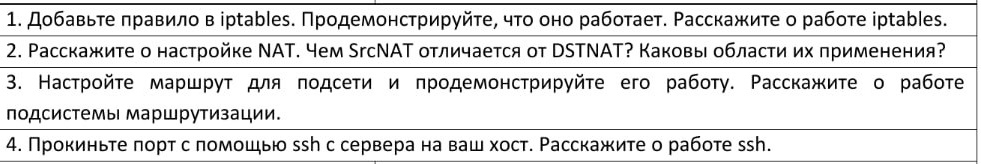
Задание:



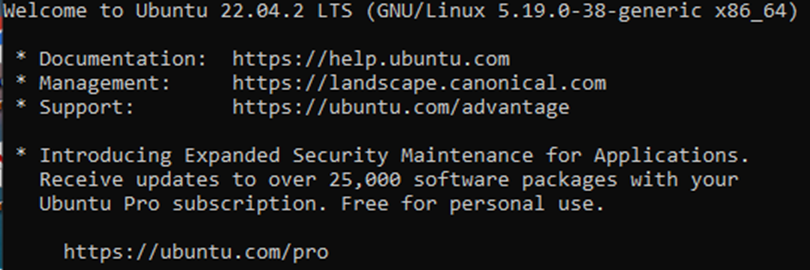
Добавление правила в iptables.

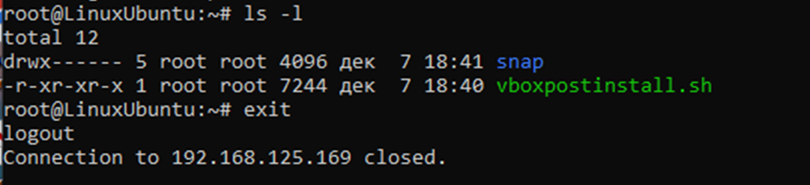
D:\my\универ\Распределенные системы\LR2 (Not done)\iptables.PNG

Первая выполненная команда закрывает порт 22 (закрывает доступ к ssh). Вторая выполненная – открывает порт 22 (открывает доступ к ssh).

До выполнения команды по закрытию порта 22, можно подключиться по ssh.







После выполнения команды по закрытию порта 22, подключиться по shh невозможно.

Iptables – утилита командной строки, является стандартным интерфейсом управления работой межсетевого экрана netfilter для ядер Linux.

Все сетевые пакеты, которые проходят через компьютер, отправляются компьютером или предназначены компьютеру, ядро направляет через фильтр iptables. Там эти пакеты поддаются проверкам и затем для каждой проверки, если она пройдена выполняется указанное в ней действие.

В фильтре iptables все пакеты делятся на три цепочки:

- input – обрабатывает входящие пакеты и подключения;

- forward – применяется для проходящих соединений;

- output – используется для исходящих пакетов и соединений.

Есть еще две дополнительные цепочки правил:

- prerouting – в эту цепочку пакет попадает перед обработкой iptables, система еще не знает куда он будет отправлен, в input, output или forward;

- postrouting – сюда попадают все проходящие пакеты, которые уже прошли цепочку forward.

Действия в правиле бывают следующие:

- ACCEPT - разрешить прохождение пакета дальше по цепочке правил;

- DROP - удалить пакет;

- REJECT - отклонить пакет, отправителю будет отправлено сообщение, что пакет был отклонен;

- LOG - сделать запись о пакете в лог файл;

- QUEUE - отправить пакет пользовательскому приложению.

Основные действия, которые позволяет выполнить iptables:

‑ A – добавить правило в цепочку;

- С – проверить все правила;

- D – удалить правило;

- I – вставить правило с нужным номером;

- L – вывести все правила в текущей цепочке;

- S – вывести все правила;

- F – очистить все правила;

- N – создать цепочку;

- X – удалить цепочку;

- P – установить действие по умолчанию.

Настройка NAT

Сначала нужно включить поддержку пересылки пакетов в /etc/sysctl.conf, чтобы трафик мог ходить между разными сетевыми интерфейсами.

Проверим текущее состояние:

sysctl net.ipv4.conf.all.forwarding

cat /proc/sys/net/ipv4/ip\_forward

Если оно равно 0, то включим следующей командой:

sysctl -w net.ipv4.conf.all.forwarding=1

Чтобы после перезапуска системы оно не сбросилось, откроем файл /etc/sysctl.conf и добавим строку:

net.ipv4.conf.all.forwarding=1

При необходимости можно очистить существующие NAT правила:

iptables -t nat –flush

Теперь осталось в iptables добавить правило, например,

iptables -t nat -A POSTROUTING -s 192.168.99.0/24 -j SNAT --to-source 172.16.16.94, где

192.168.99.0/24 внутренняя сеть, а 172.16.16.94 адрес, через который нужен выход в Интернет, аналогично прописываются другие внутренние сети.

