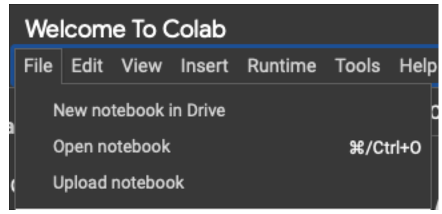
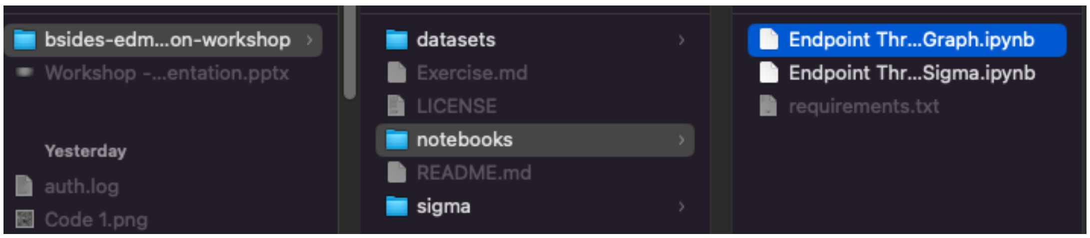
Threat Hunting with Jupyter Notebooks Workshop – Demo 2

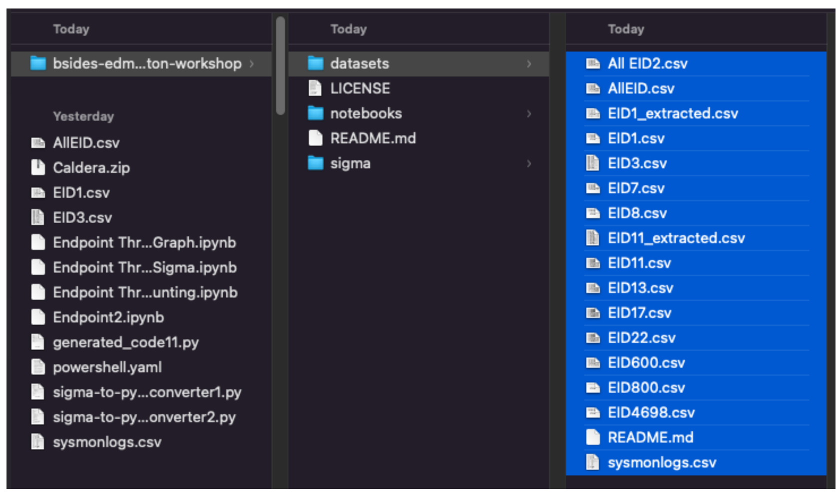
By: Kai Iyer and Meaghan Neill

1. **Go to https://colab.research.google.com**
2. **Select Upload notebook, go to the folder your "bsides-edmonton-workshop" downloaded from Github, and select the "Endpoint Threat Hunting - ML+Graph.ipynb" notebook to open it.**





3. **Upload all the csv files for this workshop.**



4. **Run Cell 1 to import the needed libraries.**

```
import json
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, LabelEncoder
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.callbacks import EarlyStopping

import plotly.express as px
import community.community_louvain
import networkx as nx
from matplotlib.pyplot import figure
import matplotlib.pyplot as plt
```

