In [1]:

```
## --- 3.1 --- ##
from pyspark import SparkConf
from pyspark.sql import SparkSession
import os
os.environ['PYSPARK_SUBMIT_ARGS'] = '--packages org.apache.spark:spark-streaming-kafka-0
-10_2.12:3.0.0,org.apache.spark:spark-sql-kafka-0-10_2.12:3.0.0 pyspark-shell'

# run Spark in local mode with 2 cores
master = "local[2]"
app_name = "Attack_Detection_Streaming"
# set master, app name, and UTC timezone
spark_conf = SparkConf().setMaster(master).setAppName(app_name).set("spark.sql.session.ti
meZone", "UTC")

spark = SparkSession.builder.config(conf=spark_conf).getOrCreate()
```

In [2]:

```
## --- 3.2 --- ##
from pyspark.sql.types import *
# ingest process activities
topic_process = "process"
df_process = spark \
          .readStream \
          .format("kafka") \
          .option("kafka.bootstrap.servers", "127.0.0.1:9092") \
          .option("subscribe", topic_process) \
          .load()

# ingest memory activities
topic_memory = "memory"
df_memory = spark \
          .readStream \
          .format("kafka") \
          .option("kafka.bootstrap.servers", "127.0.0.1:9092") \
          .option("subscribe", topic_memory) \
          .load()

# check the schema
df_process.printSchema()
df_memory.printSchema()
```

```
root
 |-- key: binary (nullable = true)
 |-- value: binary (nullable = true)
 |-- topic: string (nullable = true)
 |-- partition: integer (nullable = true)
 |-- offset: long (nullable = true)
 |-- timestamp: timestamp (nullable = true)
 |-- timestampType: integer (nullable = true)

root
 |-- key: binary (nullable = true)
 |-- value: binary (nullable = true)
 |-- topic: string (nullable = true)
 |-- partition: integer (nullable = true)
 |-- offset: long (nullable = true)
 |-- timestamp: timestamp (nullable = true)
 |-- timestampType: integer (nullable = true)
```

