Antonio Gallego Peñalver

1 Cargar el archivo en un RDD

[0]:
```
from pyspark import SparkContext, SparkConf

sc = SparkContext.getOrCreate()

rdd_logs = sc.textFile("dbfs:/FileStore/firewall_log.txt")

rdd_logs.take(5)
```

```
Out[22]: ['Feb 20 14:05:29 host IPTABLES: SRC=192.168.1.100 DPT=1007
PROTO=TCP
LEN=112 TOS=0x00 PREC=0x00 TTL=90 ID=39938',
 'Feb 20 03:12:57 host IPTABLES: SRC=192.168.1.14 DPT=21 PROTO=TCP
 LEN=117
TOS=0x00 PREC=0x00 TTL=93 ID=2860 ADMIN: Cambio de regla ejecutado',
 'Feb 20 22:35:59 host IPTABLES: SRC=192.168.1.150 DPT=3389 PROTO=TCP
 LEN=135
TOS=0x00 PREC=0x00 TTL=102 ID=48471 ADMIN: Cambio de regla
ejecutado',
 'Feb 20 17:29:26 host IPTABLES: SRC=192.168.1.9 DPT=8080 PROTO=TCP
 LEN=102
TOS=0x00 PREC=0x00 TTL=57 ID=15999',
 'Feb 20 16:51:06 host IPTABLES: SRC=192.168.1.85 DPT=443 PROTO=TCP
LEN=99 TOS=0x00 PREC=0x00 TTL=97 ID=48716']
```

2 Filtrado de Eventos de Tráfico

[0]:
```
rdd_trafico = rdd_logs.filter(lambda line: "IPTABLES:" in line)

rdd_trafico.take(5)
```

```
Out[23]: ['Feb 20 14:05:29 host IPTABLES: SRC=192.168.1.100 DPT=1007
PROTO=TCP
LEN=112 TOS=0x00 PREC=0x00 TTL=90 ID=39938',
 'Feb 20 03:12:57 host IPTABLES: SRC=192.168.1.14 DPT=21 PROTO=TCP
 LEN=117
TOS=0x00 PREC=0x00 TTL=93 ID=2860 ADMIN: Cambio de regla ejecutado',
 'Feb 20 22:35:59 host IPTABLES: SRC=192.168.1.150 DPT=3389 PROTO=TCP
 LEN=135
TOS=0x00 PREC=0x00 TTL=102 ID=48471 ADMIN: Cambio de regla
ejecutado',
 'Feb 20 17:29:26 host IPTABLES: SRC=192.168.1.9 DPT=8080 PROTO=TCP
 LEN=102
TOS=0x00 PREC=0x00 TTL=57 ID=15999',
 'Feb 20 16:51:06 host IPTABLES: SRC=192.168.1.85 DPT=443 PROTO=TCP
LEN=99 TOS=0x00 PREC=0x00 TTL=97 ID=48716']
```

3 Parsear los Datos

[0]:
```
import re

def parse_line(line):
    parts = line.split()

    timestamp = " ".join(parts[:3])

    src_match = re.search(r"SRC=([\d\.]+)", line)
    src_ip = src_match.group(1) if src_match else "UNKNOWN"

    dpt_match = re.search(r"DPT=([\d;]+)", line)
    dpt_ports = dpt_match.group(1).split(";") if dpt_match else []

    return [(src_ip, (timestamp, port)) for port in dpt_ports]

rdd_parsed = rdd_trafico.flatMap(parse_line)

rdd_parsed.take(10)
```

```
Out[24]: [('192.168.1.100', ('Feb 20 14:05:29', '1007')),
 ('192.168.1.14', ('Feb 20 03:12:57', '21')),
 ('192.168.1.150', ('Feb 20 22:35:59', '3389')),
 ('192.168.1.9', ('Feb 20 17:29:26', '8080')),
 ('192.168.1.85', ('Feb 20 16:51:06', '443')),
 ('192.168.1.110', ('Feb 20 23:08:00', '80')),
 ('192.168.1.110', ('Feb 20 23:08:00', '3306')),
 ('192.168.1.110', ('Feb 20 23:08:00', '3306')),
 ('192.168.1.110', ('Feb 20 23:08:00', '21')),
 ('192.168.1.234', ('Feb 20 20:30:33', '3389'))]
```

