System Monitoring and Intrusion Detection

- System monitoring is essential to verifying that the system is running as we expect and only running those network processes that the administrator allows.
- There are several tools that will allow us to monitor the network status, as well as inbound and outbound traffic.
- The first step is to ensure that the only programs and services running are the ones you expect. This includes correlating open ports to known services.
- The next step is to interpret what monitoring and the firewall tools are telling us when they report status and error messages in the system log files.
- It is important to understand that given the complexity of current UNIX and Linux systems we can only achieve a reasonable degree of confidence that the system is running as we expect it run.
- This is mainly due to the fact that the configuration issues cross too many boundaries, to ever be absolutely certain that everything is running correctly.
- The system logs are the key in helping us detect and identify any anomalous behavior and rogue processes.

Using *ifconfig*

- ***ifconfig***'s primary job is to configure and activate the network interfaces. It is executed from a network startup script at boot up time.
- It is a useful debugging tool for reporting the status of the network interfaces. Invoked without any arguments it reports the status of all active network interfaces.
- With the ***-a*** option, ***ifconfig*** reports the status of all network interfaces, active or not. The following is a sample output for a machine with a single network interface card.

```
eth0    Link encap:Ethernet  HWaddr 00:01:02:45:45:5B
        inet addr:192.168.1.20  Bcast:192.168.1.255 Mask:255.255.255.0
        UP BROADCAST RUNNING PROMISC MULTICAST  MTU:1500  Metric:1
        RX packets:1449310 errors:0 dropped:0 overruns:0 frame:0
        TX packets:19866 errors:0 dropped:0 overruns:0 carrier:0
        collisions:232 txqueuelen:100
        Interrupt:5 Base address:0xb400

lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
        UP LOOPBACK RUNNING  MTU:16436  Metric:1
        RX packets:70 errors:0 dropped:0 overruns:0 frame:0
        TX packets:70 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
```

- The third line of each interface's report indicates the interface's status and configuration. In this example, eth0 is active, indicated by UP.

- The MAC hardware Ethernet address, **Hwaddr**, the IP address, **inet addr**, the broadcast address, **Bcast**, and the network mask, **mask**, are also reported.

- Other generally less useful information, such as the default maximum frame size, **MTU**, number of packets received, **RX**, and number of packets transmitted, **TX**, is also reported.

- Also note that the output is reporting that the network interface has both multicasting (MULTICAST) and promiscuous (PROMISC) modes enabled.

- Multicasting is useful when accessing services such as live web broadcasts, live music servers, etc.

- Promiscuous mode is of particular interest to us as far as network security is considered. An interface operating in promiscuous mode is capable of receiving all packets on the subnet not just packets that were intended for it.

- An attacker who has gained root access will usually run a sniffer on the compromised machine in order to gain more access to your network (for example, to retrieve passwords to other hosts on the network).

- A "sniffer" is a piece of software that monitors all network traffic, unlike a standard network station that only monitors network traffic sent explicitly to it. Therefore it requires the network interface to operate in promiscuous mode.

- The security threat presented by sniffers, is their ability to catch all going network traffic, including passwords or other sensitive information.

- Detecting the process that does the sniffing is difficult, because the name of that process can be disguised as something innocent. It can even be a Trojan horse version of a known binary.

- The only way to detect the sniffer in this case is to check if the network interface is in promiscuous mode. A machine should not operate in promiscuous mode unless it is for a very good reason (such as running an intrusion detection tool), and therefore promiscuous mode is a strong indication that a sniffer is running.

Checking the Network Connection with *ping*

- For testing basic network connectivity, ping is a simple tool that can be used. If the external network interface is up but you can't connect to a remote host, ping can indicate whether packets are passing through the interface.

- A negative ping response does not necessarily indicate that a remote site is down. The site might not be honoring Echo Request ICMP messages. A positive ping response proves that packets are being transmitted and the remote host is responding.

- With only a hostname or an IP address as an argument, ping sends packets indefinitely until you kill the process, at which point ping reports its final summary statistics:

  $ ping zeus.olympus.net

  PING zeus.opus42.net (192.168.1.20): from 192.168.1.5 : 56(84) bytes of data.
  64 bytes from 192.168.1.5: icmp_seq=0 ttl=253 time=4.2 ms
  64 bytes from 192.168.1.5: icmp_seq=l ttl=253 time=4.4 ms
  64 bytes from 192.168.1.5: icmp_seq=2 ttl=253 time=4.1 ms
  64 bytes from 192.168.1.5: icmp_seq=3 ttl=253 time=5.4 ms
  64 bytes from 192.168.1.5: icmp_seq=4 ttl=253 time=3.9 ms

  --- zeus.olympus.net ping statistics --5 packets transmitted, 5 packets received, 0% packet loss round-trip min/avg/max = 3.9/4.4/5.4 ms

- A remote host might not respond to ping's Echo Request messages even if the host is up. The reason for this is that ping has a long history of use as a hacker's denial-of-service tool.