In [3]:

```
## --- 3.3 --- ##
from pyspark.sql.functions import regexp_extract
from pyspark.sql.functions import isnan, when, count, col
```

```
import pyspark.sql.functions as F


# Converting the key/value from the kafka data stream to string
df_process = df_process.selectExpr("CAST(key AS STRING)", "CAST(value AS STRING)")
df_memory = df_memory.selectExpr("CAST(key AS STRING)", "CAST(value AS STRING)")

# Check the read-in schema
print("Read-in schema for process activities")
df_process.printSchema()
print("Read-in schema for memory activities")
df_memory.printSchema()

# Specify explicit schema for both
process_schema = ArrayType(StructType([StructField("sequence", IntegerType(), True),
                            StructField("machine", IntegerType(), True),
                            StructField("PID", IntegerType(), True),
                            StructField("TRUN", IntegerType(), True),
                            StructField("TSLPI", IntegerType(), True),
                            StructField("TSLPU", IntegerType(), True),
                            StructField("POLI", StringType(), True),
                            StructField("NICE", IntegerType(), True),
                            StructField("PRI", IntegerType(), True),
                            StructField("RTPR", IntegerType(), True),
                            StructField("CPUNR", IntegerType(), True),
                            StructField("Status", StringType(), True),
                            StructField("EXC", IntegerType(), True),
                            StructField("State", StringType(), True),
                            StructField("CPU", FloatType(), True),
                            StructField("CMD", StringType(), True),
                            StructField("ts", TimestampType(), True),
                            StructField("producer ID", IntegerType(), True)
                            ]))

memory_schema = ArrayType(StructType([
                            StructField("sequence", IntegerType(), True),
                            StructField("machine", IntegerType(), True),
                            StructField("PID", IntegerType(), True),
                            StructField("MINFLT", IntegerType(), True),
                            StructField("MAJFLT", IntegerType(), True),
                            StructField("VSTEXT", IntegerType(), True),
                            StructField("VSIZE", FloatType(), True),
                            StructField("RSIZE", FloatType(), True),
                            StructField("VGROW", FloatType(), True),
                            StructField("RGROW", FloatType(), True),
                            StructField("MEM", FloatType(), True),
                            StructField("CMD", StringType(), True),
                            StructField("ts", TimestampType(), True),
                            StructField("producer ID", IntegerType(), True)
                            ]))
# apply the schema for both
df_process = df_process.select(F.from_json(F.col("value").cast("string"), process_schema
).alias('parsed_value'))
df_memory = df_memory.select(F.from_json(F.col("value").cast("string"), memory_schema).a
lias('parsed_value'))

df_process = df_process.select(F.explode(F.col("parsed_value")).alias('unnested_value'))
df_memory = df_memory.select(F.explode(F.col("parsed_value")).alias('unnested_value'))

print("Process schema after applying from_json func.")
df_process.printSchema()
print("Memory schema after applying from_json func.")
df_memory.printSchema()

# rename all of the columns for both activities
df_process = df_process.select( F.col("unnested_value.sequence").alias("sequence"),
                                F.col("unnested_value.machine").alias("machine"),
                                F.col("unnested_value.PID").alias("PID"),
                                F.col("unnested_value.TRUN").alias("TRUN"),
                                F.col("unnested_value.TSLPI").alias("TSLPI"),
                                F.col("unnested_value.TSLPU").alias("TSLPU"),
                                F.col("unnested_value.POLI").alias("POLI"),
```

```
                                F.col("unnested_value.NICE").alias("NICE"),
                                F.col("unnested_value.PRI").alias("PRI"),
                                F.col("unnested_value.RTPR").alias("RTPR"),
                                F.col("unnested_value.CPUNR").alias("CPUNR"),
                                F.col("unnested_value.Status").alias("Status"),
                                F.col("unnested_value.EXC").alias("EXC"),
                                F.col("unnested_value.State").alias("State"),
                                F.col("unnested_value.CPU").alias("CPU"),
                                F.col("unnested_value.CMD").alias("CMD"),
                                F.col("unnested_value.ts").alias("ts"),
                                F.col("unnested_value.producer ID").alias("producer ID")
                               )

df_memory = df_memory.select( F.col("unnested_value.sequence").alias("sequence"),
                                F.col("unnested_value.machine").alias("machine"),
                                F.col("unnested_value.PID").alias("PID"),
                                F.col("unnested_value.MINFLT").alias("MINFLT"),
                                F.col("unnested_value.MAJFLT").alias("MAJFLT"),
                                F.col("unnested_value.VSTEXT").alias("VSTEXT"),
                                F.col("unnested_value.VSIZE").alias("VSIZE"),
                                F.col("unnested_value.RSIZE").alias("RSIZE"),
                                F.col("unnested_value.VGROW").alias("VGROW"),
                                F.col("unnested_value.RGROW").alias("RGROW"),
                                F.col("unnested_value.MEM").alias("MEM"),
                                F.col("unnested_value.CMD").alias("CMD"),
                                F.col("unnested_value.ts").alias("ts"),
                                F.col("unnested_value.producer ID").alias("producer ID")
                               )
print("Process schema after renaming")
df_process.printSchema()
print("Memory schema after renaming")
df_memory.printSchema()

# Parse out the common format to clean the data for both activities
intExp = r'(\d+)'
floatExp = r'(\d+)(\.)(\d+)'

# apply regex using Exp variables metioned above
df_process = df_process.select(regexp_extract('sequence', intExp, 1).cast('integer').ali
as('sequence'),
                           regexp_extract('machine', intExp, 1).cast('integer').alias('mac
hine'),
                           regexp_extract('PID', intExp, 1).cast('integer').alias('PID'),
                           regexp_extract('TRUN', intExp, 1).cast('integer').alias('TRUN')
,
                           regexp_extract('TSLPI', intExp, 1).cast('integer').alias('TSLPI
'),
                           regexp_extract('TSLPU', intExp, 1).cast('integer').alias('TSLPU
'),
                           F.col('POLI'),
                           regexp_extract('NICE', intExp, 1).cast('integer').alias('NICE')
,
                           regexp_extract('PRI', intExp, 1).cast('integer').alias('PRI'),
                           regexp_extract('RTPR', intExp, 1).cast('integer').alias('RTPR')
,
                           regexp_extract('CPUNR', intExp, 1).cast('integer').alias('CPUNR
'),
                           F.col('Status'),
                           regexp_extract('EXC', intExp, 1).cast('integer').alias('EXC'),
                           F.col('State'),
                           regexp_extract('CPU', floatExp, 3).cast('float').alias('CPU'),
                           F.col('CMD'),
                           F.col('ts'),
                           regexp_extract('producer ID', intExp, 1).cast('integer').alias(
'producer ID'))

df_memory = df_memory.select(regexp_extract('sequence', intExp, 1).cast('integer').alias
('sequence'),
                           regexp_extract('machine', intExp, 1).cast('integer').alias('mac
hine'),
                           regexp_extract('PID', intExp, 1).cast('integer').alias('PID'),
                           regexp_extract('MINFLT', intExp, 1).cast('integer').alias('MINF
```

```
LT'),
                        regexp_extract('MAJFLT', intExp, 1).cast('integer').alias('MAJF
LT'),
                        regexp_extract('VSTEXT', intExp, 1).cast('integer').alias('VSTE
XT'),
                        regexp_extract('VSIZE', floatExp, 3).cast('float').alias('VSIZE
'),
                        regexp_extract('RSIZE', floatExp, 3).cast('float').alias('RSIZE
'),
                        regexp_extract('VGROW', floatExp, 3).cast('float').alias('VGROW
'),
                        regexp_extract('RGROW', floatExp, 3).cast('float').alias('RGROW
'),
                        regexp_extract('MEM', floatExp, 3).cast('float').alias('MEM'),
                        F.col('CMD'),
                        F.col('ts'),
                        regexp_extract('producer ID', intExp, 1).cast('integer').alias(
'producer ID'))
```

```
Read-in schema for process activities
root
 |-- key: string (nullable = true)
 |-- value: string (nullable = true)

Read-in schema for memory activities
root
 |-- key: string (nullable = true)
 |-- value: string (nullable = true)

Process schema after applying from_json func.
root
 |-- unnested_value: struct (nullable = true)
 |    |-- sequence: integer (nullable = true)
 |    |-- machine: integer (nullable = true)
 |    |-- PID: integer (nullable = true)
 |    |-- TRUN: integer (nullable = true)
 |    |-- TSLPI: integer (nullable = true)
 |    |-- TSLPU: integer (nullable = true)
 |    |-- POLI: string (nullable = true)
 |    |-- NICE: integer (nullable = true)
 |    |-- PRI: integer (nullable = true)
 |    |-- RTPR: integer (nullable = true)
 |    |-- CPUNR: integer (nullable = true)
 |    |-- Status: string (nullable = true)
 |    |-- EXC: integer (nullable = true)
 |    |-- State: string (nullable = true)
 |    |-- CPU: float (nullable = true)
 |    |-- CMD: string (nullable = true)
 |    |-- ts: timestamp (nullable = true)
 |    |-- producer ID: integer (nullable = true)

Memory schema after applying from_json func.
root
 |-- unnested_value: struct (nullable = true)
 |    |-- sequence: integer (nullable = true)
 |    |-- machine: integer (nullable = true)
 |    |-- PID: integer (nullable = true)
 |    |-- MINFLT: integer (nullable = true)
 |    |-- MAJFLT: integer (nullable = true)
 |    |-- VSTEXT: integer (nullable = true)
 |    |-- VSIZE: float (nullable = true)
 |    |-- RSIZE: float (nullable = true)
 |    |-- VGROW: float (nullable = true)
 |    |-- RGROW: float (nullable = true)
 |    |-- MEM: float (nullable = true)
 |    |-- CMD: string (nullable = true)
 |    |-- ts: timestamp (nullable = true)
 |    |-- producer ID: integer (nullable = true)

Process schema after renaming
root
 |-- sequence: integer (nullable = true)
```

```
 |-- machine: integer (nullable = true)
 |-- PID: integer (nullable = true)
 |-- TRUN: integer (nullable = true)
 |-- TSLPI: integer (nullable = true)
 |-- TSLPU: integer (nullable = true)
 |-- POLI: string (nullable = true)
 |-- NICE: integer (nullable = true)
 |-- PRI: integer (nullable = true)
 |-- RTPR: integer (nullable = true)
 |-- CPUNR: integer (nullable = true)
 |-- Status: string (nullable = true)
 |-- EXC: integer (nullable = true)
 |-- State: string (nullable = true)
 |-- CPU: float (nullable = true)
 |-- CMD: string (nullable = true)
 |-- ts: timestamp (nullable = true)
 |-- producer ID: integer (nullable = true)

Memory schema after renaming
root
 |-- sequence: integer (nullable = true)
 |-- machine: integer (nullable = true)
 |-- PID: integer (nullable = true)
 |-- MINFLT: integer (nullable = true)
 |-- MAJFLT: integer (nullable = true)
 |-- VSTEXT: integer (nullable = true)
 |-- VSIZE: float (nullable = true)
 |-- RSIZE: float (nullable = true)
 |-- VGROW: float (nullable = true)
 |-- RGROW: float (nullable = true)
 |-- MEM: float (nullable = true)
 |-- CMD: string (nullable = true)
 |-- ts: timestamp (nullable = true)
 |-- producer ID: integer (nullable = true)
```

