Team4

ICS344 Project

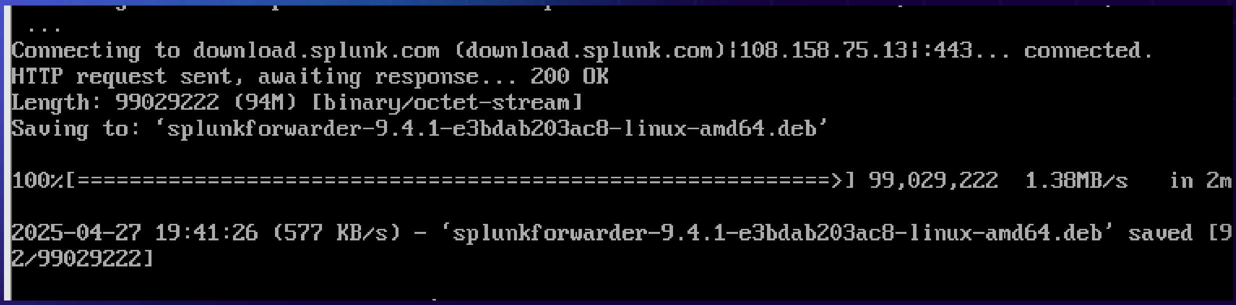
Layal albuhussain 202160450
Narjes Alsaad 202162050
Maha albasri 202180230

Phase 2

Setup and Compromise the service

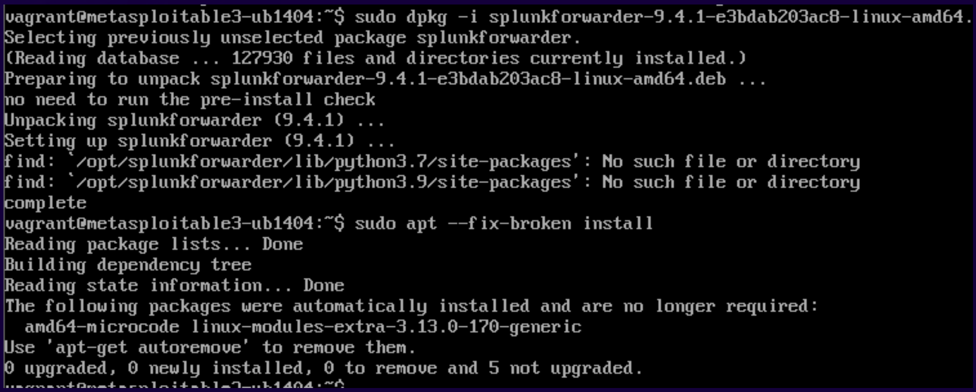
Step 1 : The Installations and Configurations

In this step, we downloaded the Splunk Server in out Kali virtual machine, and configured it as the instruction in the Github repository illustrated, After that, we downloaded the Splunk Forwarder in our victim virtual machine which is the metasploitable, and configured it using the commands mentioned in the Github repository.



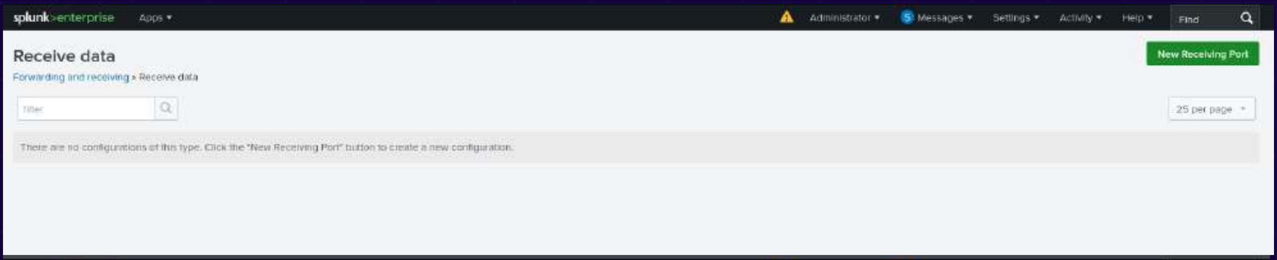
```
 ...
Connecting to download.splunk.com (download.splunk.com)|108.158.75.13|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 99029222 (94M) [binary/octet-stream]
Saving to: ‘splunkforwarder-9.4.1-e3bdab203ac8-linux-amd64.deb’

100%[=========================================================>] 99,029,222  1.38MB/s   in 2m

2025-04-27 19:41:26 (577 KB/s) - ‘splunkforwarder-9.4.1-e3bdab203ac8-linux-amd64.deb’ saved [9
2/99029222]
```

The above image shows the that the downloading process of Splunk-fotwarder in the metasploitable machine was completed successfully.



```
vagrant@metasploitable3-ub1404:~$ sudo dpkg -i splunkforwarder-9.4.1-e3bdab203ac8-linux-amd64.
Selecting previously unselected package splunkforwarder.
(Reading database ... 127930 files and directories currently installed.)
Preparing to unpack splunkforwarder-9.4.1-e3bdab203ac8-linux-amd64.deb ...
no need to run the pre-install check
Unpacking splunkforwarder (9.4.1) ...
Setting up splunkforwarder (9.4.1) ...
find: `/opt/splunkforwarder/lib/python3.7/site-packages': No such file or directory
find: `/opt/splunkforwarder/lib/python3.9/site-packages': No such file or directory
complete
vagrant@metasploitable3-ub1404:~$ sudo apt --fix-broken install
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  amd64-microcode linux-modules-extra-3.13.0-170-generic
Use 'apt-get autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 5 not upgraded.
```

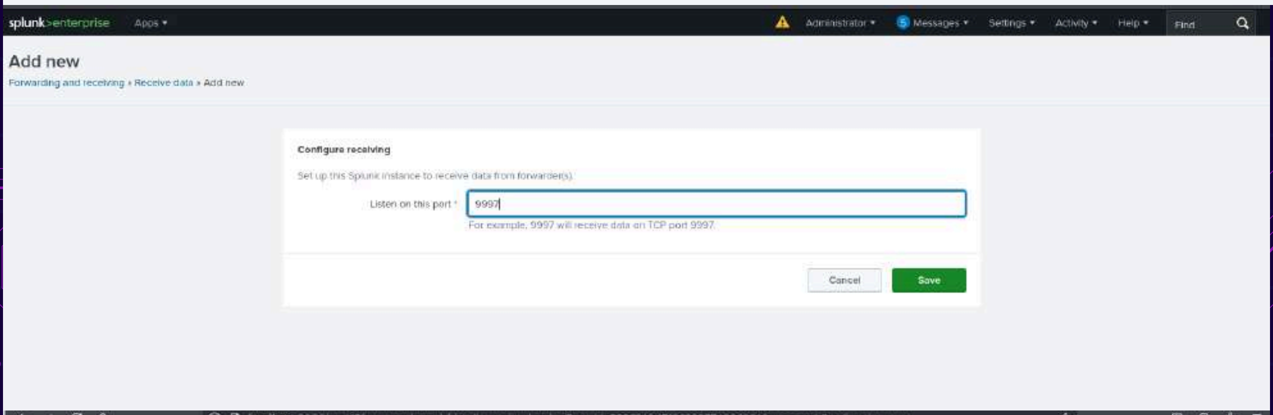
The above image shows the installation of the forwarder package and how we then fixed the broken dependencies using the appropriate commands.



Receive data

Forwarding and receiving » Receive data

There are no configurations of this type. Click the "New Receiving Port" button to create a new configuration.



Add new

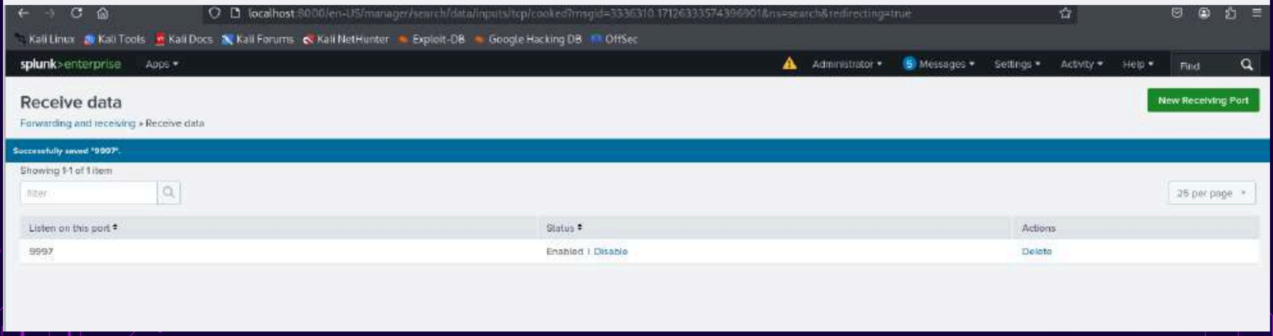
Forwarding and receiving » Receive data » Add new

Configure receiving

Set up this Splunk instance to receive data from forwarder(s).

