--I was right!!!!



Based on what we have discovered previously, we have a good idea of some of the logs to look at as well as the IP of the system in question. Because the question asked about scanning the web site for vulnerabilities, it makes sense to drill down on the sourcetype of stream:http to see what kind of web data is being seen on the wire. **Stream is a free app for Splunk that collects wire data and can focus on a number of different protocols, including but not limited to smtp, tcp, ip, http and more.**

index=botsv1 src=40.80.148.42 sourcetype=stream:http
^I guess this is how to use stream

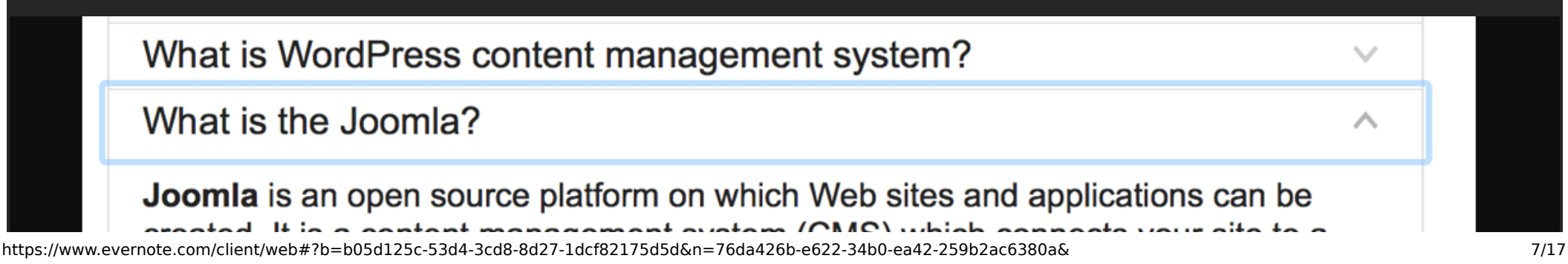
Once we have narrowed our search to stream:http data, we can again drill down and see what source IP addresses are associated with our domain. It appears that only two source IPs are seen in the http traffic. One of them is associated with 95% of the traffic, so a reasonable assumption would be that this address is the one of greatest interest.

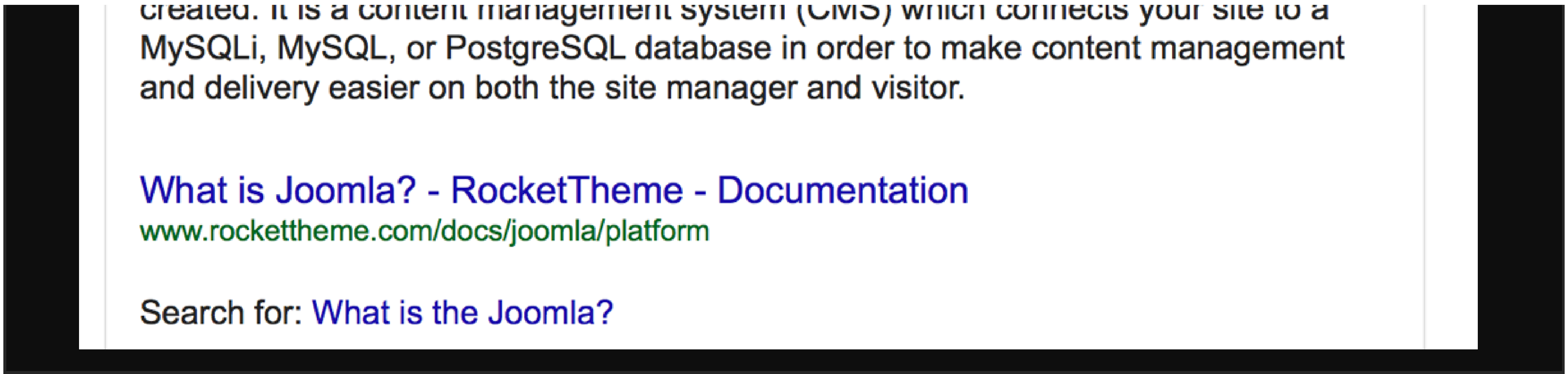
I think this is a typo; don't they mean dest ip?

Digging into the URI

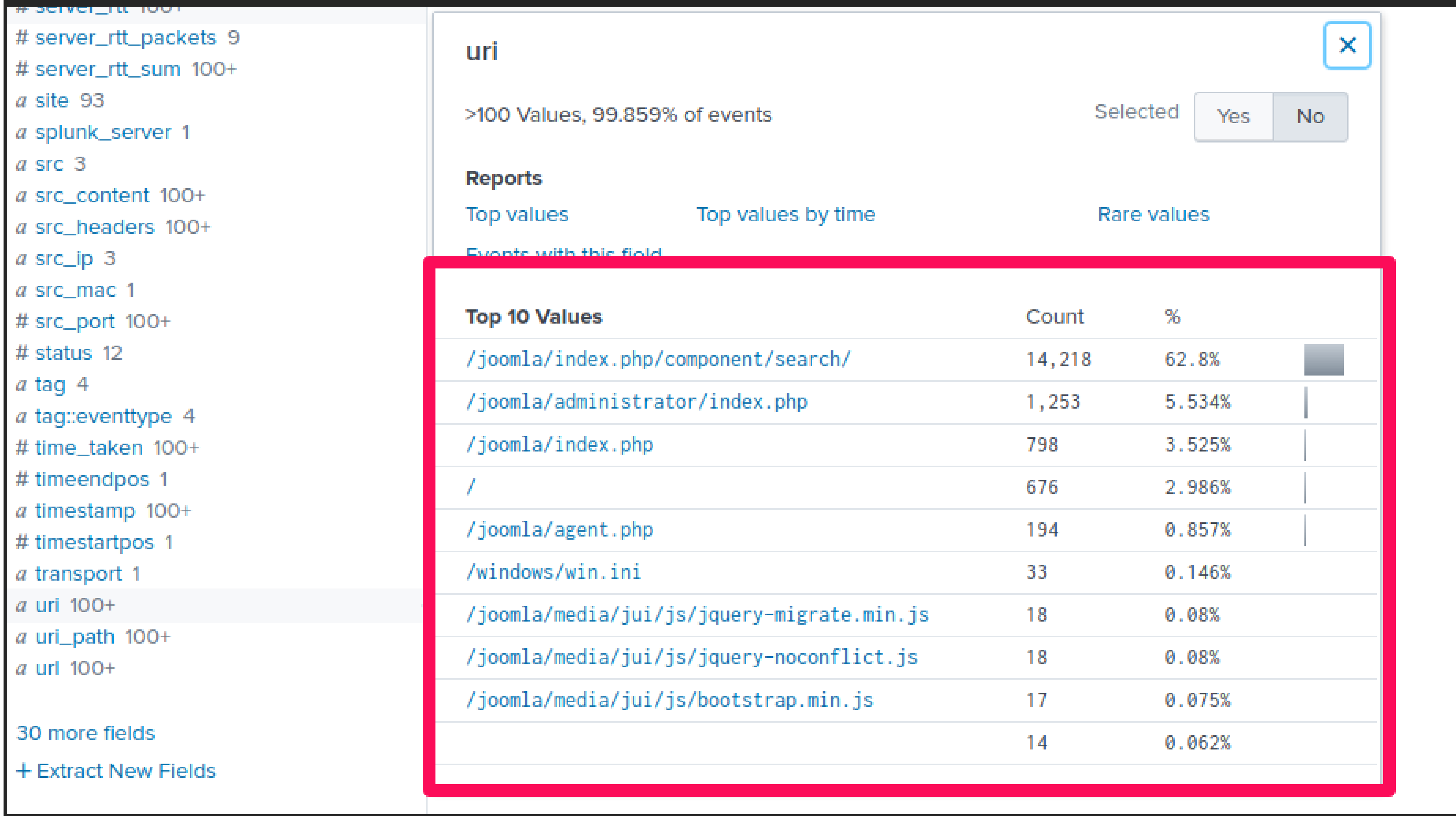
```
index=botsv1 dest=192.168.250.70 sourcetype=stream:http
```

