Incognia - An Exploratory Data Analysis

The main target of this notebook is to ***analyze*** and ***understand*** the ***Incogia database***, discovering ***patterns*** ans raising ***hypothesis*** about ***behaviors*** that might be ***fraudulent***.

Given that there is no information concerning:

- Events/devices that were, in fact, assessed as high risk
- Events/devices that were, in fact, fraudulent

The data provided will be used to infer which devices and events could have been assessed as ***high risk*** and culminated in a ***fraudlent action***.
For that, an attempt to replicate some of ***Incognia's heuristics*** will be done, then these devices and events will be better analyzed and ***compaired to low risk entities.***

Analysis Plan

This analysis will be divided into:

- Cleaning and Transformation
- Overview
- Risk Assessment
- Hypothesis and Analysis
- Conclusions

Cleaning and Transformation

In [840…

```
# import
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
```

In [841…

```
# SET OPTIONS
sns.set_style('darkgrid')
pd.set_option('display.float_format', lambda x: '%.2f' % x)

#GET DATA
df = pd.read_csv('./data.csv')

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 444758 entries, 0 to 444757
Data columns (total 10 columns):
 #   Column                        Non-Null Count   Dtype  
---  ------                        --------------   -----  
 0   event_id                      444758 non-null  object 
 1   event_timestamp               444758 non-null  int64  
 2   account_id                    444758 non-null  int64  
 3   device                        444758 non-null  int64  
 4   distance_to_frequent_location 444080 non-null  float64
 5   device_age_days               444758 non-null  int64  
 6   is_emulator                   444758 non-null  bool   
 7   has_fake_location             444758 non-null  bool   
 8   has_root_permissions          444758 non-null  bool   
 9   app_is_from_official_store    444758 non-null  bool   
dtypes: bool(4), float64(1), int64(4), object(1)
memory usage: 22.1+ MB
```

In [842…

```
# TRANSFORM INT TO DATETIME
df['event_timestamp']= pd.to_datetime(df['event_timestamp'], unit='ms')

# CHECK FOR ERROS
df['event_timestamp'].isnull().sum()
```

Out[842]:

```
0
```

In [843…

```
# CALCULATE MONTHLY EVENTS DISTRIBUTION
df['event_timestamp'].dt.strftime('%Y-%m').value_counts()
```

Out[843]:

```
2021-07    444758
Name: event_timestamp, dtype: int64
```

In [844…

```
# CREATE SEGMENTED COLUMNS FOR DATETIME
units = {'day': 'd', 'hour': 'H', 'minute': 'M', 'week_day': 'A', 'week_day

for key, value in units.items():
    df[key] = df['event_timestamp'].dt.strftime(f'%{value}')
```

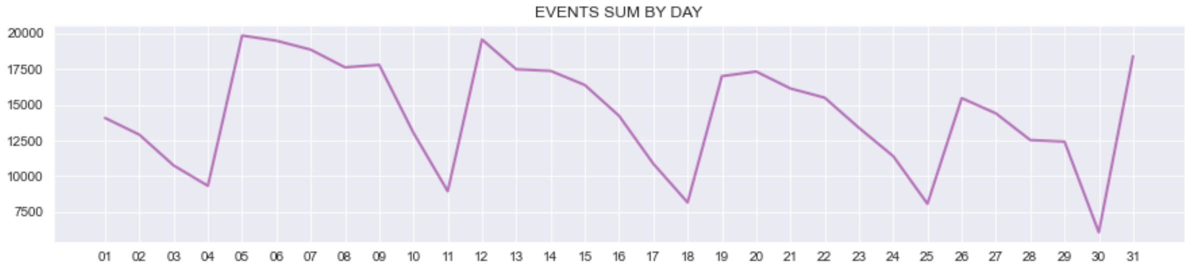
Overview

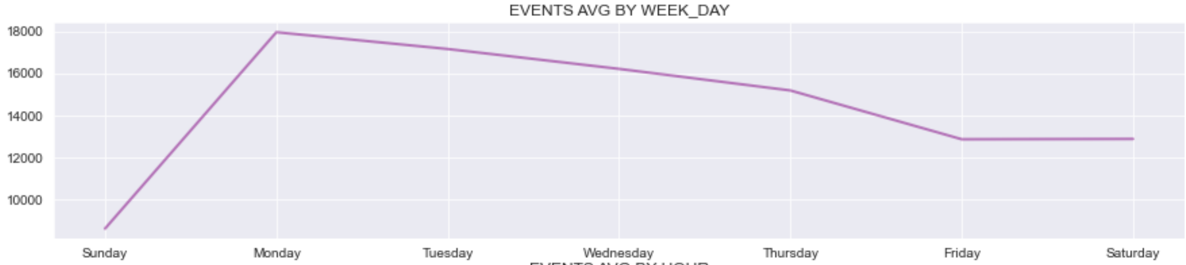
In order to have a sense of the general baheavior, the volume of events are plotted segmented by different dimensions:

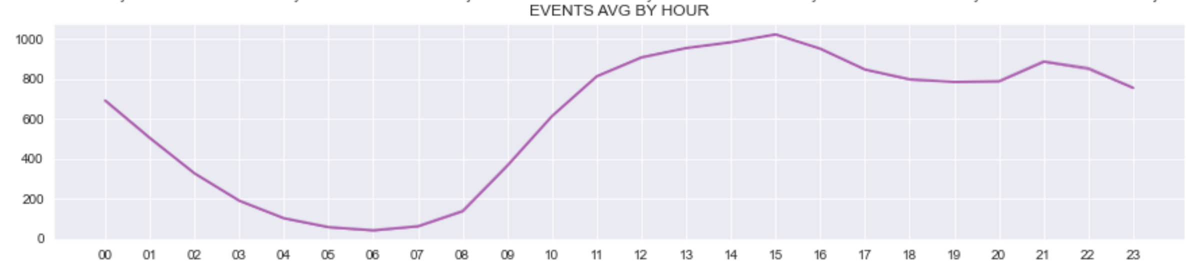
In [845…

```
# SELECT DIMENSIONS TO SEGMENT
groups = ['day', 'week_day', 'hour']
graphs = len(groups)

# CREATE FIGURE TO PLOT
fig, axs = plt.subplots(graphs,1, figsize=(15,10))

# LOOP FOR EACH DIMENSION AND CREATE AGREGATED MEASURES
for group in groups:
    ax = groups.index(group)
    vol = (df.groupby([group, 'week_day_n'])
           .agg({
               'event_id': 'count',
               'event_timestamp': lambda x: x.dt.day.nunique()
           })
```

```
            .reset_index()
            .sort_values(['week_day_n' if group=='week_day' else group])
           )

# CALCULATE AVG FOR WEEK_DAY AND KEEP THE SUM FOR THE OTHERS
    if group != 'day':
        vol['event_id'] =vol['event_id']/vol['event_timestamp']
        title= f'EVENTS AVG BY {group.upper()}'
    else:
        title= f'EVENTS SUM BY {group.upper()}'
# PLOT DATA
    sns.lineplot(data=vol, x=group, y='event_id', color='purple', alpha=.5,
    axs[ax].set(title=title, xlabel=None, ylabel=None)
```







For the `week_day` and `hour` dimensions, the average amount of events was calculated in order to ***mitigate a biased view*** due to the different number of occurrences on `week_days` and a unusual concentration of events on a single `hour`.
For sake of simplicity ***this risk was not totally avoid using median*** as the meausere, since the purpose of these visuals is only providing a glance of ***general behavior***.

- At the first graph a ***weekly pattern*** stands out, in which there are a relative low level of events on every 7 days with subsequential peaks.
- From that interpretation, analyzing the second graph it is possible to infer that the ***highs occurs on mondays while the lows do on sundays***.
- The last graph shows a behavior that could be highly related to ***bussiness hours***, especially considereing the previous graphs.

Risk Assessment

In this section some of the criterias used by Incognia in its heuristics are analyzed to replicate in some way the process of risk assesment.

Based on the data provided, the risk will be assessed and categorized following the topics:

- ***Device Integrity***
  - Location Risk: `has_fake_location` , `is_emulator` , `has_root_permissions`
  - Malware Risk: `has_root_permissions` and `app_is_from_official_store`

- ***Device Behavior***

  - Multiple Account Devices
  - Device Age
  - Distance to Freqquent Locations

- **_Account*Integrity***

  - Unusual Concentration of Events

Device Interegrity

The available data concerning the ***device integrity*** will be explored to create an ***watchlist and classify*** risk for devices in this database.

It was observed on the previous sections that the `distance_to_frequent_location` is the single column in which some of the registers are ***null***.
That may be a indication that during the occurence of such events the ***devices sensors*** were turned off or the user had not given ***location permissions*** the app.
Thus a new column will be created to classify the risk related to the `distance_to_frequent_location` , and the registers mentioned before will be tagged as `unknown`

In [846…

```
df.loc[df['distance_to_frequent_location'].isnull(), 'device_integrity'] =
```

In [847…

```
# CREATE NEW COLUMN TO FACILITATE DATA MODELING
df['app_not_from_official_store'] = df['app_is_from_official_store'].apply(

interventions = ['is_emulator', 'has_fake_location', 'has_root_permissions'

# CALCULATE SHARE OF OCCURRENCES
interventions_distribution = df[interventions].apply(sum, axis=0)
interventions_distribution/interventions_distribution.sum()
```

Out[847]:

```
is_emulator                    0.00
has_fake_location              0.03
has_root_permissions           0.42
app_not_from_official_store    0.55
dtype: float64
```

Takeaways

`has_root_permission` and `app_not_from_official_store` constitute the majority of intervention ocurrences on device integrity.
That result is expected, given that they are not directly related to fraudsters, even though its occurrence ***increases the device's vulnerability and the risk of an ATO***

In [848…

```
# CLASSIFY RISK BASED ON DEVICE INTEGRITY
for intervention in interventions:
    df.loc[df[intervention]==True, 'device_integrity'] = 'high_risk'

df.loc[df['device_integrity']!='high_risk', 'device_integrity'] = 'low_risk

# CHECK RESULTS
display(df['device_integrity'].value_counts())
```

```
low_risk     443664
high_risk      1094
Name: device_integrity, dtype: int64
```

Takeaways

The assesment made on the device's integrity shows a unbalanced database, which is a common aspect analyzing fraud events.

