Instructions

1. Add the sample web log data to Kibana.
2. Answer the following questions:
    - In the last 7 days, how many unique visitors were located in India? 224 unique visitors
    - In the last 24 hours, of the visitors from China, how many were using Mac OSX? 12 visitors using Mac OSX
    - In the last 2 days, what percentage of visitors received 404 errors? How about 503 errors? 3.333% of users received 404 errors in the last 2 days. 0% received 503 errors.
    - In the last 7 days, what country produced the majority of the traffic on the website? China (CN) produced the majority of traffic on the website in the last 7 days
    - Of the traffic that's coming from that country, what time of day had the highest amount of activity? 10:00 AM
    - List all the types of downloaded files that have been identified for the last 7 days, along with a short description of each file type (use Google if you aren't sure about a particular file type).

      -GZ File – compressed archives created by gzip compression algorithm used in Linux systems.

      -CSS File – cascading style sheet language used to interpret documentation written in markup language such as HTML.

      -Zip File – archive file format that takes up less storage space compared to uncompressed data files

      -Deb File – format and extension for software packages from Debian Linux distribution

      -RPM File – Used to store installation packages on Linux Oss within the Red Hat Package Manager file.

3. Now that you have a feel for the data, Let's dive a bit deeper. Look at the chart that shows Unique Visitors Vs. Average Bytes.
    - Locate the time frame in the last 7 days with the most amount of bytes (activity). The most amount of bytes occurred at 10/30/2021 with 8,856 bytes.
    - In your own words, is there anything that seems potentially strange about this activity? This is unusual as only one visitor accounted for all the bytes at this time.
4. Filter the data by this event.
    - What is the timestamp for this event? 10/30/2021 at 20:00
    - What kind of file was downloaded? Zip File
    - From what country did this activity originate? This activity originated in Brazil.

    - What HTTP response codes were encountered by this visitor? This visitor encountered a 200 HTTP response code.
5. Switch to the Kibana Discover page to see more details about this activity.
    - What is the source IP address of this activity? 17.111.163.53
    - What are the geo coordinates of this activity? Lat: 42.59157139 Lon: -114.7967178
    - What OS was the source machine running? Windows 7
    - What is the full URL that was accessed? https://artifacts.elastic.co/downloads/kibana/kibana-6.3.2-windows-x86_64.zip
    - From what website did the visitor's traffic originate? http://twitter.com/success/mark-kelly

6. Finish your investigation with a short overview of your insights.
    - What do you think the user was doing? The user was attempting to download a kibana-6.3.2-windows-x86_64.zip file.
    - Was the file they downloaded malicious? If not, what is the file used for? Based on the fact that the URL https://artifacts.elastic.co/downloads does not exist it would seem as if the downloaded .zip file is malicious.
    - Is there anything that seems suspicious about this activity? It seems as if the referrer page http://twitter.com/success/mark-kelly provided a false link to download Kibana.
    - Is any of the traffic you inspected potentially outside of compliance guidlines?

      The referrer is used as an optional HTTP header field that indicates the URL that is linked to the resource being requested. In this case we have a request originated in Brazil using a referrer Twitter page linked to a U.S. senator/astronaut. It seems out of compliance for the Twitter page of the U.S. Senator to contain a link that would download a Kibana zip file.

Activity File: Kibana Continued

- This week, you created the infrastructure behind a security information and event management system such as Kibana. Once that set up is complete, you will have finished the project.
- This optional activity tasks you with exploring more Kibana capabilities, some of which you will use in future projects.
- **Note**: In order to complete these activities, you will need to complete the optional Metricbeat configuration.

Scenario

In this activity, you will suppose the role of a cloud architect that has been tasked with setting up an ELK server to gather logs for the Incident Response team.

Before you hand over the server to the IR team, your senior architect has asked that you verify the ELK server is working as expected and pulling both logs and metrics from the pen-testing web servers.

