**Kibana Activity File Continued**

The purpose of this activity was to explore more Kibana capabilities by completing three tasks:

1. Generating a high amount of failed SSH login attempts and verify that Kibana is picking up this activity.
2. Generating a high amount of CPU usage on the pen-testing machines and verify that Kibana picks up this data.
3. Generating a high amount of web requests to your pen-testing servers and make sure that Kibana is picking them up.

**Task #1** **SSH Barrage**

**Goal:** In this task we were asked to generate a high amount of failed SSH login attempts and verify that Kibana is picking up this activity.

**Steps:** While this task could be performed by manually attempting to establish an SSH connection multiple times, there is a more efficient way to accomplish this task – use a command:

watch –n .1 ssh redadmin@10.0.0.5



**Syntax**

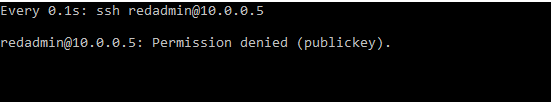
**watch: execute a program periodically**

**-n: the option specifying the time interval**

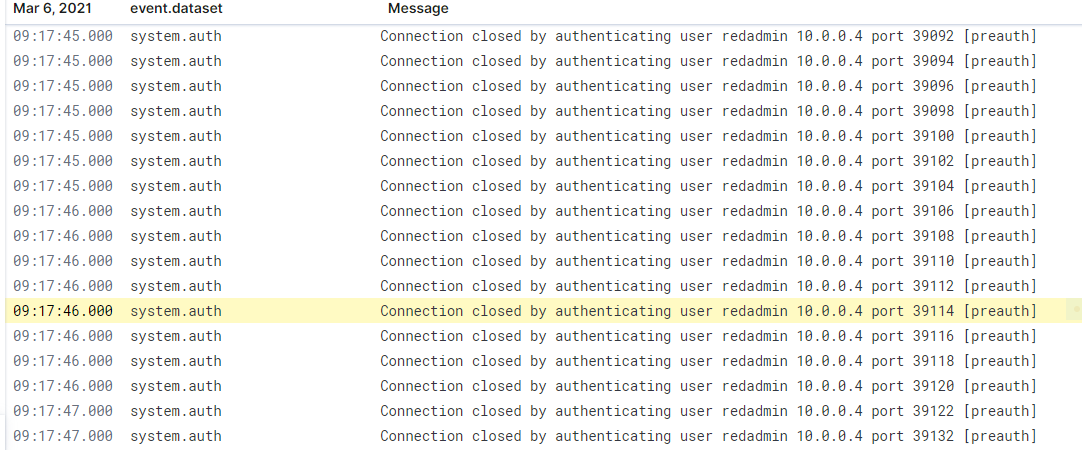
**.1: the time interval; here we are using 0.1 sec.**

**ssh redadmin@10.0.0.5 : the program we are executing periodically; this particular IP is the private IP of Web-1**

After executing this command a new window opens up, similar to the image below.



This program will continue run every 0.1 sec. until it is manually stopped with Ctl+C. After generating enough failed log in attempts, we can check Kibana for the results as seen below.



**Summary**

Writing a simple command saved much time and energy than having to manually generate failed log in attempts by repeatedly running ssh [redadmin@10.0.0.5](mailto:redadmin@10.0.0.5). More importantly, Kibana collected all of our failed log in attempts which can be used for further analysis. Since this was done intentionally, no foul play was involved. However, in a real life scenario, these results have all of the signs of a brute force attack.

