Investigating Brute-Force Attack – Splunk

What is Brute-Force Attack?

A brute-force attack attempts to gain unauthorized access by systematically trying a large number of possible passwords or usernames. In a typical scenario, an attacker repeatedly attempts to log in to a system using different password combinations for a single account or multiple accounts over a short period.

Dataset

Download the BOTSV3 dataset using the following link:

Link: https://github.com/splunk/botsv3

What is Botsv3?

Boss of the SOC (BOTS) Dataset Version 3. A sample security dataset and CTF platform for information security professionals, researchers, etc.

Note:

- BotsV3 dataset does not have MUCH brute force attacking in the traditional sense.
- There are only 3 failed logins in the entire dataset

EVENTS:

Event code 4624 = successful login
Event code 4625 = unsuccessful login

METHOD 1: Failed Logins

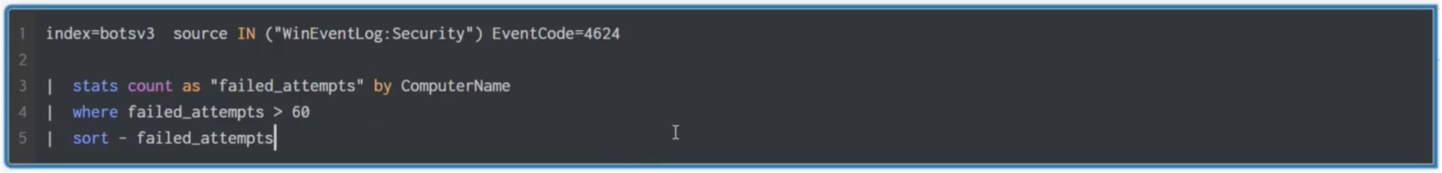
The following queries will identify failed logins.

```
Index=botsv3 source IN (“WinEventLogSecurity”) EventCode=4625

| stats count as “failed_attempts” by ComputerName
| where failed_attempts > 10
| sort – failed_attempts
```

Where;

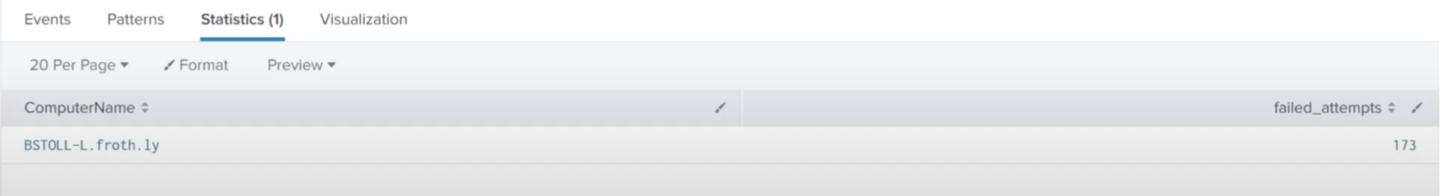
ComputerName = is the log of the machine that someone attempted to login to.
I am setting up an arbitrary threshold, meaning, if the failed_attempts is larger than 10 I would want to know about it.



```
index=botsv3  source IN ("WinEventLog:Security") EventCode=4624

|  stats count as "failed_attempts" by ComputerName
|  where failed_attempts > 60
|  sort - failed_attempts
```

Here;

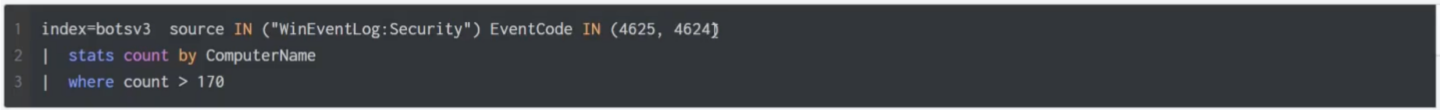


ComputerName	failed_attempts
BSTOLL-L.froth.ly	173

BSTOLL-L.froth.ly appears to be the device potentially being brute force attacked.

METHOD 2: High Authentication Attempts

The following query will identify ComputerName that make unusually high number of authentication attempts both successful & failed within a specific time frame.