Если IP адрес на внешнем сетевом интерфейсе меняется (динамический), тогда вместо SNAT укажем MASQUERADE:

iptables -t nat -A POSTROUTING -s 192.168.99.0/24 -j MASQUERADE

Сохраняем добавленные правила iptables.

nano /etc/rc.local

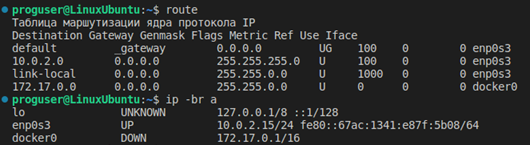
/sbin/iptables -t nat -A POSTROUTING -s 192.168.99.0/24 -o eth3 -j SNAT --to-source 172.16.90.1-172.16.90.5 –persistent

Цепочка SRCNAT (Source NAT) предназначена для трафика, условно названного «наружу». Изменяется адрес источника (source) на адрес внешнего интерфейса. Например, чтобы дать возможность клиентам локальной сети посещать сайты в интернет.

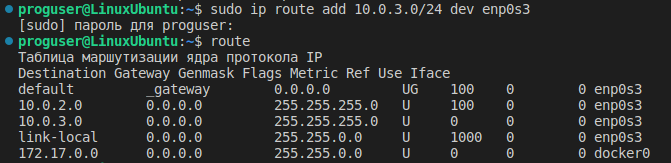
Цепочка DSTNAT (Destination NAT) определяет, как будет проходить «входящий» трафик. Роутер изменит адрес назначения (destination). Например, можно сделать доступным из вне веб-сервер, размещенный в локальной сети.

Маршрутизация

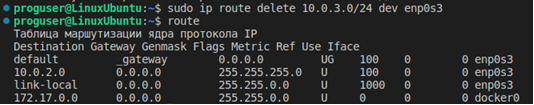
Просмотр таблицы маршрутизации осуществляется с помощью команды route.



Создадим новый маршрут с помощью команды ip route add. И проверим состояние таблицы маршрутизации.



Удаление маршрута происходит с помощью команды ip route delete. Удалим только что созданный нами маршрут и отобразим таблицу маршрутизации.



Сетевые маршруты необходимы, чтобы компьютеры могли определить по какой цепочке должен пойти пакет, чтобы достигнуть цели. Маршруты можно настроить на уровне интерфейса или маршрутизатора.

Когда компьютеру нужно отправить пакет в сеть он смотрит таблицу маршрутизации, в ней указаны IP-адреса пунктов назначения и адреса интерфейсов и роутеров в домашней сети, которые могут отправить пакет по нужному адресу. Если для цели маршрут не указан, то используется так называемый шлюз по умолчанию или маршрут по умолчанию. Точно такая же картина наблюдается на роутере. Устройство смотрит на IP адрес назначения и сверяет его со своей таблицей маршрутизации, а потом отправляет дальше.

Проброс портов через ssh

Чтобы пробросить порт с сервера на локальную машину через ssh, необходимо выполнить следующую команду на локальном ПК.

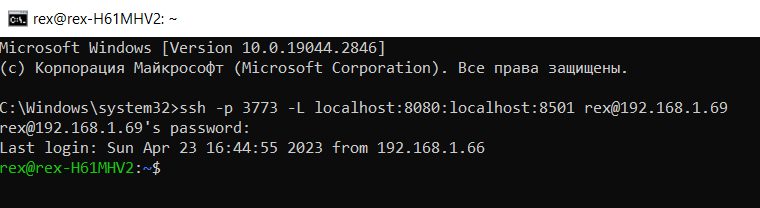
ssh -p 3773 -L localhost:8080:localhost:8501 [rex@192.168.1.69](mailto:rex@192.168.1.69), где

-p 3773 – указание порта для подключения по ssh,

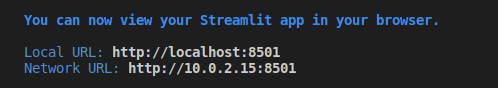
localhost:8080 – локальный адрес, на который перенаправится web-приложение

