router_log_analysis_assignment

August 29, 2023

1 Simulating various Attacks on a Household router network.

We will first import necessary libraries

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import random

# changing style
plt.style.use('default')
plt.rcParams["font.family"] = "Jetbrains Mono"
```

1.1 Strategy

1. We will try and simluate a few attacks on a router, and check whether those attacks can be detected in hindsight.
2. To do that we will start with generating some demo data for a router, inspired by my home router. This will be a monitor of active DHCP Clients.
3. We will then try and analyse the data to find out anomalies in normal usage.

2 Generating *normal* demo data

[ ]:
```
# columns
data = {
    'MAC' : [],
    'IP Address': [],
    'Device Name': [],
    'Interface': [],
    'Requested IP': [],
    'Time': []
}
```

[ ]:
```
# Creating a pandas dataframe

normal_log_db = pd.DataFrame(data)
```

```
normal_log_db
```

[ ]:
```
# Writing functions for columns that we wanna generate randomly
def generate_mac_address():
    mac = [random.randint(0x00, 0xff) for i in range(6)]
    return ':'.join(map(lambda x: "%02x" % x, mac))


def generate_dest_ip_address():
    # define the weights for each website
    website_weights = {'Youtube': 15, 'Instagram': 10, 'Facebook': 8, 'Twitter':␣
 ↪5, 'Other': 2}

    # create a list of websites based on their weights
    websites = []
    for website, weight in website_weights.items():
        websites.extend([website] * weight)

    # randomly select a website from the list
    website = random.choice(websites)

    # generate a random IP address for the website
    if website == 'Youtube':
        return ('216.58.194.45' , website)
    elif website == 'Instagram':
        return ('3.213.31.34' , website)
    elif website == 'Facebook':
        return ('69.63.176.22' , website)
    elif website == 'Twitter':
        return ('104.244.42.12' , website)
    else:
        return ('192.168.1.53' , website)

def generate_device_ip_address():
    # define a list of 10 predefined IP addresses
    ips = ['192.168.1.10', '192.168.1.20', '192.168.1.30', '192.168.1.40', '192.
 ↪168.1.50',
           '192.168.1.60', '192.168.1.70', '192.168.1.80', '192.168.1.90', '192.
 ↪168.1.100']

    # generate a random integer between 0 and 9
    index = random.randint(0, 9)

    # return the IP address at the selected index
    return ips[index]

def generate_device_name():
```

```
    device_names = ['iPhone', 'Samsung', 'OnePlus', 'Nokia', 'Xiaomi', 'Oppo',␣
↪'Vivo', 'Realme', 'Micromax', 'Lenovo']
    return random.choice(device_names)

def generate_interface():
    interfaces = ['5gz', '2.4gz']
    return random.choice(interfaces)

def generate_date_time():
    # generate random date and time, but only in the range of a few days
    start_date = pd.to_datetime('2023-01-01')

    # generate random number of days
    days_to_add = random.randint(0, 10)

    # generate random number of seconds
    seconds_to_add = random.randint(0, 86400)

    # add random days and seconds to start date
    end_date = start_date + pd.Timedelta(days=days_to_add,␣
↪seconds=seconds_to_add)

    # set the hour of the timestamp based on the time of day
    hour = end_date.hour
    if hour < 6:
        # almost no traffic between 2am and 6am
        hour = random.randint(6, 23)
    elif hour < 9:
        # more traffic during the morning hours
        hour = random.randint(6, 10)
    elif hour < 18:
        # most traffic during the daytime
        hour = random.randint(9, 17)
    else:
        # less traffic during the evening hours
        hour = random.randint(17, 23)

    # set the hour of the timestamp
    end_date = end_date.replace(hour=hour)

    # return timestamp as string
    return end_date.strftime('%Y-%m-%d %H:%M:%S')

def gen_protocols():
    protocols = ['TCP', 'UDP', 'DHCP', 'HTTP', 'HTTPS', 'FTP', 'SMTP', 'POP3',␣
↪'IMAP', 'DNS', 'ICMP']
    ports = {
```

```
        'TCP': 21,          # HTTP
        'UDP': 53,          # DNS
        'DHCP': 67,         # DHCP Server
        'HTTP': 80,         # Hypertext Transfer Protocol
        'HTTPS': 443,       # HTTP Secure (TLS/SSL)
        'FTP': 21,          # File Transfer Protocol (Control)
        'SMTP': 25,         # Simple Mail Transfer Protocol
        'POP3': 110,        # Post Office Protocol v3
        'IMAP': 143,        # Internet Message Access Protocol
        'DNS': 53,          # Domain Name System
        'ICMP': None        # Internet Control Message Protocol (does not use␣
↪ports)
    }
    weights = [0.3, 0.2, 0.1, 0.15, 0.1, 0.05, 0.05, 0.025, 0.025, 0.025, 0.030]
    selection = random.choices(protocols, weights=weights)[0]
    return (selection, ports[selection])
```

[36]:
```
# Generate normal data, consider a home environment. with 10 users. across a␣
↪span of 10 days. Visiting 100 websites per device per day.

normal_log_db = pd.DataFrame(columns=['MAC', 'IP Address', 'Device Name',␣
↪'Interface', 'Requested IP', 'Time'])

for i in range(10):
    temp_df = pd.DataFrame({
        'MAC' : [generate_mac_address() for j in range(100)],
        'IP Address': [generate_device_ip_address() for j in range(100)],
        'Device Name': [generate_device_name() for j in range(100)],
        'Interface': [generate_interface() for j in range(100)],
        'Requested IP': [generate_dest_ip_address()[0] for j in range(100)],
        'Requested Website': [generate_dest_ip_address()[1] for j in range(100)],
        'Protocol': [gen_protocols()[0] for j in range(100)],
        'Port': [gen_protocols()[1] for j in range(100)],
        'Time': [generate_date_time() for j in range(100)]
    })

    normal_log_db = pd.concat([normal_log_db, temp_df], ignore_index=True)

normal_log_db
```

[36]:

	MAC	IP Address	Device Name	Interface	Requested IP	\
0	ff:39:6e:d3:c9:e3	192.168.1.100	Micromax	5gz	69.63.176.22	
1	9b:b1:86:3a:a2:87	192.168.1.30	Lenovo	5gz	3.213.31.34	
2	63:6c:47:95:0d:9e	192.168.1.80	Nokia	2.4gz	104.244.42.12	
3	32:66:c6:a6:2a:14	192.168.1.70	OnePlus	5gz	216.58.194.45	
4	e9:63:7e:f8:c0:9d	192.168.1.70	iPhone	2.4gz	3.213.31.34	
..	...	...	...	...	...	...

```
995  af:83:ca:22:ed:17   192.168.1.70     OnePlus       5gz    3.213.31.34
996  8a:af:4d:ae:3b:7d   192.168.1.10       Oppo        5gz    3.213.31.34
997  de:c7:c7:1c:49:36  192.168.1.100    Micromax     2.4gz  216.58.194.45
998  d1:7e:fc:b7:e0:d0   192.168.1.40      Realme       5gz    3.213.31.34
999  0a:0a:08:2b:78:34   192.168.1.30      Realme     2.4gz    3.213.31.34

                    Time Requested Website Protocol   Port
0    2023-01-05 23:33:06          Facebook     HTTP   53.0
1    2023-01-06 21:26:17          Facebook      DNS    NaN
2    2023-01-06 21:40:24         Instagram      TCP   53.0
3    2023-01-01 06:59:25          Facebook    HTTPS   21.0
4    2023-01-11 16:11:33           Youtube      TCP  443.0
..                   ...               ...      ...    ...
995  2023-01-06 18:06:25           Youtube      TCP   21.0
996  2023-01-06 07:17:27           Youtube    HTTPS   21.0
997  2023-01-10 23:54:32           Twitter      DNS   80.0
998  2023-01-04 08:05:22          Facebook     POP3   21.0
999  2023-01-02 10:11:01          Facebook      TCP  443.0

[1000 rows x 9 columns]
```

2.1 Plotting Number of Requests daily.

[ ]:
```
# let us plot the number of requests per day
normal_log_db['Time'] = pd.to_datetime(normal_log_db['Time'])
normal_log_db['Date'] = normal_log_db['Time'].dt.date

