MIT World Peace University

Digital Forensics and Investigation
Third Year B. Tech, Semester 5

Analysing Simlulated Household Router Logs

Lab Assignment 2

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

October 31, 2023

Contents

1 Aim

To simulate different types of attacks on a router, or a home network.

2 Objectives

1. Simulating different types of attacks on a router, or a home network.
2. To learn about the different types of attacks that can be performed on a router, or a home network.
3. To analyze the Router logs, and make inferences about the attacks.

3 Theory

3.1 Router

Definition 1 ***Router*** *is a networking device that operates at the network layer of the OSI model. It functions as a gateway, directing data packets between different computer networks. Routers use routing tables to determine the optimal path for forwarding packets, facilitating efficient communication between devices.*

Definition 2 ***Logs*** *in the context of networking refer to records generated by various network devices, including routers, to capture significant events and activities. Router logs provide a chronological record of network operations, errors, warnings, and security-related events. These logs play a crucial role in network management, troubleshooting, and security analysis. They offer insights into network behavior, potential vulnerabilities, and unauthorized access attempts.*

3.2 Router Logs

We were unable to obtain the logs from our router, so we had to simluate the attacks on our own using a python script, which is given below. The logs were then generated using the script and then analysed.

For our reference, though, we have included a sample log file from a router, which is given below.

```
    1. [2023-08-15 10:12:34] INFO: Router successfully initialized.
    2. [2023-08-15 11:45:21] WARNING: High network traffic detected from IP 192.168.1.15.
    3. [2023-08-15 12:30:05] ERROR: Failed to establish connection with DNS server 8.8.8.8.
    4. [2023-08-16 08:20:10] INFO: New device connected with MAC address 00:1A:2B:3C:4D:5E.
    5. [2023-08-16 09:10:55] INFO: Firmware update successfully applied.
    6. [2023-08-16 09:30:40] ERROR: Unsuccessful login attempt from IP 192.168.1.25.
    7. [2023-08-17 14:05:12] WARNING: DHCP pool depletion. Only 5 IP addresses left.
    8. [2023-08-17 15:20:30] INFO: VPN tunnel established with remote gateway 203.0.113.50.
    9. [2023-08-18 08:45:02] ERROR: Port forwarding request for port 22 already exists.
    10. [2023-08-18 09:55:18] INFO: Quality of Service (QoS) rules updated for improved VoIP performance.
```

```
 11. [2023-08-18 12:15:45] WARNING: Suspicious ARP activity detected from IP 192.168.1.10.
 12. [2023-08-19 07:30:22] ERROR: NAT configuration conflict detected in rule set.
 13. [2023-08-19 10:40:17] INFO: Guest network "GuestWiFi" established with password authentication.
 14. [2023-08-19 14:05:30] INFO: Router temperature exceeds safe threshold. Cooling initiated.
 15. [2023-08-20 09:20:05] INFO: Port 80 forwarded to internal server at IP 192.168.1.50.
 16. [2023-08-20 11:10:48] WARNING: Ping sweep detected from external IP 123.456.789.10.
 17. [2023-08-21 13:25:15] ERROR: DNS cache corruption. Flushing cache for resolution.
 18. [2023-08-21 14:50:29] INFO: Network time synchronization successful with NTP server.
 19. [2023-08-22 10:15:02] INFO: Wireless channel changed to optimize signal quality.
 20. [2023-08-22 12:40:18] WARNING: MAC address spoofing attempt from device with MAC 11:22:33:44:55:66.
```

We have instead chosen to directly make a table with the relevant information. The table and its description is given in the code further below.

3.3 Attacks Simulated

The following attack scenarios were simulated by my script.

1. DOS Attack: A Denial of Service (DOS) Attack is a malicious attempt to disrupt the normal functioning of a network, service, or website by overwhelming it with a flood of traffic. This attack aims to exhaust the target's resources, causing it to become unavailable to legitimate users. DOS attacks can be achieved through various means, such as sending a high volume of requests, exploiting vulnerabilities, or using botnets.

   On the router logs, we may see a high volume of requests from a single IP address, or a high volume of requests to a single IP address. This is what we are looking for.

2. Brute Force Attack to access Instagram: A brute force attack is a trial-and-error method used to obtain information such as a user password or personal identification number (PIN). In a brute force attack, automated software is used to generate a large number of consecutive guesses as to the value of the desired data. Brute force attacks may be used by criminals to crack encrypted data, or by security analysts to test an organization's network security.

   On the router it appears as a failed login attempt, and if the attack is successful, it will appear as a successful login attempt, followed by multiple requests to the Instagram server. This is because the attacker will try to access the account and then try to change the password, which will require multiple requests to the server. He may then try and misuse the account amounting to multiple requests to the server. This is what we are looking for.

3. Port Scanning for Surveillance: Port scanning is a method used to determine which ports on a network are open and which are closed. It is used by hackers to identify vulnerable services

listening on a port that can be exploited for malicious purposes. Port scanning is also used by security analysts to discover vulnerable services and applications that can be exploited.

On the router logs, we may a large number of requests to different ports from a single IP address. This is what we are looking for.