Now that we know the sourcetype AND we know the IP of the web server, we can start looking at URLs or URIs to get an idea of the kind of files and directory structures being requested. As we look through the list, we see the word joomla on a number of these directory trees. I don't know a person named joomla in the office, so where should we go to learn more?
Google for Joomla

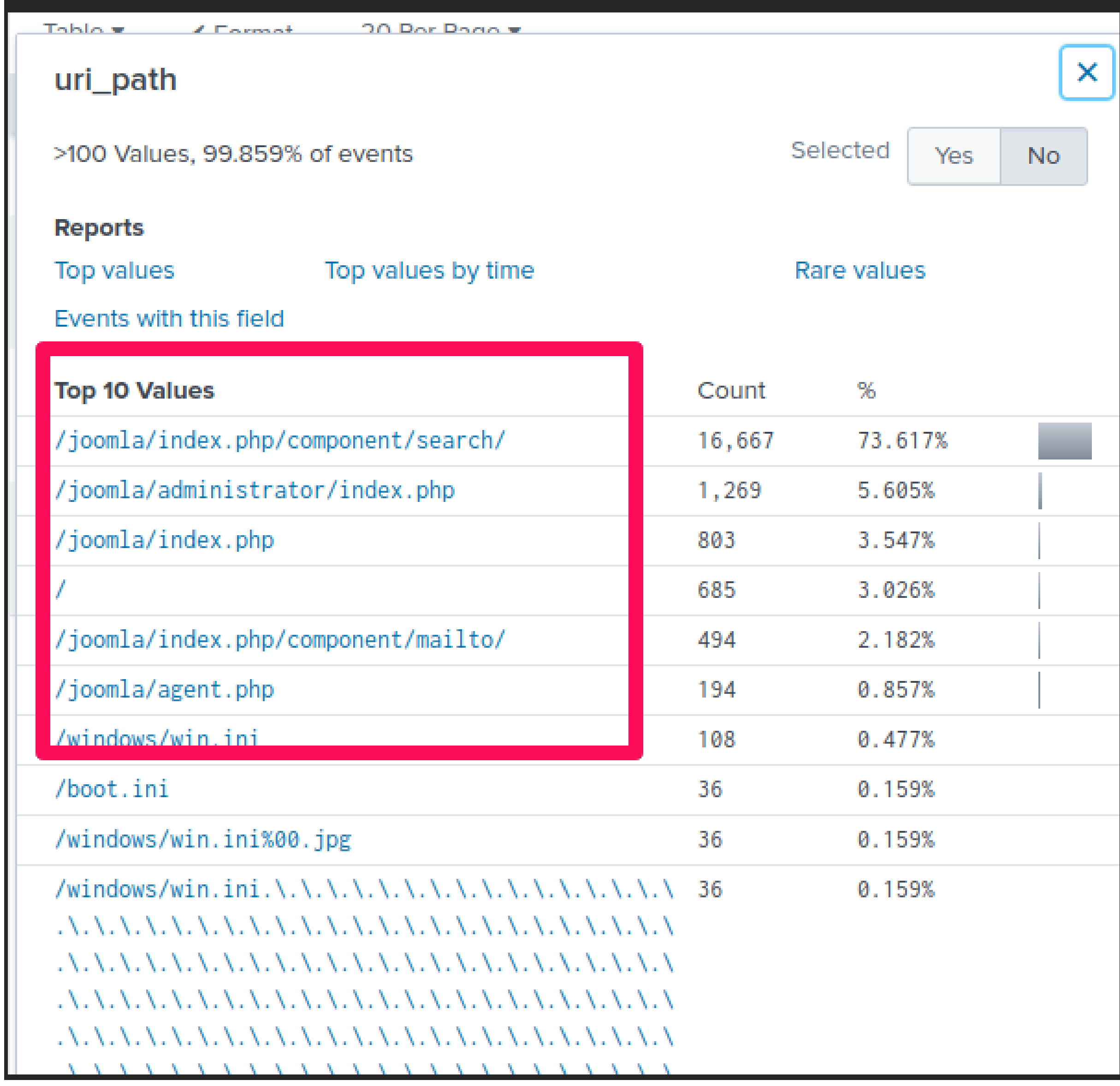




It's an open source CMS system. Sounds like we are on the right path.



Top 10 Values	Count	%
/joomla/index.php/component/search/	14,218	62.8%
/joomla/administrator/index.php	1,253	5.534%
/joomla/index.php	798	3.525%
/	676	2.986%
/joomla/agent.php	194	0.857%
/windows/win.ini	33	0.146%
/joomla/media/jui/js/jquery-migrate.min.js	18	0.08%
/joomla/media/jui/js/jquery-noconflict.js	18	0.08%
/joomla/media/jui/js/bootstrap.min.js	17	0.075%
	14	0.062%



Top 10 Values	Count	%
/joomla/index.php/component/search/	16,667	73.617%
/joomla/administrator/index.php	1,269	5.605%
/joomla/index.php	803	3.547%
/	685	3.026%
/joomla/index.php/component/mailto/	494	2.182%
/joomla/agent.php	194	0.857%
/windows/win.ini	108	0.477%
/boot.ini	36	0.159%
/windows/win.ini%00.jpg	36	0.159%
/windows/win.ini.\.	36	0.159%

PERSONAL RESEARCH
What's the difference between URL and URI?

A URL provides the location of the resource. A URI identifies the resource by name at the specified location or URL. Other URIs provide only a unique name, without a means of locating or retrieving the resource or information about it, these are Uniform Resource Names (URNs).

https://en.wikipedia.org › wiki › Uniform_Resource_Ident...

Uniform Resource Identifier - Wikipedia

Shit! URNs now!

Looking for Confirmation

```
index=botsv1 dest=192.168.250.70 sourcetype=stream:http status=200
```

Let’s take our original search to find our URIs and improve it a bit. We are going to add a status of 200 to our search to indicate successful page loads.

```
| stats count by uri
```

We can also add a transformational search command called ***stats*** that will allow us to count the number of events grouped by URI. ***stats*** is fantastic and we will be using it throughout this workshop in a number of ways. In fact, we have already been using ***stats*** in our initial searches when we looked at counts of fields that we have been inspecting.

```
| sort - count
```

index=botsv1 dest=192.168.250.70 sourcetype=stream:http status=200
| stats count by uri
| sort - count

```
1 index=botsv1 dest=192.168.250.70 sourcetype=stream:http status=200
2 | stats count by uri
3 | sort - count
```

Finally, we are going to issue another transformational command called **sort** and return results in descending order based upon the count so that our result set is formatted in the manner we would like to see it.
I'm guessing the "- count" means descending order

When we get our results back, we see the vast majority of URIs on the server start with "joomla" in the URI.