- Most if not all routers these days are configured to block ping requests because attackers use it as a reconnaissance tool to map networks.

Checking the Network Connection with *traceroute*

- ***traceroute*** is a utility that shows a route over the network between two systems, listing all the intermediate routers a connection must pass through to get to its destination.
- It can be used to determine why connections to a given server might be slow, and can often help determine out where exactly the problem is.
- It also shows how systems are connected to each other, letting us see how an ISP connects to the Internet as well as how the target system is connected.
- It is a relatively simple utility to use, on a Unix system we can invoke traceroute at the command line as follows:

  traceroute server.name

- The Windows equivalent is called ***tracert***. Open a DOS window and enter the command:

  tracert server.name

- You can also download VisualRoute, a graphical traceroute program available for Windows, Sparc Solaris, and Linux. VisualRoute helps you analyze the traceroute, and provides a world map showing you where your packets are going (though not always geographically accurate).

- Consider the following example traceroute output:

```
traceroute to library.airnews.net (206.66.12.202), 30 hops max, 40 byte
packets
 1  rbrt3 (208.225.64.50)  4.867 ms  4.893 ms  3.449 ms
 2  519.Hssi2-0-0.GW1.EWR1.ALTER.NET (157.130.0.17) 6.918 ms 8.721 ms
16.476 ms
 3  113.ATM3-0.XR2.EWR1.ALTER.NET (146.188.176.38)  6.323 ms 6.123 ms
7.011 ms
 4  192.ATM2-0.TR2.EWR1.ALTER.NET (146.188.176.82)  6.955 ms 15.400 ms
6.684 ms
 5  105.ATM6-0.TR2.DFW4.ALTER.NET (146.188.136.245) 49.105 ms 49.921 ms
47.371 ms
 6  298.ATM7-0.XR2.DFW4.ALTER.NET (146.188.240.77)  48.162 ms  48.052 ms
47.565 ms
 7  194.ATM9-0-0.GW1.DFW1.ALTER.NET (146.188.240.45)  47.886 ms  47.380
ms  50.690 ms
 8  iadfw3-gw.customer.ALTER.NET (137.39.138.74)  69.827 ms  68.112 ms
66.859 ms
 9  library.airnews.net (206.66.12.202)  174.853 ms  163.945 ms  147.501
ms
```

- The command is tracing the route to library.airnews.net. The first line of output is information showing what the command is doing; it shows the target system, that system's IP address, the maximum number of hops that will be allowed, and the size of the packets being sent.
- Then there is one line for each system or router in the path between the source and the target system.
- Each line shows the name of the system (as determined from DNS), the system's IP address, and three *round trip times* in milliseconds.
- The round trip times (or RTTs) tell us how long it took a packet to get from me to that system and back again, called the *latency* between the two systems.
- By default, three packets are sent to each system along the route, so we get three RTTs.
- Sometimes a line in the output may have one or two of the times missing, with an asterisk where it should be. For example:

```
 9  host230-142.uuweb.com (208.229.230.142)  12.619 ms * *
```

- In this case, the machine is up and responding, but for whatever reason it did not respond to the second and third packets.
- This does not necessarily indicate a problem; in fact, it is usually normal, and just means that the system discarded the packet for some reason.
- Sometimes we see an entry with just an IP address and no name:

```
   1  207.126.101.2 (207.126.101.2)  0.858 ms  1.003 ms  1.152 ms
```

- This simply means that a reverse DNS lookup on the address failed, so the name of the system could not be determined.
- Sometimes a trace ends in all timeouts:

```
12  al-fa3-0-0.austtx.ixcis.net (216.140.128.242)  84.585 ms  92.399
ms  87.805 ms
13  * * *
14  * * *
15  * * *
```

- This indicates that the target system could not be reached. More accurately, it means that the packets could not make it there and back; they may actually be reaching the target system but encountering problems on the return trip.
- This is possibly due to some kind of problem, but it may also be an intentional block due to a firewall or other security measures, and the block may affect traceroute but not actual server connections.

- A trace can end with one of several error indications indicating why the trace cannot proceed. In this example, the router is indicating that it has no route to the target host:

```
 4  rbrt3.exit109.com (208.225.64.50)  35.931 ms !H *  39.970 ms !H
```

- The **!H** is a "host unreachable" error message (it indicates that an ICMP error message was received). The trace will stop at this point.

- Possible ICMP error messages of this nature include:

  !H : Host unreachable. The router has no route to the target system.
  !N : Network unreachable.
  !P : Protocol unreachable.
  !S : Source route failed. You tried to use source routing, but the router is configured to block source-routed packets.
  !F : Fragmentation needed. This indicates that the router is misconfigured.
  !X : Communication administratively prohibited. The network administrator has blocked traceroute at this router.

- Recall that the three numbers given on each line of output show the round trip times (latency) in milliseconds.

- Smaller numbers generally mean better connections. As the latency of a connection increases, interactive response suffers.