Listen on this port * 9997

For example, 9997 will receive data on TCP port 9997.



localhost:8000/en-US/manager/search/data/inputs/tcp/cooked?msgid=3336310.17126333574396901&ns=search&redirecting=true

Receive data

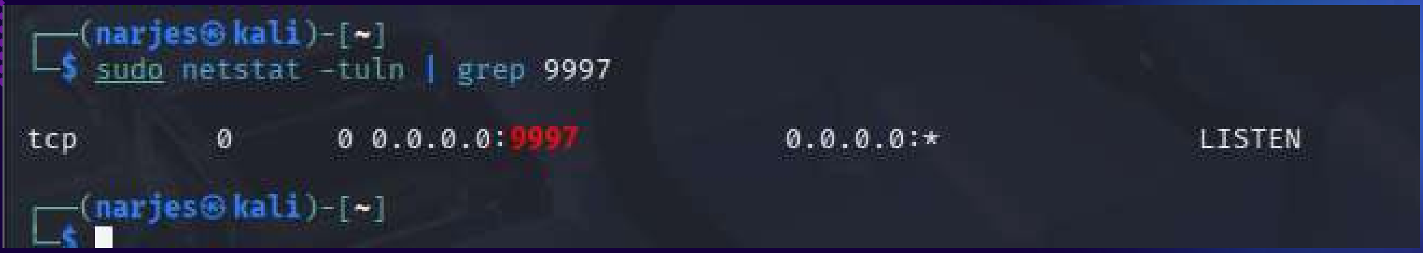
Forwarding and receiving » Receive data

Successfully saved "9997".

Showing 1-1 of 1 item

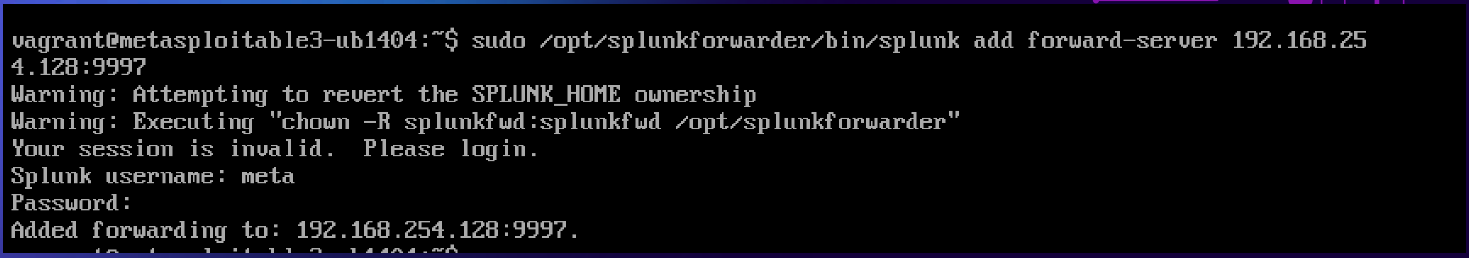
Listen on this port	Status	Actions
9997	Enabled | Disable	Delete

In the Kali machine, we run Splunk, and Added a new port (9997) to receive the logs from the victim machine as shown in the images above.



```
┌──(narjes㉿kali)-[~]
└─$ sudo netstat -tuln | grep 9997

tcp        0      0 0.0.0.0:9997            0.0.0.0:*               LISTEN

┌──(narjes㉿kali)-[~]
└─$ 
```

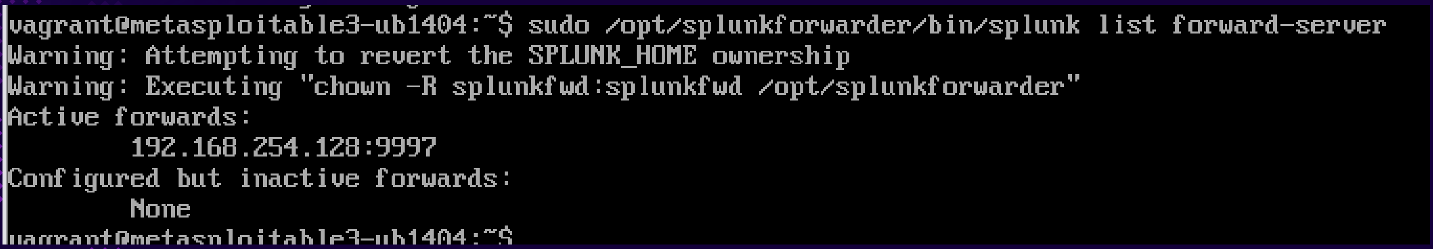
Then in Kali's terminal, we used the command shown in the image above to show that the port 9997 is open, and the Splunk server is listening, and ready to receive data from the Splunk forwarder.



```
vagrant@metasploitable3-ub1404:~$ sudo /opt/splunkforwarder/bin/splunk add forward-server 192.168.25
4.128:9997
Warning: Attempting to revert the SPLUNK_HOME ownership
Warning: Executing "chown -R splunkfwd:splunkfwd /opt/splunkforwarder"
Your session is invalid.  Please login.
Splunk username: meta
Password:
Added forwarding to: 192.168.254.128:9997.
```

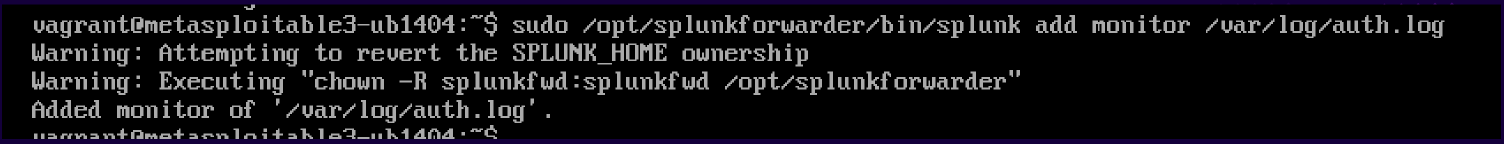
The image above shows registering a forwarder's destination, by adding the server's IP <192.168.254.128> and the listening port<9997>.

The image below shows the confirmation that the forwarder has connected to this indexer.



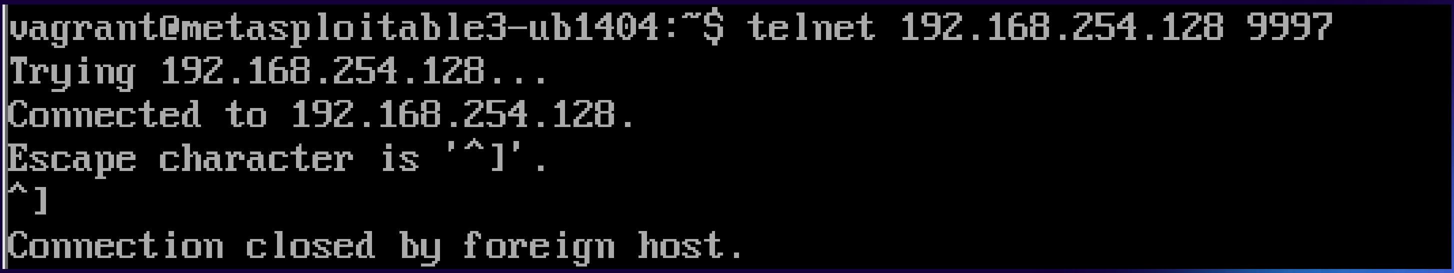
```
vagrant@metasploitable3-ub1404:~$ sudo /opt/splunkforwarder/bin/splunk list forward-server
Warning: Attempting to revert the SPLUNK_HOME ownership
Warning: Executing "chown -R splunkfwd:splunkfwd /opt/splunkforwarder"
Active forwards:
        192.168.254.128:9997
Configured but inactive forwards:
        None
vagrant@metasploitable3-ub1404:~$
```

After that we have the image below showing the command of adding a monitor, which tells the Splunk forwarder in the victim machine to always keep sending the SSH authentication logs to the Splunk indexer in the Kali machine which we added the IP for previously.



```
vagrant@metasploitable3-ub1404:~$ sudo /opt/splunkforwarder/bin/splunk add monitor /var/log/auth.log
Warning: Attempting to revert the SPLUNK_HOME ownership
Warning: Executing "chown -R splunkfwd:splunkfwd /opt/splunkforwarder"
Added monitor of '/var/log/auth.log'.
```

We finally have the last check the Network connection between the Splunk forwarder and the indexer in Kali on port 9997.

