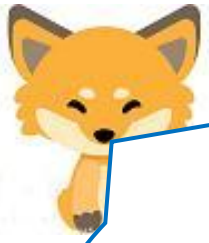
Episode 23 (Security II)

Tanuki, continuing from episode 22.
Now, let's do a practice routing exercise where packets coming in from eth7 (NIC A) are routed through eth8 (NIC B), shall we?
The objective is to access the web server in ③ from ① (terminal) and display html files. In other words, the default setting of the firewall (②) is that all servers cannot be accessed, but this is the setting when only access to your company's homepage is permitted.
Enter and execute the following on the terminal of the PC in (②).

```
# iptables –t nat –A POSTROUTING –o eth8 –s 192.168.1.0/24 –j MASQUERADE
```

There are a lot of complicated-looking commands here.
does that mean that it should be run with root privileges?
Is iptables a command that says to temporarily set up a firewall (routing, filtering) configuration?
NAT is a routing configuration. Kitsune, you can explain the rest.



OK! Tanuki, you remembered the previous explanation well.
-A is an option that says, I'm going to add a rule. The rule to add is POSTROUTING. This rule means that a private IP address belonging to the 192.168.1.0/24 network coming in from "NIC A" will be converted to a single IP address, 172.17.50.11, when leaving "NIC B". The -s (sender) option translates packets with the network address 192.168.1.0/24 coming in from eth7 to 172.17.50.11 when going out from eth8. The -o (meaning out) option is for packets going out from eth8. The -j option means that the target MASQUERADE (NAT conversion) is allowed. 24 in 192.168.1.0/24 is the prefix, which is the subnet mask. Note, however, that eth8 is assigned the IP address 172.17.50.11.

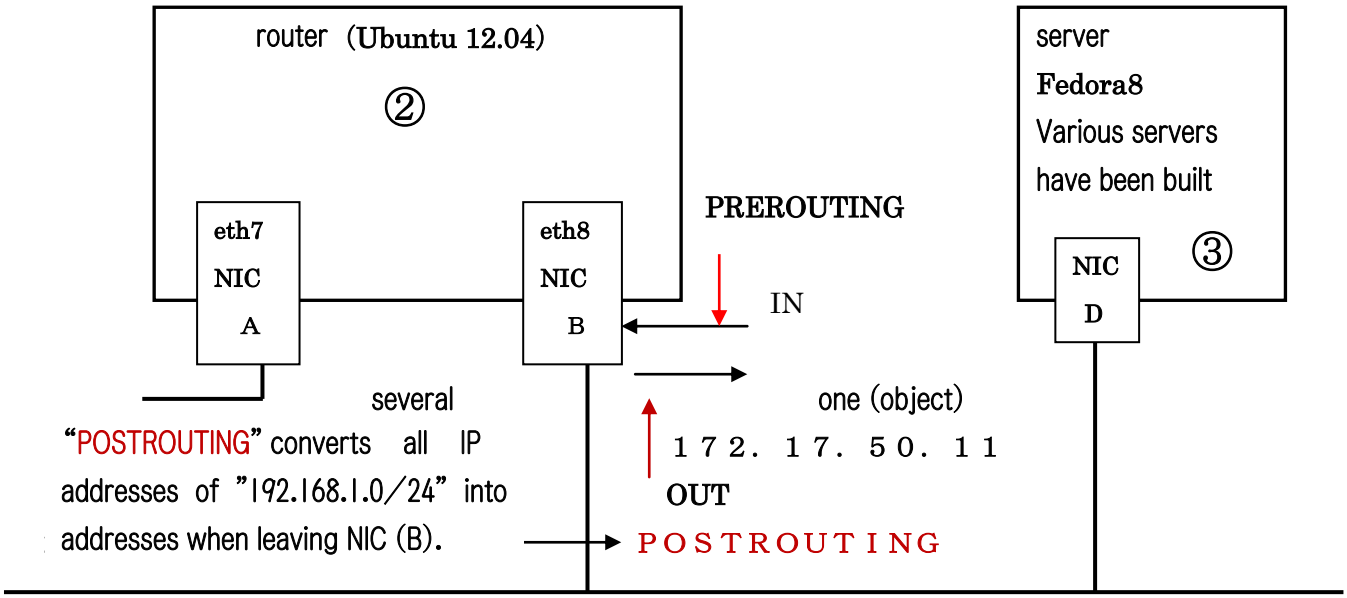
I see, by the way, what state does the IP table temporarily go into when it is executed?

