Bridgewalling - Using Netfilter in Bridge Mode

Ralf Spenneberg, ralf@spenneberg.net

Revision : 1.5

Abstract

Firewalling using packet filters is usually performed by a router. The packet filtering software decides whether the packet will be routed or not. Installing such a firewall in an existing network architecture often requires the modification of networks and IP addresses. If the firewall device functions as a bridge no change is needed. The firewall is just placed in between the machines to be firewalled. Additionally, the firewall is invisible on the IP-layer.

Unfortunately not very much documentation covering the setup of a bridging firewall on Linux exists. This is even true although Linux offers a very advanced stateful packet filtering machinery (Netfilter). This paper tries to give an overview of the current state of "bridgewalling" and the different approaches.

1 What is it about?

Physical networks can be interconnected with either routers or bridges. Both devices are able to connect these networks and to forward packets in between. Both devices do work on different layers though. A bridge (often better known as a two-port-switch) operates on layer 2, forwarding packets on the link layer based on the MAC address. A router operates on layer 3 and forwards the packets based on the destination IP address (at least usually, policy routing can do more, I know ;-).

Now, whenever a saturated network needs to be segmented both technologies may be used. If a router is used to segment the network, both segments need to contain their own logical network consisting of their own IP subnet, a common netmask and a default gateway entry in their routing table pointing at the router. Whenever a machine tries to contact a different machine it uses its netmask to distinguish between machines it can reach directly and machines it needs to reach by the use of the default gateway.

If the destination machine is on the same network it will issue an Address Resolution Protocol (ARP) request. This request will be broadcast to all network devices on the same network (wire). All network devices will see the ARP request and the appropriate device will answer using an ARP reply. Then the two machines may communicate directly.

If the comparison of the IP addresses results in the fact that the two hosts are on different networks, the source host generates an ARP request to get the MAC address of the default gateway. The default gateway will answer using an ARP reply and the packet will be sent to the gateway using the IP address of the real destination as destination address and the MAC address of the gateway as MAC destination address.

If a bridge is used to connect several segments of a former combined network no change is needed on the IP level. The changes needed on the link level usually happen automatically. As soon as the bridge is set up, it will start forwarding packets between the different segments. While doing this it will learn the MAC addresses of the different devices in all connected segments. Based on this knowledge a table can be constructed which then is used to forward the packets only to those segments where the hosts are located. Thus the bridge segments the traffic.

Bridges may be used to connect networks redundantly. The *Spanning Tree Protocol* (STP) may be used between the different switches/bridges to negotiate the shortest path available. Linux fully supports STP unlike other operating systems like FreeBSD. Therefore, Linux can be used together with other commercial switch devices.

The last big difference between a bridge and an IP router are the protocols supported. Since a bridge does not know anything about protocols higher than ARP a bridge may forward several protocols not supported by an IP router: Appletalk, IPX etc.

2 Different Approaches

Different approaches exist in Linux to enable forwarding of packets on the link layer.

2.1 Using ProxyARP

The easiest way to setup a device to forward packets as a pseudo bridge using the ProxyARP feature of the Linux Kernel 2.4. This is actually not a real bridge but more a hidden router. This feature is available in the standard kernel without the need to patch the kernel.

Essentially the pseudo bridge connects two network segments of one logical network. The routing table of the bridge has been configured to know the hosts on each side of the bridge. Whenever a host on the left side of the bridge makes an ARP request for a host on the right side, the bridge will answer the request and tell the left host to forward the packet to the bridge. The bridge will then filter and forward the packet to the host on the right side.