- Download speed can also suffer as a result of high latency (due to TCP windowing), or as a result of whatever is actually causing that high latency.

- Typically, a modem connection's inherent latency will be around 120-130ms. The latency on an DSL or cable connection is usually around 20-40ms.

- A large jump in latency from one hop to the next could indicate a problem. It could be a saturated (overused) network link; a slow network link; an overloaded router; or some other problem at that hop.

- It could also be a problem anywhere on the return route from the high-latency hop as well. You can use the ping program (described below) to get a better idea of the latency as well as the packet loss to a given site or router; traceroute only does three probes per router (by default), which isn't a very good sample on its own.

- A jump in latency can also indicate a long hop, such as a cross-country link or one that crosses an ocean. A long line is naturally going to have higher latency than a short one. For example:

```
 4  core1.telehouse.level3.net (195.66.224.77)  2.355 ms  4.932 ms
3.473 ms
 5  core1.London1.Level3.net (212.113.2.65)  2.550 ms  1.934 ms  3.110
ms
 6  atm10-0-100.core1.NewYork1.Level3.net (209.244.3.229)  77.629 ms
75.664 ms  75.351 ms
```

The link between hops 5 and 6 is transatlantic, and thus is adding more than 70ms to the latency. This is normal.

Checking Network Processes with *netstat*

- In a nutshell, ***netstat*** displays protocol statistics and current TCP/IP connections.
- If executed without options, ***netstat*** displays the current connections, performing a name lookup and displaying the resulting connections by host name.
- For example, we can use the ***-n*** option to list connections by IP address rather than host names, which can speed up results in situations where name resolutions are slow or fail.
- The ***-s*** option allows us to view per-protocol statistics for TCP, UDP, ICMP, and IP.
- We can view a specific protocol using the ***-p*** option. The following example shows how to view statistics for ICMP:

  netstat -s -p icmp

- Without specifying an interval netstat displays the results once. The following example displays IP statistics every three seconds:

  netstat -s -p ip 3

- The ***netstat*** command symbolically displays the contents of various network-related data structures. There are a number of output formats, depending on the options for the information presented.
- A wide variety of command-line options are provided to selectively display the specific types of information reported by ***netstat***.
- The following options are useful for identifying open ports, reporting whether they are in active use and by whom, and reporting which programs and specific processes are listening on the ports:

  -a: lists all ports that are either in active use or being listened to by local servers.

  -n: displays the hostnames and port identifiers in numeric format.
  Without the -n option, the hostnames and port identifiers are displayed as symbolic names. Using - n avoids a potentially long DNS look ups.

  -p: lists the name of the program listening on the socket. You must be logged in as root to use the -p option.

  -A: specifies the address family reported. The listing includes the ports in use as they are associated with your network interface cards. Local UNIX address family socket connections aren't reported, including local network-based connections in use by programs, such as any X Window program you might have running.

- Linux provides two main socket types: **AF INET** and **AF UNIX**. **AF INET** is the TCP/IP socket used across a network. **AF UNIX** is a socket type local to the kernel.

- The UNIX domain socket type is used for Interprocess communication on the same computer. It is more efficient than using TCP/IP for local sockets. Nothing goes out on the network.

- The following sample output is limited to the INET domain sockets. The listing reports all ports being listened to by network services, including the program name and the specific process ID of the listening program. It was produced with the following invocation of ***netstat***:

$ netstat - anp --ip

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address  Foreign Address State    PID/Program name
tcp        0      0 0.0.0.0:32768  0.0.0.0:*       LISTEN   523/rpc.statd
tcp        0      0 0.0.0.0:32769  0.0.0.0:*       LISTEN   735/rpc.mountd
tcp        0      0 0.0.0.0:515    0.0.0.0:*       LISTEN   707/lpd Waiting
tcp        0      0 0.0.0.0:139    0.0.0.0:*       LISTEN   880/smbd
tcp        0      0 0.0.0.0:111    0.0.0.0:*       LISTEN   508/portmap
tcp        0      0 0.0.0.0:6000   0.0.0.0:*       LISTEN   934/X
tcp        0      0 0.0.0.0:80     0.0.0.0:*       LISTEN   812/httpd
tcp        0      0 0.0.0.0:22     0.0.0.0:*       LISTEN   671/sshd
tcp        0      0 0.0.0.0:25     0.0.0.0:*       LISTEN   783/sendmail: accep
tcp        0      0 0.0.0.0:443    0.0.0.0:*       LISTEN   812/httpd
tcp        0      0 192.168.1.5:32987 142.232.66.1:22   ESTABLISHED 4291/ssh
tcp        0      0 192.168.1.5:139   192.168.1.20:1593 ESTABLISHED 3192/smbd
udp        0      0 0.0.0.0:32768  0.0.0.0:*               523/rpc.statd
udp        0      0 0.0.0.0:2049   0.0.0.0:*               -
udp        0      0 0.0.0.0:32769  0.0.0.0:*               735/rpc.mountd
udp        0      0 0.0.0.0:32770  0.0.0.0:*               -
udp        0      0 192.168.1.5:137 0.0.0.0:*              885/nmbd
udp        0      0 0.0.0.0:137     0.0.0.0:*              885/nmbd
udp        0      0 192.168.1.5:138 0.0.0.0:*              885/nmbd
udp        0      0 0.0.0.0:138     0.0.0.0:*              885/nmbd
udp        0      0 0.0.0.0:907     0.0.0.0:*              730/rpc.rquotad
udp        0      0 127.0.0.1:32802 0.0.0.0:*              3192/smbd
udp        0      0 0.0.0.0:699     0.0.0.0:*              523/rpc.statd
udp        0      0 0.0.0.0:111     0.0.0.0:*              508/portmap
```

- The ***-a*** switch is extremely useful tool for identifying active communication points. In combination with the -i option the listing will reveal any major problems with network connections through the particular network interface.

- The columns of data of most interested are "**RX-ERR**" and "**TX-ERR**" which report errors in receiving and transmitting.

- **Proto** refers to the transport protocol the service runs over TCP or UDP.

- **Recv-Q** is the number of bytes received from the remote host but not yet delivered to the local program.

- **Send-Q** is the number of bytes sent from the local program that haven't been acknowledged by the remote host yet.

- **Local Address** is the local socket, the network interface and service port pair.

- **Foreign Address** is the remote socket, the remote network interface, and service port pair.

- The local and foreign (that is, remote) addresses are listed as <**address: port**>. Under the Local Address column, the address is the name or IP address of one of your network interface cards.

- When the address is listed as *, it means that the server is listening on all network interfaces, rather than on just a single interface. The port is either the symbolic or numeric service port identifier the server is using.

- Under the Foreign Address column, the address is the name or IP address of the remote client currently participating in a connection.

- State is the local socket's connection state for sockets using the TCP protocol, **ESTABLISHED** connection, **LISTEN**ing for a connection request, as well as a number of intermediate connection establishment and shutdown states.

- These intermediate connection establishment and shutdown states include **SYN SENT, SYN RECV, FIN WAIT, FIN SENT.**

- **PID/Program name** is the process ID (PID) and program name that owns the local socket.

Checking Processes with *ps* Command

- ***ps*** reports on process status. The ***-a*** option selects all processes with controlling terminals, usually user programs running interactively in the foreground.

- The ***-x*** option selects processes without controlling terminals, usually permanent system daemons running automatically in the background.

- The two options together report all UNIX processes, including their process ID, controlling tty, run status, system time used, and the program name.

- Adding the ***-u*** option produces additional, user-oriented information, including the user login name.

- As with ***netstat***, you should be familiar with every program running on your system and why the program is running.

- With the exception of a few special UNIX system daemons-notably ***init***, ***kflushd***, ***kpiod***, ***kswapd***, ***mdrecoveryd***, and ***mingetty***, all other daemons should be services you've explicitly enabled under the run level manager, **/etc/xinetd.conf**, **/etc/inetd.conf** or **/etc/rc.d/rc.local**.

- Any other programs should be user programs you can identify. Most importantly, ***ps*** should not report any processes you don't expect to see.

- The following sample listing was produced using the ***-aux*** invocation of ***ps***:

$ ps -aux | more

```
USER       PID %CPU %MEM    VSZ  RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1   1368  544 ?        S    Jun28   0:04 init [5]
root         2  0.0  0.0      0    0 ?        SW   Jun28   0:00 [keventd]
root         3  0.0  0.0      0    0 ?        SW   Jun28   0:00 [kswapd]
root         4  0.0  0.0      0    0 ?        SW   Jun28   0:00 [kreclaimd]
root         5  0.0  0.0      0    0 ?        SW   Jun28   0:00 [bdflush]
root         6  0.0  0.0      0    0 ?        SW   Jun28   0:00 [kupdated]
root         7  0.0  0.0      0    0 ?        SW<  Jun28   0:00 [mdrecoveryd]
root        72  0.0  0.0      0    0 ?        SW   Jun28   0:00 [khubd]
root       489  0.0  0.1   1428  596 ?        S    Jun28   0:00 syslogd -m 0
root       494  0.0  0.2   2048 1116 ?        S    Jun28   0:00 klogd -2
rpc        508  0.0  0.1   1512  588 ?        S    Jun28   0:00 portmap
rpcuser    523  0.0  0.1   1564  772 ?        S    Jun28   0:00 rpc.statd
root       644  0.0  0.1   1480  640 ?        S    Jun28   0:00 /usr/sbin/automou
daemon     659  0.0  0.1   1400  560 ?        S    Jun28   0:00 /usr/sbin/atd
root       671  0.0  0.2   2588 1116 ?        S    Jun28   0:01 /usr/sbin/sshd
root       691  0.0  0.1   2240  960 ?        S    Jun28   0:00 xinetd -stayalive
lp         707  0.0  0.1   2540  956 ?        S    Jun28   0:00 lpd Waiting
root       730  0.0  0.0   1368  400 ?        S    Jun28   0:00 rpc.rquotad
root       735  0.0  0.1   1492  552 ?        S    Jun28   0:00 rpc.mountd
root       741  0.0  0.0      0    0 ?        SW   Jun28   0:00 [nfsd]
root       742  0.0  0.0      0    0 ?        SW   Jun28   0:00 [nfsd]
root       743  0.0  0.0      0    0 ?        SW   Jun28   0:00 [nfsd]
root       744  0.0  0.0      0    0 ?        SW   Jun28   0:00 [nfsd]
root       745  0.0  0.0      0    0 ?        SW   Jun28   0:00 [nfsd]
root       746  0.0  0.0      0    0 ?        SW   Jun28   0:00 [nfsd]
```

