Daemon.be Report

Daemon.be
Diestsebaan 27
2230 Herselt

Tel. +32 (014)54 44 15
Cell. +32 (0496)32 52 46
E-mail info@daemon.be

OS/2 TCP/IP Filtering

Foreword

One might ask himself why one would use an expensive OS/2-machine as a firewall, instead of a simple Linux-box, with ipchains/iptables or some other packetfiltering-tool.

The answer of this lies deep within the structure of OS/2 itself. From the beginning, OS/2 has been thought out as an operating system which should be rock-stable by itself, perform excellently concerning network-connections and be able to open many sockets at the same time without too much of an overhead.

This made OS/2 to be widely known as one of the most stable operating-systems ever developed. When correctly installed and tuned, according to my personal opinion, nothing meets uptime-demands as completely as a finely configured Warp installation, and if there are three things which are typical characteristics of “a good firewall”, they are stability, security and ofcourse, availability. OS/2 meets all of those requirements. Above that, configuring a warp-system is a lot simpler than having to configure yet another unix-box. So the question is not, why should I use an OS/2-firewall, the question is, why not consider it?

In this paper we will introduce a way to use the packet-filtering-procedures built into the TCP/IP-stack of OS/2 itself. This method has been inherited from the older AIX-TCP/IP-stack. It is already widely used, however, documentation for this kind of packet-filtering doesn’t grow as prosperous on the net as it should. When I first started out at building my os/2 firewall, I had to search the web for quite a bit of time until I had enough information to get me started, while still knowing what I was doing. So I decided to take my notes together and wrote a paper on the subject, so everyone can start building their firewall with all of the info at hand.

Please understand that this is only a very basic configuration, and needs lots of tuning before it can truly be used in production environments. With this in mind, have fun.

Configuring the Filter

Enabling the correct drivers

First of all, make sure that the following lines are in your config.sys . If they are not present, please add them.

```
DEVICE=C:\MPTN\PROTOCOL\FWIP.SYS
DEVICE=C:\MPTN\PROTOCOL\IPSEC.SYS
```

FWIP.SYS is the device which will receive packets, check them to the current ruleset and choose wether or not to discard them. IPSEC.SYS is used for the IPSEC-implementation on OS/2, however it is needed for the installation of your firewall, otherwise you will not be able to use the logging-feature.

Creating the configuration-file and securing the network-devices

The next step in building our packetfilter consists of the creation of a configuration file, in which we will list all IP-adresses of the interfaces we need to secure. All these interfaces will go into a deny-all stance after the next reboot. For example, if you have 3 NIC's (Network Interface Cards) in your machine, with ip's 212.100.41.20 , 192.168.35.21 & 172.0.35.22, create a file called:

\MPTN\ETC\FWSECAD.CNF

which contains :

```
212.100.41.20
192.168.35.21
172.0.35.22
```

You may choose not to add certain interfaces here. For example, if you only want your internet-connection to be filtered, just add the IP-adress you have on the internet, but not the ip's of any of the other cards.

Now we are going to create a ruleset-configfile. We will add rulesets after the next reboot, however you may choose to do so at this stage. My experience is that it is easier to build the rules while you have everything denied, so you can have a better look at the effect it has on the firewall (in production-environments however this may cause your boss to start yelling "why can't I read my mail" instead of him giving you your well-deserved promotion). If you create the rulesets before rebooting, you will only see the effect of all of them after the next reboot. Both of those methods have their own advantages and disadvantages, though.

To create the ruleset-file, make an empty file at:
\MPTN\ETC\SECURITY\FWFILTRS.CNF

Now, we reboot the machine before we proceed to the next step. This is done to load-up the drivers we added in config.sys.

Initializing the OS/2 packetfilter

After those preliminary jobs are done… we are ready to boot our firewall
For this, you open up an os/2-console-window and you enter the following command:

```
cfgfilt –u –i
```

CFGFILT is a program better known as the “Packet Filter Rules Dump Facility”. The parameter “-u” tells it to update all of the filter-rules, “-i” just initalizes the filter-device. Now it starts up and you get a whole lot of interesting status-info about your packet-filter. Normally you should have to bring up the firewall yourself, but OS/2 anticipates this, and brings it up using inetcfg –s firewall 1 . Inetcfg is used to configure parameters of the TCP/IP-stack. The only relevant portion of its usage here is the “firewall” parameter, which turns the firewall on or off. This is in a binary format, so 0 means “off”, 1 means “on”. You can also get the status of your firewall with the parameter “–g firewall” .

Now the firewall is up and running and all traffic is blocked, unless you chose to specify your rules earlier. Otherwise, now is the time to start defining rules.

