EDR Documentation

By Shivam Kishor Patange

*Points to Note:

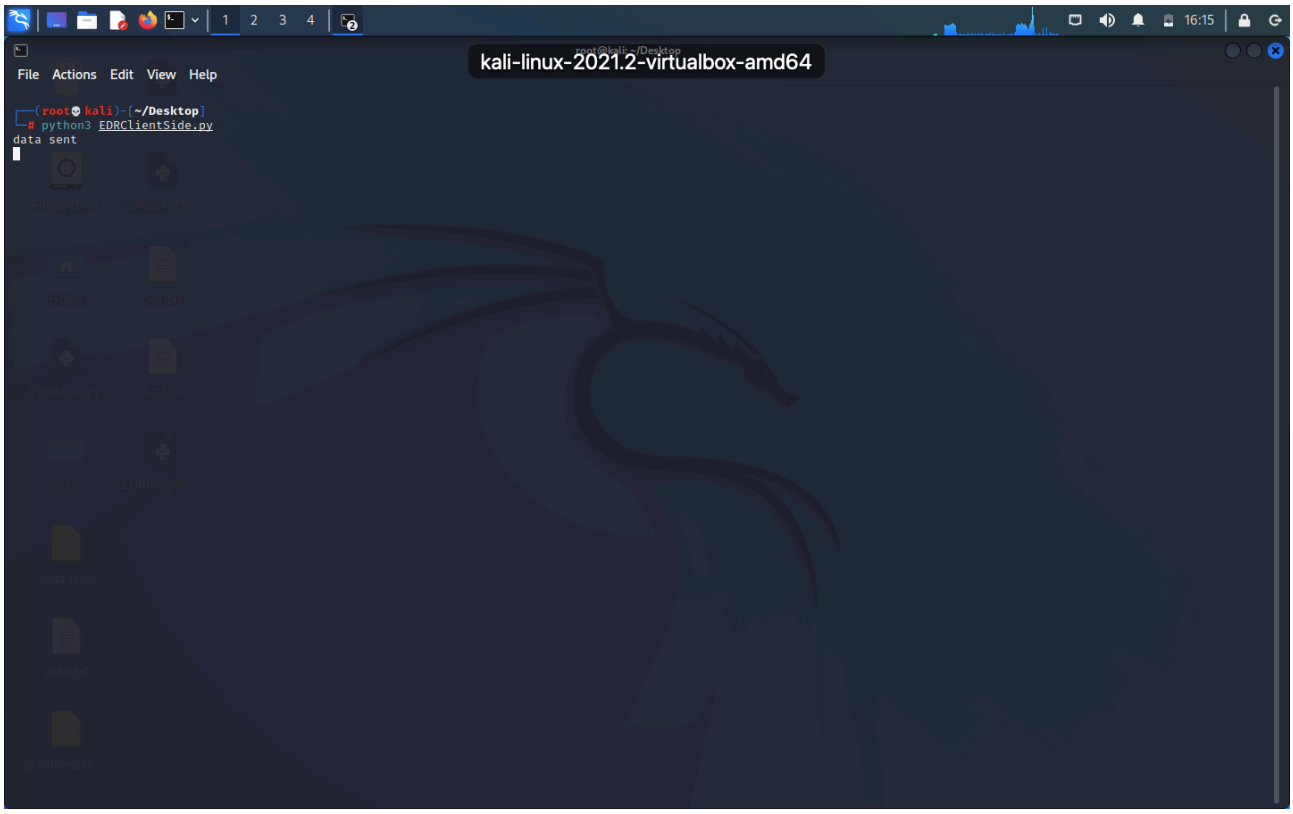
1) The messages created by the Client side psad tool and the python script can be of any size between 500KB to 10MB, So it is suggested to increase the default (1MB) Kafka Producer message size accordingly.
2) *EDRClientSide.py* same as *EDR.py*, but with interfaced Kafka Producer.
3) Requirements:
    1) Server side:
        1) Kafka-python : pip install kafka-python
        2) PyWebIO : pip3 install -U pywebio
    2) Client Side:
        1) Kafka-python : pip install kafka-python
        2) psad : sudo apt install psad

Working:

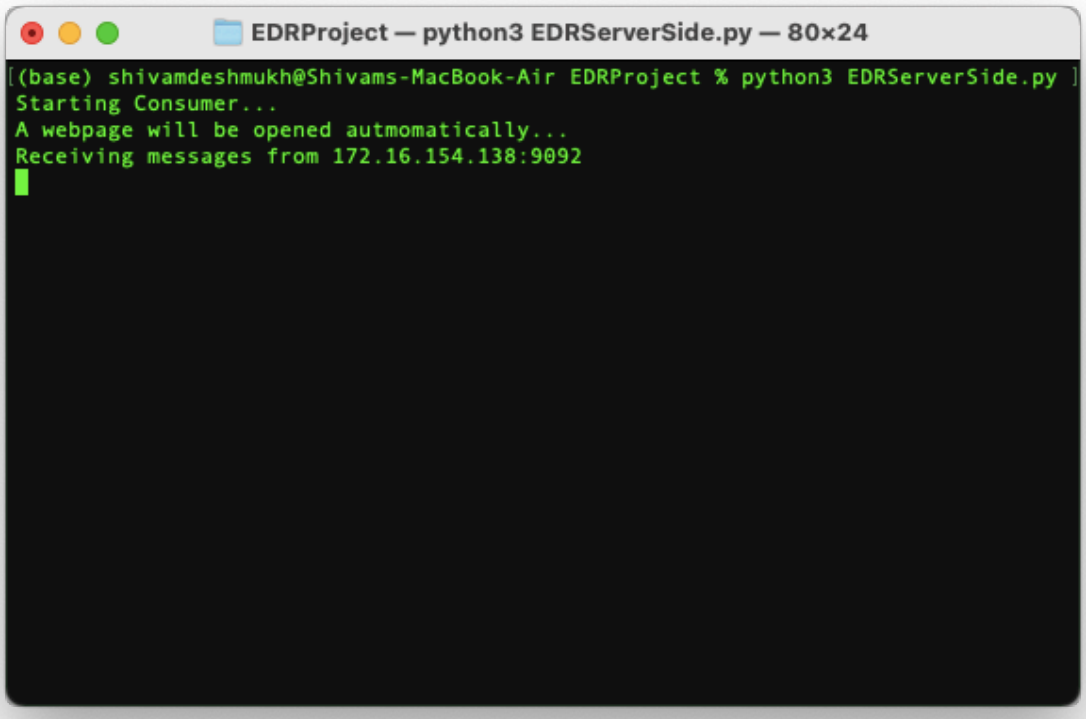
1) Starting with setting up the 1. Kafka Server, 2. Kafka Zookeeper and 3. CMAK on the central Kafka server, noting the IP addresses and ports of the respective service which we will need later to feed in the *EDRServerSide.py* and *EDRClientSide.py* scripts. Once we got everything up and running we can go ahead with the scripts.
2) Once the all the Kafka services are ensured to be up and communicating stage which we can check by browsing the IP of the CMAK on port 8080 i.e. 172.16.154.138:8080, if it is running fine we should get a CMAK configuration screen with list of available Topics.
3) As a EDR solution is is divided into two parts 1. Client (Which will suspected to be under attack) and 2. Server (Which will analyse the attack vectors, sources, scripts, etc.).
4) Both Client and Server side scripts are written in Python 3.
5) After ensuring all the requirements are satisfied we can go ahead and run the actual scripts.
6) We will start with running the *EDRClientSide.py* file on the client system.
7) This python script contains *psad* scripts which will collect the data from operating system.
8) This data along with *lsof* and *processes* data will be sent to the server via Kafka server every 10 seconds.
9) Then we will run *EDRServerSide.py* on the server machine (on which we want to see the data sent by the *EDRClientSide.py* script).
10) This *EDRServerSide.py* will receive the data once when launched and automatically open a webpage on the browser and show the received data. As *psad_data, lsof_data, process_data* and inside it *Top 50 Signature Matches, etc.*
11) The *EDRClientSide.py* will continuously (every 10 seconds) send the data to the *EDRServerSide.py,* but it will only show the data received in real time (At the time of running *EDRServerSide.py* script).
12) So it need to be run every time a user wants to see the data.

Demonstration:

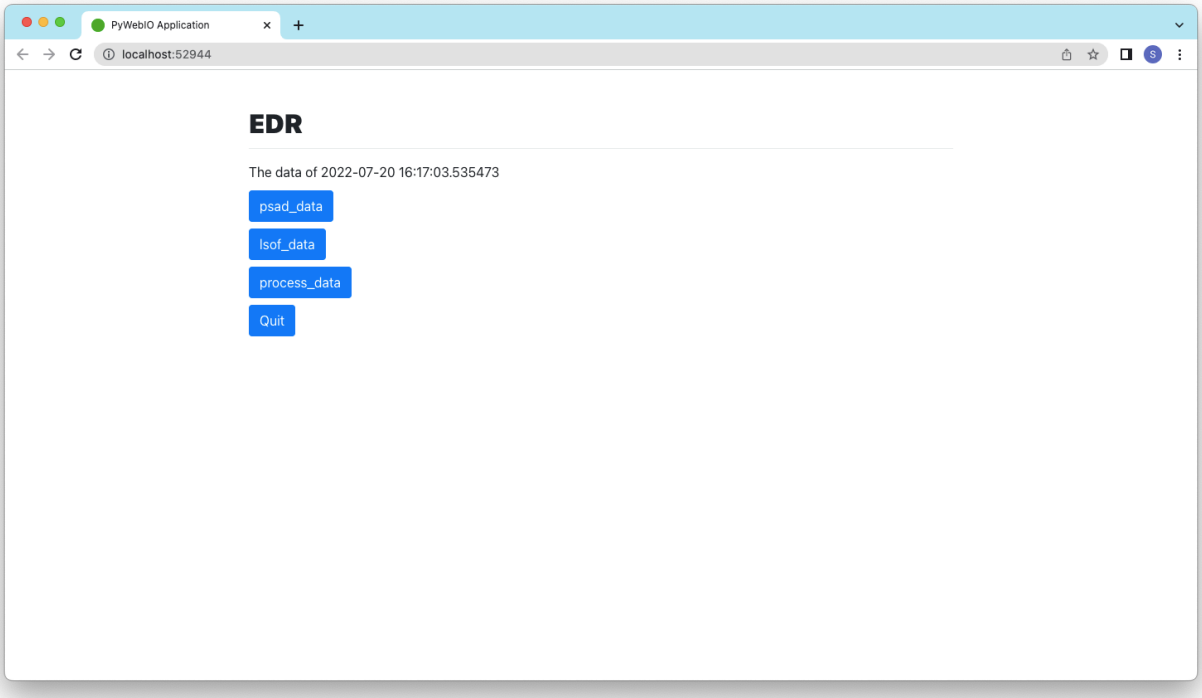
1) Running *EDRClientSide.py:*
   python3 EDRClientSide.py



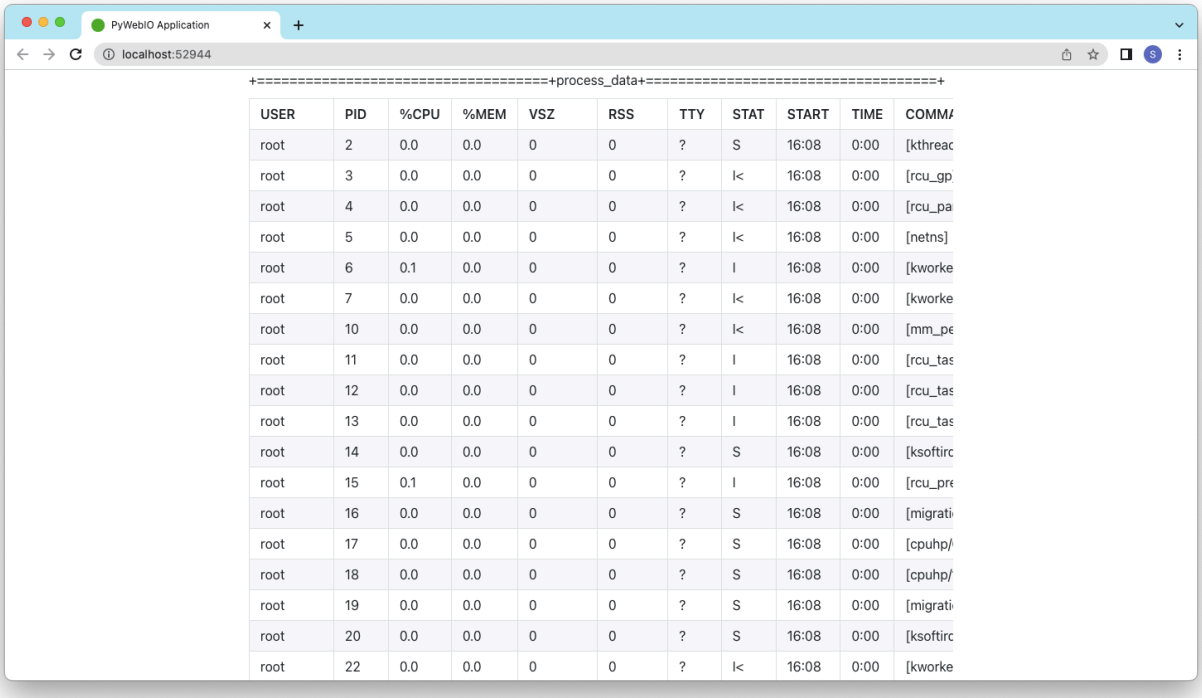
2) Running *EDRServerSide.py:*
   python3 EDRServerSide.py



