INGI2347 : Exercises[*]
Lab session 5

Xavier Carpent, Xiao Chen

March 24, 2015

Solution 1: Design of a corporate firewall

1. Correct action. Correctly implemented.
2. Correct action. Correctly implemented.
3. The action is not correct. Even if restricting such ports will partly restrict the use of P2P, it will not prevent connections to servers (on the Internet) using 1-1023 ports. The privileges on clients (trainee desktop) do not change anything to the problem. However, the action is correctly implemented.
4. Correct action. Badly implemented: the fourth and fifth rules will also match packets of trainee laptops ! All ports lower than 1024 will thus be allowed.
5. The action is not correct. Actually this policy is not really implementable with a border firewall, packets from a computer to another one inside the network will not necessary go through *FW1*. However the given action is correctly implemented.

Solution 2: Filtering Rules for a Stateless Firewall

The filtering rules that allows the mail server to send and receive mails to and from the Internet are given below.

source	port	destination	port	protocol	`SYN` packet	action
`any`	any	`128.178.1.1`	25	`tcp`	`any`	`permit`
`any`	25	`128.178.1.1`	any	`tcp`	`no`	`permit`
`128.178.1.1`	any	`any`	25	`tcp`	`any`	`permit`
`128.178.1.1`	25	`any`	any	`tcp`	`no`	`permit`
`any`	any	`any`	any	`any`	`any`	`deny, log`

The table's first line allows any machine external to the network to connect on the mail server's port 25; thus all such TCP packets are accepted: `SYN`, `SYN-ACK` and `ACK`. The second line allows an external machine to reply to a connection attempt (thus the `SYN` packets are refused). The purpose of the third line is to allow the mail server to connect on port 25 of any machine external to the network. The fourth line allows the mail server to reply to an external machine (thus the `SYN` packets are refused). Finally, the last line prohibits all other type of traffic irrespective of the

[*]A part of these exercises comes from the book "Computer System Security". The reproduction and distribution of these exercises or a part of them are thus forbidden.

machines, ports or protocols used. This type of traffic is logged since the `log` command is present in the table, allowing us to recuperate precious data in case of a problem.

Solution 3: Filtering Rules for a Stateless Firewall

source	port	destination	port	protocol	`SYN` packet	action
`any`	any	`203.167.75.1`	22	`tcp`	`any`	`permit`
`any`	22	`203.167.75.1`	any	`tcp`	`no`	`permit`
`any`	80	`203.167.75.1`	any	`tcp`	`no`	`permit`
`any`	any	`203.167.75.1`	23	`tcp`	`any`	`permit`
`203.167.75.1`	any	`any`	22	`tcp`	`any`	`permit`
`203.167.75.1`	22	`any`	any	`tcp`	`no`	`permit`
`203.167.75.1`	any	`any`	80	`tcp`	`any`	`permit`
`203.167.75.1`	23	`any`	any	`tcp`	`no`	`permit`
`any`	any	`any`	any	`any`	`any`	`deny,log`

Solution 4: Stateless vs stateful firewalls

1.

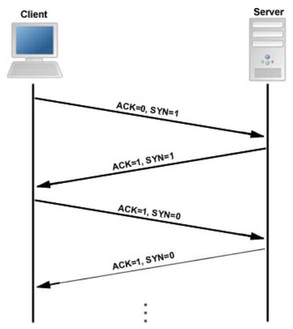

Figure 1: TCP flags

2. It is a firewall that makes filtering without keeping any state, this is analysing packet content and flags. To realize the policy « accept connections from inside to a remote host and deny connections from the outside », two solutions are used in packet filtering implementations. The first one is simply not to accept TCP packets from the outside without the `ack` bit set. The flag `ack` cannot be set in the first packet of the three-way handshake, so the connection cannot be established. The *Cisco* implementation examines this bit when the word `established` is specified. The second solution is to use the `syn` bit. *Netfilter* uses it to define the match «`--syn`» that means `SYN=1,ACK=0,RST=0` : this flag configuration is only used in the first packet of a TCP connection. Using «`!--syn`» on the input link will drop connection attempts from the outside. The only difference is that, in the second solution, a reply packet is accepted if `syn=0`, `ack=1` or `rst=1`. Packets with flags `syn=0,ack=0` or `ack=0,rst=1` will be accepted by the second solution and not by the first one.

