Masterminds

Practice analyzing malicious traffic using Brim.

- *Andres Valdivieso Pinilla - Líder de Ciberseguridad (Consultor)*
- *www.linkedin.com/in/andres-valdivieso-pinilla*

Introducción

Brim es una herramienta de código abierto utilizada para la exploración y análisis forense de tráfico de red, diseñada para trabajar con grandes volúmenes de datos de captura de paquetes (PCAP) y registros de Zeek. Ofrece una interfaz intuitiva para filtrar, buscar y visualizar eventos de red, facilitando la detección de amenazas y el análisis de incidentes de seguridad. Además, permite exportar datos a Zeek y Splunk, integrándose en flujos de trabajo de ciberseguridad.

Planteamiento del compromiso

Tres equipos del departamento de Finanzas de Pfeffer PLC se vieron comprometidos. Sospechamos que la causa inicial del ataque fue un intento de phishing y una unidad USB infectada. El equipo de Respuesta a Incidentes logró extraer los registros de tráfico de red de los endpoints. Utilice Brim para investigar el tráfico de red en busca de indicios de un ataque y determinar quién está detrás de los ataques.

NOTA: NO interactúe directamente con ningún dominio ni dirección IP en este desafío.

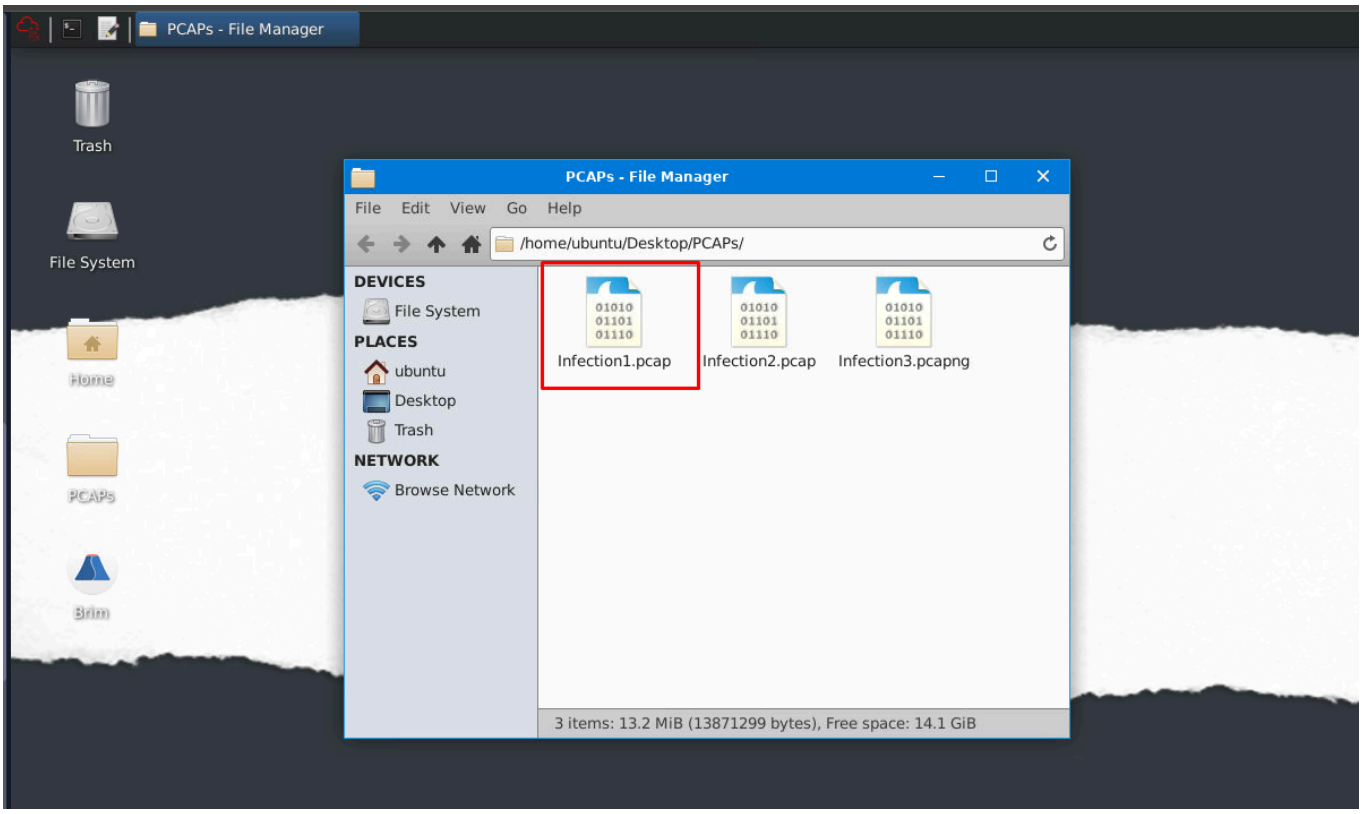
[Infección 1]

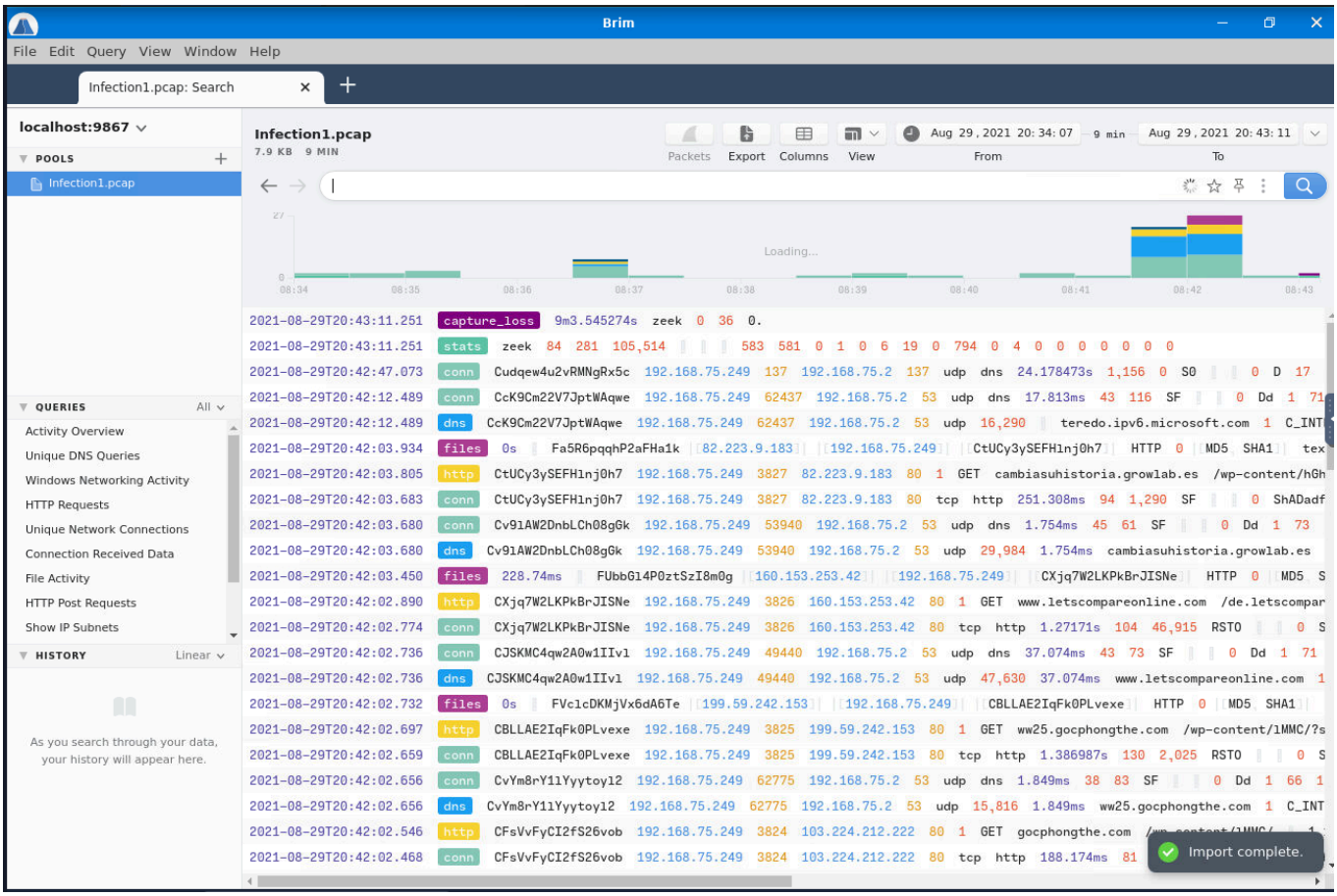
Comience cargando la captura de paquetes de Infection1 en Brim para investigar el evento de vulneración en la primera máquina. Todos los PCAP se pueden encontrar aquí: `/home/ubuntu/Desktop/PCAPs`

Nota : Para preguntas que requieren múltiples respuestas, sepárelas con una coma.

Responda las preguntas a continuación

Lo primero es abrir los pcap para ver la información con la que cuento





Proporcione la dirección IP de la víctima.

Lo primero que valide fueron los path que tengo a dispocicion en este pecap cpn esto voy a ver las conecciones de red

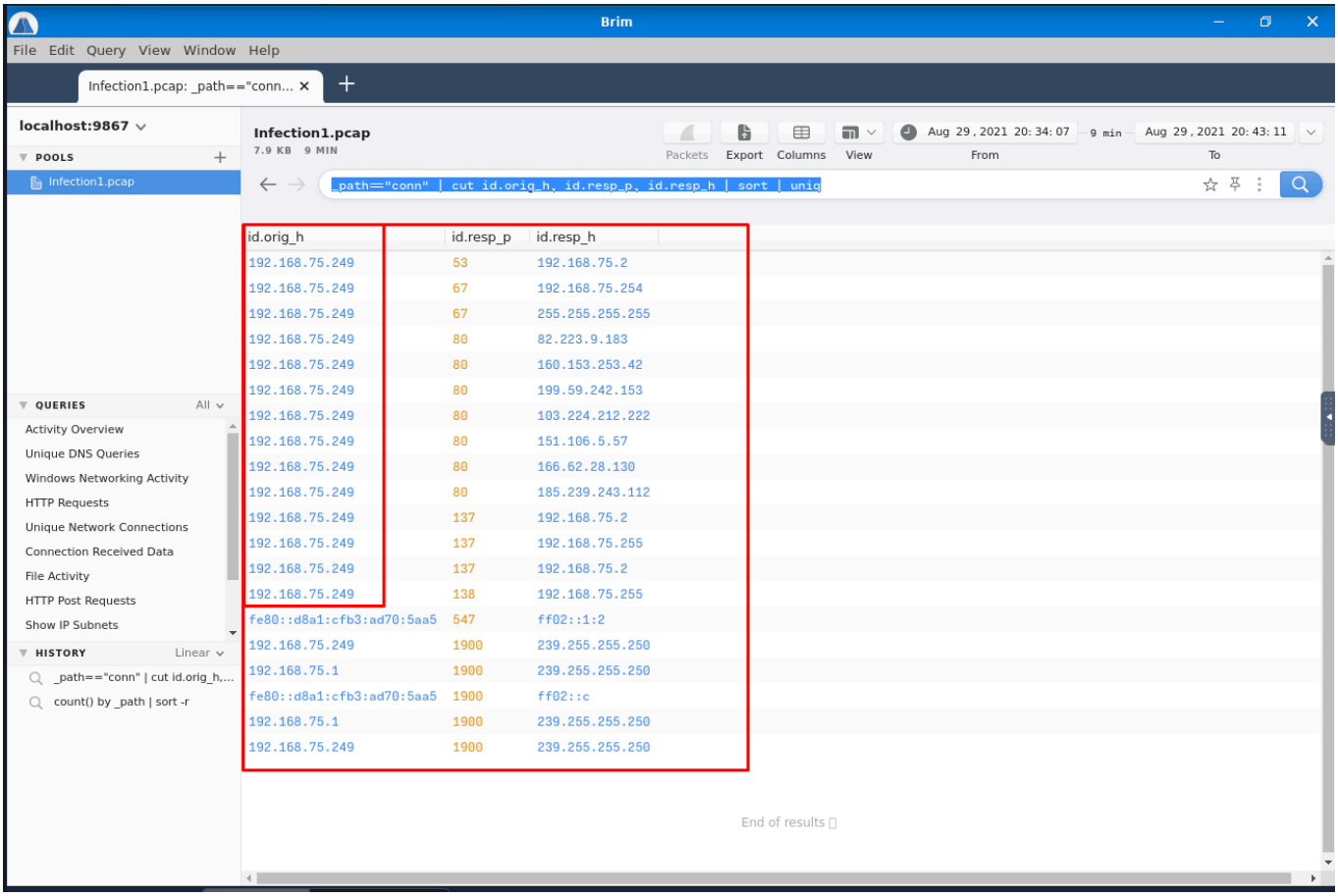
```
_path=="conn" | cut id.orig_h, id.resp_p, id.resp_h | sort | uniq
```

Explicación de la consulta:

1. `_path=="conn"`
   - Filtra los registros para trabajar solo con datos de conexiones (`conn.log`), que contienen información sobre direcciones IP, puertos, duración de sesión, bytes transferidos, etc.
2. `cut id.orig_h, id.resp_p, id.resp_h`
   - Extrae solo los siguientes campos relevantes:
     - `id.orig_h`: IP de origen (quién inició la conexión).
     - `id.resp_p`: Puerto de destino al que se conectó la IP de origen.
     - `id.resp_h`: IP de destino (el servidor o sistema remoto al que se conectó la IP de origen).
3. `sort`
   - Ordena los resultados alfabéticamente o numéricamente, dependiendo del tipo de dato.
4. `uniq`
   - Elimina registros duplicados, dejando solo valores únicos.

Lo que busco con este comando es:

- Extrae las direcciones IP de origen, los puertos de destino y las direcciones IP de destino.
- Ordena los resultados.
- Muestra solo combinaciones únicas de estos tres valores.

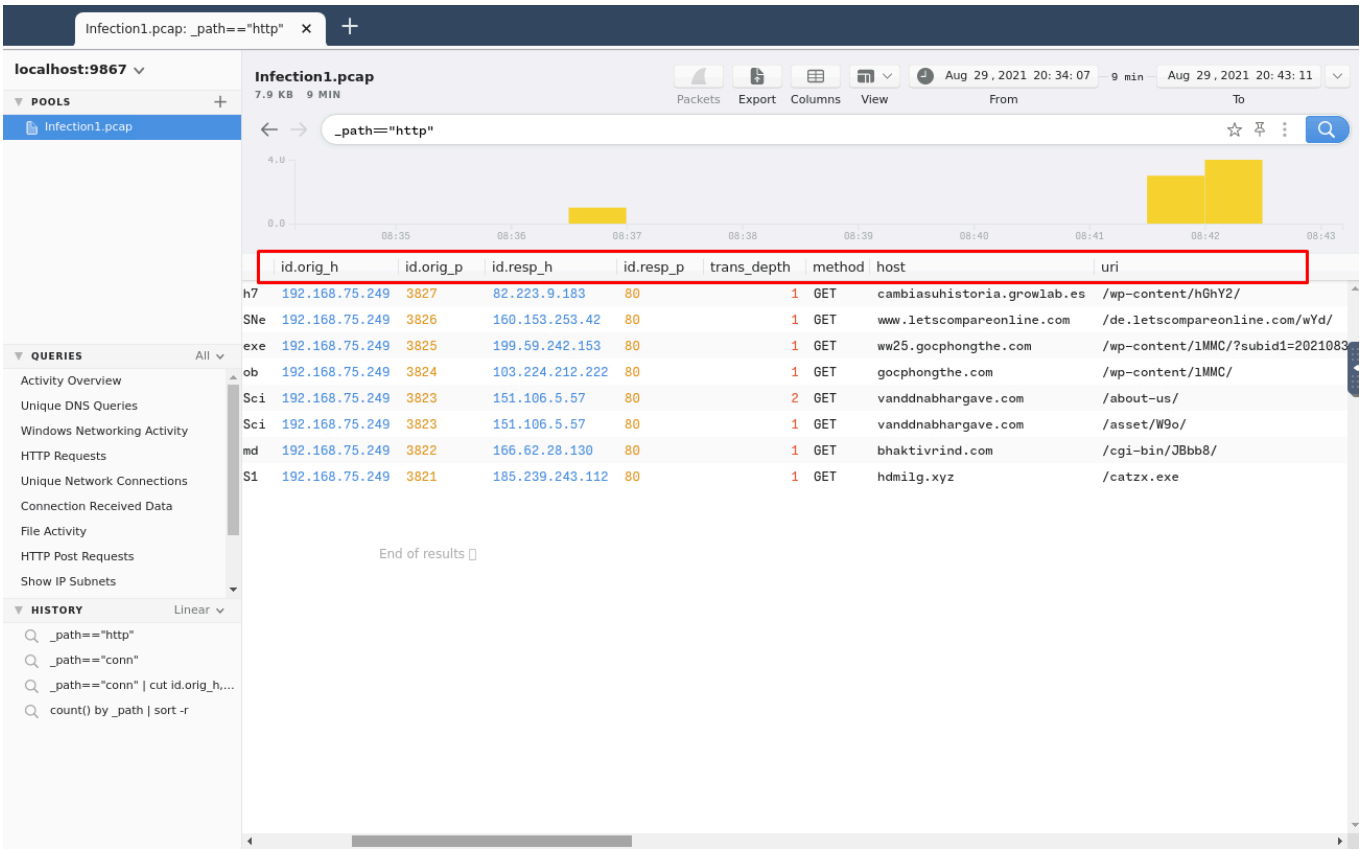


Con esto puedo determinar cual es la única ip interna.