In [849…

```
# CALCULATE NUMBER OF CONCURRENT INTERVENTIONS
df['n_interventions'] = df[['is_emulator',
                            'has_fake_location',
                            'has_root_permissions',
                            'app_not_from_official_store']
                          ].apply(sum, axis=1)

# CHECK RESULTS
df['n_interventions'].value_counts()
```

Out[849]:

```
0    443664
1      1090
2         4
Name: n_interventions, dtype: int64
```

Takeaways

Calculating the number of concurrent interventions on a device during the same event points out that ***rarely there is more than one intervention on a device***.
Thus, these cases will be quickly analyzed for a better understanding.

In [850…

```
df.query('n_interventions > 1').sort_values(['device','event_timestamp'])
```

Out[850]:

	event_id	event_timestamp	account_id	device	distance_to_frequent_loca
174510	23814bb9-dac8	2021-07-06 17:05:45.651	957966161	40111468	51
288515	e55c9740-3aed	2021-07-25 16:18:16.835	1393282325	40111468	
183518	58893316-12eb	2021-07-08 12:47:33.611	1212353253	420849688	
29645	275811e0-17b7	2021-07-14 21:18:29.487	442820130	1425676062	1

Takeaways

Even though 2 different accounts were accessed from the same device, both interventions in these cases were more related to malware vulnerabilities. Whereas the last row of the dataframe shows an event in which the device `is an emulator`, the `device_age_days` is ***0 days*** old and besides being nearly 15 meters to a trusted location, it has `root permissions` and could be ***spoofing GPS location***

In [851…

```
#CREATING A WATCHLIST

# ADD DEVICES THAT HAD SOME INTERVENTION DURING THE ENTIRE OBSERVED PERIOD
watchlist = df.query('device_integrity=="high_risk"')['device']
watchlist = np.array(watchlist)

# TAG DEVICES THAT ARE IN THE WATCHLIST
df['watchlist'] = df.apply(lambda x: True if x['device'] in watchlist else 

# CHECK RESULTS
df.query('watchlist==True')[['account_id', 'device']].nunique()
```

Out[851]:

```
account_id    1338
device         766
dtype: int64
```

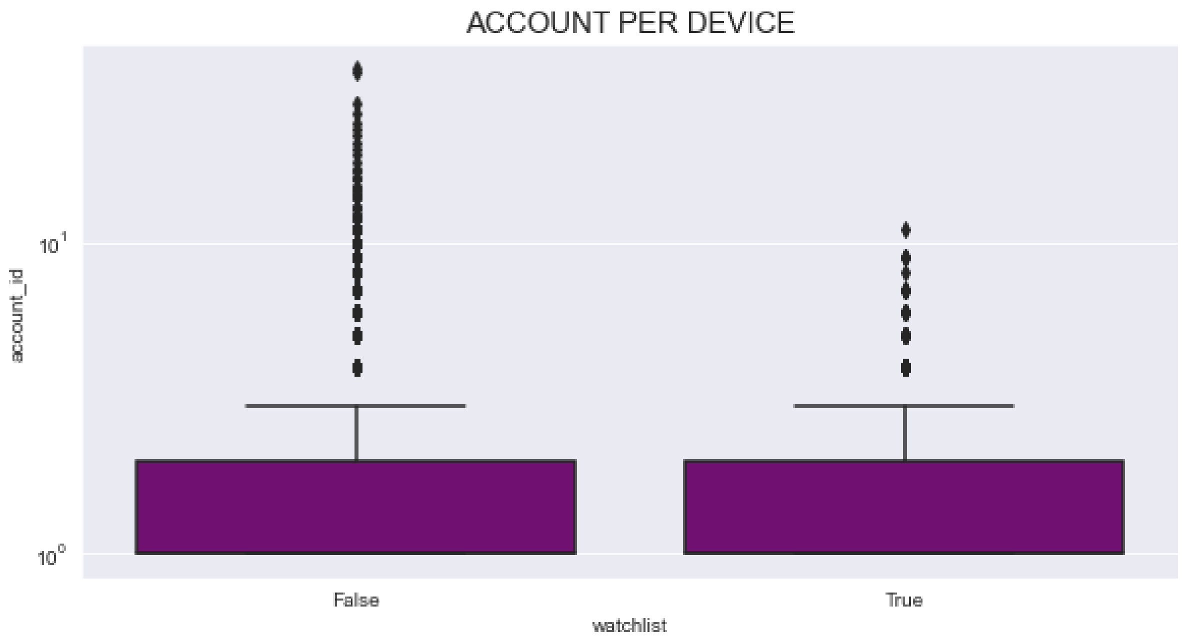
Takeaways

The number of accounts accessed by devices in the watchlist are almost the double of the latter.
At first, an ***average number of accounts accessed from the same device*** nearly to 2 seems regular.
However, the distribution across the different devices may point to some ***outliers***, as well as increase risk assessed for some device/event. This ***distribution*** will be analyzed in the next section.