You will have three tasks:

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

These activities will guide you though generating some data to visualize in Kibana. Each of these activity will require the following high level steps:

1. Use your jump-box to attack your web machines in various ways.
2. Use a Linux utility to stress the system of a webVM directly.
3. Subsequently generate traffic and logs that Kibana will collect.
4. View that traffic in various ways inside Kibanna.

It's also worth noting that these activities comprise different job roles:

- Getting the infrastructure setup and maintaining it is the role of a security engineer or cloud architect.
- Using that infrastructure by creating dashboards and alerts fall under the security analyst role. It would be rare to have a position where you would be required to do both.

That said, now that we have Kibana setup and gathering data from three web servers, its worth learning how to visualize data in Kibana.

Before getting started, we'll have to complete some metrics and logs set up.

Setup: Kibana Metrics and Logs Orientation

Before we begin generating traffic, locate the two screens inside Kibana that you will use to visualize this traffic:

- Logs
- Metrics

These pages will show you the changes in data that we will create.

Logs

- Click **Logs** to see some general system logs coming from the web machines.

- Notice that you can stream logs live from the machines.

Metrics

- Next, click **Metrics** on the left side.
    - Here we can see each of our VMs that are sending metrics.
- Click on one of the large squares that represent one of your VMs.
- Choose **View metrics** from the dropdown that appears.

- Notice that you can see CPU and memory usage here.

Now that we know where to look for this data, let's generate some unusual network traffic.

Activity Tasks

Expand the provided activity files to complete each task. These tasks can be completed in any order.

SSH Barrage

Task: Generate a high amount of failed SSH login attempts and verify that Kibana is picking up this activity.

Activity File: SSH Barrage

Scenario

- You are a cloud architect that has been tasked with setting up an ELK server to gather logs for the Incident Response team to use for training.
- Before you hand over the server to the IR team, your senior architect has asked you to verify the ELK server is working as expected and pulling both logs and metrics from the pentesting web servers.

Your Task: Generate a high amount of failed SSH login attempts and verify that Kibana is picking up this activity.

Instructions

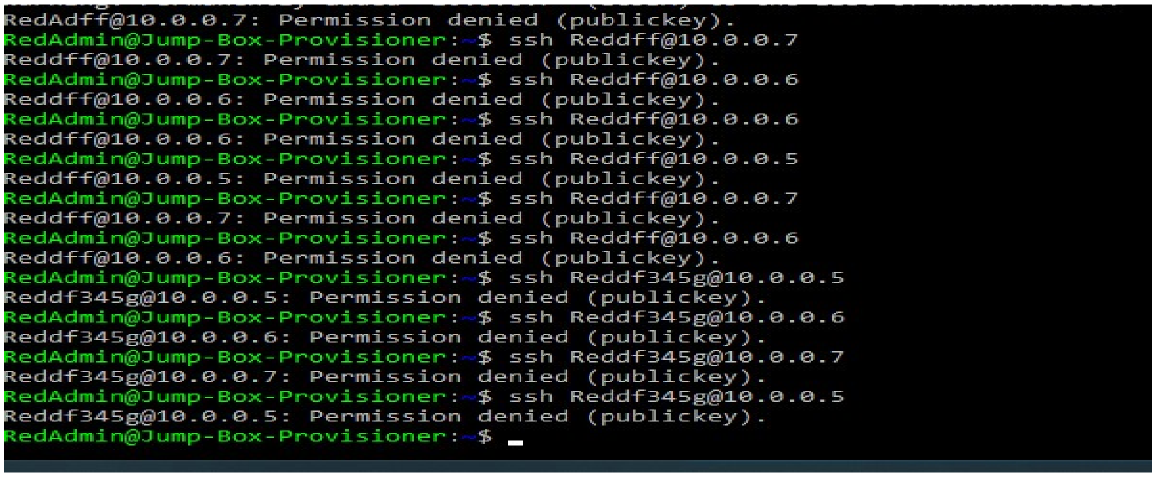
One way we can generate logs of interest is to create some failed SSH logins on our servers.

- The only environment that holds our SSH keys is our Ansible container. Attempting to create an SSH connection from any other environment will trigger a log entry.
- We can also create a log entry by attempting to log in with the wrong username.
- Note: A successful SSH login also creates a log entry, but here we will focus on failed logins.