3. A stateful firewall implements a finite state machine to control the data flows. These firewalls simply keep track of each connection thanks to the source and destination addresses and ports. The advantage of this solution in comparison with the first one is that all packet types can be analysed, even UDP and ICMP. The memory of a firewall state machine can be seen as a list of entries, one for each connection. A typical entry is composed of the protocol of the flow followed by local address, local port, remote address and remote port, and a countdown, e.g., "`udp 1.1.1.1 32000 2.2.2.2 80 60`". When a packet reaches the firewall, the system controls whether the packet flow is registered in the entry list. If so, the countdown field is reset to its initial value. Otherwise, a new entry is created if the connection is authorized. Then, the subsequent packets received for

this flow will reset the countdown at their arrival. The countdown is decremented each second; if it reaches zero, the corresponding entry is removed. This simple implementation is sufficient to observe the policy "accept the connection initiations from the inside to the outside and deny the others". To implement it, new state machine entries can only be created when a packet is leaving the network.
4. It improves dynamic changes in firewalls. In this way, restarting your firewall does not shut all connections.
5. Stateless firewalls cannot track an UDP as well as ICMP connections. Stateful firewall are able to track UDP flows under some hypothesis. They are also able to track ICMPs since they often work as a request/response scheme. Look at Netfilter documentation for more information.

Solution 5: Filtering Rules for a Stateful Firewall

When the network's architecture is complex, as is the present case, it is essential to be methodical while elaborating the filtering rules. For this we define a `dmz_proxy` zone having the proxies, a `dmz_web` zone having the Web server and finally an `internet` zone. Since the table is read sequentially until a rule is found that authorizes or prohibits the analyzed traffic, the zones have to be ordered from the most secure to the least secure: `dmz_proxy` followed by `dmz_web` and finally `internet`. It is necessary to define for each zone, filtering rules for authorized incoming traffic and for authorized outgoing traffic, and prohibit all other type of traffic. Rules that apply to two zones must appear among the rules for the more secure zone. By doing so, we make sure that a rule placed after the rules that define a zone does not have any effect on this zone. This method also has the advantage of defining unequivocally where a rule must be placed, which facilitates reading and maintenance of the table. Here is an example of the filtering table for our present case:

source	port	destination	port	protocol	action
`any`	any	`192.168.10.25`	25	`tcp`	`permit`
`any`	any	`192.168.10.80`	80	`tcp`	`permit`
`any`	any	`192.168.10.53`	53	`udp`	`permit`
`any`	any	`dmz_proxy`	any	`any`	`deny`
`192.168.10.25`	any	`any`	25	`tcp`	`permit`
`192.168.10.80`	any	`any`	80	`tcp`	`permit`
`192.168.10.53`	any	`any`	53	`udp`	`permit`
`dmz_proxy`	any	`any`	any	`any`	`deny`
`any`	any	`10.0.0.2`	80	`tcp`	`permit`
`any`	any	`dmz_web`	any	`any`	`deny`
`dmz_web`	any	`any`	any	`any`	`deny`
`any`	any	`any`	any	`any`	`deny,log`

In the table shown, the first three lines control the traffic entering the `dmz_proxy` zone and the fourth line prohibits all other type of traffic entering this zone:

`any`	any	`192.168.10.25`	25	`tcp`	`permit`
`any`	any	`192.168.10.80`	80	`tcp`	`permit`
`any`	any	`192.168.10.53`	53	`udp`	`permit`
`any`	any	`dmz_proxy`	any	`any`	`deny`

The next three lines control the traffic leaving the `dmz_proxy` zone, followed by a rule prohibiting all other type of traffic leaving this zone:

192.168.10.25	any	any	25	tcp	permit
192.168.10.80	any	any	80	tcp	permit
192.168.10.53	any	any	53	udp	permit
dmz_proxy	any	any	any	any	deny

We then proceed in a similar fashion for the dmz_web zone:

any	any	10.0.0.2	80	tcp	permit
any	any	dmz_web	any	any	deny
dmz_web	any	any	any	any	deny

It is not useful to add rules for the internet zone as they are already taken into consideration in the rules defined for the dmz_proxy and dmz_web zones. Finally, a last rule prohibiting all other type of traffic; this line is reached only when there is an error in the definition of the preceding rules, hence the reason why the traffic that reached this last rule is logged:

any	any	any	any	any	deny,log