Device Behavior

In [852…

```
acc_p_devc = df.groupby('device').agg({'account_id':'nunique', 'watchlist':

fig, axs = plt.subplots(1,1, figsize=(10,5))

sns.boxplot(data=acc_p_devc, x='watchlist', y='account_id', ax=axs, color='
axs.set_title('ACCOUNT PER DEVICE', fontsize=15)
axs.set(yscale='log');
```



In [853…

```
in_watchlist = acc_p_devc.query('watchlist==True')['account_id']
out_watchlist = acc_p_devc.query('watchlist==False')['account_id']

devices = {'in watchlist' : in_watchlist, 'out watchlist' : out_watchlist}

print('QUARTILES:\n25%,50%,75%')

for key, value in devices.items():
    percentiles = np.percentile(value, (25, 50, 75))
    print(f'{percentiles} {key}')
```

```
QUARTILES:
25%,50%,75%
[1. 1. 2.] in watchlist
[1. 1. 2.] out watchlist
```

Takeaways

Curiously, the boxplot shows an unexpected behavior:

- Devices that had its ***integrity*** assessed as `high_risk` have a very similar distribution.
- More than that, devices assessed as `low_risk` reach higher number of accounts accessed by the same device.

Given that, those devices that are considered `outliers` at the boxplot representation will be classified with a `high_risk` tag for `device_behavior` .
Since the distribution are very similar, they will be considered as one.

In [854…

```
# CREATE A FUNCTION TO CALULATE SUPERIOR OUTLIERS
def outliers (dist):
    q1 = np.nanpercentile(dist, 25)
    q3 = np.nanpercentile(dist, 75)
    iq = q3-q1
    sup_out = q3 + (1.5*iq)
    inf_out = q1 - (1.5*iq)
    return [inf_out,sup_out]
```

In [855…

```
# CALCULATE INFERIOR AND SUPERIOR LIMIT FOR OUTLIERS
out = outliers(in_watchlist)
out
```

Out[855]:

```
[-0.5, 3.5]
```

In [856…

```
# STORE DEVICES THAT ACCESS MORE THAN 2 ACCOUNTS TO BE CLASSIFIED
acc_p_devc.reset_index(inplace=True)
devices_id = acc_p_devc.query('account_id > @out[1]')['device'].unique()
devices_id = np.array(devices_id)

#CHECKING AMOUNT OF OUTLIERS DEVICES
len(devices_id)
```

Out[856]:

```
14750
```

In [857…

```
#CALCULATE THE % OF DEVICES IN EACH GROUP THAT HAS A OUTLIER BEHAVIOR
total_devices = acc_p_devc.groupby('watchlist')['device'].nunique()
deviant_devices = acc_p_devc.query('device in @devices_id').groupby('watchl

(deviant_devices/total_devices)
```

Out[857]:

```
watchlist
False    0.05
True     0.08
Name: device, dtype: float64
```

Takeaways

The assessment of ***device behavior*** has a signifcantly ***broader effect on the risk assessment*** than device integrity assesment (high_risk rate ~ 0%).
It is likely that the rate of ***false positives*** in this asssessment is higher than the previous as well.
Anyways, the ***combined analysis*** of both assessments will provide a result that is more robust than the individual analysis of each one.

In [858…

```
# CLASSIFY RISK RELATED TO DEVICE BEHAVIOR
df.loc[df['device'].isin(devices_id), 'device_behavior'] = 'high_risk'
df.loc[~df['device'].isin(devices_id), 'device_behavior'] = 'low_risk'
```

In [859…

```
# CALCULATE THE DISTRIBUTION OF EVENTS BASED ON RISK ASSESSMENT RESULTS FOR
df['device_behavior'].value_counts(normalize=True)
```

Out[859]:

```
low_risk     0.84
high_risk    0.16
Name: device_behavior, dtype: float64
```

