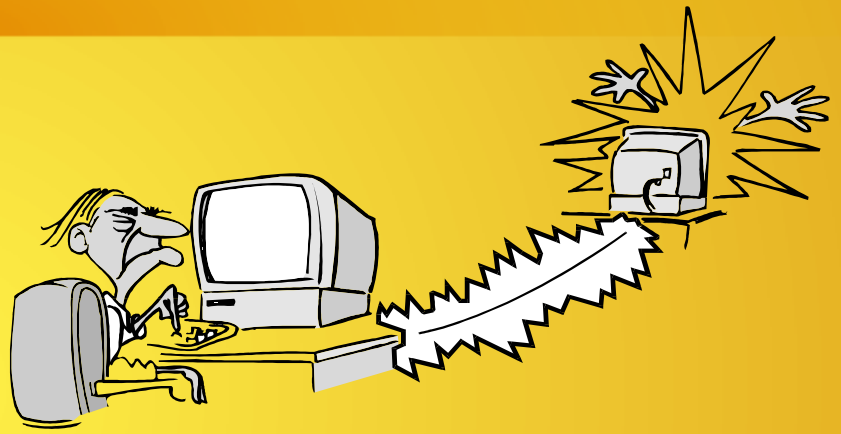
IPtables

- Objectives
  - to learn the basics of iptables
- Contents
  - Start and stop IPtables
  - Checking IPtables status
  - Input and Output chain
  - Pre and Post routing
  - Forward of address and port
  - Firewall standard rules
  - Loading/Unloading kernel driver modules
  - Connection tracking modules
- Practicals
  - working with iptables
- Summary



Firewall Programs

- Ipfwadm : Linux kernel 2.0.34
- Ipchains : Linux kernel 2.2.*
- Iptables : Linux kernel 2.4.*

What Is iptables?

- Stateful packet inspection.
  The firewall keeps track of each connection passing through it, This is an important feature in the support of active FTP and VoIP.
- Filtering packets based on a MAC address IPv4 / IPv6
  Very important in WLAN's and similar enviroments.
- Filtering packets based the values of the flags in the TCP header
  Helpful in preventing attacks using malformed packets and in restricting access.
- Network address translation and Port translating NAT/NAPT
  Building DMZ and more flexible NAT enviroments to increase security.
- Source and stateful routing and failover functions
  Route traffic more efficiant and faster than regular IP routers.
- System logging of network activities
  Provides the option of adjusting the level of detail of the reporting
- A rate limiting feature
  Helps to block some types of denial of service (DoS) attacks.
- Packet manipulation (mangling) like altering the TOS/DSCP/ECN bits of the IP header
  Mark and classify packets dependent on rules. First step in QoS.

Processing For Packets Routed By The Firewall 1/2

Queue Type	Queue Function	Packet transformation chain in Queue	Chain Function
Filter	Packet filtering	FORWARD	Filters packets to servers accessible by another NIC on the firewall.
		INPUT	Filters packets destined to the firewall.
		OUTPUT	Filters packets originating from the firewall
Nat	Network Address Translation	PREROUTING	Address translation occurs before routing. Facilitates the transformation of the destination IP address to be compatible with the firewall's routing table. Used with NAT of the destination IP address, also known as **destination NAT** or **DNAT**.

Processing For Packets Routed By The Firewall 2/2

		POSTROUTING	Address translation occurs after routing. This implies that there was no need to modify the destination IP address of the packet as in pre-routing. Used with NAT of the source IP address using either one-to-one or many-to-one NAT. This is known as **source NAT**, or **SNAT**.
		OUTPUT	Network address translation for packets generated by the firewall. (Rarely used in SOHO environments)
Mangle	TCP header modification	PREROUTING POSTROUTING OUTPUT INPUT FORWARD	Modification of the TCP packet quality of service bits before routing occurs (Rarely used in SOHO environments)

Targets And Jumps 1/2

- ACCEPT
  - `iptables` stops further processing.
  - The packet is handed over to the end application or the operating system for processing
- DROP
  - `iptables` stops further processing.
  - The packet is blocked.
- LOG
  - The packet information is sent to the syslog daemon for logging.
  - `iptables` continues processing with the next rule in the table.
  - You can't log and drop at the same time ->use two rules.