Confirm Our Suspicions Adding in Successful Page Loads

New Search

```
1 index=botsv1 dest=192.168.250.70 sourcetype=stream:http status=200
2 | stats count by uri
3 | sort - count
```

✓ 7,365 events (before 1/26/22 12:24:13.000 PM) No Event Sampling

Events Patterns **Statistics (2,438)** Visualization

20 Per Page Format Preview

uri
/joomla/index.php/component/search/
/joomla/administrator/index.php
/joomla/index.php
/

```
/joomla/agent.php
/windows/win.ini
/joomla/media/jui/js/jquery-migrate.min.js
/joomla/media/jui/js/jquery-noconflict.js
/joomla/media/jui/js/bootstrap.min.js
/joomla/media/system/js/html5fallback.js
/joomla/templates/protostar/js/template.js
/boot.ini
/joomla/media/jui/js/jquery.min.js
/windows/win.ini%00.jpg
/joomla/index.php/component/search/?Itemid=115&searchphrase=all&searchword=the
/joomla/index.php/component/search/?Itemid=101&searchphrase=all&searchword=
/joomla/index.php/component/search/?catid=9&format=opensearch&id=4&Itemid=101
/joomla/index.php/component/search/?format=opensearch&id=9&Itemid=101&layout=blog
/joomla/templates/protostar/css/template.css
/windows/win.ini&searchword=
```

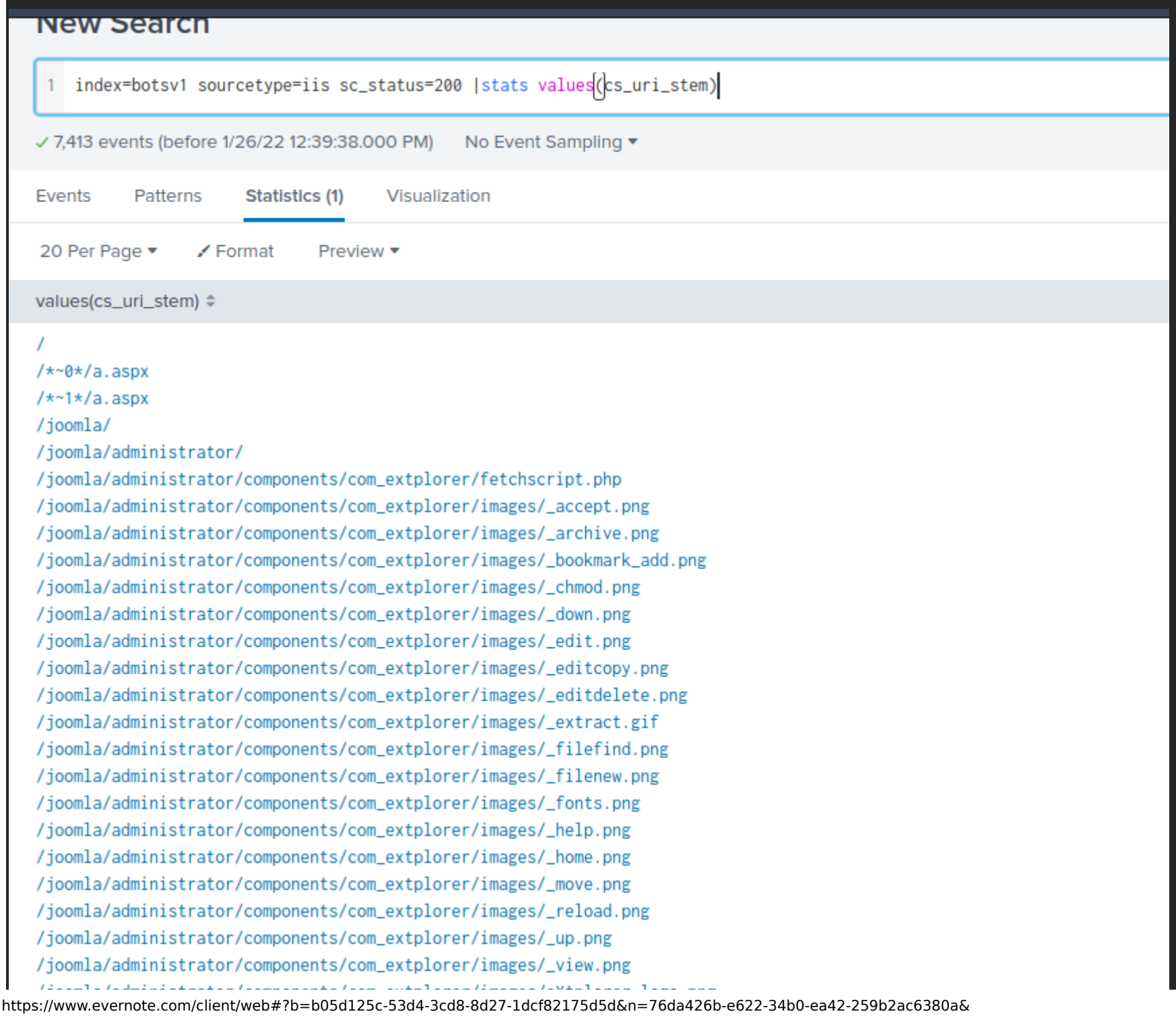
Finding the Answer With IIS

```
index=botsv1 sourcetype=iis sc_status=200 | stats values(cs_uri_stem)
```

There are many ways to arrive at this answer. As we found earlier, imreallynotbatman.com is hosted on 192.168.250.70. We have IIS logs for that host (we1149srv). We could search the IIS logs and examine the URI strings being accessed to find indicators and verify that the http response code equals 200.

index=botsv1 sourcetype=iis sc_status=200 |stats values(cs_uri_stem)

If You Wanted To Use IIS Data Instead of Stream



```
/joomla/administrator/components/com_extplorer/images/extplorer_logo.png
/joomla/administrator/components/com_extplorer/images/extension/document.png
/joomla/administrator/components/com_extplorer/images/extension/exe.png
/joomla/administrator/components/com_extplorer/images/extension/folder.png
/joomla/administrator/components/com_extplorer/images/extension/htm.png
/joomla/administrator/components/com_extplorer/images/extension/php.png
/joomla/administrator/components/com_extplorer/images/extension/png.png
/joomla/administrator/components/com_extplorer/images/extension/txt.png
/joomla/administrator/components/com_extplorer/scripts/extjs3/resources/images/default/button/arrow.gif
/joomla/administrator/components/com_extplorer/scripts/extjs3/resources/images/default/button/s-arrow-noline.gif
/joomla/administrator/components/com_extplorer/style/style.css
/joomla/administrator/index.php
```

Identifying Where a Brute Force Attack Originated

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

Background

If we are attempting to figure out the IP address that is attacking our website, what kinds of data might we need to have to determine this?