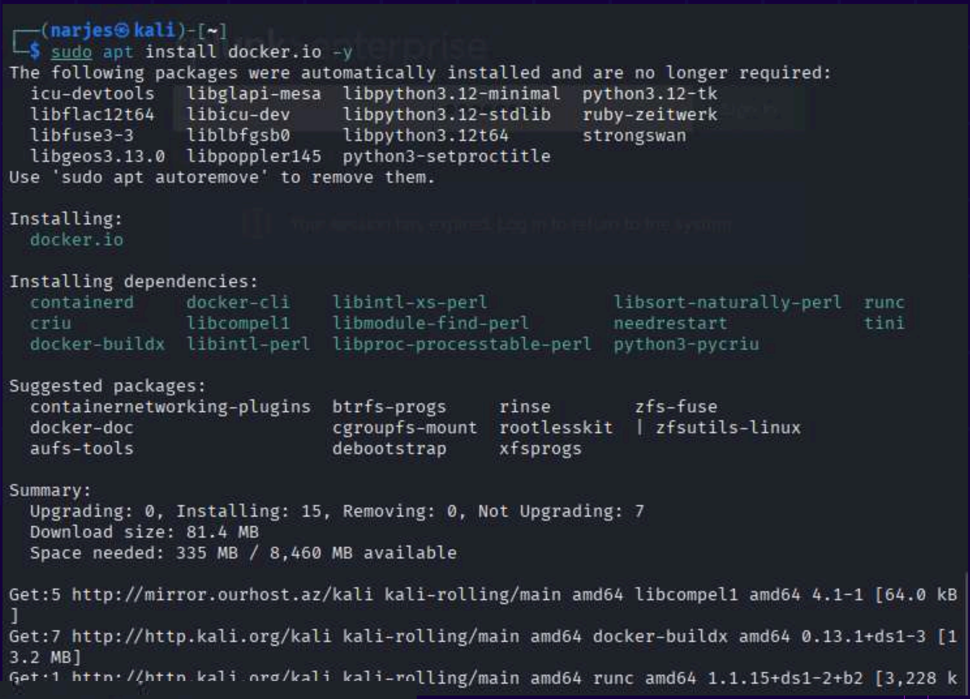


```
vagrant@metasploitable3-ub1404:~$ telnet 192.168.254.128 9997
Trying 192.168.254.128...
Connected to 192.168.254.128.
Escape character is '^]'.
^]
Connection closed by foreign host.
```

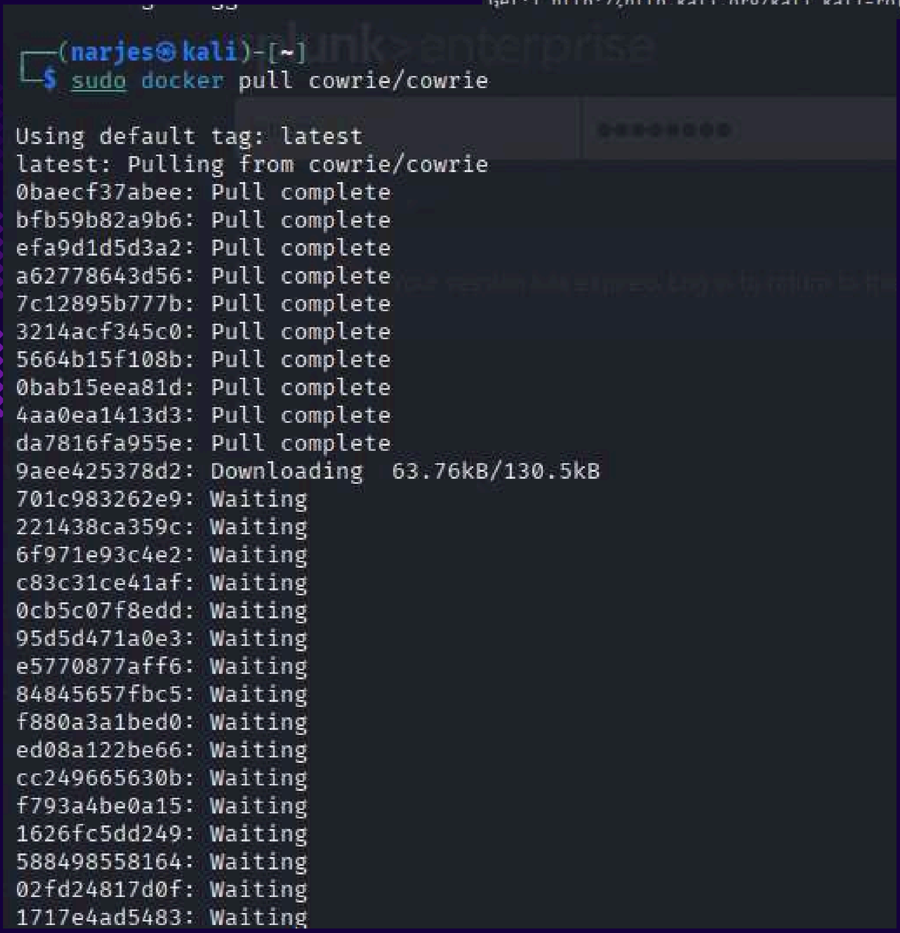
Step 2 : Deploy the Cowrie Honeypot

In Kali virtual machine, we downloaded honeypot and have installed docker into it, so that we can monitor the attack logs from both perspectives. What this means is that with this we can visualize what was the attack and what happened on the victim machine as a result of the attack.

The following image shows the installation of docker in the kali virtual machine using the command "sudo apt install docker.io -y" as shown.

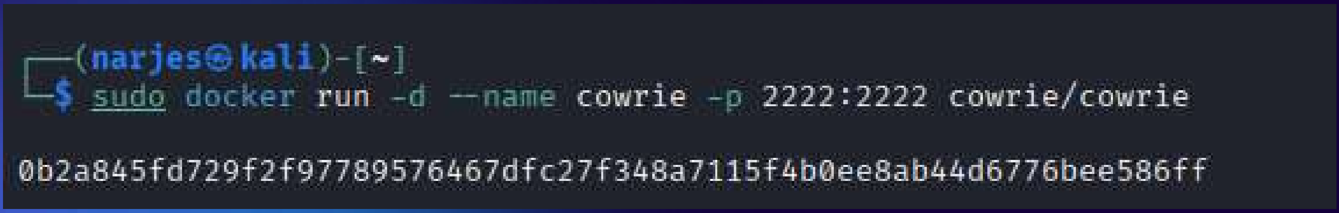


```
┌──(narjes㉿kali)-[~]
└─$ sudo apt install docker.io -y
The following packages were automatically installed and are no longer required:
  icu-devtools    libglapi-mesa   libpython3.12-minimal  python3.12-tk
  libflac12t64    libicu-dev      libpython3.12-stdlib   ruby-zeitwerk
  libfuse3-3      liblbfgsb0      libpython3.12t64       strongswan
  libgeos3.13.0   libpoppler145   python3-setproctitle
Use 'sudo apt autoremove' to remove them.

Installing:
  docker.io

Installing dependencies:
  containerd      docker-cli     libintl-xs-perl            libsort-naturally-perl  runc
  criu            libcompel1     libmodule-find-perl        needrestart             tini
  docker-buildx   libintl-perl   libproc-processtable-perl  python3-pycriu

Suggested packages:
  containernetworking-plugins  btrfs-progs    rinse        zfs-fuse
  docker-doc                   cgroupfs-mount rootlesskit  | zfsutils-linux
  aufs-tools                   debootstrap    xfsprogs

Summary:
  Upgrading: 0, Installing: 15, Removing: 0, Not Upgrading: 7
  Download size: 81.4 MB
  Space needed: 335 MB / 8,460 MB available

Get:5 http://mirror.ourhost.az/kali kali-rolling/main amd64 libcompel1 amd64 4.1-1 [64.0 kB
]
Get:7 http://http.kali.org/kali kali-rolling/main amd64 docker-buildx amd64 0.13.1+ds1-3 [1
3.2 MB]
Get:1 http://http.kali.org/kali kali-rolling/main amd64 runc amd64 1.1.15+ds1-2+b2 [3,228 k
```



```
┌──(narjes㉿kali)-[~]
└─$ sudo docker pull cowrie/cowrie

Using default tag: latest
latest: Pulling from cowrie/cowrie
0baecf37abee: Pull complete
bfb59b82a9b6: Pull complete
efa9d1d5d3a2: Pull complete
a62778643d56: Pull complete
7c12895b777b: Pull complete
3214acf345c0: Pull complete
5664b15f108b: Pull complete
0bab15eea81d: Pull complete
4aa0ea1413d3: Pull complete
da7816fa955e: Pull complete
9aee425378d2: Downloading  63.76kB/130.5kB
701c983262e9: Waiting
221438ca359c: Waiting
6f971e93c4e2: Waiting
c83c31ce41af: Waiting
0cb5c07f8edd: Waiting
95d5d471a0e3: Waiting
e5770877aff6: Waiting
84845657fbc5: Waiting
f880a3a1bed0: Waiting
ed08a122be66: Waiting
cc249665630b: Waiting
f793a4be0a15: Waiting
1626fc5dd249: Waiting
588498558164: Waiting
02fd24817d0f: Waiting
1717e4ad5483: Waiting
```