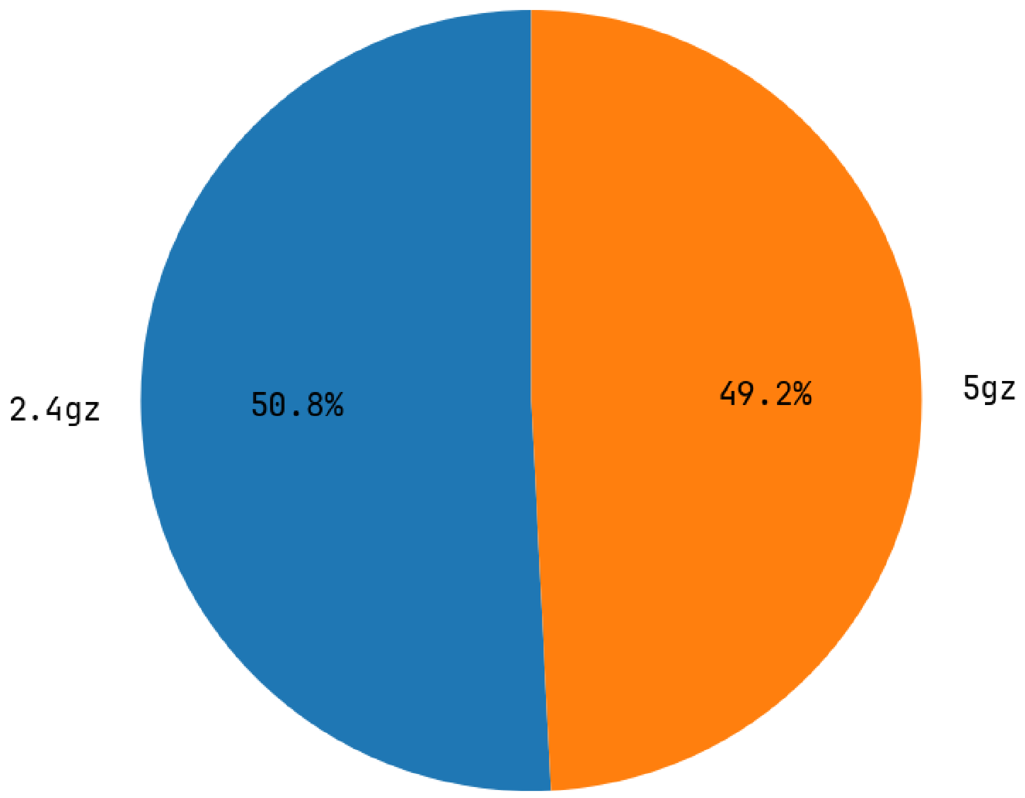
4 Analysis

4.1 Normal Usage Data

4.1.1 Interface Usage - Normal Usage

Distribution of Requests by Interface

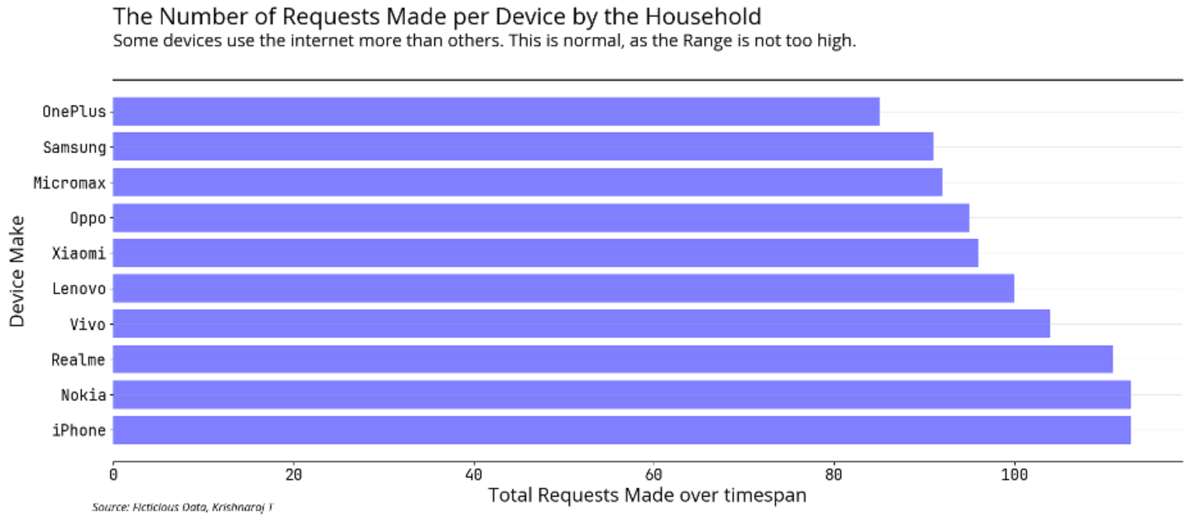
It is normal to see an equal distribution of requests across interfaces.



Source: Ficticious Data, Krishnaraj T

4.2 DOS Attack Data

4.2.1 Devices Connected - Normal Usage

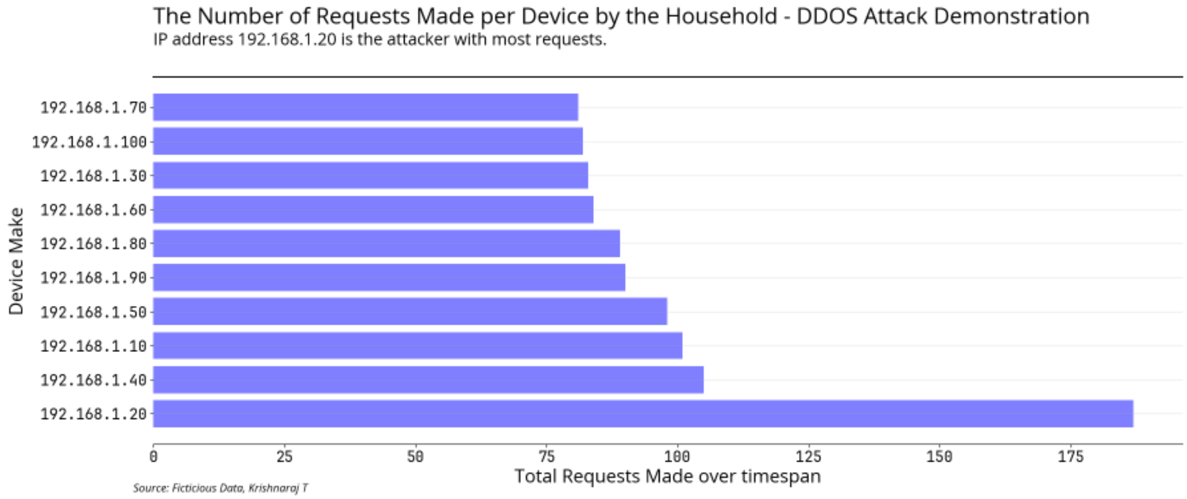


The Number of Requests Made per Device by the Household

Some devices use the internet more than others. This is normal, as the Range is not too high.

