Secure use of iptables and connection tracking helpers

Authors: Eric Leblond, Pablo Neira Ayuso, Patrick McHardy, Jan Engelhardt, Mr Dash Four

Introduction

Principle of helpers

Some protocols use different flows for signaling and data transfers. This is the case for FTP, SIP and H.323 among many others. In the setup stage, it is common that the signaling flow is used to negotiate the configuration parameters for the establishment of the data flow, i.e. the IP address and port that are used to establish the data flow. These sort of protocols are particularly harder to filter by firewalls since they violate layering by introducing OSI layer 3/4 parameters in the OSI layer 7.

In order to overcome this situation in the iptables firewall, Netfilter provides the Connection Tracking helpers, which are modules that are able to assist the firewall in tracking these protocols. These helpers create the so-called expectations, as defined by the Netfilter project jargon. An expectation is similar to a connection tracking entry, but it is stored in a separate table and generally with a limited duration. Expectations are used to signal the kernel that in the coming seconds, if a packet with corresponding parameters reaches the firewall, then this packet is RELATED to the previous connection.

These kind of packets can then be authorized thanks to modules like state or conntrack which can match RELATED packets.

This system relies on parsing of data coming either from the user or the server. It is therefore vulnerable to attack and great care must be taken when using connection tracking helpers.

Connection Tracking helpers default configuration

Due to protocol constraints, not all helpers are equal. For example, the FTP helper will create an expectation whose IP parameters are the two peers. The IRC helper creates expectations whose destination address is the client address and source address is any address. This is due to the protocol: we do not know the IP address of the person who is the target of the DCC.

The degree of freedom due to connection tracking helpers are therefore dependent on the nature of the protocol. Some protocols have dangerous extensions, and these are disabled by default by Netfilter. The user has to pass an option during loading of the module to enable this dangerous protocol features. For example, the FTP protocol can let the user choose to have the target server connect to another arbitrary server. This could lead to a hole in the DMZ and it is therefore deactivated by default. To enable it, you've got to pass the *loose* option with the *1* value.

The following list describes the different connection tracking helper modules and their associated degree of freedom:

Module	Source address	Source Port	Destination address	Destination port	Protocol	Option
amanda	Fixed	0-65535	Fixed	In CMD	TCP	
ftp	Fixed	0-65535	In CMD	In CMD	TCP	loose = 0 (default)
ftp	Full	0-65535	In CMD	In CMD	TCP	loose = 1
h323	Fixed	0-65535	Fixed	In CMD	UDP	
h323 q931	Fixed	0-65535	In CMD	In CMD	UDP	
irc	Full	0-65535	Fixed	In CMD	TCP	
netbios_ns	Iface Network	Fixed	Fixed	Fixed	UDP	
pptp	Fixed	In CMD	Fixed	In CMD	GRE	
sane	Fixed	0-65535	Fixed	In CMD	TCP	
sip rtp_rtcp	Fixed	0-65535	Fixed	In CMD	UDP	sip_direct_media = 1 (default)
sip rtp_rtcp	Full	0-65535	In CMD	In CMD	UDP	sip_direct_media = 0
sip signalling	Fixed	0-65535	Fixed	In CMD	In CMD	sip_direct_signalling = 1 (default)
sip signalling	Full	0-65535	In CMD	In CMD	In CMD	sip_direct_signalling = 0
tftp	Fixed	0-65535	Fixed	In Packet	UDP	

The following keywords are used:

- Fixed: Value of a connection tracking attribute is used. This is not a candidate for forgery.
- In CMD: Value is fetched from the payload. This is a candidate for forgery.

The options are module loading options. They permit activation of the extended but dangerous features of some protocols.

Secure use of Connection Tracking Helpers

Following the preceding remarks, it appears that it is necessary to not blindly use helpers. You must take into account the topology of your network when setting parameters linked to a helper.

