**Настройка двухфакторки для ssh в тг**

**Вставить свой токен бота и id чата**

**Для отключения удалить из файла** /etc/pam.d/sshd строку `auth include telegram-auth`

bash -c '  
apt update && apt install python3 python3-pip -y &&  
pip3 install python-telegram-bot aiofiles requests --break-system-packages &&  
cat <<EOF > /root/telegram\_auth.py  
import telegram  
from telegram import InlineKeyboardButton, InlineKeyboardMarkup  
import sys  
import os  
import asyncio  
from datetime import datetime  
import requests # Для получения информации о городе и провайдере  
import subprocess # Для выполнения системных команд  
  
# Конфигурация  
TOKEN = '7449414805:AAGuDLfYOeC1ylwooYkt1xEEbpGxRKOXc8I' # Токен вашего Telegram-бота  
CHAT\_ID = '9414805' # ID чата в Telegram, куда будут отправляться сообщения  
IP\_INFO\_URL = 'http://ipinfo.io/{}/json' # URL для получения информации о IP-адресе (город, провайдер и т.д.)  
  
# Создаем объект бота с использованием токена  
bot = telegram.Bot(token=TOKEN)  
  
# Словарь для хранения запросов на подтверждение  
request\_data = {}  
  
def get\_local\_ip():  
 """  
 Функция для получения локального IP-адреса машины, на которой выполняется скрипт.  
 Использует команду 'hostname -I' для получения IP-адресов и возвращает первый из них.  
 """  
 try:  
 result = subprocess.run(['hostname', '-I'], capture\_output=True, text=True)  
 return result.stdout.strip().split()[0] # Возвращаем первый IP из списка  
 except Exception:  
 return 'Неизвестен' # Возвращаем 'Неизвестен' в случае ошибки  
  
def get\_hostname():  
 """  
 Функция для получения имени хоста машины, на которой выполняется скрипт.  
 Использует команду 'hostname' для получения имени хоста.  
 """  
 try:  
 result = subprocess.run(['hostname'], capture\_output=True, text=True)  
 return result.stdout.strip() # Возвращаем имя хоста  
 except Exception:  
 return 'Неизвестен' # Возвращаем 'Неизвестен' в случае ошибки  
  
async def send\_telegram\_message(username, remote\_ip, request\_id):  
 """  
 Асинхронная функция для отправки сообщения в Telegram с информацией о попытке входа.  
 Сообщение включает время входа, IP-адрес, информацию о городе и провайдере,  
 локальный IP и имя хоста.  
 """  
 # Получаем информацию о IP  
 ip\_info = {}  
 try:  
 response = requests.get(IP\_INFO\_URL.format(remote\_ip))  
 ip\_info = response.json() # Преобразуем ответ в формат JSON  
 except Exception:  
 ip\_info = {} # Если возникла ошибка, оставляем словарь пустым  
  
 # Извлекаем информацию из ответа  
 city = ip\_info.get('city', 'Неизвестно')  
 provider = ip\_info.get('org', 'Неизвестно')  
 login\_time = datetime.now().strftime('%Y-%m-%d %H:%M:%S') # Текущее время в нужном формате  
 local\_ip = get\_local\_ip() # Получаем локальный IP  
 hostname = get\_hostname() # Получаем имя хоста  
  
 # Формируем текст сообщения  
 message = (f"🕒 Login Time: {login\_time}\n"  
 f"🏠 Hostname: {hostname}\n"  
 f"📍 Remote IP: {remote\_ip}\n"  
 f"🌐 System IP: {local\_ip}\n"  
 f"🔌 Provider: {provider}\n"  
 f"🏙️ City: {city}\n"  
 f"👤 Username: {username}")  
   
 # Создаем кнопки для ответа (разрешить или запретить вход)  
 reply\_markup = InlineKeyboardMarkup([  
 [InlineKeyboardButton("Разрешить", callback\_data=f"allow\_{request\_id}"),  
 InlineKeyboardButton("Запретить", callback\_data=f"deny\_{request\_id}")]  
 ])  
 try:  
 await bot.send\_message(chat\_id=CHAT\_ID, text=message, reply\_markup=reply\_markup)  
 except Exception:  
 pass # Игнорируем ошибки при отправке сообщения  
  
async def main():  
 """  
 Основная асинхронная функция, которая запускает процесс обработки входящих запросов.  
 """  
 global request\_data  
 username = os.getenv('PAM\_USER') # Получаем имя пользователя из переменной окружения PAM\_USER  
 remote\_ip = os.getenv('PAM\_RHOST') # Получаем IP-адрес удаленного хоста из переменной окружения PAM\_RHOST  
  
 if not username or not remote\_ip:  
 sys.exit(1) # Если данные отсутствуют, завершаем выполнение с кодом 1  
  
 # Создаем уникальный идентификатор запроса на основе текущего времени  
 request\_id = str(int(datetime.now().timestamp()))  
 # Сохраняем информацию о запросе в словаре  
 request\_data[request\_id] = {'username': username, 'remote\_ip': remote\_ip, 'timestamp': datetime.now().isoformat()}  
  
 # Отправляем сообщение в Telegram с запросом на подтверждение входа  
 await send\_telegram\_message(username, remote\_ip, request\_id)  
  
 update\_id = None # ID последнего обновления для бота  
 start\_time = datetime.now() # Время начала обработки запросов  
  
 while True:  
 try:  
 # Проверяем, прошло ли более 60 секунд с начала обработки запросов  
 if (datetime.now() - start\_time).total\_seconds() > 60:  
 sys.exit(1) # Завершаем выполнение, если прошло больше 60 секунд  
  
 # Получаем обновления от бота  
 updates = await bot.get\_updates(offset=update\_id, timeout=10)  
 for update in updates:  
 update\_id = update.update\_id + 1 # Обновляем ID последнего обновления  
 if update.callback\_query: # Проверяем, есть ли обратный вызов с кнопки  
 callback\_data = update.callback\_query.data # Извлекаем данные из обратного вызова  
 if callback\_data.startswith('allow\_') or callback\_data.startswith('deny\_'):  
 req\_id = callback\_data.split('\_')[1] # Извлекаем ID запроса из данных обратного вызова  
 if req\_id in request\_data:  
 if callback\_data.startswith('allow\_'):  
 del request\_data[req\_id] # Удаляем обработанный запрос из словаря  
 sys.exit(0) # Разрешаем вход  
 elif callback\_data.startswith('deny\_'):  
 del request\_data[req\_id] # Удаляем обработанный запрос из словаря  
 sys.exit(1) # Запрещаем вход  
 except Exception:  
 pass # Игнорируем ошибки в процессе обработки  
 await asyncio.sleep(1) # Ожидаем перед следующим запросом  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 asyncio.run(main()) # Запускаем основную асинхронную функцию  
EOF  
cat <<EOF > /etc/pam.d/telegram-auth  
auth requisite pam\_exec.so stdout /usr/bin/python3 /root/telegram\_auth.py  
EOF  
sed -i "/^auth\s.\*pam\_exec.so/d" /etc/pam.d/sshd  
echo "auth include telegram-auth" >> /etc/pam.d/sshd &&  
systemctl restart sshd &&  
echo "Скрипт завершен. Аутентификация через Telegram настроена и SSH перезапущен."  
'