Defining rules

As mentioned earlier, all rules are to be saved in the file \MPTN\ETC\SECURITY\FWFILTRS.CNF . Just open the file in any editor. Rules have a special format which offers a lot of flexibility to the administrator. You can filter packets using many different parameters.

A rule looks like this:

todo a.a.a.a. b.b.b.b c.c.c.c. d.d.d.d e f g h i j k l

And here’s the meaning of it all:

Todo	The action the rule should take. This can either be “permit” or “deny”. In case enter “permit” here, packets which match the rule, will be allowed passage through the firewall. When you enter “deny”, packets will be blocked.
a.a.a.a.	This is the source-IP-adress of the packet. For example 167.23.45.51. Please keep in mind the ending-dot. So it’s 167.23.45.51. when you enter it.
b.b.b.b.	Source address-mask, eg. 255.255.255.0
c.c.c.c.	Destination ip-address of the packet.
d.d.d.d.	Destination address-mask.
E	The protocol-type of the packet. The filter supports the following protocols: • ICMP • UDP • TCP • TCP/ack (TCP-packet with its ack-bit set) • IPSP • ALL (all of the above)
F	Here you can select the source-port. Yet again you have a wide range of options: • Any 0 *packets to any portnumber match the rule* • Eq port *Exact match to the port specified by “port”* • Neq port *Any port except for the one specified by “port”* • It port *When it matches any port number lower than “port”*

	• Gt port *When it matches any port number greater than "port"* • Le port *Any portnumber less than or equal to "port"* • Ge port *Any portnumber greater than or equal to "port"*
g	Here you specify the interface-type. It can be "secure" or "non-secure" • Secure: applies to packets flowing through a secure interface • Non-secure: applies to packets flowing through non-secure interfaces
h	The routing of the packets… "local", "route", or "both" • Local: packets flowing to or from the firewall itself • Route: packets en-route, which means they are just flowing through the firewall to other hosts. • Both: all of those little naughty packets ;)
i	Direction… although closely related to the routing, it still means something different. It doesn't mind its hosts, but just looks at the interface. Can be "inbound", "outbound" or "both" • Inbound : packets flowing *to* the interface • Outbound: packets flowing *from* that interface • Both: all of them
j	Logging-parameter. • l=yes (log it whenever such packet scrolls through) • l=no (do not log packets matching this rule)
k	Fragmentation. As you know, large packets can be fragmented on their way to your host. This parameter of your rule defines what to do with such a euhmmm… chopped packet ;) • F=yes (matches on headers, fragments ànd non-fragmented packets) • F=no (only matches non-fragmented packets) • F=only (only headers & fragments give a match)
l	Tunneling… through which tunnel should the packet be sent (if tunnel set) • T=tunnel_id

After you created all of those rules, update the filtering rules by issueing the following command on an os/2-console:

```
cfgfilt –u
```

All rules should now be updated and activated. You can, if you wish, run a plain vanilla "cfgfilt" on an OS/2-prompt to take a look at the rules. For interfaces which are secured, the last rule always is a "deny all", this means that all packets not explicitly allowed, will be forbidden.

Configuring logging

One of the most interesting aspects of a firewall is that you can make it log things that aren't really welcome. Thanks to the IPSEC-integration in OS/2, the packetfilter of the OS/2-tcp/ip-stack also has logging capabilities.

To configure logging, first of all, create a file \MPTN\ETC\FWLOG.CNF
In this file, you can define the level of logging you require. It only contains one line:
level = xx

Where xx is the level of logging you require. You have many options from 10 to 50, where 10 logs the most, 50 the least.
10 Debug Mode – log all
20 Informational Mode – logg info, warning, error, alert msg's
30 Warning Mode – logg warning, error, alert msg's
40 Error Mode – log error, alert msg's
50 Alert Mode – log alert messages only

Next, we should decide which rules should be logged. Like mentioned in the previous chapter, you add l=y to all of the rules which you want logged. This should be done in \MPTN\ETC\SECURITY\FWFILTRS.CNF

Now open an os/2-console window, and run the fssd-program, by just typing "fssd".
This will boot up the packet-filter syslogd. A log file will be created in \MPTN\ETC . Normally the filename will start with fw, followed by the date it was created, like 1225 for christmas.

Ofcourse, logging is not yet being done, as you haven't reloaded the rules yet to which you added the logging-parameter. This is to be done by the command
cfgfilt –u –d

Now all rules which you have designated to be logged, will be logged. You may inspect the logfiles by going into the \MPTN\ETC-directory (or putting it in your path ofcourse) and running the fwlslog-binary. As a parameter, you give "file=filename", for example:

fwlslog file=fw1225

to take a look at what happened on Christmas-day.