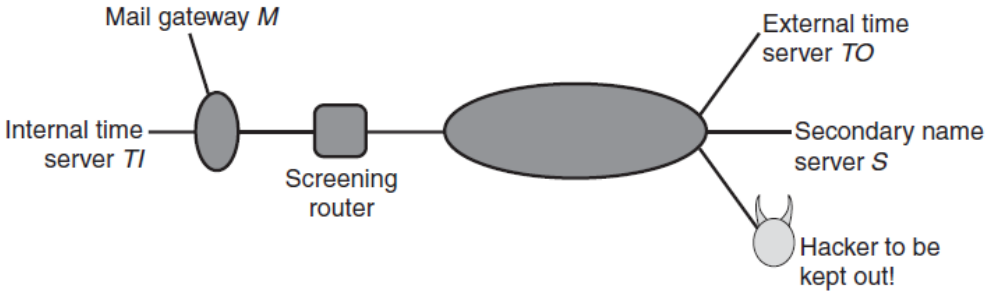
CS 7260 – INTERNET ARCHITECTURE AND PROTOCOLS

Scribe for 09/19/2012 by Anjana Ganesh

1. Oversimplified configuration of an Enterprise Firewall:



The above diagram shows an oversimplified configuration of an enterprise firewall. (In reality, firewalls are much more complex in design). It consists of the following parts:

1) **Mail gateway M (or Mail Server):** It receives mails which are going to 'NET' from the outside world. NET could be any company like gmail.com. Thus, all email addresses suffixed with @gmail.com is served by this mail server M. The mail server listens on port 25.
2) **Internal Time Server T1:** It synchronizes time on the local machine with the server's time through a protocol called the Network Time Protocol, which runs on port 123.
3) **Screening Router (Firewall):** The network is protected by the firewall. It uses a packet classification table to filter packets going to the Internet from the network, and vice versa.
4) **External Time Server T0:** Periodically synchronizes with the internal time server T1 to avoid time ambiguities.
5) **Secondary Name Server S (DNS):** The translation of domain names to IP addresses is done by the Domain Name Server or the DNS. This is necessary since it is hard for users to remember IP addresses and hard for computers to understand domain names, so we need a translation mechanism.

In a typical configuration, the mail server M also acts as an internal domain name server resolving queries and receiving mail. However, they could also be separate. If the internal DNS is unable to resolve a query, then it asks the secondary DNS to resolve it. Hackers typically try to overcome the firewall and break into the network.

2. Packet Classification Table

A packet classification table consists of rules to filter packets. The first entry in the table corresponds to the one with highest priority, and the entry in the bottom has the least priority. Each rule specifies whether a packet having a certain set of properties is to be allowed through or not.

Destination	Source	Destination Port	Source Port	Flags	Comments
M	*	25	*	*	Allow inbound mail
M	*	53	*	UDP	Allow DNS access
M	S	53	*	*	Secondary access
M	*	23	*	*	Incoming telnet
TI	*TO*	123	123	UDP	NTP time info
*	Net	*	*	*	Outgoing packets
Net	*	*	*	TCP ack	Return ACKs OK
*	*	*	*	*	Block everything!

The underlying meaning of each entry in the table is listed below.

Rule 1: If some packet wants to contact the mail server (Destination is M) and its purpose is to access mail, then allow it.
Rule 2: If some packet wants to contact the mail server and its purpose is to make a DNS query, then allow it if it is running on UDP. This is because DNS is generally performed over UDP.
Rule 3: Allow all communication between the internal DNS server and the secondary DNS server on port 53. Such exchanges could happen over UDP, Zone transfer, etc.
Rule 4: Allow all login requests to the mail server. Such requests happen through Telnet, which runs on port 23.
Rule 5: Allow all exchanges between the internal time server T1 and the external time server T0 on port 123 (Network Time Protocol) over UDP. This is necessary since time synchronization is done periodically.
Rule 6: Allow all outgoing packets from the network.
Rule 7: All all incoming packets with the TCP-ACK flag set, since this packet is an acknowledgement to a packet that was already sent from within the network.
Rule 8: Block everything else.

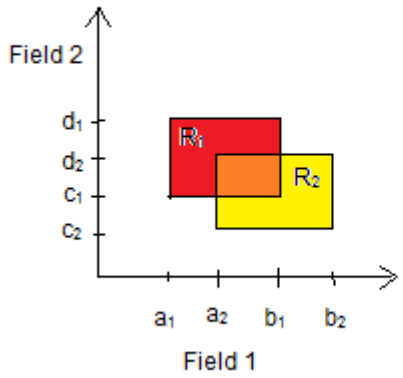
3. Packet Classification over Multiple Fields

The packet classification shown by the packet classification table filters packets based on multiple rules. This is way more complex than classifying with just two fields. Till now, the fields we used to classify packets were prefixes. We could also have a certain range of values in the field columns, as shown below:

Field 1	Field 2	Action
$[a_1, b_1]$	$[c_1,d_1]$	α_1
$[a_2,b_2]$	$[c_2,d_2]$	α_2

We could view prefixes as special ranges. For example, 11** is a range of values between 1100 and 1111. However, every range need not correspond to one single prefix.