Source: Ficticious Data, Krishnaraj T

4.2.2 IP Addresses Connected - DOS Attack



The Number of Requests Made per Device by the Household - DDOS Attack Demonstration

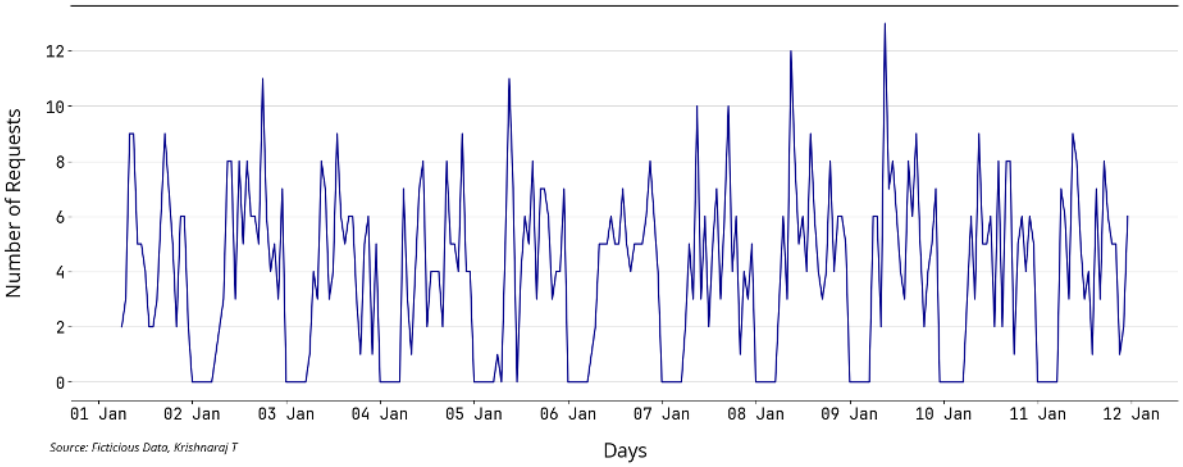
IP address 192.168.1.20 is the attacker with most requests.

Source: Ficticious Data, Krishnaraj T

4.2.3 Hourly Usage - Normal Usage

Hourly Traffic Distribution of the Household

The household is most active during the day. Almost Zero traffic is noted between hours of 2am to 5am. This is normal

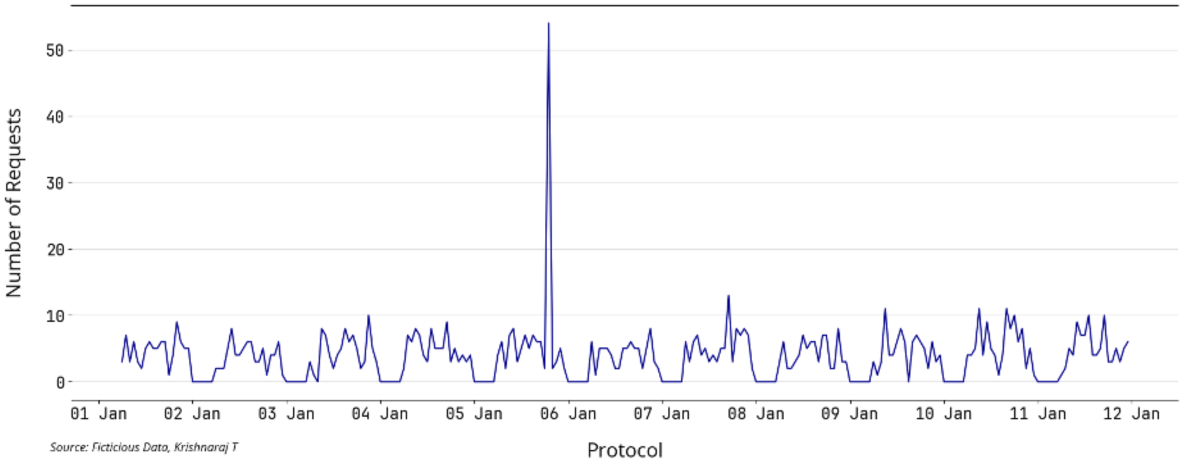


Source: Ficticious Data, Krishnaraj T

4.2.4 Hourly Usage - Normal Usage

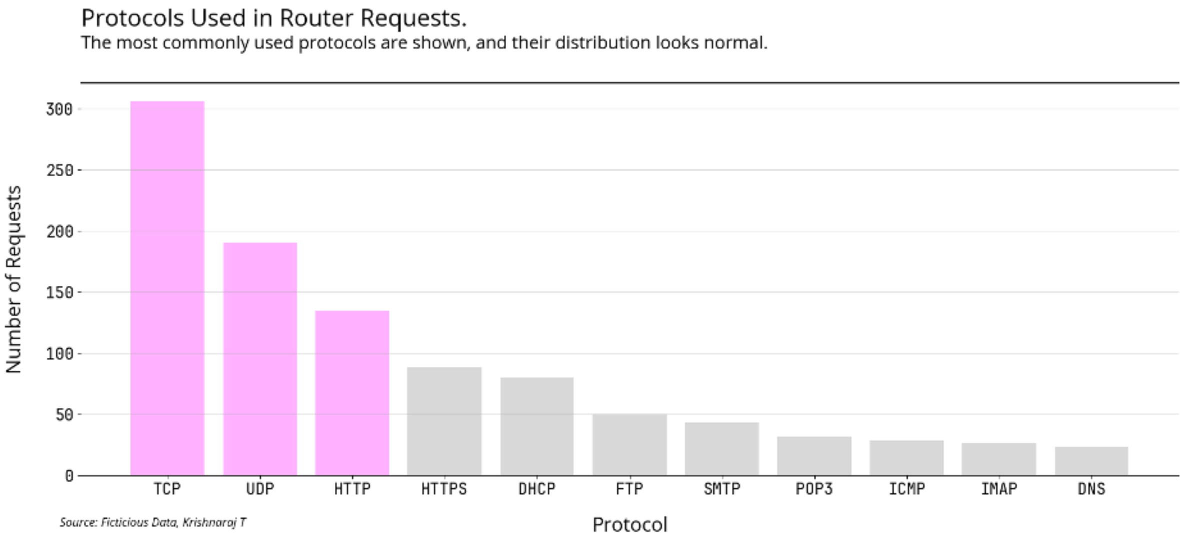
Hourly Traffic Distribution of the Household - DDoS Attack Demo

The extreme spike on Wednesday night is clearly visible as a sign of a DDoS attack