Threat Hunting with Jupyter Notebooks Workshop – Demo 2

By: Kai Iyer and Meaghan Neill

5. **Cell 2, "Anomaly Detection using Autoencoders" takes the csv input file and uses Autoencoders to find anomalies in the file, looking for the "unknown".**

```
# Load the data
data = pd.read_csv('EID1_extracted.csv')

# Select relevant features
features = ['CommandLine', 'Image', 'ParentImage', 'CurrentDirectory']
data = data[features]

data.dropna(inplace=True)

# Encode categorical variables
label_encoders = {}
for col in features:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
    label_encoders[col] = le

# Normalize the data
scaler = StandardScaler()
data_normalized = scaler.fit_transform(data)
X_train, X_test = train_test_split(data_normalized, test_size=0.2, random_state=42)

# Autoencoder Model
input_dim = X_train.shape[1]
encoding_dim = 14

input_layer = Input(shape=(input_dim,))
encoded = Dense(encoding_dim, activation='relu')(input_layer)
encoded = Dense(int(encoding_dim / 2), activation='relu')(encoded)
decoded = Dense(int(encoding_dim / 2), activation='relu')(encoded)
decoded = Dense(input_dim, activation='sigmoid')(decoded)

autoencoder = Model(inputs=input_layer, outputs=decoded)
autoencoder.compile(optimizer='adam', loss='mean_squared_error')

# Early stopping
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

# Train the model
history = autoencoder.fit(X_train, X_train,
                          epochs=100,
                          batch_size=256,
                          shuffle=True,
                          validation_split=0.2,
                          callbacks=[early_stopping])

# Evaluate the model
predictions = autoencoder.predict(X_test)
mse = np.mean(np.power(X_test - predictions, 2), axis=1)

# Define your threshold
#threshold = np.percentile(reconstruction_errors, 95)
threshold = np.mean(mse) + 2 * np.std(mse)

# Predict anomalies
anomaly_indices = np.where(mse > threshold)[0]
anomalies = data.iloc[anomaly_indices].copy()

for col in features:
    anomalies[col] = label_encoders[col].inverse_transform(anomalies[col].astype(int))

anomalies.reset_index(drop=True, inplace=True)

anomalies[['CommandLine', 'Image', 'ParentImage', 'CurrentDirectory']]
```

6. **The output of Cell 2 shows all the results that the program considers Anomalies.**

	CommandLine	Image	ParentImage	CurrentDirectory
0	"C:\Program Files\Npcap\NPFInstall.exe" -n -d	C:\Program Files\Npcap\NPFInstall.exe	C:\Program Files\Npcap\Uninstall.exe	C:\Program Files\Npcap\
1	"powershell.exe" & {Install-Module -Name AADIn...	C:\Windows\System32\WindowsPowerShell\v1.0\pow...	C:\Windows\System32\WindowsPowerShell\v1.0\pow...	C:\Users\JOHNNY~1.DOU\AppData\Local\Temp\
2	"C:\Windows\system32\cmd.exe" /c "where.exe Sy...	C:\Windows\System32\cmd.exe	C:\Windows\System32\WindowsPowerShell\v1.0\pow...	C:\Users\johnny.douche\AppData\Local\Temp\
3	where.exe Sysmon.exe	C:\Windows\System32\where.exe	C:\Windows\System32\cmd.exe	C:\Users\johnny.douche\AppData\Local\Temp\
4	"C:\Windows\System32\WindowsPowerShell\v1.0\po...	C:\Windows\System32\WindowsPowerShell\v1.0\pow...	C:\Windows\System32\WindowsPowerShell\v1.0\pow...	C:\Users\JOHNNY~1.DOU\AppData\Local\Temp\
...	...	...	...	...
80	"C:\Windows\Microsoft.NET\Framework64\v4.0.303...	C:\Windows\Microsoft.NET\Framework64\v4.0.3031...	C:\Windows\System32\WindowsPowerShell\v1.0\pow...	C:\Users\johnny.douche\AppData\Local\Temp\
81	"C:\Windows\System32\WindowsPowerShell\v1.0\po...	C:\Windows\System32\WindowsPowerShell\v1.0\pow...	C:\Windows\System32\WindowsPowerShell\v1.0\pow...	C:\Users\JOHNNY~1.DOU\AppData\Local\Temp\
82	schtasks /query /tn win32times	C:\Windows\System32\schtasks.exe	C:\Windows\System32\cmd.exe	C:\Users\JOHNNY~1.DOU\AppData\Local\Temp\
83	"powershell.exe" & {nmap 127.0.0.1}	C:\Windows\System32\WindowsPowerShell\v1.0\pow...	C:\Windows\System32\WindowsPowerShell\v1.0\pow...	C:\Users\JOHNNY~1.DOU\AppData\Local\Temp\
84	"powershell.exe" & {$S3cur3Th1sSh1t_repo='http...	C:\Windows\System32\WindowsPowerShell\v1.0\pow...	C:\Windows\System32\WindowsPowerShell\v1.0\pow...	C:\Users\JOHNNY~1.DOU\AppData\Local\Temp\

