

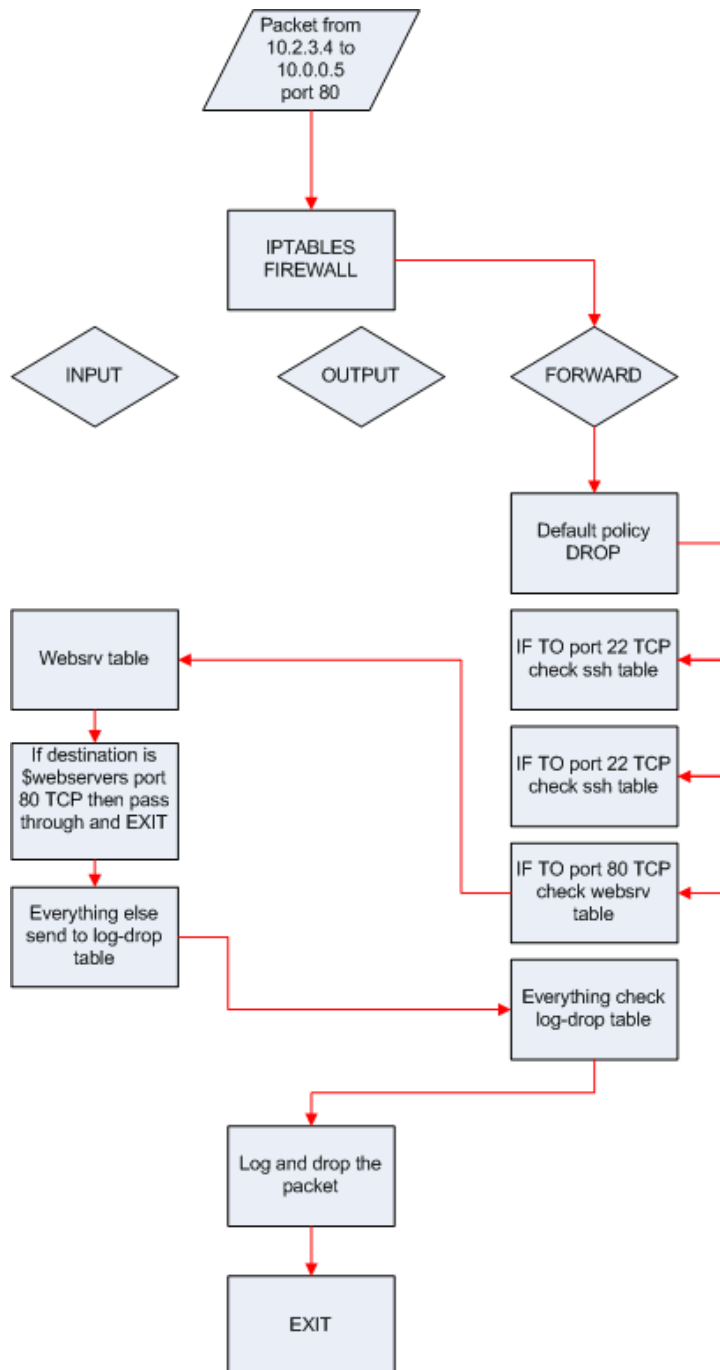
IPTABLES explained : Part 2 (or a how to for iptables about efficient rule design)

So , in the last post or tutorial (here) we went through how IPTables works and how to start writing your own script.

In this post we will go through thinking and planning your script efficiently and in the next post I will show you how to write a quite complex script already so you see how to go about it. In a later part we will then go through the different Distributions and how to apply your rules at every boot.

Ok, we got it pretty clear in Part 1 that you have to start your script with the variables and then zeroing and flushing your current chains.

Now, if you went through the IP filtering firewalls tutorial, you know how the rules work, however with the next graphic I would like to explain how the process with IP Tables works. Basically it works the same way, except that IP Tables gives you the option to make groups of rules for specific things. These groups are called chains. It goes so that you have the normal chains (INPUT; OUTPUT etc.) and you can attach small rulesets to each main chain. These rulesets only get checked IF a certain characteristic is seen in the traffic. say, if you get traffic to a port 80 (www) to one of your machines, and you have a chain that has all the allowed traffic for port 80, then the packet will get checked if it matches anything in the port 80 ruleset. If the traffic is destined for port 22 (ssh) the port 80 ruleset will be ignored. This approach gives you a LOT of flexibility of creating VERY complex and tight firewall rulesets WITHOUT taking a performance hit in any sense.



The key to efficient rules and easy management of such is to analyze the traffic you will expect on your firewall and design from there. This post is more like a "best practices" or "practical approach". You can for example, collect all the web servers in a variable at the beginning of the script like this:

```
#!/bin/sh  
#variables first!  
ipt="/sbin/iptables"  
std_ports="22,80,443"  
lan="10.0.0.0/24,192.168.0.0/24"  
webservers="10.0.0.3/32, 10.0.0.4/32, 192.168.0.5/32"  
any="0.0.0.0/0"  
  
#flush the tables!  
$ipt -F  
$ipt -Z  
$ipt -X  
  
#policies next!  
$ipt -P INPUT -j DROP  
$ipt -P FORWARD -j DROP  
$ipt -P OUTPUT -j DROP
```



As you can see the variable `webservers` contains 3 IPs. The reason for the `/32` is to do with subnet masking. This means it is a single host. It is much easier not to write a single rule for web servers with that variable then to write the same rule 3 times for each IP. When the rules are executed however it is still 3 rules as it will allow traffic to all 3 IPs. It is just easier to manage this way.

Efficient rule sets are made by "bundling" and thinking of the big picture. For example to drop a packet you do not need a rule after each rule to drop the packet if it does not match. You create a table with rules and the packet will go through all the rules until the end of the table. There you have a pointer to the log and drop chain. Also for each service you are allowing or want to allow it might SEEM more work to create a separate chain but it really isn't in the long run. if you ever have to add another web server IP or another ssh server or change your ip-range etc. if you have it in variables and bundles things go MUCH smoother. If it helps you sit down and draw the flow or write down what IPs and services you are allowing and look at this as a design opportunity, not as a firewall rule set. Be creative, make things simpler and easier.

If you have any ideas on how to do this, please share them and if you have more questions or other comments, please ask away.