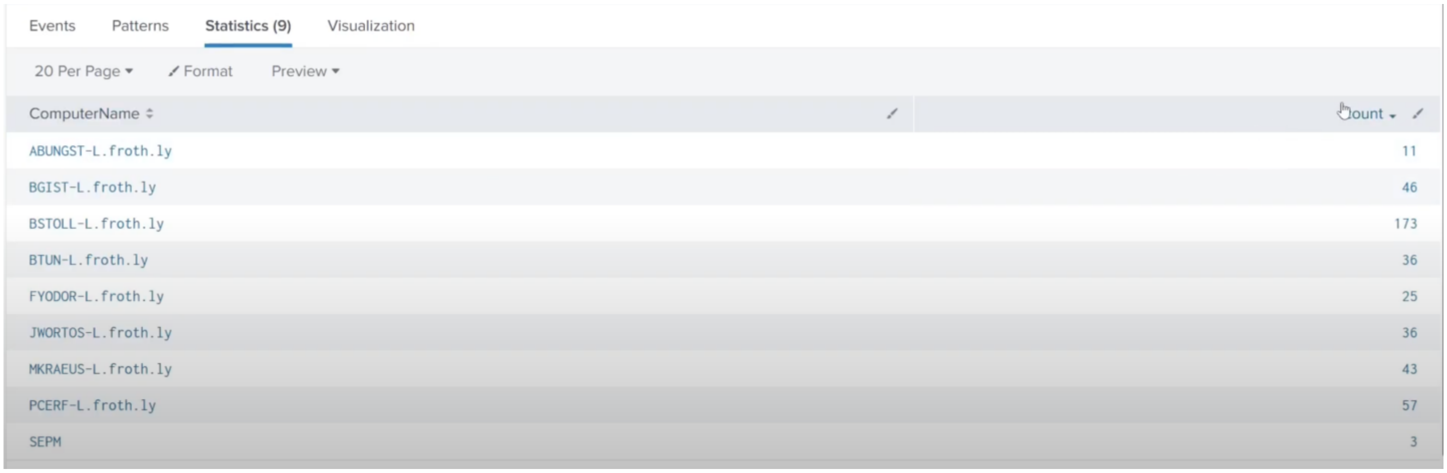


```
index=botsv3  source IN ("WinEventLog:Security") EventCode IN (4625, 4624)
| stats count by ComputerName
| where count > 170
```

Here;

I am using both 4624 & 4625 because I want to know all authentication attempts.

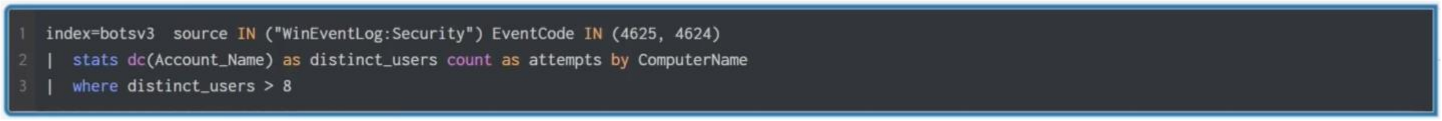
You can or cannot set your threshold to 170 based off what your traffic looks like in your environment.



ComputerName	count
ABUNGST-L.froth.ly	11
BGIST-L.froth.ly	46
BSTOLL-L.froth.ly	173
BTUN-L.froth.ly	36
FYODOR-L.froth.ly	25
JWORTOS-L.froth.ly	36
MKRAEUS-L.froth.ly	43
PCERF-L.froth.ly	57
SEPM	3

METHOD 3: Different Users

This query detects computers attempting to login as ‘different users’ which is another common brute force attack technique.

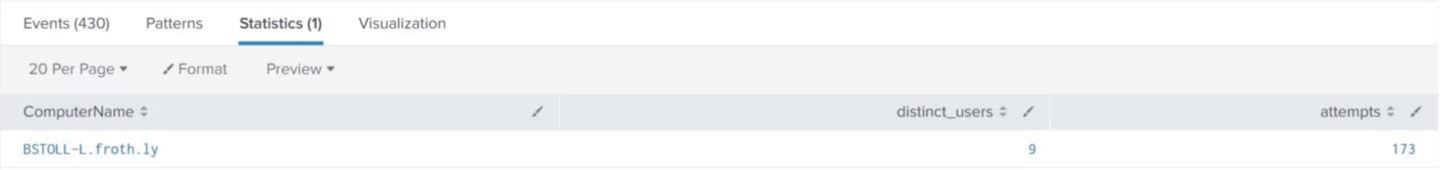


```
index=botsv3  source IN ("WinEventLog:Security") EventCode IN (4625, 4624)
| stats dc(Account_Name) as distinct_users count as attempts by ComputerName
| where distinct_users > 8
```