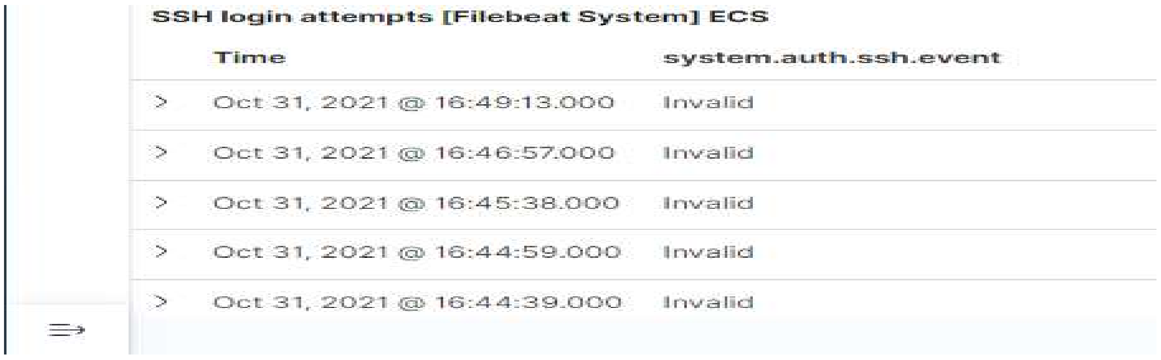
We can easily do this by trying to SSH to a web machine from our jump box directly without using the Ansible container.

1. Start by logging into your jump-box.
   - Run: `ssh username@ip.of.web.vm`
   - You should receive an error:

   -
     ```
     sysadmin@Jump-Box-Provisioner:~$ ssh sysadmin@10.0.0.5
     sysadmin@10.0.0.5: Permission denied (publickey).
     ```
   - This error was also logged and sent to Kibana.
2. Run the failed SSH command in a loop to generate failed login log entries.
   - You can use a bash `for` or `while` loop, directly on the command line, to repeatedly run the SSH command.