# sort data by date
normal_log_db = normal_log_db.sort_values(by=['Date'])

normal_log_db.head()

# now let us plot the number of requests per day
dates = normal_log_db['Date'].value_counts()

# sorting dates
dates = dates.sort_index()

# creating the plot.
fig, ax = plt.subplots(figsize=(20, 8))

# informative title + subtitle
title = 'The Number of Requests Made per day by the Household'
subtitle = 'A normal and healthy usage of the internet is seen with the␣
 ↪occassional spike here and there. '

# add title + subtitle to plot
```

```
plt.text(
    x = 0.125,y = 0.90,s = title, fontname="Open Sans",
    fontsize = 24,ha='left',transform = fig.transFigure
)

plt.text(
    x = 0.125,y = 0.86,s = subtitle, fontname="Open Sans",
    fontsize = 18,ha = 'left',transform = fig.transFigure
)

# line between titles and chart
plt.gca().plot(
    [0.125, .9], # x co-ords
    [.80, .80], # y co-ords
    transform = fig.transFigure,
    clip_on = False,
    color = 'k',
    linewidth = 1.5
)

# plotting as a time series
plt.plot_date(dates.index, dates.values, color='purple', marker='o',␣
 ↪linestyle='dashed', linewidth=1, markersize=5)

# also put labels on the markers a little over the markers for visibility
for i in range(len(dates)):
    plt.text(dates.index[i], dates.values[i]-3, dates.values[i], ha='center',␣
 ↪va='center', color='black', fontsize=16)

# set the size of the tick labels and axis labels
ax.tick_params(axis='both', which='major', labelsize=16)
plt.xlabel('Date', fontsize=20, fontname="Open Sans")
plt.ylabel('Number of requests', fontsize=20, fontname="Open Sans")

# change space on top of chart we are actually adjusting the scale of the plot␣
 ↪as well.
plt.subplots_adjust(top=0.8, wspace=0.3)

# set the font size of the tick labels
for tick in ax.xaxis.get_major_ticks():
    tick.label1.set_fontsize(16)
for tick in ax.yaxis.get_major_ticks():
    tick.label1.set_fontsize(16)

# tilt the x-axis labels by 45 degrees
for tick in ax.get_xticklabels():
    tick.set_rotation(45)
```

```
# grid lines
# keep only toned down vertical lines
plt.grid(axis = 'y',alpha = 0.4)
# plt.grid(axis='x', alpha=0.2)

# turn off spines
plt.gca().spines[['left','right', 'top']].set_visible(False)

# customize the tick labels
ax.xaxis.set_major_locator(mdates.DayLocator())
ax.xaxis.set_major_formatter(mdates.DateFormatter('%d %A'))

plt.show()
```

2.2 Let us now plot the number of requests per device

[ ]:
```
# let us now plot the number of requests per device

devices = normal_log_db['Device Name'].value_counts()

# sorting devices in descending order
devices = devices.sort_values(ascending=False)

# plotting

# creating the plot.
fig, ax = plt.subplots(figsize=(20, 8))

# informative title + subtitle
title = 'The Number of Requests Made per Device by the Household'
subtitle = 'Some devices use the internet more than others. This is normal, as␣
 ↪the Range is not too high.'

# add title + subtitle to plot
plt.text(
    x = 0.125,y = 0.90,s = title, fontname="Open Sans",
    fontsize = 24,ha='left',transform = fig.transFigure
)
plt.text(
    x = 0.125,y = 0.86,s = subtitle, fontname="Open Sans",
    fontsize = 18,ha = 'left',transform = fig.transFigure
)

# line between titles and chart
plt.gca().plot(
    [0.125, .9], # x co-ords
```

```
    [.80, .80], # y co-ords
    transform = fig.transFigure,
    clip_on = False,
    color = 'k',
    linewidth = 1.5
)

# changing space
plt.subplots_adjust(top=0.8, wspace=0.3)

# grid lines
# keep only toned down vertical lines
plt.grid(axis = 'y',alpha = 0.3)
# plt.grid(axis='x', alpha=0.2)

# turn off spines
plt.gca().spines[['left','right', 'top']].set_visible(False)


# set the size of the tick labels and axis labels
ax.tick_params(axis='both', which='major', labelsize=16)
plt.xlabel('Total Requests Made over timespan', fontsize=20, fontname="Open␣
 ↪Sans")
plt.ylabel('Device Make', fontsize=20, fontname="Open Sans")


# set the font size of the tick labels
for tick in ax.xaxis.get_major_ticks():
    tick.label1.set_fontsize(16)
for tick in ax.yaxis.get_major_ticks():
    tick.label1.set_fontsize(16)

# foot note
footnote = "Source: Ficticious Data, Krishnaraj T"
plt.text(
    x = 0.11,
    y = 0.02,
    s = footnote,
    fontname = 'Open Sans',
    fontstyle = 'italic',
    fontsize = 12,
    ha = 'left',
    transform = fig.transFigure
)

plt.barh(devices.index, devices.values, color='blue', alpha=0.5)
```

2.3 Plotting Requests per interface.

[ ]:
```
# assuming you have a DataFrame called `normal_log_db` with a column called␣
↪'Interface'
interface_counts = normal_log_db['Interface'].value_counts()

# creating the plot.
fig, ax = plt.subplots(figsize=(10, 10))

# informative title + subtitle
title = 'Distribution of Requests by Interface'
subtitle = 'It is normal to see an equal distribution of requests across␣
↪interfaces.'

# add title + subtitle to plot
plt.text(
    x = 0.125,y = 0.90,s = title, fontname="Open Sans",
    fontsize = 24,ha='left',transform = fig.transFigure
)
plt.text(
    x = 0.125,y = 0.86,s = subtitle, fontname="Open Sans",
    fontsize = 18,ha = 'left',transform = fig.transFigure
)

# line between titles and chart
plt.gca().plot(
    [0.125, .9], # x co-ords
    [.80, .80], # y co-ords
    transform = fig.transFigure,
    clip_on = False,
    color = 'k',
    linewidth = 1.5
)

# changing space
plt.subplots_adjust(top=0.8, wspace=0.3)

# set the font size of the labels
plt.rcParams['font.size'] = 16

# Create a pie chart to show the distribution of requests per interface
plt.pie(interface_counts.values, labels=interface_counts.index, autopct='%1.
↪1f%%', startangle=90)


# foot note
footnote = "Source: Ficticious Data, Krishnaraj T"
```

```
plt.text(
    x = 0.11,
    y = 0.02,
    s = footnote,
    fontname = 'Open Sans',
    fontstyle = 'italic',
    fontsize = 12,
    ha = 'left',
    transform = fig.transFigure
)

# display the plot
plt.show()
```

2.4 Websites visited by the Household

[ ]:
```
import matplotlib.pyplot as plt

# assuming you have a DataFrame called `normal_log_db` with columns called␣
↪'Requested IP' and 'Requested Website'
destination_ips = normal_log_db['Requested IP'].value_counts()
destination_ips = destination_ips.sort_values(ascending=False)

destination_websites = normal_log_db['Requested Website'].value_counts()
destination_websites = destination_websites.sort_values(ascending=False)

destination_ips = destination_ips.iloc[::-1]
destination_websites = destination_websites.iloc[::-1]

# creating the plot.
fig, ax = plt.subplots(figsize=(20, 8))

# informative title + subtitle
title = 'Websites visited by the Household'
subtitle = 'Some devices connect to more websites than others. This is normal,␣
↪as internet usage is subjective to users. '

# add title + subtitle to plot
plt.text(
    x = 0.125,y = 0.90,s = title, fontname="Open Sans",
    fontsize = 24,ha='left',transform = fig.transFigure
)
plt.text(
    x = 0.125,y = 0.86,s = subtitle, fontname="Open Sans",
    fontsize = 18,ha = 'left',transform = fig.transFigure
)
```

```
# line between titles and chart
plt.gca().plot(
    [0.125, .9], # x co-ords
    [.80, .80], # y co-ords
    transform = fig.transFigure,
    clip_on = False,
    color = 'k',
    linewidth = 1.5
)