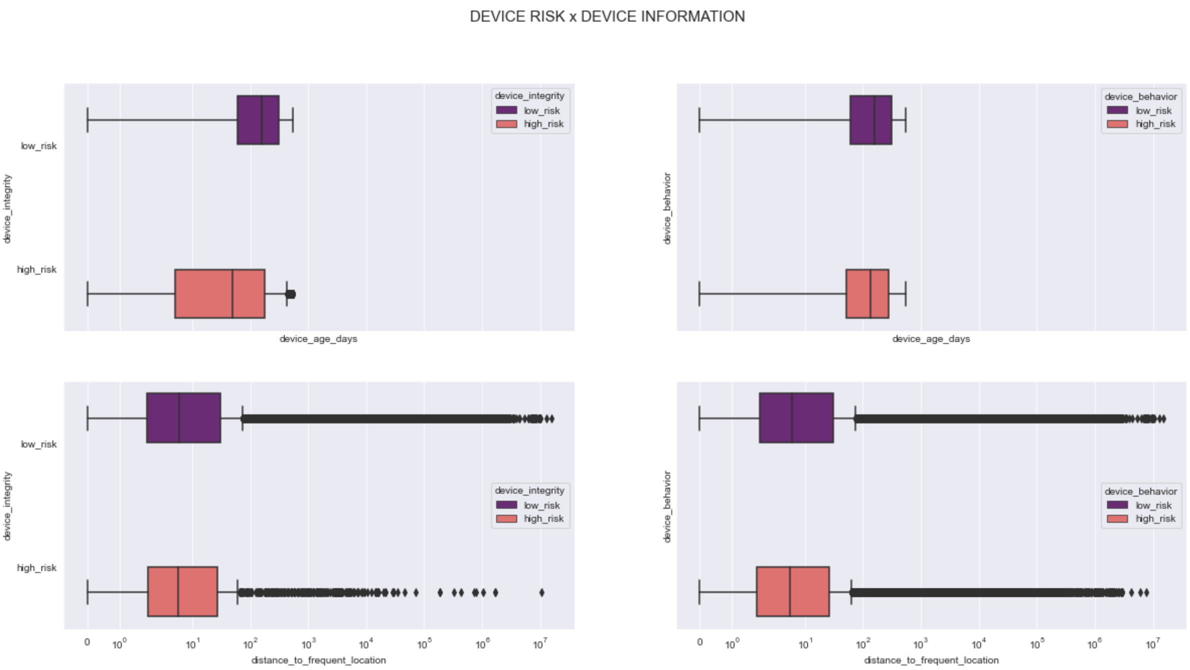
Takeaways

Besides, the percentage of devices that have a suspicious behavior being between 5% and 8%, the percentage os events executed through it represents 16% of the whole database.

It is because, by definition, the minimum value of accounts accessed by those devices is 3.5.

In [860…

```
#CREATE LISTS TO SEGMENT AND PLOT GRAPHS

dimensions = ['device_age_days', 'distance_to_frequent_location']
risks = ['device_integrity', 'device_behavior']

fig, axs = plt.subplots(2,2, figsize=(20,10), sharey=True, sharex=True)

#LOOP THROUGH LIST TO PLOT
for dim in dimensions:
    for risk in risks:
        ax_v=dimensions.index(dim)
        ax_h=risks.index(risk)
        sns.boxplot(data=df,
                        x= dim,
                        y= risk,
                        hue= risk,
                        palette='magma',
                       ax=axs[ax_v][ax_h])
        axs[ax_h][ax_v].set(xscale='symlog')
plt.suptitle('DEVICE RISK x DEVICE INFORMATION', fontsize=15);
```



Takeaways

At this graphs is remarkable how the distribution of the `device_age_days` differs from the `distance_to_frequent_location` .
It is possible to conclude:

- `device_age_days` :
  - Has a lower variability of values compared to the `distance_to_frequent_location` .
  - There is a high probability that devices accessing one account will be older than 100 days, apart from devices with `high_risk of integrity` .
  - The previous point strengthens the risk assessemnt of that kind of devices.
  - There is no evidence that differs `device_behavior` ***low and high risks*** events

- The median of high risk integrity devices will be used as ***baseline*** to classify `device_age_days` risk, once it is out of the general pattern.

- `distance_to_frequent_location` :
  - There are multiple outliers, which could be expected considering users mobility, but its scale depends much on the nature of the client's product/service.
  - In the same way, there is a high likelihood that an event will occur up to 100 meters from a trusted location.
  - There is no indication that differs `device_behavior` and `device_integrity` ***low risk*** events from ***high risk*** events
  - The `outliers` can be a good option for classifiyng ***high risk distances*** in order to scrutinize events.

In [882…

```
# CALCULATING MEDIAN DEVICE_AGE_DAYS FOR HIGH RISK DEVICE INTEGRITY
devc_integrity = df.query('device_integrity=="high_risk"')
age_median = devc_integrity['device_age_days'].median()
```

Takeaways

As already observed, the median value of `device_age_days` for high_risk devices is lesser than the general pattern.

> One hypothesis for this fact is that fraudsters avoid to access the same account for extended periods, which could decreases his/her risk exposition

In [883…

```
#CLASSIFY AGE OUTLIERS
df.loc[df['device_age_days']<age_median, ['age_outlier']] = True
df.loc[df['device_age_days']>=age_median, ['age_outlier']] = False
```

In [884…

```
# CALCULATE DISTANCE THAT ARE CONSIDERED OUTLIERS IN THE BOXPLOT
distance_out = outliers(df['distance_to_frequent_location'])[1]
distance_out
```

Out[884]:

```
74.12444401918893
```

In [885…

```
# PERCENTAGE OF EVENTS THAT ARE OUTLIERS BASED ON TRUSTED LOCATIONS
len(df.query('distance_to_frequent_location> @distance_out'))/len(df)
```

Out[885]:

```
0.20780739188502512
```

Takeaways

Apparently there is a strong trending of people executing an event in places around trusted locations.
However, the cases considered as outliers are equivalent to 20% of the events.
This amount is relatively high, nevertheless the combination with other criterias may provide a more accurate risk assessment.