Ahora nos piden validar la conexiones HTTP que la víctima intentó establecer con dos dominios sospechosos con el estado "404 No encontrado". y nos piden proporcionar los hosts/dominios solicitados.

Para esto lo primero es validar los registros en el http

```
_path=="http"
```



Con esto ya saco las casillas que debo usar para la consulta completa

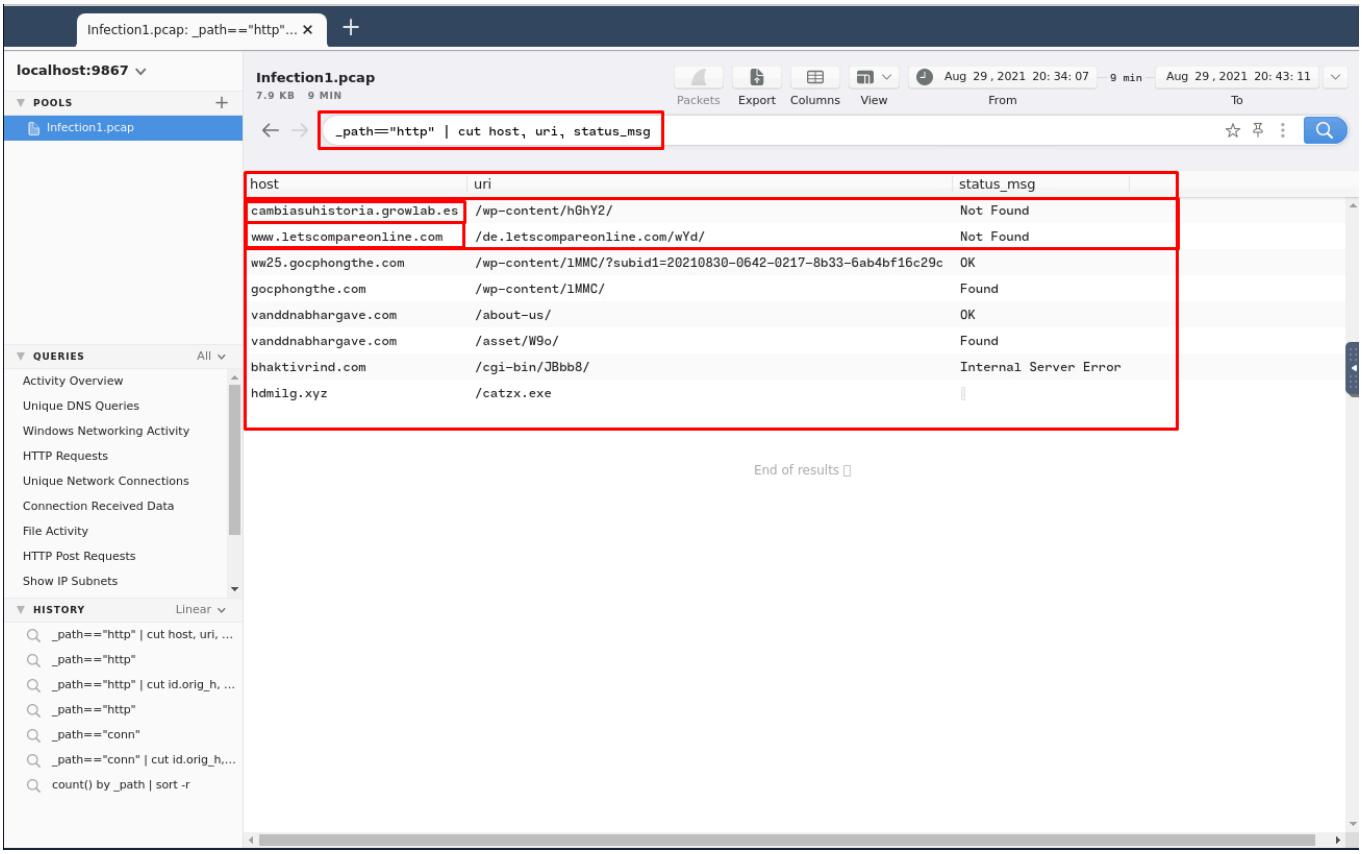
```
_path=="http" | cut host, uri, status_msg
```

Explicación de la consulta:

1. `_path=="http"`
   - Filtra los registros para trabajar únicamente con eventos HTTP (provenientes de `http.log` en Zeek).
2. `cut host, uri, status_msg`
   - Extrae solo los siguientes campos:
     - `host`: Nombre del servidor o dominio al que se realizó la solicitud HTTP.
     - `uri`: La ruta de la solicitud HTTP (por ejemplo, `/index.html`).
     - `status_msg`: El mensaje de estado HTTP asociado a la respuesta (por ejemplo, "OK", "Not Found", "Forbidden").

Lo que busco ver con esta consulta es:

- Filtra los registros HTTP.
- Extrae solo el **host**, la **URI** solicitada y el **mensaje de estado** de la respuesta.



Podemos identificar que las únicas dos URL que están asociadas al Not Found son las dos primeras url.

La víctima se conectó correctamente a uno de los dominios mediante HTTP y recibió un valor de response_body_len de 1309 (**tamaño del contenido sin comprimir de los datos transferidos desde el servidor**). con esto nos piden Indicar el dominio y la dirección IP de destino.

Esta consulta tambien se realiza desde el archivo del http que tiene la consulta ahora lo unico que tenemos que tener en cuenta es el tamaño del contenido ya que nos proporcionan el valor especifico es mas facil encontarr los valores solicitados usando esta consulta:

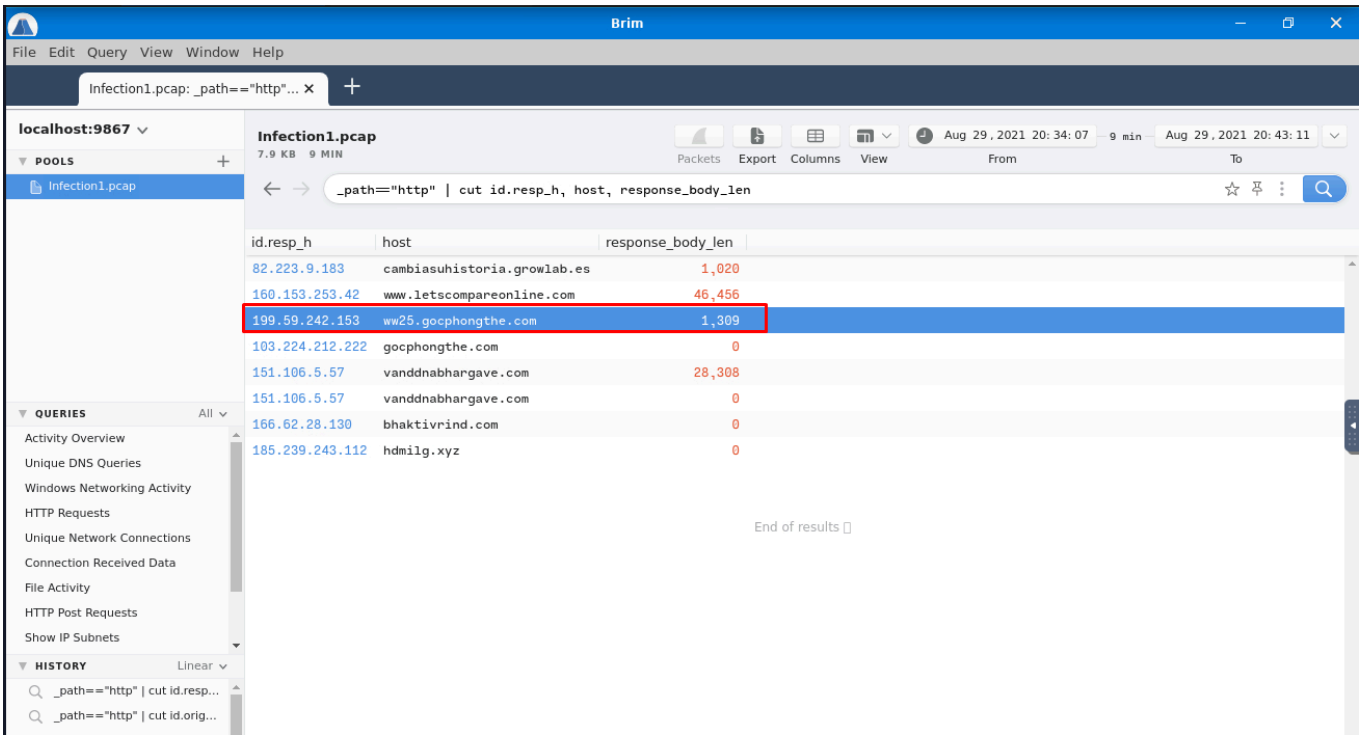
```
_path=="http" | cut id.orig_h, host, response_body_len
```

Explicación de la consulta:

1. `_path=="http"`
   - Filtra los registros para trabajar únicamente con eventos HTTP (provenientes de `http.log` en Zeek).
2. `cut id.orig_h, host, response_body_len`
   - Extrae solo los siguientes campos relevantes:
     - `id.orig_h` → Dirección IP del cliente que realizó la solicitud HTTP.
     - `host` → Nombre del servidor o dominio al que se realizó la solicitud HTTP.
     - `response_body_len` → Tamaño (en bytes) del cuerpo de la respuesta HTTP.

Con esta podemos:

- Filtra los eventos HTTP.
- Extrae la IP de origen, el host al que se conectó y el tamaño del contenido recibido en la respuesta.



Asi podemos identificar cual es el dominio y ip a la cual se conecto la victima.

¿Cuántas solicitudes de DNS únicas se realizaron al dominio cab[.]myfkn[.]com (incluido el dominio en mayúsculas)?

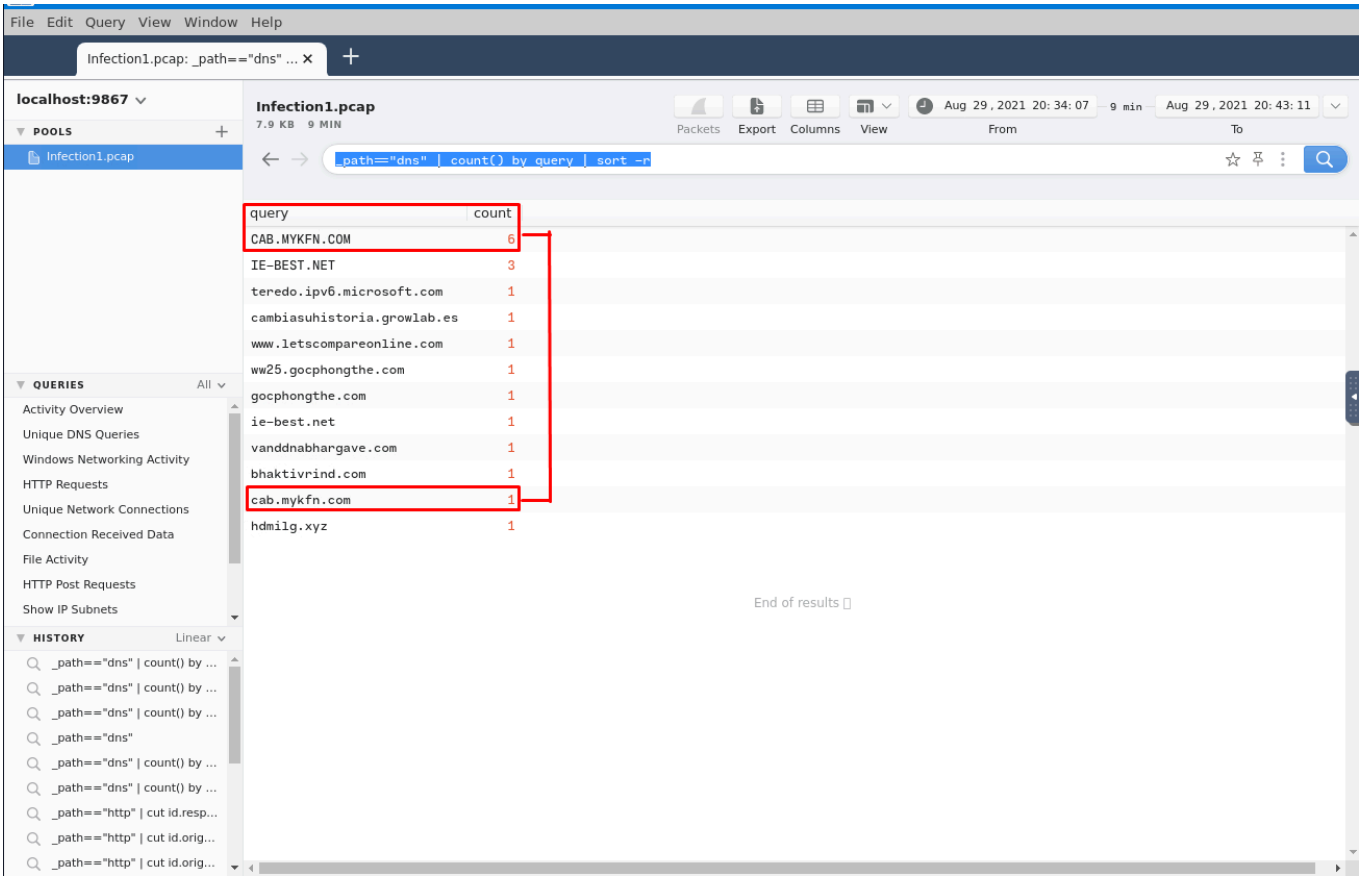
```
_path=="dns" | count() by query | sort -r
```

Explicación de la consulta:

1. `_path=="dns"`
   Filtra los registros para trabajar únicamente con eventos DNS.
2. `count() by query`
   Cuenta cuántas veces aparece cada consulta de dominio (`query`).
3. `sort -r`
   Ordena los resultados en orden descendente por la cantidad de consultas.

---

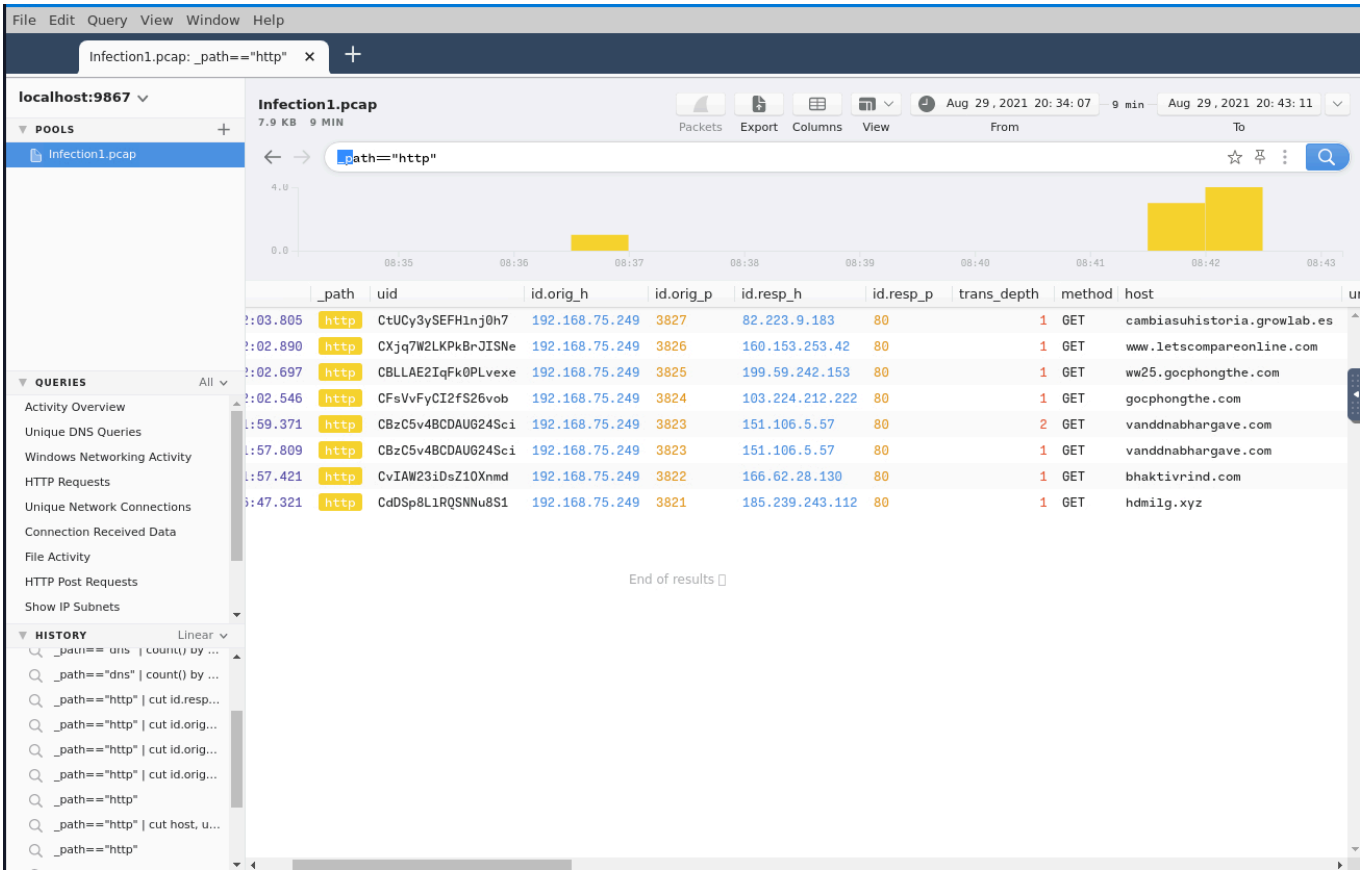
lo que quiero es que me cuente cuántas veces se ha consultado cada dominio y me ordene los resultados en orden descendente, mostrando los dominios más consultados primero.