To enable ProxyARP on a Linux machine the following steps have to be taken:

1. Get a machine with at least two interfaces
2. Assign IP addresses to the interfaces
3. Set up the routing table so that the bridge knows which host resides on what interface
4. Turn on ProxyARP on the participating network cards:

   ```
   sysctl -w net.ipv4.conf.$dev$.proxy_arp=1
   ```

5. Turn on IP forwarding for all devices or for each device separately:

   ```
   sysctl -w net.ipv4.ip_forward=1  or
   sysctl -w net.ipv4.conf.$dev$.forwarding=1
   ```

6. Setup your iptables rules

2.2 Using Bridge-nf

The bridge-netfilter patch [1] is maintained by Lennert Buytenhek. This patch extends the already built in bridging functionality of the Linux kernel. The packets forwarded by the bridge are now passed through the Linux Netfilter tables and chains.

2.2.1 Configuration of the bridge

As soon as the kernel is patched, configured, compiled and installed the bridge can be configured. In the following example a bridge with two ports is configured. For configuration the bridge control command `brctl` is used.

To create the bridge use the command `brctl addbr br0`. This command creates the bridge `br0`. This device is immediately available:

```
# ip link show br0
4: br0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop
    link/ether 00:00:00:00:00:00 brd ff:ff:ff:ff:ff:ff
```

Now those network cards, which should become part of the bridge, can be added to the bridge. These network cards will become the ports of the bridge/switch. Once they have been added the network cards and the bridge may be enabled:

```
# brctl addif br0 eth0
# brctl addif br0 eth1
# ip link set eth0 up
# ip link set eth1 up
# ip link set br0 up
# ip link show
1: lo: <LOOPBACK,UP> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,PROMISC,UP> mtu 1500 qdisc pfifo_fast
qlen 100
    link/ether 00:20:e0:6c:72:1e brd ff:ff:ff:ff:ff:ff
3: eth1: <BROADCAST,MULTICAST,PROMISC,UP> mtu 1500 qdisc pfifo_fast
qlen 100
    link/ether 00:10:a4:c3:26:cb brd ff:ff:ff:ff:ff:ff
4: br0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue
    link/ether 00:10:a4:c3:26:cb brd ff:ff:ff:ff:ff:ff
# brctl show
bridge name bridge id STP enabled interfaces
br0 8000.0010a4c326cb yes eth0
```

It is important to note that the enslaved interfaces of the bridge `eth0`and `eth1` must not have an IP address! The virtual bridge interface `br0` may use an IP address. This is only needed if the machine is to be administered remotely. To configure an IP address for the bridge interface the `ifconfig` or the `ip` command may be used. This paper solely uses the new `ip` command:

```
# ip addr add 192.168.0.5/24 brd 192.168.0.255 dev br0
# ip route add default via 192.168.0.1
```

To delete a port of the bridge the command `brctl delif` may be used. To delete the bridge itself the command `brctl delbr` is used. The ports of the bridge have to be deleted first!

Now the characteristics of the bridge can be tuned. This includes the ageing of the MAC database and the participation in the *Spanning Tree Protocol* (STP). For a discussion of STP read the last section.

2.2.2 Tuning of the bridge - Ageing

As soon as the bridge is UP the bridge starts forwarding packets it tries to learn the associated MAC addresses. If the bridge knows where the destination MAC resides it only forwards the packet to the appropriate port. If it does not know where the destination is, it will forward the packet on all ports.

Since machines get moved around, this information is not static but can change very fast. To take a look at the current MAC table of the bridge the command `brctl showmacs br0` may be used for the bridge `br0`

```
# brctl showmacs br0
port no mac addr is local? ageing timer
  1 00:20:e0:6c:72:1e yes    0.00
```

To tune the ageing behavior of the bridge to the local network the ageing timer may be adjusted using `brctl setageing br0 <seconds>`. This will define the ageing time for the entries in the MAC table. The default ageing timer is 300 seconds.

For performance reasons the bridge does not search the MAC table for timed out entries each second but every 4 seconds by default. This garbage collector interval may be adjusted using `brctl setgcint br0 <seconds>`.

2.2.3 Tuning of the bridge - STP

The Spanning Tree Protocol allows two bridges to operate redundantly. Normally redundant bridges would result in duplicated packets which would saturate the connected networks (see last section). If the bridges may use STP they negotiate the shortest possible link between the connected networks and disable all other possible links. As soon as a link fails STP recalculates the links and can enable a workaround for the failed link.