In [4]:

```
# --- 3.3 Mapping PRI/NICE --- #
from pyspark.sql.functions import udf, concat, lit, when, col
# mapping column NICE based on the value of column PRI
df_process = df_process.withColumn("NICE", when((140 > col("PRI")) & (col("PRI") > 99),
col("PRI")))
df_process = df_process.withColumn("NICE", when((140 > col("NICE")) & (col("NICE") > 99)
, col("NICE")-120))
```

In [5]:

```
# --- check the PRI/NICE mapping --- #
from time import sleep
dfq = df_process.select("PRI", "NICE")
test = dfq \
    .writeStream \
    .outputMode("append") \
    .format("memory") \
    .queryName("dfq") \
    .trigger(processingTime='5 seconds') \
    .start()
while True:
    spark.sql("select * from dfq").show(truncate=False)
    sleep(5)
```

```
+---+----+
|PRI|NICE|
+---+----+
+---+----+

+---+----+
|PRI|NICE|
+---+----+
+---+----+
```

```
+---+----+
|PRI|NICE|
+---+----+
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|100|-20 |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
+---+----+
only showing top 20 rows

+---+----+
|PRI|NICE|
+---+----+
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|100|-20 |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
+---+----+
only showing top 20 rows

+---+----+
|PRI|NICE|
+---+----+
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|100|-20 |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
```

```
|120|0   |
|120|0   |
|120|0   |
+---+----+
only showing top 20 rows

+---+----+
|PRI|NICE|
+---+----+
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|100|-20 |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
+---+----+
only showing top 20 rows

+---+----+
|PRI|NICE|
+---+----+
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|100|-20 |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
+---+----+
only showing top 20 rows

+---+----+
|PRI|NICE|
+---+----+
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
```

```
|100|-20 |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
+---+----+
only showing top 20 rows

+---+----+
|PRI|NICE|
+---+----+
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|100|-20 |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
|120|0   |
+---+----+
only showing top 20 rows


---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-5-899778dacb94> in <module>
     11 while True:
     12     spark.sql("select * from dfq").show(truncate=False)
---> 13     sleep(5)

KeyboardInterrupt:
```

In [6]:

```
# stop testing query for PRI/NICE mapping
test.stop()
```

In [7]:

```
## --- 3.4 --- ##
from pyspark.sql.functions import udf, concat, lit

# create a new column CMD_PID for process and memory activities
df_process = df_process.withColumn("CMD_PID", concat(col("CMD"), lit(" "), col("PID")))
df_memory = df_memory.withColumn("CMD_PID", concat(col("CMD"), lit(" "), col("PID")))

# create a new column event_time for process and memory activities and apply watermark
df_process = df_process.withColumn("event_time", col("ts"))\
                       .withWatermark("event_time", '20 seconds')
df_memory = df_memory.withColumn("event_time", col("ts"))\
                       .withWatermark("event_time", '20 seconds')
```

In [8]:

```
## --- 3.5 --- ##
```

```
# write stream to parquet files
process_query_file_sink = df_process.writeStream.format("parquet")\
                          .outputMode("append")\
                          .option("path", "process.parquet/df_process")\
                          .option("checkpointLocation", "process.parquet/df_process/checkp
oint")\
                          .start()

memory_query_file_sink = df_process.writeStream.format("parquet")\
                          .outputMode("append")\
                          .option("path", "memory.parquet/df_process")\
                          .option("checkpointLocation", "memory.parquet/df_process/checkpo
int")\
                          .start()
```

In [10]:

```
## --- 3.6 --- ##

#unzip the models
#import zipfile
#with zipfile.ZipFile("process_pipeline_model.zip", 'r') as zip_ref:
#    zip_ref.extractall()
#with zipfile.ZipFile("memory_pipeline_model.zip", 'r') as zip_ref:
#    zip_ref.extractall()

# load process and memory models
from pyspark.ml.pipeline import PipelineModel
process_model = PipelineModel.load("process_pipeline_model")
memory_model = PipelineModel.load("memory_pipeline_model")

# generate prediction for both
process_prediction = process_model.transform(df_process)
memory_prediction = memory_model.transform(df_memory)
```

In [38]:

```
## --- 3.7.a --- ##
# group by 2-min window and drop duplicated CMD_PID in a 2-min window
process_grouped_by_win = process_prediction \
                          .where(col("prediction") == 1)\
                          .groupBy(F.window(process_prediction.ts, "2 minutes"), process_p
rediction.machine)\
                          .agg(F.approx_count_distinct("CMD_PID").alias("total attack pred
iction"))\
                          .select("window", "machine", "total attack prediction")
# write stream with complete mode to memory
process_detection_query = process_grouped_by_win \
                          .writeStream \
                          .outputMode("complete") \
                          .format("memory") \
                          .queryName("process_attack_detection") \
                          .trigger(processingTime='5 seconds') \
                          .start()
# group by 2-min window and drop duplicated CMD_PID in a 2-min window
memory_grouped_by_win = memory_prediction \
                          .where(col("prediction") == 1)\
                          .groupBy(F.window(memory_prediction.ts, "2 minutes"), memory_pre
diction.machine)\
                          .agg(F.approx_count_distinct("CMD_PID").alias("total attack pred
iction"))\
                          .select("window", "machine", "total attack prediction")
# write stream with complete mode to memory
memory_detection_query = memory_grouped_by_win \
                          .writeStream \
                          .outputMode("complete") \
                          .format("memory") \
                          .queryName("memory_attack_detection") \
                          .trigger(processingTime='5 seconds') \
                          .start()
```

