Техническое задание

Для организации единой точки входа нужно организовать на домене p.omexai.com инфраструктуру для приема запросов Connect и последующим Upstream на оригинальный прокси сервер.

Установка Squid

Для решения этой задачи было решено на домене p.omexai.com установить прокси - сервер Squid https://github.com/squid-cache/squid который позволит принимать входящие запросы типа Connect.
Сперва — немного подправим конфиг сервера (/etc/squid/squid.conf).
Откроем доступ к прокси для всех (нужно вставить до записи 'http_access deny purge'):

```
acl all src 0.0.0.0/0.0.0.0
http_access allow all
```

На сервере нужно развернуть несколько Squid (приблизительно 5 прокси серверов) на разных портах. Можно развернуть копии Squid на разных портах (каждый со своим конфигом) через контейнеры Docker.

Допустим, для обслуживания провайдера OxyLabs у нас будет Squid на порте 8080. Для BrightData мы выделяем порт 8081. И таким образом создаем 5 прокси-серверов Squid на разных портах.
С 8080 по 8084 у нас порты должны слушаться и для каждого Squid создан свой конфиг файл.

Авторизация

При входящем запросе к нашему прокси - серверу пользователи будут стучаться по виртуальному (рандомному) логину и паролю. То есть для подключения мы выдаем пользователю такие креды для использования (коннекта) прокси:

Host: **p.omexai.com**
Port: **8080**
User: **some_user**
Password: **some_password**

Логин и пароль (виртуальные) при покупке прокси на стороне нашего фронта будут сохраняться в БД, которая посажена рядом с Squid. А нам при запросе на подключение к Squid нужно настроить авторизацию через MySQL. Как это сделать - описано в этой доке:

https://wiki.squid-cache.org/ConfigExamples/Authenticate/Mysql

Указываем основной прокси-сервер
cache_peer proxy.example.com parent 3128 0 no-query default

Опционально, задаем ACL для управления маршрутизацией запросов
acl all_requests src 0.0.0.0/0

Условия для использования upstream прокси
cache_peer_access proxy.example.com allow all_requests
cache_peer_access proxy.example.com deny all

Нужна такая структура таблицы ‘passwd’:

```
CREATE TABLE `passwd` (
    `user` varchar(32) NOT NULL default '', - будет храниться вирт. логин
    `password` varchar(35) NOT NULL default '', - будет храниться вирт. пароль
    `enabled` tinyint(1) NOT NULL default '1',
    `fullname` varchar(60) default NULL,
    `comment` varchar(60) default NULL,
    `proxy_id` int(11) NOT NULL default 0, - будет храниться id proxy в системе

    PRIMARY KEY  (`user`)
```

Запись и обновление БД мы делаем своими силами. Важно, чтобы мы авторизовали запрос к прокси серверу Squid, если запись в БД есть. Если вирт. данные (логин и пароль) присутствуют в БД, то мы разрешаем подключение к прокси.
Вот документация по cache_peer: https://www.squid-cache.org/Doc/config/cache_peer/

Upstream на Parent Proxy

После успешной авторизации нам нужно перенаправить пользователя к нужному прокси. Если пользователь обратился к прокси на порте **8081**, мы точно знаем что его нужно направить на прокси BrightData, данные для upstream которого хранятся в конфиге Squid.

Вот статья как конфигурить парент прокси:
https://www.christianschenk.org/blog/using-a-parent-proxy-with-squid/

В нашем вашем случае нужно добавить опцию login=user:password для соответствующих cache_peer и, возможно, включить TLS.

Что пишет AI по поводу upstream

Для настройки `upstream` на другой прокси-сервер в Squid, вы используете директиву

`cache_peer`. Эта директива позволяет указать другие прокси-серверы, к которым Squid может направлять запросы.
Вот как правильно задать условие для `upstream` на другой прокси-сервер:

```
# Указываем основной прокси-сервер
cache_peer proxy.example.com parent 3128 0 no-query default

# Опционально, задаем ACL для управления маршрутизацией запросов
acl all_requests src 0.0.0.0/0

# Условия для использования upstream прокси
cache_peer_access proxy.example.com allow all_requests
cache_peer_access proxy.example.com deny all
```

Описание директив:

1. cache_peer: Указывает другой прокси-сервер (в данном примере `proxy.example.com`) с портом 3128. Параметры:
    - `parent`: указывает, что это родительский прокси (upstream).
    - `0` и `no-query`: указывают, что Squid не будет запрашивать этот прокси для информации о кэше.
    - `default`: определяет, что этот прокси используется по умолчанию для любых запросов.
2. acl: Задает доступ к потокам. В данном случае `acl all_requests src 0.0.0.0/0` разрешает доступ для всех.
3. cache_peer_access: Определяет, какие запросы будут отправляться на `upstream` прокси. В этом примере все запросы разрешены.

Вы можете создать различные ACL для более сложной маршрутизации.
Чтобы настроить условие `upstream` в Squid в зависимости от логина пользователя, необходимо использовать `external_acl_type`. Это позволит вам создать пользовательскую логику аутентификации и маршрутизации запросов на основе логинов.