```
RedAdff@10.0.0.7: Permission denied (publickey).
RedAdmin@Jump-Box-Provisioner:~$ ssh Reddff@10.0.0.7
Reddff@10.0.0.7: Permission denied (publickey).
RedAdmin@Jump-Box-Provisioner:~$ ssh Reddff@10.0.0.6
Reddff@10.0.0.6: Permission denied (publickey).
RedAdmin@Jump-Box-Provisioner:~$ ssh Reddff@10.0.0.6
Reddff@10.0.0.6: Permission denied (publickey).
RedAdmin@Jump-Box-Provisioner:~$ ssh Reddff@10.0.0.5
Reddff@10.0.0.5: Permission denied (publickey).
RedAdmin@Jump-Box-Provisioner:~$ ssh Reddff@10.0.0.7
Reddff@10.0.0.7: Permission denied (publickey).
RedAdmin@Jump-Box-Provisioner:~$ ssh Reddff@10.0.0.6
Reddff@10.0.0.6: Permission denied (publickey).
RedAdmin@Jump-Box-Provisioner:~$ ssh Reddf345g@10.0.0.5
Reddf345g@10.0.0.5: Permission denied (publickey).
RedAdmin@Jump-Box-Provisioner:~$ ssh Reddf345g@10.0.0.6
Reddf345g@10.0.0.6: Permission denied (publickey).
RedAdmin@Jump-Box-Provisioner:~$ ssh Reddf345g@10.0.0.7
Reddf345g@10.0.0.7: Permission denied (publickey).
RedAdmin@Jump-Box-Provisioner:~$ ssh Reddf345g@10.0.0.5
Reddf345g@10.0.0.5: Permission denied (publickey).
RedAdmin@Jump-Box-Provisioner:~$
```

3. Search through the logs in Kibana to locate your generated failed login attempts.


SSH login attempts [Filebeat System] ECS

	Time	system.auth.ssh.event
>	Oct 31, 2021 @ 16:49:13.000	Invalid
>	Oct 31, 2021 @ 16:46:57.000	Invalid
>	Oct 31, 2021 @ 16:45:38.000	Invalid
>	Oct 31, 2021 @ 16:44:59.000	Invalid
>	Oct 31, 2021 @ 16:44:39.000	Invalid

Bonus: Create a nested loop that generates SSH login attempts across all three of your VM's.

