# Transparent proxy support

This feature adds Linux 2.2-like transparent proxy support to current kernels. To use it, enable the socket match and the TPROXY target in your kernel config. You will need policy routing too, so be sure to enable that as well.

From Linux 4.18 transparent proxy support is also available in nf_tables.

## 1. Making non-local sockets work

The idea is that you identify packets with destination address matching a local socket on your box, set the packet mark to a certain value:

```
# iptables -t mangle -N DIVERT
# iptables -t mangle -A PREROUTING -p tcp -m socket -j DIVERT
# iptables -t mangle -A DIVERT -j MARK --set-mark 1
# iptables -t mangle -A DIVERT -j ACCEPT
```

Alternatively you can do this in nft with the following commands:

```
# nft add table filter
# nft add chain filter divert "{ type filter hook prerouting priority -150; }"
# nft add rule filter divert meta l4proto tcp socket transparent 1 meta mark set 1 accept
```

And then match on that value using policy routing to have those packets delivered locally:

```
# ip rule add fwmark 1 lookup 100
# ip route add local 0.0.0.0/0 dev lo table 100
```

Because of certain restrictions in the IPv4 routing output code you'll have to modify your application to allow it to send datagrams _from_ non-local IP addresses. All you have to do is enable the (SOL_IP, IP_TRANSPARENT) socket option before calling bind:

```
fd = socket(AF_INET, SOCK_STREAM, 0);
/* - 8< -*/
int value = 1;
setsockopt(fd, SOL_IP, IP_TRANSPARENT, &value, sizeof(value));
/* - 8< -*/
name.sin_family = AF_INET;
name.sin_port = htons(0xCAFE);
name.sin_addr.s_addr = htonl(0xDEADBEEF);
bind(fd, &name, sizeof(name));
```

A trivial patch for netcat is available here: http://people.netfilter.org/hidden/tproxy/netcat-ip_transparent-support.patch

## 2. Redirecting traffic

Transparent proxying often involves "intercepting" traffic on a router. This is usually done with the iptables REDIRECT target; however, there are serious limitations of that method. One of the major issues is that it actually modifies the packets to change the destination address -- which might not be acceptable in certain situations. (Think of proxying UDP for example: you won't be able to find out the original destination address. Even in case of TCP getting the original destination address is racy.)

The 'TPROXY' target provides similar functionality without relying on NAT. Simply add rules like this to the iptables ruleset above:

```
# iptables -t mangle -A PREROUTING -p tcp --dport 80 -j TPROXY \
  --tproxy-mark 0x1/0x1 --on-port 50080
```

Or the following rule to nft:

# nft add rule filter divert tcp dport 80 tproxy to :50080 meta mark set 1 accept

Note that for this to work you'll have to modify the proxy to enable (SOL_IP, IP_TRANSPARENT) for the listening socket.

As an example implementation, tcprdr is available here: https://git.breakpoint.cc/cgit/fw/tcprdr.git/ This tool is written by Florian Westphal and it was used for testing during the nf_tables implementation.

## 3. Iptables and nf_tables extensions

To use tproxy you'll need to have the following modules compiled for iptables:

- NETFILTER_XT_MATCH_SOCKET
- NETFILTER_XT_TARGET_TPROXY

Or the floowing modules for nf_tables:

- NFT_SOCKET

- NFT_TPROXY

# 4. Application support

## 4.1. Squid

Squid 3.HEAD has support built-in. To use it, pass '--enable-linux-netfilter' to configure and set the 'tproxy' option on the HTTP listener you redirect traffic to with the TPROXY iptables target.

For more information please consult the following page on the Squid wiki: http://wiki.squid-cache.org/Features/Tproxy4