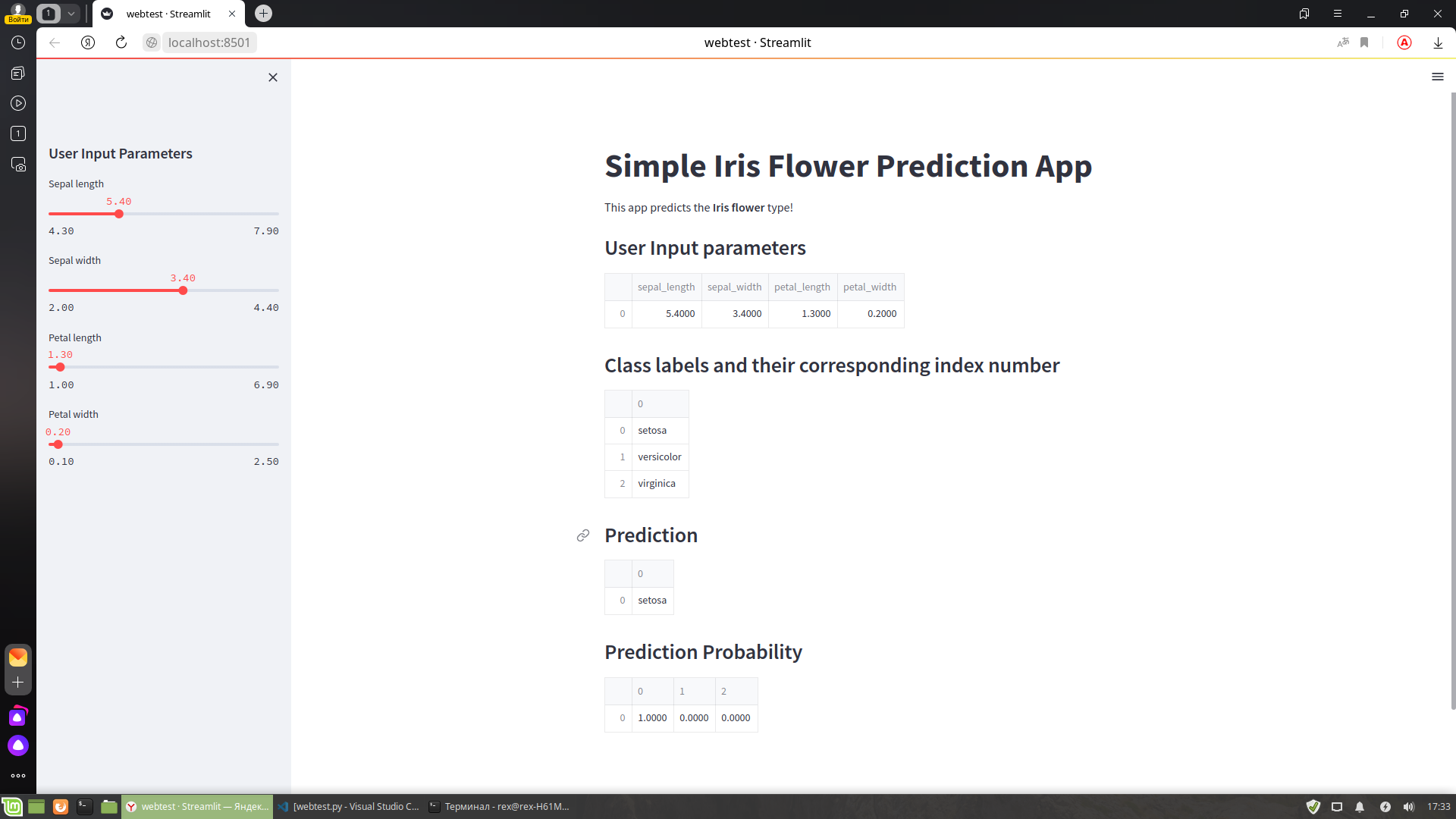
localhost:8501 – удаленный адрес, на котором запущено web-приложение