Se puede ver que el conteo esta separado en por lo que toca tenerlo encuenta a la hora de responder por o qe se suman los dos resultados.

Proporcione la URI del dominio bhaktivrind[.]com al que la víctima accedió a través de HTTP.

```
_path=="http"
```




```
_path=="http" | cut host, id.orig_h, uri | bhaktivrind.com
```

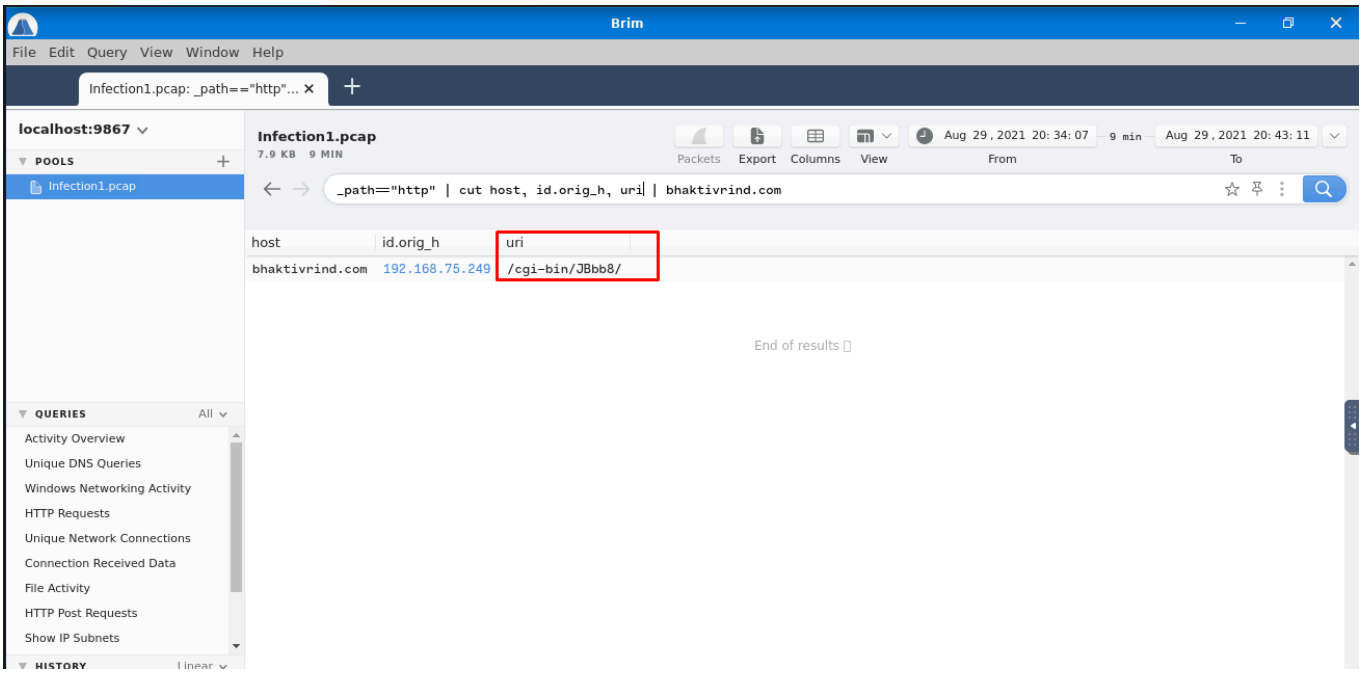
Explicación de la consulta:

1. `_path=="http"`
   Filtra los registros para trabajar únicamente con eventos HTTP.
2. `cut host, id.orig_h, uri`
   Extrae solo los siguientes campos relevantes:
   - **host** → Nombre del servidor o dominio al que se realizó la solicitud HTTP.
   - **id.orig_h** → Dirección IP del cliente que realizó la solicitud HTTP.

- **uri** → La ruta de la solicitud HTTP.

3. `bhaktivrind.com`
   Buscar específicamente solicitudes hacia el dominio **bhaktivrind.com**.

Esta consulta solo filtra eventos HTTP, extrae el host, la IP de origen y la URI de las solicitudes, y luego busca específicamente registros relacionados con el dominio que nos están solicitando que es **bhaktivrind.com**.



Proporcione la dirección IP del servidor malicioso y el ejecutable que la víctima descargó del servidor.

```
_path=="http" | cut id.orig_h, id.resp_h, id.resp_p, method,host, uri
| uniq -c
```

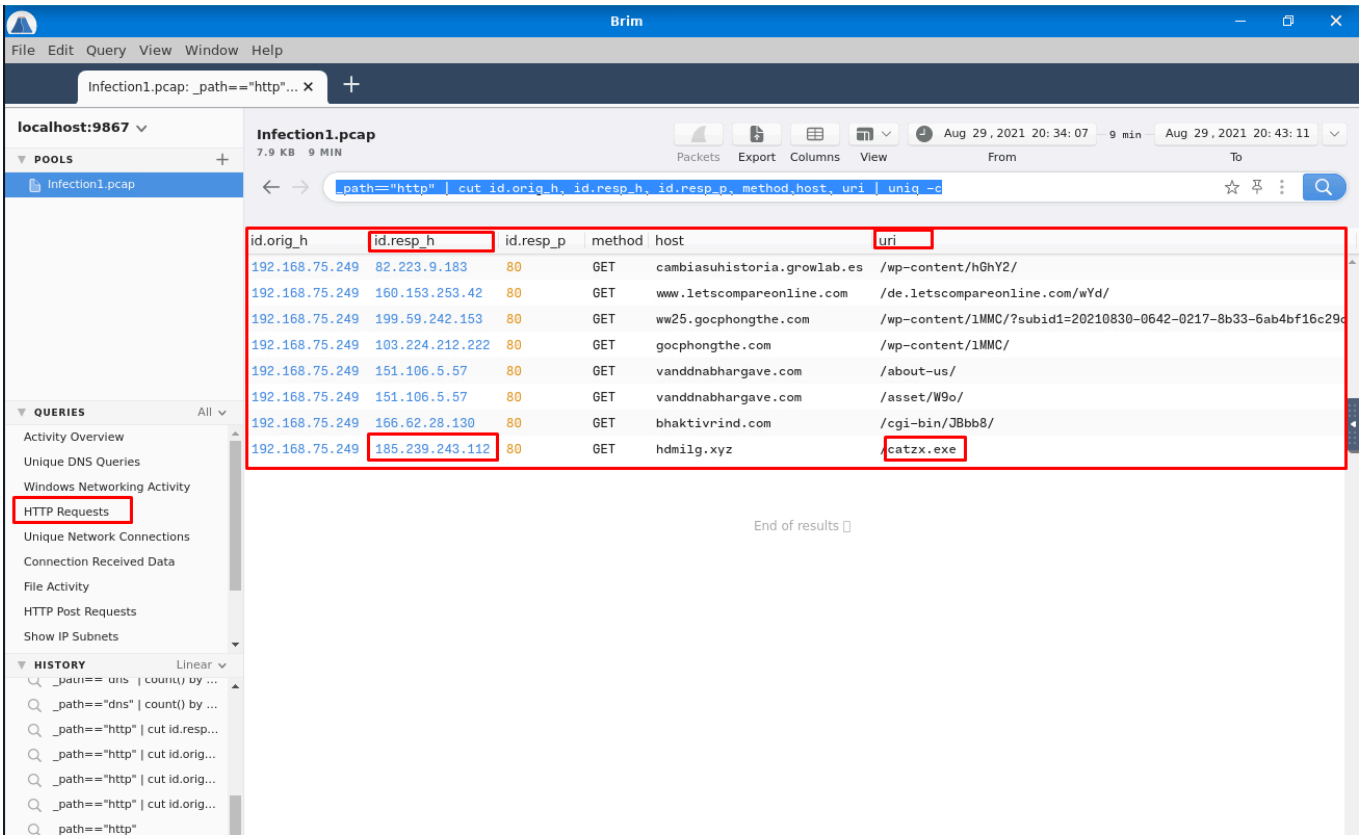
Explicación por partes:

1. `_path=="http"`
   Filtra los registros para trabajar únicamente con eventos HTTP.
2. `cut id.orig_h, id.resp_h, id.resp_p, method, host, uri`
   Extrae los siguientes campos relevantes:
   - **id.orig_h** → Dirección IP del cliente que realizó la solicitud HTTP.

- **id.resp_h** → Dirección IP del servidor de destino.
- **id.resp_p** → Puerto de destino del servidor.
- **method** → Método HTTP utilizado (GET, POST, etc.).
- **host** → Nombre del servidor o dominio al que se realizó la solicitud HTTP.
- **uri** → La URI (ruta) solicitada en la petición HTTP.

3. `uniq -c`
   Cuenta las ocurrencias únicas de las combinaciones de los campos extraídos.

Con este extraemos información sobre las solicitudes (IP de origen, IP de destino, puerto, método HTTP, host y URI), y luego cuenta cuántas veces ocurren combinaciones únicas de esos campos.

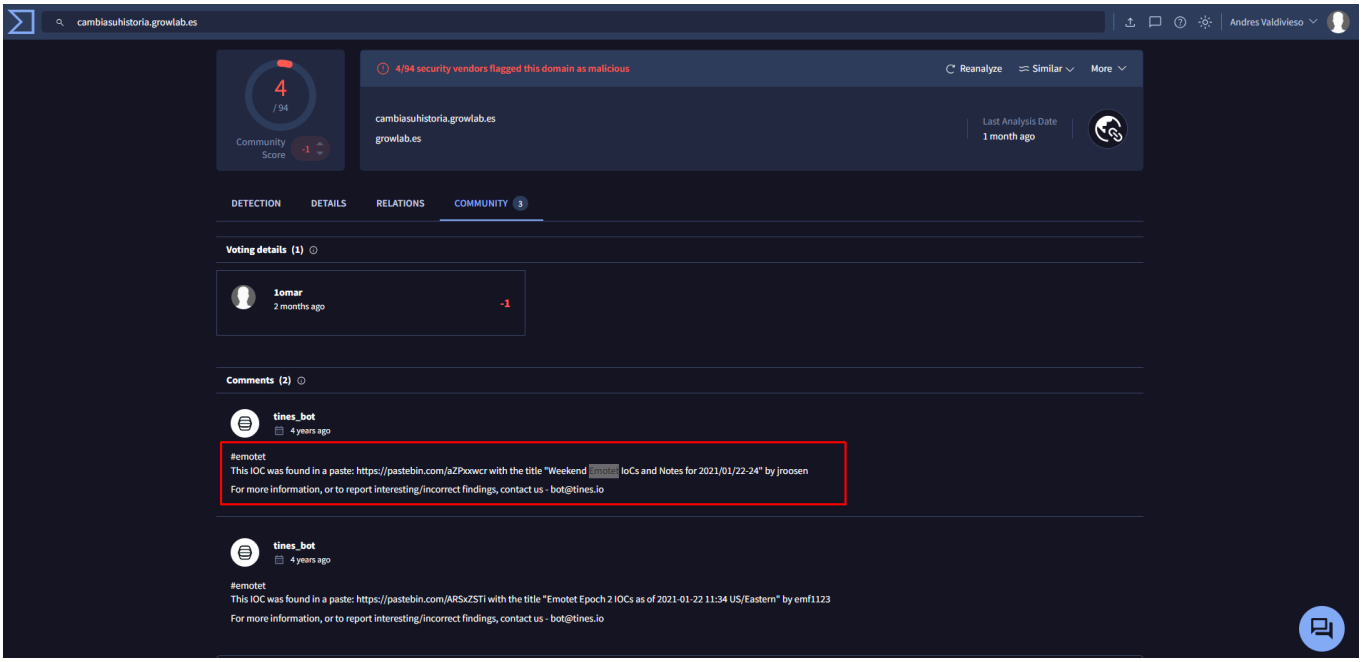


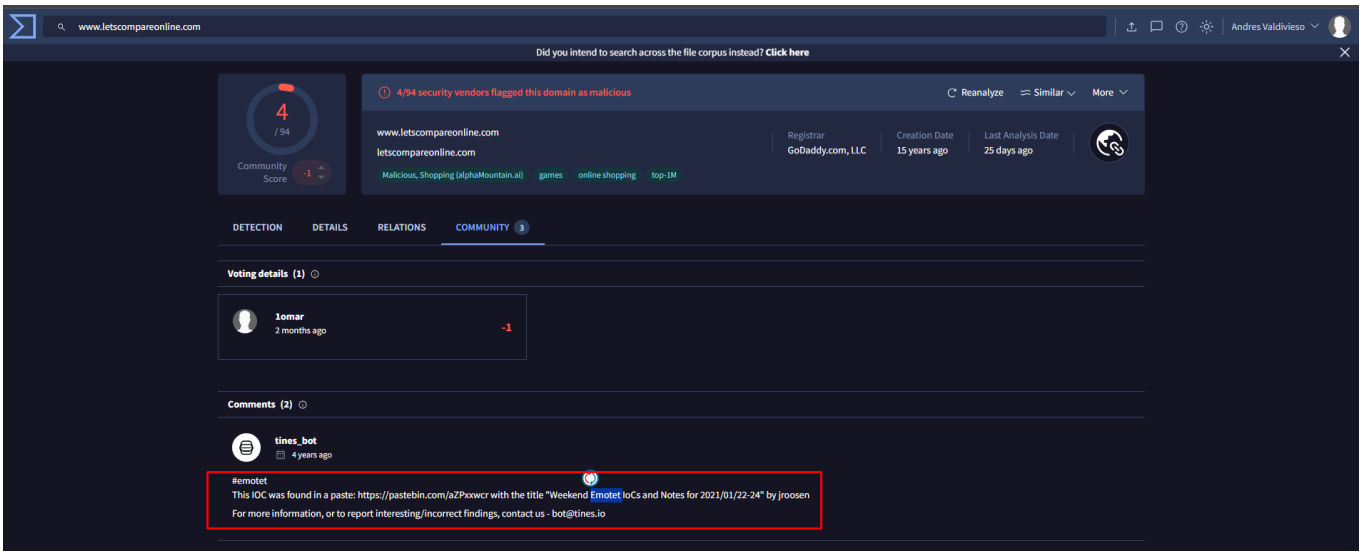
La respuesta esta asociada al único .exe que se ve dentro de las url.

Con base en la información recopilada en la segunda pregunta, proporcione el nombre del malware que utiliza VirusTotal .

En la segunda pregunta nos preguntaban por las conexiones HTTP a dos dominios sospechosos con el estado "404 No encontrado". en este teniamos el

hosts/dominios solicitados estos eran **cambiasuhistoria.growlab.es,www.letscompareonline.com** con esto solo vamos a virustotal y ponemos el dominio:





Para los dos esta el mismo nombre de malware que es Emotet.