# changing space
plt.subplots_adjust(top=0.8, wspace=0.3)

# grid lines
# keep only toned down vertical lines
plt.grid(axis = 'y',alpha = 0.3)
# plt.grid(axis='x', alpha=0.2)

# turn off spines
plt.gca().spines[['left','right', 'top']].set_visible(False)


# set the size of the tick labels and axis labels
ax.tick_params(axis='both', which='major', labelsize=16)
plt.xlabel('Total Requests Made over timespan', fontsize=20, fontname="Open␣
 ↪Sans", labelpad=20)
plt.ylabel('Website', fontsize=20, fontname="Open Sans", labelpad=20)


# set the font size of the tick labels
for tick in ax.xaxis.get_major_ticks():
    tick.label1.set_fontsize(16)
for tick in ax.yaxis.get_major_ticks():
    tick.label1.set_fontsize(16)

# set the y-axis tick labels to the website names
plt.yticks(range(len(destination_websites)), destination_websites.index)

# invert the y-axis so that the website names are displayed from top to bottom
# plt.gca().invert_yaxis()

# add the names of the websites inside their individual bars
for i, v in enumerate(destination_ips.index):
    plt.text(x=destination_ips.values[i] / 3, y=i, s=v, color='black',␣
 ↪fontweight='bold', fontsize=14)
```

```
# foot note
footnote = "Source: Ficticious Data, Krishnaraj T"
plt.text(
    x = 0.11,
    y = 0.02,
    s = footnote,
    fontname = 'Open Sans',
    fontstyle = 'italic',
    fontsize = 12,
    ha = 'left',
    transform = fig.transFigure
)

plt.barh(range(len(destination_websites)), destination_websites.values,␣
↪color='green', alpha=0.5)


plt.show()
```

3 Plotting Protocols

[ ]:
```
# let us now plot what protocols were used to make requests
# count the number of requests for each protocol

protocol_counts = normal_log_db['Protocol'].value_counts()

# set the color of the first rectangle to pink and the color of the other␣
↪rectangles to gray
colors = ['magenta'] * 3 + ['gray'] * (len(protocol_counts) - 3)

# creating the plot.
fig, ax = plt.subplots(figsize=(20, 8))

# informative title + subtitle
title = 'Protocols Used in Router Requests. '
subtitle = 'The most commonly used protocols are shown, and their distribution␣
↪looks normal.'

# add title + subtitle to plot
plt.text(
    x = 0.125,y = 0.90,s = title, fontname="Open Sans",
    fontsize = 24,ha='left',transform = fig.transFigure
)
plt.text(
    x = 0.125,y = 0.86,s = subtitle, fontname="Open Sans",
    fontsize = 18,ha = 'left',transform = fig.transFigure
```

```
)

# line between titles and chart
plt.gca().plot(
    [0.125, .9], # x co-ords
    [.80, .80], # y co-ords
    transform = fig.transFigure,
    clip_on = False,
    color = 'k',
    linewidth = 1.5
)

# changing space
plt.subplots_adjust(top=0.8, wspace=0.3)

# grid lines
# keep only toned down vertical lines
plt.grid(axis = 'y',alpha = 0.5)
# plt.grid(axis='x', alpha=0.5)

# turn off spines
plt.gca().spines[['left','right', 'top']].set_visible(False)

# set the size of the tick labels and axis labels
ax.tick_params(axis='both', which='major', labelsize=16)
plt.xlabel('Total Requests Made over timespan', fontsize=20, fontname="Open␣
 ↪Sans", labelpad=20)
plt.ylabel('Website', fontsize=20, fontname="Open Sans", labelpad=20)


# set the font size of the tick labels
for tick in ax.xaxis.get_major_ticks():
    tick.label1.set_fontsize(16)
for tick in ax.yaxis.get_major_ticks():
    tick.label1.set_fontsize(16)

plt.bar(protocol_counts.index, protocol_counts.values, color=colors, alpha=0.3)

# set the title and axis labels
# ax.set_title('Protocols Used to Make Requests')
ax.set_xlabel('Protocol')
ax.set_ylabel('Number of Requests')

# set the x-axis tick labels to be rotated for better readability
# plt.xticks(rotation=45)
```

```
# foot note
footnote = "Source: Ficticious Data, Krishnaraj T"
plt.text(
    x = 0.11,
    y = 0.02,
    s = footnote,
    fontname = 'Open Sans',
    fontstyle = 'italic',
    fontsize = 12,
    ha = 'left',
    transform = fig.transFigure
)
```

4 Hourly Traffic of the Household

[ ]:
```
temp_df = normal_log_db.copy()
# convert the 'Time' column to a datetime object
temp_df['Time'] = pd.to_datetime(temp_df['Time'])

# set the 'Time' column as the index
temp_df.set_index('Time', inplace=True)

# resample the data by hour and count the number of requests
hourly_counts = temp_df.resample('H').count()['MAC']


# creating the plot.
fig, ax = plt.subplots(figsize=(20, 8))

# informative title + subtitle
title = 'Hourly Traffic Distribution of the Household'
subtitle = 'The household is most active during the day. Almost Zero traffic is␣
 ↪noted between hours of 2am to 5am. This is normal'

# add title + subtitle to plot
plt.text(
    x = 0.125,y = 0.90,s = title, fontname="Open Sans",
    fontsize = 24,ha='left',transform = fig.transFigure
)
plt.text(
    x = 0.125,y = 0.86,s = subtitle, fontname="Open Sans",
    fontsize = 18,ha = 'left',transform = fig.transFigure
)

# line between titles and chart
plt.gca().plot(
```

```
    [0.125, .9], # x co-ords
    [.80, .80], # y co-ords
    transform = fig.transFigure,
    clip_on = False,
    color = 'k',
    linewidth = 1.5
)

# changing space
plt.subplots_adjust(top=0.8, wspace=0.3)

# grid lines
# keep only toned down vertical lines
plt.grid(axis = 'y',alpha = 0.5)
# plt.grid(axis='x', alpha=0.5)

# turn off spines
plt.gca().spines[['left','right', 'top']].set_visible(False)

# set the size of the tick labels and axis labels
ax.tick_params(axis='both', which='major', labelsize=16)
plt.xlabel('Total Requests Made over timespan', fontsize=20, fontname="Open␣
 ↪Sans", labelpad=20)
plt.ylabel('Website', fontsize=20, fontname="Open Sans", labelpad=20)


# set the font size of the tick labels
for tick in ax.xaxis.get_major_ticks():
    tick.label1.set_fontsize(16)
for tick in ax.yaxis.get_major_ticks():
    tick.label1.set_fontsize(16)

ax.set_xlabel('Days')
ax.set_ylabel('Number of Requests')

# set the x-axis tick labels to be rotated for better readability
# plt.xticks(rotation=45)

plt.plot(hourly_counts.index, hourly_counts.values, color='darkblue')

# foot note
footnote = "Source: Ficticious Data, Krishnaraj T"
plt.text(
    x = 0.11,
    y = 0.02,
    s = footnote,
    fontname = 'Open Sans',
```

```
    fontstyle = 'italic',
    fontsize = 12,
    ha = 'left',
    transform = fig.transFigure
)

# customize the tick labels
ax.xaxis.set_major_locator(mdates.DayLocator())
ax.xaxis.set_major_formatter(mdates.DateFormatter('%d %b'))
```

4.1 Plotting Ports

[ ]:
```
port_counts = normal_log_db['Port'].value_counts()

# change indices from float to int
port_counts.index = port_counts.index.astype(int)
port_counts.index = port_counts.index.astype(str)

port_counts.index

# set the color of the first rectangle to pink and the color of the other␣
 ↪rectangles to gray
colors = ['magenta'] * 3 + ['gray'] * (len(port_counts) - 3)

# creating the plot.
fig, ax = plt.subplots(figsize=(20, 8))

# informative title + subtitle
title = 'Ports Appearing in Requests'
subtitle = 'The most commonly used ports are shown, and their distribution looks␣
 ↪normal.'