For the Linux bridge to take part in this negotiation, the Spanning Tree Protocol needs to be enabled. STP can be switched on and off using `brctl stp br0 on|off`. It is turned on by default when adding the bridge. The STP protocol should only be turned off if the Linux bridge is the only bridge within the network.

The STP protocol first elects a *root bridge* (look in the last section for an explanation of the root bridge). The root bridge is the bridge with the lowest priority within the network. If several bridges have the same priority assigned the bridge with the lowest MAC address is chosen. To configure the bridge priority the following command may be used: `brctl setbridgeprio br0 <value>`. The possible range is 0-65535. The default value used is 0x8000.

When using the STP protocol the bridge may not immediately start forwarding packets on all ports when turned on. The forward delay timer defines the time to stay in the listening and learning state before entering the forwarding state. This timer may be adjusted using `brctl setfd br0 <seconds>`. The default is 15 seconds.

To communicate with other bridges the bridge sends BPDUs in hello messages. The interval between these hello messages is initially set to 2 seconds. It can be adjusted using `brctl sethello br0 <seconds>`.

If the last hello message has been seen more than 20 seconds ago this bridge will start a new root bridge election. This is called the maximum (message) age. It can be modified using `brctl setmaxage br0 <seconds>`

To calculate the shortest link available when negotiating the STP protocol, the bridge needs to know the costs associated with sending a packet on a port. This cost value can be specified individually for each port using the command `brctl setpathcost br0 eth0 <value>`. `br0` is the bridge and `eth0` is the port to set the cost for. The ports with lower cost get used first.

If two ports are available for the same link to a different network the bridge sometimes does not choose the optimal ports. The priority for each port can be configured using `brctl setportprio br0 eth0 <0-255>`. When in doubt the bridge now choose the port with the higher priority (lower value). This value is used in the election of the designated port (see below).

When the STP protocol has been configured all values can be displayed using the command `brctl showstp br0`.

```
# brctl showstp br0
br0
 bridge id 000b.0020e06c721e
 designated root 000b.0020e06c721e
 root port    0 path cost    0
 max age  100.00 bridge max age  100.00
 hello time    2.00 bridge hello time    2.00
 forward delay   15.00 bridge forward delay   15.00
 ageing time    5.00 gc interval    4.00
 hello timer    0.25 tcn timer    0.00
 topology change timer    0.00 gc timer    2.25
 flags


eth0 (1)
 port id 8001 state forwarding
 designated root 000b.0020e06c721e path cost  100
```

```
 designated bridge 000b.0020e06c721e message age timer    0.00
 designated port 8001 forward delay timer    0.00
 designated cost    0 hold timer    0.25
 flags

-- rest skipped --
```

2.2.4 Using iptables to filter on this bridge

Once the bridge has been configured and enabled Netfilter will filter all packets forwarded by the bridge if the mentioned bridge-nf patch has been applied.

Now, to filter packets the rules are defined as usual. To filter packets between the ports `eth0` and `eth1` on `br0` the following syntax may be used.

```
# iptables -A FORWARD --in-interface eth0 --out-interface eth1 \
-j ACCEPT
```

There is one caveat when using iptables on a bridge. While learning MAC addresses the bridge will forward the packet on all ports if it does not know where the destination MAC is located. When REJECT rules are used in such a scenario this can cause some problems. Take a look at the following line:

```
# iptables -A FORWARD --in-interface eth1 --out-interface eth0 \
-j REJECT
```

If a host connected to Port `eth1` tries to contact another host on Port `eth1` the bridge will see the packet, too. If the bridge has not seen the destination address before it will not know where the destination host is located. Therefore it will forward that packet on all ports. While forwarding the Netfilter code will notice that the packet has to be rejected. In the end the original packet will not be forwarded by the bridge but a REJECT packet will be send to the source resulting in a dropped connection. This will happen each time when the destination MAC address is not known by the bridge. Further packets rejected by this rule would be broadcast and multicast packets.