**Task #2 Linux Stress**

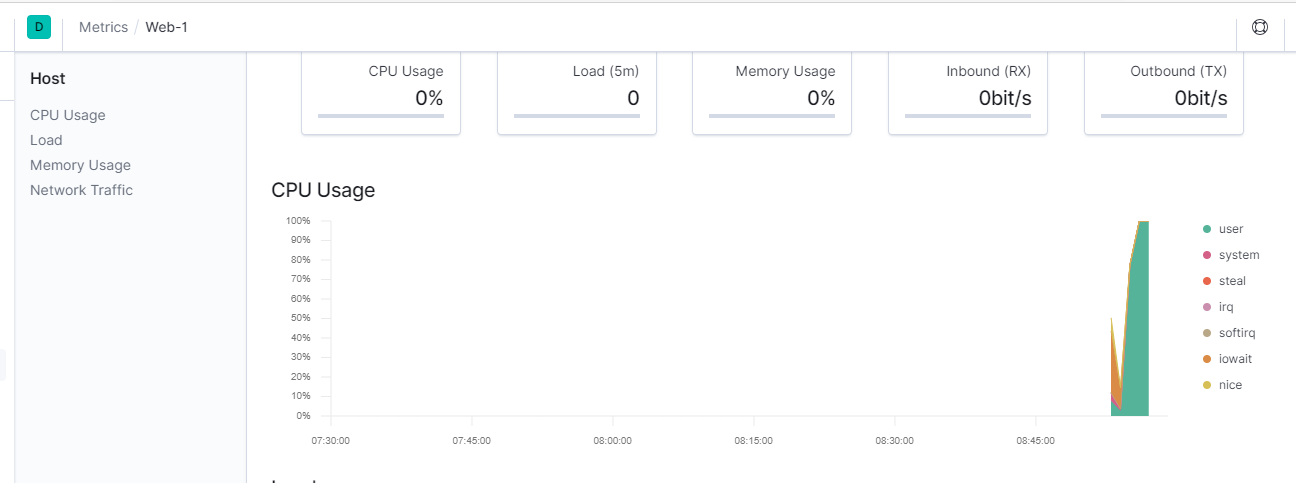
**Goal:** Generate a high amount of CPU usage on the pentesting machines and verify that Kibana picks up this data

**Steps:**

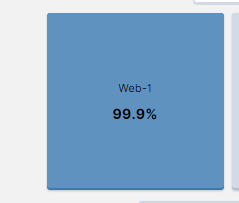
1. From the jump box, start up your Ansible container and attach to it (sudo attach stupefied\_chaum).
2. SSH from the Ansible container to Web-1, Web-2, and Web-3 (ssh redadmin@10.0.0.5, ssh redadmin@10.0.0.6, ssh redadmin@10.0.0.7).
3. Run sudo apt install stress to install the stress program on each VM.
4. Run sudo stress --cpu 1 and allow stress to run for a few minutes.
5. View the Metrics page for that VM in Kibana.
6. Run the stress program on all three of your VMs and take screenshots of the data generated on the Metrics page of Kibana.