4 Agrupar los Puertos por IP

[0]:
```
rdd_grouped = (
    rdd_parsed
    .map(lambda x: (x[0], (x[1][0],  [1][1])))
    .groupByKey()
    .mapValues(lambda values: sorted(set(values)))
)

rdd_grouped.take(5)
```

```
Out[25]: [('192.168.1.100',
  [('Feb 20 14:05:22', '1027'),
   ('Feb 20 14:05:29', '1007'),
   ('Feb 20 14:09:26', '1004'),
```

```
   ('Feb 20 14:09:26', '1019'),
   ('Feb 20 14:13:53', '1000'),
   ('Feb 20 14:13:53', '1020'),
   ('Feb 20 14:13:53', '1026'),
   ('Feb 20 14:13:53', '1028'),
   ('Feb 20 14:18:31', '1005'),
   ('Feb 20 14:18:31', '1010'),
   ('Feb 20 14:18:31', '1025'),
   ('Feb 20 14:19:28', '1012'),
   ('Feb 20 14:20:27', '1008'),
   ('Feb 20 14:20:27', '1014'),
   ('Feb 20 14:20:27', '1021'),
   ('Feb 20 14:21:01', '1001'),
   ('Feb 20 14:21:01', '1013'),
   ('Feb 20 14:21:01', '1017'),
   ('Feb 20 14:26:08', '1006'),
   ('Feb 20 14:27:05', '1003'),
   ('Feb 20 14:31:00', '1022'),
   ('Feb 20 14:32:45', '1024'),
   ('Feb 20 14:35:01', '1023'),
   ('Feb 20 14:37:23', '1002'),
   ('Feb 20 14:37:23', '1011'),
   ('Feb 20 14:37:23', '1016'),
   ('Feb 20 14:37:23', '1029'),
   ('Feb 20 14:38:31', '443'),
   ('Feb 20 14:41:25', '1018'),
   ('Feb 20 14:54:32', '1015'),
   ('Feb 20 14:56:14', '1009'),
   ('Feb 20 21:42:38', '3306')]),
 ('192.168.1.110',
  [('Feb 20 01:10:25', '22'),
   ('Feb 20 01:10:25', '8080'),
   ('Feb 20 23:08:00', '21'),
   ('Feb 20 23:08:00', '3306'),
   ('Feb 20 23:08:00', '80')]),
 ('192.168.1.234',
  [('Feb 20 00:44:12', '53'),
   ('Feb 20 04:28:17', '21'),
   ('Feb 20 04:28:17', '25'),
   ('Feb 20 04:28:17', '53'),
   ('Feb 20 19:09:33', '443'),
   ('Feb 20 20:30:33', '3389'),
   ('Feb 20 23:33:04', '21')]),
 ('192.168.1.41',
  [('Feb 20 01:41:46', '21'),
   ('Feb 20 01:41:46', '3389'),
```

```
   ('Feb 20 01:41:46', '8080'),
   ('Feb 20 09:29:22', '8080'), ('Feb 20 10:28:32', '22'),
   ('Feb 20 10:56:39', '80'),
   ('Feb 20 11:36:57', '53'),
   ('Feb 20 11:36:57', '8080'),
   ('Feb 20 11:55:26', '53'),
   ('Feb 20 16:38:15', '8080'),
   ('Feb 20 18:57:00', '8080')]),
 ('192.168.1.207',
  [('Feb 20 01:13:31', '22'),
   ('Feb 20 03:29:22', '443'),
   ('Feb 20 13:02:36', '21'),
   ('Feb 20 13:02:36', '25'),
   ('Feb 20 13:02:36', '443'),
   ('Feb 20 17:08:39', '22')])]
```

5 Detectar IPs con Escaneo de Puertos

```
[0]: def detectar_escaneo(data):
    ip, registros = data
    registros_ordenados =
    sorted(registros) puertos_por_hora =
    {} for timestamp, puerto in
    registros_ordenados:
        hora = timestamp[:13] if hora not in puertos_por_hora:
puertos_por_hora[hora] = set() puertos_por_hora[hora].add(puerto)
return ip, {h: len(p) for h, p in puertos_por_hora.items() if len(p)
> 1} rdd_escaneo = rdd_grouped.map(detectar_escaneo).filter(lambda
x: len(x[1]) > 0)

print("Cantidad de registros en rdd_escaneo:", rdd_escaneo.count())
print("Ejemplos de rdd_escaneo:", rdd_escaneo.take(5))
```

```
Cantidad de registros en rdd_escaneo: 153
Ejemplos de rdd_escaneo: [('192.168.1.100', {'Feb 20 14:05:': 2, 'Feb
20
14:09:': 2, 'Feb 20 14:13:': 4, 'Feb 20 14:18:': 3, 'Feb 20 14:20:':
3, 'Feb 20
14:21:': 3, 'Feb 20 14:37:': 4}), ('192.168.1.110', {'Feb 20 01:10:':
2, 'Feb 20
23:08:': 3}), ('192.168.1.234', {'Feb 20 04:28:': 3}),
('192.168.1.41', {'Feb 20 01:41:': 3, 'Feb 20 11:36:': 2}),
('192.168.1.207', {'Feb 20 13:02:': 3})]
```