The rules in the table above can be represented geometrically. If we consider field 1 in the X axis and field 2 in the Y axis, we have the following graph:



A rule with 2 fields can be represented by a rectangle in a two dimensional graph, and a rule with 3 fields takes the form of a cuboid. Similarly, a rule with k fields can be represented by a k dimensional cuboid in a k dimensional graph. A single rule is represented by a single point in the k-dimensional space. These results with lower boundaries have been derived from computational geometry. From this observation, we can arrive at a problem:

Given any point in a k dimensional space, which set of k-cuboids cover it and which among these has the highest priority?

A lower bound was established for this problem in 1980, and it stated that if we have N rules, with each rule having k fields, then the worst case computational complexity is $\Omega((\log_2 N)^{k-1})$, if the space complexity is O(N).

We can see that this result is very positive for k=2, in that it gives us a very feasible computational complexity of $\Omega(\log_2 N)$. But if k=5, then the computational complexity goes up to the order of $(20)^4$, which is extremely high and infeasible. Thus, we need to arrive at a trade-off between computational and space complexity. For instance, if we want to have very small computational complexity of O(1), then we need to have space complexity of $O(N^k)$. This is evident if we imagine each rule as making a cut on the overall space. If we make N cuts in a k dimensional space, we can get at most N^k pieces. The minimum space complexity we can achieve is O(N) since we need to store each rule at least once.

The bound for the computational complexity here looks quite hopeless, but it turns out that in the real world, rule sets do not have such worst-case complexity. There are certain patterns in these rule sets that allow us to reduce the computational complexity.

There are certain properties of rule sets (specified by the author George Varghese, but not confirmed) that allow us to reduce the computational complexity even for high values of k such as 5, and they have been summarized below.

a) **Prefix containment is rare.**

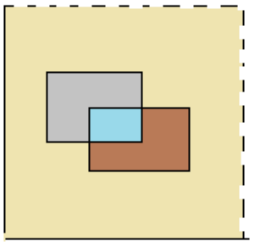
This means that we rarely have rules which specify actions on prefixes in which one prefix is contained in the other. For example, it is rare for the prefixes 1011* and 101* to have different actions specified.

b) Many fields are not general ranges.

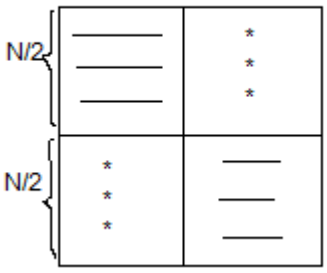
If rules contain prefixes, then they are not general ranges since prefixes constitute very specific ranges. Also, a specific value (such as 25 for destination port) is not a general range. A wildcard is also not a general range since it could be anything. Since a typical rule consists of prefixes, values and wildcards, we can see very few general ranges in a field.

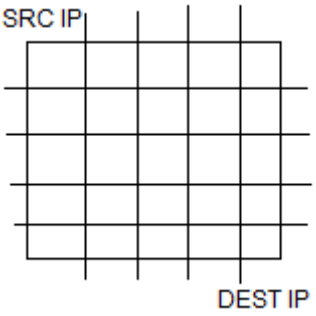
c) The number of disjoint classification regions is small.

Let us take the example of cutting a cake, and let each rule in a rule set correspond to one cut of the cake.



Two cuts produce four pieces of cake. Each piece is shown in the diagram by a differently colored region. This observation states that the number of pieces of cake is small, usually of the order of O(N) (i.e. in real rule sets). It is only in the worst case that we can cut out $O(N^k)$ pieces. This is evident from the following example. Suppose we have N rules, and we make a single cut that divides the set into two sets of N/2 rules each, as shown:





The first half has specific values in the first field and wildcards in the second field, and vice versa for the second rule. We saw that such a situation is the worst case in the Naïve Algorithm 1. If the first field is the SRC IP and the second the DEST IP, then we can view the cuts being made as shown in the diagram above. The number of pieces in this case is $O(N^2)$.

If we have, say 5 fields, then we can depict the scenario using the following table.

RA	*	*	*	*
*	RA	*	*	*
*	*	RA	*	*
*	*	*	RA	*
*	*	*	*	RA

RA represents range. So if we make 5 cuts, then we can see that N/5 rules have general range in field 1, N/5 rules have a general range in field 2, and so on. We have $O((N/5)^5)$ pieces of cake in this case. Although this is theoretically possible, it is not possible with real world rule sets.

d) Many of them are essentially Source-Destination matching

Given any random source and destination IP address pair, we can only find a few rules that match the given pair, and we can perform our search on these rules.

e) Almost all prefixes are of length 24 and below

In a real world rule set, the prefixes encountered have length of 24 bits or less.