Linux Stress

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

Activity File: Linux Stress

Scenario

- You are a cloud architect that has been tasked with setting up an ELK server to gather logs for the Incident Response team to use for training.
- Before you hand over the server to the IR team, your senior architect has asked that you verify the ELK server is working as expected and pulling both logs and metrics from the pen-testing web servers.

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

Notes

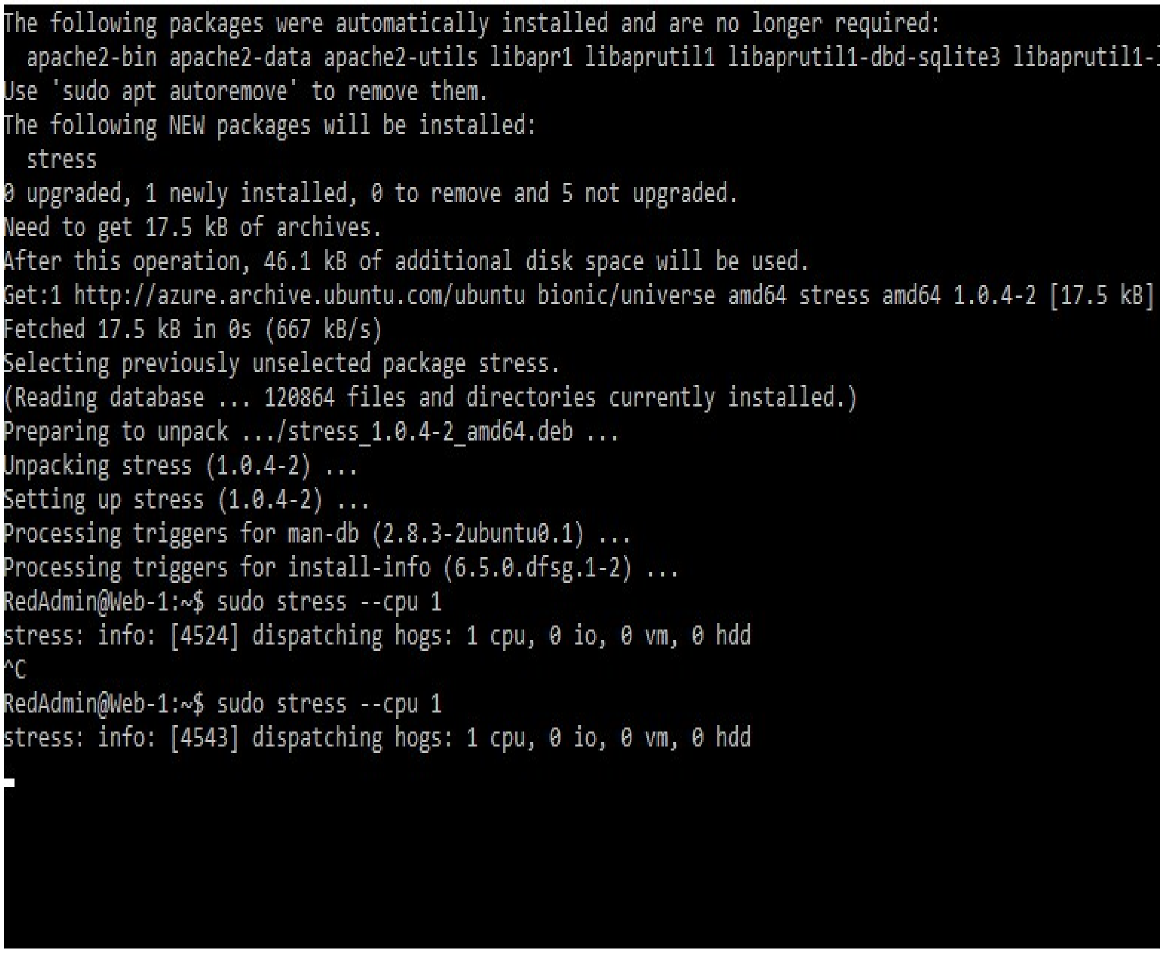
The Metrics page for a single VM shows the CPU usage for that machine. This shows how much work the machine is doing. Excessively high CPU usage is typically a cause for concern, as overworked computers are at greater risk for failure.

- Metricbeat forwards data about CPU load to Elasticsearch, which can be visualized with Kibana.
- In this activity, you will intentionally stress the CPU of one of your VMs, then find evidence of the increased activity in Kibana.

Linux has a common, easy-to-use diagnostic program called `stress`. It is easy to use and can be downloaded via `apt`.

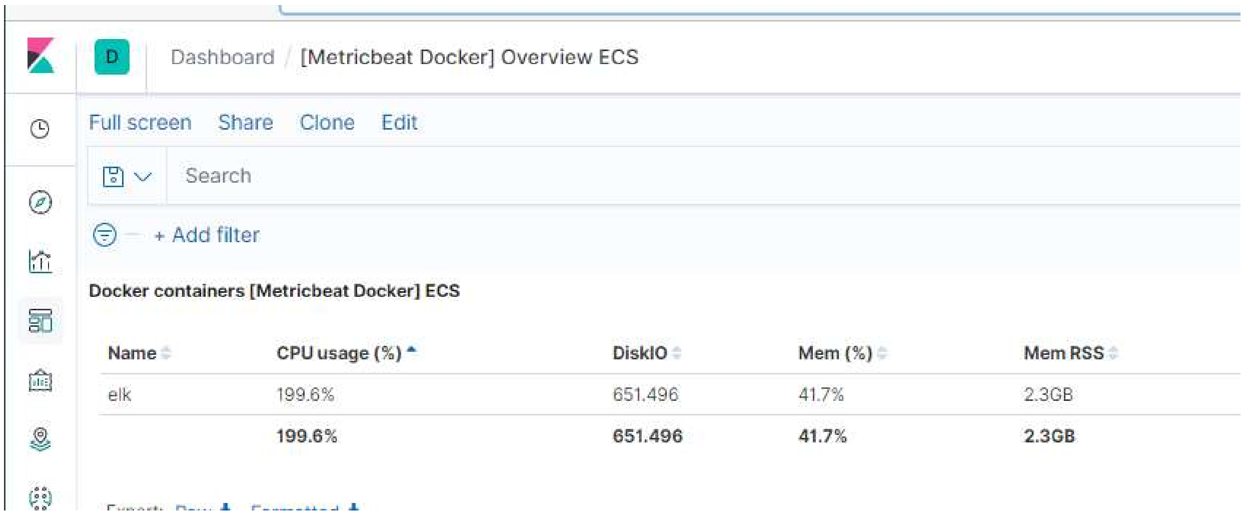
Instructions

1. From your jump box, start up your Ansible container and attach to it.
2. SSH from your Ansible container to one of your WebVM's.
3. Run `sudo apt install stress` to install the stress program.
4. Run `sudo stress --cpu 1` and allow `stress` to run for a few minutes.