In [39]:

```
from time import sleep
# show some results of memory sinks for both activities
while True:
    print("Process Detection")
    spark.sql("select * from process_attack_detection").show(truncate=False)
    print("Memory Detection")
    spark.sql("select * from memory_attack_detection").show(truncate=False)
    sleep(10)
```

```
Process Detection
+------+-------+----------------------+
|window|machine|total attack prediction|
+------+-------+----------------------+
+------+-------+----------------------+

Memory Detection
+------+-------+----------------------+
|window|machine|total attack prediction|
+------+-------+----------------------+
+------+-------+----------------------+

Process Detection
+------+-------+----------------------+
|window|machine|total attack prediction|
+------+-------+----------------------+
+------+-------+----------------------+

Memory Detection
+------+-------+----------------------+
|window|machine|total attack prediction|
+------+-------+----------------------+
+------+-------+----------------------+

Process Detection
+-----------------------------------------+-------+----------------------+
|window                                   |machine|total attack prediction|
+-----------------------------------------+-------+----------------------+
|[2020-10-31 15:12:00, 2020-10-31 15:14:00]|7      |10                    |
|[2020-10-31 15:12:00, 2020-10-31 15:14:00]|5      |15                    |
|[2020-10-31 15:12:00, 2020-10-31 15:14:00]|8      |10                    |
+-----------------------------------------+-------+----------------------+

Memory Detection
+-----------------------------------------+-------+----------------------+
|window                                   |machine|total attack prediction|
+-----------------------------------------+-------+----------------------+
|[2020-10-31 15:12:00, 2020-10-31 15:14:00]|7      |11                    |
|[2020-10-31 15:12:00, 2020-10-31 15:14:00]|4      |6                     |
|[2020-10-31 15:12:00, 2020-10-31 15:14:00]|5      |11                    |
|[2020-10-31 15:12:00, 2020-10-31 15:14:00]|8      |7                     |
+-----------------------------------------+-------+----------------------+

Process Detection
+-----------------------------------------+-------+----------------------+
|window                                   |machine|total attack prediction|
+-----------------------------------------+-------+----------------------+
|[2020-10-31 15:12:00, 2020-10-31 15:14:00]|7      |32                    |
|[2020-10-31 15:12:00, 2020-10-31 15:14:00]|5      |27                    |
|[2020-10-31 15:12:00, 2020-10-31 15:14:00]|8      |28                    |
+-----------------------------------------+-------+----------------------+

Memory Detection
+-----------------------------------------+-------+----------------------+
|window                                   |machine|total attack prediction|
+-----------------------------------------+-------+----------------------+
|[2020-10-31 15:12:00, 2020-10-31 15:14:00]|7      |21                    |
|[2020-10-31 15:12:00, 2020-10-31 15:14:00]|4      |6                     |
|[2020-10-31 15:12:00, 2020-10-31 15:14:00]|5      |14                    |
|[2020-10-31 15:12:00, 2020-10-31 15:14:00]|8      |7                     |
+-----------------------------------------+-------+----------------------+
```

Process Detection

window	machine	total attack prediction
[2020-10-31 15:12:00, 2020-10-31 15:14:00]	7	32
[2020-10-31 15:12:00, 2020-10-31 15:14:00]	5	29
[2020-10-31 15:12:00, 2020-10-31 15:14:00]	8	39

Memory Detection

window	machine	total attack prediction
[2020-10-31 15:12:00, 2020-10-31 15:14:00]	7	22
[2020-10-31 15:12:00, 2020-10-31 15:14:00]	4	8
[2020-10-31 15:12:00, 2020-10-31 15:14:00]	5	15
[2020-10-31 15:12:00, 2020-10-31 15:14:00]	8	11

Process Detection

window	machine	total attack prediction
[2020-10-31 15:12:00, 2020-10-31 15:14:00]	7	32
[2020-10-31 15:12:00, 2020-10-31 15:14:00]	5	41
[2020-10-31 15:12:00, 2020-10-31 15:14:00]	8	47

Memory Detection

window	machine	total attack prediction
[2020-10-31 15:12:00, 2020-10-31 15:14:00]	7	32
[2020-10-31 15:12:00, 2020-10-31 15:14:00]	4	10
[2020-10-31 15:12:00, 2020-10-31 15:14:00]	5	16
[2020-10-31 15:12:00, 2020-10-31 15:14:00]	8	11

Process Detection

window	machine	total attack prediction
[2020-10-31 15:12:00, 2020-10-31 15:14:00]	7	32
[2020-10-31 15:12:00, 2020-10-31 15:14:00]	5	41
[2020-10-31 15:12:00, 2020-10-31 15:14:00]	8	47

Memory Detection

window	machine	total attack prediction
[2020-10-31 15:12:00, 2020-10-31 15:14:00]	7	47
[2020-10-31 15:12:00, 2020-10-31 15:14:00]	4	11
[2020-10-31 15:12:00, 2020-10-31 15:14:00]	5	17
[2020-10-31 15:12:00, 2020-10-31 15:14:00]	8	11

Process Detection

window	machine	total attack prediction
[2020-10-31 15:12:00, 2020-10-31 15:14:00]	7	32
[2020-10-31 15:12:00, 2020-10-31 15:14:00]	5	42
[2020-10-31 15:12:00, 2020-10-31 15:14:00]	8	47

