IP ACCOUNTING

- The main objective of IP accounting is to know how much data is being transmitted and received on a network.

- For an Internet Service Provider this is used to charge customers by volume of data transferred.

- Technical reasons for doing this have to do with traffic management on the servers within a network.

- For example it would be very useful to determine how much data is being generated by each network service offered on a server. This information could then be used to scale hardware to meet the demand.

- The Linux kernel provides a facility that allows you to collect a lot of useful information about the network traffic it is handling. This facility is called **IP accounting.**

Configuring IP Accounting

- IP accounting is closely related to IP firewall, therefore we use the same tools to configure it as we did with IP firewalls.

- We can use ***iptables*** are used to configure IP accounting. The command syntax is very similar to that for firewall rules, the only difference being that we use the commands now to gather information on the nature of network traffic.

- The general command syntax is*:*

    # iptables –A chain rule-specification

- The commands are the same as firewall rules, except that the policy rules do not apply in the case of IP accounting.

- We can add, insert, delete, and list accounting rules. Using ***iptables,*** all valid rules are accounting rules, and any command that doesn't specify the ***-j*** option performs accounting only.

- The rule specification parameters for IP accounting are the same as those used for IP firewall. These are what we use to specify the type of network traffic we wish to keep count of and the tallies.

Accounting by Address

- Let us suppose that we have a Linux-based router that serves two networks on a campus. The router has two Ethernet devices, **eth0** and **eth1**, each of which services a network.

- We will also assume that we have a third card (**eth2**) that serves a connection from our campus to a remote campus.

- For management purposes we want to know the total traffic generated between each of networks on the same campus and for billing purposes we wish to know the traffic generated across the third remote link.

- The following table shows the interface addresses we will use in our example:

Interface	Address	Netmask
eth0	142.232.66.0	255.255.255.0
eth1	142.232.100.0	255.255.255.0
eth2	142.232.168.0	255.255.240.0

- To determine the amount of data each network generates on the remote link we could use a rule as follows:

- Using ***iptables***:

```
# iptables -A FORWARD -i eth2 -d 142.232.66.0/24
# iptables -A FORWARD -o eth2 -s 142.232.66.0/24
# iptables -A FORWARD -i eth2 -d 142.232.100.0/24
# iptables -A FORWARD -o eth2 -s 142.232.100.0/24
```

- These sets of rules will count all data traveling in either direction across the interface named **eth2** with a source or destination address shown.

- To determine the amount of data travelling between the two networks we would use a rule as follows:

- Using ***iptables***:

```
# iptables -A FORWARD -s 142.232.66.0/24 -d 142.232.100.0/24
# iptables -A FORWARD -s 142.232.100.0/24 -d 142.232.66.0/24
```

- These rules will count all packets with a source address belonging to one of the networks and a destination address belonging to the other.

Accounting by Service Port

- Let us suppose we want to know much of the link FTP, smtp, and World Wide Web services are consuming.
- A script of rules to enable us to collect this information might look like:
- Using ***iptables***:

```
# !/bin/ sh
# ftp, smtp and www statistics using iptables.

# iptables -A FORWARD -i eth0 -m tcp -p tcp --sport ftp-data:ftp
# iptables -A FORWARD -o eth0 -m tcp -p tcp --dport ftp-data:ftp
# iptables -A FORWARD -i eth0 -m tcp -p tcp --sport smtp
# iptables -A FORWARD -o eth0 -m tcp -p tcp --dport smtp
# iptables -A FORWARD -i eth0 -m tcp -p tcp --sport www
# iptables -A FORWARD -o eth0 -m tcp -p tcp --dport www
```

- The syntax "***ftp-data:ftp***" means "ports ftp-data (20) through ftp (21)", and is used to encode ranges of ports in ***iptables***.
- When we specify a list of ports in an accounting rule, it means that any data received for any of the ports in the list will cause the data to be added to that entry's totals.
- Recall that the FTP service uses two ports, the command port and the data transfer port, we've added them together to total the FTP traffic.
- Now suppose that we classify FTP, SMTP, and WWW traffic as essential traffic, and all other traffic as nonessential. We are now interested in seeing the ratio of essential traffic to nonessential traffic.
- Recall that ***iptables*** allows only one argument in the port specification. In order to accomplish the above task we can exploit user-defined chains in accounting just as easily as in firewall rules.
- The ***iptables*** implementation would look like:

```
# iptables -N ess-traffic
# iptables -N noness-traffic
# iptables -A ess-traffic -j ACCEPT
# iptables -A noness-traffic -j ACCEPT
# iptables -A FORWARD -i eth0 -m tcp -p tcp --sport ftp-data:ftp -j ess-traffic
# iptables -A FORWARD -i eth0 -m tcp -p tcp --sport smtp -j ess-traffic
# iptables -A FORWARD -i eth0 -m tcp -p tcp --sport www -j ess-traffic
# iptables -A FORWARD     -j noness-traffic
```

- We have created two user-defined chains, one called ***ess-traffic***, where we capture accounting data for essential services and another called ***noness-traffic***, where we capture accounting data for nonessential services.

- We then add rules to our forward chain that match our essential services and jump to the ***ess-traffic*** chain, where we have just one rule that accepts all datagrams and counts them.