Kill Chain Phase

- Exploitation

Sourcetypes

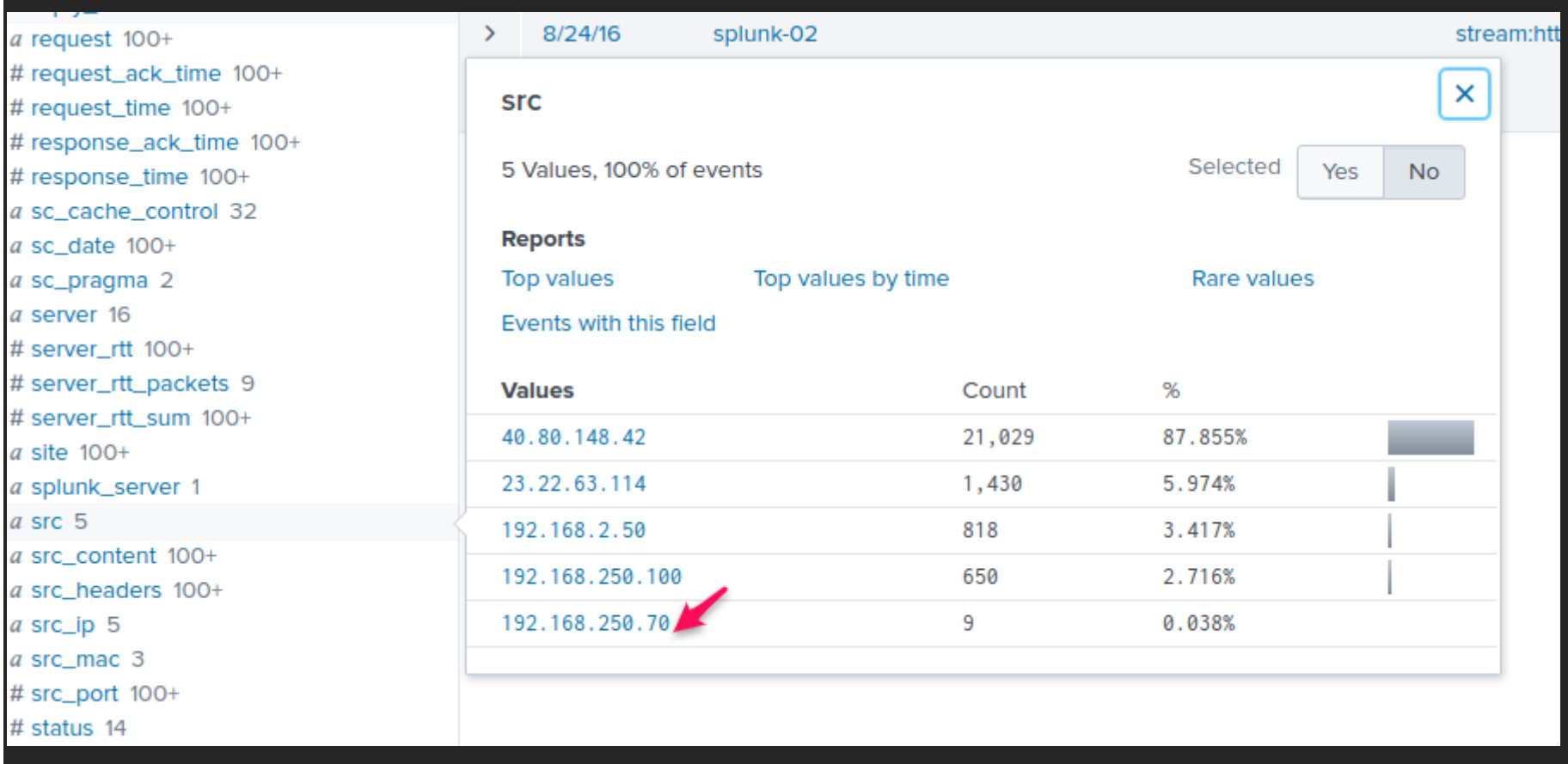
- stream:http

Looking in Wire Data - stream:http

```
index=botsv1 sourcetype=stream:http
```

Because stream data allows us to look at HTTP traffic, this would seem to be a good choice of data to start with.

If we start with a very broad search of the index and the sourcetype, we can pivot into the source of the traffic to see that there are a number of source IP addresses that are associated with http events. Because we have been taking good notes through our investigation, we know what the IP address of imreallynotbatman.com is, right?

