

The background features several concentric circles in a light gray color, some solid and some dashed, creating a ripple effect. A large blue speech bubble is centered on the page, containing the title and author information.

Automating Packet Analysis with Python

Joe McManus
mcmanus@automox.com

The background of the slide features several sets of thin, curved lines in light gray and blue, creating a sense of motion or a stylized globe. A large blue speech bubble is positioned on the left side of the slide.

Who?

- Currently: CISO of Automox & Sr. Researcher CERT/SEI/CMU
- Past: Professor @ CU, Director of Security @ SolidFire, Head of R&D @ Webroot.
- MS @ Carnegie Mellon
- BS @ U of MD
- PhD From CU 2019(? , working on it)











What?

- Analyzing Packets with Python
- Use Scapy to pull out DNS queries, URLs, etc
- Use Pandas for time series
- Use Plotly for graphs

The background of the slide features several sets of thin, curved lines in light gray and blue, creating a sense of motion or a stylized globe. A blue speech bubble is positioned on the left side of the slide.

Why?

- During incident response you tend to do the same steps.
- Wireshark is neat, but time consuming
- Some things are better left to automation
- Python is fun
- **Packets don't lie!**

The slide features a light gray background with several sets of thin, curved lines in the corners, some solid and some dashed. A blue speech bubble is positioned on the left side.

Where?

- In a Virtual Machine
 - Fedora 28
- But you can do this anywhere
 - Requires python3-scapy
 - Prettytable
 - Pandas
 - Plotly
 - scapy_http
 - networkx

The slide features a light gray background with several sets of curved, concentric lines in the corners, resembling stylized orbits or signal waves. A blue speech bubble is positioned on the left side, containing the text 'When?'.

When?

- Anytime you have a PCAP
 - Incident Response
 - Troubleshooting
 - Application Security Analysis

The background of the slide features several sets of thin, curved lines in a light gray color, creating a sense of motion or a stylized globe. These lines are primarily located on the left and right sides of the slide.

Basics

- Scapy works at the network layer.
- Lets review the 7 OSI Model

OSI Model

Layer 7

Application
Layer

HTTP

Layer 6

Presentation
Layer

JSON/XML

Layer 5

Session Layer

SMB

Layer 4

Transport Layer

TCP/UDP

Layer 3

Network Layer

IP

Layer 2

Data Link Layer

Ethernet

Layer 1

Physical Layer

Cat 5



Scapy

- Scapy is a Python module, but also a stand alone application.
- `pip3 install scapy`

The background of the slide features several sets of concentric, curved lines in a light gray color, creating a sense of motion or a stylized globe. These lines are more prominent on the left and right sides of the slide.

Scapy

- Scapy works at liked the layers of the OSI model.
- You go from Layer 2 up.
- To find an IP address you start at layer 3.
- To print the source IP : `print(pkt[IP].src)`
- Destination IP: `print(pkt[IP].dst)`

The Scapy logo is a blue speech bubble with the word "Scapy" in white. It is positioned on the left side of the slide, with a blue rectangular bar above it. The background of the slide features decorative curved lines in the corners.

Scapy

- What layer of the OSI Model is DNS?
- To print a DNS record you would check to see if the packet has the layer.
- Then print the lookup out.
- ```
if pkt.haslayer(DNS)
 print((pkt.getlayer(DNS).qd.qname)
 .decode("utf-8"))
```



# Hands On!

- Lets get started with code.
- First we need to create a pcap file.
  - `sudo tcpdump -c2000 -w example.pcap`
  - Open Chromium and browse the web for a few minutes.





## Code Snippets

- You can download examples here:  
<http://bit.ly/pficexamples>
- I find code in slides hard to follow.

The background of the slide features several sets of thin, curved lines in light gray and blue, creating a sense of motion or a stylized globe. A large blue rectangle with a white border is positioned on the left side, containing the word 'Coding' in white text.

# Coding

- You can download a complete swiss army knife called PacketExaminer which uses all of these examples.
- `wget http://bit.ly/packetex`
- Use your favorite editor:
  - `vi`
  - `nano (sudo yum install nano -y)`
  - `less packetexaminer.py`

## Coding

- If you want to download all of the examples in advance:

<http://bit.ly/pficcode>

```
unzip pficcode
```

```
cd packetexaminer-master/training/
```

```
.. dnsExample.py ipExample.py
[joe@fedora28 training]$ ls -l
dnsExample.py
dnsPlotExample.py
httpExample.py
ipExample.py
packetTimeAgg.py
plotlyExample.py
sortedIPExample.py
```





The background of the slide features several sets of thin, curved lines in light gray, some solid and some dashed, creating a sense of motion or a stylized globe. A blue speech bubble is positioned on the left side, containing the word 'Imports' in white text.

# Imports

```
from scapy.all import *
from prettytable import PrettyTable
from collections import Counter, defaultdict
```

The background of the slide features several sets of thin, curved lines in a light gray color, creating a sense of motion or a stylized globe. These lines are more densely packed on the left side and become sparser towards the right.

Read the file

- Next we tell scapy to read the pcap file
- `packets = rdpcap('example.pcap')`

A decorative background featuring several sets of concentric, curved lines in a light gray color, resembling a stylized globe or a network of connections. These lines are positioned in the corners and along the sides of the slide.

## Process the Packets

- To just read each packet and print the source use a loop.

```
for pkt in packets:
 if IP in pkt:
 try:
 print(pkt[IP].src)
 except:
 pass
```

A decorative background featuring several sets of concentric, curved lines in light gray, some solid and some dashed, creating a sense of motion or a stylized globe.