Memory Detection

window	machine	total attack prediction
[2020-10-31 15:12:00, 2020-10-31 15:14:00]	7	48
[2020-10-31 15:12:00, 2020-10-31 15:14:00]	4	11

```
|[2020-10-31 15:12:00, 2020-10-31 15:14:00]|5      |17                     |
|[2020-10-31 15:12:00, 2020-10-31 15:14:00]|8      |14                     |
+-----------------------------------------+-------+-----------------------+

Process Detection
+-----------------------------------------+-------+-----------------------+
|window                                   |machine|total attack prediction|
+-----------------------------------------+-------+-----------------------+
|[2020-10-31 15:14:00, 2020-10-31 15:16:00]|7      |14                     |
|[2020-10-31 15:12:00, 2020-10-31 15:14:00]|7      |32                     |
|[2020-10-31 15:14:00, 2020-10-31 15:16:00]|8      |26                     |
|[2020-10-31 15:12:00, 2020-10-31 15:14:00]|5      |42                     |
|[2020-10-31 15:12:00, 2020-10-31 15:14:00]|8      |47                     |
|[2020-10-31 15:14:00, 2020-10-31 15:16:00]|5      |17                     |
+-----------------------------------------+-------+-----------------------+

Memory Detection
+-----------------------------------------+-------+-----------------------+
|window                                   |machine|total attack prediction|
+-----------------------------------------+-------+-----------------------+
|[2020-10-31 15:14:00, 2020-10-31 15:16:00]|7      |11                     |
|[2020-10-31 15:12:00, 2020-10-31 15:14:00]|7      |48                     |
|[2020-10-31 15:12:00, 2020-10-31 15:14:00]|4      |11                     |
|[2020-10-31 15:14:00, 2020-10-31 15:16:00]|8      |3                      |
|[2020-10-31 15:12:00, 2020-10-31 15:14:00]|5      |17                     |
|[2020-10-31 15:12:00, 2020-10-31 15:14:00]|8      |14                     |
|[2020-10-31 15:14:00, 2020-10-31 15:16:00]|4      |21                     |
|[2020-10-31 15:14:00, 2020-10-31 15:16:00]|5      |11                     |
+-----------------------------------------+-------+-----------------------+

Process Detection

---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-39-8bb78d917bf2> in <module>
      3 while True:
      4     print("Process Detection")
----> 5     spark.sql("select * from process_attack_detection").show(truncate=False)
      6     print("Memory Detection")
      7     spark.sql("select * from memory_attack_detection").show(truncate=False)

~/.local/lib/python3.8/site-packages/pyspark/sql/dataframe.py in show(self, n, truncate,
vertical)
    440             print(self._jdf.showString(n, 20, vertical))
    441         else:
--> 442             print(self._jdf.showString(n, int(truncate), vertical))
    443
    444     def __repr__(self):

~/.local/lib/python3.8/site-packages/py4j/java_gateway.py in __call__(self, *args)
   1301             proto.END_COMMAND_PART
   1302
-> 1303         answer = self.gateway_client.send_command(command)
   1304         return_value = get_return_value(
   1305             answer, self.gateway_client, self.target_id, self.name)

~/.local/lib/python3.8/site-packages/py4j/java_gateway.py in send_command(self, command,
retry, binary)
   1031         connection = self._get_connection()
   1032         try:
-> 1033             response = connection.send_command(command)
   1034             if binary:
   1035                 return response, self._create_connection_guard(connection)

~/.local/lib/python3.8/site-packages/py4j/java_gateway.py in send_command(self, command)
   1198
   1199         try:
-> 1200             answer = smart_decode(self.stream.readline()[:-1])
   1201             logger.debug("Answer received: {0}".format(answer))
   1202             if answer.startswith(proto.RETURN_MESSAGE):

/usr/lib/python3.8/socket.py in readinto(self, b)
```

```
    667         while True:
    668             try:
--> 669                 return self._sock.recv_into(b)
    670             except timeout:
    671                 self._timeout_occurred = True

KeyboardInterrupt:
```

In [40]:

```
process_detection_query.stop()
memory_detection_query.stop()
```

In [41]:

```
## --- 3.7.b --- ##
from datetime import datetime
from pyspark.sql.functions import expr

#SELECT *
#FROM process_prediction AS pp
#JOIN memory_prediction AS mp
#ON pp.CMD_PID = mp.CMD_PID
#WHERE pp.prediction = 1 AND mp.prediction = 1

#-- The one below doesn't work because the subquery is an infinite stream and
#-- you can't check if an element exists in an infinite stream.
#SELECT *
#FROM process_prediction
#WHERE CMD_PID IN (SELECT CMD_PID FROM memory_prediction WHERE prediction = 1)
#AND prediction = 1

def add_prefix(sdf, prefix):
    for c in sdf.columns:
        sdf = sdf.withColumnRenamed(c, '{}{}'.format(prefix, c))
    return sdf


process_stream = process_prediction \
                .withWatermark("event_time", "30 seconds") \
                .where(col("prediction") == 1)

memory_stream = memory_prediction \
                .withWatermark("event_time", "30 seconds") \
                .where(col("prediction") == 1)

# add prefix to the columns in memory_stream to avoid ambiguous references after joining
memory_stream = add_prefix(memory_stream, "Memory_")

joined_stream = process_stream \
                .join(memory_stream, expr("""CMD_PID == Memory_CMD_PID"""), "inner")\
                .filter(F.abs(process_stream.event_time.cast('long')-memory_stream.Memor
y_event_time.cast('long'))<30)\
                .select(process_stream["*"], memory_stream["*"])\

# add column processing_time
joined_stream = joined_stream.withColumn('processing_time', F.current_timestamp())

# check the joined_stream
process_memory_query =  joined_stream \
                        .writeStream \
                        .outputMode("append") \
                        .format("memory") \
                        .queryName("real_attack_alarm") \
                        .trigger(processingTime='5 seconds') \
                        .start()

# check the columns are renamed properly
joined_stream.printSchema()
```

```
root
 |-- sequence: integer (nullable = true)
```

```
 |-- machine: integer (nullable = true)
 |-- PID: integer (nullable = true)
 |-- TRUN: integer (nullable = true)
 |-- TSLPI: integer (nullable = true)
 |-- TSLPU: integer (nullable = true)
 |-- POLI: string (nullable = true)
 |-- NICE: integer (nullable = true)
 |-- PRI: integer (nullable = true)
 |-- RTPR: integer (nullable = true)
 |-- CPUNR: integer (nullable = true)
 |-- Status: string (nullable = true)
 |-- EXC: integer (nullable = true)
 |-- State: string (nullable = true)
 |-- CPU: float (nullable = true)
 |-- CMD: string (nullable = true)
 |-- ts: timestamp (nullable = true)
 |-- producer ID: integer (nullable = true)
 |-- CMD_PID: string (nullable = true)
 |-- event_time: timestamp (nullable = true)
 |-- POLI_idx: double (nullable = false)
 |-- Status_idx: double (nullable = false)
 |-- State_idx: double (nullable = false)
 |-- POLI_vec: vector (nullable = true)
 |-- Status_vec: vector (nullable = true)
 |-- State_vec: vector (nullable = true)
 |-- tmp_CMD: array (nullable = true)
 |    |-- element: string (containsNull = true)
 |-- CMD_vec: vector (nullable = true)
 |-- features: vector (nullable = true)
 |-- scaled_features: vector (nullable = true)
 |-- rawPrediction: vector (nullable = true)
 |-- probability: vector (nullable = true)
 |-- prediction: double (nullable = false)
 |-- Memory_sequence: integer (nullable = true)
 |-- Memory_machine: integer (nullable = true)
 |-- Memory_PID: integer (nullable = true)
 |-- Memory_MINFLT: integer (nullable = true)
 |-- Memory_MAJFLT: integer (nullable = true)
 |-- Memory_VSTEXT: integer (nullable = true)
 |-- Memory_VSIZE: float (nullable = true)
 |-- Memory_RSIZE: float (nullable = true)
 |-- Memory_VGROW: float (nullable = true)
 |-- Memory_RGROW: float (nullable = true)
 |-- Memory_MEM: float (nullable = true)
 |-- Memory_CMD: string (nullable = true)
 |-- Memory_ts: timestamp (nullable = true)
 |-- Memory_producer ID: integer (nullable = true)
 |-- Memory_CMD_PID: string (nullable = true)
 |-- Memory_event_time: timestamp (nullable = true)
 |-- Memory_tmp_CMD: array (nullable = true)
 |    |-- element: string (containsNull = true)
 |-- Memory_CMD_vec: vector (nullable = true)
 |-- Memory_features: vector (nullable = true)
 |-- Memory_scaled_features: vector (nullable = true)
 |-- Memory_rawPrediction: vector (nullable = true)
 |-- Memory_probability: vector (nullable = true)
 |-- Memory_prediction: double (nullable = false)
 |-- processing_time: timestamp (nullable = false)
```