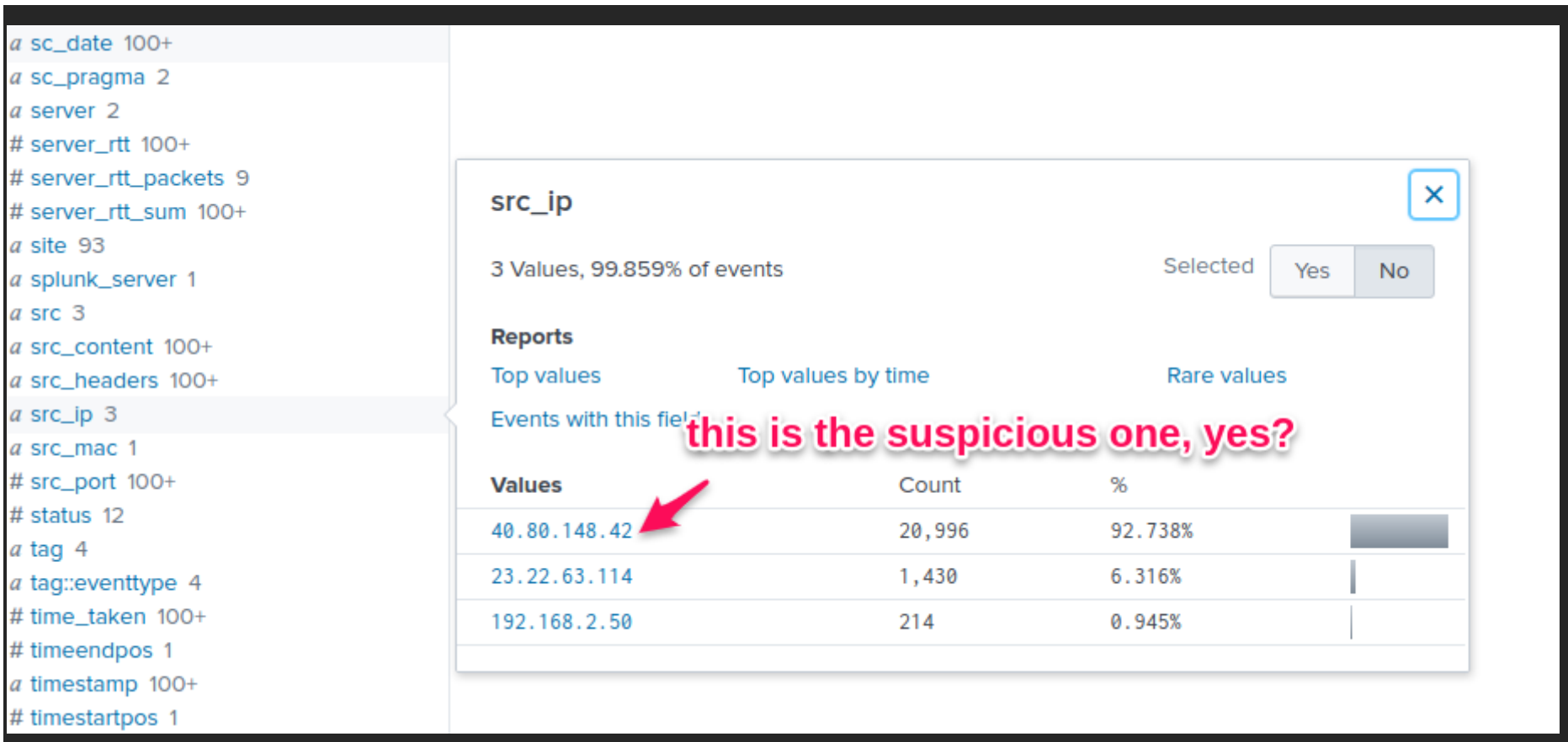


Refine Our Search with the Web Server Address

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

Refine Our Search with the Web Server Address

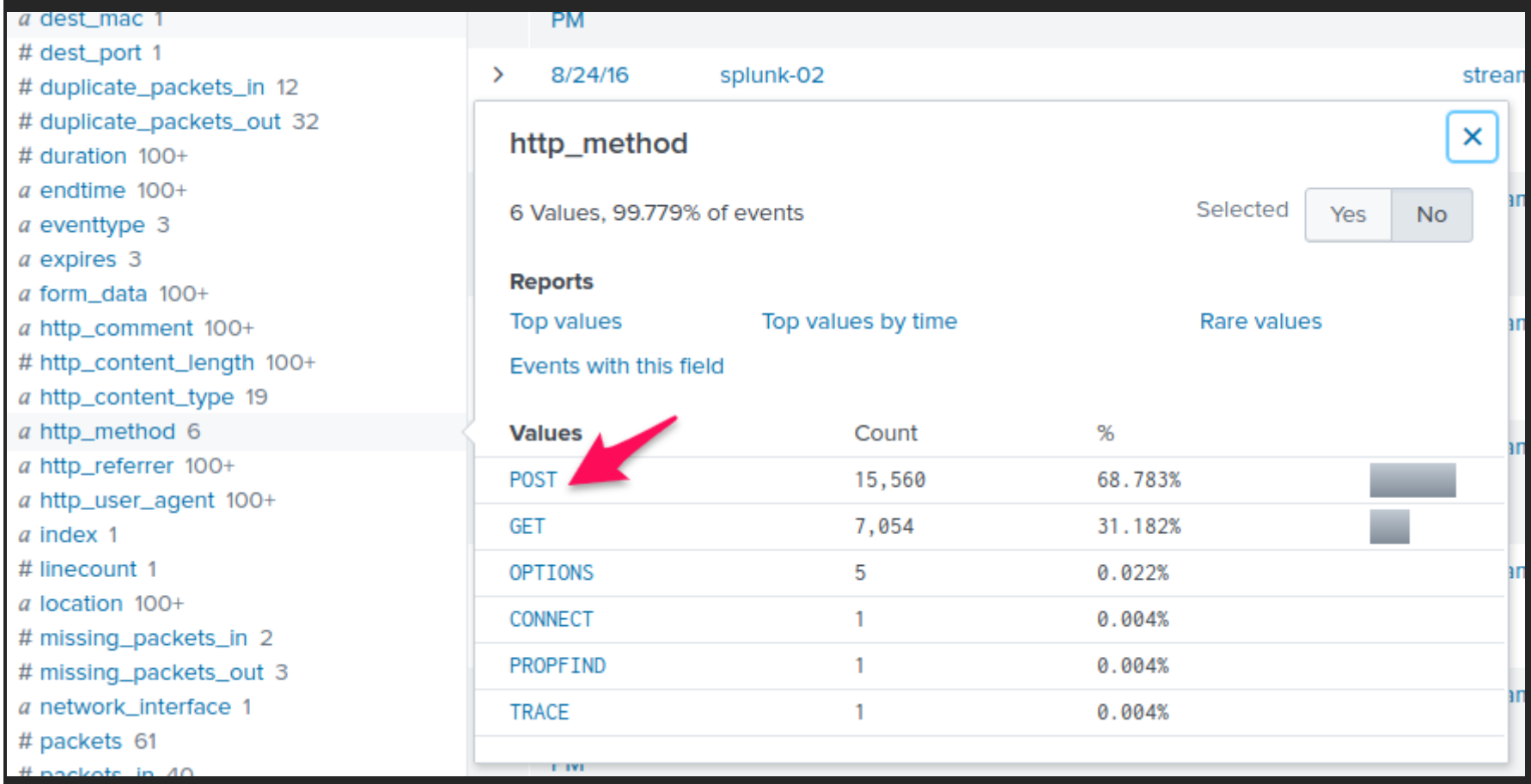
```
index=botsv1 sourcetype=stream:http dest="192.168.250.70"
```



Same Search, Different Pivot

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

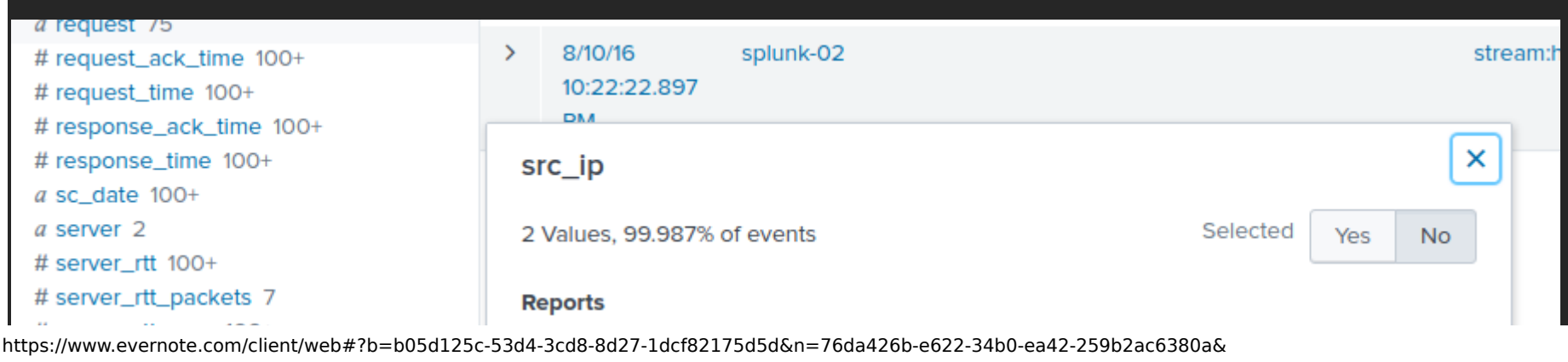
If we pivot to look at the http_method of these requests, we see nearly twice as many POSTs as GETs. Since we are talking about a brute force password attack of a web site, what is the most likely http method for this kind of attack?--POST



Adding a HTTP Method to Narrow Our Results

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

Because we determined that a brute force password attack to a web page would most likely be a POST, let's add this to our search. Now we see only two source IP addresses with the vast majority coming from the 40.80.148.42.



At this point, let's take a quick poll, who thinks we have enough information to answer this question?

Who thinks the answer is 40.80.148.42? Who thinks the answer is 23.22.63.114? Why?

Finding Passwords in HTTP Wire Data

Where might we see passwords during a web login?

If the website is not encrypted or it is being decrypted for logging, we may be able to see login attempts.

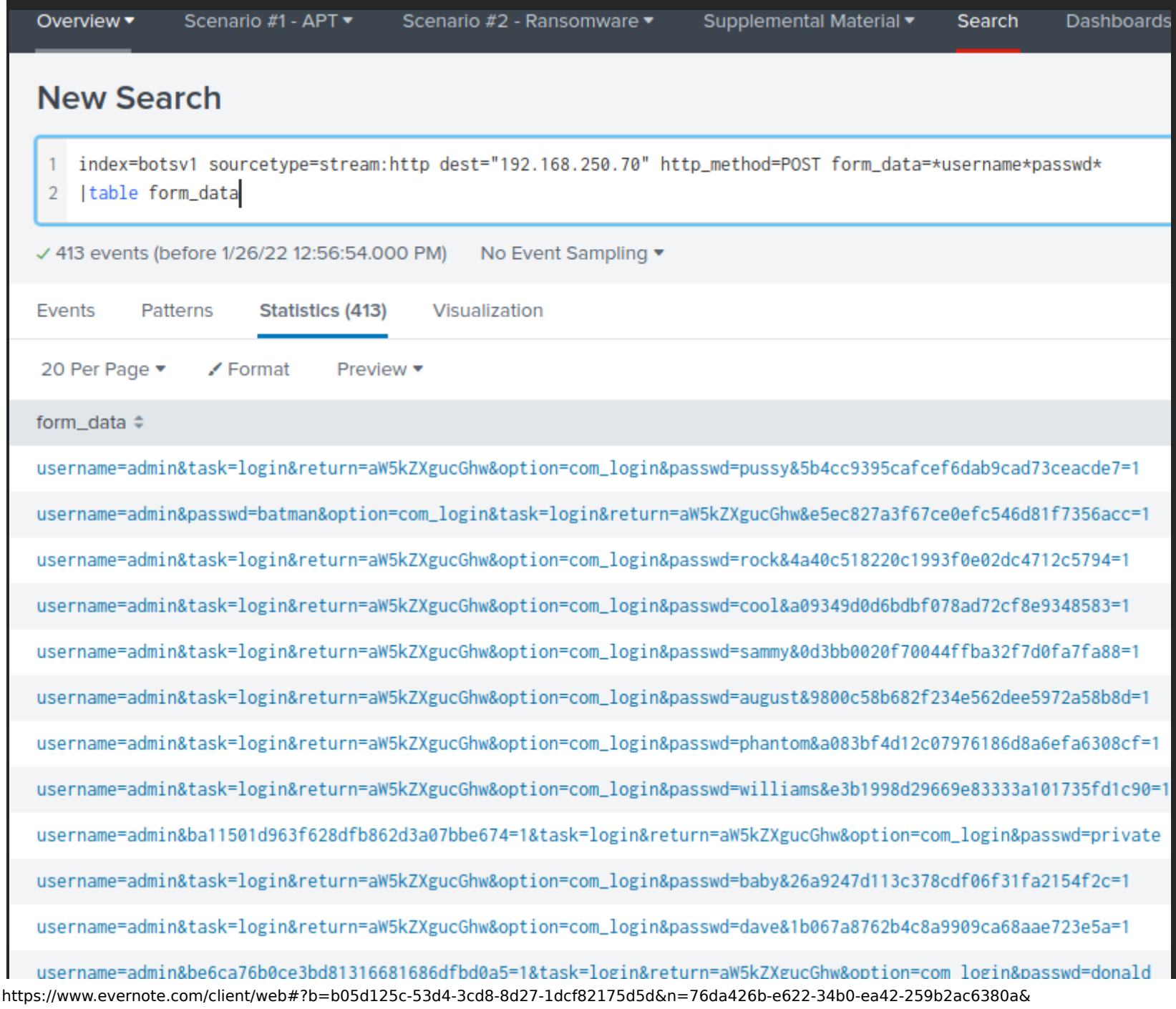
What field might we see username and passwords in?

```
index=botsv1 sourcetype=stream:http dest="192.168.250.70" http_method=POST form_data=*username*passwd*
```