Here;

Account_Name from “WinEventLog:Security” .. so we have distinct ‘users’ & ‘CompterName’

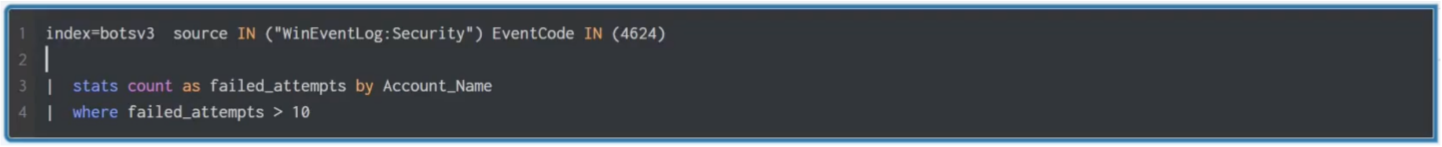
dc = distinct counts meaning every unique account name
we are including both 4624 & 4625 both to identify both positive & negative authentications.



ComputerName	distinct_users	attempts
BSTOLL-L.froth.ly	9	173

METHOD 4: Number of Failed Attempts

We will look at a user account and how many times it fails.

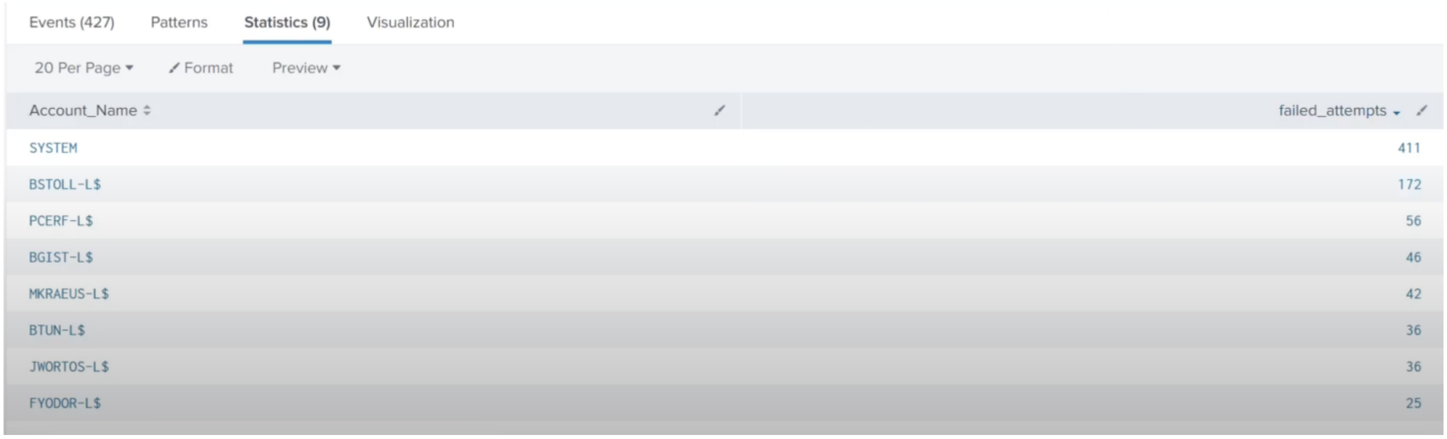


```
index=botsv3  source IN ("WinEventLog:Security") EventCode IN (4624)

| stats count as failed_attempts by Account_Name
| where failed_attempts > 10
```

Where;