```
The following packages were automatically installed and are no longer required:
  apache2-bin apache2-data apache2-utils libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-l
Use 'sudo apt autoremove' to remove them.
The following NEW packages will be installed:
  stress
0 upgraded, 1 newly installed, 0 to remove and 5 not upgraded.
Need to get 17.5 kB of archives.
After this operation, 46.1 kB of additional disk space will be used.
Get:1 http://azure.archive.ubuntu.com/ubuntu bionic/universe amd64 stress amd64 1.0.4-2 [17.5 kB]
Fetched 17.5 kB in 0s (667 kB/s)
Selecting previously unselected package stress.
(Reading database ... 120864 files and directories currently installed.)
Preparing to unpack .../stress_1.0.4-2_amd64.deb ...
Unpacking stress (1.0.4-2) ...
Setting up stress (1.0.4-2) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for install-info (6.5.0.dfsg.1-2) ...
RedAdmin@Web-1:~$ sudo stress --cpu 1
stress: info: [4524] dispatching hogs: 1 cpu, 0 io, 0 vm, 0 hdd
^C
RedAdmin@Web-1:~$ sudo stress --cpu 1
stress: info: [4543] dispatching hogs: 1 cpu, 0 io, 0 vm, 0 hdd
```

5. View the Metrics page for that VM in Kibana. What indicates that CPU usage increased?