For each helper, you must carefully open the RELATED flow. All iptables statements using "-m conntrack --ctstate RELATED" should be used in conjunction with the choice of a helper and of IP parameters. By doing that, you will be able to describe how the helper must be used with respect to your network and information system architecture.

Example: FTP helper

For example, if you run an FTP server, you can setup

```
iptables -A FORWARD -m conntrack --ctstate RELATED -m helper \
       --helper ftp -d $MY_FTP_SERVER -p tcp \
       --dport 1024: -j ACCEPT
```

If your clients are authorized to access FTP outside of your network, you can add

```
iptables -A FORWARD -m conntrack --ctstate RELATED -m helper \
       --helper ftp -o $OUT_IFACE -p tcp \
       --dport 1024: -j ACCEPT
iptables -A FORWARD -m conntrack --ctstate RELATED -m helper \
       --helper ftp -i $OUT_IFACE -p tcp \
       --dport 1024: -j ACCEPT
```

The same syntax applies to IPV6

```
ip6tables -A FORWARD -m conntrack --ctstate RELATED -m helper \
       --helper ftp -o $OUT_IFACE -p tcp \
       --dport 1024: -j ACCEPT
ip6tables -A FORWARD -m conntrack --ctstate RELATED -m helper \
       --helper ftp -i $OUT_IFACE -p tcp \
       --dport 1024: -j ACCEPT
```

Example: SIP helper

You should limit the RELATED connection due to the SIP helper by restricting the destination address to the RTP server farm of your provider

```
iptables -A FORWARD -m conntrack --ctstate RELATED -m helper \
       --helper sip -d $ISP_RTP_SERVER -p udp -j ACCEPT
```

Example: h323 helper

The issue is the same as the one described for SIP, you should limit the opening of the RELATED connection to the RTP server addresses of your VOIP provider.

Securing the signaling flow

You will also need to build carefully crafted rules for the authorization of flows involving connection tracking helpers. In particular, you have to do strict anti-spoofing (as described below) to avoid traffic injection from other interfaces.

Using the CT target to refine security

Introduction

One classic problem with helpers is the fact that helpers listen on predefined ports. If a service does not run on standard port, it is necessary to declare it. Before 2.6.34, the only method to do so was to use a module option. This was resulting in having a systematic parsing of the added port by the chosen helper. This was clearly suboptimal and the CT target has been introduced in 2.6.34. It allows to specify what helper to use for a specific flow. For example, let's say we have a FTP server on IP address 1.2.3.4 running on port 2121.

To declare it, we can simply do

```
iptables -A PREROUTING -t raw -p tcp --dport 2121 \
        -d 1.2.3.4 -j CT --helper ftp
```

Therefore, the use of the module options is NOT recommended anymore - please use the CT target instead.

Disable helper by default

Principle

Once a helper is loaded, it will treat packets for a given port and all IP addresses. As explained before, this is not optimal and is even a security risk. A better solution is to load the module helper and deactivate their parsing by default. Each helper we need to use is then set by using a call to the CT target.

Method

Since Linux 3.5, it is possible to desactivate the automatic conntrack helper assignment. This can be done when loading the nf_conntrack module

```
modprobe nf_conntrack nf_conntrack_helper=0
```

This can also be done after the module is loading by using a /proc entry

```
echo 0 > /proc/sys/net/netfilter/nf_conntrack_helper
```

Please note that flows that already got a helper will keep using it even if automatic helper assignment has been disabled.

For older kernel, it is possible to obtain this behavior for most connection tracking helper modules by setting the port number for the module to 0. For example

```
modprobe nf_conntrack_$PROTO ports=0
```

By doing this, the following modules will be deactivated on all flows by default:

- ftp
- irc
- sane
- sip
- tftp

Due to the absence of a “ports” parameter, some modules will not work:

- amanda
- h323
- netbios_ns
- pptp
- snmp

Please note, this will cause a renaming of the conntrack helper which will be named $PROTO-0. The CT rules must then be updated to reflect this change. For example, if the option has been used for the ftp helper, one should use

```
iptables -A PREROUTING -t raw -p tcp --dport 21 \
       -d 2.3.4.5 -j CT --helper ftp-0
```

Anti-spoofing

Helpers and anti-spoofing

Helpers rely on the parsing of data that come from client or from server. Therefore, it is important to limit spoofing attacks that could be used to feed the helpers with forged data. Helpers are IP only and are not doing, as the rest of the connection tracking, any coherence check on the network architecture.