# add title + subtitle to plot
plt.text(
    x=0.125, y=0.90, s=title, fontname="Open Sans",
    fontsize=24, ha='left', transform=fig.transFigure
)
plt.text(
    x=0.125, y=0.86, s=subtitle, fontname="Open Sans",
    fontsize=18, ha='left', transform=fig.transFigure
)

# line between titles and chart
plt.gca().plot(
    [0.125, .9],  # x co-ords
    [.80, .80],  # y co-ords
    transform=fig.transFigure,
```

```
    clip_on=False,
    color='k',
    linewidth=1.5
)

# changing space
plt.subplots_adjust(top=0.8, wspace=0.3)

# grid lines
# keep only toned down vertical lines
plt.grid(axis='y', alpha=0.5)
# plt.grid(axis='x', alpha=0.5)

# turn off spines
plt.gca().spines[['left', 'right', 'top']].set_visible(False)

# set the size of the tick labels and axis labels
ax.tick_params(axis='both', which='major', labelsize=16)
plt.xlabel('Total Requests Made over timespan', fontsize=20, fontname="Open␣
 ↪Sans", labelpad=20)
plt.ylabel('Website', fontsize=20, fontname="Open Sans", labelpad=20)

# set the font size of the tick labels
for tick in ax.xaxis.get_major_ticks():
    tick.label1.set_fontsize(16)
for tick in ax.yaxis.get_major_ticks():
    tick.label1.set_fontsize(16)

plt.bar(port_counts.index, port_counts.values, color=colors, alpha=0.3)

# set the title and axis labels
# ax.set_title('Protocols Used to Make Requests')
ax.set_xlabel('Protocol')
ax.set_ylabel('Number of Requests')

# set the x-axis tick labels to be rotated for better readability
# plt.xticks(rotation=45)


# footnote
footnote = "Source: Ficticious Data, Krishnaraj T"
plt.text(
    x=0.11,
    y=0.02,
    s=footnote,
    fontname='Open Sans',
    fontstyle='italic',
```

```
    fontsize=12,
    ha='left',
    transform=fig.transFigure
)
```

5 Let us now simulate some attacks

5.1 DOS Attack

[37]:
```
# Generate ddos attack data, consider a home environment. with 10 users. across␣
↪a span of 10 days. Visiting 100 websites per device per day.

ddos_log_db = pd.DataFrame(columns=['MAC', 'IP Address', 'Device Name',␣
↪'Interface', 'Requested IP', 'Time'])

for i in range(10):


    # check if time columns is on 4th jan
    if i == 4:
        temp_df = pd.DataFrame({
            'MAC' : [generate_mac_address() for j in range(100)],
            'IP Address': [generate_attacker_ip_address() for j in range(100)],
            'Device Name': [generate_device_name() if j > 50 else 'Vivo' for j␣
↪in range(100)],
            'Interface': [generate_interface() for j in range(100)],
            'Requested IP': [generate_dest_ip_address()[0] for j in range(100)],
            'Requested Website': [generate_dest_ip_address()[1] for j in␣
↪range(100)],
            'Protocol': [gen_attacker_protocols()[0] for j in range(100)],
            'Port': [gen_attacker_protocols()[1] for j in range(100)],
            'Time': [generate_attacker_date_time() if j < 50 else␣
↪generate_date_time() for j in range(100)]
        })

    else:
        temp_df = pd.DataFrame({
            'MAC' : [generate_mac_address() for j in range(100)],
            'IP Address': [generate_device_ip_address() for j in range(100)],
            'Device Name': [generate_device_name() for j in range(100)],
            'Interface': [generate_interface() for j in range(100)],
            'Requested IP': [generate_dest_ip_address()[0] for j in range(100)],
            'Requested Website': [generate_dest_ip_address()[1] for j in␣
↪range(100)],
            'Protocol': [gen_protocols()[0] for j in range(100)],
            'Port': [gen_protocols()[1]  for j in range(100)],
```

```
            'Time': [generate_date_time() for j in range(100)]
    })

    ddos_log_db = pd.concat([ddos_log_db, temp_df], ignore_index=True)

ddos_log_db
```

[37]:

```
                   MAC     IP Address Device Name Interface   Requested IP  \
0    8f:03:95:ed:b2:fa   192.168.1.60     OnePlus     2.4gz  216.58.194.45
1    0b:ea:5d:f7:3b:d4   192.168.1.30      Xiaomi       5gz   69.63.176.22
2    b4:e4:6c:fc:2e:89   192.168.1.50        Vivo       5gz    3.213.31.34
3    e9:6d:f4:7f:26:84   192.168.1.70     OnePlus       5gz  216.58.194.45
4    a5:e0:0e:07:30:df   192.168.1.30       Nokia       5gz    3.213.31.34
..                 ...           ...         ...       ...            ...
995  62:00:c7:be:a4:4e   192.168.1.50     OnePlus     2.4gz    3.213.31.34
996  fa:44:4c:6e:a3:2b   192.168.1.50    Micromax     2.4gz   192.168.1.53
997  3c:65:92:8c:3a:87   192.168.1.40    Micromax       5gz    3.213.31.34
998  39:41:8d:f9:6f:85  192.168.1.100     Samsung       5gz    3.213.31.34
999  87:4d:8a:6c:5f:f5   192.168.1.10      Xiaomi     2.4gz  216.58.194.45

                    Time Requested Website Protocol  Port
0    2023-01-09 16:16:10         Instagram      TCP  53.0
1    2023-01-05 12:51:11         Instagram      UDP  21.0
2    2023-01-01 07:31:49           Youtube      UDP  80.0
3    2023-01-03 10:55:30             Other    HTTPS  25.0
4    2023-01-04 16:08:25          Facebook      UDP  80.0
..                   ...               ...      ...   ...
995  2023-01-07 23:59:27          Facebook      UDP  67.0
996  2023-01-01 06:40:22           Twitter      TCP  21.0
997  2023-01-04 12:40:32          Facebook    HTTPS  21.0
998  2023-01-10 16:33:16           Twitter      UDP  53.0
999  2023-01-04 07:38:09          Facebook     DHCP  80.0

[1000 rows x 9 columns]
```

5.2 Hourly Traffic Distribution of the Household - DDoS Attack Demo

[ ]:

```
temp_df = ddos_log_db.copy()
# convert the 'Time' column to a datetime object
temp_df['Time'] = pd.to_datetime(temp_df['Time'])

# set the 'Time' column as the index
temp_df.set_index('Time', inplace=True)

# resample the data by hour and count the number of requests
hourly_counts = temp_df.resample('H').count()['MAC']
```

```
# creating the plot.
fig, ax = plt.subplots(figsize=(20, 8))

# informative title + subtitle
title = 'Hourly Traffic Distribution of the Household - DDoS Attack Demo'
subtitle = 'The extreme spike on Wednesday night is clearly visible as a sign of␣
 ↪a DDoS attack'

# add title + subtitle to plot
plt.text(
    x = 0.125,y = 0.90,s = title, fontname="Open Sans",
    fontsize = 24,ha='left',transform = fig.transFigure
)
plt.text(
    x = 0.125,y = 0.86,s = subtitle, fontname="Open Sans",
    fontsize = 18,ha = 'left',transform = fig.transFigure
)

# line between titles and chart
plt.gca().plot(
    [0.125, .9], # x co-ords
    [.80, .80], # y co-ords
    transform = fig.transFigure,
    clip_on = False,
    color = 'k',
    linewidth = 1.5
)

# changing space
plt.subplots_adjust(top=0.8, wspace=0.3)

# grid lines
# keep only toned down vertical lines
plt.grid(axis = 'y',alpha = 0.5)
# plt.grid(axis='x', alpha=0.5)

# turn off spines
plt.gca().spines[['left','right', 'top']].set_visible(False)

# set the size of the tick labels and axis labels
ax.tick_params(axis='both', which='major', labelsize=16)
plt.xlabel('Total Requests Made over timespan', fontsize=20, fontname="Open␣
 ↪Sans", labelpad=20)
plt.ylabel('Website', fontsize=20, fontname="Open Sans", labelpad=20)
```