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.



Docker containers [Metricbeat Docker] ECS

Name	CPU usage (%)	DiskIO	Mem (%)	Mem RSS
elk	199.6%	651.496	41.7%	2.3GB
	199.6%	**651.496**	**41.7%**	**2.3GB**

wget-DoS

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

Activity File: wget-DoS

Scenario

- You are a cloud architect that has been tasked with setting up an ELK server to gather logs for the Incident Response team to use for training.
- Before you hand over the server to the IR team, your senior architect has asked that you verify the ELK server is working as expected and pulling both logs and metrics from the pen-testing web servers.

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

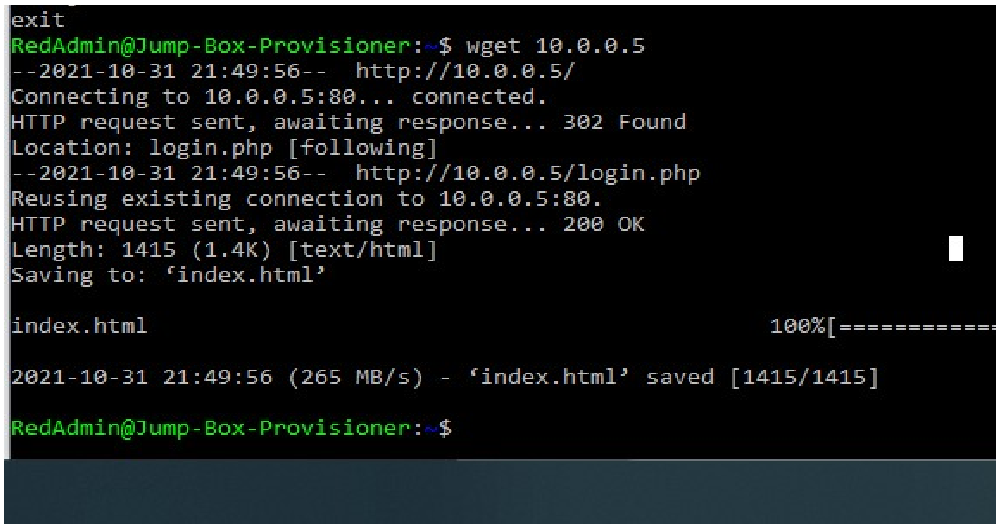
Instructions

The Metrics section for a single VM will show Load and Network Traffic data.

We can generate abnormal data to view by creating a DoS web attack. The command-line program `wget` can do this easily.

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

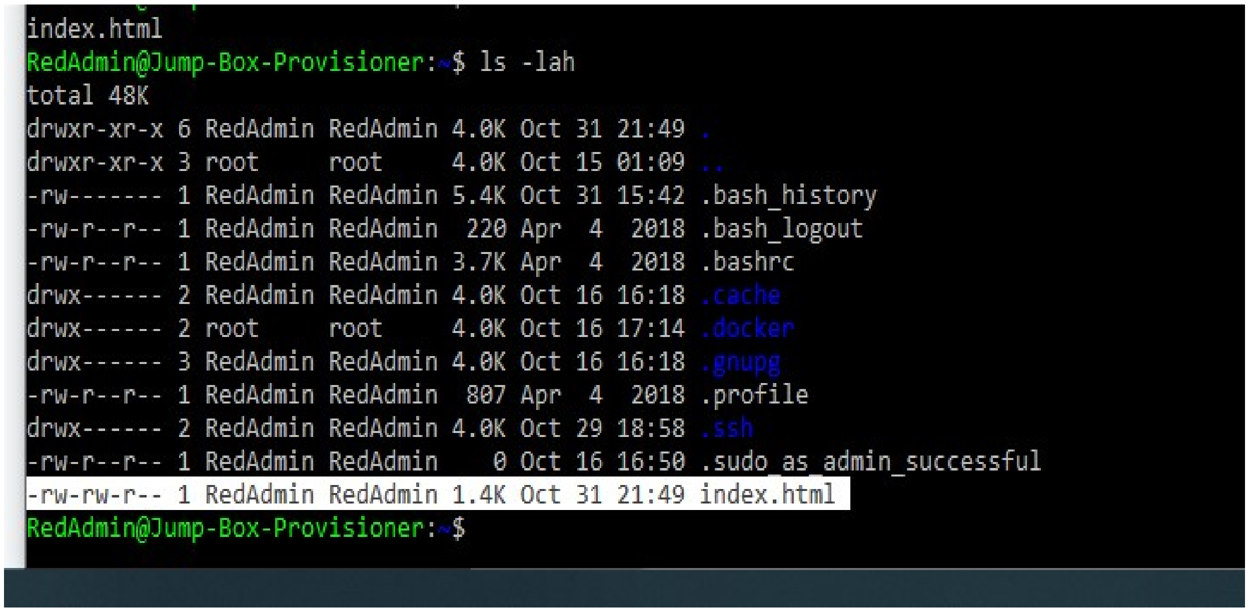
1. Log into your jump box.
2. Run `wget ip.of.web.vm`.