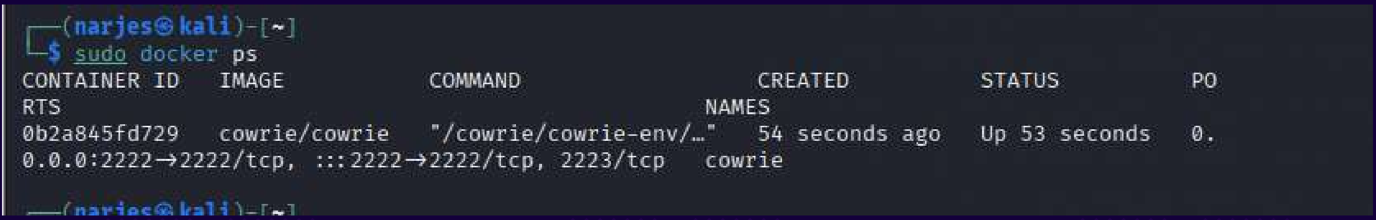
The command shown in the image on the left, downloads the Cowrie Image package from the cowrie/cowrie repository, so that we have all the configurations and required files of the Cowrie Image locally.



```
┌──(narjes㉿kali)-[~]
└─$ sudo docker run -d --name cowrie -p 2222:2222 cowrie/cowrie

0b2a845fd729f2f97789576467dfc27f348a7115f4b0ee8ab44d6776bee586ff
```

After that we used the command in the image above running the Cowrie as a Docker container (runtime instance of Docker image) naming it cowrie, with port 2222 which maps the Kali host with the the docker container.

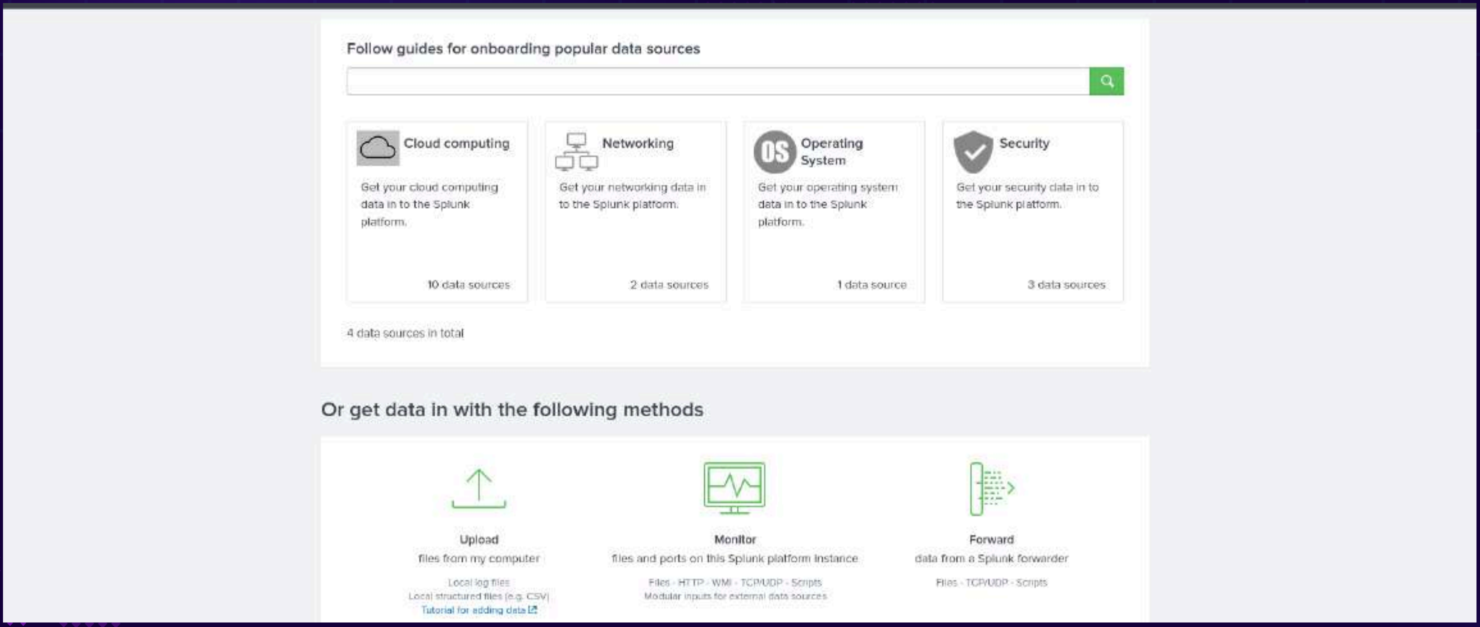
Next, we verified that the Cowrie Honeyput container is running without any issues and that the port 2222 is up, mapped and is ready to receive data, as shown below using the command “sudo docker ps”.



```
┌──(narjes㉿kali)-[~]
└─$ sudo docker ps
CONTAINER ID   IMAGE           COMMAND                  CREATED          STATUS          PO
RTS                                            NAMES
0b2a845fd729   cowrie/cowrie   "/cowrie/cowrie-env/…"   54 seconds ago   Up 53 seconds   0.
0.0.0:2222→2222/tcp, :::2222→2222/tcp, 2223/tcp   cowrie

┌──(narjes㉿kali)-[~]
```

Step 3 : Ingest Honeypot Logs

In this step, after we connected kali to the Cowrie honeypot container, in Splunk, we then click “Monitor” in the get data section shown below. By this, Splunk ingests or in other words imports and indexes the the the logs written as JSON files .



Follow guides for onboarding popular data sources

Cloud computing	Networking	Operating System	Security
Get your cloud computing data in to the Splunk platform.	Get your networking data in to the Splunk platform.	Get your operating system data in to the Splunk platform.	Get your security data in to the Splunk platform.
10 data sources	2 data sources	1 data source	3 data sources

4 data sources in total

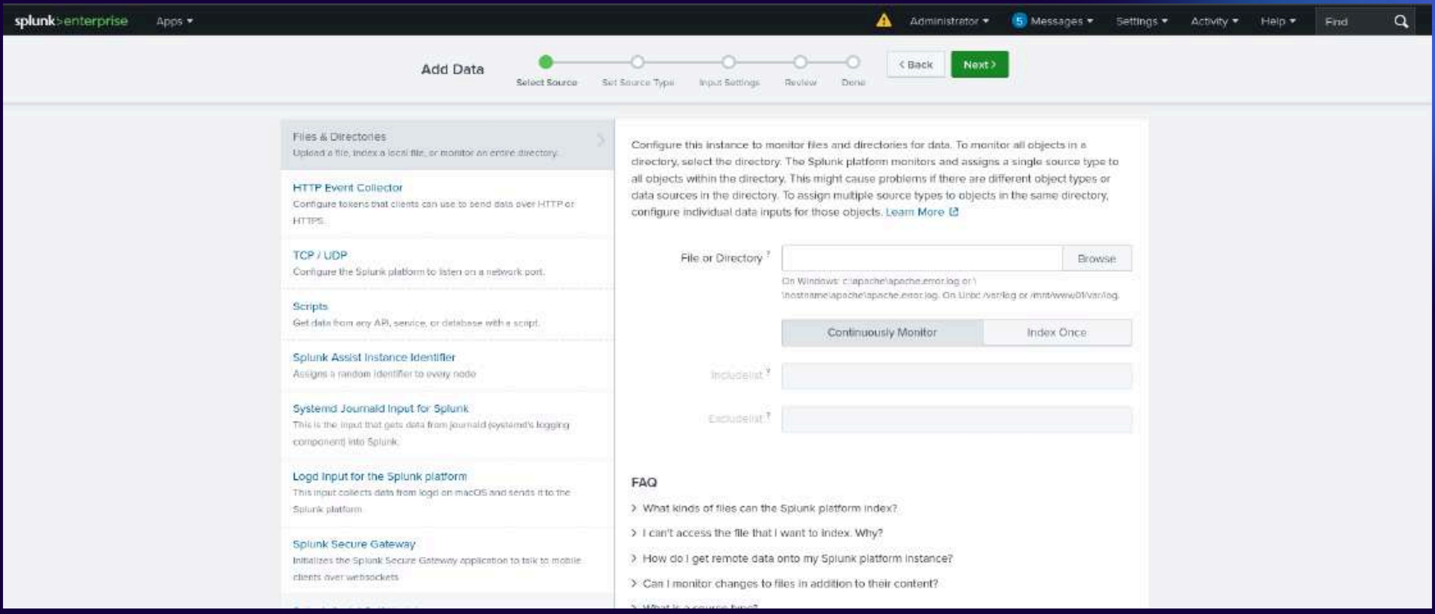
Or get data in with the following methods