In [42]:

```
while True:
    spark.sql("select event_time, Memory_event_time, CMD_PID, Memory_CMD_PID, processing_
time from real_attack_alarm")\
             .show(truncate=False)
    sleep(10)
```

```
+----------+-----------------+-------+--------------+---------------+
|event_time|Memory_event_time|CMD_PID|Memory_CMD_PID|processing_time|
+----------+-----------------+-------+--------------+---------------+
+----------+-----------------+-------+--------------+---------------+
```

```
+----------+-----------------+-------+-------------+---------------+
|event_time|Memory_event_time|CMD_PID|Memory_CMD_PID|processing_time|
+----------+-----------------+-------+-------------+---------------+
+----------+-----------------+-------+-------------+---------------+

+----------+-----------------+-------+-------------+---------------+
|event_time|Memory_event_time|CMD_PID|Memory_CMD_PID|processing_time|
+----------+-----------------+-------+-------------+---------------+
+----------+-----------------+-------+-------------+---------------+

+----------+-----------------+-------+-------------+---------------+
|event_time|Memory_event_time|CMD_PID|Memory_CMD_PID|processing_time|
+----------+-----------------+-------+-------------+---------------+
+----------+-----------------+-------+-------------+---------------+

+----------+-----------------+-------+-------------+---------------+
|event_time|Memory_event_time|CMD_PID|Memory_CMD_PID|processing_time|
+----------+-----------------+-------+-------------+---------------+
+----------+-----------------+-------+-------------+---------------+

+----------+-----------------+-------+-------------+---------------+
|event_time|Memory_event_time|CMD_PID|Memory_CMD_PID|processing_time|
+----------+-----------------+-------+-------------+---------------+
+----------+-----------------+-------+-------------+---------------+

+-------------------+-------------------+-------------------+-------------------+------
-----------------+
|event_time         |Memory_event_time  |CMD_PID            |Memory_CMD_PID     |process
ing_time        |
+-------------------+-------------------+-------------------+-------------------+------
-----------------+
|2020-10-31 15:16:13|2020-10-31 15:16:07|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:07|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:33|2020-10-31 15:16:07|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:17|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:17|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:33|2020-10-31 15:16:17|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:38|2020-10-31 15:16:37|unity-panel-se 3534|unity-panel-se 3534|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:07|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:07|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:07|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:37|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:37|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:37|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:18|2020-10-31 15:16:07|atop 3294          |atop 3294          |2020-10-
31 15:16:39.341|
```

```
+-------------------+-------------------+-------------------+-------------------+------
-----------------+
only showing top 20 rows

+-------------------+-------------------+-------------------+-------------------+------
-----------------+
|event_time         |Memory_event_time  |CMD_PID            |Memory_CMD_PID     |process
ing_time        |
+-------------------+-------------------+-------------------+-------------------+------
-----------------+
|2020-10-31 15:16:13|2020-10-31 15:16:07|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:07|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:33|2020-10-31 15:16:07|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:17|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:17|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:33|2020-10-31 15:16:17|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:38|2020-10-31 15:16:37|unity-panel-se 3534|unity-panel-se 3534|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:07|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:07|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:07|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:37|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:37|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:37|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:18|2020-10-31 15:16:07|atop 3294          |atop 3294          |2020-10-
31 15:16:39.341|
+-------------------+-------------------+-------------------+-------------------+------
-----------------+
only showing top 20 rows

+-------------------+-------------------+-------------------+-------------------+------
-----------------+
|event_time         |Memory_event_time  |CMD_PID            |Memory_CMD_PID     |process
ing_time        |
+-------------------+-------------------+-------------------+-------------------+------
-----------------+
|2020-10-31 15:16:13|2020-10-31 15:16:07|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:07|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:33|2020-10-31 15:16:07|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:17|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:17|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:33|2020-10-31 15:16:17|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
```

```
|2020-10-31 15:16:38|2020-10-31 15:16:37|unity-panel-se 3534|unity-panel-se 3534|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:07|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:07|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:07|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:37|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:37|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:37|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:18|2020-10-31 15:16:07|atop 3294          |atop 3294          |2020-10-
31 15:16:39.341|
+-------------------+-------------------+-------------------+-------------------+------
-----------------+
only showing top 20 rows

+-------------------+-------------------+-------------------+-------------------+------
-----------------+
|event_time         |Memory_event_time  |CMD_PID            |Memory_CMD_PID     |process
ing_time        |
+-------------------+-------------------+-------------------+-------------------+------
-----------------+
|2020-10-31 15:16:13|2020-10-31 15:16:07|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:07|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:33|2020-10-31 15:16:07|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:17|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:17|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:33|2020-10-31 15:16:17|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:38|2020-10-31 15:16:37|unity-panel-se 3534|unity-panel-se 3534|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:07|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:07|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:07|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:37|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
```

```
|2020-10-31 15:16:23|2020-10-31 15:16:37|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:37|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:18|2020-10-31 15:16:07|atop 3294          |atop 3294          |2020-10-
31 15:16:39.341|
+-------------------+-------------------+-------------------+-------------------+------
-----------------+
only showing top 20 rows

+-------------------+-------------------+-------------------+-------------------+------
-----------------+
|event_time         |Memory_event_time  |CMD_PID            |Memory_CMD_PID     |process
ing_time        |
+-------------------+-------------------+-------------------+-------------------+------
-----------------+
|2020-10-31 15:16:13|2020-10-31 15:16:07|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:07|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:33|2020-10-31 15:16:07|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:17|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:17|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:33|2020-10-31 15:16:17|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:38|2020-10-31 15:16:37|unity-panel-se 3534|unity-panel-se 3534|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:07|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:07|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:07|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:37|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:37|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:37|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:18|2020-10-31 15:16:07|atop 3294          |atop 3294          |2020-10-
31 15:16:39.341|
+-------------------+-------------------+-------------------+-------------------+------
-----------------+
only showing top 20 rows

+-------------------+-------------------+-------------------+-------------------+------
-----------------+
|event_time         |Memory_event_time  |CMD_PID            |Memory_CMD_PID     |process
ing_time        |
+-------------------+-------------------+-------------------+-------------------+------
-----------------+
|2020-10-31 15:16:13|2020-10-31 15:16:07|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:07|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:33|2020-10-31 15:16:07|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
```

```
|2020-10-31 15:16:13|2020-10-31 15:16:17|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:17|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:33|2020-10-31 15:16:17|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:38|2020-10-31 15:16:37|unity-panel-se 3534|unity-panel-se 3534|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:07|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:07|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:07|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:37|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:37|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:37|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:18|2020-10-31 15:16:07|atop 3294          |atop 3294          |2020-10-
31 15:16:39.341|
+-------------------+-------------------+-------------------+-------------------+------
-----------------+
only showing top 20 rows

+-------------------+-------------------+-------------------+-------------------+------
-----------------+
|event_time         |Memory_event_time  |CMD_PID            |Memory_CMD_PID     |process
ing_time        |
+-------------------+-------------------+-------------------+-------------------+------
-----------------+
|2020-10-31 15:16:13|2020-10-31 15:16:07|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:07|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:33|2020-10-31 15:16:07|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:17|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:17|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:33|2020-10-31 15:16:17|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:38|2020-10-31 15:16:37|unity-panel-se 3534|unity-panel-se 3534|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:07|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:07|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:07|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
```

```
|2020-10-31 15:16:23|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:37|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:37|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:37|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:18|2020-10-31 15:16:07|atop 3294          |atop 3294          |2020-10-
31 15:16:39.341|
+-------------------+-------------------+-------------------+-------------------+------
-----------------+
only showing top 20 rows

+-------------------+-------------------+-------------------+-------------------+------
-----------------+
|event_time         |Memory_event_time  |CMD_PID            |Memory_CMD_PID     |process
ing_time        |
+-------------------+-------------------+-------------------+-------------------+------
-----------------+
|2020-10-31 15:16:13|2020-10-31 15:16:07|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:07|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:33|2020-10-31 15:16:07|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:17|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:17|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:33|2020-10-31 15:16:17|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:38|2020-10-31 15:16:37|unity-panel-se 3534|unity-panel-se 3534|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:07|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:07|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:07|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:37|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:37|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:37|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:18|2020-10-31 15:16:07|atop 3294          |atop 3294          |2020-10-
31 15:16:39.341|
+-------------------+-------------------+-------------------+-------------------+------
-----------------+
only showing top 20 rows

+-------------------+-------------------+-------------------+-------------------+------
-----------------+
|event_time         |Memory_event_time  |CMD_PID            |Memory_CMD_PID     |process
ing_time        |
+-------------------+-------------------+-------------------+-------------------+------
-----------------+
```

```
|2020-10-31 15:16:13|2020-10-31 15:16:07|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:07|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:33|2020-10-31 15:16:07|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:17|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:17|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:33|2020-10-31 15:16:17|gnome-terminal 3058|gnome-terminal 3058|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:38|2020-10-31 15:16:37|unity-panel-se 3534|unity-panel-se 3534|2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:07|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:07|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:07|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:17|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:13|2020-10-31 15:16:37|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:23|2020-10-31 15:16:37|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:28|2020-10-31 15:16:37|vmtoolsd 2882      |vmtoolsd 2882      |2020-10-
31 15:16:39.341|
|2020-10-31 15:16:18|2020-10-31 15:16:07|atop 3294          |atop 3294          |2020-10-
31 15:16:39.341|
+-------------------+-------------------+-------------------+-------------------+------
-----------------+
only showing top 20 rows


---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-42-f11d5581a087> in <module>
      2     spark.sql("select event_time, Memory_event_time, CMD_PID, Memory_CMD_PID, pro
cessing_time from real_attack_alarm")\
      3             .show(truncate=False)
----> 4     sleep(10)

KeyboardInterrupt:
```