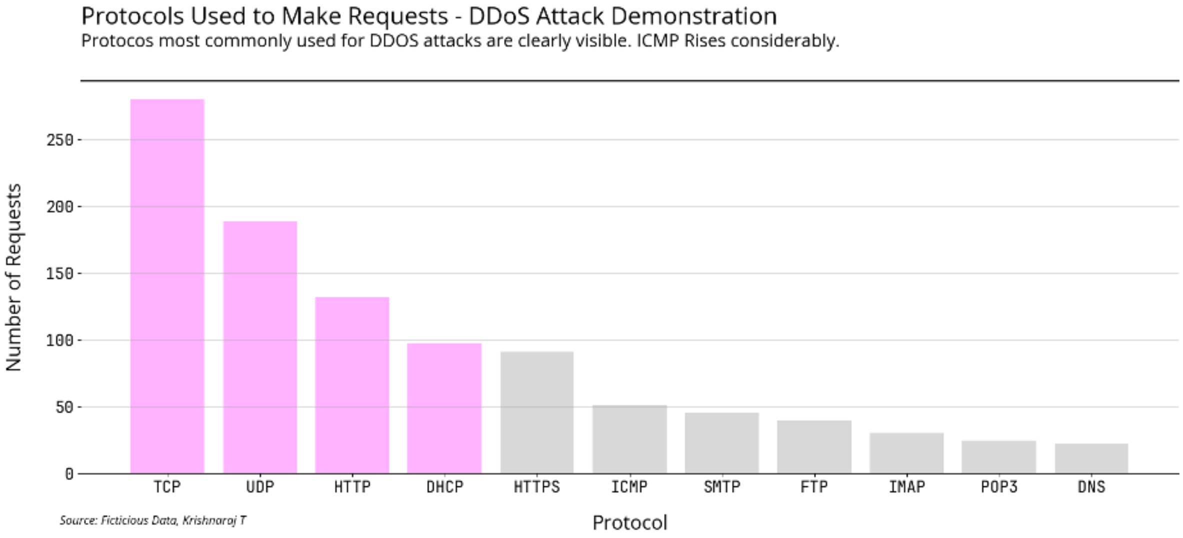


Source: Ficticious Data, Krishnaraj T

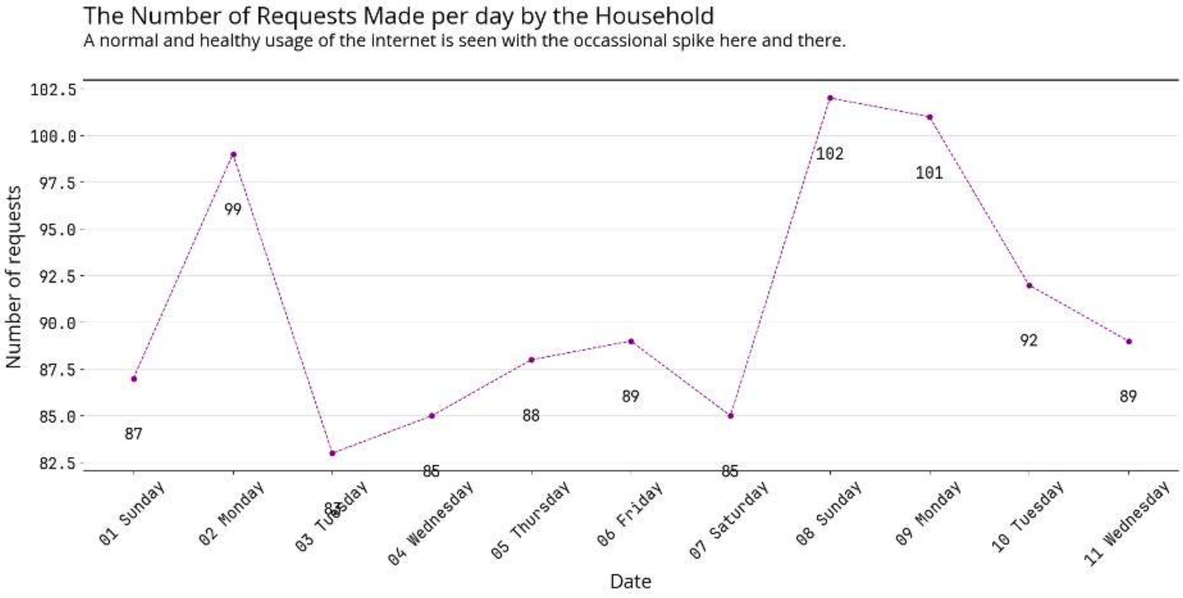
4.2.5 Protocols Used - Normal Usage



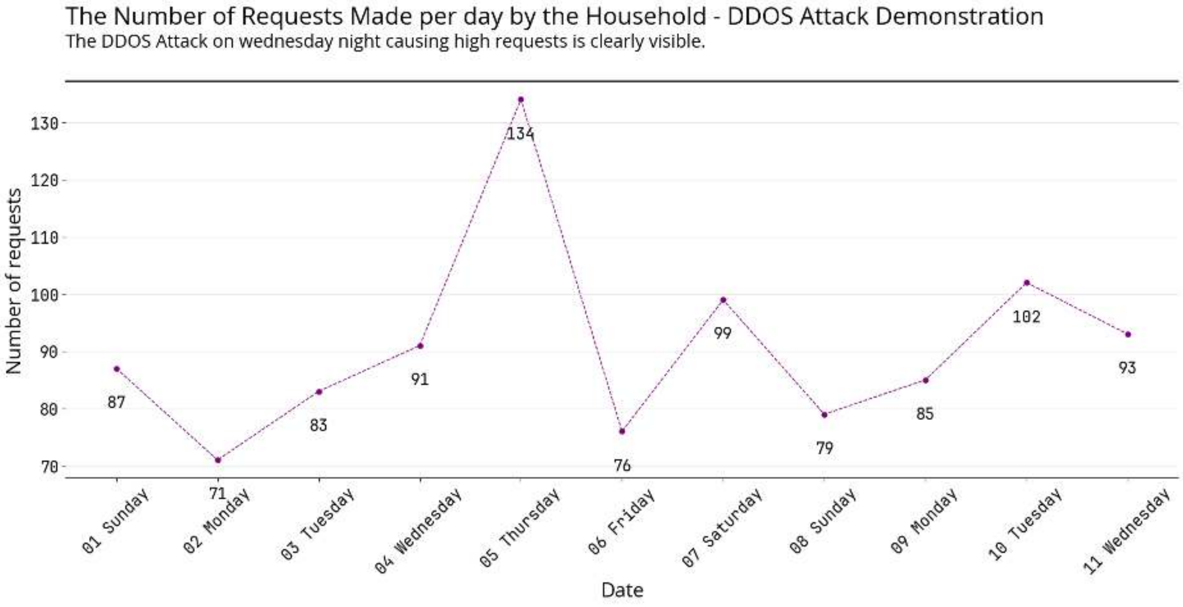
4.2.6 Protocol Usage - DOS Attack



4.2.7 Daily Usage - Normal Usage

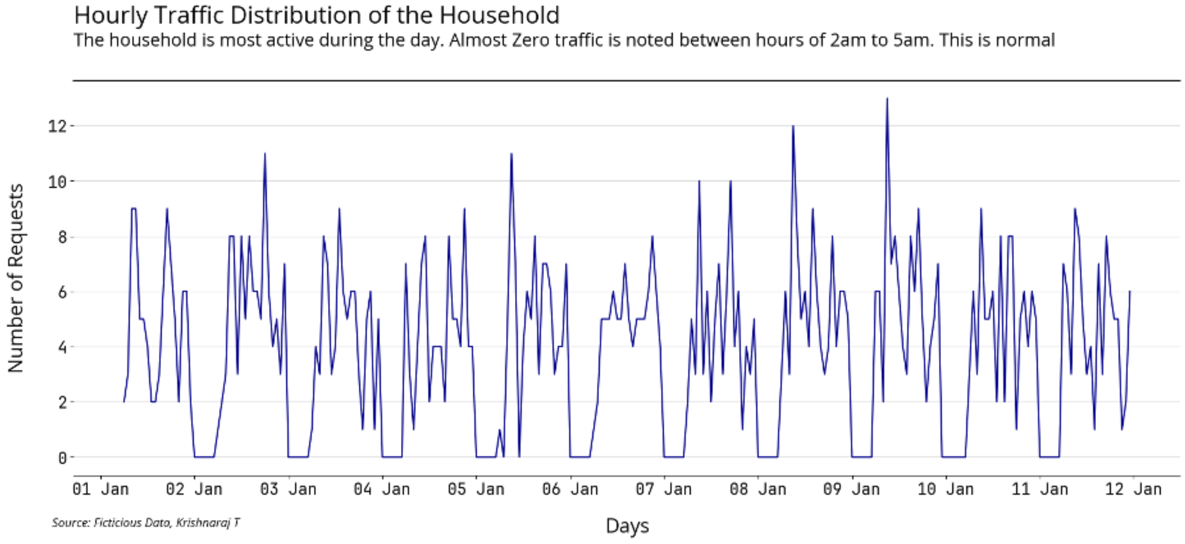