Nota:

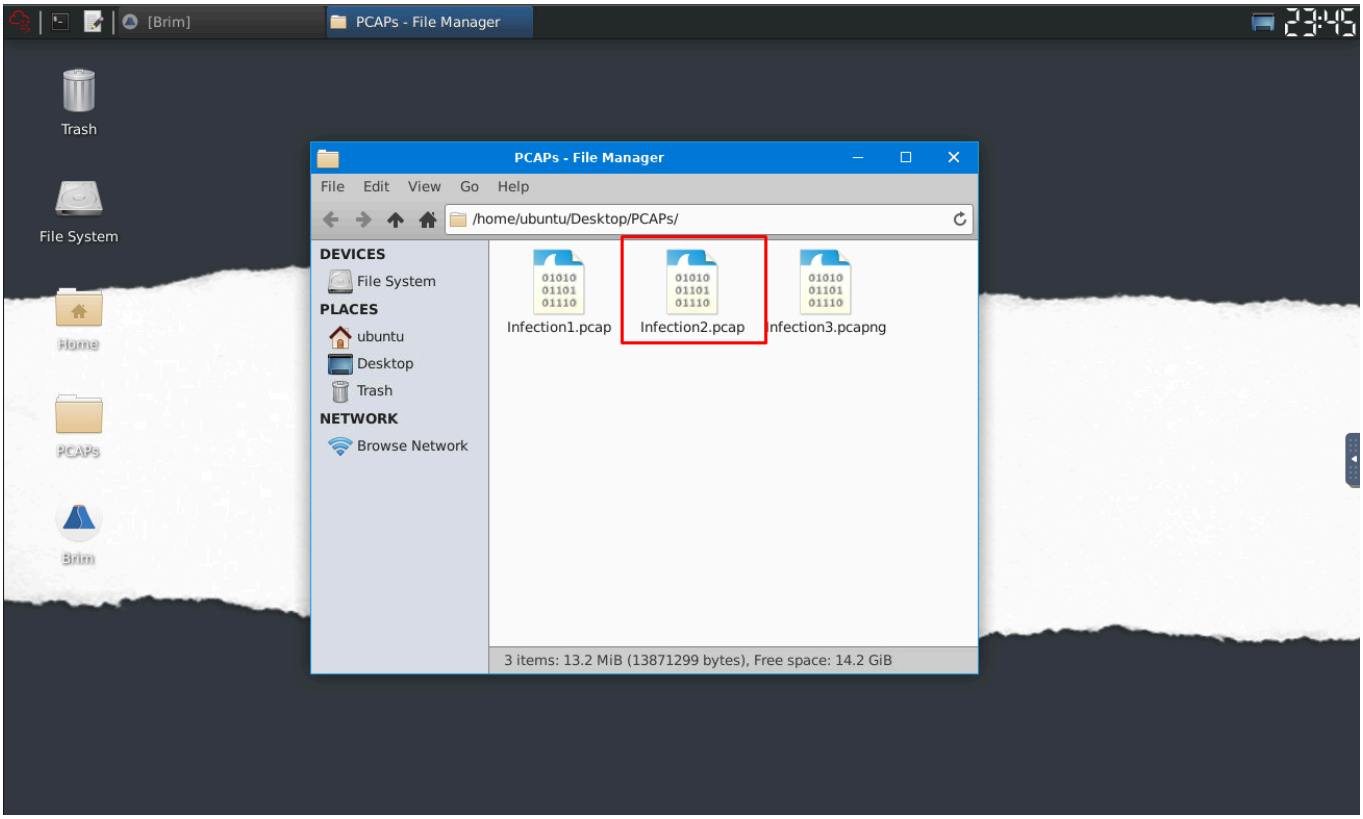
Para los siguientes ejercicios se sigue el mismo principio con las consultas.

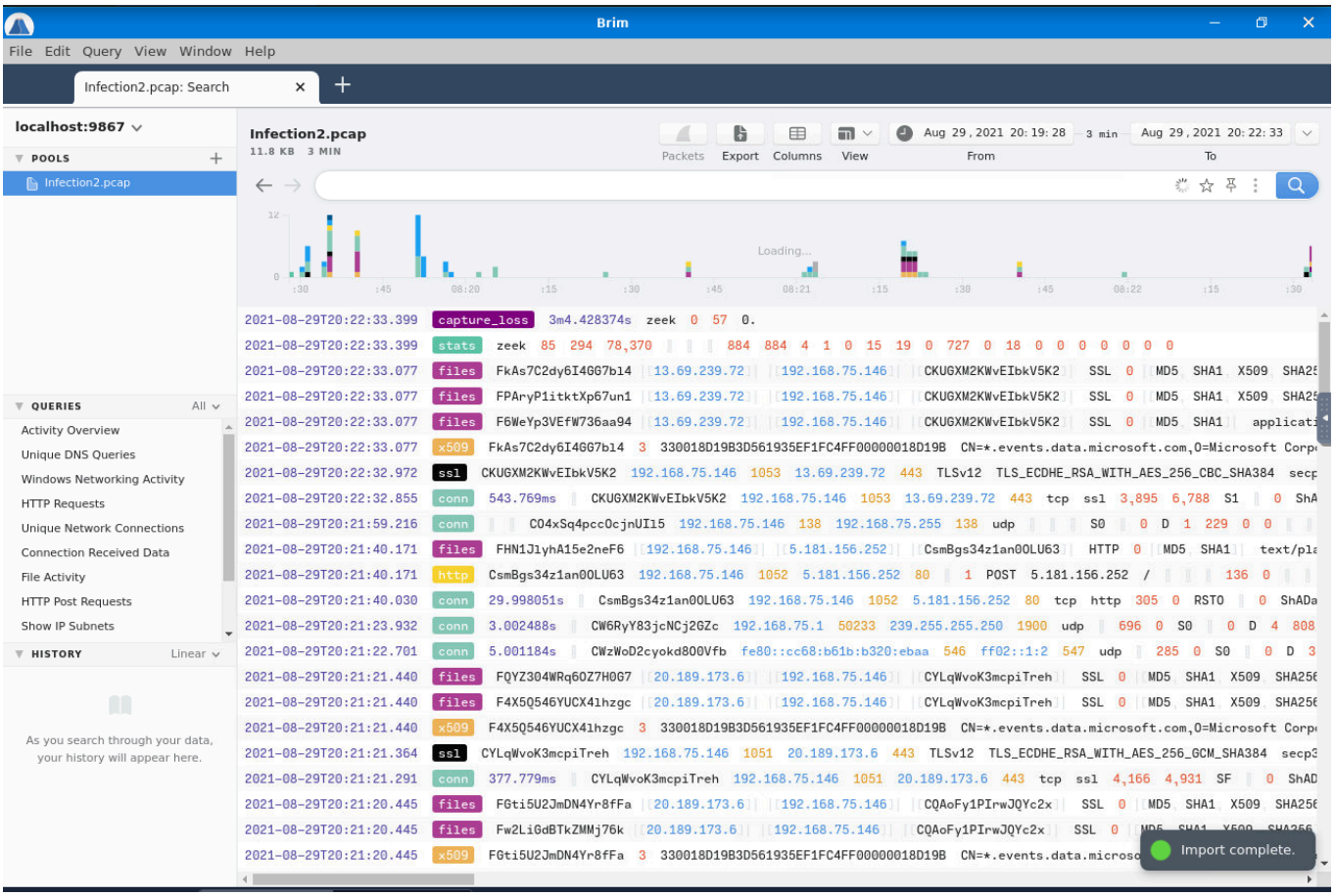
[Infección 2]

Por favor, navegue hasta la captura de paquetes de Infection2 en Brim para investigar el evento de compromiso para la segunda máquina.

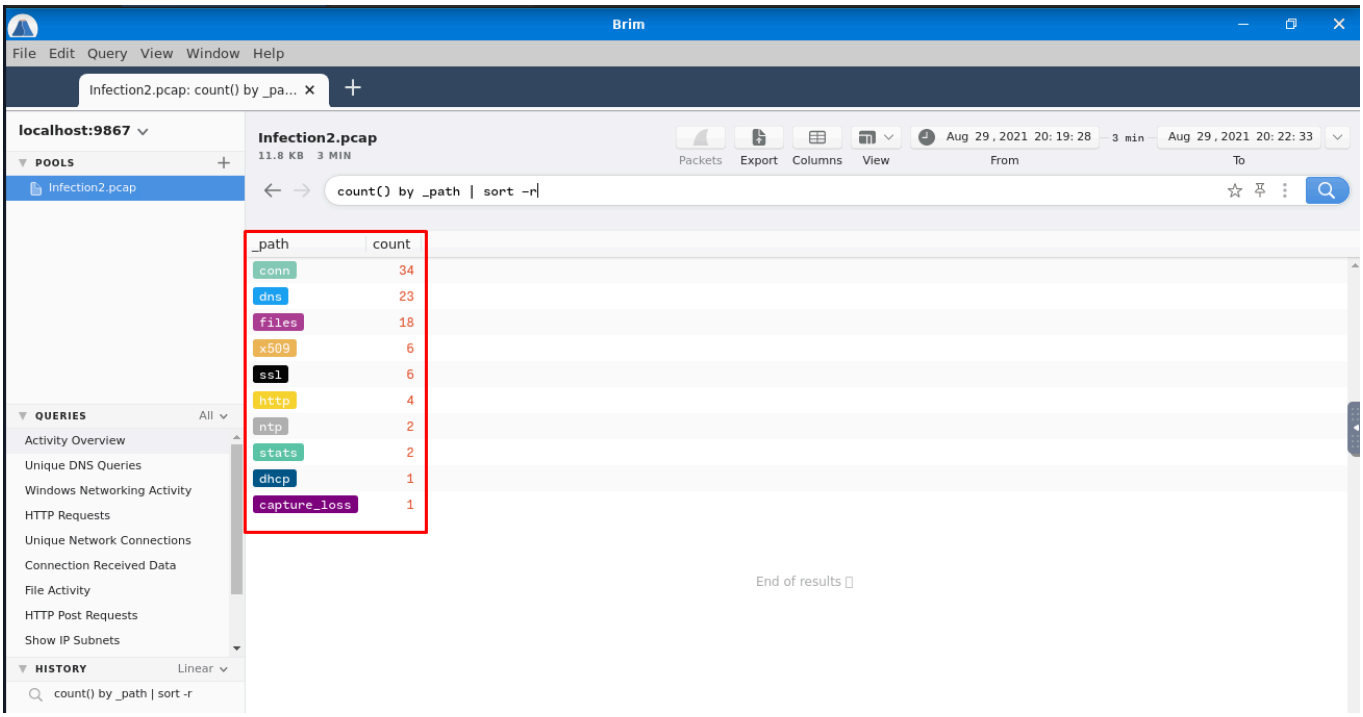
Nota : Para preguntas que requieren múltiples respuestas, sepárelas con una coma.

Responda las preguntas a continuación

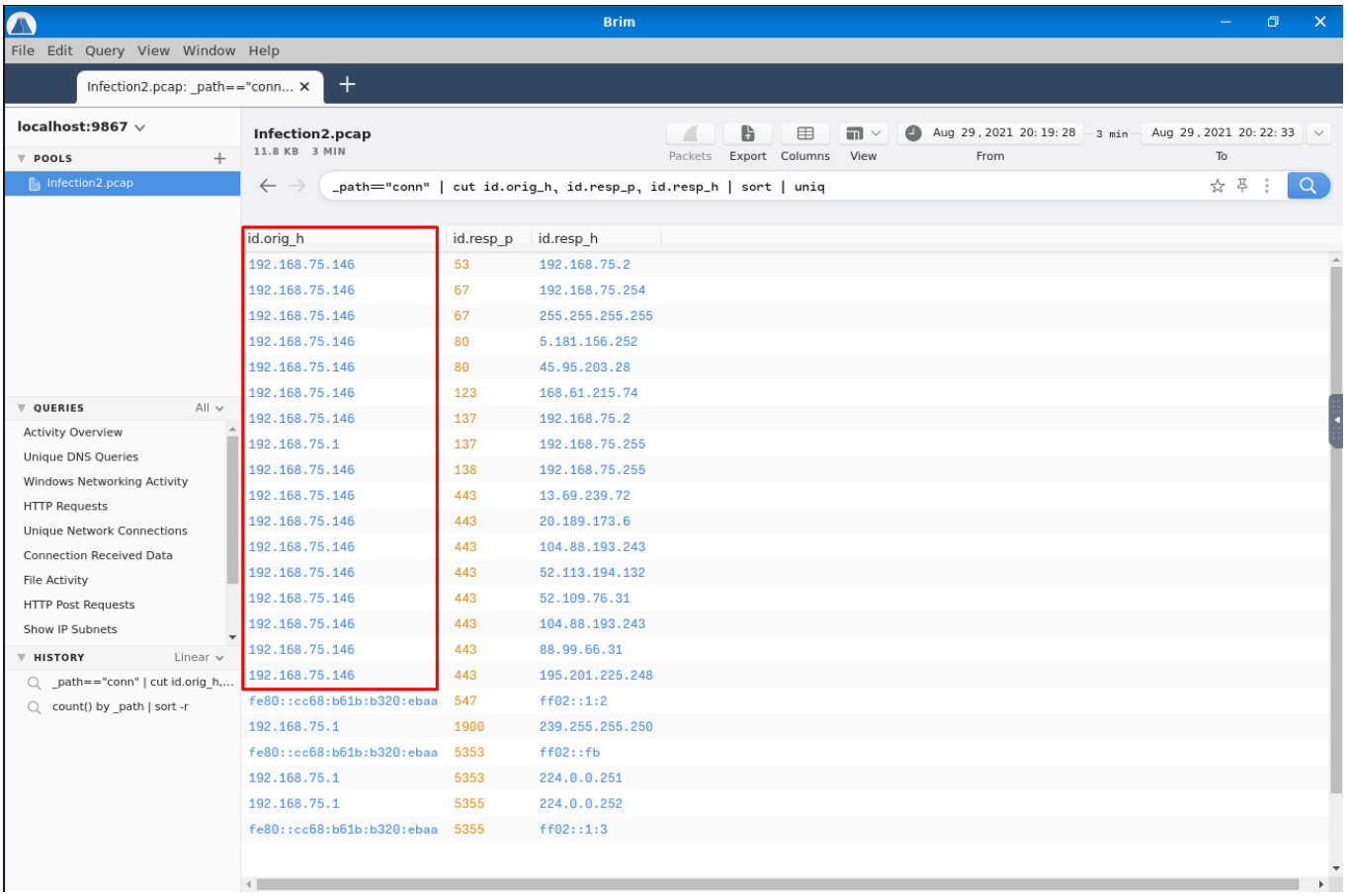




Proporcione la dirección IP de la máquina víctima.

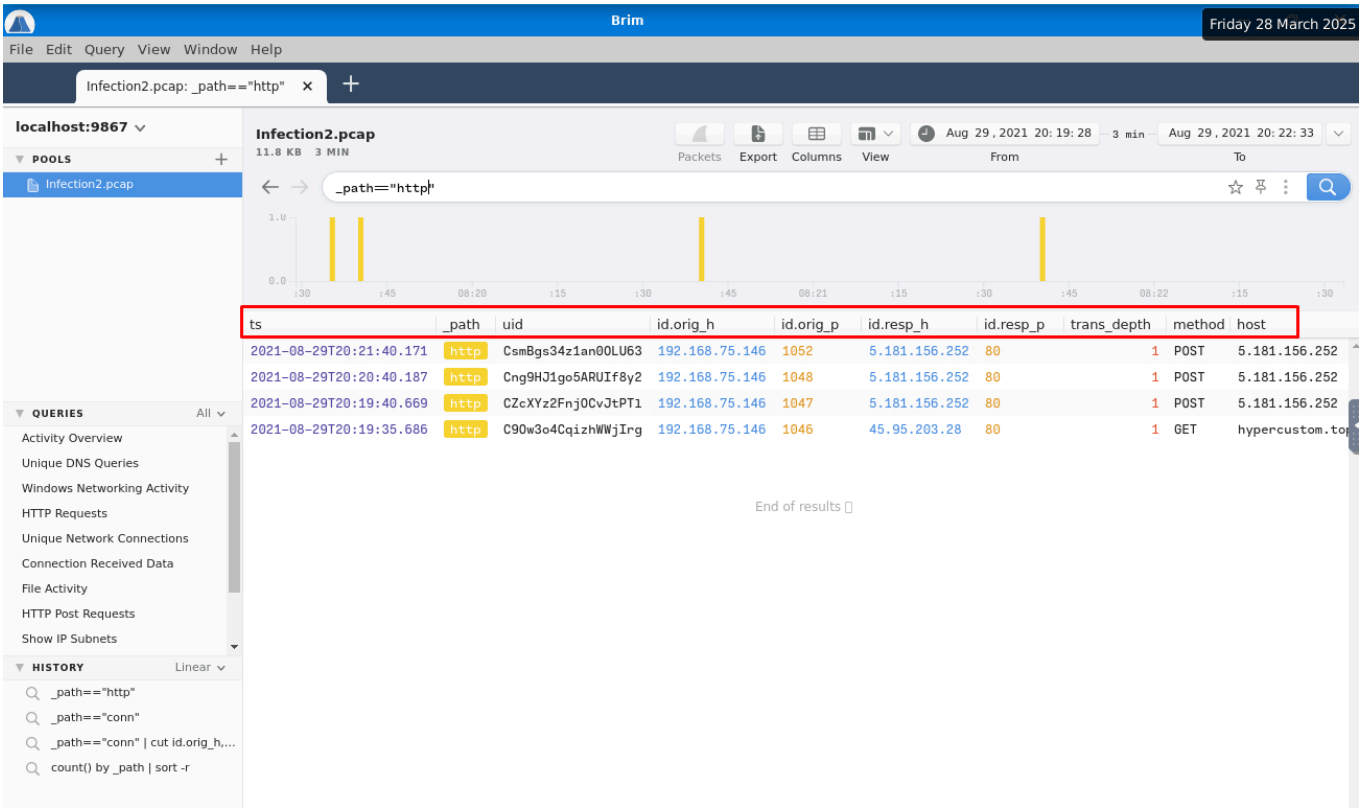



```
_path=="conn" | cut id.orig_h, id.resp_p, id.resp_h | sort | uniq
```

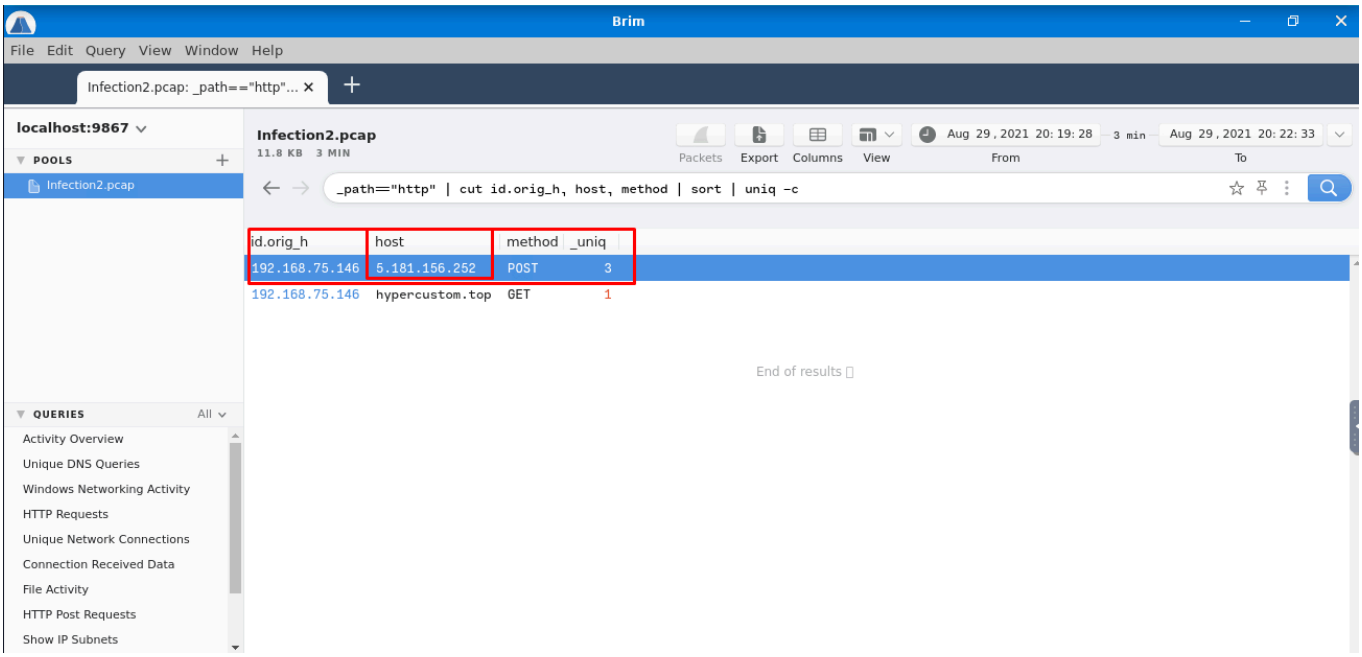


Proporcione la dirección IP a la que la víctima realizó las conexiones POST.