To get around this problem the IP addresses should be specified in the iptables rules. This might pose a problem. Usually when using a bridge the IP addresses cannot be specified easily using a network and a netmask. Netfilter offers the `ippool` command and the options `--srcpool` and `--dstpool`. The tool and the iptables options are part of patch-o-matic and not enabled by default in the Linux kernel. They allow the easy specification of the IP addresses and their definition in iptables rules:

```
# ippool -F
# ippool -N left
# ippool -A left 192.168.0.5
# ippool -A left 192.168.0.17
# ippool -A left 192.168.0.18
#
# iptables -A FORWARD -i eth1 -o eth0 --dstpool left -j REJECT
```

2.3 Using ebtables

The bridge-nf patch by Lennert Buytenhek already implements firewall in bridge mode. Why do we need an additional patch? The bridge-nf patch is not able to filter Ethernet packets! On a bridge we might want to filter other protocols than IPv4 and IPv6. The *ebtables*-Patch [2] can filter based on the protocol transferred in the Ethernet frame. This patch introduces three further tables: broute, nat, filter. The broute table enables bridge-routing (brouting) capabilities. This means that the brouter is able to route IPv4 based traffic between two networks but can still bridge non-routable traffic (like NetBEUI, Appletalk, etc.) between the two networks. The bridge-nf patch always bridged the traffic between the two networks. There is no option to route parts of the traffic.

The BROUTING chain decides whether to route or to bridge the packet. In addition to the broute table and the BROUTING chain, the ebtables patch adds a PREROUTING chain which is traversed after the BROUTING chain but before the bridge decision is made. This is similar to the iptables PREROUTING chain which is traversed before the ROUTING decision is made. Thus the ebtables patch is able to do destination network address translation (DNAT) on the link layer using MAC addresses.

As a link layer PREROUTING chain is added by the ebtables patch, naturally a POSTROUTING chain is added, too. The POSTROUTING chain can be used to do SNAT on the link layer modifying the source mac address.

2.3.1 Filtering non IP protocols

To filter non IP protocols the command `ebtables` may be used. The available types are stored in the file `/etc/ethertypes`. Some examples are:

```
IPv4    0800
X25     0805
ARP     0806
```

```
802_1Q  8100    802.1Q Virtual LAN tagged frame
IPX     8137
IPv6    86DD
NetBEUI 8191
```

The protocol can be specified using either the name or the number. Ethernet 802.2/802.3 networks used the protocol field as a length field. The number in this field is less or equal to 0x0600. To filter these packets the name `LENGTH` may be used.

The following example allows Appletalk and IPv4 but blocks NetBEUI and everything else:

```
# ebtables -A FORWARD -p AARP -j ACCEPT
# ebtables -A FORWARD -p ATALK -j ACCEPT
# ebtables -A FORWARD -p ARP -j ACCEPT
# ebtables -A FORWARD -p IPv4 -j ACCEPT
# ebtables -P FORWARD DROP
```

2.3.2 Routing one protocol and bridging the other

The ebtables patch allows for routing the IPv4 protocol and for bridging other protocols like NetBEUI and Appletalk between the same two networks. The decision whether to route or to bridge the packet the broute table with the BROUTING chain is used. This chain allows for two targets: ACCEPT and DROP.

These targets have a special meaning when used in this chain:

- `ACCEPT` The packet is bridged.
- `DROP` The packet is routed.

So to route all IPv4 traffic and to bridge all other the following rules can be used:

```
# ebtables -t broute -A BROUTING -i eth0 -p IPv4 -j DROP
# ebtables -t broute -A BROUTING -i eth1 -p IPv4 -j DROP
# ebtables -t broute -A BROUTING -i eth0 -p ARP -j DROP
# ebtables -t broute -A BROUTING -i eth1 -p ARP -j DROP
```

You have to make sure, that the ARP request are not bridged either.