```
exit
RedAdmin@Jump-Box-Provisioner:~$ wget 10.0.0.5
--2021-10-31 21:49:56--  http://10.0.0.5/
Connecting to 10.0.0.5:80... connected.
HTTP request sent, awaiting response... 302 Found
Location: login.php [following]
--2021-10-31 21:49:56--  http://10.0.0.5/login.php
Reusing existing connection to 10.0.0.5:80.
HTTP request sent, awaiting response... 200 OK
Length: 1415 (1.4K) [text/html]
Saving to: ‘index.html’

index.html                                   100%[===========

2021-10-31 21:49:56 (265 MB/s) - ‘index.html’ saved [1415/1415]

RedAdmin@Jump-Box-Provisioner:~$
```

```
3. sysadmin@Jump-Box-Provisioner:~$ wget 10.0.0.5
4. --2020-05-08 15:44:00--  http://10.0.0.5/
5. Connecting to 10.0.0.5:80... connected.
6. HTTP request sent, awaiting response... 302 Found
7. Location: login.php [following]
8. --2020-05-08 15:44:00--  http://10.0.0.5/login.php
9. Reusing existing connection to 10.0.0.5:80.
10.  HTTP request sent, awaiting response... 200 OK
11.  Length: 1523 (1.5K) [text/html]
12.  Saving to: ‘index.html’
13.
14.  index.html            100%[=======================>]   1.49K  --.-KB/s
   in 0s
15.
   2020-05-08 15:44:00 (179 MB/s) - ‘index.html’ saved [1523/1523]
```

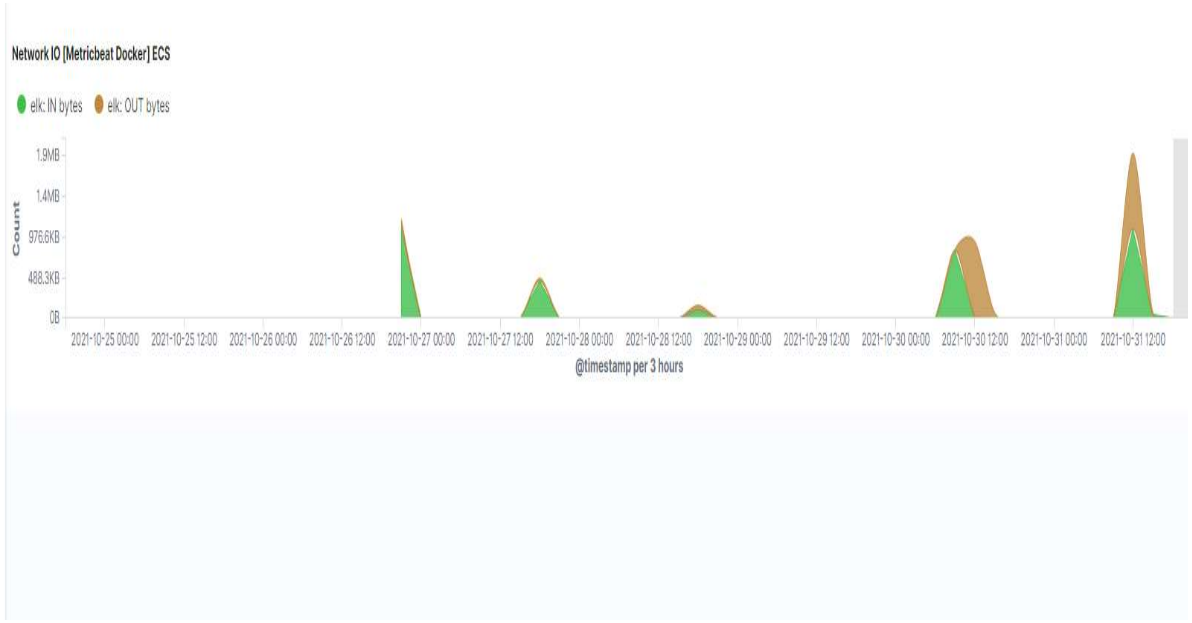
16. Run `ls` to view the file you downloaded from your web VM to your jump box.



```
index.html
RedAdmin@Jump-Box-Provisioner:~$ ls -lah
total 48K
drwxr-xr-x 6 RedAdmin RedAdmin 4.0K Oct 31 21:49 .
drwxr-xr-x 3 root     root     4.0K Oct 15 01:09 ..
-rw------- 1 RedAdmin RedAdmin 5.4K Oct 31 15:42 .bash_history
-rw-r--r-- 1 RedAdmin RedAdmin  220 Apr  4  2018 .bash_logout
-rw-r--r-- 1 RedAdmin RedAdmin 3.7K Apr  4  2018 .bashrc
drwx------ 2 RedAdmin RedAdmin 4.0K Oct 16 16:18 .cache
drwx------ 2 root     root     4.0K Oct 16 17:14 .docker
drwx------ 3 RedAdmin RedAdmin 4.0K Oct 16 16:18 .gnupg
-rw-r--r-- 1 RedAdmin RedAdmin  807 Apr  4  2018 .profile
drwx------ 2 RedAdmin RedAdmin 4.0K Oct 29 18:58 .ssh
-rw-r--r-- 1 RedAdmin RedAdmin    0 Oct 16 16:50 .sudo_as_admin_successful
-rw-rw-r-- 1 RedAdmin RedAdmin 1.4K Oct 31 21:49 index.html
RedAdmin@Jump-Box-Provisioner:~$
```

17.
```
sysadmin@Jump-Box-Provisioner:~$ ls
index.html
```

18. Run the `wget` command in a loop to generate many web requests.
    - You can use a bash `for` or `while` loop, directly on the command line, just as you did with the SSH command.
19. Open the Metrics page for the web machine you attacked and answer the following questions:
    - Which of the VM metrics were affected the most from this traffic? Network IO:



Bonus: Notice that your `wget` loop creates a lot of duplicate files on your jump box.

- Write a command to delete *all* of these files at once.
- Find a way to run the `wget` command without generating these extra files.
    - Look up the flag options for `wget` and find the flag that lets you choose a location to save the file it downloads.
    - Save that file to the Linux directory known as the "void" or the directory that doesn't save anything.

Bonus: Write a nested loop that sends your `wget` command to all three of your web VMs over and over.
