TProxy - Transparent proxying, again

Balázs Scheidler, Krisztián Kovács

BalaBit IT Ltd.

Netfilter Workshop 2008
2008.10.01.

Architecture

- Proxies run in userspace.
- TCP stack gives some additional features that enable proxies to operate in a transparent manner.
- Primary aim is to work with routed traffic, albeit there are demands for L2 transparent mode.

TProxy features

- Redirect portions of the routed traffic to the local processes (similar to REDIRECT in NAT)
  - Use case: routed port 80 traffic needs to go through Squid.
- Initiate connections from a non-local IP as a source.
  - Use case: active FTP DATA connection should use the IP used by the client.
- Listen on non-local IP addresses without iptables rules.
  - Use case: passive FTP data connections are coming to a random port, adding explicit rules to redirect traffic would be infeasable.

How does it work?

- Divert traffic to local proxies: use the TPROXY target in the "mangle" table
  - The target looks up the socket hash, if there's a matching socket:
    - it assigns an nfmark value to the skb
    - it assigns a socket reference to skb->sk
  - A socket matches iff:
    - The port number matches, AND
    - It has the IP_RANSPARENT setsockopt set.
- The user is responsible for setting up routing in a way that the TPROXY assigned mark is routed locally:

Routing

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

How does it work?

- Initiating or accepting connections from/to non-local addresses:
    - A match in the “mangle” table: socket
    - The match identifies packets that are destined to a local socket (it does a socket lookup)
- The user is responsible for setting up an iptables rule that MARKs packets that directs routing to route those locally.

Core kernel changes

- route_output() check is loosened for IP_TRANSPARENT sockets:
  - Does not require the source address to be local;
  - Needs the “flags” member in struct flowi that was dropped in 2.6.27.
- The IP_TRANSPARENT value needs to be propagated to all route_output() calls
- TCP handshake: the stack uses the port in the header of the incoming packet instead of the listener socket when replying with SYN-ACK
- UDP and TCP lookup function has to be exported

Core kernel changes

- Split Netfilter defragmentation hooks into a separate module, this makes TProxy independent of conntrack, and a separate defrag module makes sense (you can use transparent proxies without conntrack)
- TCP and UDP input path is modified to use the socket reference from skb->sk instead of looking it up (if present, ie. packets intercepted with the TPROXY target)

URLs

Kernel

- `http://people.netfilter.org/hidden/tproxy`
- `git://people.netfilter.org/hidden/tproxy.git`

iptables

- `git://git.balabit.hu/bazsi/iptables-tproxy.git`

Users

- Squid 3.HEAD:
  `http://www.squid-cache.org/bzr/squid3/trunk`