**Note:** The stress program will run until you quit with Ctrl+C

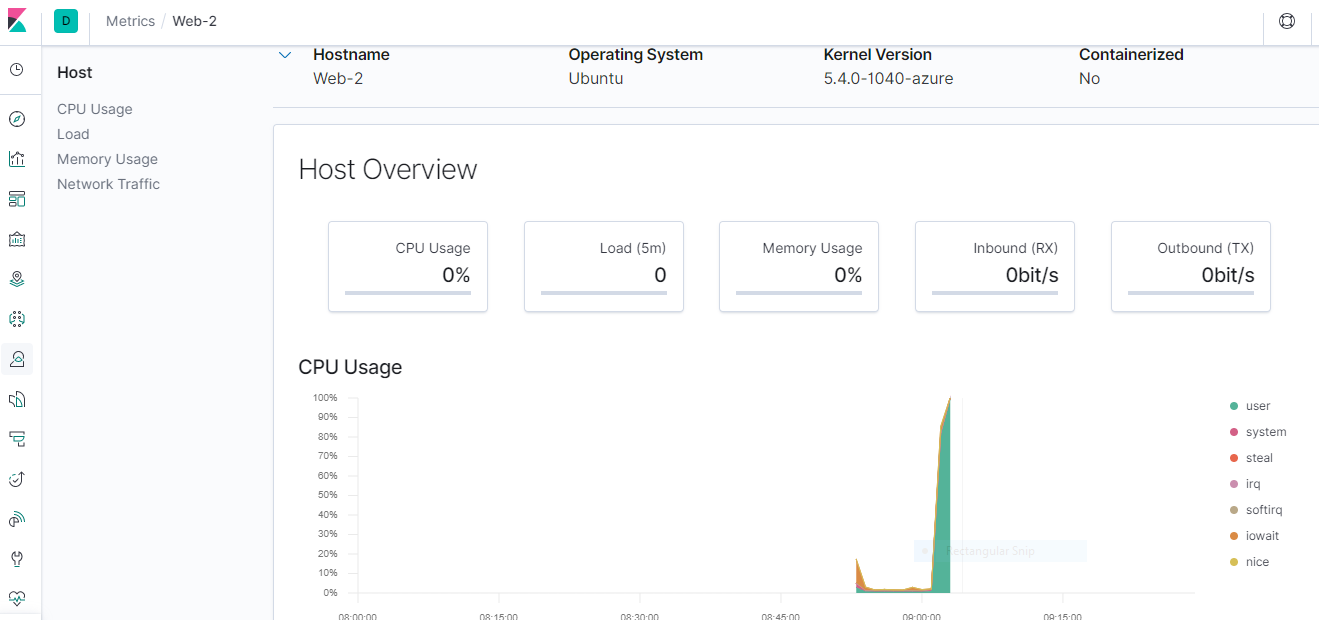
**Web-1 Results**

**** Greenrepresents activity from the user.  ****

Notice how there was no activity until the stress command generated a huge spike close to 100% CPU usage.