- The last rule in the forward chain jumps to the ***noness-traffic*** chain, where we have just one rule that accepts all datagrams and counts them.

- The rule that jumps to the ***noness-traffic*** chain will not be reached by any of our essential services, as they will have been accepted in their own chain.

- Thus, the tallies for essential and nonessential services will be available in the rules within those chains.

- Recall that the MTU value defines the largest datagram that will be transmitted on a network device.

- When a datagram is received by a router that is larger than the MTU of the interface that needs to retransmit it, the router fragments the large datagram into small pieces no larger than the MTU of the interface and then transmits these pieces.

- The router builds new headers to put in front of each of these pieces, and these are what the remote machine uses to reconstruct the original data.

- During the fragmentation process the port is listed in only the first fragment. This means that the IP accounting can't properly count all fragmented datagrams.

- The Netfilter implementation has a solution for this. We use:

```
# iptables -A FORWARD -i eth0 -m tcp -p tcp -f
```

- These won't tell us what the original port for this data was, but at least we are able to see how much of our data is fragments, and be able to account for the volume of traffic they consume.

Accounting of ICMP Datagrams

- The ICMP protocol does not use service port numbers and is therefore a little bit more difficult to collect details on.
- ICMP uses a number of different types of datagrams, mostly for control and troubleshooting purposes.
- A very common and malicious denial of service attack is to deny network access to users by generating large numbers of ICMP messages. This is commonly called **ping flooding.**
- While IP accounting cannot do anything to prevent this problem (IP firewalling can help) we can at least put accounting rules in place that will detect this attack.
- ICMP doesn't use ports as TCP and UDP do, instead it has ICMP message types. Therefore we can build rules to account for each ICMP message type.
- To do this, we place the ICMP message and type number in place of the port field in the accounting commands.
- An IP accounting rule to collect information about the volume of ping data that is being sent to you or that you are generating might look as follows:

```
 # iptables -A FORWARD -m icmp -p icmp --sports echo-request
# iptables -A FORWARD -m icmp -p icmp --sports echo-reply
# iptables -A FORWARD -m icmp -p icmp -f
```

- The first rule collects information about the "ICMP Echo Request" packets (ping requests), and the second rule collects information about the "ICMP Echo Reply" packets (ping replies).
- The third rule collects information about ICMP datagram fragments. This is similar to the rule described earlier for fragmented TCP and UDP datagrams.
- By specifying source and/or destination addresses in the rules, we can keep track of where the pings are coming from, such as whether they originate inside or outside your network.
- Once we have determined the source of the rogue packets we can implement firewall rules to block them.

Accounting by Protocol

- Suppose we are interested in knowing how much of the traffic on our link is TCP, UDP, and ICMP.

- We would use rules like the following:

```
# iptables -A FORWA.RD –i eth0 -m tcp -p tcp
# iptables -A FORWARD -o eth0 -m tcp -p tcp
# iptables -A FORWARD –i eth0 -m udp -p udp
# iptables -A FORWARD -o eth0 -m udp -p udp
# iptables -A FORWARD –i eth0 -m icmp -p icmp
# iptables -A FORWARD -o eth0 -m icmp -p icmp
```

- Using these rules the traffic flowing across the eth0 interface will be analyzed to determine whether it is TCP, UDP, or ICMP traffic, and the appropriate counters will be updated for each.

Listing Accounting Data with *ipchains* and *iptables*

- The ***iptables*** commands will not display our accounting data (packet and byte counters) unless we supply it the ***-v*** argument.

- The simplest means of listing our accounting data is as follows:

```
# iptables -L –n -v
```

- We can display the packet and byte counters in units by using the expanded output mode using the -x argument:

```
# iptables -L –n -v -x
```

Resetting the Counters

- The IP accounting counters will overflow if left to run indefinitely. After overflow there is now way of determining the value they actually represent.

- To avoid this problem, you should record the accounting data periodically and then reset the counters back to zero to begin collecting accounting information for the next accounting period.

- This can be done very simply:

```
# iptables -Z
```

- We can combine the list and zeroing actions together to ensure that no accounting data is lost in between:

```
# iptables -L –n -Z -v
```

- These commands will first list the accounting data and then immediately zero the counters and begin counting again.

- A good idea would be to put these commands into a script that record the output and stored it in a file, and execute the script periodically using the ***cron*** command.

Flushing the Ruleset

- Flush all the IP accounting rules is most useful when you want to modify the ruleset without rebooting the machine.
- The ***-F*** argument is used to accomplish this:

```
# iptables -F
```

- This flushes all of your configured IP accounting rules, removing them all and saving us the effort of having to remove each of them individually.

Passive Collection of Accounting Data

- If your Linux machine is connected to an Ethernet, you can apply accounting rules to all of the data from the segment, not only that which it is transmitted by or destined for it.
- Your machine will passively listen to all of the data on the segment and count it.
- Make sure you first turn IP forwarding off on your Linux machine so that it does not try to route the datagrams it receives (assuming it is not a router!).
- Enable promiscuous mode on your Ethernet interface using the ***ifconfig*** command. Now you can establish accounting rules that allow you to collect information about the datagrams flowing across your Ethernet.