Python Proxy для VPS

Vibe Coding

AI QWEN или DeepSeek позволяют быстро сгенерировать и модернизировать Python код для двух Proxy-серверов, создающих HTTP туннель.

Здесь приведен универсальный Prompt для генерирования кода в AI QWEN, Deepseek, GigaCHAT, Perplexity, ChatGPT и т.д.

Можно с помощью ИИ переделать запрос для генерирования Proxy-сервера на PHP (например).

Описание.

Proxy 1 запускается на ПК пользователя. Браузер настраивается на работу через proxy с IP=10.10.0.1

Proxy 2 запускается на VPS под Ubuntu. По prompt ИИ подскажет вам детально, как установить и запустить. Это займет 30-60 минут.

То, что представлен не python-код, а prompt, имеет свои преимущества.

Вы можете крайне легко модернизировать код, добавлять и менять функции. Именно из этих Proxy были сделаны Proxy-серверы, работающие через неблокируемые пиринговые сети (IPFS).

https://proposed-gray-cattle.myfilebase.com/ipfs/QmekgpzdGroq3Qdzmf9CX7f8i13T5zaPqjmDWxGr1RDt1N

https://proposed-gray-cattle.myfilebase.com/ipfs/QmcFoytGtSx9zvJc5wifYv1kczJSTb4kjsV55pkYqAeRrz

Можно написать proxy, работающие через Bit Torrent или сеть Mythos

PROMPT

PROMPT: Генерация двух Python прокси-серверов с HEX-туннелем и HTTP-маскировкой

0. Общая архитектура

Тебе нужно сгенерировать два Python скрипта:

proxy1.py – работает на локальном ПК (Windows). Принимает соединения от браузера, преобразует трафик в HEX, добавляет текстовый HTTP-префикс, отправляет на VPS.

proxy2.py – работает на VPS (Ubuntu). Удаляет HTTP-префикс, преобразует HEX обратно в бинарные данные, отправляет в интернет.

Обратный трафик (интернет → браузер) идёт без префикса, только HEX.

1. Общие требования к обоим скриптам

1.1. Режимы работы

CONVERT_MODE:

1 – HEX-конвертация (основной режим)

0 – прозрачный туннель (без конвертации)

REPLACE_ON: пока 0 (замена текста не используется, оставлено для будущего)

1.2. Переменные для HTTP-префикса (маскировка)

HTTP_PREFIX (bytes) – текстовый заголовок, например HTTP-запрос. Добавляется в начале каждого пакета от Proxy1 к Proxy2.

HTTP_PREFIX_SIZE (int) – длина HTTP_PREFIX в байтах.

Если 0 – вычисляется автоматически при старте.

Администратор вручную копирует значение из лога Proxy1 в оба скрипта.

SEPARATOR = b"\n---HEX---\n" – уникальный разделитель между префиксом и HEX-строкой.

1.3. Таймауты и лимиты

IDLE_TIMEOUT = 60 – таймаут бездействия (сек)

CONNECT_TIMEOUT = 10 – таймаут подключения

BUFFER_LIMIT = 65536 – лимит буфера (байт)

MAX_CLIENTS = 500

1.4. Общие функции

CONVERTER(data: bytes) -> bytes

Преобразует бинарные данные в HEX-строку + b"\n"

DECONVERTER(line: bytes) -> bytes

Преобразует HEX-строку обратно в бинарные данные

Проверяет чётность длины и валидность HEX-символов

Возвращает b'' при ошибке

extract_host_from_request(data: bytes) -> str

Извлекает домен из HTTP/HTTPS запроса для логирования

Использует regex: CONNECT, Host:, GET/POST

2. Специфические требования для Proxy1 (локальный ПК)

2.1. Конфигурационные переменные

python

```
LISTEN_IP = "10.10.0.1"      # IP для браузера
LISTEN_PORT = 8443           # Порт для браузера
SOURCE_IP = None             # Исходящий IP к VPS (None = авто)
TARGET_HOST = "153.181.x.x"   # Публичный IP VPS
TARGET_PORT = 8080           # Порт VPS
HTTP_PREFIX = b"GET /wiki/ HTTP/1.1\r\nHost: example.com\r\n\r\n"
HTTP_PREFIX_SIZE = 0
```

2.2. Функция ADDTEXT1(hex_data: bytes) -> bytes

Возвращает HTTP_PREFIX + SEPARATOR + hex_data

2.3. Логика работы Proxy1 (asyncio)

connect_to_target()

Создаёт TCP-сокет, включает TCP_NODELAY

Привязывается к SOURCE_IP, если указан

Подключается к TARGET_HOST:TARGET_PORT

Возвращает (reader, writer)

handle_client(client_reader, client_writer)

Принимает соединение от браузера

Устанавливает соединение с Proxy2

Запускает две асинхронные задачи:

client_to_server() (браузер → Proxy2)

Читает данные от браузера

Вызывает TO_SERVER (CONVERTER)

Вызывает ADDTEXT1

Отправляет в Proxy2

server_to_client() (Proxy2 → браузер) – обратный канал

Читает данные от Proxy2 (уже HEX, без префикса)

Накопление буфера, разбивка по \n

Вызывает TO_CLIENT (DECONVERTER)

Отправляет браузеру

2.4. Логирование

При старте выводится HTTP_PREFIX_SIZE

Логируются новые соединения, запросы к сайтам, ошибки

3. Специфические требования для Proxy2 (VPS Ubuntu)

3.1. Конфигурационные переменные

python

```
LISTEN_IP = "53.181.x.x"      # Внешний IP VPS
LISTEN_PORT = 8080            # Порт для приёма от Proxy1
EXTERNAL_IP = "53.181.x.x"    # IP для выхода в интернет
```

EXTERNAL_PORT = 0 # 0 = динамический

HTTP_PREFIX = ... # ТОТ ЖЕ, что в Proxy1

HTTP_PREFIX_SIZE = 0

3.2. Функция REMOVETEXT2(packet: bytes) -> bytes

Проверяет длину пакета

Сравнивает первые HTTP_PREFIX_SIZE байт с HTTP_PREFIX

Сравнивает следующие len(SEPARATOR) байт с SEPARATOR

При несовпадении – логирует и возвращает b''

Возвращает оставшуюся часть (HEX-строка)

3.3. Логика работы Proxy2 (threading)

tunnel_to_internet(client_sock, internet_sock, ...)