Try it!

- Read your PCAP and print out each IP.
- Spend about 5 minutes.
- <http://bit.ly/pficex1>

```
[joes-MacBook-Pro:training joe$ python3 ipExample.py | head
192.168.128.93
192.168.128.93
52.26.208.84
192.168.128.6
192.168.128.6
52.26.208.84
192.168.128.93
192.168.128.188
192.168.128.93
192.168.128.93
-
```

A decorative background featuring several sets of concentric, curved lines in light gray, some solid and some dashed, creating a sense of motion or data flow. A blue speech bubble is positioned on the left side of the slide.

Count

- Obviously that was a bad way to view data.
- Lets add it to a list then run it through a counter

```
srcIP=[]
for pkt in packets:
 if IP in pkt:
 try:
 srcIP.append(pkt[IP].src)
 except:
 pass
```



## Analyze the Data

- Use a counter to create a count

```
cnt=Counter()
```

```
for ip in srcIP:
 cnt[ip] += 1
```

# PrettyTable

- A favorite Python module of mine is PrettyTable.
- We'll use another loop to create a sorted table of results.

```
table= PrettyTable(["IP", "Count"])
for ip, count in cnt.most_common():
 table.add_row([ip, count])
print(table)
```



# PrettyTable

```
joes-MacBook-Pro:training joe$./sortedIPExample.py
```

| IP              | Count |
|-----------------|-------|
| 192.168.128.6   | 2948  |
| 172.217.1.78    | 583   |
| 172.217.1.65    | 505   |
| 192.168.128.93  | 422   |
| 172.217.1.196   | 399   |
| 104.20.117.11   | 380   |
| 13.32.168.175   | 297   |
| 13.32.168.96    | 224   |
| 216.105.38.15   | 157   |
| 151.101.130.2   | 145   |
| 13.32.168.48    | 102   |
| 13.32.168.208   | 94    |
| 192.30.253.113  | 83    |
| 172.217.2.1     | 68    |
| 74.125.129.189  | 67    |
| 208.67.222.222  | 67    |
| 107.20.162.225  | 57    |
| 192.168.128.10  | 52    |
| 192.168.128.208 | 51    |
| 54.86.160.138   | 45    |
| 34.205.105.193  | 39    |

<http://bit.ly/pficex2>



Try it!

## Print a table of results.

```
#!/usr/bin/env python3
from scapy.all import *
from prettytable import PrettyTable
from collections import Counter

#Read the packets from file
packets = rdpcap('example.pcap')

srcIP=[]
#Read each packet and append to the srcIP
for pkt in packets:
 if IP in pkt:
 try:
 srcIP.append(pkt[IP].src)
 except:
 pass

#Create an empty list to hold the count of
cnt=Counter()

#Create a list of IPs and how many times tl
for ip in srcIP:
 cnt[ip] += 1

#Create header
table= PrettyTable(["IP", "Count"])

#Add records to table
for ip, count in cnt.most_common():
 table.add_row([ip, count])
print(table)
```

The background of the slide features several sets of thin, curved lines in light gray and blue, creating a sense of motion or data flow. A prominent blue speech bubble shape is positioned on the left side, containing the title text.

## Plotting Data

- I find graphs and charts to be much better tools for looking at network data than a simple table.
- In the past we use Matplotlib.
  - Slow, picky and unattractive.
- Plotly fixes all of this.

The background of the slide features several sets of thin, curved lines in light gray, some solid and some dashed, creating a sense of motion or a stylized globe. On the left, there is a blue rectangular box with a white border and a small white triangle pointing downwards at its bottom center. Inside this box, the word "Plotly" is written in white.

# Plotly

- To install run:

```
pip3 install plotly
```

- Then just add the import in your program.

```
from scapy.all import *
from collections import Counter
import plotly
```



## Building

- Make a copy of your previous script, and we will just add on to it.
- After printing the table add two new lists for X and Y data.

```
xData=[]
```

```
yData=[]
```

- Then loop through the IP and X and Y data

```
for ip, count in cnt.most_common():
 xData.append(ip)
 yData.append(count)
```

The background of the slide features several sets of thin, curved lines in light gray, some solid and some dashed, creating a sense of motion or data flow. A blue speech bubble is positioned on the left side, containing the title text.