2.4 Spanning Tree Algorithm/Protocol

The spanning tree protocol is a link management protocol that provides path redundancy while preventing undesirable loops in the network. When using an Ethernet network only one active path must exist between two stations. The operations of the spanning tree protocol are transparent to endstations which cannot detect any difference between being connected with a single LAN segment or a switched LAN of multiple segments.

The spanning tree protocol allows the creation of fault-tolerant networks. It ensures that a loop-free path between all nodes in the network exists. To ensure this it calculates the best loop-free path throughout the switched network. All switches take part in this process by sending spanning-tree frames at regular intervals.

To achieve this loop-free path, first a root switch is elected which then forces all redundant paths into a blocked state using STP. If a path or the root switch fails the spanning tree algorithm recalculates the topology and activates the blocked standby path.

All switches in the layer 2 network exchange spanning tree information using data messages called Bridge Protocol Data Units (BPDUs). This exchange results in:

- *Election of the root switch.* The switch with the highest bridge priority (lowest numerical value) is elected as the root switch. If several switches are assigned the same priority, the switch with the lowest MAC address is used.

- *Election of the designated switch for every switched LAN segment* As soon as the root switch is chosen, the shortest distance to the root switch is calculated for each additional switch. Based on this information the designated switch for each segment is chosen. This is the switch which is closest to the root switch. It will be used to forward frames from this segment to the root switch.

- *Removal of loops by blocking switch ports connected to redundant links*

The BPDUs are sent and received by the bridges and used to calculate the spanning tree information. They are not forwarded by the bridges.

2.4.1 STP port states

A port on a bridge must not start forwarding packet when the bridge is turned on. This can introduce loops in the switched LAN. The port first has

to wait for the STP information to propagate through the LAN. Maybe this port is a redundant link which will be put into a blocked state.

The ports cycle through five different states:

- Blocking
- Listening
- Learning
- Forwarding
- Disabled

When turned on each port starts in the blocking state. When enabled by the STP protocol the port enters the listening state. When the forward delay timer expires the port enters the learning state and resets the timer. In the learning state the port learns MAC addresses of the connected stations, but still does not forward any frames. The port again waits for the expiration of the forward delay timer and moves to the forwarding state. This state enables learning and forwarding. In all states the port can be switched into the blocking state by the STP protocol. In all states but the disabled state, the port receives and processes BPDUs. The disabled state does not process BPDUs. The disabled state is configured by the operator.

2.5 Filtering using redundant bridges

Redundant filtering poses problems when using Netfilter. Netfilter is a stateful packet-filter. Up to now there is no possible solution to synchronize the connection tracking table on two machines using Netfilter. When using two bridges redundantly, the second bridge has no access to the connection tracking information of the first bridge when it fails.

Fortunately bridges usually do not use any network address translation (NAT). Only the actual connection information is stored in the connection tracking table. When using stateful rules Netfilter allows for automatic pickup of established connection. This is possible, because Netfilter does not enforce a new TCP packet to be a SYN packet. An ACK packet is treated as a new packet too.

When using stateful rules to allow connections through the bridge in one direction redundant bridges are possible. When the first bridge fails and eventually the second bridge picks up the forwarding of packets, the Netfilter connection tracking table will pick up the connection with the first ACK packet going in the same direction as the first SYN packet.

3 Conclusion

Firewalling using packet filters usually used a router. The packet filtering software decides weather the packet will be routed or not. Installing a firewall in an existing network architecture often required the modification of the connected networks and the associated IP addresses. If the firewall device functions as a bridge no change is needed. The firewall is just placed in between the machines to be firewalled. Additionally, the firewall is invisible on the IP-layer.

The Linux operating system is fully capable of handling this task. It even supports redundant setups using several bridges. It offers a very powerful alternative to cost-intensive commercial solutions.

References

[1] `http://bridge.sourceforge.net`

[2] `http://users.pandora.be/bart.de.schuymer/ebtables/`