In [886…

```
# CLASSIFY DISTANCE OUTLIERS
```

```
df.loc[df['distance_to_frequent_location']>distance_out, ['distance_outlier
df.loc[df['distance_to_frequent_location']<=distance_out, ['distance_outlie
```

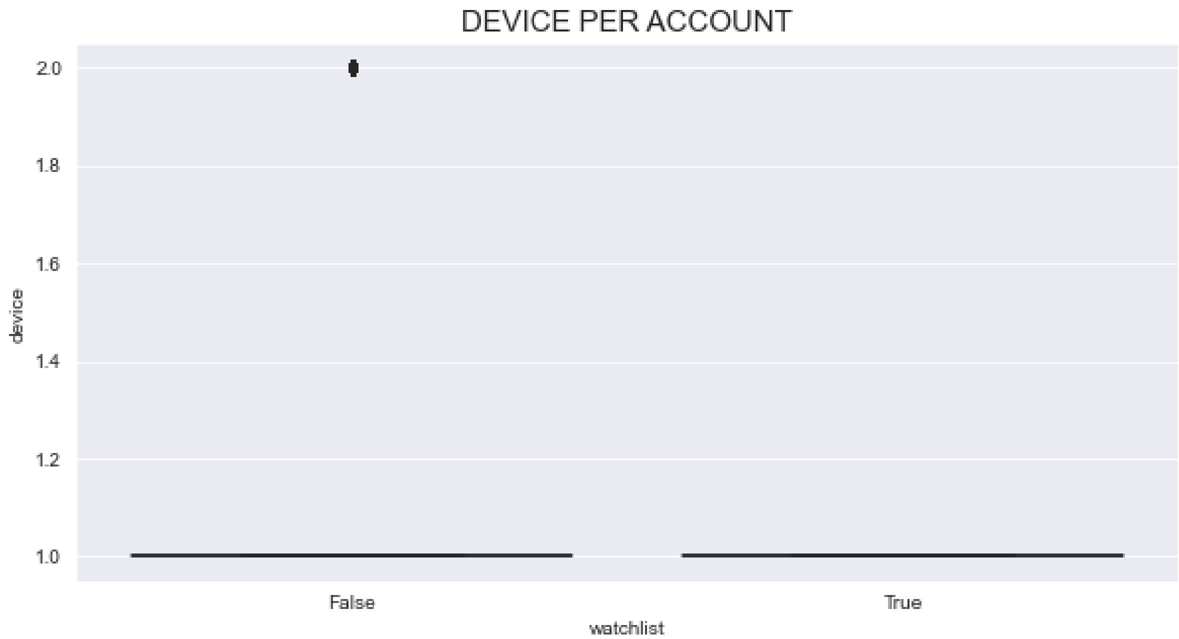
> The classfications of `multiple account devices` , `distance to trsuted location` and `device age` were all consolidated in one.
> For this, the cases in which all three attributes were tagged as `high_risk` or `outlier` were reclassified as `high_risk` as well

Account Integrity

The target of this section is understanding patterns about account usage and define unusual concentration of events that might be related to fraud attempts

In [888…

```
devc_p_acc = df.groupby(['account_id', 'watchlist']).agg({'device':'nunique

fig, axs = plt.subplots(1,1, figsize=(10,5))

sns.boxplot(data=devc_p_acc, x='watchlist' ,y='device', ax=axs, color='purp
axs.set_title('DEVICE PER ACCOUNT', fontsize=15);
```



Takeaways

At this boxplot it is clear that de amount of devices that access one account is roughly 1/account.
This behavior is interesting, because it was verified that the number of account per devices reachs values way bigger.
It could be, and probably is, a reflection of the ***regular use*** of the app.
Anyways, this fact stands out even more the cases of multiple account device, since these two relations ***(device/account and account/device)*** usually are nearly 1:1.
In conclusion, no `high_risk` classification will be done concerning this evidence.

In [889…

```
# NUMBER OF ACCOUNTS THAT WERE ACCESSED BY MORE THAN 1 DEVICE
devc_p_acc.query('device>1')['account_id'].nunique()
```

Out[889]: 36

There is a hypothesis that fraudster could act trying to perfom the same or different events in a small period of time, in order to execute as much events as they can.
Thus, it was calculated the amaount of events performed on the same account at the same day and hour.

In [100…

```
# CALCULATE AMOUNT OF EVENTS THAT HAVE HAPPENED AT THE SAME DAY AND HOUR FO
same_hour_acc_events = df.groupby(['account_id', 'day', 'hour']).agg(
                                                                    {'event_id
                                                                     'device':

same_hour_acc_events.rename(columns={'event_id':'number_of_events'}, inplac

# PERCENTAGE DISTRIBUTION
same_hour_acc_events[['number_of_events']].value_counts(normalize=True)
```

Out[1006]:

```
number_of_events
1                   1.00
2                   0.00
dtype: float64
```

In [100…

```
#ABSOLUT DISTRIBUTION
same_hour_acc_events[['number_of_events']].value_counts()
```

Out[1007]:

```
number_of_events
1                   444756
2                        1
dtype: int64
```

Takeaways

Here we see that is the usually a regular usage of the application does not require more than two event triggerd within the same hour for the same account.