In the last issue, you explained how to use the command to display the routing status of the IP table. Do you remember? I believe it was "# iptables -n nat -L".
If you run this, you should see something like this.

```
root@ubuntu12-04: ~
oruser@ubuntu12-04:~$ sudo -i
[sudo] password for oruser:
root@ubuntu12-04:~# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination

Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination

Chain POSTROUTING (policy ACCEPT)
target     prot opt source               destination
MASQUERADE  all  --  192.168.1.0/24       anywhere
root@ubuntu12-04:~#
```

The "target" in the above figure refers to MASQUERADE (NAT conversion).PROT (protocol) is ALL, which means all protocols are covered.opt (optional) is -- i.e., no options are set.
The source (sender) is all PCs belonging to the network address 192.168.1.0/24.The destination is ANYWHERE, meaning that it will be sent to all PCs with the IP address set to eth8 (172.17.50.11).
The above is illustrated in more detail in the following figure.

[Illustration.]



Kitsune, you said this is a temporary setup, what would you do to make it permanent?

Tanuki, the command "# iptables save" will make the configuration permanent. That is, the temporary setting is written to /etc/sysconfig/iptables (file) and executed each time the server is started. So, if you make a mistake, it will run incorrectly until you remove the configuration from the iptables file. I would suggest that you write to the file only after you understand the contents of the file.

So, fox, what do you do when you want to temporarily remove a temporary setting?

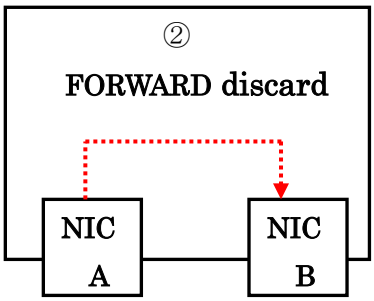
Tanuki, you can type the following command.
The option -D means Delete.

```
# iptables –t nat –D POSTROUTING –o eth8 –s 192.168.1.0/24 –j MASQUERADE
```

Let's interrupt our discussion of routing to talk about filtering. I explained in Episode 22 that the PC in ② is, by default, unfiltered, and all packets are allowed to pass through, right? Tanuki, remember, the filtering rule (also called chain) that allows packets to pass from eth7 to eth8 is FORWARD.
The configuration to filter this rule to discard (not pass) packets is shown below. I'll illustrate the explanation.

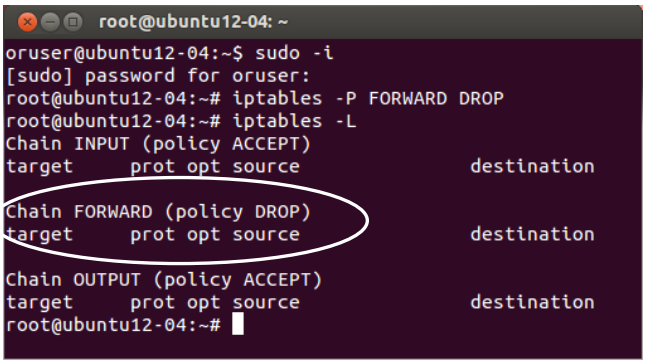
```
# iptables –t filter -P FORWARD DROP
```

[Illustration.]



I know the following order was to check the filter settings.