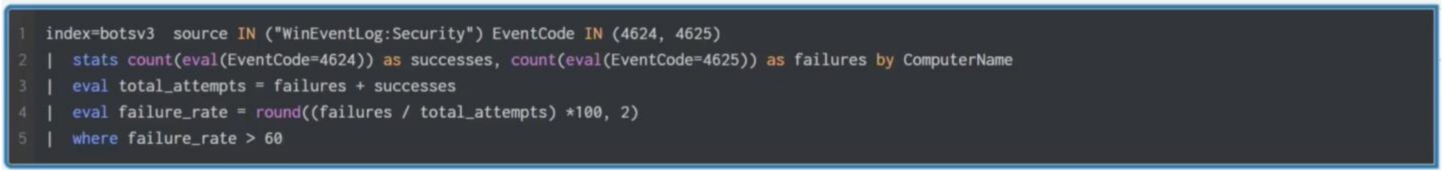
We will be using Account_Name instead of ComputerName



Account_Name	failed_attempts
SYSTEM	411
BSTOLL-L$	172
PCERF-L$	56
BGIST-L$	46
MKRAEUS-L$	42
BTUN-L$	36
JWORTOS-L$	36
FYODOR-L$	25

METHOD 5: Ratio Analysis

This is the failed login ratio analysis. The following query calculates the ratio of ‘Failed’ to ‘Total login attempts’ by ComputerName with unusually high failure rates.
We will do a ‘stats count’



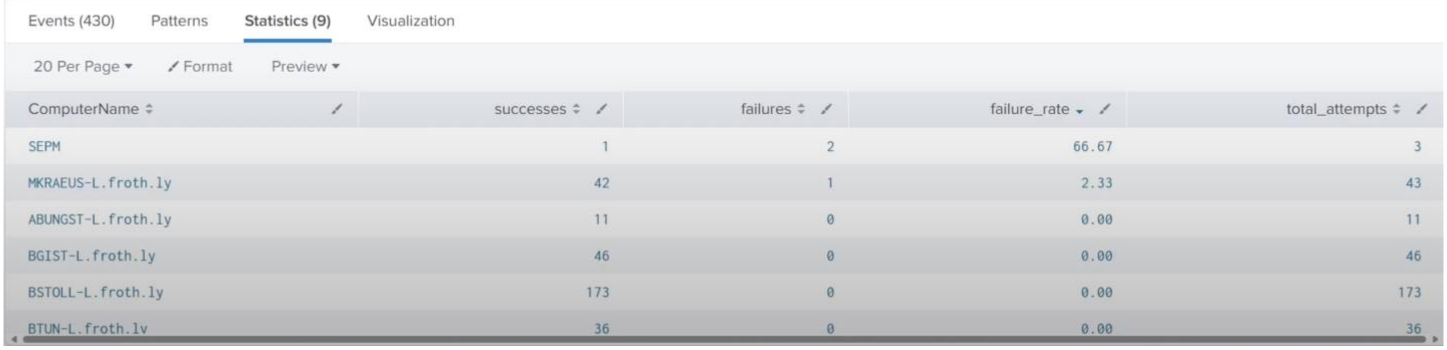
```
index=botsv3  source IN ("WinEventLog:Security") EventCode IN (4624, 4625)
| stats count(eval(EventCode=4624)) as successes, count(eval(EventCode=4625)) as failures by ComputerName
| eval total_attempts = failures + successes
| eval failure_rate = round((failures / total_attempts) *100, 2)
| where failure_rate > 60
```

Stats count (eval (EventCode = 4624)) as success .. meaning … Every time I see 4624, count that and call it successes.

And then count 4625 as failures and count by ComputerName

Then we will count total_attempts which will be failure + success.

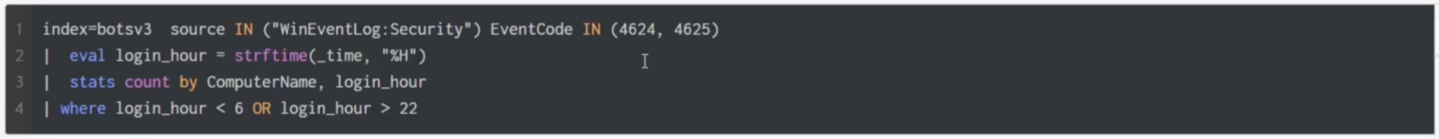
Failure rate will be rounded …. Failures divided by total attempts which will be result in a decimal number, so we will multiply with 100 to get us a percentage and we round it to 2 (meaning I want 2 decimal places)



ComputerName	successes	failures	failure_rate	total_attempts
SEPM	1	2	66.67	3
MKRAEUS-L.froth.ly	42	1	2.33	43
ABUNGST-L.froth.ly	11	0	0.00	11
BGIST-L.froth.ly	46	0	0.00	46
BSTOLL-L.froth.ly	173	0	0.00	173
BTUN-L.froth.ly	36	0	0.00	36

METHOD 6: Login Time Analysis

We will look at logins during unusual hours and we can also put thresholds on it.