The form_data field contains information being passed from the client browser to the web server. If we search the form_data, we can use wildcards to look for values that contain the strings "username" and "passwd" within the field. This could be a very expensive search done at scale which is why it is important to set the index= and the time picker to narrow the amount of data to be searched.

```
| table form_data
```

index=botsv1 sourcetype=stream:http dest="192.168.250.70" http_method=POST form_data=*username*passwd*
|table form_data



```
username=admin&task=login&return=aW5kZXgucGhw&option=com_login&passwd=lifehack&5804a636e6c85901f132655dae4add9b=1
username=admin&task=login&return=aW5kZXgucGhw&option=com_login&passwd=godzilla&44cb1c161d5917d961c94eb708ac7c1a=1
username=admin&923b285652f28decc583dab0141b5c13=1&task=login&return=aW5kZXgucGhw&option=com_login&passwd=4444
username=admin&0960d493674eb04861bd64da9b662118=1&task=login&return=aW5kZXgucGhw&option=com_login&passwd=arthur
username=admin&task=login&return=aW5kZXgucGhw&option=com_login&passwd=yankee&4e356e4bbd0129a852b8364810ff675f=1
username=admin&task=login&return=aW5kZXgucGhw&option=com_login&passwd=runner&dd3217f4da1cf80f7e97dbd8b1b94194=1
username=admin&task=login&return=aW5kZXgucGhw&option=com_login&passwd=ou812&7a858d7107d9a2946a31987c12794595=1
username=admin&task=login&return=aW5kZXgucGhw&option=com_login&passwd=bill&496a906140f93f84bb2368a606eed63a=1
10.10.222.60:8000/en-US/app/investigate_workshop/search?q=search index%3Dbotsv1 sourcetype%3Dstream%3
```

Using stats for the Win!

```
index=botsv1 sourcetype=stream:http dest="192.168.250.70" http_method=POST form_data=*username*passwd*
```

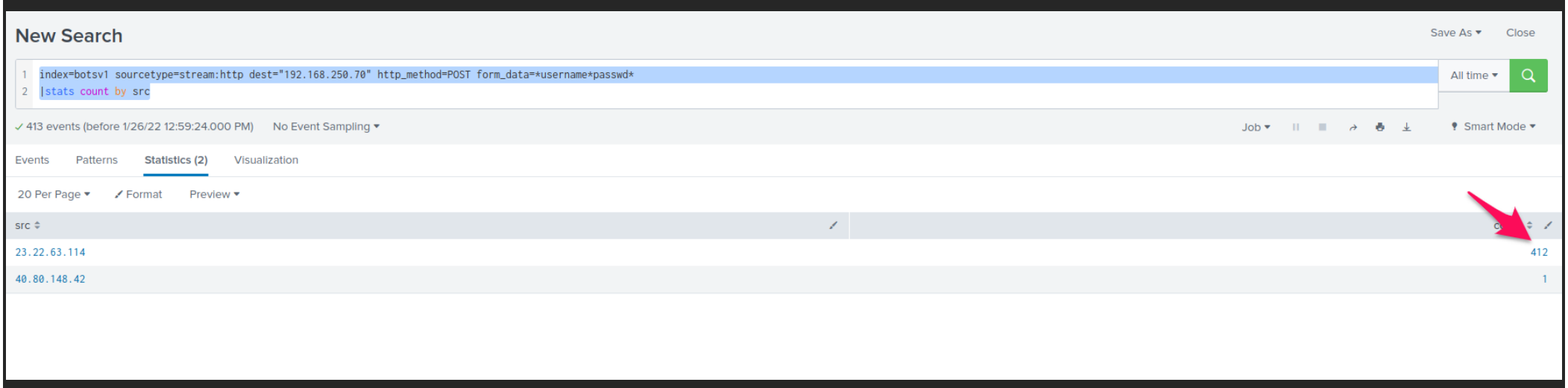
Now that we know form_data is the field to look at for brute force login attempts and that the string “*username*passwd*” is what we are looking for, we can use the stats command to count all the source addresses and provide a nice tabulation for us.

```
| stats count by src
```

index=botsv1 sourcetype=stream:http dest="192.168.250.70" http_method=POST form_data=*username*passwd*
|stats count by src

The stats command allows us to do computations grouped by one or more fields. We will talk about other ways to use stats in a bit but for now we just want a count grouped by the source address, or src field.

Based on our result, it appears that the source address of 23.22.63.114 is seen 412 times, so this is likely where the brute force attack is originating from. How many people picked 23.22.63.114?



Identifying the First Password Attempted in a Brute Force Attack

What was the first brute force password used?

Background

We know from our prior search that a string with wildcards between the values of username and passwd will return a username and password combination. If you also recall we used the table command to provide a tabular layout of that data. Because we are looking for the first password used, we will want to include the form_data and at least one additional field as well. Which field would that be?
_time