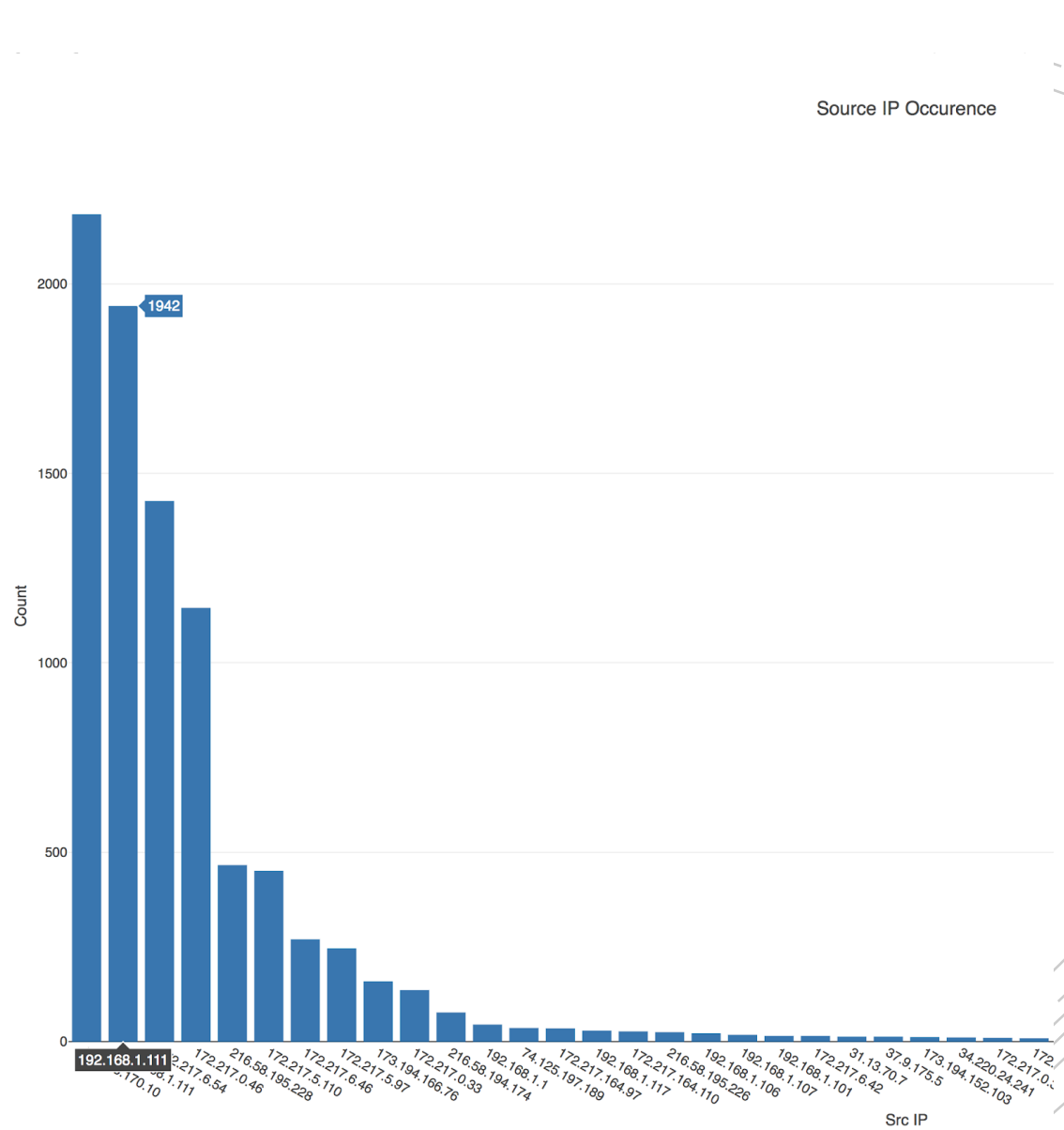
## Plot the Plotly

- Plotly is a great tool, it opens your system web browser to create interactive graphs.

```
plotly.offline.plot({
 "data": [plotly.graph_objs.Bar(x=xData, y=yData)]
})
```



Plot the Plotly





Try it!

Add the following to your script.

```
#Create an empty list to hold the count of ips
cnt=Counter()

#Create a list of IPs and how many times they appeared
for ip in srcIP:
 cnt[ip] += 1

xData=[]
yData=[]
#Sort data and create x and y
for ip, count in cnt.most_common():
 xData.append(ip)
 yData.append(count)

#Create a graph
plotly.offline.plot({
 "data":[plotly.graph_objs.Bar(x=xData, y=yData)]})
□
```

<http://bit.ly/pficex3>

## Add Labels

Refine it

```
plotly.offline.plot({
 "data": [plotly.graph_objs.Bar(x=xData, y=yData)],
 "layout": plotly.graph_objs.Layout(
 title="Source IP Occurrence",
 xaxis=dict(title="Src IP"),
 yaxis=dict(title="Count"))})
```

# HTTP URLs

- URLs can be scraped from the packets.

- To get the uri use:

```
(pkt[http.HTTPRequest].Path).decode("utf-8")
```

- To get the host use:

```
(pkt[http.HTTPRequest].Host).decode("utf-8")
```

# HTTP URLs

```
if http.HTTPRequest in pkt:
 uri=(pkt[http.HTTPRequest].Path).decode("utf-8")
 host=(pkt[http.HTTPRequest].Host).decode("utf-8")
 url=host+uri
 print(url)
```



Try it!

- <http://bit.ly/pficex4>

```
--Reading pcap file
Unique URLs
+-----+-----+
| URL | Count |
+-----+-----+
| ojsp.digicert.com/ | 4 |
| github.com/joemcmanus | 1 |
+-----+-----+
```



## Plot DNS

- With a simple change we can plot DNS lookups.
- You can also print a table of DNS lookups.



The background of the slide features several sets of thin, curved lines in light gray and blue, creating a sense of motion or data flow. A prominent blue speech bubble is positioned on the left side, containing the title text.

# Plot DNS

```
for pkt in packets:
 if IP in pkt:
 if pkt.haslayer(DNS) and pkt.getlayer(DNS).qr == 0:
 lookup=(pkt.getlayer(DNS).qd.qname).decode("utf-8")
 print(lookup)
```



Try it!