```
root       747  0.0  0.0     0    0 ?        SW   Jun28   0:00 [nfsd]
root       748  0.0  0.0     0    0 ?        SW   Jun28   0:00 [nfsd]
root       749  0.0  0.0     0    0 ?        SW   Jun28   0:00 [lockd]
root       750  0.0  0.0     0    0 ?        SW   Jun28   0:00 [rpciod]
root       783  0.0  0.3  5004 1888 ?        S    Jun28   0:00 sendmail: accepti
root       796  0.0  0.0  1396  492 ?        S    Jun28   0:00 gpm -t ps/2 -m /d
root       812  0.0  1.2 12932 6296 ?        S    Jun28   0:01 /usr/sbin/httpd -
root       824  0.0  0.1  1552  692 ?        S    Jun28   0:00 crond
xfs        868  0.0  0.9  6056 4920 ?        S    Jun28   0:02 xfs -droppriv -da
root       880  0.0  0.2  3908 1336 ?        S    Jun28   0:00 smbd -D
root       885  0.0  0.2  3252 1408 ?        S    Jun28   0:00 nmbd -D
root       920  0.0  0.0  1340  436 tty1     S    Jun28   0:00 /sbin/mingetty tt
root       922  0.0  0.0  1340  436 tty3     S    Jun28   0:00 /sbin/mingetty tt
root       923  0.0  0.0  1340  436 tty4     S    Jun28   0:00 /sbin/mingetty tt
root       924  0.0  0.0  1340  436 tty5     S    Jun28   0:00 /sbin/mingetty tt
root       925  0.0  0.0  1340  436 tty6     S    Jun28   0:00 /sbin/mingetty tt
root       926  0.0  0.7 15740 3940 ?        S    Jun28   0:00 /usr/bin/kdm -nod
root       934  0.0  1.0 81820 5588 ?        SL   Jun28   1:45 /etc/X11/X -auth
root       942  0.0  1.3 17020 7112 ?        S    Jun28   0:00 -:0
root       957  0.0  1.0 13932 5240 ?        S    Jun28   0:00 ksmserver --resto
root      1039  0.0  0.9 17804 4952 ?        S    Jun28   0:00 kdeinit: dcopserv
root      1041  0.0  1.0 18196 5592 ?        S    Jun28   0:00 kdeinit: klaunche
root      1043  0.0  1.0 17792 5424 ?        S    Jun28   0:00 kdeinit: kded
root      1046  0.0  0.6  4700 3168 ?        S    Jun28   0:23 artsd -F 10 -S 40
root      1053  0.0  1.0 17860 5376 ?        S    Jun28   0:00 kdeinit: kxmlrpcd
root      1058  0.0  0.8 17664 4484 ?        S    Jun28   0:00 kdeinit: Running.
root      1060  0.0  1.3 17532 6956 ?        S    Jun28   0:01 knotify
root      1061  0.0  1.4 18704 7568 ?        S    Jun28   0:03 kdeinit: kwin -se
root      1063  0.0  2.0 20296 10316 ?       S    Jun28   0:05 kdeinit: kdesktop
root      1065  0.0  1.9 20780 9936 ?        S    Jun28   0:07 kdeinit: kicker
root      1068  0.0  1.4 18748 7684 ?        S    Jun28   0:00 kdeinit: klipper
root      1071  0.0  1.2 18132 6504 ?        S    Jun28   0:00 kdeinit: khotkeys
root      1072  0.0  1.3 18392 6848 ?        S    Jun28   0:00 kdeinit: kwrited
root      1073  0.0  1.6 19168 8400 ?        S    Jun28   0:01 kdeinit: konsole
root      1074  0.0  1.6 19200 8428 ?        S    Jun28   0:02 kdeinit: konsole
root      1075  0.0  1.6 19188 8424 ?        S    Jun28   0:10 kdeinit: konsole
root      1076  0.0  0.0  1532  512 pts/0    S    Jun28   0:00 /bin/cat
root      1078  0.0  0.2  2464 1420 pts/3    S    Jun28   0:00 /bin/bash
root      1079  0.0  0.2  2476 1444 pts/1    S    Jun28   0:00 /bin/bash
root      1082  0.0  0.2  2468 1432 pts/2    S    Jun28   0:00 /bin/bash
root      1758  0.0  0.6  5472 3588 ?        S    Jun28   0:02 snort -Afull -D -
root      3192  0.0  0.4  5272 2256 ?        S    Jun30   0:00 smbd -D