Upload
files from my computer
Local log files
Local structured files (e.g. CSV)
Tutorial for adding data

Monitor
files and ports on this Splunk platform instance
Files · HTTP · WMI · TCP/UDP · Scripts
Modular inputs for external data sources

Forward
data from a Splunk forwarder
Files · TCP/UDP · Scripts

In the image below we are telling Splunk which directory to monitor so it can read the logs that the Cowrie honeypot has produced.



Add Data

Select Source · Set Source Type · Input Settings · Review · Done

< Back · Next >

Files & Directories
Upload a file, index a local file, or monitor an entire directory.

HTTP Event Collector
Configure tokens that clients can use to send data over HTTP or HTTPS.

TCP / UDP
Configure the Splunk platform to listen on a network port.

Scripts
Get data from any API, service, or database with a script.

Splunk Assist Instance Identifier
Assigns a random identifier to every node

Systemd Journald Input for Splunk
This is the input that gets data from journald (systemd's logging component) into Splunk.

Logd Input for the Splunk platform
This input collects data from logd on macOS and sends it to the Splunk platform

Splunk Secure Gateway
Initializes the Splunk Secure Gateway application to talk to mobile clients over websockets

Configure this instance to monitor files and directories for data. To monitor all objects in a directory, select the directory. The Splunk platform monitors and assigns a single source type to all objects within the directory. This might cause problems if there are different object types or data sources in the directory. To assign multiple source types to objects in the same directory, configure individual data inputs for those objects. Learn More

File or Directory ? [ ] Browse
On Windows: c:\apache\apache.error.log or \\hostname\apache\apache.error.log. On Unix: /var/log or /mnt/www01/var/log.

[✓ Continuously Monitor] Index Once

Includelist ?

Excludelist ?

FAQ

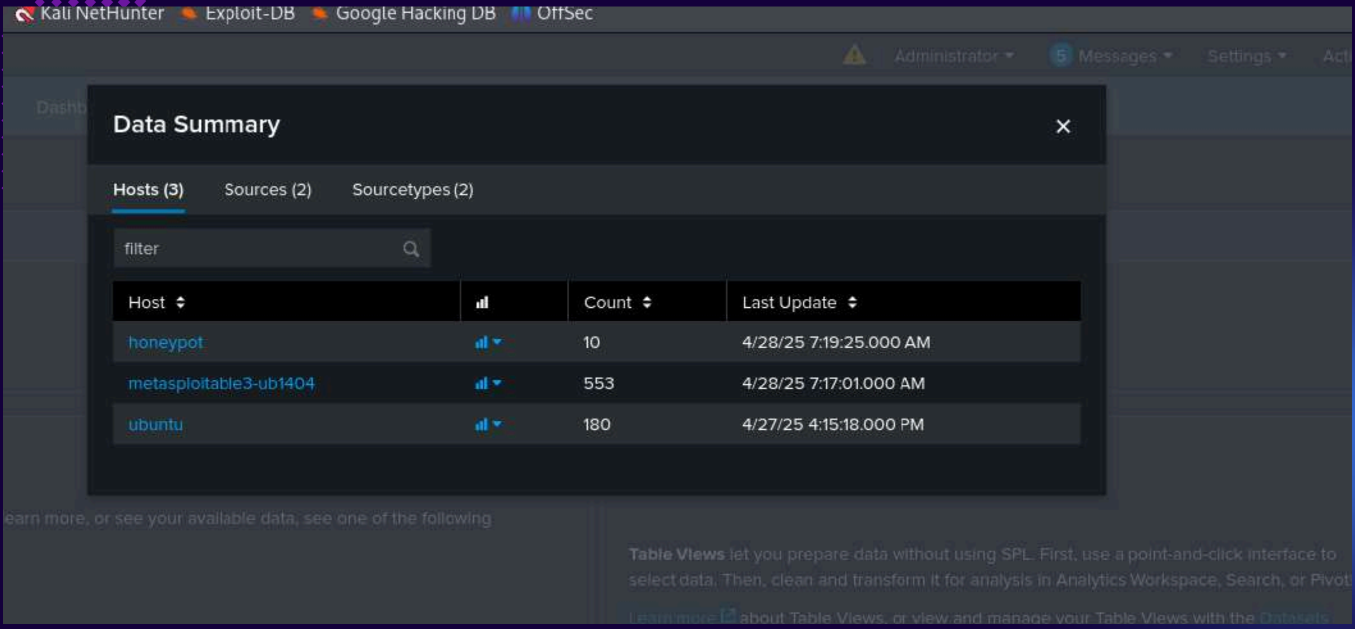
> What kinds of files can the Splunk platform index?
> I can't access the file that I want to index. Why?
> How do I get remote data onto my Splunk platform instance?
> Can I monitor changes to files in addition to their content?

Step 4 : Attack events and Visualization Analysis

In this step, in Splunk, we are studying the SSH authentication events. After that we visualize them using charts and analyze the data.

In order to get logs that we can analyze, we have repeated we have did in the first phase like logging in, attacking the SSH of the victim machine to perform some actions.

We have the image below, in the Splunk web UI, showing the hosts which Splunk has received data or events from and how many events it has received as well as when was the last one.

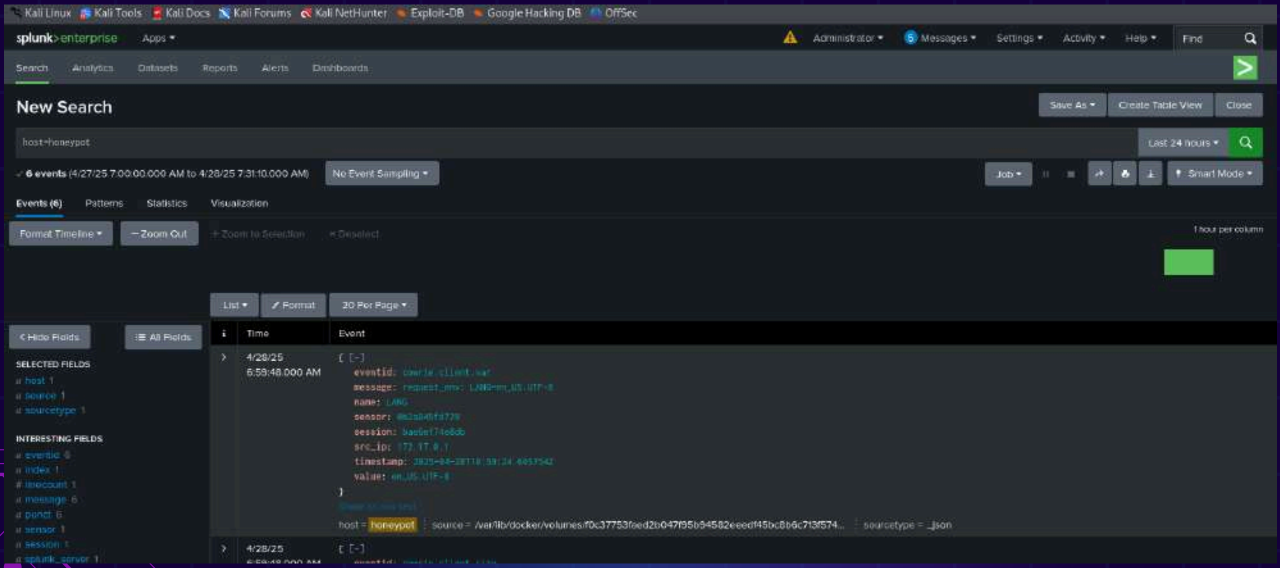


Host		Count	Last Update
honeypot		10	4/28/25 7:19:25.000 AM
metasploitable3-ub1404		553	4/28/25 7:17:01.000 AM
ubuntu		180	4/27/25 4:15:18.000 PM

earn more, or see your available data, see one of the following

Table Views let you prepare data without using SPL. First, use a point-and-click interface to select data. Then, clean and transform it for analysis in Analytics Workspace, Search, or Pivot