In [43]:

```
# stop checking joined_stream
process_memory_query.stop()
```

In [44]:

```
## --- 3.7.b Persist Data To Parquet Files --- ##
joined_stream_file_sink = joined_stream.writeStream.format("parquet")\
                        .outputMode("append")\
                        .option("path", "process_memory_attack.parquet/df_joined")\
                        .option("checkpointLocation", "process_memory_attack.parquet/df_
joined/checkpoint")\
                        .start()
```

In [52]:

```
## --- 3.8 Write Stream --- ##

# group by 2-min window and drop duplicated CMD_PID in a 2-min window
process_attack = process_prediction \
                .where(col("prediction") == 1)\
                .groupBy(F.window(process_prediction.ts, "2 minutes"), process_predictio
n.machine)\
                .agg(F.approx_count_distinct("CMD_PID").alias("total attack prediction")
)\
                .select("window", "machine", "total attack prediction")

# group by 2-min window and drop duplicated CMD_PID in a 2-min window
memory_attack = memory_prediction \
                .where(col("prediction") == 1)\
                .groupBy(F.window(memory_prediction.ts, "2 minutes"), memory_prediction.
machine)\
                .agg(F.approx_count_distinct("CMD_PID").alias("total attack prediction")
)\
                .select("window", "machine", "total attack prediction")

# PROCESS memory sink
process_detection_query = process_attack \
                          .writeStream \
                          .outputMode("complete") \
                          .format("memory") \
                          .queryName("process_attack_detection") \
                          .trigger(processingTime='5 seconds') \
                          .start()

# MEMORY memory sink
memory_detection_query = memory_grouped_by_win \
                          .writeStream \
                          .outputMode("complete") \
                          .format("memory") \
                          .queryName("memory_attack_detection") \
                          .trigger(processingTime='5 seconds') \
                          .start()
```