```
# set the font size of the tick labels
for tick in ax.xaxis.get_major_ticks():
    tick.label1.set_fontsize(16)
for tick in ax.yaxis.get_major_ticks():
    tick.label1.set_fontsize(16)

# ax.set_title('Protocols Used to Make Requests')
ax.set_xlabel('Protocol')
ax.set_ylabel('Number of Requests')

# set the x-axis tick labels to be rotated for better readability
# plt.xticks(rotation=45)

plt.plot(hourly_counts.index, hourly_counts.values, color='darkblue')

# foot note
footnote = "Source: Ficticious Data, Krishnaraj T"
plt.text(
    x = 0.11,
    y = 0.02,
    s = footnote,
    fontname = 'Open Sans',
    fontstyle = 'italic',
    fontsize = 12,
    ha = 'left',
    transform = fig.transFigure
)

# customize the tick labels
ax.xaxis.set_major_locator(mdates.DayLocator())
ax.xaxis.set_major_formatter(mdates.DateFormatter('%d %b'))
```

5.3 Protocols used by the Household - DDoS Attack Demo

[ ]:
```
# let us now plot what protocols were used to make requests
# count the number of requests for each protocol

protocol_counts = ddos_log_db['Protocol'].value_counts()

# set the color of the first rectangle to pink and the color of the other␣
 ↪rectangles to gray
colors = ['magenta'] * 4 + ['gray'] * (len(protocol_counts) - 4)

# creating the plot.
fig, ax = plt.subplots(figsize=(20, 8))

# informative title + subtitle
```

```
title = 'Protocols Used to Make Requests - DDoS Attack Demonstration'
subtitle = 'Protocos most commonly used for DDOS attacks are clearly visible.␣
 ↪ICMP Rises considerably. '

# add title + subtitle to plot
plt.text(
    x = 0.125,y = 0.90,s = title, fontname="Open Sans",
    fontsize = 24,ha='left',transform = fig.transFigure
)
plt.text(
    x = 0.125,y = 0.86,s = subtitle, fontname="Open Sans",
    fontsize = 18,ha = 'left',transform = fig.transFigure
)

# line between titles and chart
plt.gca().plot(
    [0.125, .9], # x co-ords
    [.80, .80], # y co-ords
    transform = fig.transFigure,
    clip_on = False,
    color = 'k',
    linewidth = 1.5
)

# changing space
plt.subplots_adjust(top=0.8, wspace=0.3)

# grid lines
# keep only toned down vertical lines
plt.grid(axis = 'y',alpha = 0.5)
# plt.grid(axis='x', alpha=0.5)

# turn off spines
plt.gca().spines[['left','right', 'top']].set_visible(False)

# set the size of the tick labels and axis labels
ax.tick_params(axis='both', which='major', labelsize=16)
plt.xlabel('Total Requests Made over timespan', fontsize=20, fontname="Open␣
 ↪Sans", labelpad=20)
plt.ylabel('Website', fontsize=20, fontname="Open Sans", labelpad=20)


# set the font size of the tick labels
for tick in ax.xaxis.get_major_ticks():
    tick.label1.set_fontsize(16)
for tick in ax.yaxis.get_major_ticks():
    tick.label1.set_fontsize(16)
```

```
plt.bar(protocol_counts.index, protocol_counts.values, color=colors, alpha=0.3)

# set the title and axis labels
# ax.set_title('Protocols Used to Make Requests')
ax.set_xlabel('Protocol')
ax.set_ylabel('Number of Requests')

# set the x-axis tick labels to be rotated for better readability
# plt.xticks(rotation=45)


# foot note
footnote = "Source: Ficticious Data, Krishnaraj T"
plt.text(
    x = 0.11,
    y = 0.02,
    s = footnote,
    fontname = 'Open Sans',
    fontstyle = 'italic',
    fontsize = 12,
    ha = 'left',
    transform = fig.transFigure
)
```

5.4 Daily Traffic Distribution of the Household - DDoS Attack Demo

[ ]:
```
# let us plot the number of requests per day
ddos_log_db['Time'] = pd.to_datetime(ddos_log_db['Time'])
ddos_log_db['Date'] = ddos_log_db['Time'].dt.date

# sort data by date
ddos_log_db = ddos_log_db.sort_values(by=['Date'])

ddos_log_db.head()

# now let us plot the number of requests per day
dates = ddos_log_db['Date'].value_counts()

# sorting dates
dates = dates.sort_index()

# creating the plot.
fig, ax = plt.subplots(figsize=(20, 8))

# informative title + subtitle
```

```
title = 'The Number of Requests Made per day by the Household - DDOS Attack␣
 ↪Demonstration'
subtitle = 'The DDOS Attack on wednesday night causing high requests is clearly␣
 ↪visible. '

# add title + subtitle to plot
plt.text(
    x = 0.125,y = 0.90,s = title, fontname="Open Sans",
    fontsize = 24,ha='left',transform = fig.transFigure
)

plt.text(
    x = 0.125,y = 0.86,s = subtitle, fontname="Open Sans",
    fontsize = 18,ha = 'left',transform = fig.transFigure
)

# line between titles and chart
plt.gca().plot(
    [0.125, .9], # x co-ords
    [.80, .80], # y co-ords
    transform = fig.transFigure,
    clip_on = False,
    color = 'k',
    linewidth = 1.5
)

# plotting as a time series
plt.plot_date(dates.index, dates.values, color='purple', marker='o',␣
 ↪linestyle='dashed', linewidth=1, markersize=5)

# also put labels on the markers a little over the markers for visibility
for i in range(len(dates)):
    plt.text(dates.index[i], dates.values[i]-6, dates.values[i], ha='center',␣
 ↪va='center', color='black', fontsize=16)

# set the size of the tick labels and axis labels
ax.tick_params(axis='both', which='major', labelsize=16)
plt.xlabel('Date', fontsize=20, fontname="Open Sans")
plt.ylabel('Number of requests', fontsize=20, fontname="Open Sans")

# change space on top of chart we are actually adjusting the scale of the plot␣
 ↪as well.
plt.subplots_adjust(top=0.8, wspace=0.3)

# set the font size of the tick labels
for tick in ax.xaxis.get_major_ticks():
    tick.label1.set_fontsize(16)
```

```
for tick in ax.yaxis.get_major_ticks():
    tick.label1.set_fontsize(16)

# tilt the x-axis labels by 45 degrees
for tick in ax.get_xticklabels():
    tick.set_rotation(45)

# grid lines
# keep only toned down vertical lines
plt.grid(axis = 'y',alpha = 0.4)
# plt.grid(axis='x', alpha=0.2)

# turn off spines
plt.gca().spines[['left','right', 'top']].set_visible(False)

# customize the tick labels
ax.xaxis.set_major_locator(mdates.DayLocator())
ax.xaxis.set_major_formatter(mdates.DateFormatter('%d %A'))

plt.show()
```

5.5 let us now plot the number of requests per device IP

[ ]:
```
devices = ddos_log_db['IP Address'].value_counts()
devices

# sorting devices in descending order
devices = devices.sort_values(ascending=False)

# plotting

# creating the plot.
fig, ax = plt.subplots(figsize=(20, 8))

# informative title + subtitle
title = 'The Number of Requests Made per Device by the Household - DDOS Attack␣
 ↪Demonstration'
subtitle = 'IP address 192.168.1.20 is the attacker with most requests. '

# add title + subtitle to plot
plt.text(
    x = 0.125,y = 0.90,s = title, fontname="Open Sans",
    fontsize = 24,ha='left',transform = fig.transFigure
)
plt.text(
    x = 0.125,y = 0.86,s = subtitle, fontname="Open Sans",
    fontsize = 18,ha = 'left',transform = fig.transFigure
```

```
)

# line between titles and chart
plt.gca().plot(
    [0.125, .9], # x co-ords
    [.80, .80], # y co-ords
    transform = fig.transFigure,
    clip_on = False,
    color = 'k',
    linewidth = 1.5
)