7. **Go to the Visualization Cells. Run Visualization Cell 1 to read the csv input file. In this case it is reading the sysmonlogs.csv. Make sure to change this with whatever you want to visualize.**

```
df = pd.read_csv('sysmonlogs.csv')
df.columns, df.shape
```

```
(Index(['IP', 'computer_name', 'Image'], dtype='object'), (14527, 3))
```

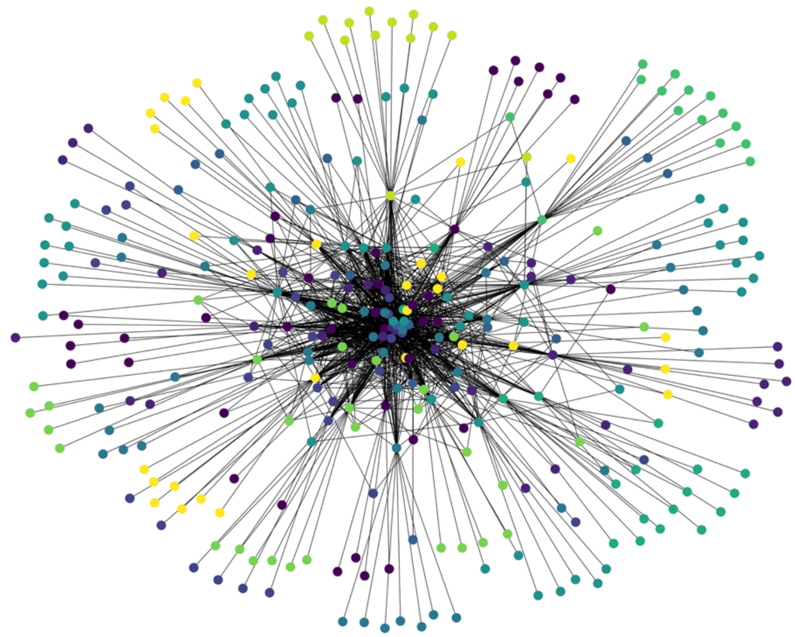
8. **Run Visualization Cell 2, which uses "computer_name" and "Image". This will create a Edgelist Graph for to view data.**

```
G = nx.Graph()
G = nx.from_pandas_edgelist(df, 'computer_name','Image')

part = community.community_louvain.best_partition(G)
values = [part.get(node) for node in G.nodes()]

figure(figsize=(25, 20))
nx.draw_spring(G, node_color=values,with_labels=False)
```

Threat Hunting with Jupyter Notebooks Workshop – Demo 2

By: Kai Iyer and Meaghan Neill

9. **Based on the output, it doesn't look like a graph using these choices "computer_name" and "Image" would be any help in spotting anomalies or outliers.**



10. **To make the visualizations more useful, you will want to play around with the fields and see what you can find. In our example, we switched the fields being used to "Image" and "IP".**

```
G = nx.Graph()
G = nx.from_pandas_edgelist(df, 'Image','IP')

part = community.community_louvain.best_partition(G)
values = [part.get(node) for node in G.nodes()]

figure(figsize=(25, 20))
nx.draw_spring(G, node_color=values,with_labels=False)
```