- <http://bit.ly/pficex5>

```
joes-MacBook-Pro:training joe$./dnsExample.py | head
BRW70188BEF4AC4.local.
7aba4b1e-6522-c66d-f64f-92b0ceb31544.local.
enceladus.local.
ENCELADUS._smb._tcp.local.
7aba4b1e-6522-c66d-f64f-92b0ceb31544.local.
github.com.
7aba4b1e-6522-c66d-f64f-92b0ceb31544.local.
assets-cdn.github.com.
avatars0.githubusercontent.com.
avatars1.githubusercontent.com.
```

# Plot DNS

```
from scapy.all import *
from collections import Counter, defaultdict
import plotly

packets = rdpcap("example.pcap")

lookups=[]
for pkt in packets:
 if IP in pkt:
 try:
 if pkt.haslayer(DNS) and pkt.getlayer(DNS).qr == 0:
 lookup=(pkt.getlayer(DNS).qd.qname).decode("utf-8")
 lookups.append(lookup)
 except:
 pass

cnt=Counter()
for lookup in lookups:
 cnt[lookup] += 1

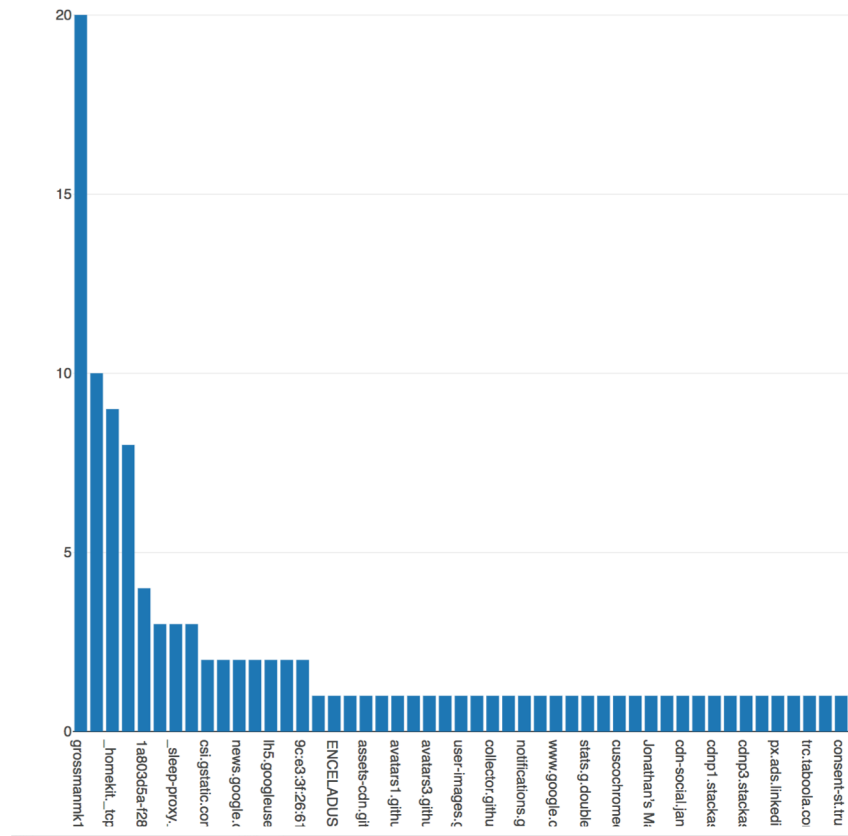
xDData=[]
yDData=[]

for lookup, count in cnt.most_common():
 xDData.append(lookup)
 yDData.append(count)

plotly.offline.plot({
 "data": [plotly.graph_objs.Bar(x=xDData, y=yDData)] })
```

Try it!

- <http://bit.ly/pficex6>





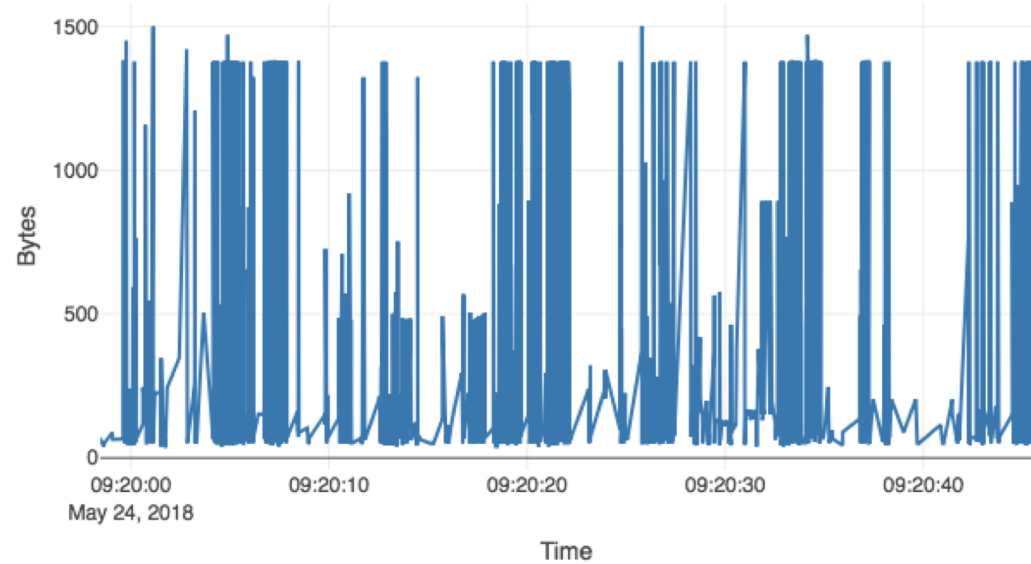


## Time Series

- You will often want to plot data over time.
- The first thought is to just look at the length of each packet.
- The problem with that is you almost always plot the maximum MTU (usually 1500)

# Time Series

Bytes over Time





The background of the slide features several sets of thin, curved lines in light gray and blue, creating a sense of motion or data flow. These lines are primarily located on the left and right sides of the slide, framing the central content.