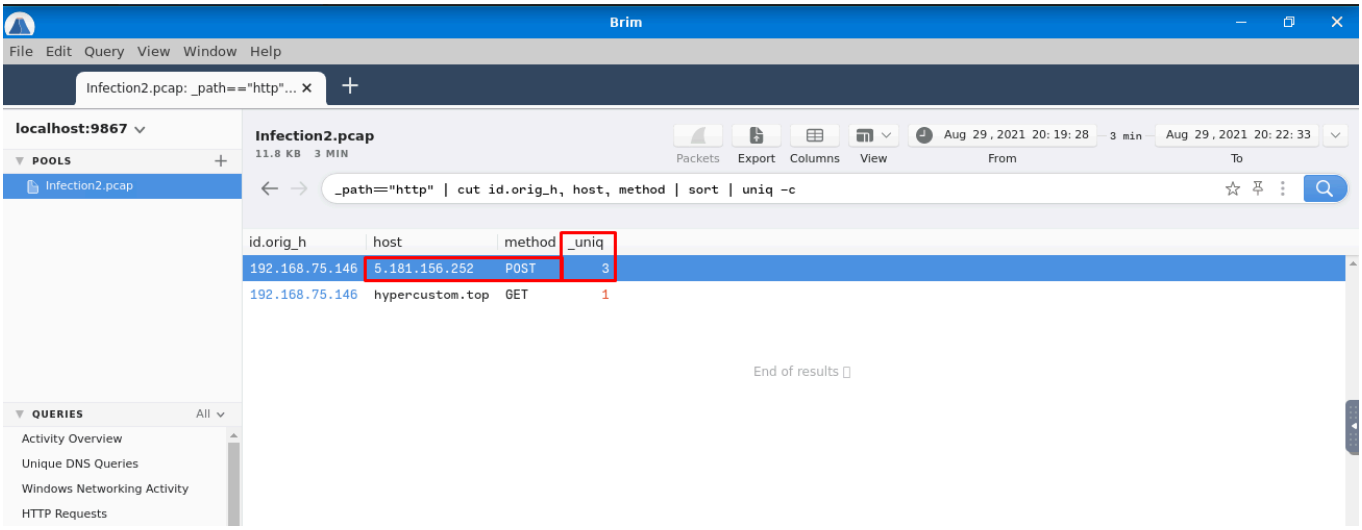
```
_path=="http"
```




```
_path=="http" | cut id.orig_h, host method | sort | uniq -c
```

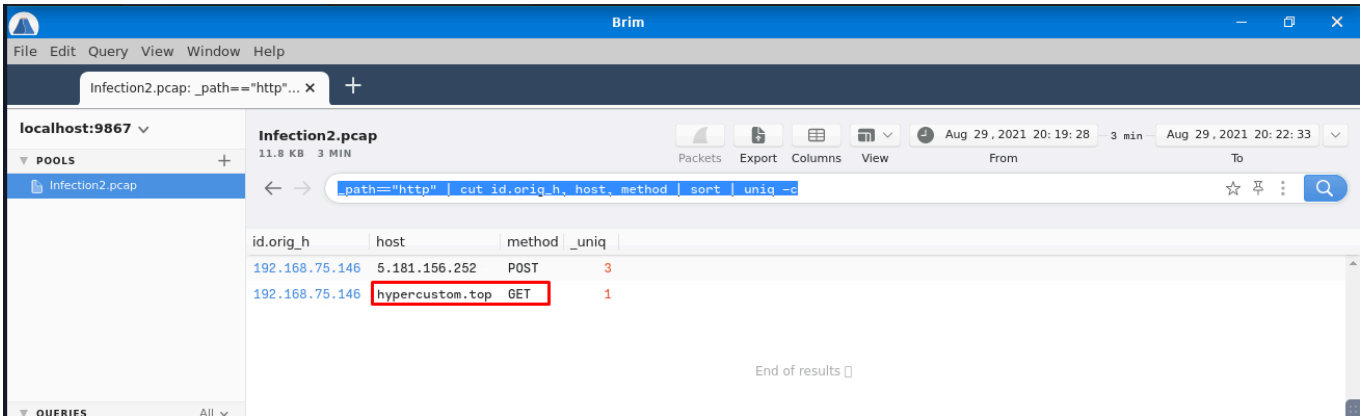


¿Cuántas conexiones POST se realizaron a la dirección IP en la pregunta anterior?



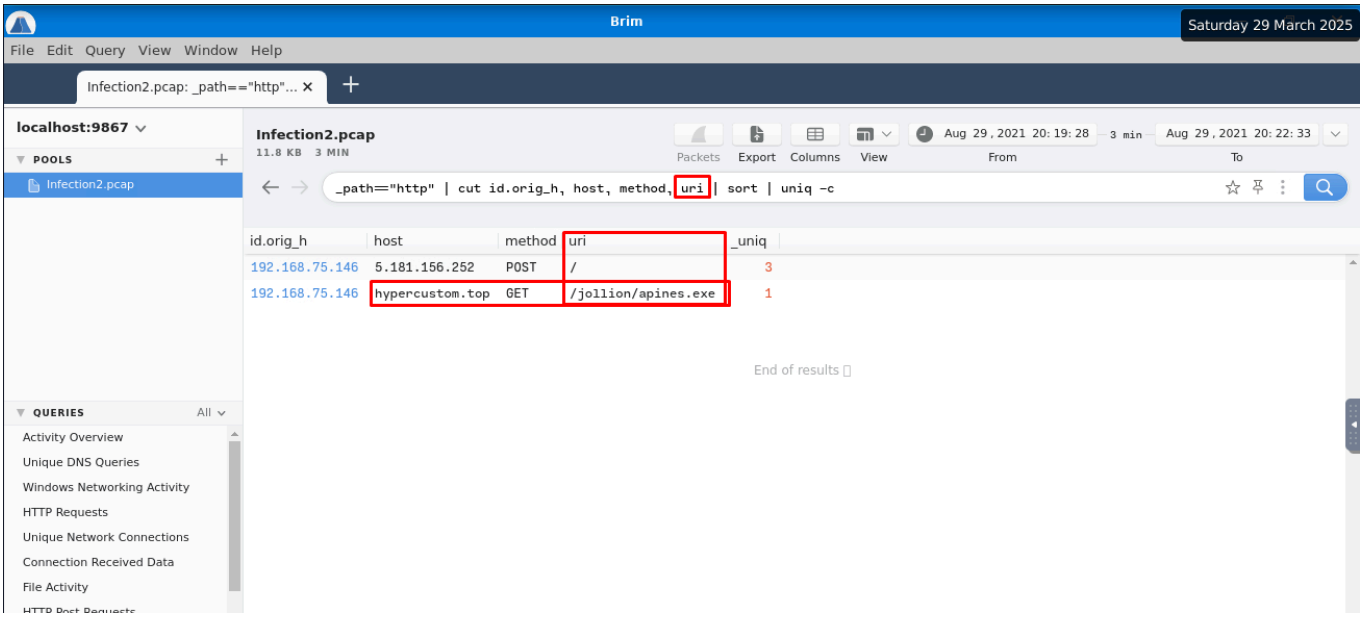
Proporcione el dominio desde donde se descargó el binario.

```
_path=="http" | cut id.orig_h, host method | sort | uniq -c
```



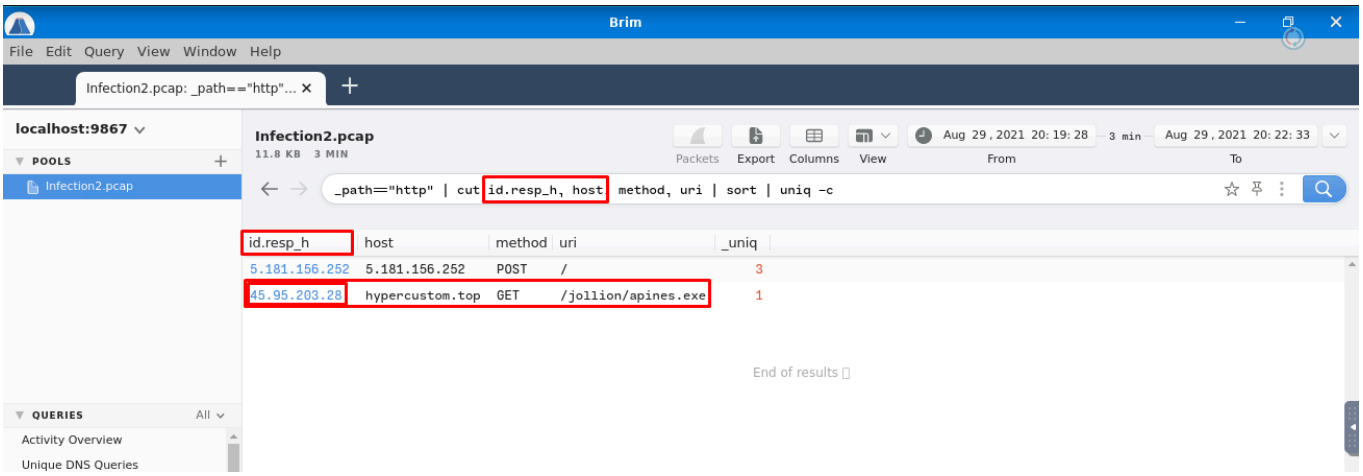
Proporcione el nombre del binario incluido el URI completo.

```
_path=="http" | cut id.orig_h, host method, uri | sort | uniq -c
```

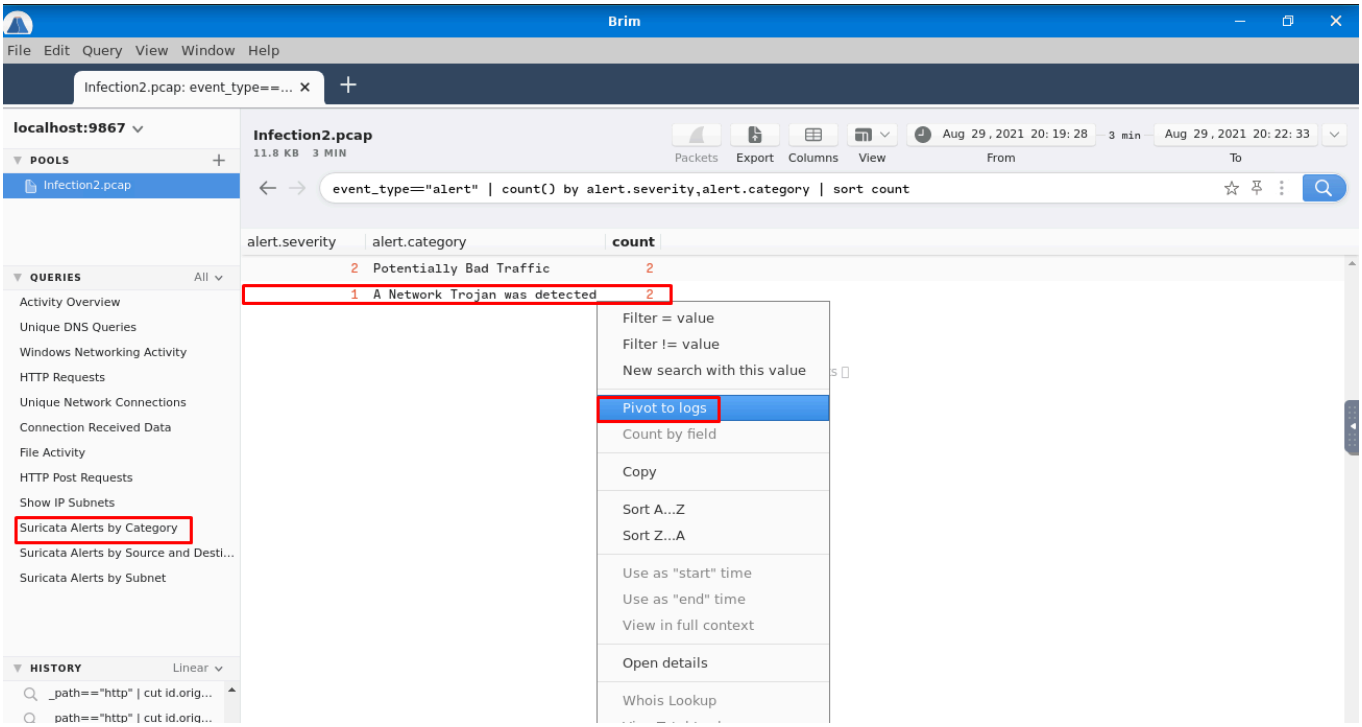


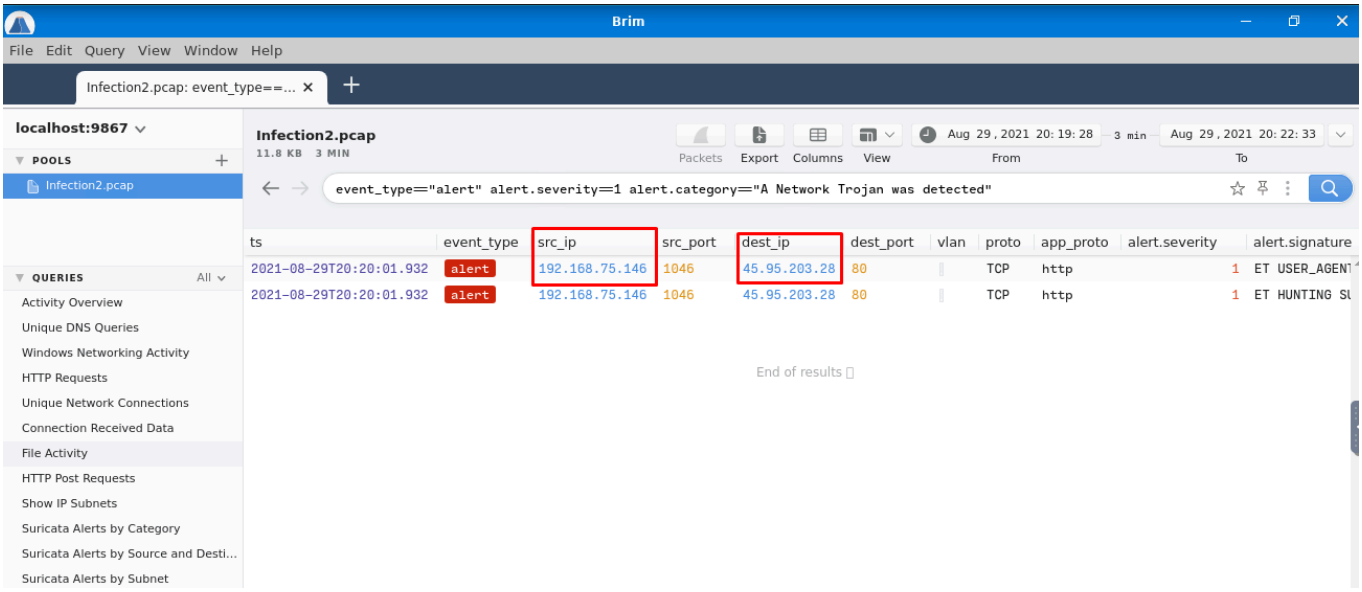
Proporcione la dirección IP del dominio que aloja el binario.

```
_path=="http" | cut id.resp_h, host method, uri | sort | uniq -c
```