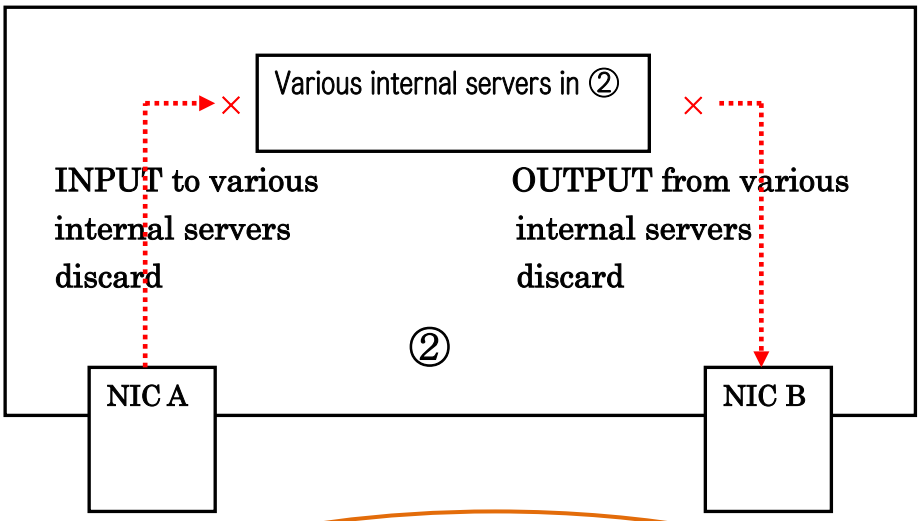
```
# iptables –t filter -L
```



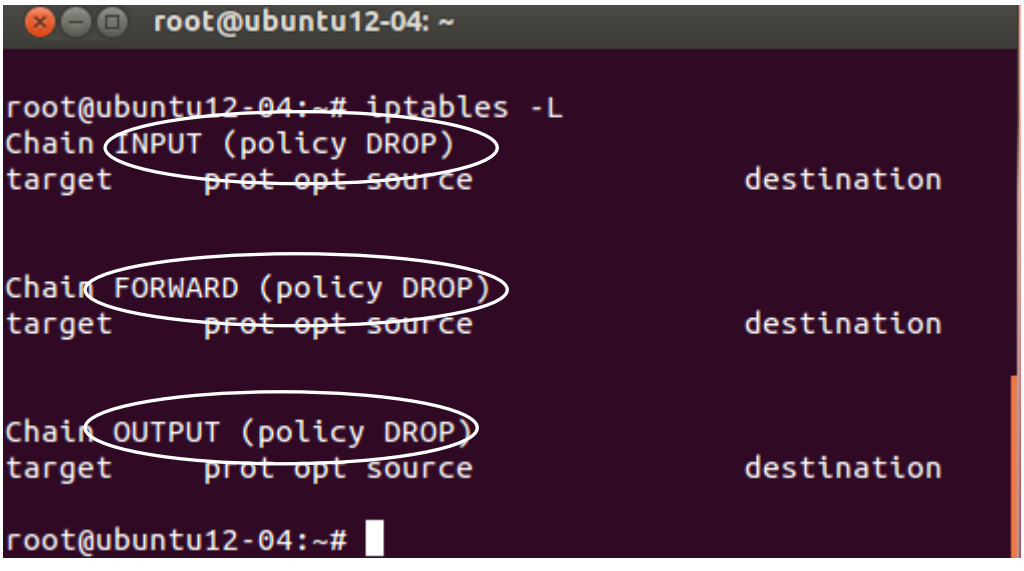
In the above figure, if the FORWARD policy is set to DROP (discard), it means that the filter is applied.

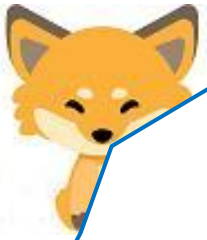
Next, we talk about INPUT and OUTPUT, the filtering rules.
This understanding is very important. In other words, the INPUT is where the packet is going in and the OUTPUT is where the packet is going out.
All of this is inside the PC in ② where the firewall is set up.
Since it is easier to illustrate, it is shown below.

[Illustration] The meaning of INPUT and OUTPUT in filtering.



iptables –t filter -P INPUT DROP
iptables –t filter -P OUTPUT DROP
If this is the case, the server (if any) inside the PC in ② cannot be accessed from the outside, and data leakage from the internal server can be prevented.
Are you convinced?
"#iptables -t fileter -L" to check the configuration.