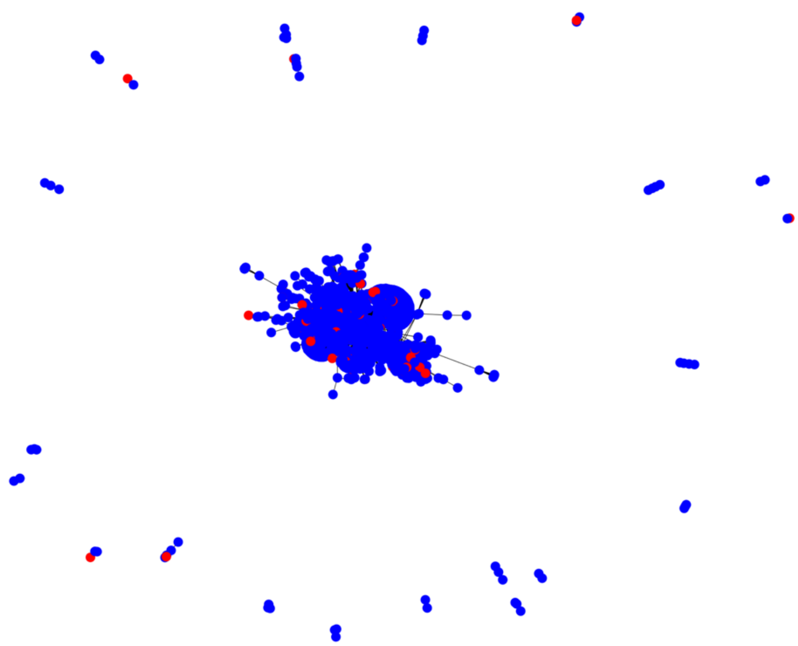
11. **After running Visualization Cell 3, you can see from the output that this is much more effective in seeing outliers. But seeing outliers doesn't always mean that these are anomalies.**

Threat Hunting with Jupyter Notebooks Workshop – Demo 2

By: Kai Iyer and Meaghan Neill

12. To better see anomalies, you can use Isolation Forests Cell. Isolation forests operate by isolating anomalies within a dataset through a process of recursive partitioning.

```python
from sklearn.preprocessing import LabelEncoder
from sklearn.ensemble import IsolationForest

le = LabelEncoder()
df['IP_encoded'] = le.fit_transform(df['IP'])
df['Image_encoded'] = le.fit_transform(df['Image'])

node_features = df[['IP_encoded', 'Image_encoded']]

# Train Isolation Forest
clf = IsolationForest(contamination=0.05)
pred = clf.fit_predict(node_features)

# Add anomaly information to the partition dictionary
for i, node in enumerate(G.nodes()):
    part[node] = pred[i] + 1

anomaly_color = ['red' if pred[i] == -1 else 'blue' for i in range(len(G.nodes()))]

# Visualize the graph with anomaly color
figure(figsize=(25, 20))
nx.draw_spring(G, node_color=anomaly_color, with_labels=False)
```



13. While looking at the data, the red was anomalies and blue was regular activity. What this can show you is that there are some anomalies within the non-outliers. This is an important concept when looking for “unknown” because it can mimic regular activity.

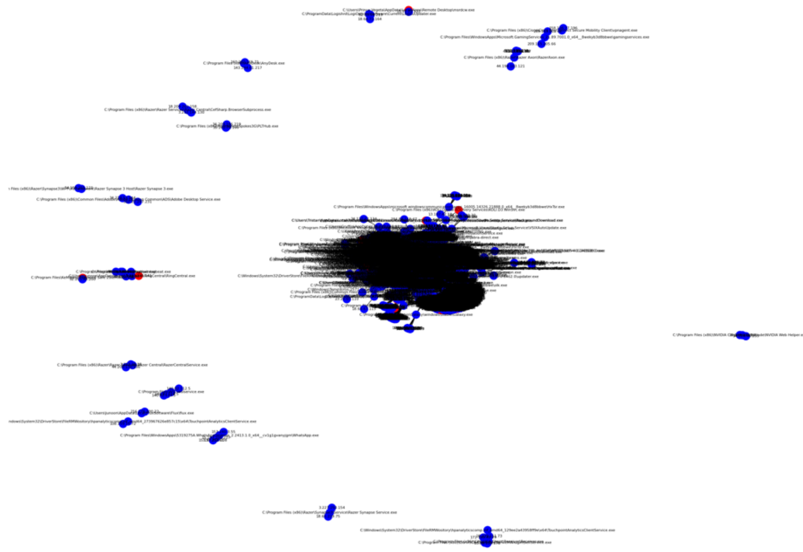
14. You can also add labels to all those datapoints and print the Isolation Forest.

```python
import matplotlib.pyplot as plt

anomaly_threshold = 5

anomalies = [node for node, score in zip(G.nodes(), clf.decision_function(node_features)) if score < anomaly_threshold]

node_labels = {node: node if node in anomalies else '' for node in G.nodes()}

# Visualize the graph with labeled anomalies
figure(figsize=(25, 20))
nx.draw_spring(G, node_color=anomaly_color, labels=node_labels, font_size=8)

outliers_list = [(df.loc[df['IP'] == node, 'Image'].values[0] if len(df.loc[df['IP'] == node, 'Image']) > 0 else None, node) for node in anomalies]

outliers_list = [(image, ip) for image, ip in outliers_list if image is not None]

print("List of Outliers (Image, IP):")
print(outliers_list, "\n")

plt.show()
```