Вот как это можно сделать шаг за шагом:
Для настройки Squid с использованием другого прокси-сервера в качестве родительского (`upstream`) с авторизацией по логину и паролю, вы можете использовать параметры `cache_peer` и `proxy_authenticate` в конфигурации Squid. Вот как это сделать шаг за шагом:

Шаги для настройки Squid

1. Создание файла с именами пользователей и паролями: Вам нужно сначала создать файл, содержащий логины и пароли для доступа к родительскому прокси. Этот файл может быть создан с помощью утилиты `htpasswd` из пакета Apache. Например: `htpasswd -c /etc/squid/pwdfile username`

В этом примере `username` — это имя пользователя, и команда предложит вам ввести пароль.

Настройка конфигурации Squid: Откройте файл конфигурации Squid (обычно located в `/etc/squid/squid.conf` или `/etc/squid3/squid.conf`) и добавьте следующие строки:
Указываем родительский прокси с авторизацией

cache_peer proxy.example.com parent 3128 0 no-query

Файл с логинами и паролями для авторизации
cache_peer_authenticate proxy.example.com basic /etc/squid/pwdfile

Определяем ACL для разрешения доступа клиентам
acl all_requests src 0.0.0.0/0

Определяем, какие запросы пойдут на родительский прокси
cache_peer_access proxy.example.com allow all_requests

Объяснение конфигурации:

- `cache_peer`: Указывает адрес и порт родительского прокси. В данном случае это `proxy.example.com` с портом `3128`.
- `cache_peer_authenticate`: Указывает, что для доступа к родительскому прокси требуется базовая аутентификация, а файл `/etc/squid/pwdfile` содержит необходимые учетные данные.
- `acl` и `cache_peer_access`: Определяет, каким запросам разрешено использование родительского прокси. В этом примере разрешены все запросы.

Если вам также нужно управлять различными уровнями доступа в зависимости от пользователей, вы можете использовать ACL для создания более сложных условий. Например:

Определяем ACL для разных групп пользователей
acl admin_users proxy_auth username_admin
acl regular_users proxy_auth username_user

Маршрутизация запросов на основе роли
cache_peer_access proxy.example.com allow admin_users
cache_peer_access proxy.example.com allow regular_users
cache_peer_access proxy.example.com deny all
После внесения всех изменений в конфигурацию не забудьте перезапустить Squid, чтобы изменения вступили в силу.

Чтобы настроить Squid таким образом, чтобы прокси-сервер, к которому он подключается, зависел от пароля (или другого параметра аутентификации), вам нужно

разделить запросы на разные ACL и соответственно направлять их к различным upstream прокси-серверам. Это можно сделать, создав пользовательский внешний скрипт, который будет определять, какой прокси-класс следует использовать в зависимости от переданного пароля или логина.

Пример реализации

Вот как это можно сделать шаг за шагом:

1. Создание файла с паролями
Создайте файл с логинами и паролями для разных прокси-серверов с помощью утилиты `htpasswd`.

Например:
htpasswd -c /etc/squid/proxy1_passwd user1

htpasswd -b /etc/squid/proxy2_passwd user2 password2

2. Написание скрипта для определения прокси
Создайте внешний скрипт, который будет проверять введенный логин и пароль и возвращать соответствующий прокси:
Пример на Python (proxy_selector.py):

```
#!/usr/bin/env python3
import sys

# Логика для определения прокси на основе логина и пароля
def get_proxy(username, password):
    if username == 'user1' and password == 'password1':
        return "proxy1.example.com:3128"
    elif username == 'user2' and password == 'password2':
        return "proxy2.example.com:3128"
    else:
        return None

if __name__ == "__main__":
    for line in sys.stdin:
        username, password = line.strip().split()
        proxy = get_proxy(username, password)
        if proxy:
            print(proxy)
        else:
            print("deny")  # Или любое другое значение, которое вы хотите использовать
```

3. Настройка конфигурации Squid
В файле конфигурации Squid добавьте следующие строки:

```
# Указываем внешнее значение для определения прокси
```

```
external_acl_type proxy_selector %LOGIN %PASSWORD /path/to/proxy_selector.py

# ACL для доступа к разным прокси в зависимости от результата внешнего метода
acl use_proxy1 external proxy_selector user1 password1
acl use_proxy2 external proxy_selector user2 password2

# Определяем родительские прокси
cache_peer proxy1.example.com parent 3128 0 no-query
cache_peer proxy2.example.com parent 3128 0 no-query

# Разрешаем доступ к необходимым прокси в зависимости от аутентификации
cache_peer_access proxy1.example.com allow use_proxy1
cache_peer_access proxy2.example.com allow use_proxy2
cache_peer_access proxy1.example.com deny all
cache_peer_access proxy2.example.com deny all
```

4. Объяснение конфигурации

- `external_acl_type proxy_selector`: Определяет, что Squid будет использовать внешний скрипт для определения прокси на основании логина и пароля.
- ACL: С помощью ACL определяются условия, при которых будут отправляться запросы на разные прокси. В зависимости от введенных логинов и паролей Squid будет перенаправлять запросы к разным родительским прокси.
- `cache_peer`: Указывает адреса двух разных прокси-серверов.

Логгирование и контроль трафика

Для контроля потребления трафика нужно создать возможность читать файл логгирования (stdin, stdout), куда записывается инфа о потребляемых ресурсах конкретного пользователя. Идеально сделать его доступным из веба на чтение этого файла, чтобы мы кроном дергали этот файл и читали записи о трафике.
Если какой то из пользователей потребляет сверх лимита - то нужно создать возможность отключить такого пользователя от прокси.

Установить на веб - сервер программу https://samm.kiev.ua/sqstat/ для контроля потребляемого трафика..