4.2.8 Daily Usage - DOS Attack



4.3 Brute Force Attack Data

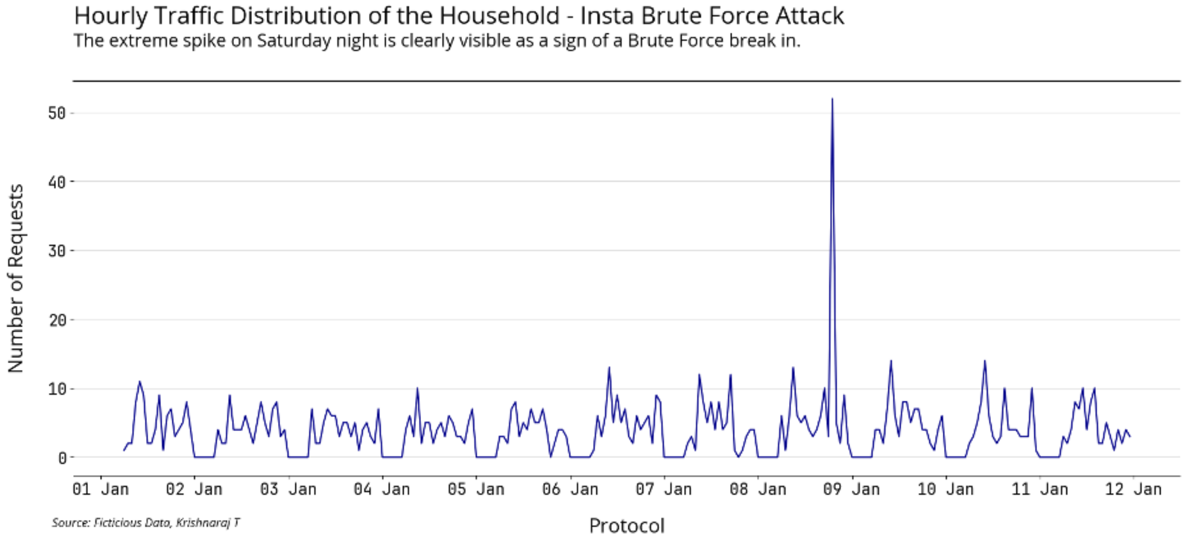
4.3.1 Hourly Usage - Normal Usage


Hourly Traffic Distribution of the Household

The household is most active during the day. Almost Zero traffic is noted between hours of 2am to 5am. This is normal