3) Webpage Output:

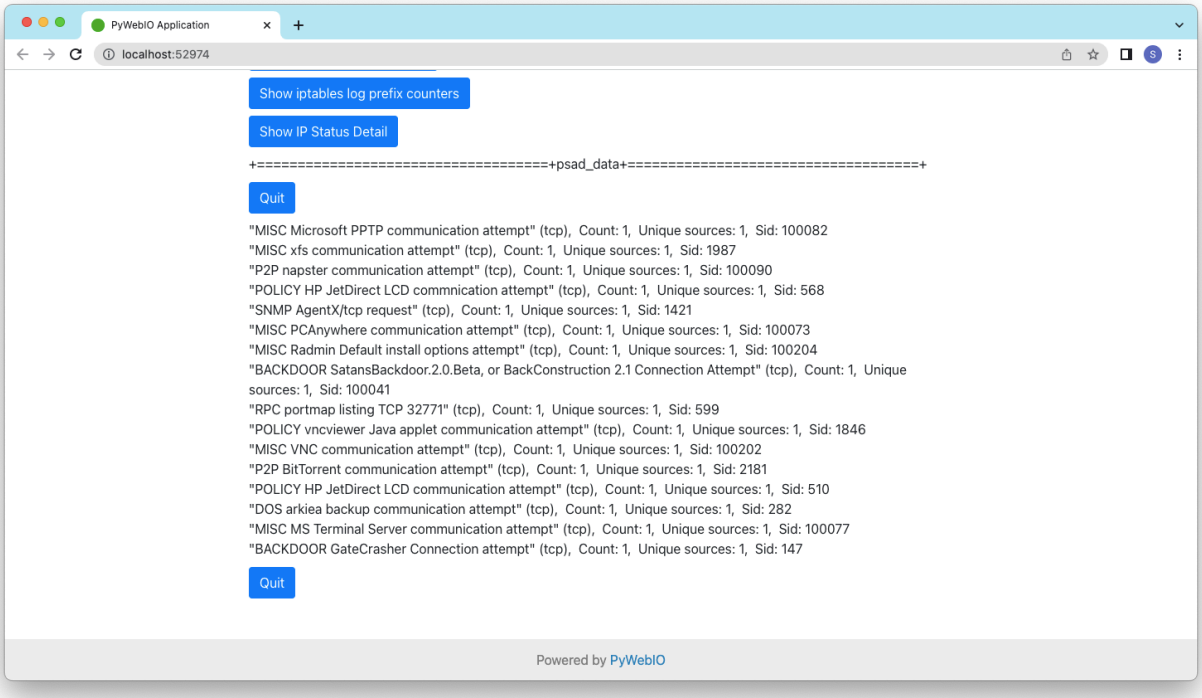


4) process_data:

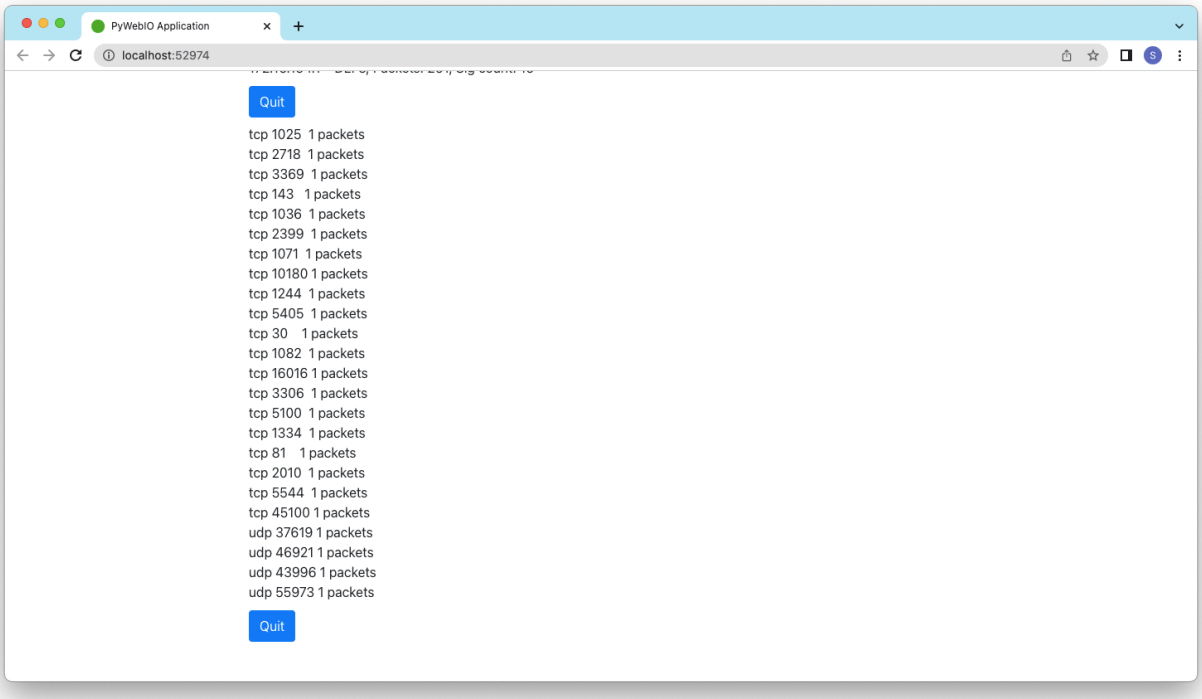


USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMA
root	2	0.0	0.0	0	0	?	S	16:08	0:00	[kthreac
root	3	0.0	0.0	0	0	?	I<	16:08	0:00	[rcu_gp
root	4	0.0	0.0	0	0	?	I<	16:08	0:00	[rcu_pa
root	5	0.0	0.0	0	0	?	I<	16:08	0:00	[netns]
root	6	0.1	0.0	0	0	?	I	16:08	0:00	[kworke
root	7	0.0	0.0	0	0	?	I<	16:08	0:00	[kworke
root	10	0.0	0.0	0	0	?	I<	16:08	0:00	[mm_pe
root	11	0.0	0.0	0	0	?	I	16:08	0:00	[rcu_tas
root	12	0.0	0.0	0	0	?	I	16:08	0:00	[rcu_tas
root	13	0.0	0.0	0	0	?	I	16:08	0:00	[rcu_tas
root	14	0.0	0.0	0	0	?	S	16:08	0:00	[ksoftirc
root	15	0.1	0.0	0	0	?	I	16:08	0:00	[rcu_pre
root	16	0.0	0.0	0	0	?	S	16:08	0:00	[migrati
root	17	0.0	0.0	0	0	?	S	16:08	0:00	[cpuhp/
root	18	0.0	0.0	0	0	?	S	16:08	0:00	[cpuhp/
root	19	0.0	0.0	0	0	?	S	16:08	0:00	[migrati
root	20	0.0	0.0	0	0	?	S	16:08	0:00	[ksoftirc
root	22	0.0	0.0	0	0	?	I<	16:08	0:00	[kworke

5) Top 50 Signature matches:



6) Top 20 Scanned ports:



Code Explanation :

1) Client Side Code:

On the client side we are using Kafka Producer to send the data to the central Kafka Server.

```
#JSON serializer configuration part
def json_serializer(data):
    return json.dumps(data).encode("utf-8")

#bootstrap server's address and dedicated port
producer = KafkaProducer(bootstrap_servers = ["172.16.154.138:9092"],
value_serializer=json_serializer)
```

First we are serialising the data using json_serialzer() function, and then sending it to the given set list of bootstrap servers (In our case only one i.e 172:16:154:138:9092) using KafkaProducer().

2) Server Side Code:

We are using Kafka consumer to receive the messages from the Kafka server to show on the page, for that we are using PyWebIO for creating web pages which will show the received information. We are using methods like put_button(), put_table(), etc. to show it on the page.

```
    address = "172.16.154.138:9092"    #Address of the Topic
(Kafka Server)
    consumer = KafkaConsumer(
        "EDR",                  #Topic
        bootstrap_servers = address,
        auto_offset_reset="latest",   #Printing method
        group_id= "consumer-grpA"     #Group
    )
```

As shown in code above we pass the bootstrap server address, Topic name, auto offset preference, and consumer group id to the Kafka Consumer method.

Then by using PyWebIO methods we are showing the information on the web page.

```
    put_markdown('<h1><b>EDR</b></h1>')
    put_text("The data of "+a["timestamp"])
    put_button('psad_data', onclick=psad_data_fun)
    put_button('lsof_data', onclick=lsof_data_fun)
    put_button('process_data', onclick=process_data_fun)
    put_button('Quit', onclick=killApplication)
```