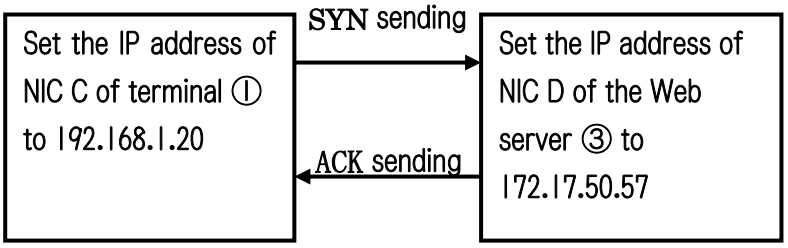


Tanuki, here comes the problem.
As shown above, the PC in ② is completely filtered and access to the server in ③ is not possible.
This is the first correct filtering setting. However, as it is, no external access at all is allowed to the server at ③. From this state, we are going to remove the setting only for the servers that allow access to external users. Now, let's set the filtering setting to ② such that only access to the web server at ③ is enabled. As shown below, two rules will be executed: one rule will allow connection request (SYN transmission) packets sent from external users to the Web server to pass, and the other rule will allow packets saying that the Web server will allow access (ACK transmission) to external users to pass.

[Unfilter the Web Server.]

```
# iptables -t filter –A FORWARD –p tcp --dport 80 –d 172.17.50.57 –j ACCEPT

# iptables -t filter –A FORWARD –p tcp ! --syn –m state --state
                                ESTABLISHED --sport 80 –s 172.17.50.57 –j ACCEPT
```

This condition is illustrated on the right.



Could you elaborate on the fox, options, etc.

I see, I still think this is a bit rambling.
I will summarize the supplementary explanation on the next page.

[Supplemental commentary]

SYN sending: The first connection request sent from the client to the server.

-t filter is omitted.

-p : Protocol Specification

tcp : TCP Protocol

--dport : destination port

80 : Port number 80 is http (web server)

-- d : Destination IP Address

172.17.50.57 : IP address of the NIC in ③

-j : target

ACCEPT : Allow packets to pass through

[Supplemental commentary].

ACK sending: Allow response packets from the server to the client.

! –syn : Allow all flags except the SYN flag.

-m state --state : Designation of status.

ESTABLISHED : One of the statuses. Connection permission for both server and client.

sport : Sender source port

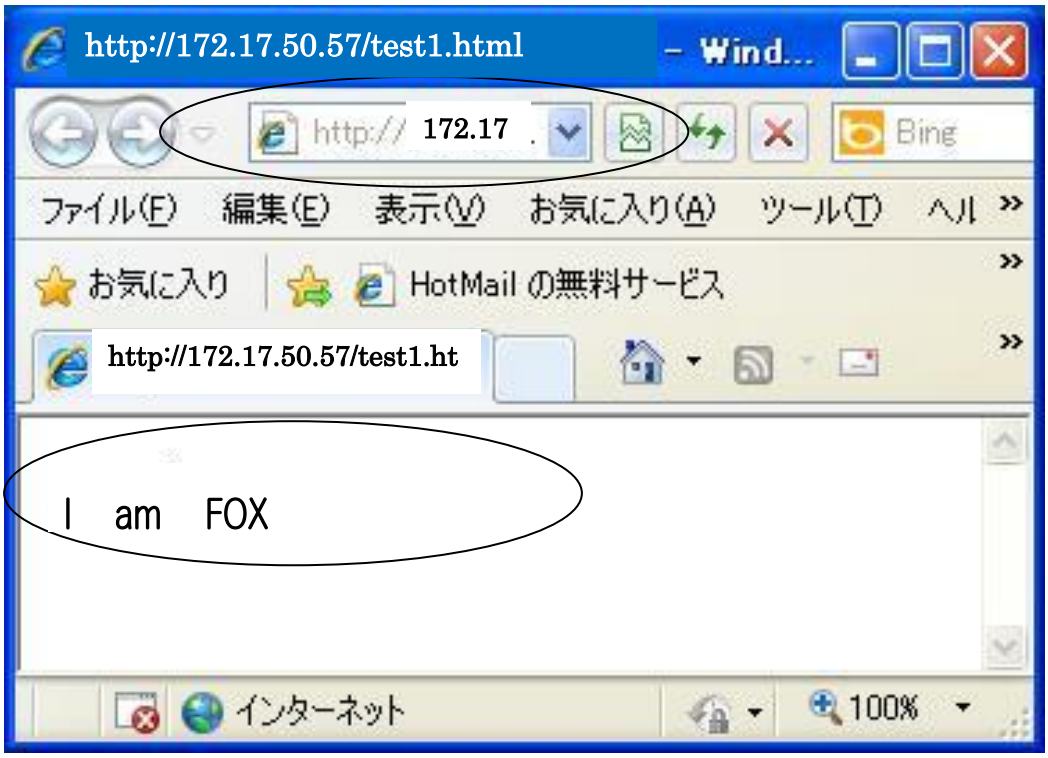
80 : Sender source (Web server) port number