Source: Ficticious Data, Krishnaraj T

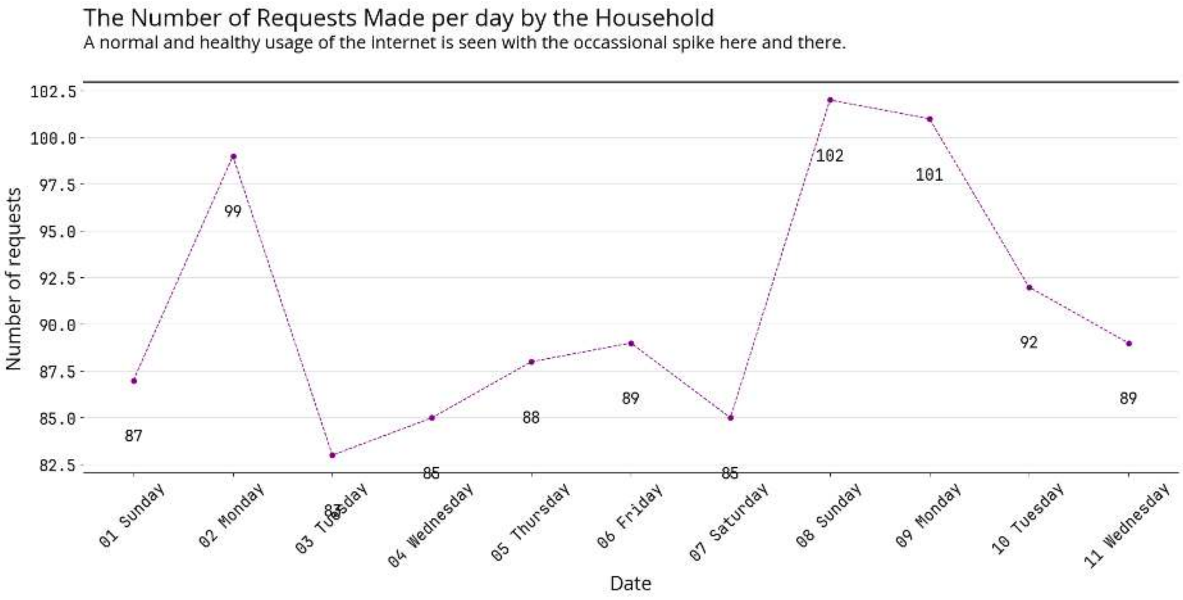
4.3.2 Hourly Usage - Brute Force Attack


Hourly Traffic Distribution of the Household - Insta Brute Force Attack

The extreme spike on Saturday night is clearly visible as a sign of a Brute Force break in.

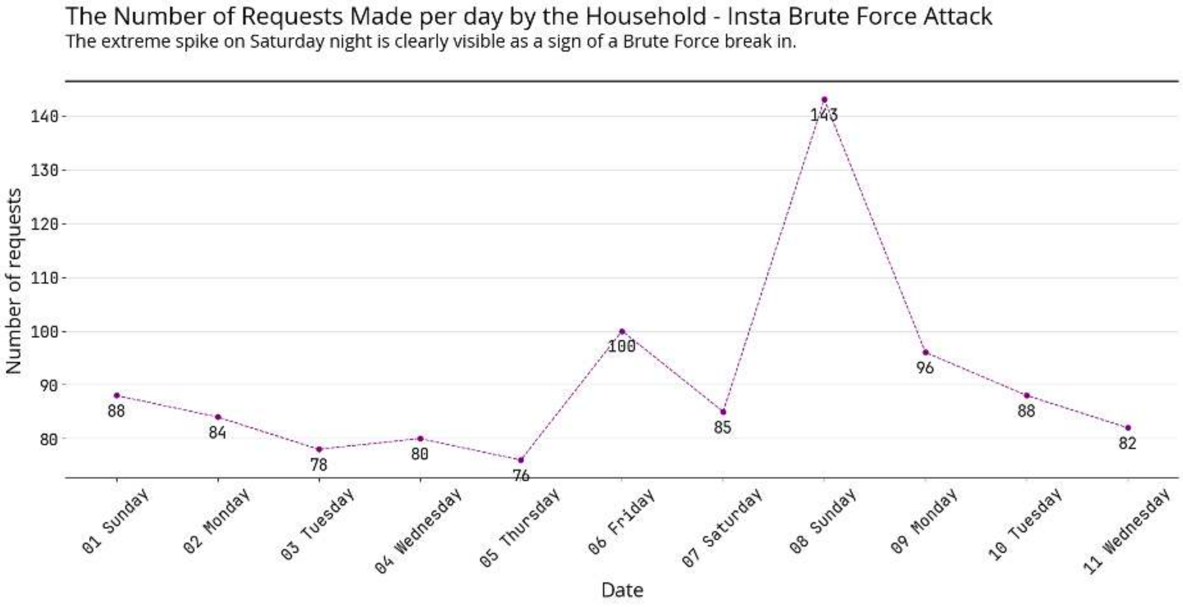
Source: Ficticious Data, Krishnaraj T

4.3.3 Daily Usage - Normal Usage


The Number of Requests Made per day by the Household

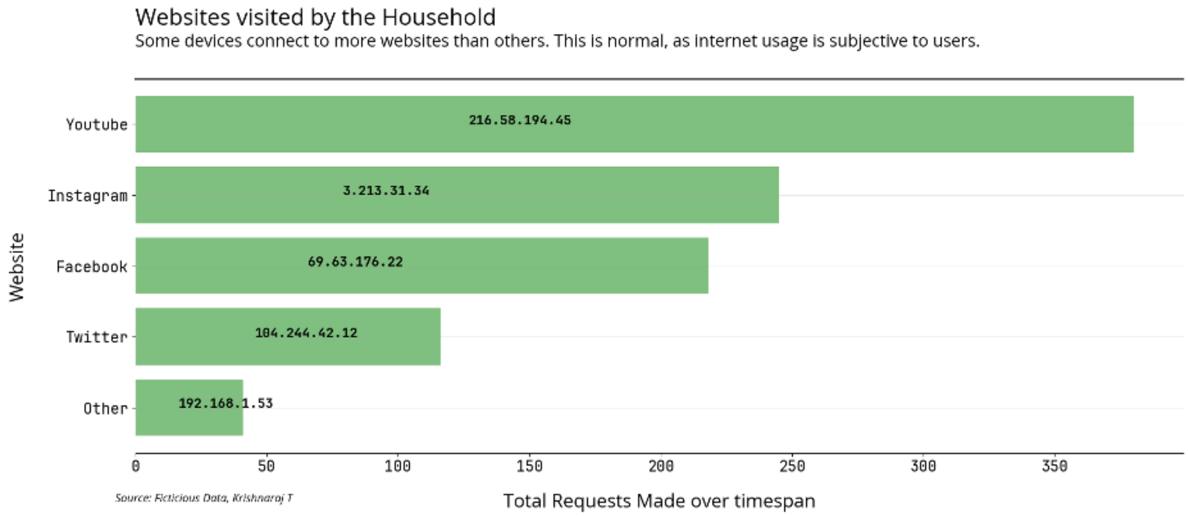
A normal and healthy usage of the internet is seen with the occassional spike here and there.

4.3.4 Daily Usage - Brute Force Attack


The Number of Requests Made per day by the Household - Insta Brute Force Attack

The extreme spike on Saturday night is clearly visible as a sign of a Brute Force break in.

