Measuring User Behavior

findingbad.blogspot.com/2021/10/measuring-user-behavior.html

I'm always looking for different ways to look at data. Things that will give me insights into how users act as they go about their daily activities. Having done this for a while now I can say that people typically like routines as they go about their work, but how they go about their work is largely dictated by their job role. Examples may be the developer that often creates archive files or the sales person that authenticates from multiple locations per day.

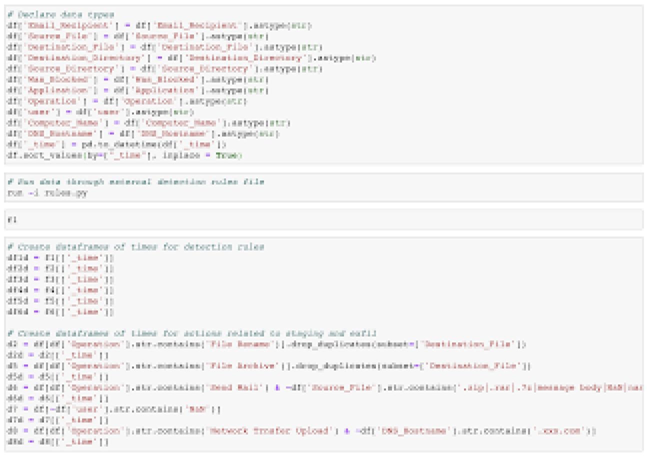
Operating under the assumption that people will generally look the same from day to day it would make sense to understand why someone may deviate from their normal routine, especially from a security context. One way we can do this is to measure their own, unique behavior and apply some anomaly detection to highlight days of interest.

For the example I'm going to show, I wanted to focus on insider from a data theft standpoint, but this method can be applied to many different scenarios as long as you can define the points of interest to observe. The hypothesis that I'm using for insider data theft is:

1. Users who knowingly steal data will often use deceptive actions.
2. They will perform actions that are new or rare for them.
3. They will use uncommon exfiltration paths.
4. Rare actions across multiple phases can be identified.

The phases (categories) I'm using here are Deception, data staging and exfiltration.

Below I'm pasting screenshots of the relevant portion of the Jupyter notebook I'm using. The cool thing about using Jupyter is it's independent of the log source. I can use this as long as the data can be exported or accessed via api. Code blocks below are commented to describe the function.



```
# Declare data types
df['Email_Recipient'] = df['Email_Recipient'].astype(str)
df['Source_File'] = df['Source_File'].astype(str)
df['Destination_File'] = df['Destination_File'].astype(str)
df['Destination_Directory'] = df['Destination_Directory'].astype(str)
df['Source_Directory'] = df['Source_Directory'].astype(str)
df['Was_Blocked'] = df['Was_Blocked'].astype(str)
df['Application'] = df['Application'].astype(str)
df['Operation'] = df['Operation'].astype(str)
df['user'] = df['user'].astype(str)
df['Computer_Name'] = df['Computer_Name'].astype(str)
df['DNS_Hostname'] = df['DNS_Hostname'].astype(str)
df['_time'] = pd.to_datetime(df['_time'])
df.sort_values(by=['_time'], inplace = True)
```

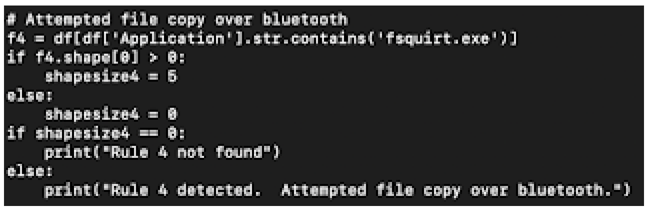
```
# Run data through external detection rules file
run -i rules.py
```

```
f1
```

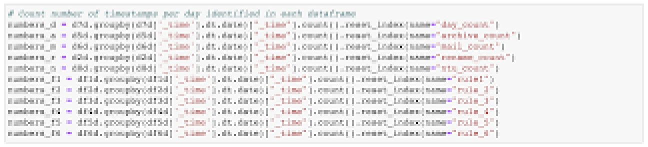
```
# Create dataframes of times for detection rules
df1d = f1[['_time']]
df2d = f2[['_time']]
df3d = f3[['_time']]
df4d = f4[['_time']]
df5d = f5[['_time']]
df6d = f6[['_time']]

# Create dataframes of times for actions related to staging and exfil
d2 = df[df['Operation'].str.contains('File Rename')].drop_duplicates(subset=['Destination_File'])
d2d = d2[['_time']]
d5 = df[df['Operation'].str.contains('File Archive')].drop_duplicates(subset=['Destination_File'])
d5d = d5[['_time']]
d6 = df[df['Operation'].str.contains('Send Mail') & ~df['Source_File'].str.contains('.zip|.rar|.7z|message body|NaN|na
d6d = d6[['_time']]
d7 = df[~df['user'].str.contains('NaN')]
d7d = d7[['_time']]
d8 = df[df['Operation'].str.contains('Network Trasfer Upload') & ~df['DNS_Hostname'].str.contains('.xxx.com')]
d8d = d8[['_time']]
```

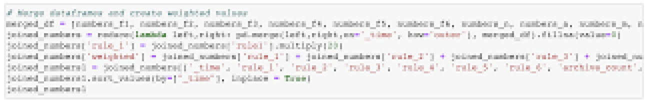
Below is an example rule from an external rule file.