-s : Specify sender source IP address

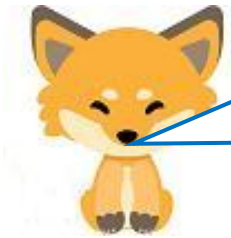
172.17.50.57 : IP address of the NIC in ③

Now I understand better. Thanks, fox.
So, how do we test if external users can now access the web server?

That's right. You need a test server both to verify that the filtering is on and to verify that it is off. This is what I mean when I say you can't practice being a burglar in a field without a home. That's why you need the ability to build a server.
But you have to make sure, right? On the web server of the PC in ③, there is a directory /var/www/html/. In that directory, you create a file called test1.html and save it. The content of the file should be something like "I am FOX". Start the browser of the PC in terminal ①, enter the URL [http://172.17.50.57/test1.html], and if the following message is displayed, it means that the file has been unlocked.



Fox, it's showing up!



Oh well, if the DNS servers were built, we would have more authentic access. In this state, the DNS server is also filtered, so it must be deactivated. If you do the following, you'll be able to access the site by domain name.

[Unfilter DNS servers.]

```
# iptables -t filter –A FORWARD –p udp --dport 53 –j ACCEPT

# iptables -t filter –A FORWARD –p udp --sport 53 –j ACCEPT
```

The 53 above is the port number of the DNS server, which opens the port. This means that we can access the DNS server. Note that the protocol is UDP, not TCP, because the UDP protocol is a connectionless protocol (it does not check for incoming packets). Now you can access the server by domain name and use the "nslookup" command at the same time.

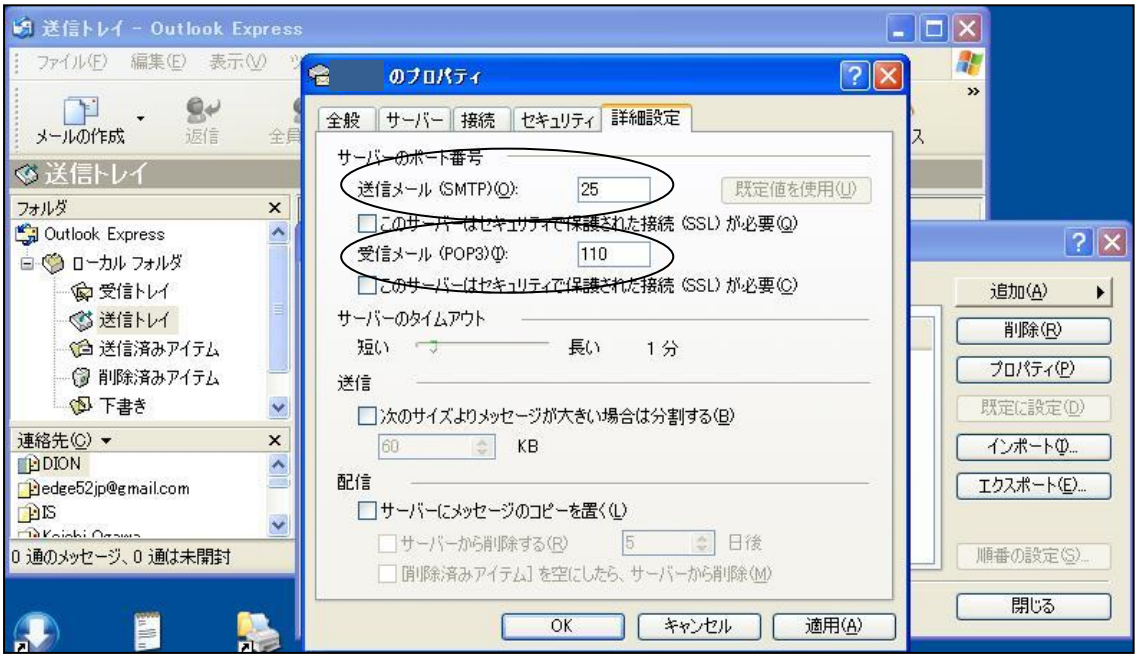
Tanuki, you should deactivate the filter so that you can use the mail server as well. The mail server has an SMTP server and a POP server, and both of these must be deactivated to be able to send and receive mail. You can deactivate them by doing the following