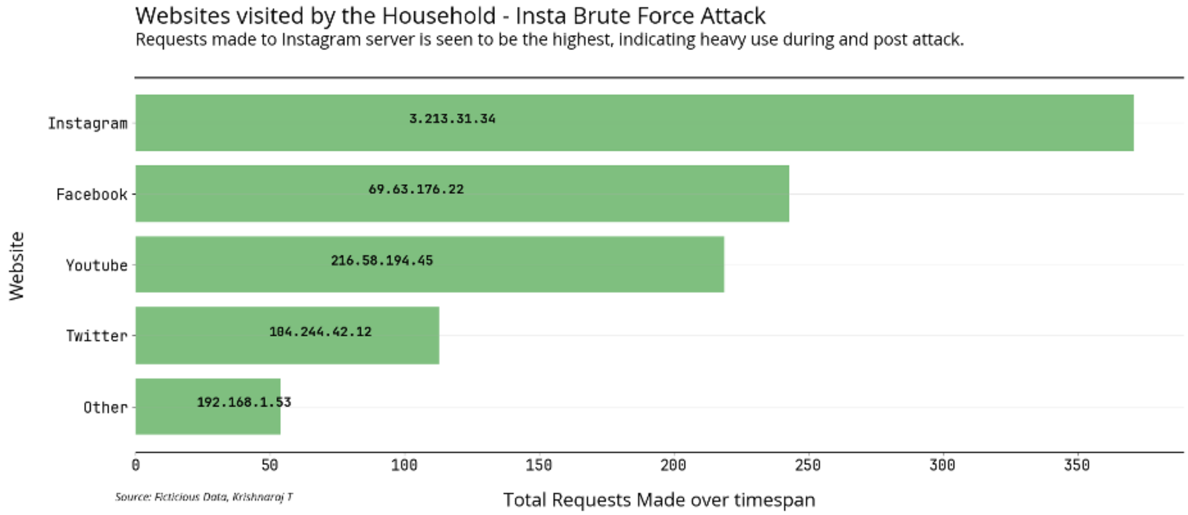
4.3.5 Websites Visited - Normal Usage


Websites visited by the Household

Some devices connect to more websites than others. This is normal, as internet usage is subjective to users.

Source: Ficticious Data, Krishnaraj T

4.3.6 Websites Visited - Brute Force Attack

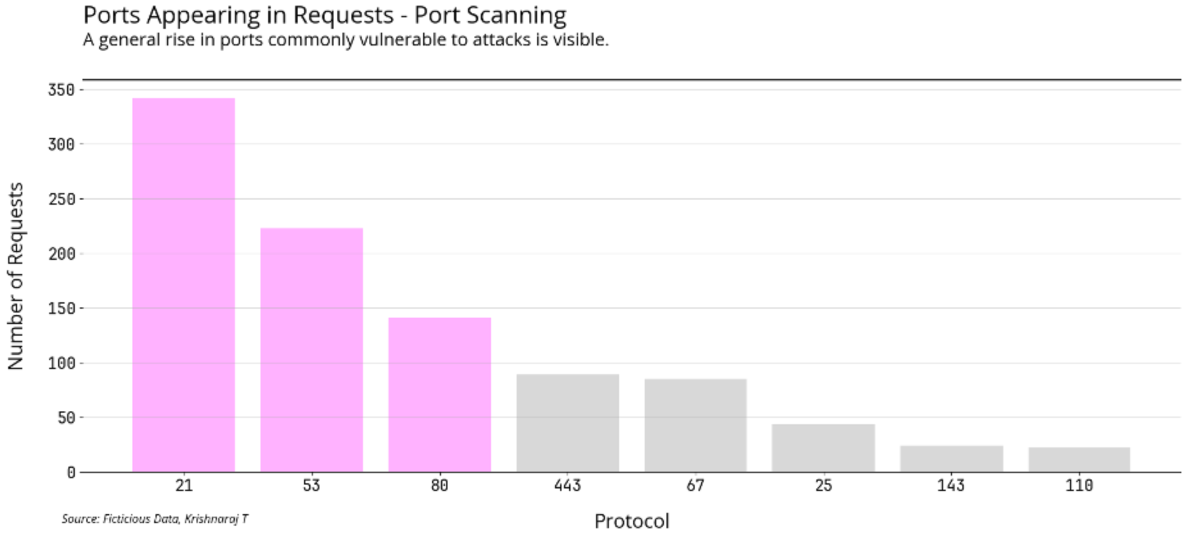

Websites visited by the Household - Insta Brute Force Attack

Requests made to Instagram server is seen to be the highest, indicating heavy use during and post attack.