Kill Chain Phase

- Exploitation

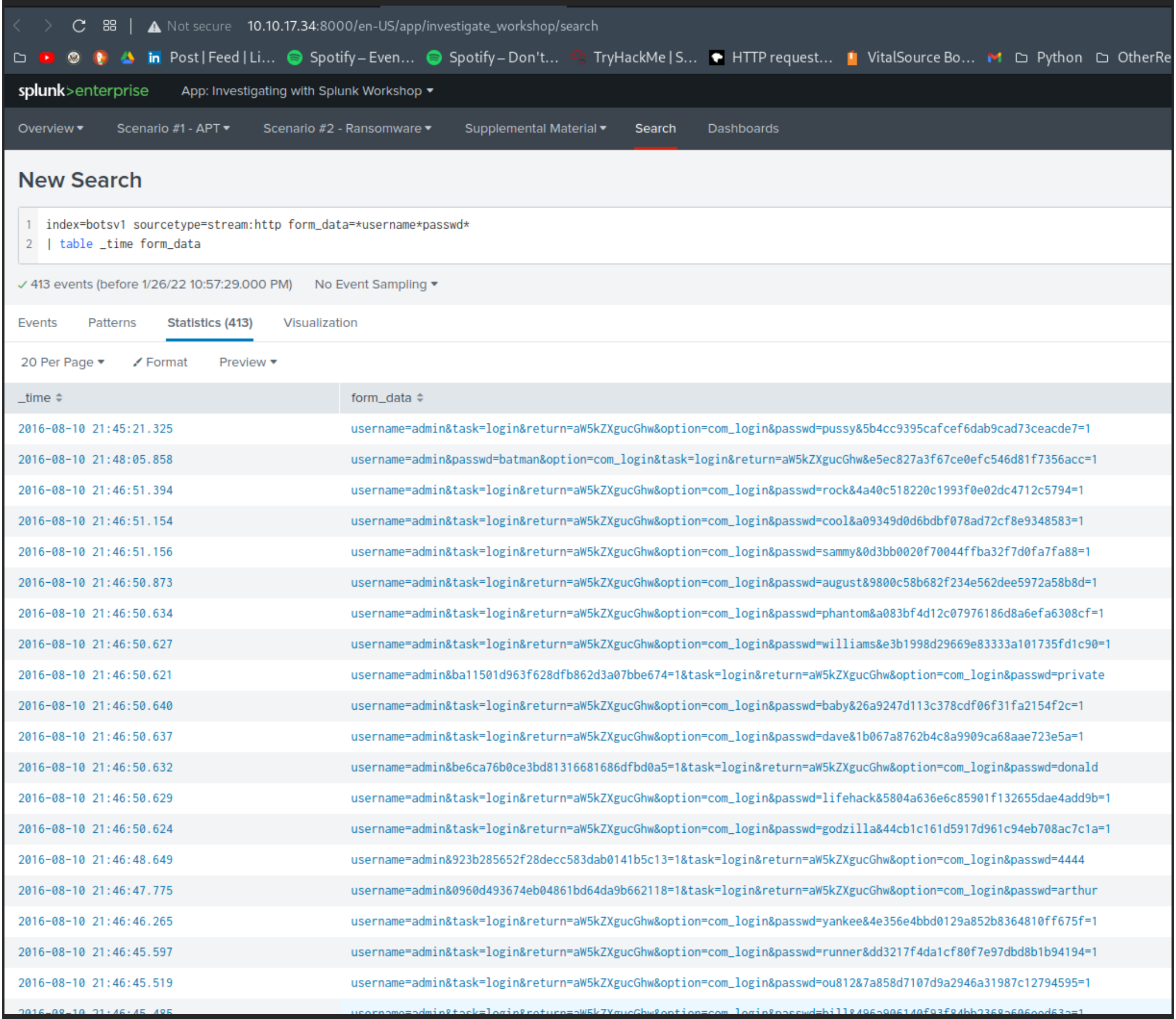
Sourcetypes

- stream:http

Adding Time

```
index=botsv1 sourcetype=stream:http form_data=*username*passwd* | table _time form_data
```

We have the full form_data strings here with the time. If we wanted to change the way this is displayed, what are some ways that we can do this?

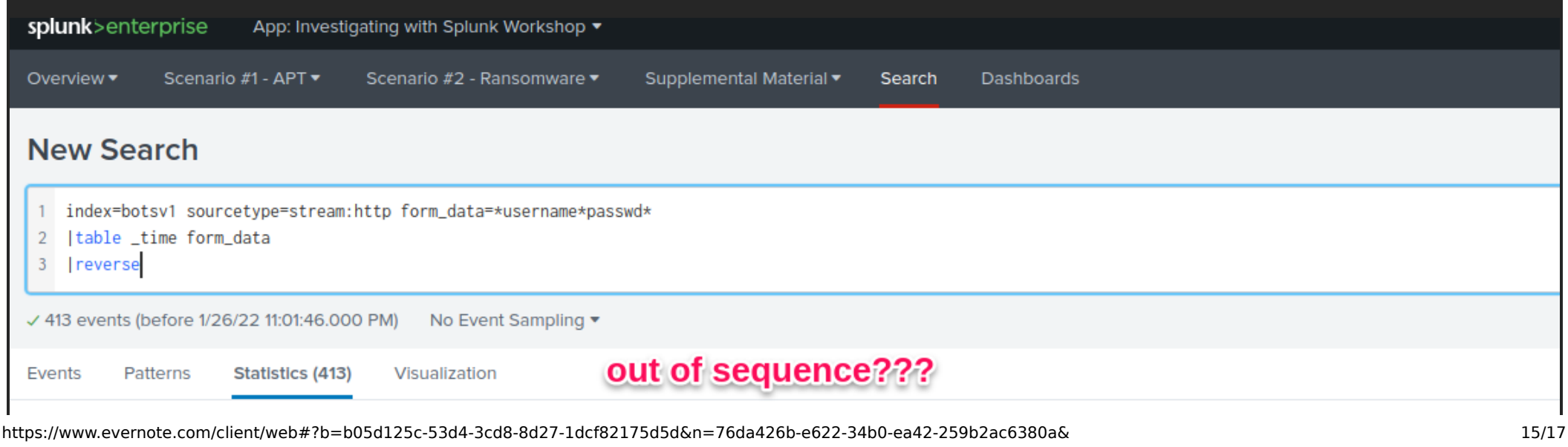


Reversing the Order of Output (Oldest First)

```
index=botsv1 sourcetype=stream:http form_data=*username*passwd* | table _time form_data
```

```
| reverse
```

The ***reverse*** command takes the output of our result set and flips the order of the results. In this case, our oldest results are now on top. This gives us the form_data for the first brute force password attempt and we can look within it to find the password string.



20 Per Page ▾ Format Preview ▾

_time	form_data
2016-08-10 21:45:21.226	username=admin&task=login&return=aW5kZXgucGhw&option=com_login&passwd=12345678&9d873c2becd118318849d13cf18b60ff=1
2016-08-10 21:45:21.247	username=admin&task=login&return=aW5kZXgucGhw&option=com_login&passwd=qwerty&af4df60674155567dee0566f87045251=1
2016-08-10 21:45:21.250	username=admin&task=login&return=aW5kZXgucGhw&option=com_login&passwd=1234&aaf6297ae5c1e3df78a421bc55548d16=1
2016-08-10 21:45:21.241	username=admin&863349a657c211fbfeb90ebe9427654c=1&task=login&return=aW5kZXgucGhw&option=com_login&passwd=letmein
2016-08-10 21:45:21.260	username=admin&task=login&return=aW5kZXgucGhw&option=com_login&76e93e8488d9a46878468d88954a0d54=1&passwd=123456
2016-08-10 21:45:21.263	username=admin&task=login&return=aW5kZXgucGhw&option=com_login&passwd=football&d1181413b1a70460b8d425cec799cdca=1
2016-08-10 21:45:21.826	username=admin&task=login&return=aW5kZXgucGhw&option=com_login&passwd=michael&bd5d773b68cdb015b021218ff36a2c4a=1
2016-08-10 21:45:22.000	username=admin&task=login&return=aW5kZXgucGhw&option=com_login&passwd=master&0a5d789b1af633e3de4b3dc95daa1877=1
2016-08-10 21:45:22.002	username=admin&task=login&return=aW5kZXgucGhw&option=com_login&passwd=superman&cab483edaec06d4e888547d33f57becb=1
2016-08-10 21:45:22.012	username=admin&task=login&return=aW5kZXgucGhw&option=com_login&passwd=dragon&45b663f3374634ca3fd860101177601c=1
2016-08-10 21:45:22.008	username=admin&task=login&return=aW5kZXgucGhw&option=com_login&passwd=shadow&5dd67f80e801dcd4f24920dae0fb2fd6=1
2016-08-10 21:45:22.005	username=admin&c77289d5d71cfee66a59479cb2706f4a=1&task=login&return=aW5kZXgucGhw&option=com_login&passwd=1234567
2016-08-10 21:45:22.034	username=admin&ad30d20bb0b8474d4a81199d95be395f=1&task=login&return=aW5kZXgucGhw&option=com_login&passwd=111111
2016-08-10 21:45:22.063	username=admin&462a0006d3b8398ba84d11967ceecec0=1&task=login&return=aW5kZXgucGhw&option=com_login&passwd=baseball
2016-08-10 21:45:22.570	username=admin&d9df6e24cc184413a74e1025ad274cd2=1&task=login&return=aW5kZXgucGhw&option=com_login&passwd=mustang
2016-08-10 21:45:22.710	username=admin&task=login&return=aW5kZXgucGhw&option=com_login&passwd=jennifer&ad1c15a0951d0ff17e2f4cfa4f4922f9=1
2016-08-10 21:45:22.874	username=admin&task=login&return=aW5kZXgucGhw&option=com_login&passwd=12345&af11d05632bad88f0985ed162bf0aad1=1
2016-08-10 21:45:22.911	username=admin&task=login&return=aW5kZXgucGhw&option=com_login&passwd=2000&204fafee060cec14a7b7daadf76e7174=1