[Unfilter the SMTP and POP servers.]

```
# iptables -t filter –A FORWARD –p tcp --dport 25 –j ACCEPT
# iptables -t filter –A FORWARD –p tcp --sport 25 –j ACCEPT

# iptables -t filter –A FORWARD –p tcp --dport 110 –j ACCEPT
# iptables -t filter –A FORWARD –p tcp --sport 110 –j ACCEPT
```

If you run it as above, all external users will be able to use your mail server. Tanuki, don't forget to check if the filtering has been removed.

25 was the port number of the SMTP server and 110 was the port number of the POP server, right? I thought I was supposed to use an appropriate mailer to check. I use outlook express.

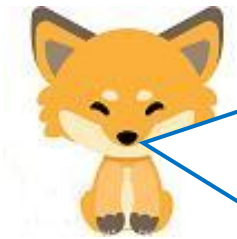


Kitsune, while you're at it, can you tell me how to set it up so that only certain external users can access the mail server for added security, rather than all users?

Roger. If you do the following, only PC ① (IP address: 192.168.1.20) will be able to access the mail server of PC ③ (IP address: 172.17.50.57).

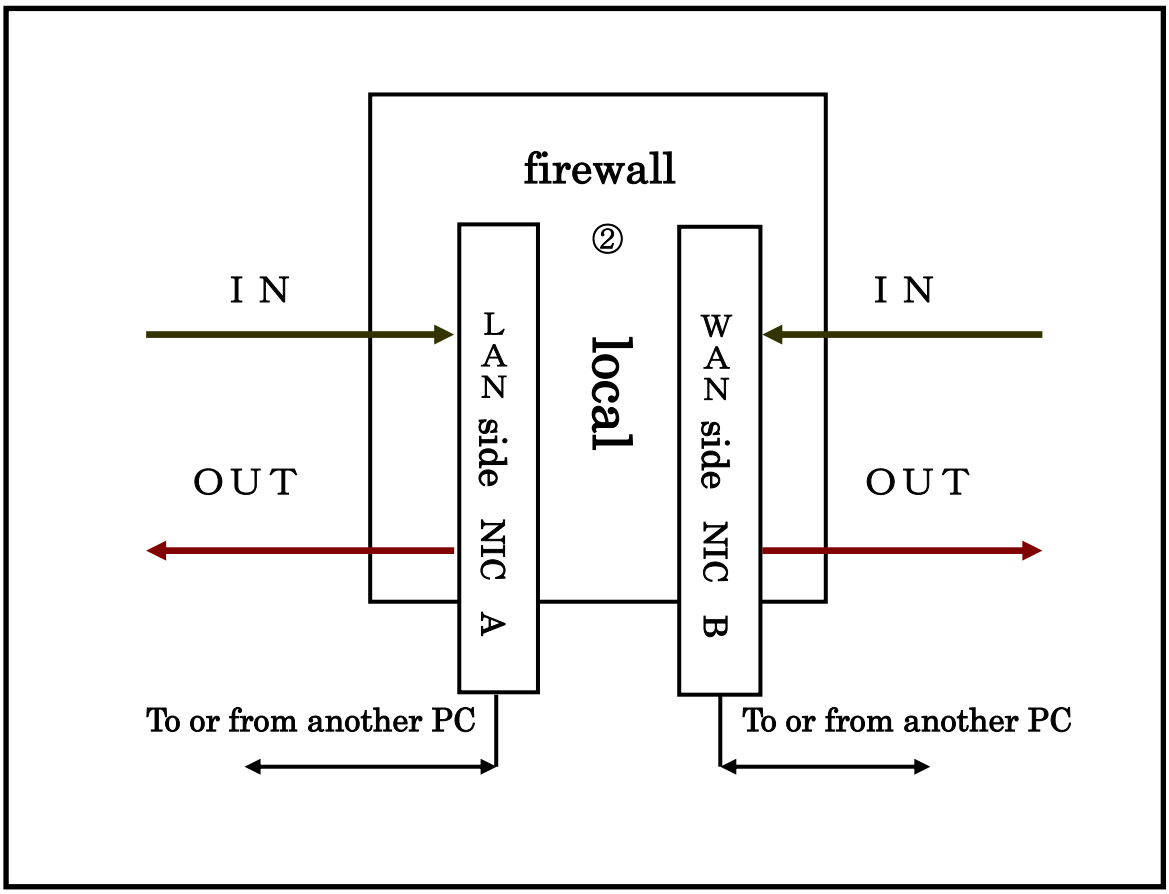
[Unfilter specific users only.]