## Time Series

- To get around this you want to bin packets over time.
- The package Pandas makes this incredibly easy for us.

# Time Series

- Start with the same imports, plus pandas.

```
from scapy.all import *
import plotly
from datetime import datetime
import pandas as pd
```

# Time Series

- PCAPs have time in epoch, we need to convert to human readable times.

```
#Read each packet and append to the lists.
```

```
for pkt in packets:
```

```
 if IP in pkt:
```

```
 try:
```

```
 pktBytes.append(pkt[IP].len)
```

```
 pktTime=datetime.fromtimestamp(pkt.time)
```

```
 pktTimes.append(pktTime.strftime("%Y-%m-%d
%H:%M:%S.%f"))
```

```
 except:
```

```
 pass
```

The background of the slide features several sets of thin, curved lines in shades of gray, creating a sense of motion or data flow. A prominent blue speech bubble is positioned on the left side, containing the text 'Time Series'.

## Time Series

- Next convert the list to a pandas time series.

```
bytes = pd.Series(pktBytes).astype(int)
```

# Time Series

- Next convert the timestamp to a date\_time for Pandas.

```
times = pd.to_datetime(pd.Series(pktTimes).astype(str),
errors='coerce')
```

# Time Series

- Create a Pandas data frame

```
df = pd.DataFrame({"Bytes": bytes, "Times":times})
```

The background of the slide features several sets of thin, curved lines in shades of gray, creating a sense of motion or a stylized globe. A blue speech bubble is positioned on the left side, containing the text 'Time Series'.

# Time Series

- Create a Pandas timestamp
- ```
df = df.set_index('Times')
```

The background of the slide features several sets of thin, curved lines in light gray and blue, creating a sense of motion or data flow. A prominent blue speech bubble shape is positioned on the left side, containing the text 'Time Series'.

Time Series

- Resample the data to 2 second bins

```
df2=df.resample('2S').sum()  
print(df2)
```

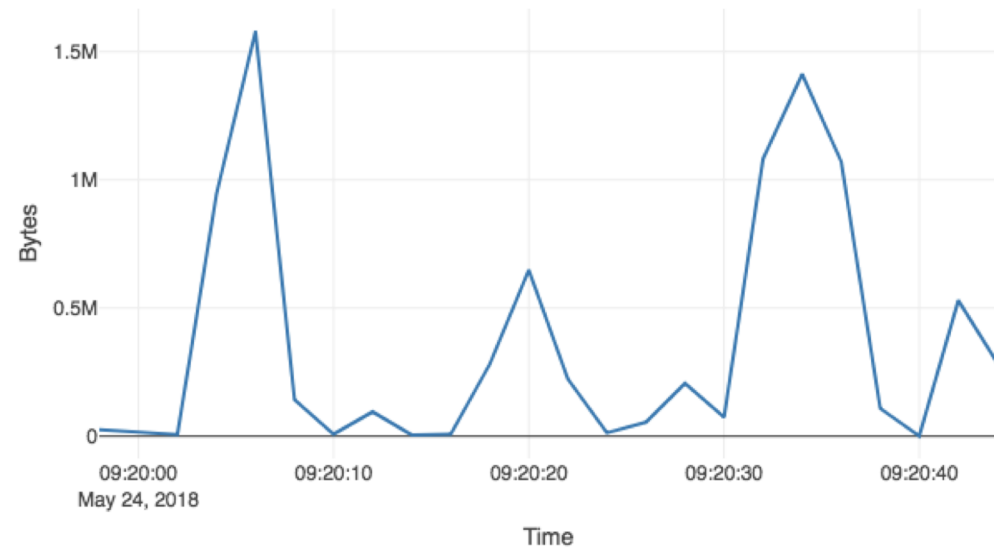

Time Series

- Print the results

```
plotly.offline.plot({  
  "data": [plotly.graph_objs.Scatter(x=df2.index,  
                                       y=df2['Bytes'])],  
  "layout": plotly.graph_objs.Layout(  
    title="Bytes over Time ",  
    xaxis=dict(title="Time"),  
    yaxis=dict(title="Bytes"))})
```

Time Series

Bytes over Time



Try it!

- <http://bit.ly/pficex7>

```
[joes-MacBook-Pro:training joe$ ./packetTimeAgg.py
Bytes
```

```
Times
2018-05-22 14:22:24      874
2018-05-22 14:22:26    11941
2018-05-22 14:22:28   59670
2018-05-22 14:22:30   63916
2018-05-22 14:22:32  120133
2018-05-22 14:22:34   16384
2018-05-22 14:22:36  337209
2018-05-22 14:22:38   37100
2018-05-22 14:22:40   50255
2018-05-22 14:22:42  784837
2018-05-22 14:22:44  577396
2018-05-22 14:22:46 1079281
2018-05-22 14:22:48   691862
2018-05-22 14:22:50   21759
2018-05-22 14:22:52  132390
2018-05-22 14:22:54   13489
2018-05-22 14:22:56   11294
2018-05-22 14:22:58    9606
2018-05-22 14:23:00   10373
2018-05-22 14:23:02   13453
2018-05-22 14:23:04    9374
```



The background of the slide features several sets of thin, curved lines in a light gray color, some solid and some dashed, creating a sense of motion or a stylized globe. A blue speech bubble is positioned on the left side of the slide.