# changing space
plt.subplots_adjust(top=0.8, wspace=0.3)

# grid lines
# keep only toned down vertical lines
plt.grid(axis = 'y',alpha = 0.3)
# plt.grid(axis='x', alpha=0.2)

# turn off spines
plt.gca().spines[['left','right', 'top']].set_visible(False)


# set the size of the tick labels and axis labels
ax.tick_params(axis='both', which='major', labelsize=16)
plt.xlabel('Total Requests Made over timespan', fontsize=20, fontname="Open␣
 ↪Sans")
plt.ylabel('Device IP', fontsize=20, fontname="Open Sans")


# set the font size of the tick labels
for tick in ax.xaxis.get_major_ticks():
    tick.label1.set_fontsize(16)
for tick in ax.yaxis.get_major_ticks():
    tick.label1.set_fontsize(16)

# foot note
footnote = "Source: Ficticious Data, Krishnaraj T"
plt.text(
    x = 0.11,
    y = 0.02,
    s = footnote,
    fontname = 'Open Sans',
    fontstyle = 'italic',
    fontsize = 12,
    ha = 'left',
```

```
    transform = fig.transFigure
)

plt.barh(devices.index, devices.values, color='blue', alpha=0.5)
```

5.6 Instagram Account Brute Force Attack

[38]:
```
# Generate insta brute force attack data, consider a home environment. with 10␣
↪users. across a span of 10 days. Visiting 100 websites per device per day.

insta_brute_force_db = pd.DataFrame(columns=['MAC', 'IP Address', 'Device Name',␣
↪'Interface', 'Requested IP', 'Time'])

for i in range(10):
    # check if time columns is on 4th jan
    if i == 7:
        temp_df = pd.DataFrame({
            'MAC' : [generate_attacker_mac_address() for j in range(100)],
            'IP Address': [generate_attacker_ip_address() for j in range(100)],
            'Device Name': [generate_device_name() if j > 50 else 'Vivo' for j␣
↪in range(100)],
            'Interface': [generate_interface() for j in range(100)],
            'Requested IP': [generate_dest_ip_address()[0] for j in range(100)],
            'Requested Website': [generate_attacker_dest_ip_address()[1] for j␣
↪in range(100)],
            'Protocol': [gen_attacker_protocols() if j < 50 else gen_protocols()␣
↪for j in range(100)],
            'Time': [generate_attacker_date_time() if j < 50 else␣
↪generate_date_time() for j in range(100)]
        })

    else:
        temp_df = pd.DataFrame({
            'MAC' : [generate_mac_address() for j in range(100)],
            'IP Address': [generate_device_ip_address() for j in range(100)],
            'Device Name': [generate_device_name() for j in range(100)],
            'Interface': [generate_interface() for j in range(100)],
            'Requested IP': [generate_dest_ip_address()[0] for j in range(100)],
            'Requested Website': [generate_dest_ip_address()[1] for j in␣
↪range(100)],
            'Protocol': [gen_attacker_protocols() for j in range(100)],
            'Time': [generate_date_time() for j in range(100)]
    })

    insta_brute_force_db = pd.concat([insta_brute_force_db, temp_df],␣
↪ignore_index=True)
```

```
insta_brute_force_db
```

[38]:

	MAC	IP Address	Device Name	Interface	Requested IP	\
0	e8:e4:0c:11:92:26	192.168.1.10	iPhone	5gz	3.213.31.34	
1	7b:92:19:a9:c4:e2	192.168.1.20	Samsung	2.4gz	216.58.194.45	
2	8e:21:dc:bb:28:c4	192.168.1.20	iPhone	5gz	69.63.176.22	
3	a2:d9:ae:9c:32:ce	192.168.1.10	Xiaomi	5gz	69.63.176.22	
4	a1:7d:75:ab:a4:1a	192.168.1.20	Vivo	5gz	216.58.194.45	
..	...	...	...	...	...	
995	ff:8d:0e:2d:b9:f9	192.168.1.10	Lenovo	2.4gz	3.213.31.34	
996	c2:e4:76:2d:5c:c3	192.168.1.100	Vivo	5gz	3.213.31.34	
997	d3:7b:0a:00:e6:35	192.168.1.20	Oppo	5gz	69.63.176.22	
998	be:28:65:6c:ae:d0	192.168.1.100	Micromax	2.4gz	216.58.194.45	
999	f7:b3:5b:0a:98:e6	192.168.1.50	Xiaomi	5gz	69.63.176.22	

	Time	Requested Website	Protocol
0	2023-01-08 20:00:03	Twitter	(FTP, 21)
1	2023-01-11 15:05:35	Facebook	(TCP, 21)
2	2023-01-11 22:12:24	Facebook	(FTP, 21)
3	2023-01-10 20:22:50	Facebook	(HTTPS, 443)
4	2023-01-01 13:02:45	Twitter	(HTTPS, 443)
..	...	...	...
995	2023-01-06 15:39:16	Facebook	(IMAP, 143)
996	2023-01-06 23:32:12	Instagram	(TCP, 21)
997	2023-01-02 12:05:54	Youtube	(HTTPS, 443)
998	2023-01-11 16:49:49	Facebook	(TCP, 21)
999	2023-01-09 09:33:55	Youtube	(DHCP, 67)

[1000 rows x 8 columns]

5.7 Hourly Traffic Distribution of the Household - Insta Brute Force Attack

[ ]:
```
import matplotlib.pyplot as plt
import pandas as pd

temp_df = insta_brute_force_db.copy()
# convert the 'Time' column to a datetime object
temp_df['Time'] = pd.to_datetime(temp_df['Time'])

# set the 'Time' column as the index
temp_df.set_index('Time', inplace=True)

# resample the data by hour and count the number of requests
hourly_counts = temp_df.resample('H').count()['MAC']
```

```
# creating the plot.
fig, ax = plt.subplots(figsize=(20, 8))

# informative title + subtitle
title = 'Hourly Traffic Distribution of the Household - Insta Brute Force Attack'
subtitle = 'The extreme spike on Saturday night is clearly visible as a sign of␣
 ↪a Brute Force break in. '

# add title + subtitle to plot
plt.text(
    x = 0.125,y = 0.90,s = title, fontname="Open Sans",
    fontsize = 24,ha='left',transform = fig.transFigure
)
plt.text(
    x = 0.125,y = 0.86,s = subtitle, fontname="Open Sans",
    fontsize = 18,ha = 'left',transform = fig.transFigure
)

# line between titles and chart
plt.gca().plot(
    [0.125, .9], # x co-ords
    [.80, .80], # y co-ords
    transform = fig.transFigure,
    clip_on = False,
    color = 'k',
    linewidth = 1.5
)

# changing space
plt.subplots_adjust(top=0.8, wspace=0.3)

# grid lines
# keep only toned down vertical lines
plt.grid(axis = 'y',alpha = 0.5)
# plt.grid(axis='x', alpha=0.5)

# turn off spines
plt.gca().spines[['left','right', 'top']].set_visible(False)

# set the size of the tick labels and axis labels
ax.tick_params(axis='both', which='major', labelsize=16)
plt.xlabel('Total Requests Made over timespan', fontsize=20, fontname="Open␣
 ↪Sans", labelpad=20)
plt.ylabel('Website', fontsize=20, fontname="Open Sans", labelpad=20)


# set the font size of the tick labels
```

```
for tick in ax.xaxis.get_major_ticks():
    tick.label1.set_fontsize(16)
for tick in ax.yaxis.get_major_ticks():
    tick.label1.set_fontsize(16)

# ax.set_title('Protocols Used to Make Requests')
ax.set_xlabel('Hourly Distribution')
ax.set_ylabel('Number of Requests')

# set the x-axis tick labels to be rotated for better readability
# plt.xticks(rotation=45)

plt.plot(hourly_counts.index, hourly_counts.values, color='darkblue')

# foot note
footnote = "Source: Ficticious Data, Krishnaraj T"
plt.text(
    x = 0.11,
    y = 0.02,
    s = footnote,
    fontname = 'Open Sans',
    fontstyle = 'italic',
    fontsize = 12,
    ha = 'left',
    transform = fig.transFigure
)

# customize the tick labels
ax.xaxis.set_major_locator(mdates.DayLocator())
ax.xaxis.set_major_formatter(mdates.DateFormatter('%d %b'))
```

5.8 Daily Requests

[ ]:
```
# let us plot the number of requests per day
insta_brute_force_db['Time'] = pd.to_datetime(insta_brute_force_db['Time'])
insta_brute_force_db['Date'] = insta_brute_force_db['Time'].dt.date

# sort data by date
insta_brute_force_db = insta_brute_force_db.sort_values(by=['Date'])

insta_brute_force_db.head()

# now let us plot the number of requests per day
dates = insta_brute_force_db['Date'].value_counts()

# sorting dates
dates = dates.sort_index()
```

```
# creating the plot.
fig, ax = plt.subplots(figsize=(20, 8))

# informative title + subtitle
title = 'The Number of Requests Made per day by the Household - Insta Brute␣
 ↪Force Attack'
subtitle = 'The extreme spike on Saturday night is clearly visible as a sign of␣
 ↪a Brute Force break in.'