Takeaways

These last matrices shows that 99% of the events occur in different accounts, days and hours.
Similarly, for just 5 times there was a device that accessed the same account, at the same day and hour.
Thus, considering that the regular usage of the application requires 1 or 2 daily events at the same hour, the cases in which it happened 3 or 4 times will be taggged as `high_risk`

In [100…

```
# REMODEL DATA AND AGREGATING DATA TO ASSESS RISK
df['multiple_account'] = df['device_behavior'].apply(lambda x: x=='high_ris
df['device_behavior'] = df[['distance_outlier', 'age_outlier', 'multiple_ac
df['device_behavior'] = df['device_behavior'].apply(lambda x: 'high_risk' i
```

In [100…

```
df['device_behavior'].value_counts()
```

Out[1009]:
```
low_risk     441379
high_risk      3379
Name: device_behavior, dtype: int64
```

Hypothesis and Analysis

An ATO can happen mainly by two manners:

- A Fraudster gets physical access to the victim's device
- A Fraudster gets virtual access to the victim's device

In the first case information about geolocation may be of great help, since the device will get out of the usual tracking.
However, in the second case the geolocation will be probably forged.

Thus, `device_behavior` `high_risk` events will be used as an approximation to ***physical fraudulent actions***, whilst `device_integrity` `high risk` events will be to ***virtual frauds.***

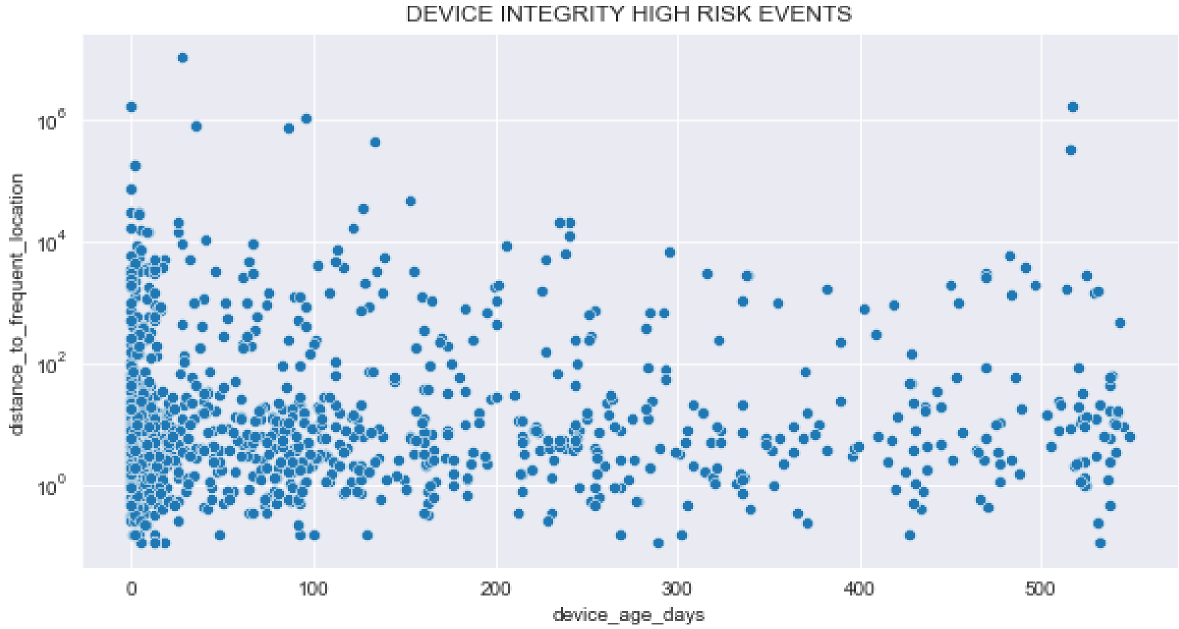
In [101…
```
# UNDERSTANDING HOW THE RISKS ARE DISTRIBUTED
df[['device_integrity', 'device_behavior']].value_counts()
```

Out[1010]:
```
device_integrity  device_behavior
low_risk          low_risk           440296
                  high_risk            3368
high_risk         low_risk             1083
                  high_risk              11
dtype: int64
```

Virtual Frauds

In [101…
```
# EVNTS OF DEVICES WWITH HIGH RISK INTEGRITY
devc_int = df.query('device_integrity=="high_risk"')
```

In [101…
```
fig, axs = plt.subplots(1,1, figsize=(10,5))
sns.scatterplot(data=devc_int, x='device_age_days', y='distance_to_frequent_
axs.set_title('DEVICE INTEGRITY HIGH RISK EVENTS')
axs.set(yscale='log');
```



Takeaways

This graph displays how devices with high risk of integrity imitate some regular behaviors. However, a part from the `distance_to_frequent_location` being con cenrate around 0 anf 100 m, similarly to genuine events characteristics, the device age gives a hint that those events are suspicious. In other words, the devices' age are concentrate, around 0-100 days. Nevertheless, as analyzed before, most part of the events happens through devices older than 100 days.