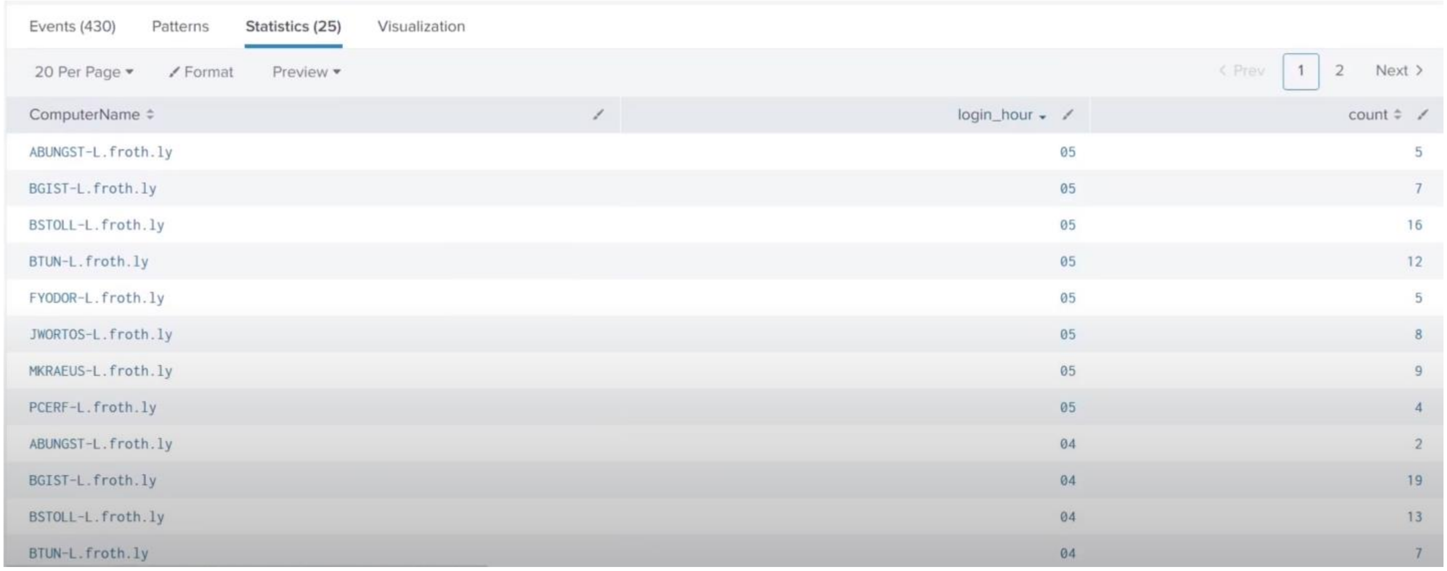


```
index=botsv3  source IN ("WinEventLog:Security") EventCode IN (4624, 4625)
|  eval login_hour = strftime(_time, "%H")
|  stats count by ComputerName, login_hour
| where login_hour < 6 OR login_hour > 22
```

We are evaluating login_hour using ‘striftime’ and we provide it time and percentage. That will give us hour field from the epoch time.

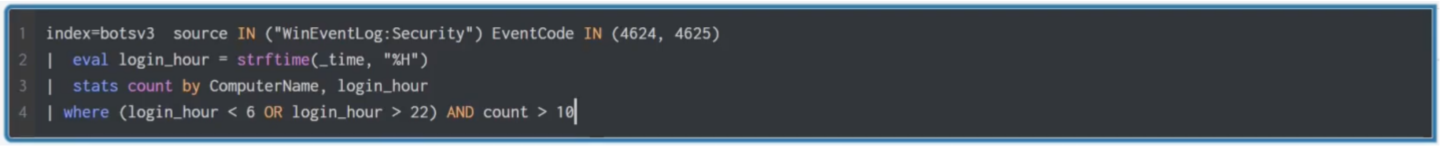
Then we do stat count by ComputerName and login_hour

Where login_hour is before 6 am or after 10 pm. Which will be the unusual login hours.