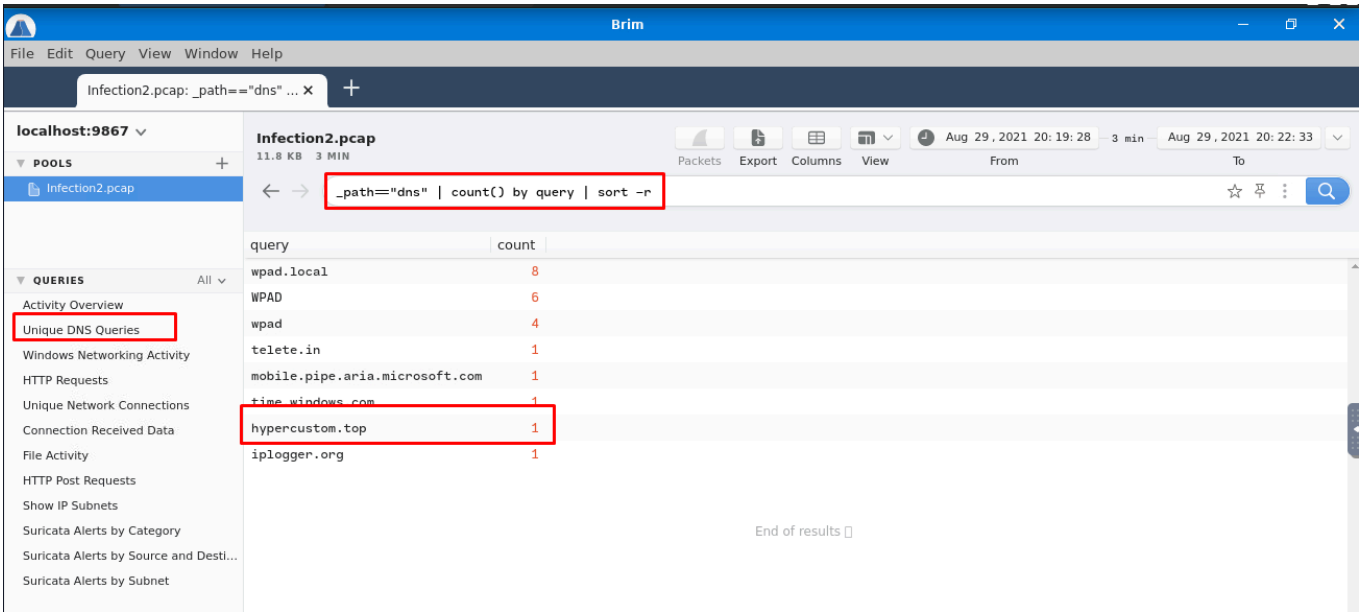
Se recibieron dos alertas de Suricata sobre la detección de un troyano de red.
¿Cuáles eran las direcciones IP de origen y destino?



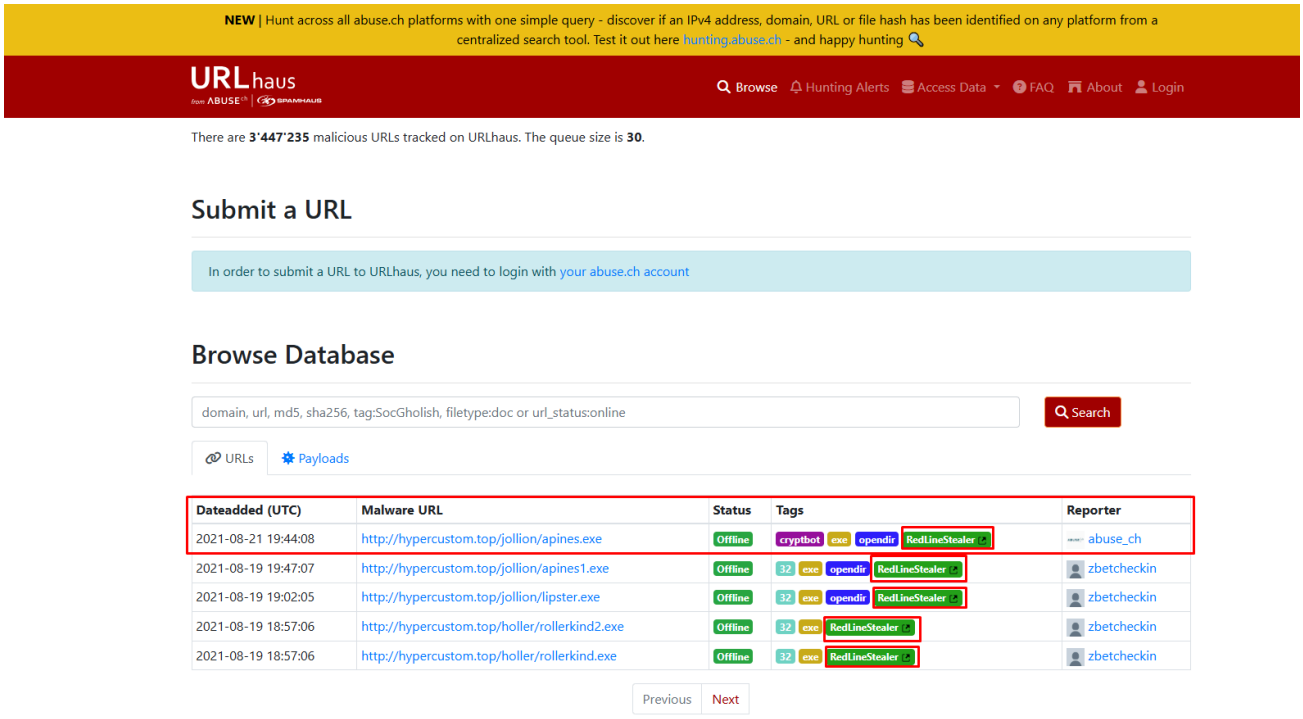


Al observar el dominio .top en las solicitudes HTTP, proporcione el nombre del ladrón (troyano que recopila información de un sistema) involucrado en esta captura

de paquetes utilizando la base de datos URLhaus .



- Informe



NEW | Hunt across all abuse.ch platforms with one simple query - discover if an IPv4 address, domain, URL or file hash has been identified on any platform from a centralized search tool. Test it out here hunting.abuse.ch - and happy hunting

URLhaus from ABUSE.ch | SPAMHAUS

Browse | Hunting Alerts | Access Data | FAQ | About | Login

Payload delivery

The table below documents all payloads that URLhaus retrieved from this particular URL.

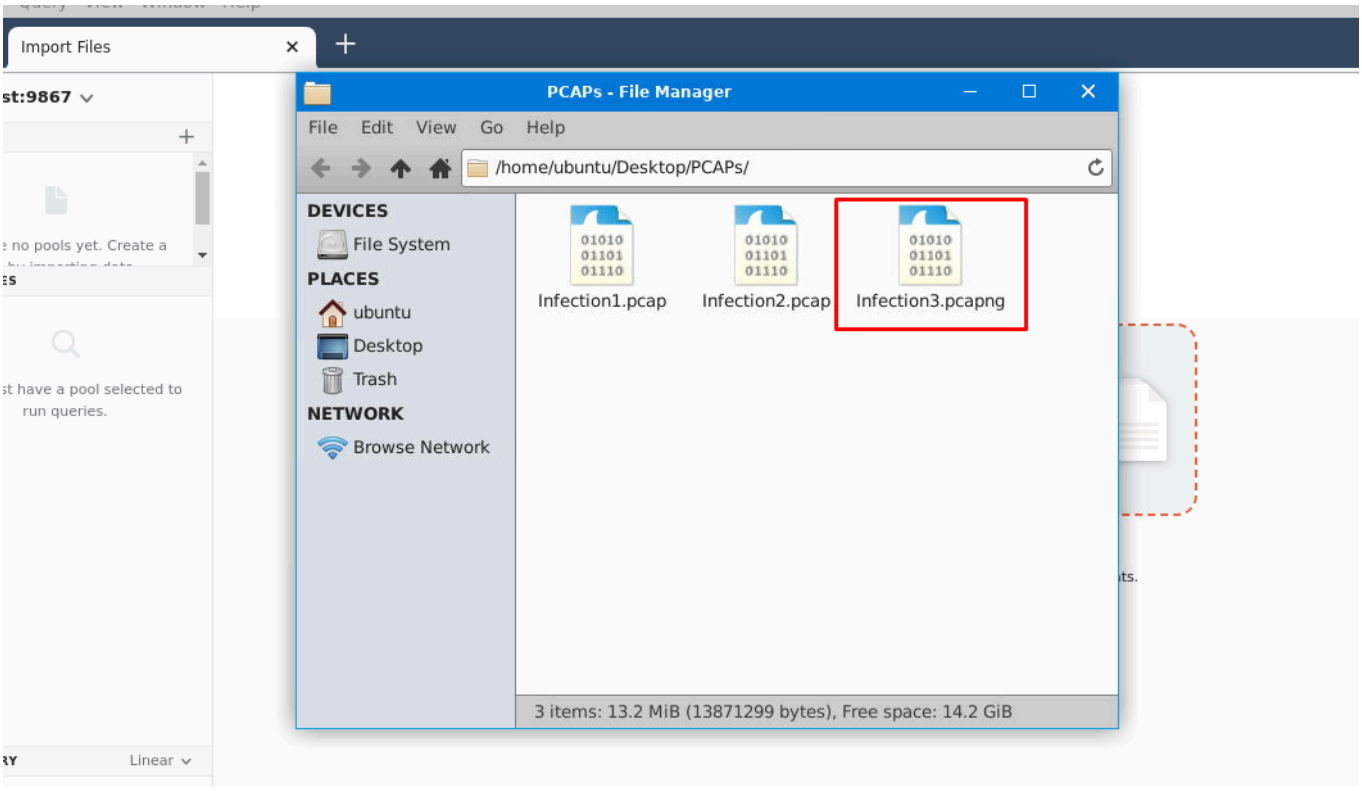
Firstseen	Filename	File Type	Payload (SHA256)	VT	Bazaar	Signature
2021-09-01	n/a	exe	d6007d59009ba577d65fbd1e122dffeec2cd0771c34445cd263e1c2699c21368	n/a		RedLineStealer
2021-08-31	n/a	exe	0648b1ab9937a98ee4aba023573e64bfa6b533f5f15d29c42a0cae43844e026d	n/a		RedLineStealer
2021-08-30	n/a	exe	4da095e0a59cecad3fbfc6fa4f33c00e71edc9fffae041a521552390825732f7	n/a		RedLineStealer
2021-08-29	n/a	exe	82b0161335f3e5103150aedc18f94ea8cecbe00871668de002af752e9cfff9dd	n/a		RedLineStealer
2021-08-28	n/a	exe	0c53ee5bbb0e0624fb8281a8b8850e6f190a97aae229fb9f7bfd0e054988f5e8	n/a		RedLineStealer
2021-08-27	n/a	exe	45a87dd4abe07b6c5f296a9ff716f5224b38dd17e4fd8789819a6c9cbcfda55c	n/a		RedLineStealer
2021-08-26	n/a	exe	25a8d52e15f54022a9efdfaf2c3f8bf94e36dc475d428fcb880c953f1bbaf2c6	n/a		RedLineStealer
2021-08-25	n/a	exe	cce16e3c61f188234483344b7e95f174df1b6708002a6a0ae63ab197745c8090	n/a		RedLineStealer
2021-08-25	n/a	exe	abc87c7b821bb7bba854958ecd20760e63c9365aff2091edc2b8449040263a67	56.72%		CryptBot
2021-08-24	n/a	exe	8b46a621bdeee821d8688fc152c9ebcba9b4d274cba2d4550787a79da83dce04	n/a		RedLineStealer
2021-08-23	n/a	exe	374fe48e2677be33b362d63a5d0f306a29fd7cd0e19302b8bdf4dbbebf46b16b	n/a		RedLineStealer
2021-08-22	n/a	exe	1995d5c5569c14b92d2dd5d849b8a06b5e27280ac6cddaaacc3c00584741a6d0	n/a		RedLineStealer
2021-08-22	n/a	exe	45682ad0823016643d27985aa1d5d0dcffe4be09ddaac5419d0447342d9c6f0b	n/a		RedLineStealer
2021-08-21	n/a	exe	fdcde927665c3300db41bbe2c93005756f7da11fef5557ec5c9c5fa7196bf32c	n/a		RedLineStealer
2021-08-21	n/a	exe	c5749c5654d0508326fdf6a51dd8ec091724a6beb711654650c032ab618e9749	53.73%		RedLineStealer

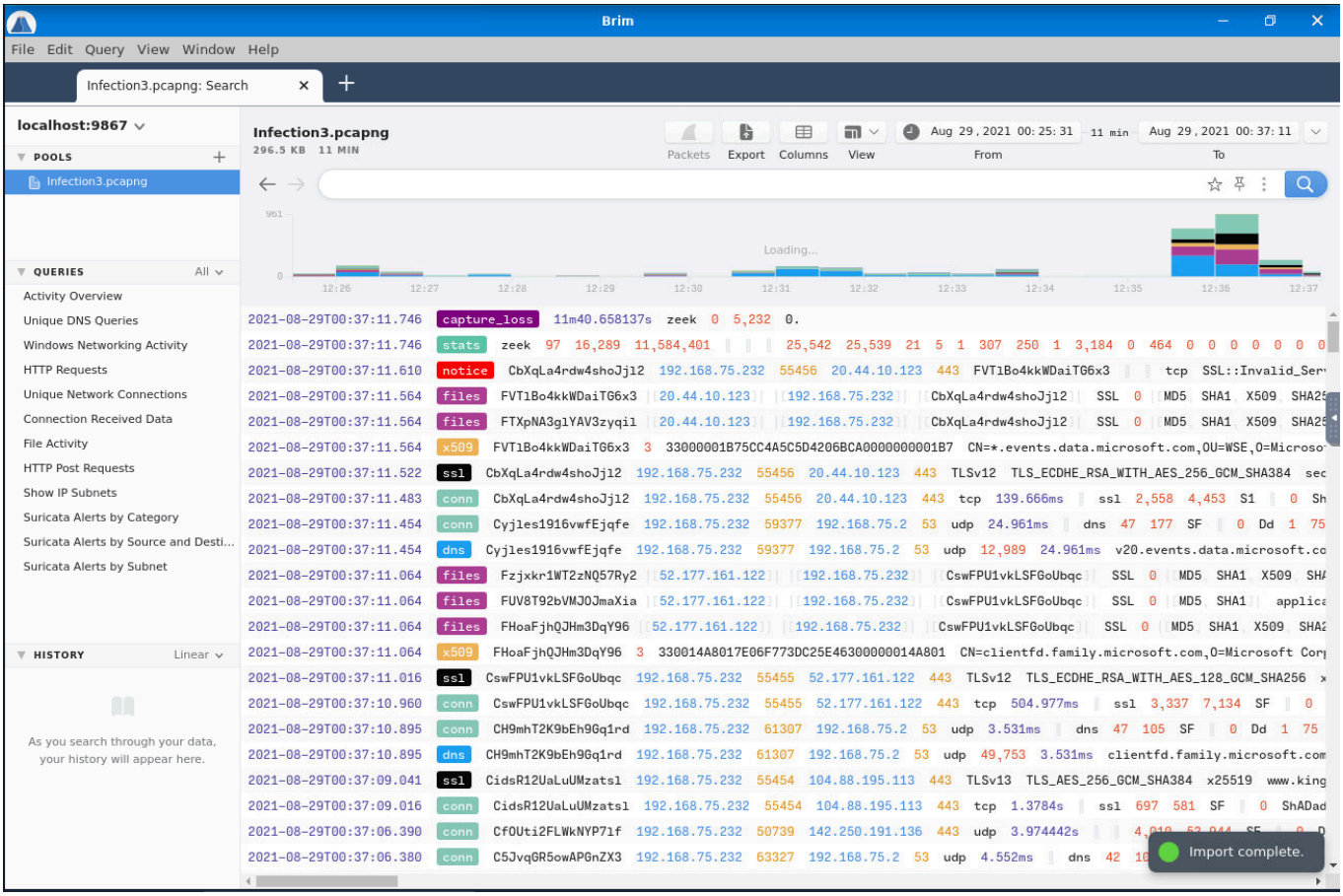
[Infección 3]

Por favor, cargue la captura de paquetes de Infection3 en Brim para investigar el evento de compromiso para la tercera máquina.