Learn more about Table Views, or view and manage your Table Views with the Datasets



```
host=honeypot
```

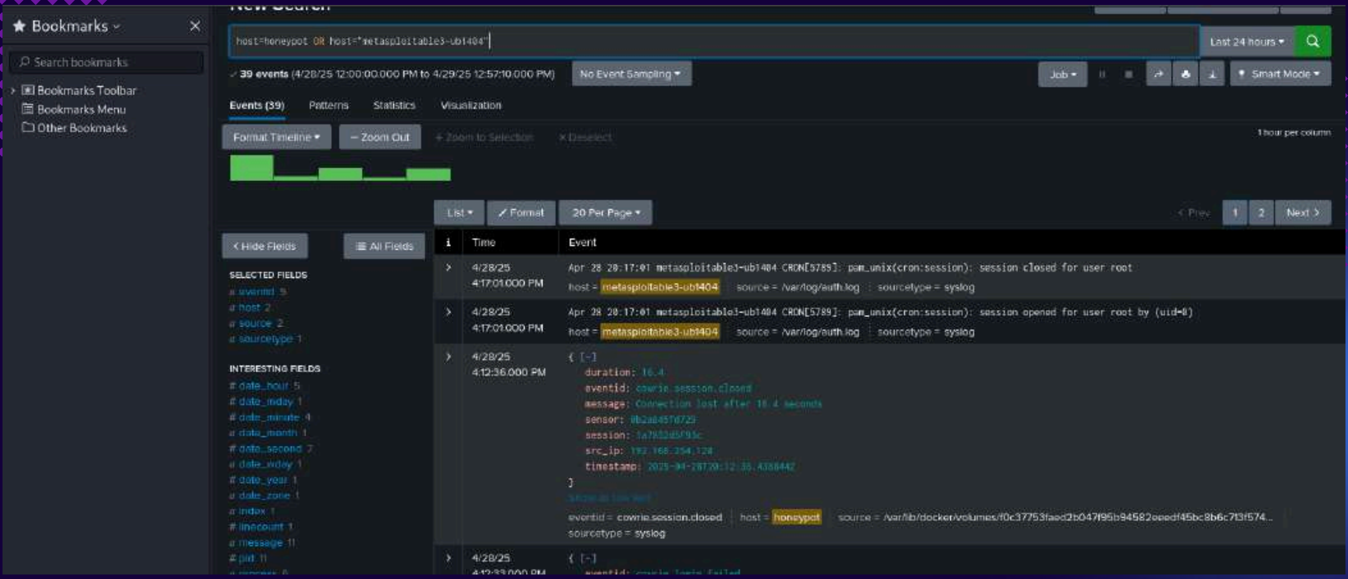
✓ 6 events (4/27/25 7:00:00.000 AM to 4/28/25 7:31:10.000 AM)

i	Time	Event
>	4/28/25 6:59:48.000 AM	see below

```
{ [-]
   eventid: cowrie.client.var
   message: request_env: LANG=en_US.UTF-8
   name: LANG
   sensor: 0b2a8457d729
   session: 9ae6e174e8db
   src_ip: 172.17.0.1
   timestamp: 2025-04-28T10:59:24.4057542
   value: en_US.UTF-8
}
host = honeypot   source = /var/lib/docker/volumes/f0c37753faed2b047f95b94582eeedf45bc8b6c713f574...   sourcetype = _json
```

Here we have some events the Splunk has ingested from Cowrie honeypot written in JSON as shown above.

In the next images we will show specific logs or charts produced after data filtration. first we looked for logs that are from either the Cowrie honeypot or the metasplotable VMs as shown below.



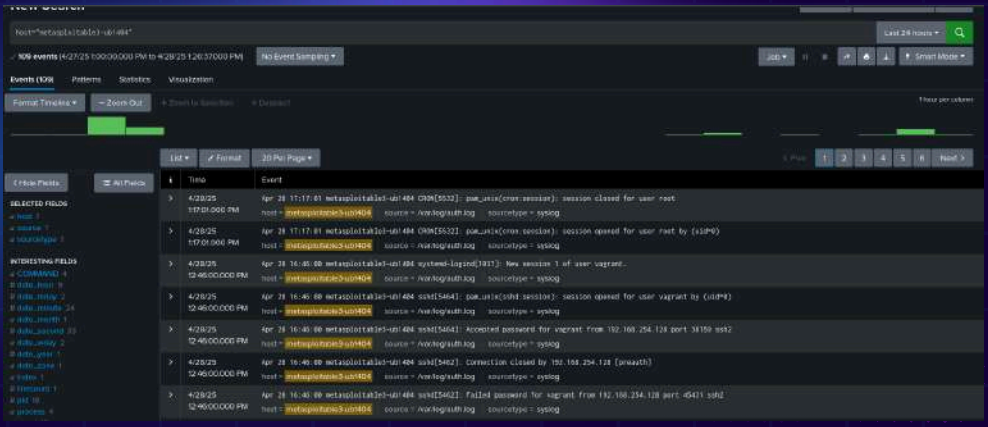
```
host=honeypot OR host="metasploitable3-ub1404"
```

✓ 39 events (4/28/25 12:00:00.000 PM to 4/29/25 12:57:10.000 PM)

```
4/28/25 4:17:01.000 PM
Apr 28 20:17:01 metasploitable3-ub1404 CRON[5789]: pam_unix(cron:session): session closed for user root
host = metasploitable3-ub1404   source = /var/log/auth.log   sourcetype = syslog

4/28/25 4:17:01.000 PM
Apr 28 20:17:01 metasploitable3-ub1404 CRON[5789]: pam_unix(cron:session): session opened for user root by (uid=0)
host = metasploitable3-ub1404   source = /var/log/auth.log   sourcetype = syslog

4/28/25 4:12:36.000 PM
{ [-]
   duration: 16.4
   eventid: cowrie.session.closed
   message: Connection lost after 16.4 seconds
   sensor: 0b2a8457d729
   session: 1a7932d5f93c
   src_ip: 192.168.254.128
   timestamp: 2025-04-28T20:12:35.4358442
}
eventid = cowrie.session.closed   host = honeypot   source = /var/lib/docker/volumes/f0c37753faed2b047f95b94582eeedf45bc8b6c713f574...
sourcetype = syslog
```

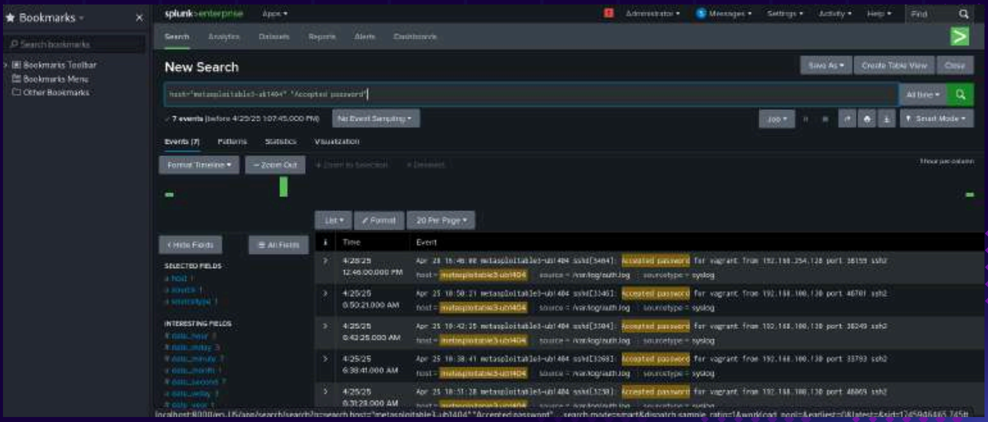
The images below show example of searching the logs for specific events

logs that are only from metasplotable VM using the SPL query “host="metasploitable3-b1404"”