6 Convertir RDD a DataFrame

[0]:
```
from pyspark.sql import SparkSession, Row
spark = SparkSession.builder.getOrCreate()

df_escaneos = rdd_escaneo.flatMap( lambda x: [(x[0], hora,
num_puertos) for hora, num_puertos in x[1].items()] ).map(lambda x:
Row(SRC_IP=x[0], Hora=x[1], PuertosUnicos=x[2])).toDF()
df_escaneos.orderBy("PuertosUnicos", ascending=False).show()
```

```
+-------------+-------------+-------------+
|      SRC_IP|      Hora|PuertosUnicos|
+-------------+-------------+-------------+
| 192.168.1.13|Feb 20 23:43:|       4|
|192.168.1.100|Feb 20 14:13:|       4|
| 192.168.1.77|Feb 20 03:13:|       4|
|192.168.1.100|Feb 20 14:37:|       4|
| 192.168.1.14|Feb 20 08:59:|       4|
|192.168.1.231|Feb 20 01:12:|       4|
|192.168.1.146|Feb 20 00:47:|       4|
| 192.168.1.39|Feb 20 10:50:|       4|
| 192.168.1.83|Feb 20 22:42:|       4|
| 192.168.1.20|Feb 20 00:15:|       4|
| 192.168.1.1|Feb 20 03:39:|       4|
|192.168.1.176|Feb 20 20:39:|       4|
|192.168.1.159|Feb 20 09:18:|       4|
|192.168.1.130|Feb 20 10:24:|       4|
|192.168.1.159|Feb 20 22:51:|       4|
| 192.168.1.93|Feb 20 01:14:|       4|
|192.168.1.229|Feb 20 08:42:|       4|
| 192.168.1.95|Feb 20 01:48:|       4|
| 192.168.1.12|Feb 20 01:24:|       4|
| 192.168.1.91|Feb 20 01:16:|       4|
+-------------+-------------+-----------
--+ only showing top 20 rows
```

7 Guardar el Resultado

[0]:
```
df_escaneos.write.mode("overwrite").csv("dbfs:/FileStore/deteccion_e
scaneo") df_escaneos.show(50)
```

```
+-------------+-------------+-------------+
|      SRC_IP|      Hora|PuertosUnicos|
+-------------+-------------+-------------+
```

```
|192.168.1.100|Feb 20 14:05:|      2|
|192.168.1.100|Feb 20 14:09:|      2|
|192.168.1.100|Feb 20 14:13:|      4|
|192.168.1.100|Feb 20 14:18:|      3|
|192.168.1.100|Feb 20 14:20:|      3|
|192.168.1.100|Feb 20 14:21:|      3|
|192.168.1.100|Feb 20 14:37:|      4|
|192.168.1.110|Feb 20 01:10:|      2|
|192.168.1.110|Feb 20 23:08:|      3|
|192.168.1.234|Feb 20 04:28:|      3|
| 192.168.1.41|Feb 20 01:41:|      3|
| 192.168.1.41|Feb 20 11:36:|      2|
|192.168.1.207|Feb 20 13:02:|      3|
|192.168.1.105|Feb 20 10:47:|      2|
| 192.168.1.53|Feb 20 14:38:|      2|
|192.168.1.191|Feb 20 08:18:|      3|
|192.168.1.191|Feb 20 17:52:|      2|
|192.168.1.191|Feb 20 18:28:|      2|
|192.168.1.149|Feb 20 04:36:|      3|
|192.168.1.216|Feb 20 02:48:|      3|
|192.168.1.231|Feb 20 01:12:|      4|
|192.168.1.170|Feb 20 05:27:|      3|
| 192.168.1.2|Feb 20 05:43:|       3|
| 192.168.1.2|Feb 20 18:07:|       2|
|192.168.1.153|Feb 20 07:45:|      2|
|192.168.1.103|Feb 20 13:30:|      3|
|192.168.1.135|Feb 20 13:59:|      3|
|192.168.1.135|Feb 20 17:11:|      3|
| 192.168.1.84|Feb 20 00:34:|      3|
|192.168.1.228|Feb 20 15:06:|      3|
|192.168.1.201|Feb 20 07:47:|      3|
|192.168.1.201|Feb 20 16:23:|      2|
| 192.168.1.20|Feb 20 00:15:|      4|
| 192.168.1.38|Feb 20 11:31:|      3|
| 192.168.1.63|Feb 20 17:22:|      3|
|192.168.1.162|Feb 20 03:10:|      2|
|192.168.1.162|Feb 20 07:34:|      3|
| 192.168.1.71|Feb 20 20:25:|      3|
| 192.168.1.71|Feb 20 23:29:|      3|
| 192.168.1.25|Feb 20 02:50:|      2|
| 192.168.1.25|Feb 20 06:48:|      2|
| 192.168.1.25|Feb 20 17:53:|      3|
| 192.168.1.19|Feb 20 02:32:|      3|
| 192.168.1.19|Feb 20 15:20:|      2|
|192.168.1.202|Feb 20 08:01:|      2|
|192.168.1.202|Feb 20 10:22:|      3|
|192.168.1.202|Feb 20 15:56:|      2|
| 192.168.1.75|Feb 20 06:46:|      3|
```

```
| 192.168.1.26|Feb 20 21:18:|      3|
|192.168.1.214|Feb 20 17:55:|      3|
+-------------+-------------+-----------
--+ only showing top 50 rows
```