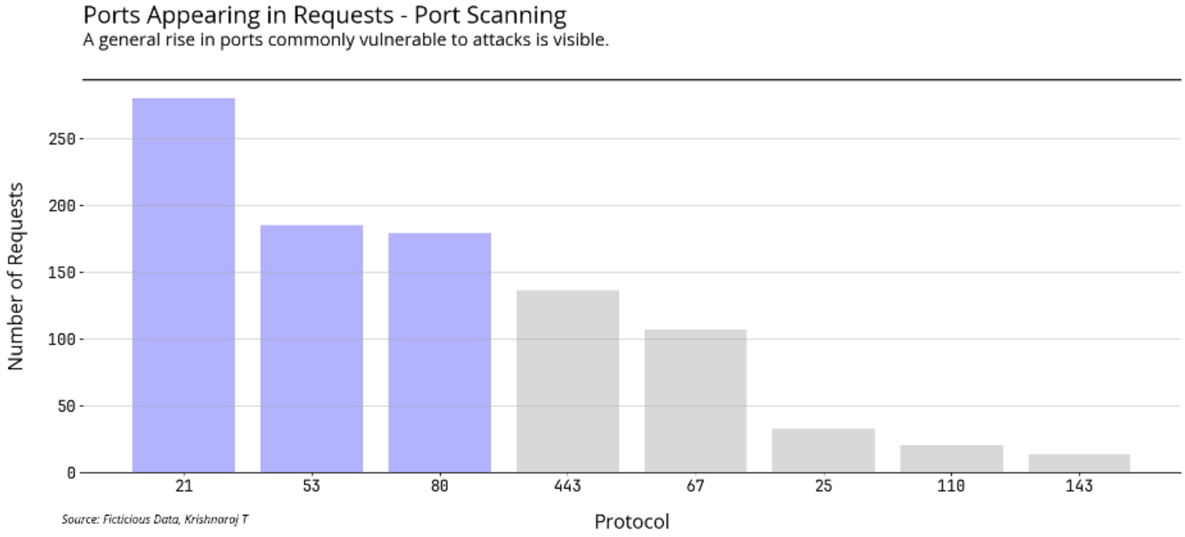
Source: Ficticious Data, Krishnaraj T

4.4 Port Scanning Data

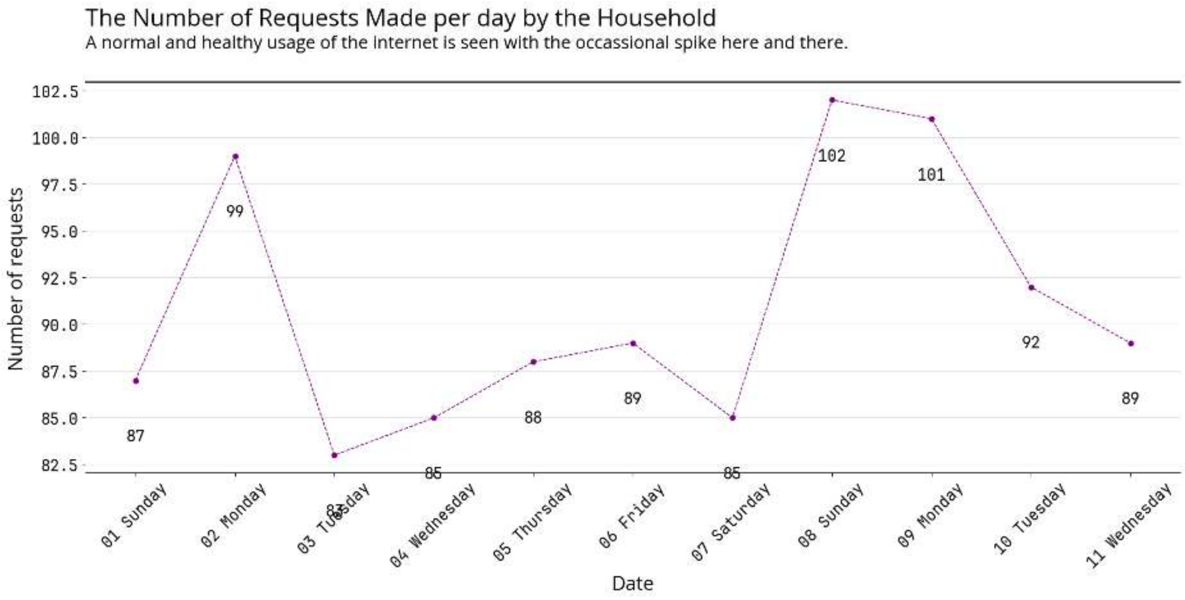
4.4.1 Ports Used - Normal Usage



4.4.2 Port Usage - Port Scanning Attack



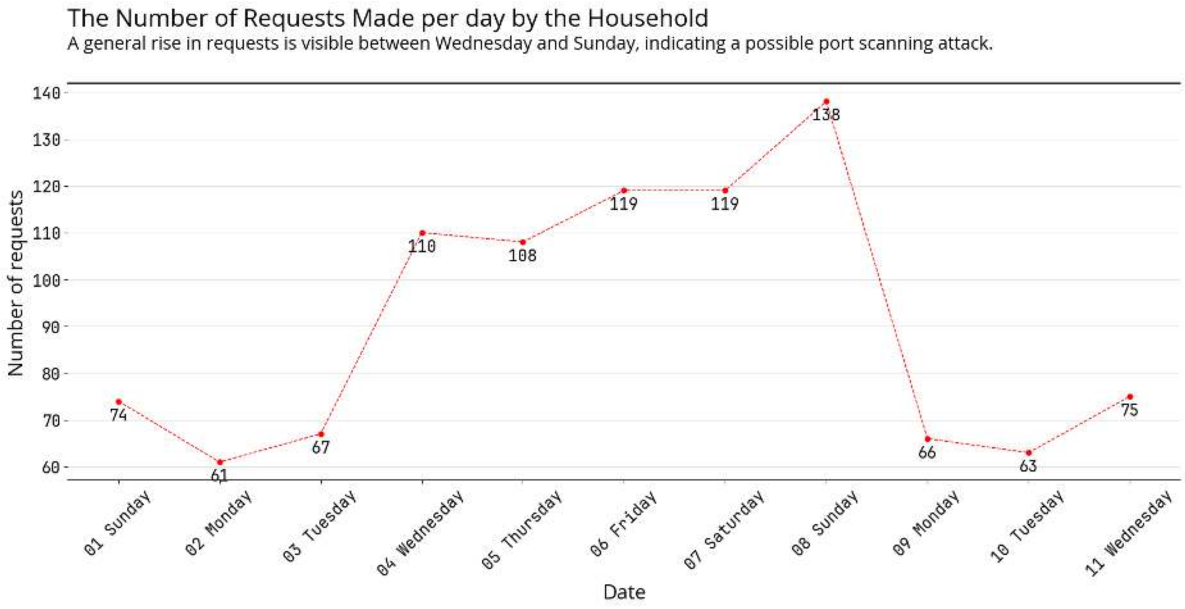
4.4.3 Daily Usage - Normal Usage



The Number of Requests Made per day by the Household

A normal and healthy usage of the internet is seen with the occassional spike here and there.

4.4.4 Daily Usage - Port Scanning Attack



The Number of Requests Made per day by the Household

A general rise in requests is visible between Wednesday and Sunday, indicating a possible port scanning attack.

5 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code
Compilers or Interpreters: Python 3.11

6 Code

6.1 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"
```

6.2 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.

7 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]
```

8 Let us now simulate some attacks

8.1 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]
```

8.2 Hourly Traffic Distribution of the Household - DDoS Attack Demo

8.3 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]
```

8.4 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]
```

9 Conclusion

Thus, we have successfully simulated the attacks and analysed the logs generated by the router. We have also learnt how to analyse the logs and what to look for in the logs.

References

[1] *Digital Evidence and Computer Crime: Forensic Science, Computers, and the Internet.* Academic Press.

[2] Vskills - Digital Forensic Tools