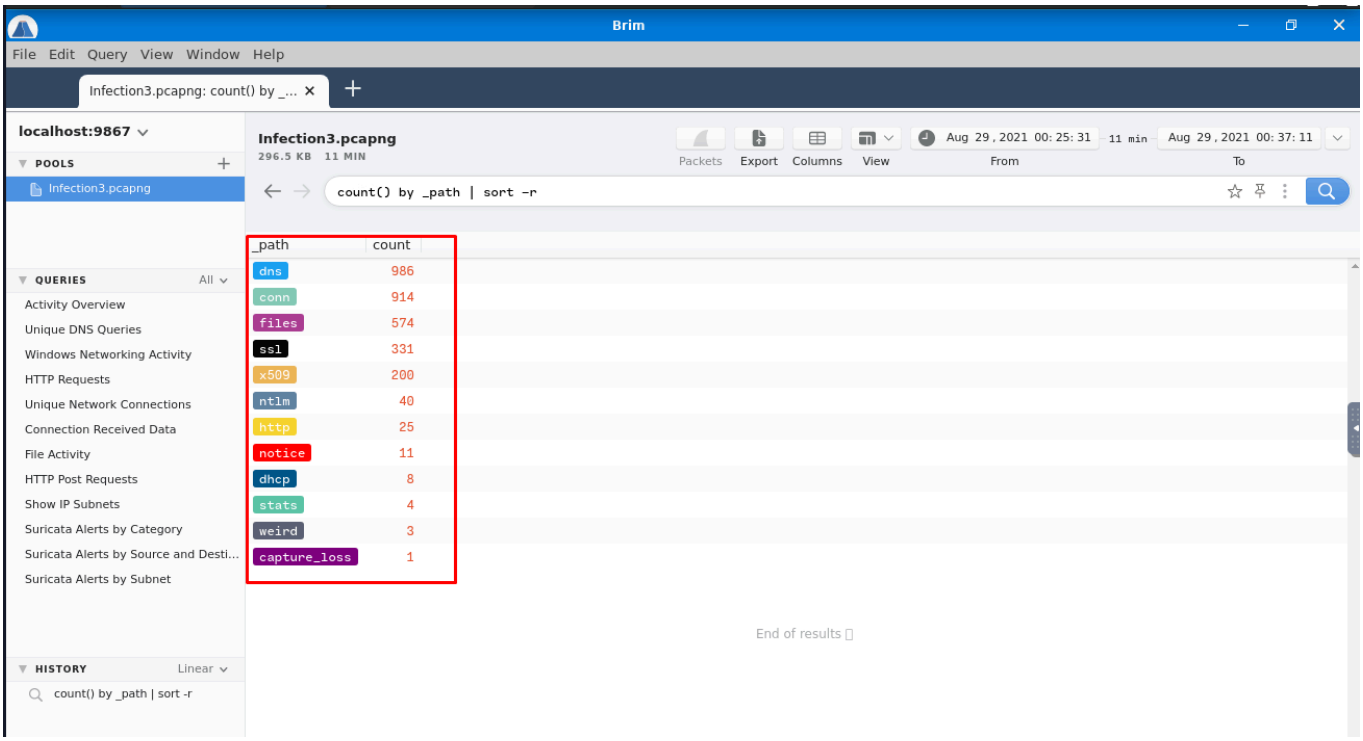
Nota : Para preguntas que requieren múltiples respuestas, sepárelas con una coma.

Responda las preguntas a continuación

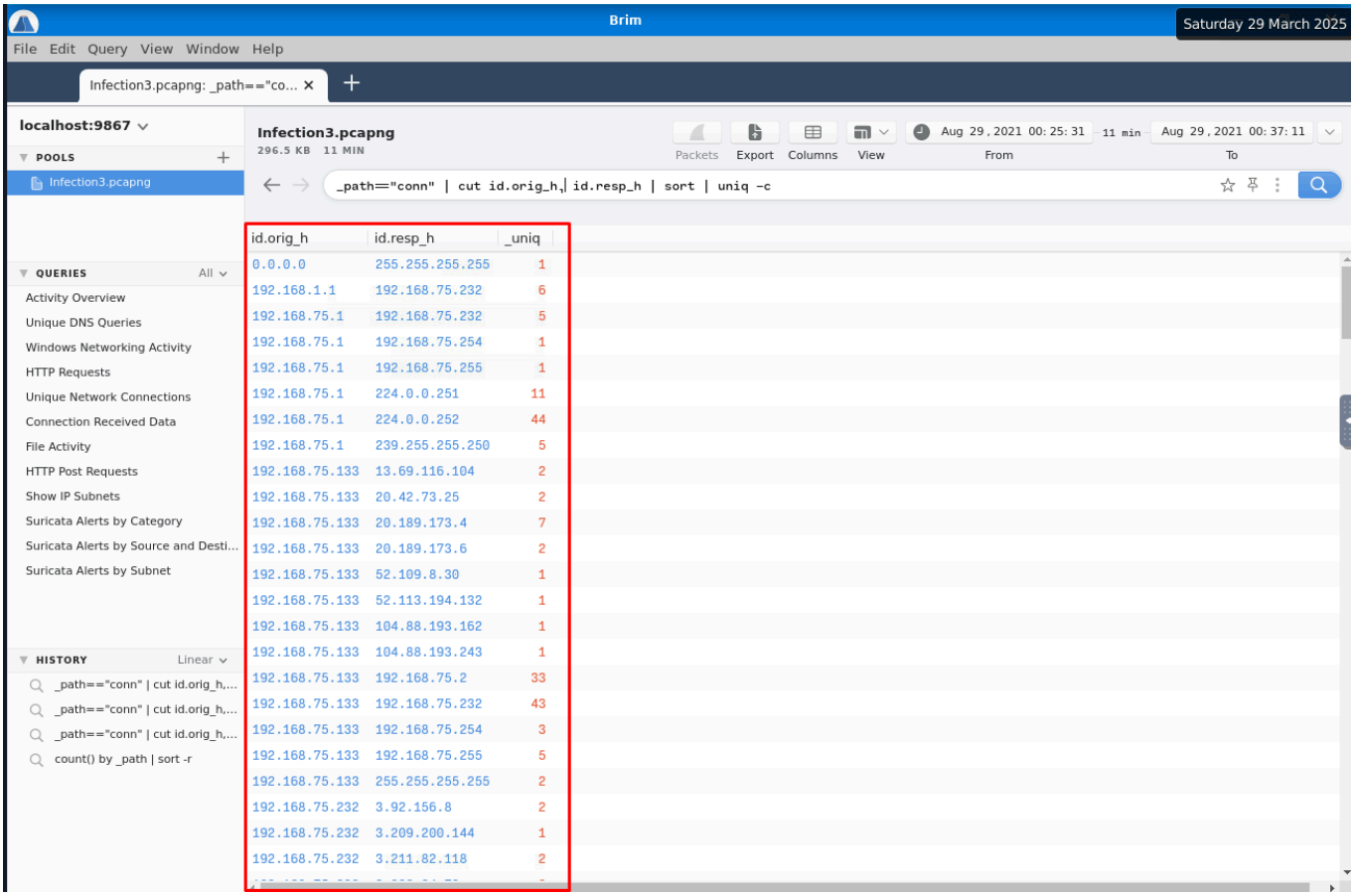




Proporcione la dirección IP de la máquina víctima.




```
_path=="conn" | cut id.orig_h, id.resp_h | sort | uniq -c
```



En esta consulta tuve problemas en la identificación de la IP, por lo que como no

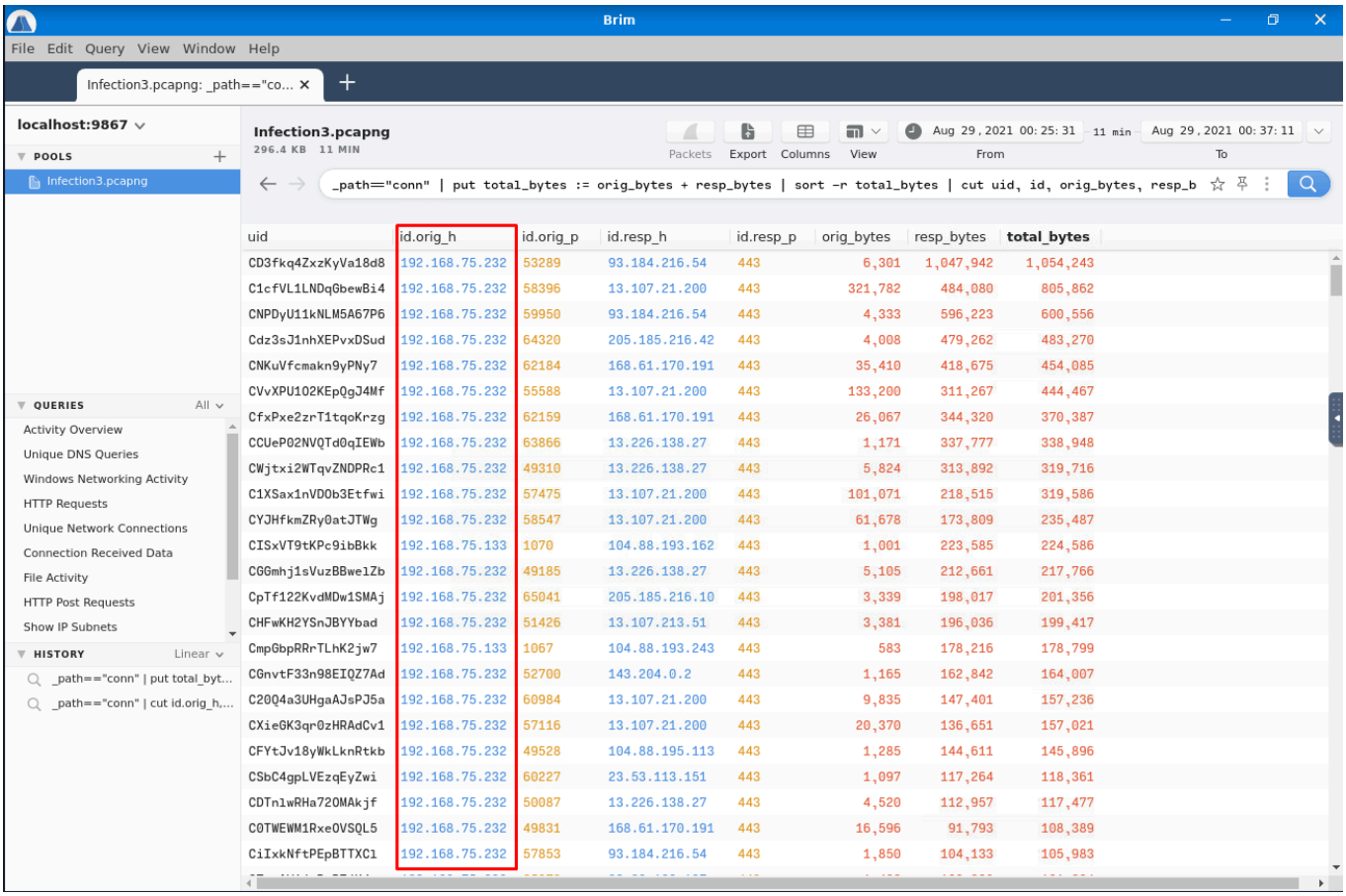
queda claro cual es la ip debido a la cantidad de trafico se puede validar el total de bytes transferidos entre los puntos finales con el siguiente comando

```
_path=="conn" | put total_bytes := orig_bytes + resp_bytes | sort -r total_bytes | cut uid, id, orig_bytes, resp_bytes, total_bytes
```

Explicación de la consulta:

1. `_path=="conn"`
   - Filtra únicamente los registros que pertenecen a la conexión de red (`conn`).
2. `put total_bytes := orig_bytes + resp_bytes`
   - Calcula una nueva columna llamada `total_bytes`, que es la suma de los bytes enviados (`orig_bytes`) y los bytes recibidos (`resp_bytes`).
3. `sort -r total_bytes`
   - Ordena los resultados de manera descendente (`-r` significa "reverse"), mostrando primero las conexiones con mayor tráfico total (`total_bytes`).
4. `cut uid, id, orig_bytes, resp_bytes, total_bytes`
   - Finalmente, selecciona las columnas clave que queremos visualizar en el resultado final:
     - `uid`: Identificador único de la conexión.
     - `id`: Contiene los detalles de identificación de la conexión.
     - `orig_bytes`: Bytes enviados desde la IP de origen.
     - `resp_bytes`: Bytes recibidos por la IP de destino.
     - `total_bytes`: Total de bytes intercambiados (calculado previamente).

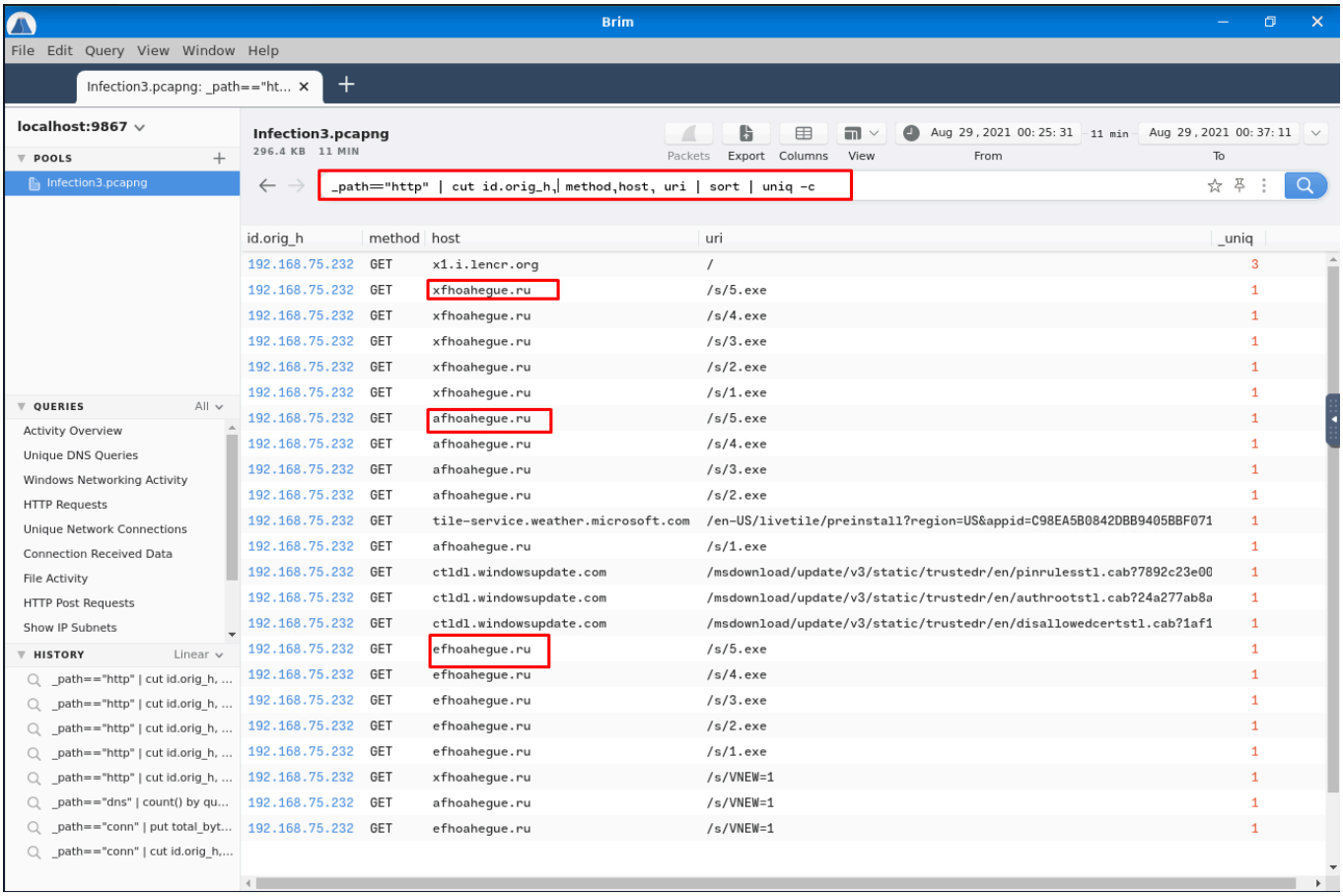
Con esto lo que buscamos es filtrar el log de conexiones (`conn.log`), extrae las direcciones IP, calcula el total de bytes transferidos, ordenar las conexiones por mayor tráfico y mostrar solo la información relevante.



Así ya podemos ver cual es la ip con mayor transferencia de bytes.