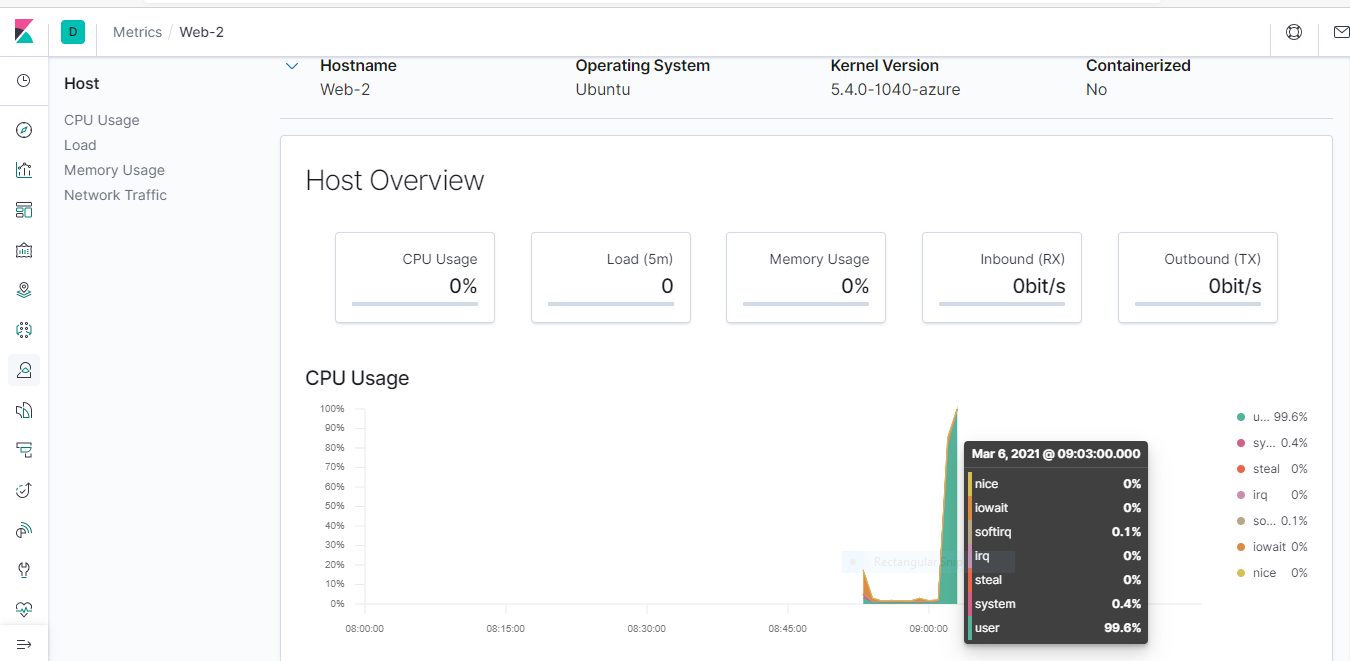


**Web-2 Results**

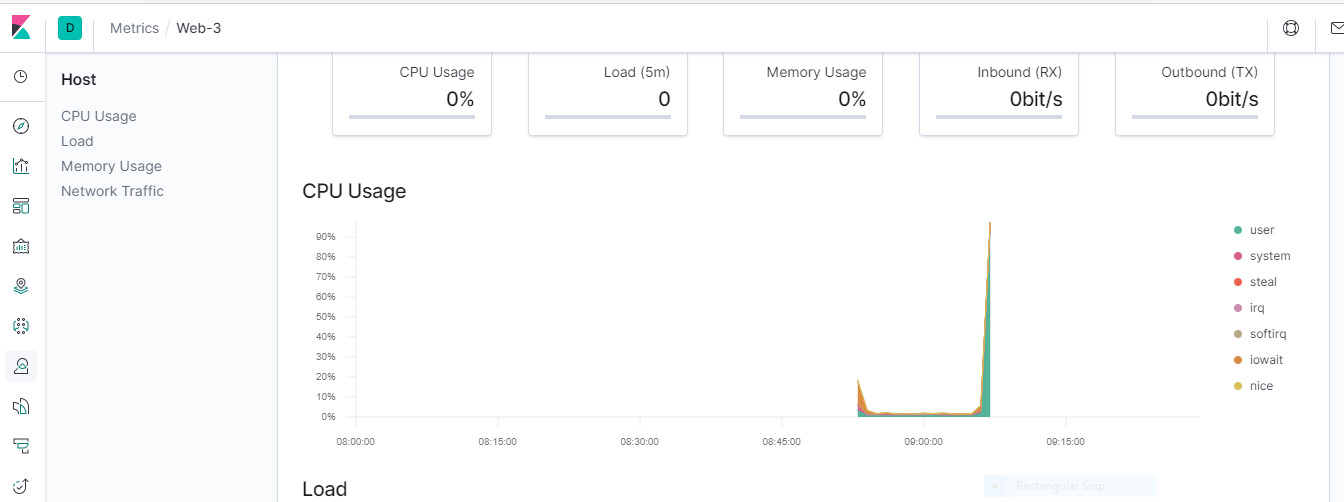


Greenrepresents activity from the user. ****

Notice how there was very little activity until the stress command generated a huge spike close to 100% CPU usage.

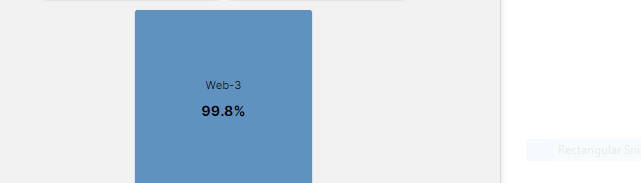


**Web-3 Results**

****

Green represents activity from the user ****

Notice how there was very little activity untilt he stress command ran and genered a huge spike. Like the other two machines, the CPU usage almost reach full capacity, 100%.



**Summary**

All three machines had little to no activity until the stress command ran. When the stress command ran, it triggered a huge spike in CPU usage, close to 100% in each machine. Since this was done intentionally, no foul play was involved. However, in a real life scenario, these results have all of the signs of an attack aimed to hog up all of machine’s resources in an effort to take it down.