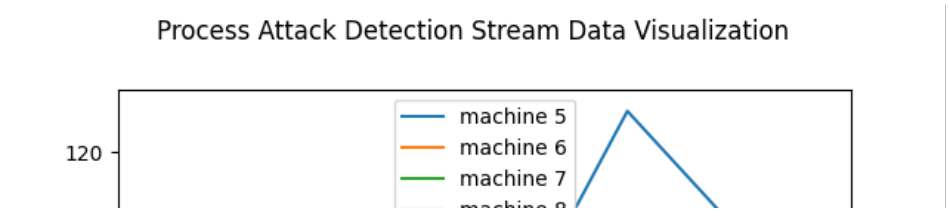
In [56]:

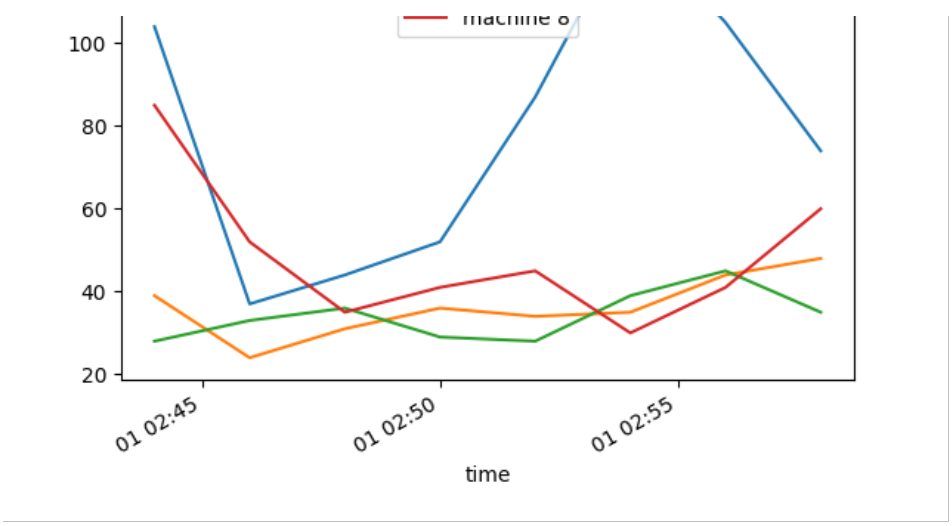
```
## --- 3.8 Catch Stream and Plot Line Chart For Process Activities --- ##
import matplotlib
import matplotlib.pyplot as plt

# show the plot in notebook
%matplotlib notebook

fig = plt.figure()
ax = plt.gca()
fig.show()
fig.canvas.draw()
fig.suptitle('Process Attack Detection Stream Data Visualization')

while True:
    plot = spark.sql("select * from process_attack_detection").toPandas()
    ax.clear()
    plot['time'] = plot['window'].map(lambda x: x[0])
    for machine, group in plot.groupby('machine'):
        group.plot(x='time', y='total attack prediction', label="machine "+str(machine),
ax=ax)
    #ax.legend(loc="upper left")
    fig.canvas.draw()
    sleep(600)
```






```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-56-5181d6b85f1d> in <module>
     20     #ax.legend(loc="upper left")
     21     fig.canvas.draw()
---> 22     sleep(600)

KeyboardInterrupt:
```

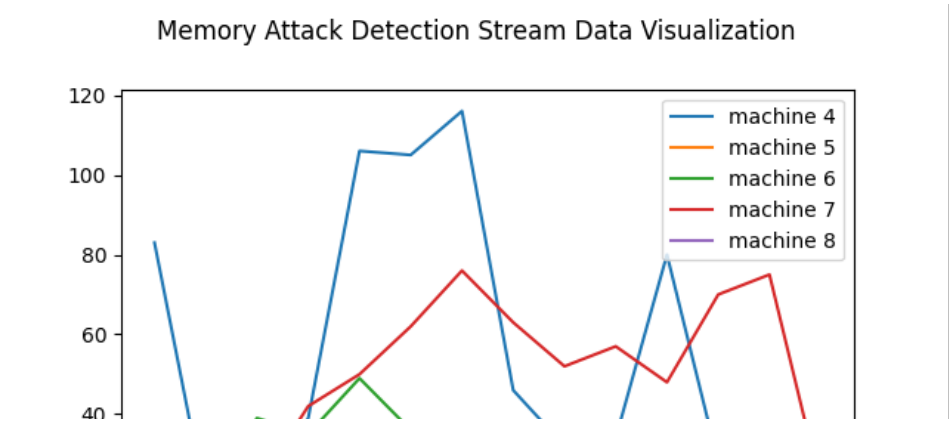
In [59]:

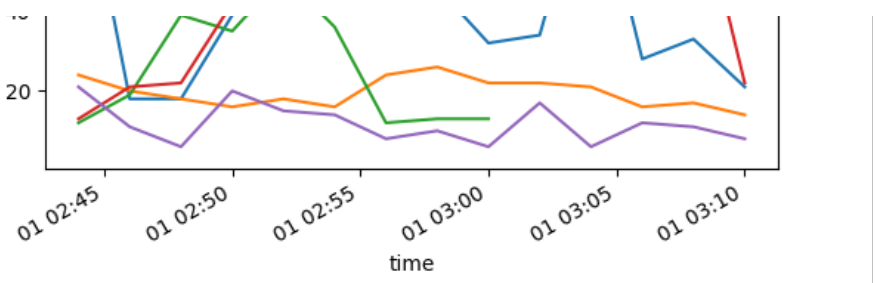
```
## --- 3.8 Catch Stream and Plot Line Chart For Memory Activities --- ##
import matplotlib
import matplotlib.pyplot as plt

# show the plot in notebook
%matplotlib notebook

fig = plt.figure()
ax = plt.gca()
fig.show()
fig.canvas.draw()
fig.suptitle('Memory Attack Detection Stream Data Visualization')

while True:
    plot = spark.sql("select * from memory_attack_detection").toPandas()
    ax.clear()
    plot['time'] = plot['window'].map(lambda x: x[0])
    for machine, group in plot.groupby('machine'):
        group.plot(x='time', y='total attack prediction', label="machine "+str(machine),
ax=ax)
    ax.legend()
    fig.canvas.draw()
    sleep(600)
```






```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-59-48f9c5c2b0c4> in <module>
     20     ax.legend()
     21     fig.canvas.draw()
---> 22     sleep(600)

KeyboardInterrupt:
```

In [60]:

```
process_detection_query.stop()
memory_detection_query.stop()
```