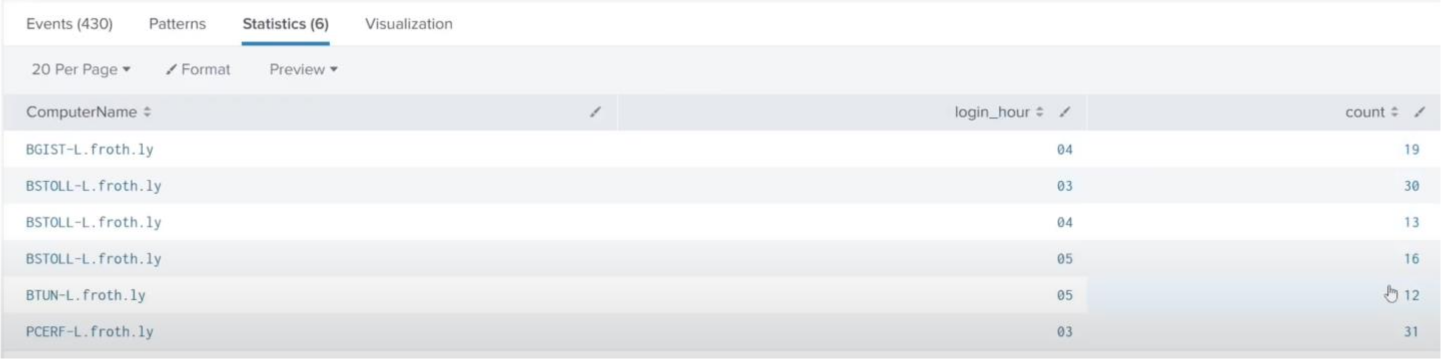
ComputerName	login_hour	count
ABUNGST-L.froth.ly	05	5
BGIST-L.froth.ly	05	7
BSTOLL-L.froth.ly	05	16
BTUN-L.froth.ly	05	12
FYODOR-L.froth.ly	05	5
JWORTOS-L.froth.ly	05	8
MKRAEUS-L.froth.ly	05	9
PCERF-L.froth.ly	05	4
ABUNGST-L.froth.ly	04	2
BGIST-L.froth.ly	04	19
BSTOLL-L.froth.ly	04	13
BTUN-L.froth.ly	04	7

If we add ‘Count’ greater than 10



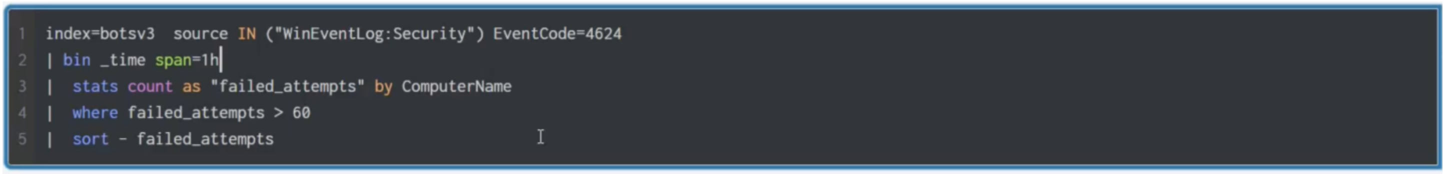
```
index=botsv3  source IN ("WinEventLog:Security") EventCode IN (4624, 4625)
|  eval login_hour = strftime(_time, "%H")
|  stats count by ComputerName, login_hour
| where (login_hour < 6 OR login_hour > 22) AND count > 10
```

then it will give us unusual hours and ‘multiple events’ which is another method of doing Brute Force attacks.