GeolP

- It can be helpful to batch resolve locations in your data.

```
pip3 install maxminddb-geolite2
```

GeoIP

- The data is in JSON format.

```
{
  "city": {
    "geoname_id": 5375480,
    "names": {
      "de": "Mountain View",
      "en": "Mountain View",
      "fr": "Mountain View",
      "ja": "マウンテンビュー",
      "ru": "Маунтин-Вью",
      "zh-CN": "芒廷维尤"
    },
    "continent": {
      "code": "NA",
      "geoname_id": 6255149,
      "names": {
        "de": "Nordamerika",
        "en": "North America",
        "es": "Norteamérica",
        "fr": "Amérique du Nord",
        "ja": "北アメリカ",
        "pt-BR": "América do Norte",
        "ru": "Северная Америка",
        "zh-CN": "北美洲"
      },
      "country": {
        "geoname_id": 6252001,
        "iso_code": "US",
        "names": {
          "de": "USA",
          "en": "United States",
          "es": "Estados Unidos",
          "fr": "États-Unis",
          "ja": "アメリカ合衆国",
          "pt-BR": "Estados Unidos",
          "ru": "США",
          "zh-CN": "美国"
        },
        "location": {
          "accuracy_radius": 1000,
          "latitude": 37.419200000000004,
          "longitude": -122.0574,
          "metro_code": 807,
          "time_zone": "America/Los_Angeles",
          "postal": {
            "code": "94043"
          },
          "registered_country": {
            "geoname_id": 6252001,
            "iso_code": "US",
            "names": {
              "de": "USA",
              "en": "United States",
              "es": "Estados Unidos",
              "fr": "États-Unis",
              "ja": "アメリカ合衆国",
              "pt-BR": "Estados Unidos",
              "ru": "США",
              "zh-CN": "美国"
            },
            "subdivisions": [
              {
                "geoname_id": 5332921,
                "iso_code": "CA",
                "names": {
                  "de": "Kalifornien",
                  "en": "California",
                  "es": "California",
                  "fr": "Californie",
                  "ja": "カリフォルニア州",
                  "pt-BR": "Califórnia",
                  "ru": "Калифорния",
                  "zh-CN": "加利福尼亚州"
                }
              }
            ]
          }
        }
      }
    }
  }
}
```



GeoIP

- Add a new imports
- ```
from geoip import geolite2
import json
```



The background of the slide features several sets of thin, curved lines in a light gray color, creating a sense of motion or a stylized globe. These lines are primarily located on the left and right sides of the slide.

# GeolP

- Add a new imports

```
from geolite2 import geolite2
import json
```



The background of the slide features several sets of thin, curved lines in light gray, some solid and some dashed, creating a sense of motion or a stylized globe. A blue speech bubble is positioned on the left side, containing the text 'GeoIP'.

## GeoIP

- Access the data

```
reader = geolite2.reader()
```

```
match = reader.get(IP)
```

```
country=match['country']['names']['en']
```



# GeolP

- Use a lot of try/except to handle issues.

```
if match:
 try:
 country=match['country']['names']['en']
 except:
 country="unknown"
 try:
 city=match['city']['names']['en']
 except:
 city="unknown"
else:
 country="unknown"
 city="unknown"
```

- Add location to your script.
- <http://bit.ly/pficex8>

Try it!

```
joes-MacBook-Pro:training joe$./geoIPExample.py
```

| IP              | Count | Location                    |
|-----------------|-------|-----------------------------|
| 192.168.128.6   | 2948  | unknown/unknown             |
| 172.217.1.78    | 583   | United States/Mountain View |
| 172.217.1.65    | 505   | United States/Mountain View |
| 192.168.128.93  | 422   | unknown/unknown             |
| 172.217.1.196   | 399   | United States/Mountain View |
| 104.20.117.11   | 380   | United States/unknown       |
| 13.32.168.175   | 297   | United States/Seattle       |
| 13.32.168.96    | 224   | United States/Seattle       |
| 216.105.38.15   | 157   | United States/San Diego     |
| 151.101.130.2   | 145   | United States/San Francisco |
| 13.32.168.48    | 102   | United States/Seattle       |
| 13.32.168.208   | 94    | United States/Seattle       |
| 192.30.253.113  | 83    | United States/San Francisco |
| 172.217.2.1     | 68    | United States/Mountain View |
| 74.125.129.189  | 67    | United States/Mountain View |
| 208.67.222.222  | 67    | United States/San Francisco |
| 107.20.162.225  | 57    | United States/Ashburn       |
| 192.168.128.10  | 52    | unknown/unknown             |
| 192.168.128.208 | 51    | unknown/unknown             |
| 54.86.160.138   | 45    | United States/Ashburn       |

The background of the slide features several sets of concentric, curved lines in a light gray color, creating a sense of motion or a stylized wave pattern. These lines are more prominent on the left side and fade towards the right.