Threat Hunting with Jupyter Notebooks Workshop – Demo 2

By: Kai Iyer and Meaghan Neill

15. As you can see from the graph, the datapoints on the anomalies displayed in this way is not very readable.



16. After viewing the anomalies in a visual fashion and confirming there is some, you can then view them in a list. This makes them readable.

```
outliers_list
```

```
('C:\\Windows\\System32\\svchost.exe', '172.16.26.226'),
('C:\\Windows\\System32\\svchost.exe', '1.1.1.1'),
('C:\\Program Files\\Galaxy\\bin\\osqueryd\\windows\\stable\\osqueryd.exe',
 '15.222.106.248'),
('System', '172.16.26.20'),
('C:\\Program Files\\Galaxy\\bin\\osqueryd\\windows\\stable\\osqueryd.exe',
 '3.97.206.33'),
('C:\\Program Files\\Windows Defender Advanced Threat Protection\\MsSense.exe',
 '13.89.179.11'),
('C:\\Program Files\\WinlogBeat\\winlogbeat.exe', '35.182.199.174'),
('C:\\Program Files\\Galaxy\\bin\\osqueryd\\windows\\stable\\osqueryd.exe',
 '15.223.124.170'),
('System', '172.16.26.255'),
('System', '172.16.26.234'),
('C:\\Windows\\System32\\svchost.exe', '13.89.178.27'),
('C:\\Program Files\\Mozilla Firefox\\firefox.exe', '34.149.100.209'),
('C:\\Program Files\\WindowsApps\\91750D7E.Slack_4.37.101.0_x64__8she8kybcnzg4\\app\\Slack.exe',
 '99.77.145.43'),
('C:\\Program Files\\WindowsApps\\91750D7E.Slack_4.37.101.0_x64__8she8kybcnzg4\\app\\Slack.exe',
 '44.216.98.239'),
('C:\\Program Files\\WinlogBeat\\winlogbeat.exe', '99.79.100.155'),
('C:\\Windows\\System32\\svchost.exe', '64.71.255.198'),
('C:\\Windows\\System32\\svchost.exe', '192.168.50.1'),
('C:\\Windows\\System32\\svchost.exe', '192.168.1.1'),
('C:\\Windows\\System32\\svchost.exe', '64.71.255.204'),
('C:\\Windows\\System32\\svchost.exe', '207.164.234.129'),
('C:\\Windows\\System32\\svchost.exe', '192.168.2.1'),
('C:\\Windows\\System32\\svchost.exe', '209.148.170.203'),
('C:\\Windows\\System32\\svchost.exe', '72.136.195.19'),
('C:\\Windows\\System32\\svchost.exe', '192.168.32.14'),
('C:\\Windows\\System32\\svchost.exe', '72.136.195.42'),
('System', '192.168.32.255'),
('System', '192.168.32.201'),
('System', '192.168.56.255'),
('System', '192.168.56.1'),
('C:\\Program Files\\limacharlie\\bluepill.exe', '15.223.109.237'),
```