Using rpfilter module

A rpfilter Netfilter module is available since Linux 3.3 and iptables 1.4.13. It provides a convenient match that can be used to detect invalid packets. To use it on IPv6 and IPv4, one can for example use

```
iptables -A PREROUTING -t raw -m rpfilter --invert -j DROP
ip6tables -A PREROUTING -t raw -m rpfilter --invert -j DROP
```

Using rp_filter

Linux provides a routing-based implementation of reverse path filtering. This is available for IPv4. To activate it, you need to ensure that */proc/sys/net/ipv4/conf/*/rp_filter* files contain 1. Complete documentation about *rp_filter* is available in the file *ip-sysctl.txt* in the *Documentation/networking/* directory of the Linux tree.

The documentation at the time of the writing is reproduced here

```
rp_filter - INTEGER
   0 - No source validation.
   1 - Strict mode as defined in RFC3704 Strict
       Reverse Path. Each incoming packet is
       tested against the FIB and if the interface
       is not the best reverse path the packet
       check will fail. By default, failed packets
       are discarded.
   2 - Loose mode as defined in RFC3704 Loose
       Reverse Path. Each incoming packet's source
       address is also tested against the FIB
       and if the source address is not reachable
       via any interface, the packet check will fail.

   Current recommended practice in RFC3704 is to
   enable strict mode to prevent IP spoofing from
   DDos attacks. If using asymmetric routing
   or other complicated routing, then loose mode
   is recommended.

   The max value from conf/{all,interface}/rp_filter
   is used when doing source validation on the
   {interface}.

   Default value is 0. Note that some distributions
   enable it in startup scripts.
```

At the time of the writing, there is no routing-based implementation of *rp_filter* in the Linux kernel for IPv6, therefore manual anti-spoofing via Netfilter rules is thus needed.

Manual anti-spoofing

The best way to do anti-spoofing is to use filtering rules in the RAW table. This has the great advantage of bypassing the connection tracking and helps to reduce the load that could be created by some flooding.

Anti-spoofing must be done on a per-interface basis. For each interface, we must list the authorized network on the interface. There is an exception, which is the interface with the default route where an inverted logic must be used. In our example, let's take eth1, which is a LAN interface, and have eth0 being the interface with the default route. Let's also have $NET_ETH1 being the network connected to $ETH1 and $ROUTED_VIA_ETH1 a network routed by this interface. With this setup, we can do anti-spoofing with the following rules

```
iptables -A PREROUTING -t raw -i eth0 -s $NET_ETH1 -j DROP
iptables -A PREROUTING -t raw -i eth0 -s $ROUTED_VIA_ETH1 -j DROP
iptables -A PREROUTING -t raw -i eth1 -s $NET_ETH1 -j ACCEPT
iptables -A PREROUTING -t raw -i eth1 -s $ROUTED_VIA_ETH1 -j ACCEPT
iptables -A PREROUTING -t raw -i eth1 -j DROP
```

The IPv6 case is similar if we omit the case of the local link network

```
ip6tables -A PREROUTING -t raw -i eth0 -s $NET_ETH1 -j DROP
ip6tables -A PREROUTING -t raw -i eth0 -s $ROUTED_VIA_ETH1 -j DROP
ip6tables -A PREROUTING -t raw -s fe80::/64 -j ACCEPT
ip6tables -A PREROUTING -t raw -i eth1 -s $NET_ETH1 -j ACCEPT
ip6tables -A PREROUTING -t raw -i eth1 -s $ROUTED_VIA_ETH1 -j ACCEPT
```