## Tips

- I hate hardcoding filenames.
- You create a parser object and add options.
- `parser=argparse.ArgumentParser(description='Example Command Line Parser')`
- `parser.add_argument('filename', action="store")`
- For troubleshooting, use:
- `print(parser.parse_args())`

## Tips

```
parser = argparse.ArgumentParser(description='PCAP File Examiner')
parser.add_argument('file', help="Source PCAP File, i.e. example.pcap", type=str)
parser.add_argument('--flows', help="Display flow summary", action="store_true")
parser.add_argument('--dst', help="Display count of destination IPs", action="store_true")
parser.add_argument('--src', help="Display count of source IPs", action="store_true")
parser.add_argument('--dport', help="Display count of destination ports", action="store_true")
parser.add_argument('--sport', help="Display count of source ports", action="store_true")
parser.add_argument('--ports', help="Display count of all ports", action="store_true")
parser.add_argument('--portbytes', help="Display ports by bytes", action="store_true")
parser.add_argument('--bytes', help="Display source and destination byte counts", action="store_true")
parser.add_argument('--dns', help="Display all DNS Lookups in PCAP", action="store_true")
parser.add_argument('--url', help="Display all ULRs in PCAP", action="store_true")
parser.add_argument('--netmap', help="Display a network Map", action="store_true")
parser.add_argument('--xfiles', help="Extract files from PCAP", action="store_true")
parser.add_argument('--resolve', help="Resolve IPs", action="store_true")
parser.add_argument('--details', help="Display additional details where available", action="store_true")
parser.add_argument('--graphs', help="Display graphs where available", action="store_true")
parser.add_argument('--timeseries', help="Display data over time", action="store_true")
parser.add_argument('--all', help="Display all", action="store_true")
parser.add_argument('--limit', help="Limit results to X", type=int)
parser.add_argument('--skipopts', help="Don't display the options at runtime", action="store_true")
parser.add_argument('--outdir', help="Output directory for files, default = pwd ", action="store")
args=parser.parse_args()
```

- I've bundled all of this and more in to a open source tool for DFIR called PacketExaminer.
- <https://github.com/joemcmanus/packetexaminer>

**PacketExaminer Network Map**

The network map shows a central node connected to numerous peripheral nodes, each labeled with an IP address. The connections are represented by lines of varying thickness, indicating the volume of traffic. The IP addresses include:

- 151.101.65.67
- 151.103.1.67
- 54.225.203.48
- 54.230.5.100
- 172.217.1.202
- 172.217.1.187
- 172.217.1.74
- 23.1.136.197
- 172.217.1.225
- 17.186.136.186
- 181.45
- 52.45.36.2
- 172.217.1.1238
- 151.103.0.64
- 23.99.206.151
- 50.31.18.92
- 192.168.1.26
- 151.101.64.175
- 23.217.164.153
- 199.58.148.85
- 54.227.264.207
- 52.5.245.83
- 172.217.1.1276
- 104.244.42.194
- 192.168.1.15
- 178.217.11.229
- 66.235.153.37
- 193.50.130.10
- 208.67.222.222
- 54.192.4.158
- 54.230.5.240
- 54.230.5.84
- 54.235.190.145
- 199.117.103.133
- 199.66.57.6

**Traffic by Port and Bytes**

The pie chart shows the distribution of traffic across different ports. The legend indicates the following ports and their corresponding colors:

- 443 (Blue)
- 80 (Orange)
- 8080 (Green)
- 8081 (Yellow)
- 8082 (Red)
- 8083 (Purple)
- 8084 (Brown)
- 8085 (Pink)
- 8086 (Light Blue)
- 8087 (Light Green)
- 8088 (Light Yellow)
- 8089 (Light Purple)
- 8090 (Light Brown)
- 8091 (Light Pink)
- 8092 (Light Blue)
- 8093 (Light Green)
- 8094 (Light Yellow)
- 8095 (Light Purple)
- 8096 (Light Brown)
- 8097 (Light Pink)
- 8098 (Light Blue)
- 8099 (Light Green)
- 8100 (Light Yellow)
- 8101 (Light Purple)
- 8102 (Light Brown)
- 8103 (Light Pink)
- 8104 (Light Blue)
- 8105 (Light Green)
- 8106 (Light Yellow)
- 8107 (Light Purple)
- 8108 (Light Brown)
- 8109 (Light Pink)
- 8110 (Light Blue)
- 8111 (Light Green)
- 8112 (Light Yellow)
- 8113 (Light Purple)
- 8114 (Light Brown)
- 8115 (Light Pink)
- 8116 (Light Blue)
- 8117 (Light Green)
- 8118 (Light Yellow)
- 8119 (Light Purple)
- 8120 (Light Brown)
- 8121 (Light Pink)
- 8122 (Light Blue)
- 8123 (Light Green)
- 8124 (Light Yellow)
- 8125 (Light Purple)
- 8126 (Light Brown)
- 8127 (Light Pink)
- 8128 (Light Blue)
- 8129 (Light Green)
- 8130 (Light Yellow)
- 8131 (Light Purple)
- 8132 (Light Brown)
- 8133 (Light Pink)
- 8134 (Light Blue)
- 8135 (Light Green)
- 8136 (Light Yellow)
- 8137 (Light Purple)
- 8138 (Light Brown)
- 8139 (Light Pink)
- 8140 (Light Blue)
- 8141 (Light Green)
- 8142 (Light Yellow)
- 8143 (Light Purple)
- 8144 (Light Brown)
- 8145 (Light Pink)
- 8146 (Light Blue)
- 8147 (Light Green)
- 8148 (Light Yellow)
- 8149 (Light Purple)
- 8150 (Light Brown)
- 8151 (Light Pink)
- 8152 (Light Blue)
- 8153 (Light Green)
- 8154 (Light Yellow)
- 8155 (Light Purple)
- 8156 (Light Brown)
- 8157 (Light Pink)
- 8158 (Light Blue)
- 8159 (Light Green)
- 8160 (Light Yellow)
- 8161 (Light Purple)
- 8162 (Light Brown)
- 8163 (Light Pink)
- 8164 (Light Blue)
- 8165 (Light Green)
- 8166 (Light Yellow)
- 8167 (Light Purple)
- 8168 (Light Brown)
- 8169 (Light Pink)
- 8170 (Light Blue)
- 8171 (Light Green)
- 8172 (Light Yellow)
- 8173 (Light Purple)
- 8174 (Light Brown)
- 8175 (Light Pink)
- 8176 (Light Blue)
- 8177 (Light Green)
- 8178 (Light Yellow)
- 8179 (Light Purple)
- 8180 (Light Brown)
- 8181 (Light Pink)
- 8182 (Light Blue)
- 8183 (Light Green)
- 8184 (Light Yellow)
- 8185 (Light Purple)
- 8186 (Light Brown)
- 8187 (Light Pink)
- 8188 (Light Blue)
- 8189 (Light Green)
- 8190 (Light Yellow)
- 8191 (Light Purple)
- 8192 (Light Brown)
- 8193 (Light Pink)
- 8194 (Light Blue)
- 8195 (Light Green)
- 8196 (Light Yellow)
- 8197 (Light Purple)
- 8198 (Light Brown)
- 8199 (Light Pink)
- 8200 (Light Blue)
- 8201 (Light Green)
- 8202 (Light Yellow)
- 8203 (Light Purple)
- 8204 (Light Brown)
- 8205 (Light Pink)
- 8206 (Light Blue)
- 8207 (Light Green)
- 8208 (Light Yellow)
- 8209 (Light Purple)
- 8210 (Light Brown)
- 8211 (Light Pink)
- 8212 (Light Blue)
- 8213 (Light Green)
- 8214 (Light Yellow)
- 8215 (Light Purple)
- 8216 (Light Brown)
- 8217 (Light Pink)
- 8218 (Light Blue)
- 8219 (Light Green)
- 8220 (Light Yellow)
- 8221 (Light Purple)
- 8222 (Light Brown)
- 8223 (Light Pink)
- 8224 (Light Blue)
- 8225 (Light Green)
- 8226 (Light Yellow)
- 8227 (Light Purple)
- 8228 (Light Brown)
- 8229 (Light Pink)
- 8230 (Light Blue)
- 8231 (Light Green)
- 8232 (Light Yellow)
- 8233 (Light Purple)
- 8234 (Light Brown)
- 8235 (Light Pink)
- 8236 (Light Blue)
- 8237 (Light Green)
- 8238 (Light Yellow)
- 8239 (Light Purple)
- 8240 (Light Brown)
- 8241 (Light Pink)
- 8242 (Light Blue)
- 8243 (Light Green)
- 8244 (Light Yellow)
- 8245 (Light Purple)
- 8246 (Light Brown)
- 8247 (Light Pink)
- 8248 (Light Blue)
- 8249 (Light Green)
- 8250 (Light Yellow)
- 8251 (Light Purple)
- 8252 (Light Brown)
- 8253 (Light Pink)
- 8254 (Light Blue)
- 8255 (Light Green)
- 8256 (Light Yellow)
- 8257 (Light Purple)
- 8258 (Light Brown)
- 8259 (Light Pink)
- 8260 (Light Blue)
- 8261 (Light Green)
- 8262 (Light Yellow)
- 8263 (Light Purple)
- 8264 (Light Brown)
- 8265 (Light Pink)
- 8266 (Light Blue)
- 8267 (Light Green)
- 8268 (Light Yellow)
- 8269 (Light Purple)
- 8270 (Light Brown)
- 8271 (Light Pink)
- 8272 (Light Blue)
- 8273 (Light Green)
- 8274 (Light Yellow)
- 8275 (Light Purple)
- 8276 (Light Brown)
- 8277 (Light Pink)
- 8278 (Light Blue)
- 8279 (Light Green)
- 8280 (Light Yellow)
- 8281 (Light Purple)
- 8282 (Light Brown)
- 8283 (Light Pink)
- 8284 (Light Blue)
- 8285 (Light Green)
- 8286 (Light Yellow)
- 8287 (Light Purple)
- 8288 (Light Brown)
- 8289 (Light Pink)
- 8290 (Light Blue)
- 8291 (Light Green)
- 8292 (Light Yellow)
- 8293 (Light Purple)
- 8294 (Light Brown)
- 8295 (Light Pink)
- 8296 (Light Blue)
- 8297 (Light Green)
- 8298 (Light Yellow)
- 8299 (Light Purple)
- 8300 (Light Brown)
- 8301 (Light Pink)
- 8302 (Light Blue)
- 8303 (Light Green)
- 8304 (Light Yellow)
- 8305 (Light Purple)
- 8306 (Light Brown)
- 8307 (Light Pink)
- 8308 (Light Blue)
- 8309 (Light Green)
- 8310 (Light Yellow)
- 8311 (Light Purple)
- 8312 (Light Brown)
- 8313 (Light Pink)
- 8314 (Light Blue)
- 8315 (Light Green)
- 8316 (Light Yellow)
- 8317 (Light Purple)
- 8318 (Light Brown)
- 8319 (Light Pink)
- 8320 (Light Blue)
- 8321 (Light Green)
- 8322 (Light Yellow)
- 8323 (Light Purple)
- 8324 (Light Brown)
- 8325 (Light Pink)
- 8326 (Light Blue)
- 8327 (Light Green)
- 8328 (Light Yellow)
- 8329 (Light Purple)
- 8330

Multi Platform  
Cloud Patching  
& Management



**AUTOMOX**



Questions?

- Any questions?
- [mcmanus@automox.com](mailto:mcmanus@automox.com)
- [www.linkedin.com/in/networkforensics/](https://www.linkedin.com/in/networkforensics/)
- [github.com/joemcmanus](https://github.com/joemcmanus)