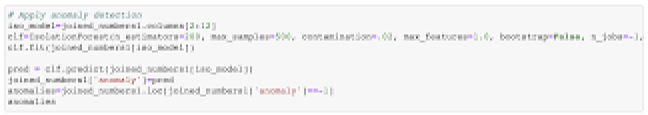
```
# Attempted file copy over bluetooth
f4 = df[df['Application'].str.contains('fsquirt.exe')]
if f4.shape[0] > 0:
    shapesize4 = 5
else:
    shapesize4 = 0
if shapesize4 == 0:
    print("Rule 4 not found")
else:
    print("Rule 4 detected.  Attempted file copy over bluetooth.")
```



```
# Count number of timestamps per day identified in each dataframe
numbers_d = d7d.groupby(d7d['_time'].dt.date)["_time"].count().reset_index(name="day_count")
numbers_a = d5d.groupby(d5d['_time'].dt.date)["_time"].count().reset_index(name="archive_count")
numbers_m = d6d.groupby(d6d['_time'].dt.date)["_time"].count().reset_index(name="mail_count")
numbers_r = d2d.groupby(d2d['_time'].dt.date)["_time"].count().reset_index(name="rename_count")
numbers_n = d8d.groupby(d8d['_time'].dt.date)["_time"].count().reset_index(name="ntu_count")
numbers_f1 = df1d.groupby(df1d['_time'].dt.date)["_time"].count().reset_index(name="rule1")
numbers_f2 = df2d.groupby(df2d['_time'].dt.date)["_time"].count().reset_index(name="rule_2")
numbers_f3 = df3d.groupby(df3d['_time'].dt.date)["_time"].count().reset_index(name="rule_3")
numbers_f4 = df4d.groupby(df4d['_time'].dt.date)["_time"].count().reset_index(name="rule_4")
numbers_f5 = df5d.groupby(df5d['_time'].dt.date)["_time"].count().reset_index(name="rule_5")
numbers_f6 = df6d.groupby(df6d['_time'].dt.date)["_time"].count().reset_index(name="rule_6")
```



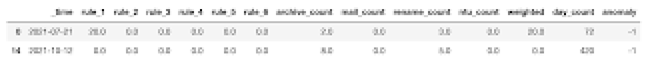
```
# Merge dataframes and create weighted values
merged_df = [numbers_f1, numbers_f2, numbers_f3, numbers_f4, numbers_f5, numbers_f6, numbers_r, numbers_a, numbers_m, n
joined_numbers = reduce(lambda left,right: pd.merge(left,right,on='_time', how='outer'), merged_df).fillna(value=0)
joined_numbers['rule_1'] = joined_numbers['rule1'].multiply(20)
joined_numbers['weighted'] = joined_numbers['rule_1'] + joined_numbers['rule_2'] + joined_numbers['rule_3'] + joined_nu
joined_numbers1 = joined_numbers[['_time', 'rule_1', 'rule_2', 'rule_3', 'rule_4', 'rule_5', 'rule_6', 'archive_count',
joined_numbers1.sort_values(by=["_time"], inplace = True)
joined_numbers1
```



```
# Apply anomaly detection
iso_model=joined_numbers1.columns[2:12]
clf=IsolationForest(n_estimators=200, max_samples=500, contamination=.02, max_features=1.0, bootstrap=False, n_jobs=-1,
clf.fit(joined_numbers1[iso_model])

pred = clf.predict(joined_numbers1[iso_model])
joined_numbers1['anomaly']=pred
anomalies=joined_numbers1.loc[joined_numbers1['anomaly']==-1]
anomalies
```

Below would be the output from the anomaly detection.



	_time	rule_1	rule_2	rule_3	rule_4	rule_5	rule_6	archive_count	mail_count	rename_count	ntu_count	weighted	day_count	anomaly
6	2021-07-21	20.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	0.0	20.0	72	-1
14	2021-10-12	0.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	5.0	0.0	0.0	420	-1

The fields that are being measured are:

Rule_1 - Rule_6: Detection rules from rules file
Archive_Count: The number of archives created by day
Mail_Count: The number by day of emails sent to a personal email provider with an attachment
Rename_Count: The number of files renamed in a day
NTU_Count: The number of files uploaded to an external source in a day
Weighted: The sum by day of additional values given to rule detections
Day_Count: The total count of all events by day

This method won't tell you 100% that anomaly is malicious. But you may be able to infer what the user was doing and the likely hood of needing further investigation. An example may be 2 archive events + 2 email events may result in 2 zip files being emailed to a personal email address. That may or may not need investigating depending on the users role.