A More Elegant Way To View the Passwords (Foreshadowing!!!)

```
index=botsv1 sourcetype=stream:http http_method=POST
```

```
| rex field=form_data "passwd=(?<userpassword>\w+)"
```

Using the ***rex*** command in Splunk provides a way, using regular expression syntax, to do an extraction of the password that creates a new field containing password string values that can be easily manipulated with Splunk.

```
| search userpassword=*
```

Returns only events where the new field userpassword exists and contains data

```
| reverse
```

Reverse the output returned (Oldest first)

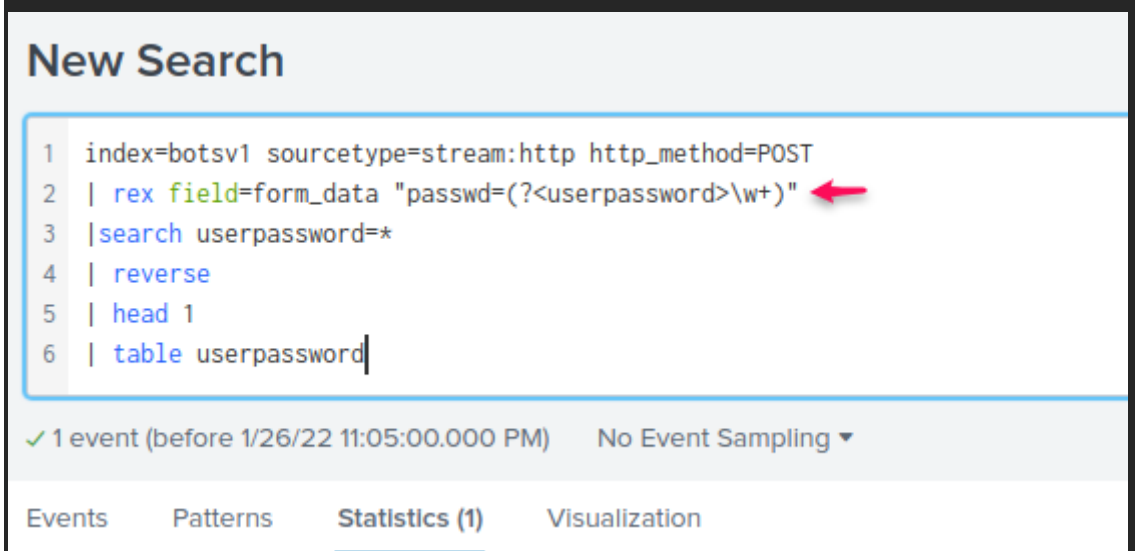
```
| head 1
```

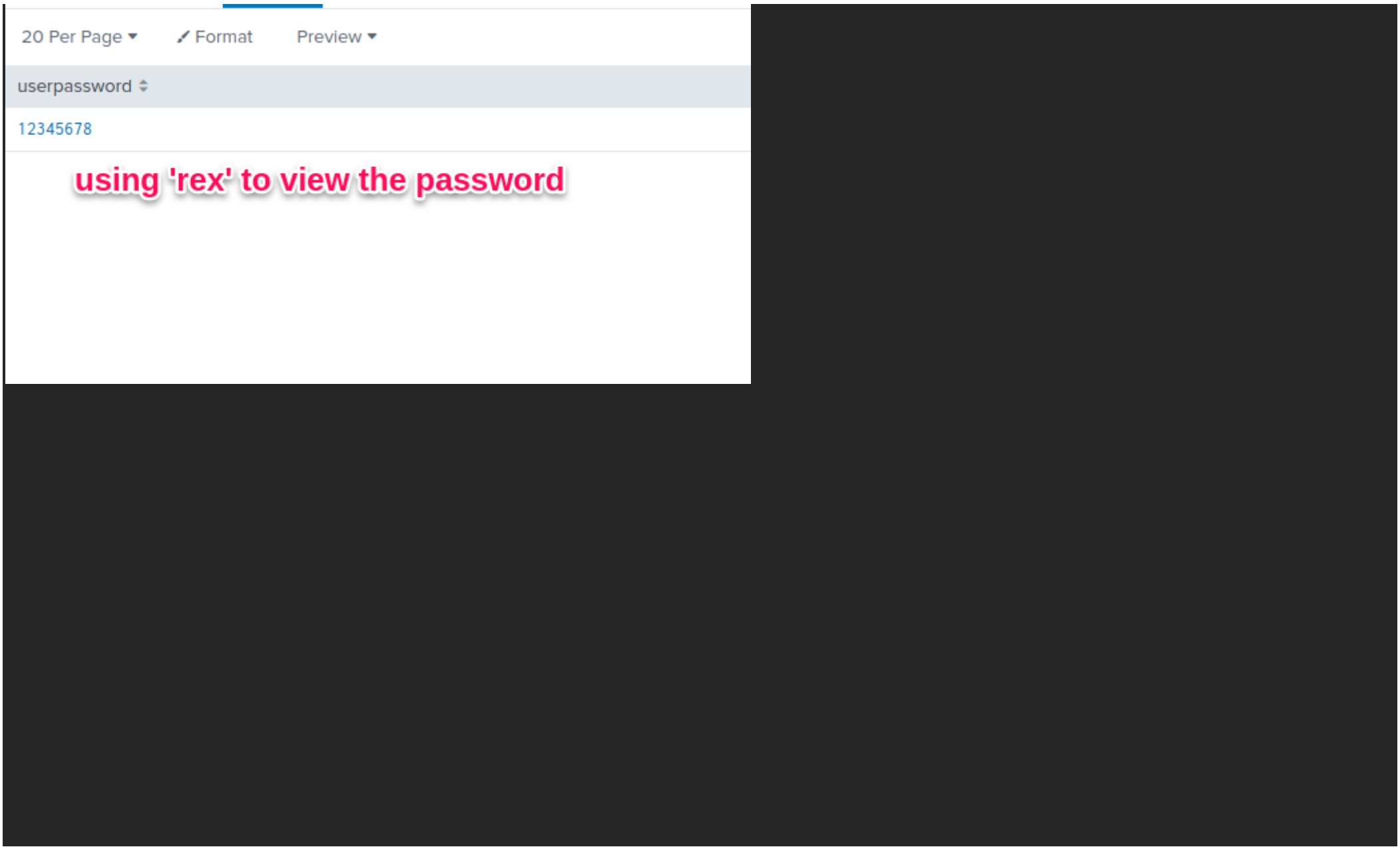
Return the first record in the data set

```
| table userpassword
```

Output just the userpassword field

Let’s wrap this one up and then look at another example where extracting that password will become important.




20 Per Page
Format
Preview
userpassword
12345678
using 'rex' to view the password