Накопление буфера

Поиск SEPARATOR

Проверка позиции разделителя (должна равняться HTTP_PREFIX_SIZE)

Вызов REMOVETEXT2

Вызов FROM_FIRST_PROXY (HEX_TO_BINARY)

Отправка в интернет

tunnel_to_first_proxy(internet_sock, client_sock, ...)

Получение ответа из интернета

Вызов TO_FIRST_PROXY (BINARY_TO_HEX)

Отправка в Proxy1 (только HEX, без префикса)

handle_client(client_sock, client_addr)

Приём первого пакета от Proxy1

Проверка разделителя и префикса

Разбор CONNECT/HTTP запроса

Подключение к целевому сайту в интернет

Запуск двух тредов (туннелей)

3.4. Обработка CONNECT

Для HTTPS (CONNECT) отправляется ответ 200 Connection established

Сохраняется early_data_buffer – остаток данных после заголовка

4. Технологии и библиотеки

asyncio (Proxy1) – асинхронная работа с большим числом соединений

threading (Proxy2) – классическая многопоточность для простоты на VPS

socket – низкоуровневый TCP

binascii – преобразование HEX

re – извлечение хостов из HTTP

signal (Proxy2) – корректное завершение

5. Ограничения и обработка ошибок

5.1. Ограничения

Префикс должен быть строго одинаковым в обоих скриптах

Разделитель не должен встречаться внутри префикса или HEX-данных

При фрагментации TCP возможно накопление буфера

Не поддерживается HTTP/2 (только HTTP/1.1)

Нет аутентификации между прокси

5.2. Обработка ошибок

Несовпадение префикса → закрытие соединения, лог

Разделитель не на своей позиции → закрытие

Некорректный HEX → b'' (пакет теряется, соединение не закрывается)

Таймауты → логирование, продолжение работы

Переполнение буфера → сброс

Ошибки подключения к интернету → 502 Bad Gateway

6. Инструкция по установке, настройке и использованию

6.1. На ПК с Windows

Установить Python 3.8+ с python.org

Скачать proxy1.py

Отредактировать переменные:

TARGET_HOST = IP вашего VPS

HTTP_PREFIX (при желании)

Запустить:

cmd

python proxy1.py

Настроить браузер на ПК под Windows:

HTTP/HTTPS прокси: 10.10.0.1:8443

Без SOCKS

6.2. На VPS с Ubuntu

Подключиться по SSH

Установить Python:

bash

sudo apt update && sudo apt install python3 -y

Скопировать proxy2.py

Открыть порт 8080 (если нужно):

bash

sudo ufw allow 8080/tcp

Запустить:

bash

python3 proxy2.py

6.3. Синхронизация HTTP_PREFIX_SIZE

Запустить proxy1.py на Windows

В логе найти строку с длиной заголовка:

Text (например)

HTTP_PREFIX длина: 347 байт

Скопировать число (например, 347)

В ОБОИХ скриптах установить:

python

HTTP_PREFIX_SIZE = 347

Перезапустить оба прокси

7. Требования к коду

Должен быть самодокументированным (комментарии на русском или английском)

Обрабатывать исключения без падения всего сервера

Не использовать сторонние библиотеки (только стандартный Python)

Работать под Windows и Linux без изменений

Выводить в консоль: статус, соединения, ошибки, количество активных клиентов

8. Пример использования

После запуска обоих прокси:

Браузер → 10.10.0.1:8443

Proxy1 добавляет префикс, конвертирует в HEX

VPS Proxy2 удаляет префикс, декодирует HEX

Трафик уходит в интернет с IP VPS

9. Диагностика проблем

Проблема	Возможная причина	Решение
Proxy2 не слушает порт	Порт занят или фаервол	sudo lsof -i :8080, ufw allow

Несовпадение префикса	Разные HTTP_PREFIX	Синхронизировать текст
Ошибка декодирования HEX	Повреждённый пакет	Проверить разделитель
Нет соединения с VPS	Неправильный IP/порт	telnet <VPS_IP> 8080

10. Возможные улучшения (на будущее)

Сжатие трафика перед HEX

Случайный выбор префикса из пула

Аутентификация по токену

Логирование в файл

systemd-сервис для VPS

Конец PROMPT

IPFS Tunnels Proxy

https://proposed-gray-cattle.myfilebase.com/ipfs/QmeU3EYvwA3HDnRwyB2xUAfD39WQgnF9WfimVwumw6s5Ba

https://proposed-gray-cattle.myfilebase.com/ipfs/Qmdjn3PA14PLyzmz1kwFfcmQnZRxBgYvcE8qcU2kAX86gi

(c) by Valery Shmelev (Deutsche: Valery Shmeleff)

PS. Если не понятен принцип работы Prompt или сгенерированного кода, можно попросить AI объяснить что и как делается.

https://oflameron.com

Vibe Coding

https://chat.deepseek.com/share/9ez7b9v1le3n5xzv0e