In [101…

```
# TOP 5 DAYS WITH THE BIGGEST VOLUME OF POSSIBLE FRAUDULENT EVENTS
((devc_int.groupby(['day', 'week_day']).agg({'event_id':'nunique'}))
 /len(devc_int)
).reset_index().sort_values('event_id', ascending=False).head()
```

Out[1013]:

	day	week_day	event_id
19	20	Tuesday	0.05
4	05	Monday	0.05
30	31	Saturday	0.05
5	06	Tuesday	0.05
18	19	Monday	0.04

Takeaways

The table above brings the TOP 5 days in volume of suspicious events.
it might be a coincidence, given that some of the top days of the month are also the ones that are expected to be the top of the weeks.
Anyways, the first 3 days are usually payday in a significant amount of companies, which raises the hypothesis that our client could operate banking services, and these days of the month represent a bigger opportunity to profit.

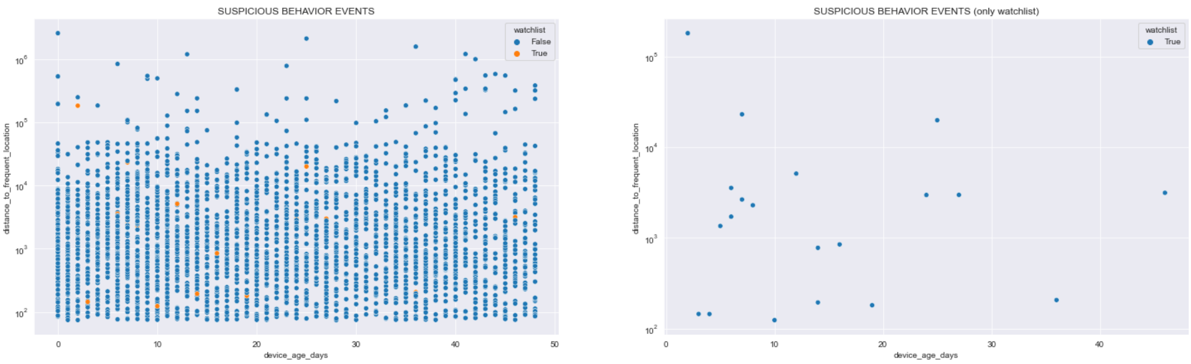
Physical Frauds

In [101…

```
#EVENTS OF DEVICES THAT HAVE BEHAVED SUSPISICOUSLY
devc_behav = df.query('device_behavior=="high_risk"')
```

In [101…

```
fig, axs = plt.subplots(1,2, figsize=(25,7))
sns.scatterplot(data=devc_behav, x='device_age_days', y='distance_to_freque
sns.scatterplot(data=devc_behav.query('watchlist==True'), x='device_age_day
axs[0].set(yscale='log')
axs[0].set_title('SUSPICIOUS BEHAVIOR EVENTS')
axs[1].set(yscale='log')
axs[1].set_title('SUSPICIOUS BEHAVIOR EVENTS (only watchlist)');
```



Takeaways

As mentioned before, the amount of suspicious `device_behavior` events is much bigger than `device_integrity`.
This is due to the way that the first was classified, once that the outliers were used to separate a group with higher likelihood to perform a fraudulent act. That is why we see, on the left, a graph that has no behavior pattern.
However, it is possible to use that information and narrow the perspective a bit more. That is done on the right, filtering just devices that were at the ***watchlist*** .
Once more, there is no relevant pattern. However the points are concentrate on the left bottom, as well as the points of the ***Virtual Frauds*** (previous topic)

Conclusions

The main conclusions of this analysis are that:

- Device Integrity is the strongest piece of evidence concerning a possibile fraud
- Within that sort of risk, the one that are more strongly related to ***malwares*** (*rooting and apps from unoficial sources*) represent more than 90% of the events.
- Still, rooting a device may be used to *spoof GPS location*. That fact stands out the importance to relate ***risks that when combined generate higher scenario risk than individually.***
- Storing and processing a ***watchlist*** based on historic behavior ***levarages the risk assessemnt potentcial***
- During the month analyzed *(July-21)*, the **_high*risk*** events have not presented any seasonal patttern (hourly, daily, weekly)
- Considering median values of application's regular usage, ***events tend to happen up to 100 meters from a trusted location and the devices used are aorund 100 days***

years old as well, which may be related to the beginning of the devices data tracking
- Similarly, usually it is expected that one account will be accessed by only one device and that no more than 1 event happen within the same day and hour

As next steps of the analysis, it is important to learn about the client's business context, which could help to raise hyposthesis for better inference and understanding of data patterns. Finally, information about events that were, in fact, fraudlents and false positives are importante as well to a more accurate analysis.