**Task #3 wget-DoS**

**Goal**: Generate a high amount of web requests to your pen-testing servers and make sure that Kibana is picking them up.

**Steps:** The command-line program wget can easily generate abnormal data to view by creating a DoS web attack.

wget will download a file from any web server. Use man pages for more info on wget.

1. Log into the jump box.
2. Run:

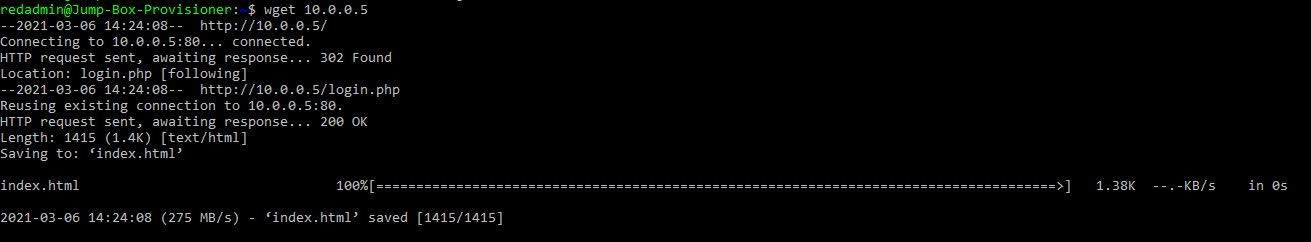
wget 10.0.0.5 for Web-1

Wget 10.0.0.6 for Web-2

or

Wget 10.0.0.7 for Web-3

In this situation one machine was attacked, Web-1 (10.0.0.5).



1. Run ls to view the file you downloaded from your web VM to your jump box. The result is seen in the image below.



1. Run the wget command in a loop to generate many web requests: watch –n .1 wget 10.0.0.5

C:\Users\Owner\Desktop\snapshots\wgetcommand.PNG

Note: This command is similar to the one used in task 1.

**Syntax**

**watch: execute a program periodically**

**-n: the option specifying the time interval**

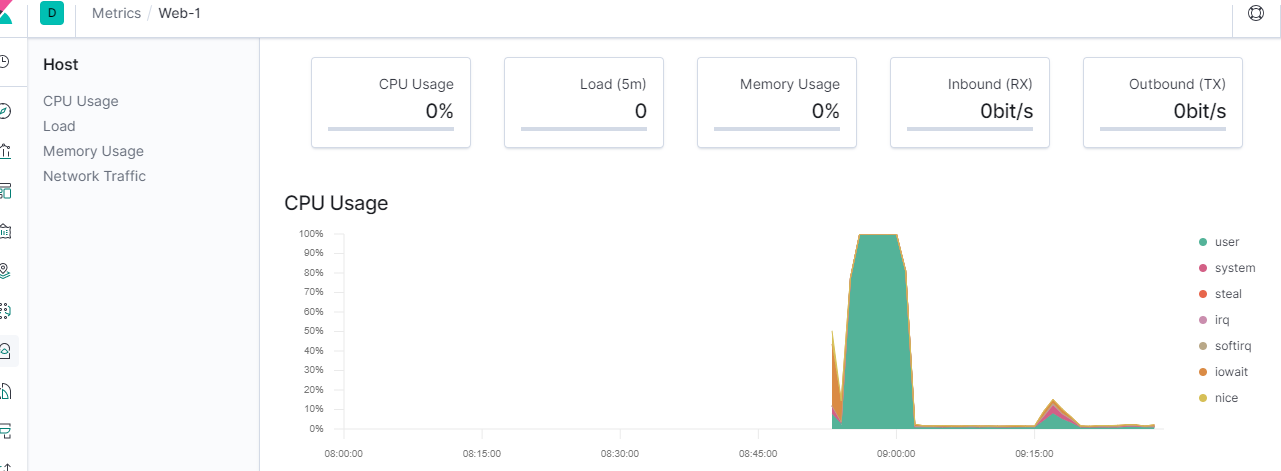
**.1: the time interval; here we are using 0.1 sec.**