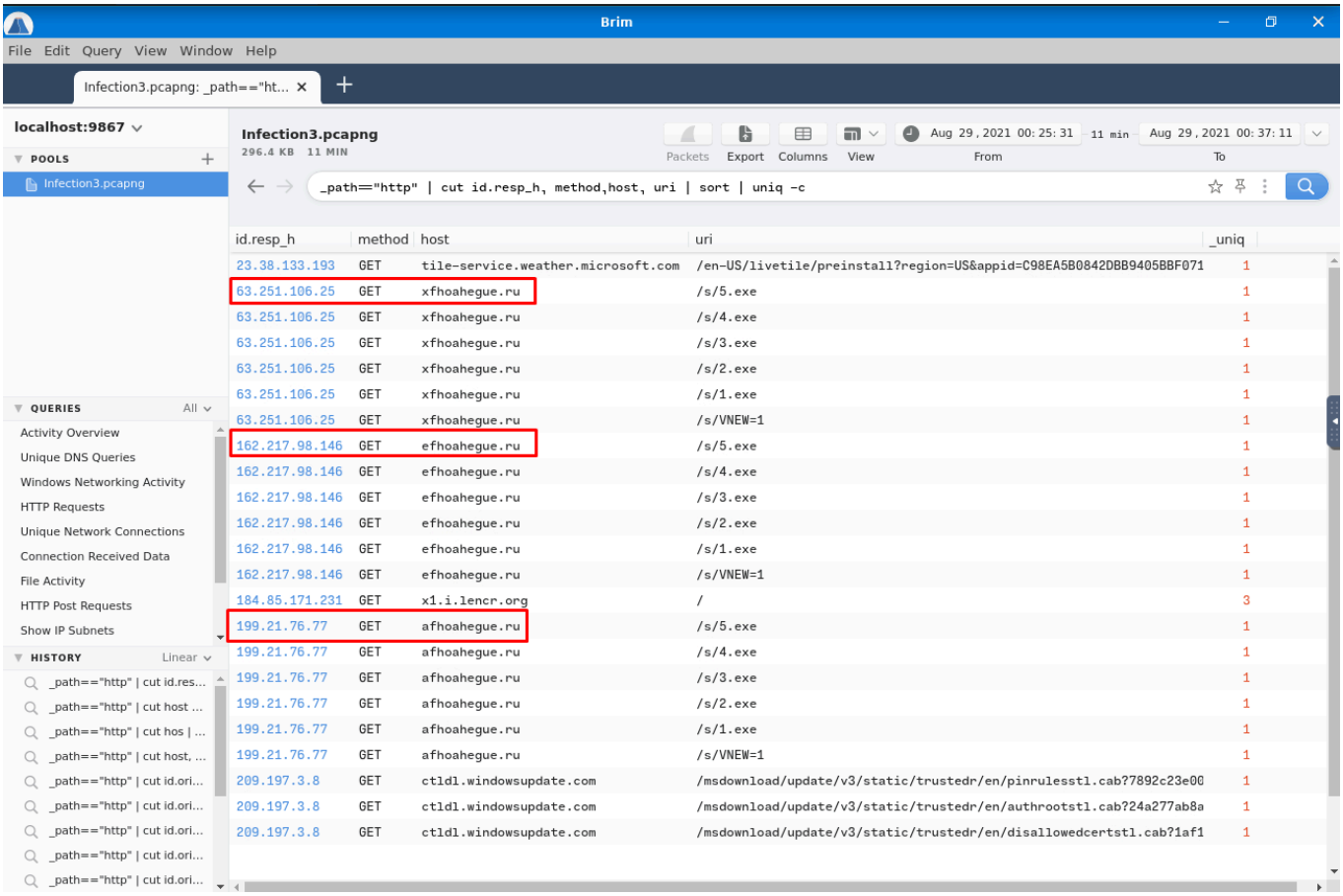
Proporcione tres dominios C2 desde los cuales se descargaron los binarios (comenzando desde el más antiguo hasta el más reciente en la marca de tiempo)

```
_path=="http" | cut id.orig_h, method,host, uri | sort | uniq -c
```



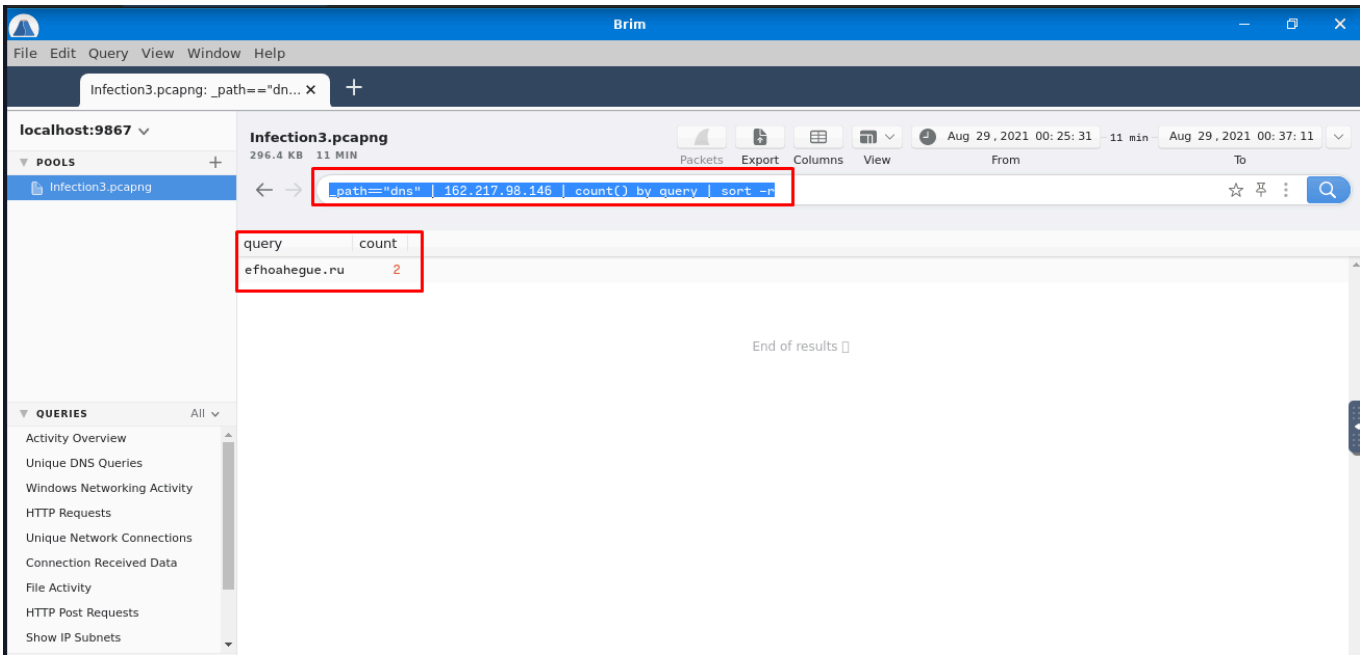
Proporcione las direcciones IP de los tres dominios en la pregunta anterior.

```
_path=="http" | cut id.resp_h, method,host, uri | sort | uniq -c
```



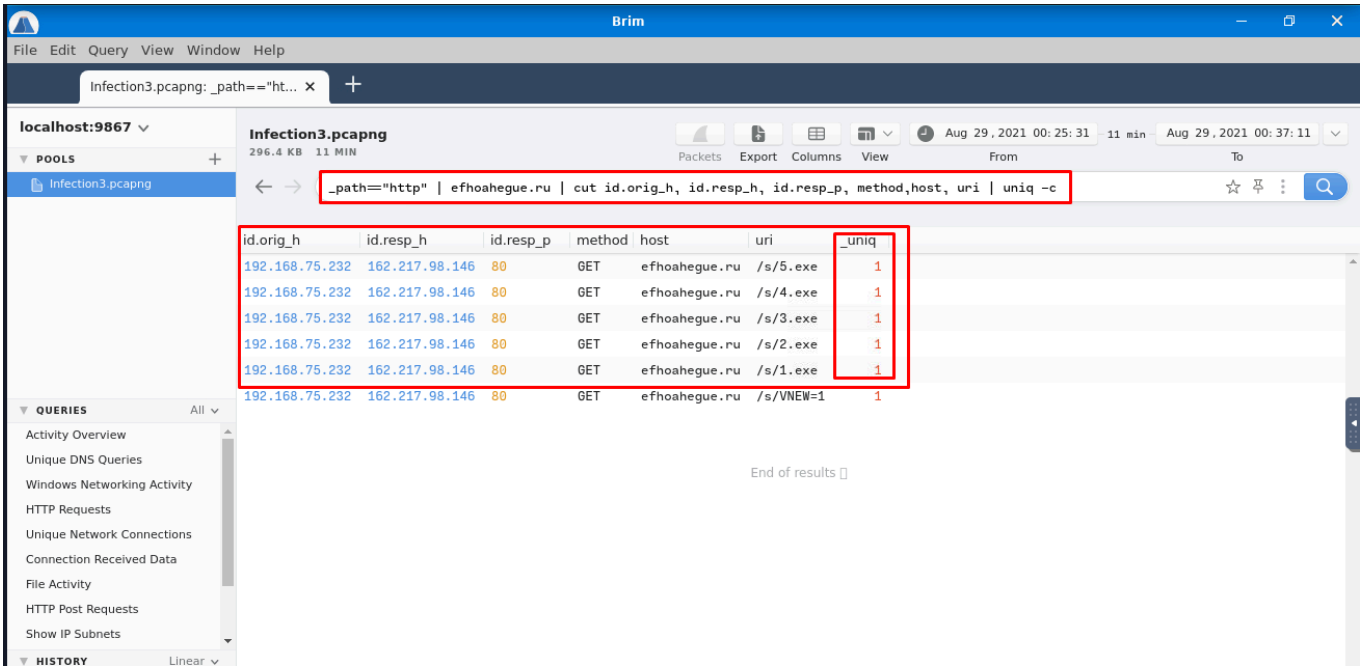
¿Cuántas consultas DNS únicas se realizaron al dominio asociado desde la primera dirección IP de la respuesta anterior?

```
_path=="dns" | 162.217.98.146 | count() by query | sort -r
```



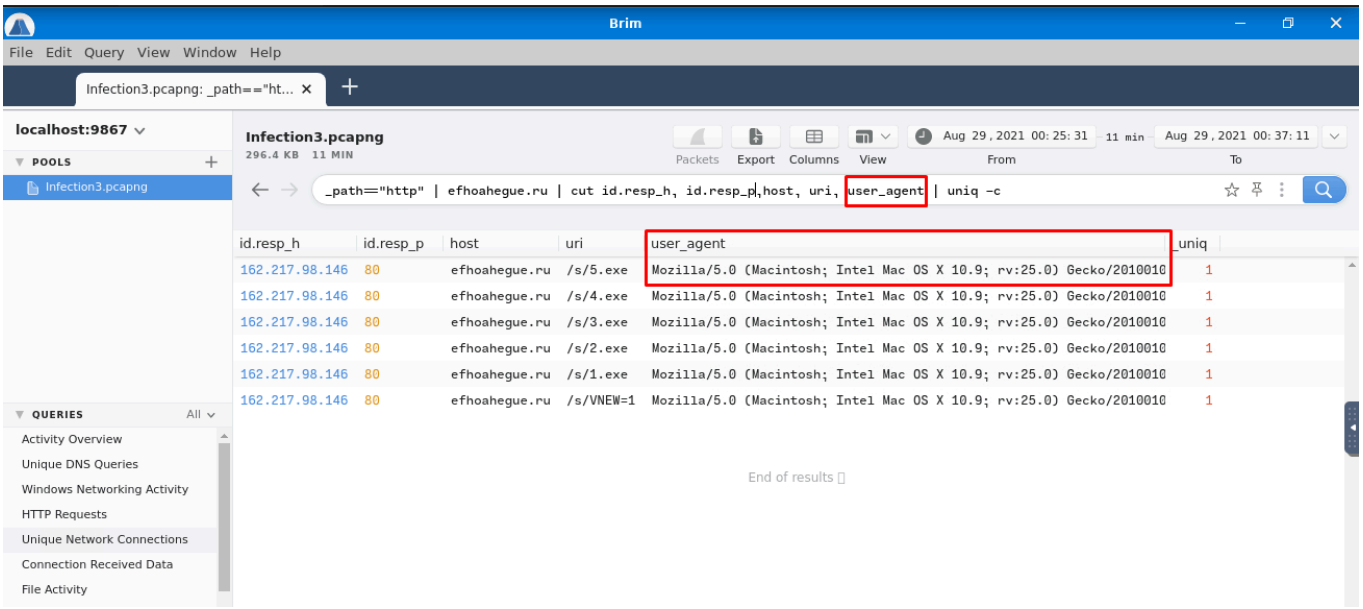
¿Cuántos binarios se descargaron en total del dominio mencionado anteriormente?

```
_path=="http" | efhoahegue.ru | cut id.orig_h, id.resp_h, id.resp_p,
method,host, uri | uniq -c
```



Proporcione el agente de usuario indicado para descargar los binarios.

```
_path=="http" | efhoahegue.ru | cut id.resp_h, id.resp_p,host, uri,
user_agent | uniq -c
```



Proporcione la cantidad de conexiones DNS realizadas en total para esta captura de paquetes.

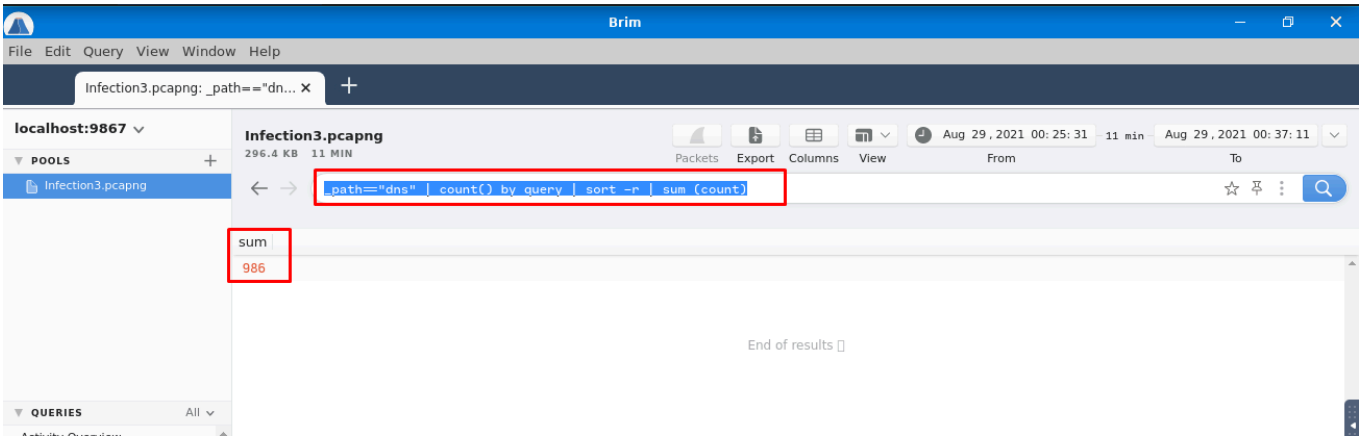
```
_path=="dns" | count() by query | sort -r | sum (count)
```

Explicación de la consulta:

1. `_path=="dns"`
   - Filtra solo los registros que pertenecen a eventos de consultas DNS (Zeek los guarda en `dns.log`).
2. *`count() by query`
   - Cuenta cuántas veces aparece cada consulta DNS (`query`).
   - Agrupa los resultados por el campo `query`, que representa el dominio consultado.
3. `sort -r`
   - Ordena los resultados en orden descendente (`-r`), mostrando primero las consultas más frecuentes.
4. `sum(count)`
   - Suma el total de consultas DNS registradas en el dataset.

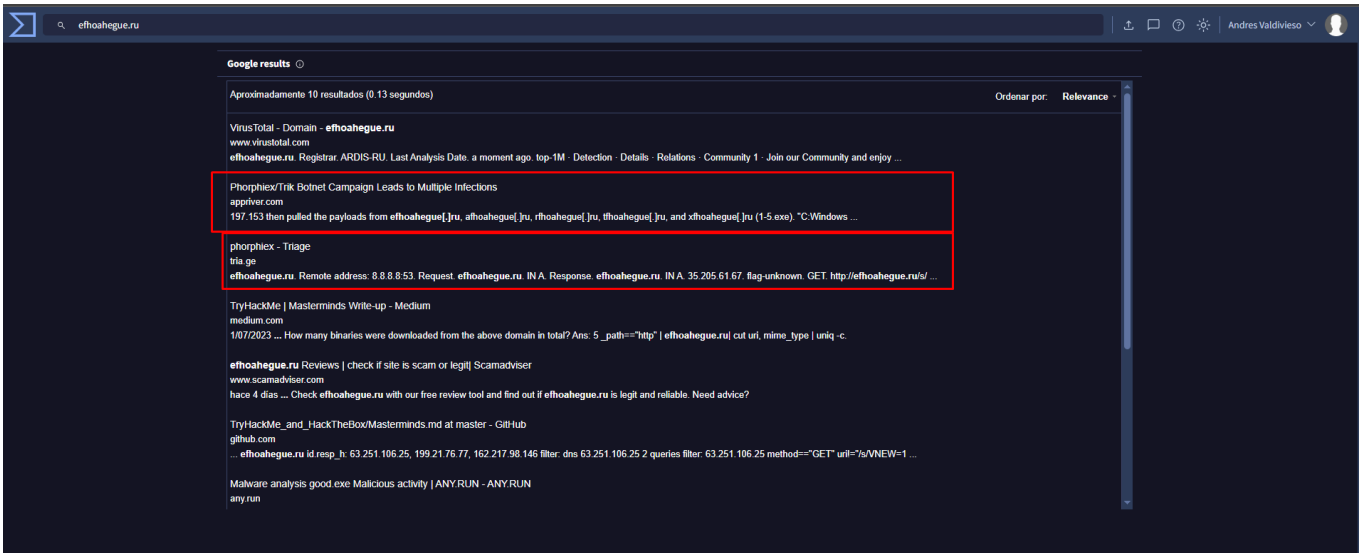
En resumen lo que busco es:

1. Filtra solo los eventos de DNS.
2. Contar cuántas veces se consultó cada dominio.
3. Ordena los dominios más consultados de mayor a menor.
4. Calcula el total de consultas DNS registradas.



Con algunas habilidades de OSINT, proporcione el nombre del gusano utilizando el primer dominio que haya logrado recopilar de la Pregunta 2. (Utilice comillas para las búsquedas de Google, no utilice .ru en su búsqueda y NO interactúe con el dominio directamente).

Lo primero es buscar el dominio lo hice en inicio con virustotal con esto encontramos este Informe en el se puede ver en la parte de los detalles algunos resultados de google:



lo que hice fue validar los dos mas relevantes para entender a que hacia referencia y el otro es el triage del malware:

- Phorphiex/Trik Botnet Campaign Leads to Multiple Infections
- Triage Phorphiex