# add title + subtitle to plot
plt.text(
    x = 0.125,y = 0.90,s = title, fontname="Open Sans",
    fontsize = 24,ha='left',transform = fig.transFigure
)

plt.text(
    x = 0.125,y = 0.86,s = subtitle, fontname="Open Sans",
    fontsize = 18,ha = 'left',transform = fig.transFigure
)

# line between titles and chart
plt.gca().plot(
    [0.125, .9], # x co-ords
    [.80, .80], # y co-ords
    transform = fig.transFigure,
    clip_on = False,
    color = 'k',
    linewidth = 1.5
)

# plotting as a time series
plt.plot_date(dates.index, dates.values, color='purple', marker='o',␣
 ↪linestyle='dashed', linewidth=1, markersize=5)

# also put labels on the markers a little over the markers for visibility
for i in range(len(dates)):
    plt.text(dates.index[i], dates.values[i]-3, dates.values[i], ha='center',␣
 ↪va='center', color='black', fontsize=16)

# set the size of the tick labels and axis labels
ax.tick_params(axis='both', which='major', labelsize=16)
plt.xlabel('Date', fontsize=20, fontname="Open Sans")
plt.ylabel('Number of requests', fontsize=20, fontname="Open Sans")

# change space on top of chart we are actually adjusting the scale of the plot␣
 ↪as well.
```

```
plt.subplots_adjust(top=0.8, wspace=0.3)

# set the font size of the tick labels
for tick in ax.xaxis.get_major_ticks():
    tick.label1.set_fontsize(16)
for tick in ax.yaxis.get_major_ticks():
    tick.label1.set_fontsize(16)

# tilt the x-axis labels by 45 degrees
for tick in ax.get_xticklabels():
    tick.set_rotation(45)

# grid lines
# keep only toned down vertical lines
plt.grid(axis = 'y',alpha = 0.4)
# plt.grid(axis='x', alpha=0.2)

# turn off spines
plt.gca().spines[['left','right', 'top']].set_visible(False)

# customize the tick labels
ax.xaxis.set_major_locator(mdates.DayLocator())
ax.xaxis.set_major_formatter(mdates.DateFormatter('%d %A'))

plt.show()
```

5.9 Plotting Visited Website

[ ]:
```
# assuming you have a DataFrame called `normal_log_db` with columns called␣
↪'Requested IP' and 'Requested Website'

destination_ips = insta_brute_force_db['Requested IP'].value_counts()
destination_ips = destination_ips.sort_values(ascending=False)

destination_websites = insta_brute_force_db['Requested Website'].value_counts()
destination_websites = destination_websites.sort_values(ascending=False)

destination_ips = destination_ips.iloc[::-1]
destination_websites = destination_websites.iloc[::-1]

# creating the plot.
fig, ax = plt.subplots(figsize=(20, 8))

# informative title + subtitle
title = 'Websites visited by the Household - Insta Brute Force Attack'
subtitle = 'Requests made to Instagram server is seen to be the highest,␣
↪indicating heavy use during and post attack. '
```

```
# add title + subtitle to plot
plt.text(
    x = 0.125,y = 0.90,s = title, fontname="Open Sans",
    fontsize = 24,ha='left',transform = fig.transFigure
)
plt.text(
    x = 0.125,y = 0.86,s = subtitle, fontname="Open Sans",
    fontsize = 18,ha = 'left',transform = fig.transFigure
)

# line between titles and chart
plt.gca().plot(
    [0.125, .9], # x co-ords
    [.80, .80], # y co-ords
    transform = fig.transFigure,
    clip_on = False,
    color = 'k',
    linewidth = 1.5
)

# changing space
plt.subplots_adjust(top=0.8, wspace=0.3)

# grid lines
# keep only toned down vertical lines
plt.grid(axis = 'y',alpha = 0.3)
# plt.grid(axis='x', alpha=0.2)

# turn off spines
plt.gca().spines[['left','right', 'top']].set_visible(False)


# set the size of the tick labels and axis labels
ax.tick_params(axis='both', which='major', labelsize=16)
plt.xlabel('Total Requests Made over timespan', fontsize=20, fontname="Open␣
 ↪Sans", labelpad=20)
plt.ylabel('Website', fontsize=20, fontname="Open Sans", labelpad=20)


# set the font size of the tick labels
for tick in ax.xaxis.get_major_ticks():
    tick.label1.set_fontsize(16)
for tick in ax.yaxis.get_major_ticks():
    tick.label1.set_fontsize(16)

# set the y-axis tick labels to the website names
```

```
plt.yticks(range(len(destination_websites)), destination_websites.index)

# invert the y-axis so that the website names are displayed from top to bottom
# plt.gca().invert_yaxis()

# add the names of the websites inside their individual bars
for i, v in enumerate(destination_ips.index):
    plt.text(x=destination_ips.values[i] / 3, y=i, s=v, color='black',␣
 ↪fontweight='bold', fontsize=14)


# foot note
footnote = "Source: Ficticious Data, Krishnaraj T"
plt.text(
    x = 0.11,
    y = 0.02,
    s = footnote,
    fontname = 'Open Sans',
    fontstyle = 'italic',
    fontsize = 12,
    ha = 'left',
    transform = fig.transFigure
)

plt.barh(range(len(destination_websites)), destination_websites.values,␣
 ↪color='green', alpha=0.5)


plt.show()
```

5.10 Port Scanning

This is a surveillance technique that is used to identify open ports on a system. This is used by hackers to identify vulnerable ports on a system.

[39]:
```
# Generate insta brute force attack data, consider a home environment. with 10␣
 ↪users. across a span of 10 days. Visiting 100 websites per device per day.

port_scanning_db = pd.DataFrame(columns=['MAC', 'IP Address', 'Device Name',␣
 ↪'Interface', 'Requested IP', 'Time'])

for i in range(10):
    # check if time columns is on 4th jan
    if i in [3, 4, 5, 6, 7]:
        temp_df = pd.DataFrame({
            'MAC' : [generate_attacker_mac_address() for j in range(100)],
            'IP Address': [generate_attacker_ip_address() for j in range(100)],
```