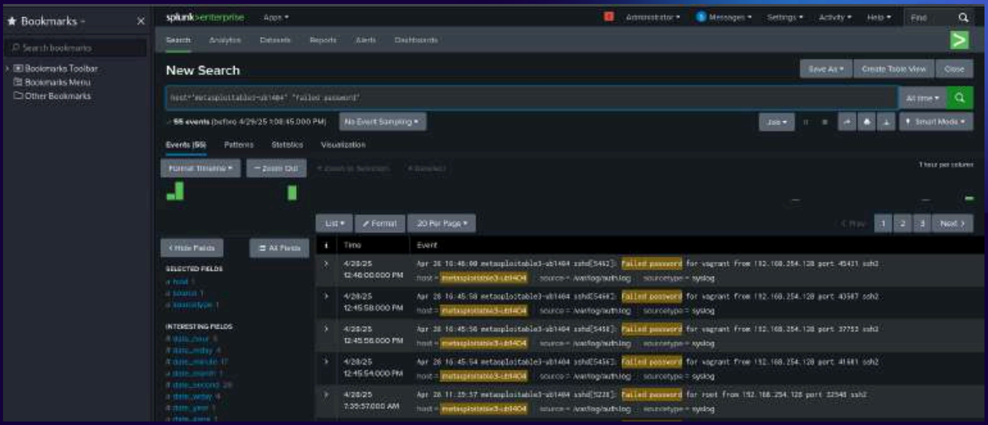
```
host="metasploitable3-ub1404"
✓ 109 events (4/27/25 1:00:00.000 PM to 4/28/25 1:26:37.000 PM)

Time                  Event
4/28/25 1:17:01.000 PM    Apr 28 17:17:01 metasploitable3-ub1404 CRON[5532]: pam_unix(cron:session): session closed for user root
                          host = metasploitable3-ub1404   source = /var/log/auth.log   sourcetype = syslog
4/28/25 1:17:01.000 PM    Apr 28 17:17:01 metasploitable3-ub1404 CRON[5532]: pam_unix(cron:session): session opened for user root by (uid=0)
                          host = metasploitable3-ub1404   source = /var/log/auth.log   sourcetype = syslog
4/28/25 12:46:00.000 PM   Apr 28 16:46:00 metasploitable3-ub1404 systemd-logind[1031]: New session 1 of user vagrant.
                          host = metasploitable3-ub1404   source = /var/log/auth.log   sourcetype = syslog
4/28/25 12:46:00.000 PM   Apr 28 16:46:00 metasploitable3-ub1404 sshd[5464]: pam_unix(sshd:session): session opened for user vagrant by (uid=0)
                          host = metasploitable3-ub1404   source = /var/log/auth.log   sourcetype = syslog
4/28/25 12:46:00.000 PM   Apr 28 16:46:00 metasploitable3-ub1404 sshd[5464]: Accepted password for vagrant from 192.168.254.128 port 38159 ssh2
                          host = metasploitable3-ub1404   source = /var/log/auth.log   sourcetype = syslog
4/28/25 12:46:00.000 PM   Apr 28 16:46:00 metasploitable3-ub1404 sshd[5462]: Connection closed by 192.168.254.128 [preauth]
                          host = metasploitable3-ub1404   source = /var/log/auth.log   sourcetype = syslog
4/28/25 12:46:00.000 PM   Apr 28 16:46:00 metasploitable3-ub1404 sshd[5462]: Failed password for vagrant from 192.168.254.128 port 45431 ssh2
                          host = metasploitable3-ub1404   source = /var/log/auth.log   sourcetype = syslog
```

logs that are only from metasplotable VM and the password was accepted (successful log in) using the SPL query “host="metasploitable3-b1404" “Accepted password””



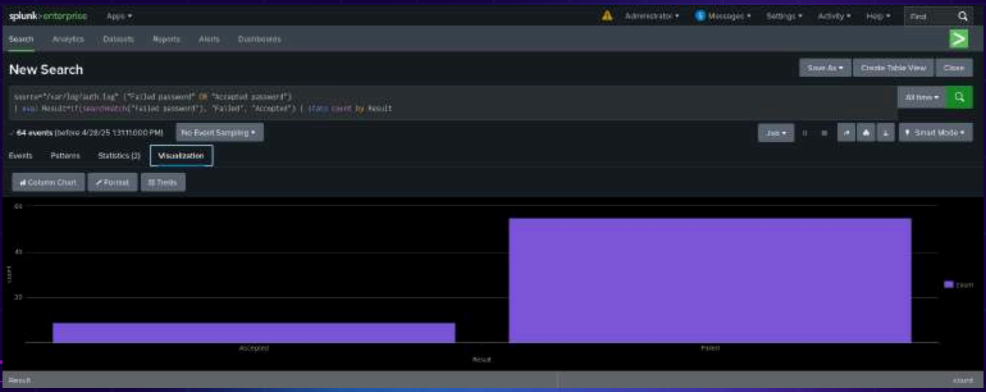
```
New Search
host="metasploitable3-ub1404" "Accepted password"
✓ 7 events (before 4/25/25 1:07:45.000 PM)

Time                  Event
4/28/25 12:46:00.000 PM   Apr 28 16:46:00 metasploitable3-ub1404 sshd[5464]: Accepted password for vagrant from 192.168.254.128 port 38159 ssh2
                          host = metasploitable3-ub1404   source = /var/log/auth.log   sourcetype = syslog
4/25/25 6:50:21.000 AM    Apr 25 10:50:21 metasploitable3-ub1404 sshd[3346]: Accepted password for vagrant from 192.168.100.130 port 46701 ssh2
                          host = metasploitable3-ub1404   source = /var/log/auth.log   sourcetype = syslog
4/25/25 6:42:25.000 AM    Apr 25 10:42:25 metasploitable3-ub1404 sshd[3304]: Accepted password for vagrant from 192.168.100.130 port 38249 ssh2
                          host = metasploitable3-ub1404   source = /var/log/auth.log   sourcetype = syslog
4/25/25 6:38:41.000 AM    Apr 25 10:38:41 metasploitable3-ub1404 sshd[3268]: Accepted password for vagrant from 192.168.100.130 port 33793 ssh2
                          host = metasploitable3-ub1404   source = /var/log/auth.log   sourcetype = syslog
4/25/25 6:31:28.000 AM    Apr 25 10:31:28 metasploitable3-ub1404 sshd[3238]: Accepted password for vagrant from 192.168.100.130 port 48069 ssh2
```

logs that are only from metasplotable VM and the password was wrong (denied log in) using the SPL query “host="metasploitable3-b1404" “Failed password””



```
New Search
host="metasploitable3-ub1404" "Failed password"
✓ 55 events (before 4/29/25 1:08:45.000 PM)

Time                  Event
4/28/25 12:46:00.000 PM   Apr 28 16:46:00 metasploitable3-ub1404 sshd[5462]: Failed password for vagrant from 192.168.254.128 port 45431 ssh2
                          host = metasploitable3-ub1404   source = /var/log/auth.log   sourcetype = syslog
4/28/25 12:45:58.000 PM   Apr 28 16:45:58 metasploitable3-ub1404 sshd[5460]: Failed password for vagrant from 192.168.254.128 port 43587 ssh2
                          host = metasploitable3-ub1404   source = /var/log/auth.log   sourcetype = syslog
4/28/25 12:45:56.000 PM   Apr 28 16:45:56 metasploitable3-ub1404 sshd[5458]: Failed password for vagrant from 192.168.254.128 port 37753 ssh2
                          host = metasploitable3-ub1404   source = /var/log/auth.log   sourcetype = syslog
4/28/25 12:45:54.000 PM   Apr 28 16:45:54 metasploitable3-ub1404 sshd[5456]: Failed password for vagrant from 192.168.254.128 port 41681 ssh2
                          host = metasploitable3-ub1404   source = /var/log/auth.log   sourcetype = syslog
4/28/25 7:35:57.000 AM    Apr 28 11:35:57 metasploitable3-ub1404 sshd[5228]: Failed password for root from 192.168.254.128 port 32548 ssh2
                          host = metasploitable3-ub1404   source = /var/log/auth.log   sourcetype = syslog
```