**ssh redadmin@10.0.0.5 : the program we are executing periodically; this particular IP is the private IP of Web-1 .**

**Task 3 Results**

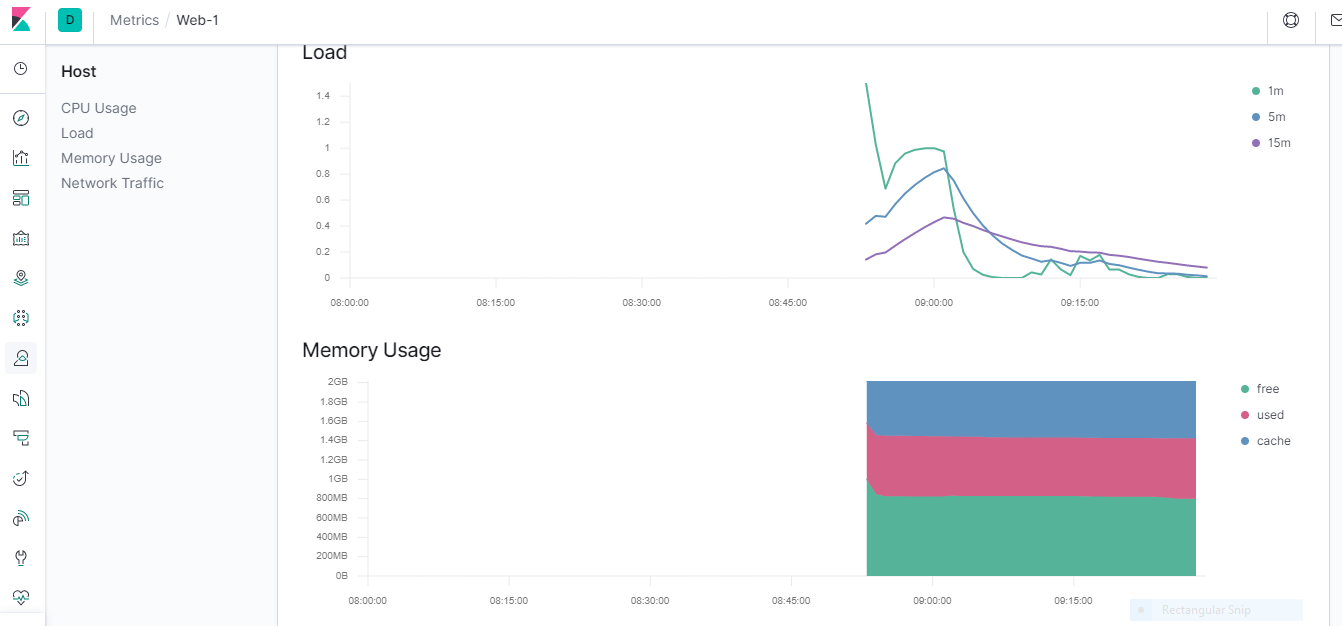
****

Notice the huge spike in network traffic, both inbound and outbound.

****

Note**:** This section of the graph is result of running wget ****

The gigantic spike to the left of this one is the result of task 1 and has no association with task 3. This particular graph does not look abnormal or display any warning signals.



This chart displaying memory usage looks common and has no warning signals.

**Summary**

After running wget on machine Web-1 Kibana demonstrated a huge increase in network traffic. This is exactly what expected since we performed a DDos attack and a DDos attack works by generating an overwhelming number of requests to take down a server.

**Activity Summary**

These purpose of these activities enabled us to see how our machines respond when it is attacked under different real life situations and also how Kibana responds by collecting and displaying those results, which are then left open for human interpretation and analysis.