```
            'Device Name': [generate_device_name() if j > 50 else 'Vivo' for j␣
↪in range(100)],
            'Interface': [generate_interface() for j in range(100)],
            'Requested IP': [generate_dest_ip_address()[0] for j in range(100)],
            'Requested Website': [generate_attacker_dest_ip_address()[1] for j␣
↪in range(100)],
            'Protocol': [gen_attacker_protocols()[0] for j in range(100)],
            'Port': [gen_attacker_protocols()[1]  for j in range(100)],
            'Time': [generate_attacker_date_time() if j < 50 else␣
↪generate_date_time() for j in range(100)]
        })

    else:
        temp_df = pd.DataFrame({
            'MAC' : [generate_mac_address() for j in range(100)],
            'IP Address': [generate_device_ip_address() for j in range(100)],
            'Device Name': [generate_device_name() for j in range(100)],
            'Interface': [generate_interface() for j in range(100)],
            'Requested IP': [generate_dest_ip_address()[0] for j in range(100)],
            'Requested Website': [generate_dest_ip_address()[1] for j in␣
↪range(100)],
            'Protocol': [gen_protocols()[0] for j in range(100)],
            'Port': [gen_protocols()[1] for j in range(100)],
            'Time': [generate_date_time() for j in range(100)]
    })

    port_scanning_db = pd.concat([port_scanning_db, temp_df], ignore_index=True)

port_scanning_db
```

[39]:

	MAC	IP Address	Device Name	Interface	Requested IP	\
0	d0:08:9a:38:b9:4b	192.168.1.30	iPhone	5gz	3.213.31.34	
1	f6:65:bb:3f:87:f2	192.168.1.20	Micromax	5gz	216.58.194.45	
2	6e:46:7f:30:8e:c2	192.168.1.60	iPhone	5gz	3.213.31.34	
3	ec:e3:ce:d6:90:a1	192.168.1.60	Micromax	5gz	192.168.1.53	
4	ca:2b:1a:31:5d:c1	192.168.1.90	iPhone	5gz	3.213.31.34	
..	...	...	...	...	...	
995	40:d1:01:ce:fa:97	192.168.1.90	OnePlus	5gz	3.213.31.34	
996	a8:60:0a:a4:8f:ad	192.168.1.60	Vivo	5gz	69.63.176.22	
997	7a:f4:35:70:51:9a	192.168.1.90	Samsung	2.4gz	3.213.31.34	
998	96:13:ad:eb:d3:b2	192.168.1.20	Nokia	5gz	216.58.194.45	
999	32:d0:50:44:6e:70	192.168.1.40	Oppo	2.4gz	69.63.176.22	

	Time	Requested Website	Protocol	Port
0	2023-01-06 11:16:44	Youtube	HTTP	80.0
1	2023-01-09 17:02:11	Instagram	UDP	110.0
2	2023-01-06 17:10:24	Instagram	TCP	80.0

```
3     2023-01-09 06:04:57          Facebook      DNS  443.0
4     2023-01-02 19:39:35         Instagram      UDP    NaN
..                    ...               ...      ...    ...
995   2023-01-08 10:21:05         Instagram      UDP   53.0
996   2023-01-04 20:25:40          Facebook     IMAP  143.0
997   2023-01-08 10:36:44           Twitter    HTTPS   53.0
998   2023-01-10 09:34:39          Facebook     POP3   80.0
999   2023-01-06 11:20:31           Twitter      TCP   53.0

[1000 rows x 9 columns]
```

5.11 Plotting Ports

[ ]:
```
# let us now plot what protocols were used to make requests
# count the number of requests for each protocol

port_counts = port_scanning_db['Port'].value_counts()

# change indices from float to int
port_counts.index = port_counts.index.astype(int)
port_counts.index = port_counts.index.astype(str)

port_counts.index
```

[ ]:
```
# set the color of the first rectangle to pink and the color of the other␣
↪rectangles to gray
colors = ['blue'] * 3 + ['gray'] * (len(port_counts) - 3)

# creating the plot.
fig, ax = plt.subplots(figsize=(20, 8))

# informative title + subtitle
title = 'Ports Appearing in Requests - Port Scanning'
subtitle ='A general rise in ports commonly vulnerable to attacks is visible.'

# add title + subtitle to plot
plt.text(
    x = 0.125,y = 0.90,s = title, fontname="Open Sans",
    fontsize = 24,ha='left',transform = fig.transFigure
)
plt.text(
    x = 0.125,y = 0.86,s = subtitle, fontname="Open Sans",
    fontsize = 18,ha = 'left',transform = fig.transFigure
)

# line between titles and chart
plt.gca().plot(
```

```
    [0.125, .9], # x co-ords
    [.80, .80], # y co-ords
    transform = fig.transFigure,
    clip_on = False,
    color = 'k',
    linewidth = 1.5
)

# changing space
plt.subplots_adjust(top=0.8, wspace=0.3)

# grid lines
# keep only toned down vertical lines
plt.grid(axis = 'y',alpha = 0.5)
# plt.grid(axis='x', alpha=0.5)

# turn off spines
plt.gca().spines[['left','right', 'top']].set_visible(False)

# set the size of the tick labels and axis labels
ax.tick_params(axis='both', which='major', labelsize=16)
plt.xlabel('Total Requests Made over timespan', fontsize=20, fontname="Open␣
 ↪Sans", labelpad=20)
plt.ylabel('Website', fontsize=20, fontname="Open Sans", labelpad=20)


# set the font size of the tick labels
for tick in ax.xaxis.get_major_ticks():
    tick.label1.set_fontsize(16)
for tick in ax.yaxis.get_major_ticks():
    tick.label1.set_fontsize(16)

plt.bar(port_counts.index, port_counts.values, color=colors, alpha=0.3)

# set the title and axis labels
# ax.set_title('Protocols Used to Make Requests')
ax.set_xlabel('Protocol')
ax.set_ylabel('Number of Requests')

# set the x-axis tick labels to be rotated for better readability
# plt.xticks(rotation=45)


# footnote
footnote = "Source: Ficticious Data, Krishnaraj T"
plt.text(
    x = 0.11,
```

```
    y = 0.02,
    s = footnote,
    fontname = 'Open Sans',
    fontstyle = 'italic',
    fontsize = 12,
    ha = 'left',
    transform = fig.transFigure
)
```

6 Plotting Daily Traffic

[ ]:
```
# let us plot the number of requests per day
port_scanning_db['Time'] = pd.to_datetime(port_scanning_db['Time'])
port_scanning_db['Date'] = port_scanning_db['Time'].dt.date

# sort data by date
port_scanning_db = port_scanning_db.sort_values(by=['Date'])

# now let us plot the number of requests per day
dates = port_scanning_db['Date'].value_counts()

# sorting dates
dates = dates.sort_index()

# creating the plot.
fig, ax = plt.subplots(figsize=(20, 8))

# informative title + subtitle
title = 'The Number of Requests Made per day by the Household'
subtitle = 'A general rise in requests is visible between Wednesday and Sunday,␣
 ↪indicating a possible port scanning attack.'

# add title + subtitle to plot
plt.text(
    x = 0.125,y = 0.90,s = title, fontname="Open Sans",
    fontsize = 24,ha='left',transform = fig.transFigure
)

plt.text(
    x = 0.125,y = 0.86,s = subtitle, fontname="Open Sans",
    fontsize = 18,ha = 'left',transform = fig.transFigure
)

# line between titles and chart
plt.gca().plot(
    [0.125, .9], # x co-ords
```

```
    [.80, .80], # y co-ords
    transform = fig.transFigure,
    clip_on = False,
    color = 'k',
    linewidth = 1.5
)

# plotting as a time series
plt.plot_date(dates.index, dates.values, color='red', marker='o',␣
 ↪linestyle='dashed', linewidth=1, markersize=5)

# also put labels on the markers a little over the markers for visibility
for i in range(len(dates)):
    plt.text(dates.index[i], dates.values[i]-3, dates.values[i], ha='center',␣
 ↪va='center', color='black', fontsize=16)

# set the size of the tick labels and axis labels
ax.tick_params(axis='both', which='major', labelsize=16)
plt.xlabel('Date', fontsize=20, fontname="Open Sans")
plt.ylabel('Number of requests', fontsize=20, fontname="Open Sans")

# change space on top of chart we are actually adjusting the scale of the plot␣
 ↪as well.
plt.subplots_adjust(top=0.8, wspace=0.3)

# set the font size of the tick labels
for tick in ax.xaxis.get_major_ticks():
    tick.label1.set_fontsize(16)
for tick in ax.yaxis.get_major_ticks():
    tick.label1.set_fontsize(16)

# tilt the x-axis labels by 45 degrees
for tick in ax.get_xticklabels():
    tick.set_rotation(45)

# grid lines
# keep only toned down vertical lines
plt.grid(axis = 'y',alpha = 0.4)
# plt.grid(axis='x', alpha=0.2)

# turn off spines
plt.gca().spines[['left','right', 'top']].set_visible(False)

# customize the tick labels
ax.xaxis.set_major_locator(mdates.DayLocator())
ax.xaxis.set_major_formatter(mdates.DateFormatter('%d %A'))
```

```
plt.show()
```