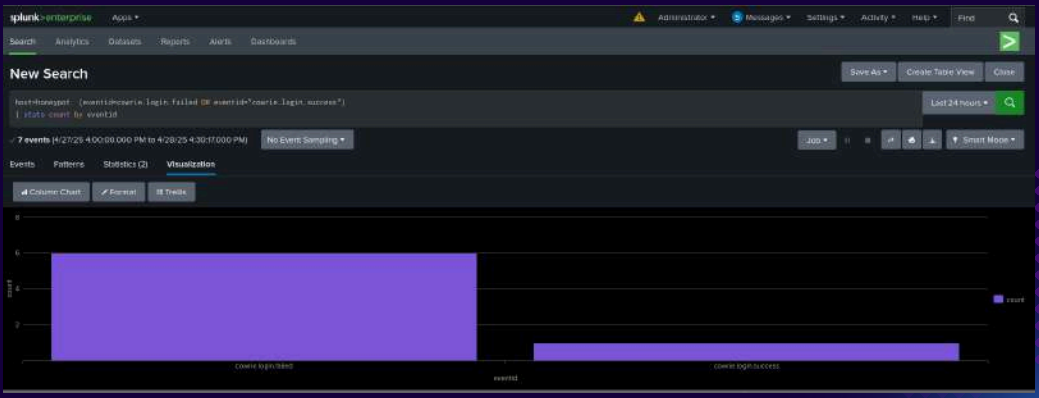
The images below show visualization of some charts we produced.



```
source="/var/log/auth.log" ("Failed password" OR "Accepted password")
| eval Result=if(searchmatch("Failed password"), "Failed", "Accepted") | stats count by Result
```

64 events (before 4/28/25 1:31:11.000 PM)

In this chart we have log in attempts ingested from the victim machine classified as accepted and failed. As the failed attempts are so high compared to the accepted ones this can indicate a brute-force attack on the metasploitable VM. this was produced using the query "source="/var/log/auth.log" ("Failed password" OR "Accepted password")
| eval Result=if(searchmatch("Failed password"), "Failed", "Accepted")
| stats count by Result"

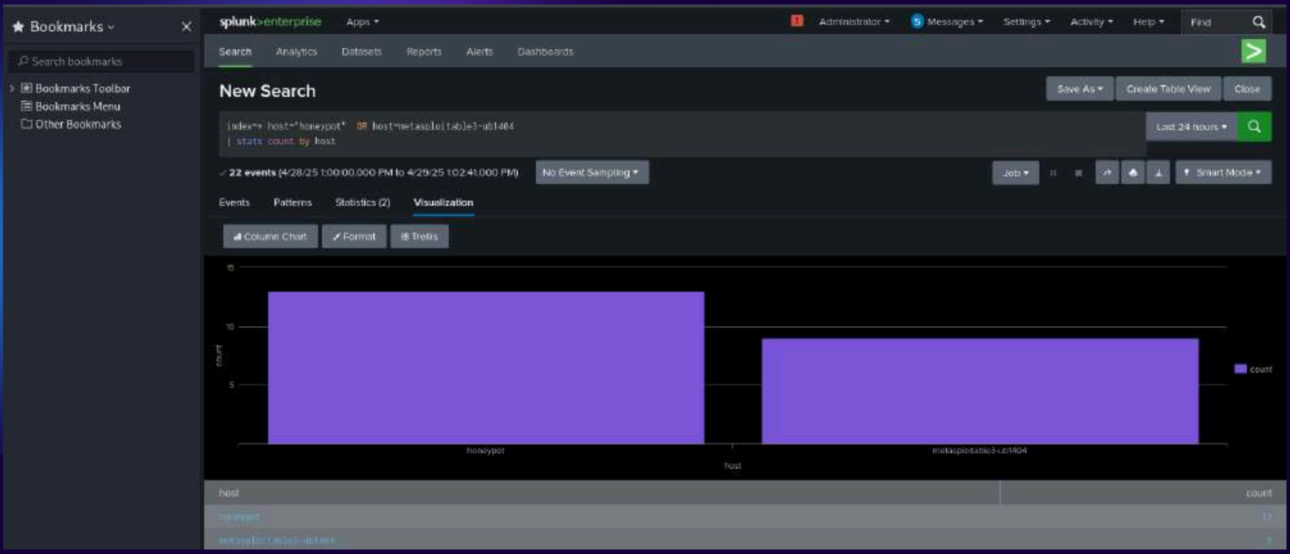


```
host=honeypot (eventid=cowrie.login.failed OR eventid="cowrie.login.success")
| stats count by eventid
```

7 events (4/27/25 4:00:00.000 PM to 4/28/25 4:30:17.000 PM)

Avobe we logs from the honeypot, of log in attacks against Cowrie honeypot, we are comparing the successful vs failed log in attacks, with the failure one peeking, it shows that attackers probing fake credentials

using the query "host="honeypot" (eventid=cowrie.login.failed OR eventid=cowrie.login.success) stats count by eventid"



```
index=* host="honeypot" OR host=metasploitable3-ub1404
| stats count by host
```

✓ 22 events (4/28/25 1:00:00.000 PM to 4/29/25 1:02:41.000 PM)

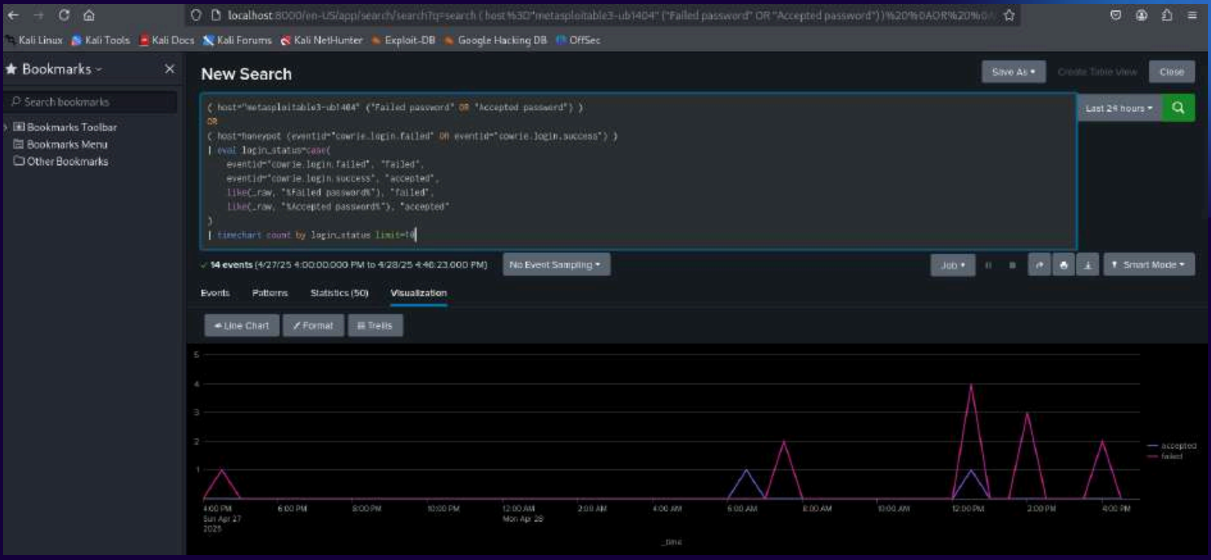
host	count
honeypot	13
metasploitable3-ub1404	9

The above chart shows a comparison between the events count from both the metasploitable VM vs the honeypot. with honeypot having 13 attempts while the metasploitable having 9 attempts only.

using the query “index=* (host="honeypot" OR host="metasploitable3-ub1404") | stats count by host”

Finally, the timechart below combines login attempts against both the Cowrie honeypot and the Metasploitable3 victim, classifying each as failed or accepted. The chart clearly shows peaks in failed attempts, highlighting periods of intense brute-force activity.

using the query “(
host="metasploitable3-ub1404" ("Failed password" OR "Accepted password")) OR (
host="honeypot" (eventid=cowrie.login.failed OR eventid=cowrie.login.success)) | eval login_status = case(
eventid=="cowrie.login.failed", "failed",
eventid=="cowrie.login.success", "accepted",
like(_raw, "%Failed password%"), "failed",
like(_raw, "%Accepted password%"),"accepted")
| timechart count BY login_status limit=10”



localhost:8000/en-US/app/search/search?q=search (host%3D"metasploitable3-ub1404" ("Failed password" OR "Accepted password"))%20%0AOR%20%0

```
( host="metasploitable3-ub1404" ("Failed password" OR "Accepted password") )
OR
( host=honeypot (eventid="cowrie.login.failed" OR eventid="cowrie.login.success") )
| eval login_status=case(
    eventid="cowrie.login.failed", "failed",
    eventid="cowrie.login.success", "accepted",
    like(_raw, "%Failed password%"), "failed",
    like(_raw, "%Accepted password%"), "accepted"
)
| timechart count by login_status limit=10
```

✓ 14 events (4/27/25 4:00:00.000 PM to 4/28/25 4:46:23.000 PM)