[rex@192.168.1.69](mailto:rex@192.168.1.69) – удаленный сервер

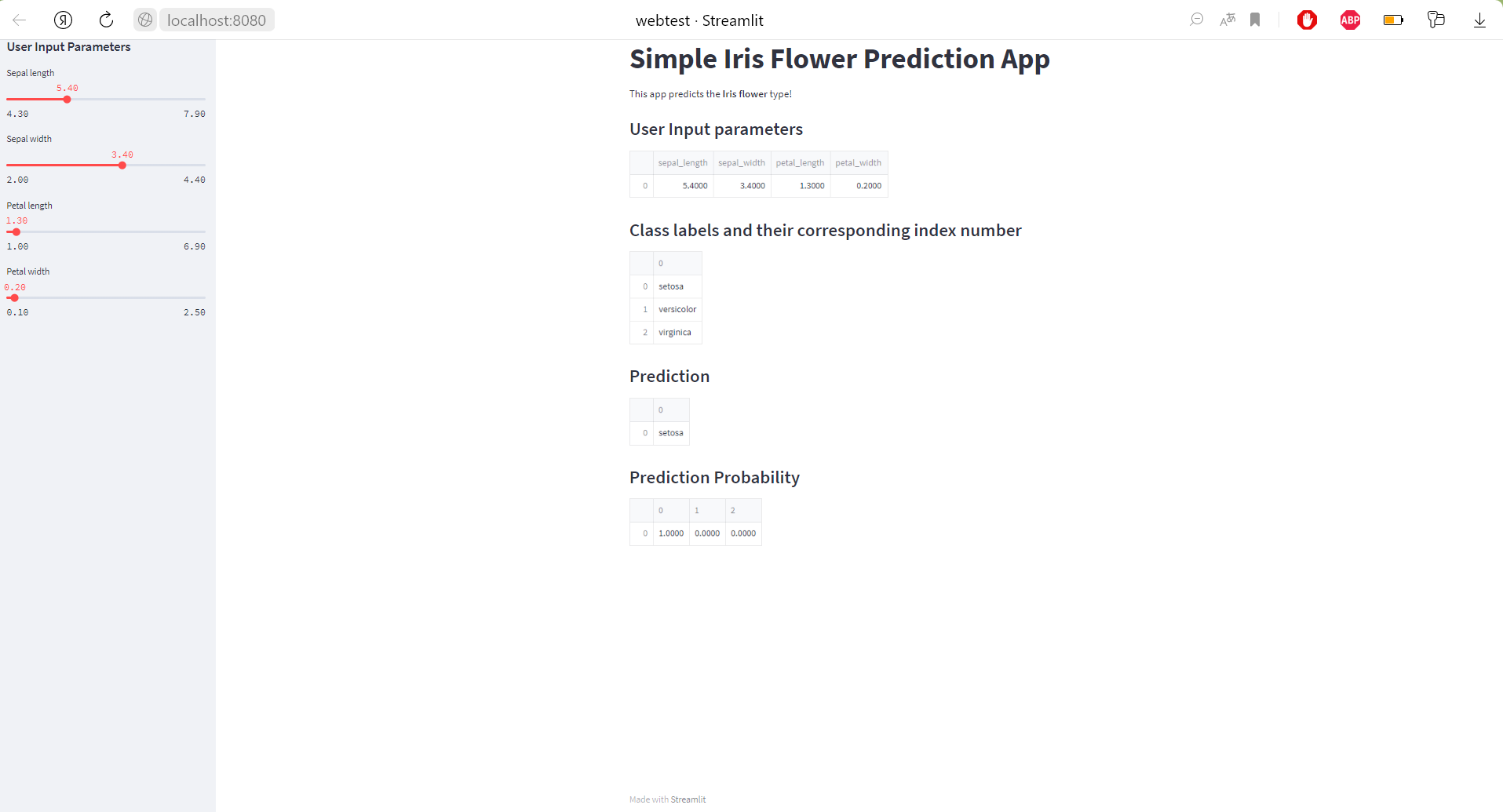


Приложение, которое запущено на сервере на localhost:8501.





Приложение, которое перенаправлено на локальную машину на localhost:8080.



SSH – это протокол, использующий клиент-серверную модель для аутентификации удаленных систем и обеспечения шифрования данных, обмен которыми происходит в рамках удаленного доступа.

По умолчанию для работы протокола используется TCP-22 порт: на нем сервер (хост) ожидает входящее подключение и после получения команды и проведения аутентификации организует запуск клиента, открывая выбранную пользователем оболочку. При необходимости пользователь может изменять используемый порт.

Для создания подключения SSH клиент должен инициировать соединение с сервером, обеспечив защищенное соединение и подтвердив свой идентификатор (проверяются соответствие идентификатора с предыдущими записями, хранящимися в RSA-файле, и личные данные пользователя для аутентификации).

SSH клиент под Linux можно использовать не только для безопасного подключения к консоли других компьютеров. Эта программа имеет ряд полезных функций, одной из которых является создание защищенных туннелей и эмуляция SOCKS5 прокси на локальной машине.

С помощью SSH-туннелей можно соединиться с сервером и использовать его сервисы через проброс портов и сокетов, и, наоборот, можно пробросить на сервер ресурсы со своего локального ПК, или создать прокси сервер с динамическим портом. И всё это с теми же средствами защиты данных, какие присутствуют при стандартном соединении с сервером по SSH.