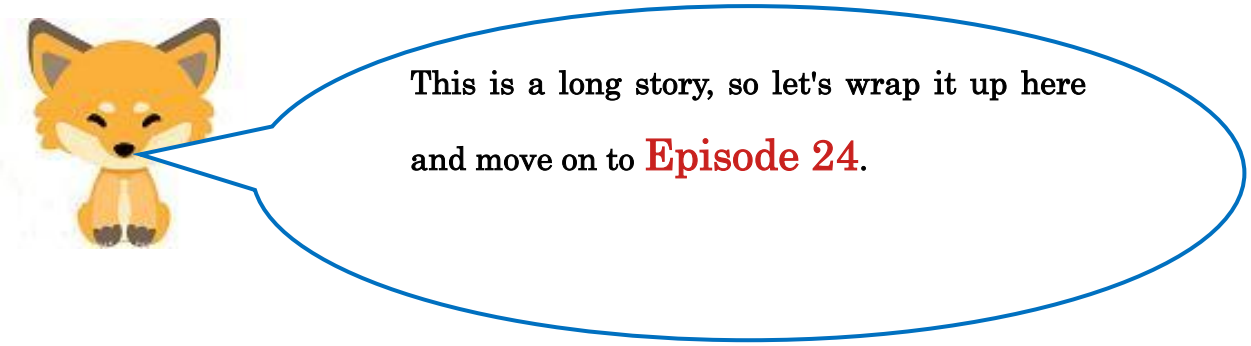
```
# iptables -t filter –A FORWARD –p tcp -s 192.168.1.20 --dport 25
                                                -d 172.17.50.57 –j ACCEPT
# iptables -t filter –A FORWARD –p tcp -s 172.17.50.57 --sport 25
                                                -d 192.168.1.20 –j ACCEPT

# iptables -t filter –A FORWARD –p tcp -s 192.168.1.20 --dport 110
                                                -d 172.17.50.57 –j ACCEPT
# iptables -t filter –A FORWARD –p tcp -s 172.17.50.57 --sport 110
                                                -d 192.168.1.20 –j ACCEPT
```



Maybe we should limit it to specific users like Tanuki said. Malicious hackers will do a port scan of your server and go after open ports with lax security, so I would suggest that you start out with full filtering and only unfilter those ports that you need.
There is one important thing to keep in mind. This is explained on the next page.

like to comment on this since it is often misunderstood!
The "-i" and "-o" options in the "-i eth7" and "-o eth8" rule settings mentioned so far refer to IN and OUT to the NIC.
Note that many students incorrectly interpret these IN and OUT options as IN coming in from the Internet to the LAN side and OUT going out from the LAN side to the Internet.
Whether on the WAN or LAN side, when a packet enters the NIC, it is IN, and conversely, when it leaves the NIC, it is OUT.
This is a major point when running iptables.
They want to make sure that the distinction between IN and OUT (the basics are the same for both router-only equipment and Linux routing) is well understood.
The illustration is as follows.





Translated at DeepL