root      3306  0.0  1.5 19416 8116 ?        S    Jun30   0:00 kdeinit: kio_uise
apache    4855  0.0  1.2 12980 6156 ?        S    04:02   0:00 /usr/sbin/httpd -
apache    4856  0.0  1.2 12980 6156 ?        S    04:02   0:00 /usr/sbin/httpd -
apache    4857  0.0  1.2 12980 6156 ?        S    04:02   0:00 /usr/sbin/httpd -
apache    4858  0.0  1.2 12980 6156 ?        S    04:02   0:00 /usr/sbin/httpd -
apache    4859  0.0  1.2 12980 6156 ?        S    04:02   0:00 /usr/sbin/httpd -
apache    4860  0.0  1.2 12980 6156 ?        S    04:02   0:00 /usr/sbin/httpd -
apache    4861  0.0  1.2 12980 6156 ?        S    04:02   0:00 /usr/sbin/httpd -
apache    4862  0.0  1.2 12980 6156 ?        S    04:02   0:00 /usr/sbin/httpd -
root     11064  0.0  0.0  1340  436 tty2     S    10:03   0:00 /sbin/mingetty tt
root     11115  0.0  0.1  1564  724 ?        S    11:01   0:00 CROND
root     11116  0.0  0.1  1920  908 ?        S    11:01   0:00 /bin/bash /usr/bi
root     11126  0.0  0.1  1656  552 ?        S    11:01   0:00 awk -v progname=/
root     11127  0.0  0.1  1904  880 ?        S    11:01   0:00 /bin/sh /usr/lib/
root     11129  0.0  0.0  1352  512 ?        S    11:01   0:00 /usr/lib/sa/sadc
root     11167  0.0  0.1  2808  904 pts/1    R    11:20   0:00 ps -aux
root     11168  0.0  0.1  1572  552 pts/1    S    11:20   0:00 more
```

- The listing shows all the processes running. For example, Line 2 shows that ***init*** is running. It is the parent of all other processes and it always runs.
-
- Other system process and daemons include ***bdflushd*** (periodically flushes modified file system buffers back to disk), ***kswapd*** **(**a kernel thread that selects physical memory pages for swapping from memory to swap space on disk to free memory for other processes), etc.

- Also note that the system is running **NFS**, **SAMBA** and the **Apache** web server.

- If you see unfamiliar or unexpected processes, someone might have gained access to your machine. That will have to be confirmed with system logs and Intrusion Detection reports.

Monitoring usage of system resources using *lsof*

- LiSt Open Files (***lsof***) is a Linux utility that allows you to view current network connections and the files associated with them.
- ***lsof*** is a tool to list open files, including open internet connections that all services are listening on and open ports that rpc programs are using.
- It provides verbose output and is useful in extracting a variety of information. For example, it allows you to see what program is operating on an open port, which daemons have established connections, and what ports are open on your server.
- While similar in many ways to utilities like netstat and fuser, ***lsof*** has many unique options that let you find specific information on ports, users, processes, and files.

- Invoking ***lsof*** by itself will produce an output showing all open files corresponding to every active process on the box. This can be quite lengthy, so it's best to narrow the listing by identifying what you are looking for in advance.
- For example the ***lsof –i*** command lists all open files associated with Internet connections. It is similar in format to ***netstat -a -p*** and will look something like:

```
COMMAND     PID    USER   FD   TYPE DEVICE SIZE NODE NAME
portmap     567     rpc    3u  IPv4    937       UDP *:sunrpc
portmap     567     rpc    4u  IPv4    950       TCP *:sunrpc (LISTEN)
rpc.statd   586 rpcuser    4u  IPv4   1011       UDP *:32768
rpc.statd   586 rpcuser    6u  IPv4   1014       TCP *:32768 (LISTEN)
sshd        696    root    3u  IPv4   1345       TCP *:ssh (LISTEN)
xinetd      711    root    5u  IPv4   1380       TCP
localhost.localdomain:32769 (LISTEN)
ntpd        725     ntp    4u  IPv4   1422       UDP *:ntp
ntpd        725     ntp    5u  IPv4   1423       UDP localhost.localdomain:ntp
ntpd        725     ntp    6u  IPv4   1424       UDP 192.168.1.10:ntp
lpd         739      lp    6u  IPv4   1447       TCP *:printer (LISTEN)
rpc.rquot   755    root    3u  IPv4   1494       UDP *:932
rpc.rquot   755    root    4u  IPv4   1499       TCP *:935 (LISTEN)
rpc.mount   774    root    3u  IPv4   1534       UDP *:32771
rpc.mount   774    root    4u  IPv4   1537       TCP *:32770 (LISTEN)
sendmail    793    root    4u  IPv4   1609       TCP *:smtp (LISTEN)
httpd       823    root    3u  IPv4   1654       TCP *:http (LISTEN)
httpd       823    root    4u  IPv4   1655       TCP *:https (LISTEN)
smbd        896    root    9u  IPv4   1769       TCP *:netbios-ssn (LISTEN)
nmbd        900    root    6u  IPv4   1772       UDP *:netbios-ns
nmbd        900    root    7u  IPv4   1773       UDP *:netbios-dgm
nmbd        900    root    8u  IPv4   1777       UDP 192.168.1.10:netbios-ns
nmbd        900    root    9u  IPv4   1778       UDP 192.168.1.10:netbios-dgm
smbd      12377    root    5u  IPv4  20673        UDP
localhost.localdomain:32773
smbd      12377    root   12u  IPv4  20672       TCP 192.168.1.10:netbios-ssn-
>zeus.Olympus.net:1052 (ESTABLISHED)
```

- By default, lsof lists detailed information about each connection. In the listing above we see the command or program involved, the process ID (PID), the user running the command, the file descriptor (FD), the type of connection, the device number, the Internet protocol, and the name of the file or Internet address.

- The ***-i*** option can be especially useful when attempting to secure a Linux system. It will allow us to quickly determine what ports are open and listening for incoming connections.

- Since lsof will also associate them with a program name, we can quickly identify unnecessary security risks and shut them down.

- Ports that are awaiting connections have the keyword LISTEN appended to them. These are ports that are open and accepting connections.

- The keyword ESTABLISHED indicates that a connection on the given port has been made.

- We can see in the listing that there is a NetBIOS session from 192.168.1.10 to zeus.Olympus.net.

- We can narrow the search by specifying a particular port, service, or host name using techniques such as:

  lsof –i :587
  lsof –i :smtp
  lsof –i @somewhere.remote.net

- We can check for an application listening at local TCP port 5555 as follows:

  lsof -i tcp:5555

- The above examples will list connections listening or established on port 587, list connections associated with the well-known service SMTP, and list connections coming from or going to the host *somewhere.remote.net*, respectively.

- These techniques are very useful if you know what you are looking for in advance. You can watch and see whether inbound SMTP connections are taking too long, possibly causing timeouts.

- You can verify that the service is in fact running and what port it is listening on. And you can see if anyone from a certain device is connected to your system, whether it is via SSH, Telnet, FTP, or just about any other way possible.

- lsof will also accept a PID and output all open files it is using. For example, we execute ***lsof -i*** to determine what PID number NameD (BIND DNS service) was operating under.

- Say we discovered it was 555. We then execute the command ***lsof -p 505***. The output will look as follows:

```
CMD    PID   USER  FD   TYPE   DEV  SIZE    NODE   NAME

named  555   root  cwd  DIR    3,1  4096    66512  /var/cache/bind
named  555   root  txt  REG    3,1  513512  387815 /usr/sbin/named
named  555   root  mem  REG    3,1  82503   80767  /lib/ld-2.2.5.so
named  555   root  4u   IPv4   458          UDP   *:32768
named  555   root  21u  IPv4   453          TCP
localhost:domain(LISTEN)
```

- We can observe the different FDs, or file descriptors immediately.
- The *cwd* variable represents the current working directory of the process.
- The *txt* defines the program text, which is the executable itself. The
- The *mem* is a file held in memory, in this case a library; the 4 and 21 represent files in use by this particular process; and the *u* designator defines them as having both read and write access.
- All these pieces of information will help us determine whether something physically exists on the system, is being used by the process, or is being held in memory.

Interpreting the System Logs

- ***syslogd*** is the service daemon that logs system events. ***syslogd***'s main system log file is **/var/log/messages**.
- Many programs use ***syslogd***'s standard logging services. Other programs, such as the Apache Web server (http) and SAMBA maintain their own separate log files.
- By default, system log files are written in the **/var/log** directory. The rules for logging are specified in the configuration file: **/etc/syslog.conf**.
- Most Linux distributions are preconfigured to write messages to at least into four separate files: **messages**, **secure**, **maillog**, and **spooler**.
- **/var/log/messages** is the system catchall file that contains a copy of whatever messages are written to the console, any operating system messages written to the kernel's internal log buffer, and any messages produced by programs that use the ***syslog*** system call, such as ***named***, ***sendmail***, and ***login***.
- **/var/log/secure** contains reports of **root logins**, **user logins** and ***su*** attempts to other users. Reports of connections from other systems and root login failures at the system daemon level are also written here. Each login is recorded.

- **/var/log/maillog** contains a record of incoming and outgoing mail traffic and server status.

- **/va r/log/spooler** is not used by most systems. The file contains error messages from the ***uucp*** and news server (***innd***) daemons.

- Not all log messages are equally important, or useful as far as security is concerned. You can customize the log output to suit your own needs and objectives using the configuration file **/etc/syslog.conf** allows you to customize the log output to meet your own needs.

- Messages are categorized by the subsystem that produces them. In the ***man*** pages, these categories are called facilities as shown in the table below:

Facility	Message Category
auth or security	Security/authorization
authpriv	Private security/authorization
cron	cron daemon messages
daemon	System daemon-generated messages
ftp	FTP server messages
kern	Kernel messages
lpr	Printer subsystem
news	Network news subsystem
syslog	syslogd-generated messages
user	User program-generated messages
UUCP	UUCP subsystem
mail	Mail subsystem

- Within any given facility category, log messages are divided into **priority** types. The priorities, in increasing order of importance, are listed below:

Priority	Message Type
debug	Debug messages
info	Informational status messages
notice	Normal but important conditions
warning or warn	Warning messages
err or error	Error messages
crit	Critical conditions
alert	Immediate attention required
emerg or panic	System is unusable

- Each entry in **syslog.conf** specifies a **logging facility**, its **priority**, and where to write the messages. Note that the priority is inclusive, meaning all messages at that priority and higher. If you specify messages at the ***error*** priority, for example, all messages at priority ***error*** and higher are included.

- Logs can be written to devices, such as the **console**, to **files**, and **to remote machines**.

- The following two entries write all kernel messages to both the console and to **/var/log/messages**. Messages can be duplicated to multiple destinations:

    kern. ***/dev/console***
    kern. ***/var/log/messages***

- The following entry writes panic messages to all default locations, including **/var/log/messages**, the console, and all user terminal sessions:

    ****.emerg***

- The next two entries write authentication information related to root privilege and connections to **/var/log/secure**, and user authorization information to **/var/log/auth**. With the priority defined at the **info** level, debug messages won't be written:

    authpriv.info ***/var/log/secure***
    auth.info ***/var/log/auth***

- The next two entries write general daemon information to **/var/log/daemon**, and mail traffic information to **/var/log/maillog**:

    daemon.notice ***/var/log/daemon***
    mail.info ***/var/log/maillog***

- For security purposes, log files are often not readable by general users. The security log file, /**var/log/secure**, in particular, is readable by root alone.

Remote Logging

- When a black hat compromises a system, the first thing he will want to do is to conceal all traces of the entry. To do this, most hackers will modify system logs to remove any evidence of their existence.

- To state the obvious, log files hold critical system information that could be used to trace, charge and convict a hacker. To that end it is absolutely imperative to ensure that the log files are valid and have not been tampered with.

- The best way to ensurte this is to set up a remote machine whose sole purpose is to receive and archive log files.

- That way even if a black hat breaks into your primary system and tampers with the logs, you will still have another set of log files that have not been altered.

- The ***syslogd*** provides support for full remote logging to a remote host also running ***syslogd***.

- The first step when setting up a remote log server is to configure your local machine's syslog daemon to send its log files to a remote log server.
-
- The following entry in the **/etc/syslogd.conf** will log data and where it will log it. For example, to forward **all** messages to a remote host use the following **syslog.conf** entries:

    # Sample syslogd entry to forward all messages to a remote host.
    ****.* @athena***

- More examples:

    # To forward all kernel messages to a remote host.
    kern.* @athena

    # Remote Logging which includes local logging as well (for fault tolerance)
    kern.crit @athena
    kern.crit /dev/console

- These entries are repeated for all log files to kept offsite.

- Note that any changes made to **syslog.conf** will require you to restart ***syslogd*** to affect the new configuration. This can be done in either one of the two following ways:

    $ killall -HUP syslogd
    $ /etc/rc.d/init.d/syslog restart

- On the log server, make sure you launch ***syslogd*** with the ***-r*** switch, specifying remote logging capabilities.

- Also ensure that the entry "**514/udp**" exists in the **/etc/services** file as well, on the server, or it will not listen to the specified port.

- One of the most crucial steps in this project is to fully lock down the machine that is receiving all of the logs. This is only a log server, so all other services should not be turned on.

- Go through the process of disabling service through the **xinetd.d** directory and editing the individual service files.

- Disable all RPC services as well through the **/etc/rc.d** directory. Anything listed in the rc. directories that start with a capital "S" means that the service will start at boot time.

- In order to disable it from starting, simply 'mv to (a lowercase 's'). e.g. mv S11portmap s11portmap. This disables ***portmapper*** from starting at boot time.

- You will want to do this with all unneeded services in this directory. These include: ***nfslock, apmd, netfs, identd, autofs, portmap, atd, pcmcia, sendmail, gpm, httpd, vmware, xfs,*** etc.

- Go through all of the **rc0.d-rc6.d** directories looking for these same files to disable. Also go through the password file and remove every account that isn't being used.

- There are several good tools for monitoring the log files. Check out the following:

  Logcheck - www.psionic.com
  Swatch - www.swatch.org