Obtener IP origen sin terminar conexión SSL

En este escenario:

- Edge router. Es el nginx que recibe la conexión del cliente, y expone a la DMZ los puertos TCP 443 y TCP 80. Si recibe la conexión al puerto TCP 443 lo redirige (capa 4) a “behind edge” usando proxy protocol para poder proporcionarle la IP origen
- Behind edge. Servidor que expone puertos TCP 443 y TCP 80 a 'edge router'. Cuando recibe la conexión TCP 443, por protocolo 'proxy protocol', termina la conexión SSL, y por tanto entrega los certificados (autorfirmados) SSL. Tanto por TCP 80 como por TCP 443 redirige las conexiones a 'php'
- Php. Pinta las cabeceras que recibe

Requisitos:

- Docker instalado
- Docker compose instalados
- El servidor tiene que tener una IP pública
- Nombre DNS (en este ejemplo “example.com”) que apunte a la IP pública del servidor

1. Edge router

1.1. Crear directorio:

```
mkdir edge
```

1.2. Crear 'default.conf':

```
vim edge/default.conf
```

Con el siguiente contenido:

```
proxy_set_header X-Real-IP       $proxy_protocol_addr;
proxy_set_header X-Forwarded-For $remote_addr;
# To avoid 404. Credits:
# https://serverfault.com/a/407983/570054
proxy_set_header Host $host:$server_port;

server {
    listen 80;
    location / {
        proxy_pass http://behind_edge;
    }
        error_page 404 /404.html;
        location = /404.html {
                root /usr/share/nginx/html/;
                internal;
        }
}
```

1.3. Crear 'nginx.conf':

```
vim edge/nginx.conf
```

Con el siguiente contenido:

```
user  nginx;
worker_processes  1;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request"
'
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;

    include /etc/nginx/conf.d/*.conf;

    upstream behind_edge {
        # Docker container named 'behind_edge' sharing same docker network
                server behind_edge:80;
    }
}

stream {

    server {
        listen 443;

        proxy_pass behind_edge_ssl;
        proxy_protocol on;
```

```
    }

        upstream behind_edge_ssl {
        # Docker container named 'behind_edge' sharing same docker network
                server behind_edge:443;
        }

    log_format basic '$remote_addr - [$time_local] '
                        '$status '
                        '';

    access_log  /var/log/nginx/access.log  basic;

}
```

2. Behind edge

2.1. Crear directorio:

```
mkdir behind_edge
```

2.2. Crear 'default.conf':

```
vim behind_edge/default.conf
```

Con el siguiente contenido:

```
proxy_set_header X-Real-IP       $proxy_protocol_addr;
proxy_set_header X-Forwarded-For $remote_addr;
# To avoid 404. Credits:
# https://serverfault.com/a/407983/570054
proxy_set_header Host $host:$server_port;

proxy_set_header X-Custom-IP    $proxy_protocol_addr;
proxy_set_header X-Custom-Port  $proxy_protocol_port;

server {
        listen 80;
        location / {
                proxy_pass http://php;
        }
}

server {
        listen 443 ssl proxy_protocol;
        location / {
                proxy_pass https://php_ssl;
        }
        ssl_certificate /etc/ssl/certs/ssl-cert-snakeoil.pem;
        ssl_certificate_key /etc/ssl/private/ssl-cert-snakeoil.key;
```

```
}
```

2.3. Crear 'nginx.conf':

```
vim behind_edge/nginx.conf
```

Con el siguiente contenido:

```
user  nginx;
worker_processes  1;

error_log  /var/log/nginx/error.log warn;
pid        /var/run/nginx.pid;


events {
    worker_connections  1024;
}


http {
    include       /etc/nginx/mime.types;
    default_type  application/octet-stream;

    log_format  main  '$remote_addr - $remote_user [$time_local] "$request"
'
                      '$status $body_bytes_sent "$http_referer" '
                      '"$http_user_agent" "$http_x_forwarded_for"';

    access_log  /var/log/nginx/access.log  main;

    sendfile        on;
    #tcp_nopush     on;

    keepalive_timeout  65;

    #gzip  on;

    include /etc/nginx/conf.d/*.conf;

        upstream php {
        # Docker container named 'php' sharing same docker network
                server php:80;
        }

        upstream php_ssl {
        # Docker container named 'php' sharing same docker network
                server php:443;
        }
```

```
}
```

2.4. Crear:

```
vim behind_edge/Dockerfile
```

Con el siguiente contenido:

```
FROM nginx

RUN apt-get update && apt-get install -y  \
  ssl-cert
```

3. Php

NOTA: de hecho la parte de SSL en el contenedor PHP ya no es necesaria, pero bueno...

3.1. Crear directorio:

```
mkdir php
```

3.2. Crear:

```
vim php/index.php
```

Con el siguiente contenido:

```
<?php

foreach (getallheaders() as $name => $value) {
    echo "$name: $value<br>\n";
}

?>
```

3.3. Crear:

```
vim php/Dockerfile
```

Con el siguiente contenido:

```
FROM php:7-apache

RUN apt-get update && apt-get install -y  \
  ssl-cert

RUN a2enmod \
  remoteip \
  ssl
```

```
RUN a2ensite default-ssl.conf

COPY index.php /var/www/html/
```

4. Crear:

```
vim docker-compose.yml
```

Con el siguiente contenido:

```
services:

  behind_edge:
    build:
      context: ./behind_edge
    container_name: behind_edge
    volumes:
      - "/home/debian/behind_edge/nginx.conf:/etc/nginx/nginx.conf:ro"
      -
"/home/debian/behind_edge/default.conf:/etc/nginx/conf.d/default.conf:ro"

  edge:
    container_name: edge
    image: nginx
    ports:
      - 80:80
      - 443:443
    volumes:
      - "/home/debian/edge/nginx.conf:/etc/nginx/nginx.conf:ro"
      - "/home/debian/edge/default.conf:/etc/nginx/conf.d/default.conf:ro"

  php:
    build:
      context: ./php
    container_name: php

version: "3.3"
```

Ajustar las rutas a los archivos, en este ejemplo “/home/debian”. Resumen:

```
.
├── behind_edge
│   ├── default.conf
│   ├── Dockerfile
│   └── nginx.conf
├── docker-compose.yml
├── edge
│   ├── default.conf
│   └── nginx.conf
└── php
    ├── Dockerfile
```

```
    └── index.php
```

5. Probar. Ejecutar:

```
curl -s -k https://example.com
```

Resultado esperado similar a:

```
X-Real-IP: 8.8.8.8<br>
X-Forwarded-For: 172.25.0.3<br>
Host: php_ssl<br>
Connection: close<br>
User-Agent: curl/7.74.0<br>
Accept: */*<br>
```

Vemos que la cabecera 'X-Real-IP' contiene la IP pública del cliente que ha enviado la petición, en este caso '8.8.8.8'