ComputerName	login_hour	count
BGIST-L.froth.ly	04	19
BSTOLL-L.froth.ly	03	30
BSTOLL-L.froth.ly	04	13
BSTOLL-L.froth.ly	05	16
BTUN-L.froth.ly	05	12
PCERF-L.froth.ly	03	31

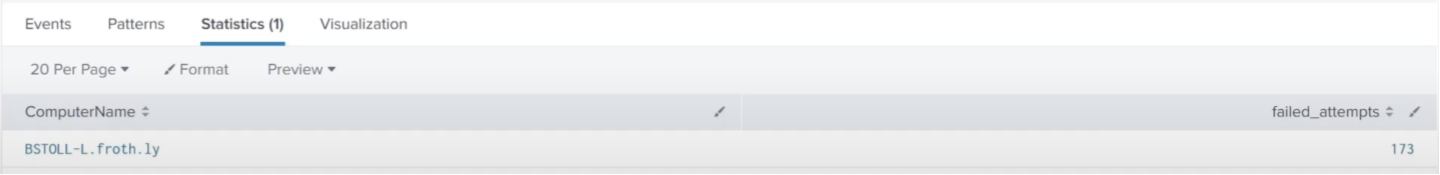
These search commands are used all the time and you might want to bucketize them (bin them)



```
1  index=botsv3  source IN ("WinEventLog:Security") EventCode=4624
2  | bin _time span=1h
3  |  stats count as "failed_attempts" by ComputerName
4  |  where failed_attempts > 60
5  |  sort - failed_attempts
```

We add the line 2 where we included bin_time equals span 1 hour. That will break up your time into the buckets of 1 hour.

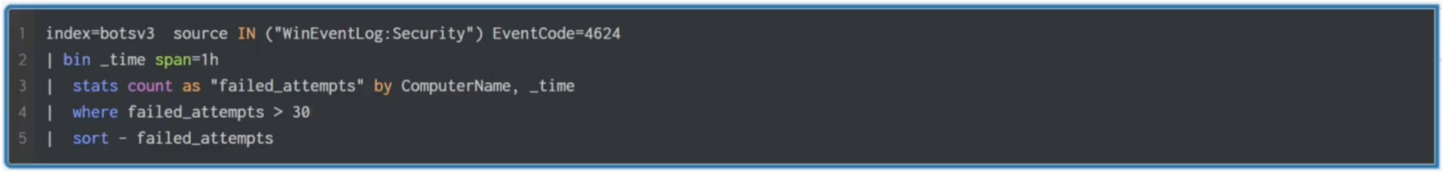
This will show us chunks (not overall time) because Brute Force is typically lots of authentications in a short amount of time.



ComputerName	failed_attempts
BSTOLL-L.froth.ly	173

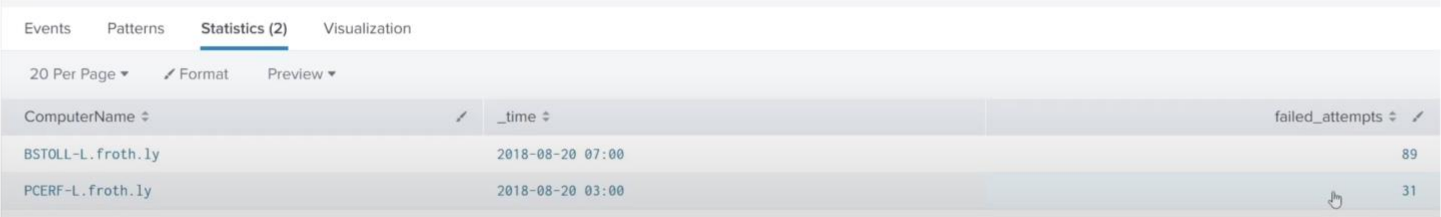
Here, since our dataset is small the output is just 1

When we break it down by ‘time’ in line 3 shown as follows



```
1  index=botsv3  source IN ("WinEventLog:Security") EventCode=4624
2  | bin _time span=1h
3  |  stats count as "failed_attempts" by ComputerName, _time
4  |  where failed_attempts > 30
5  |  sort - failed_attempts
```

We will get a lot of smaller buckets.



ComputerName	_time	failed_attempts
BSTOLL-L.froth.ly	2018-08-20 07:00	89
PCERF-L.froth.ly	2018-08-20 03:00	31

Here we have 89 attempts in 7:00 hour for BSTOLL and 31 attempts for PCERF.

CHEERS !!