    --log-prefix ”reason"
- REJECT
  - Works like the DROP target, but will also return an error message to the host sending the packet that the packet was blocked

    --reject-with qualifier Qualifier is an ICMP message

Targets And Jumps 2/2

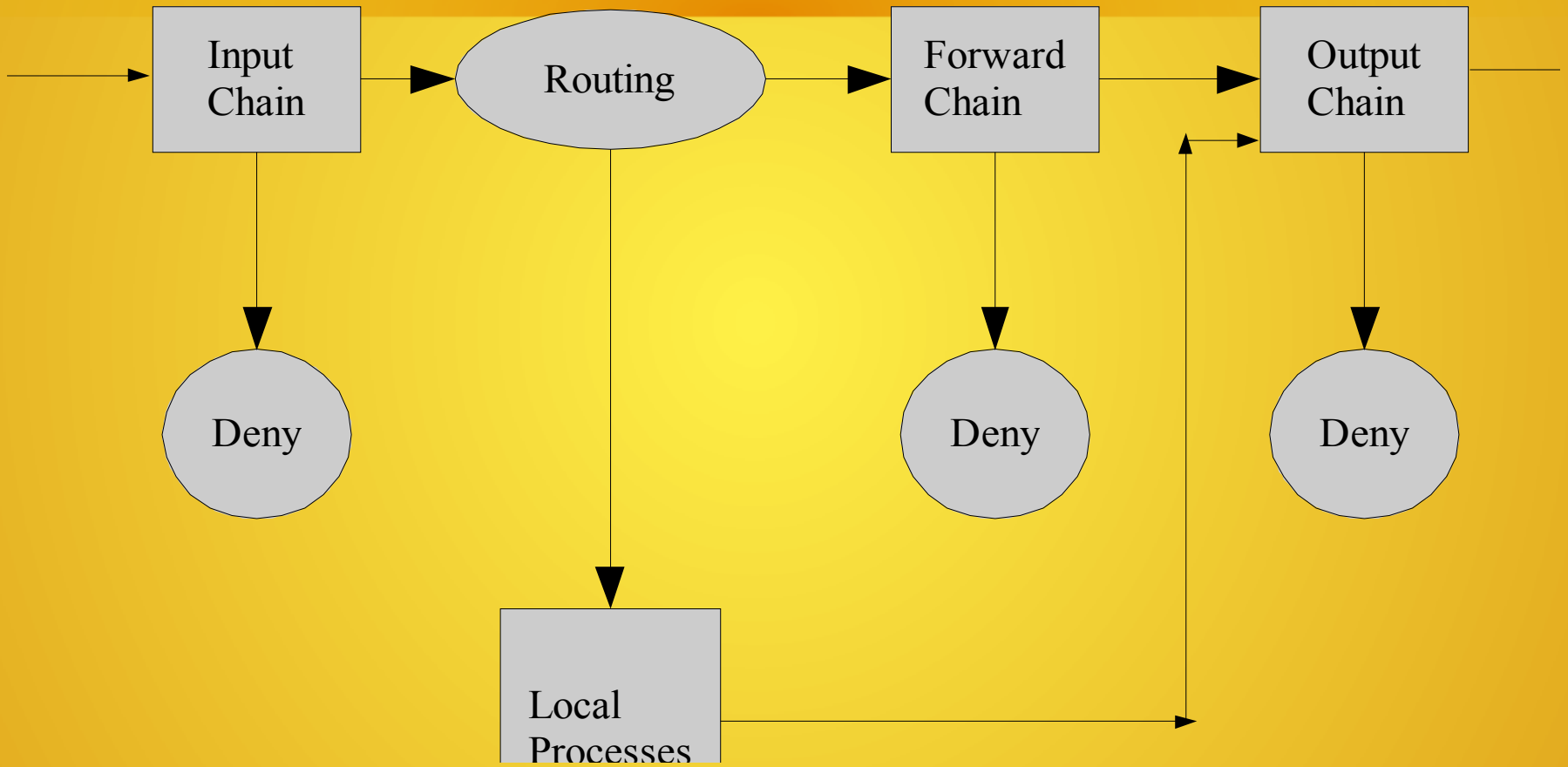
- SNAT
  - Used to do source network address translation rewriting the source IP address of the packet
  - The source IP address is user defined

    --to-source <address>[-<address>][:<port>-<port>]
- DNAT
  - Used to do destination network address translation. ie. rewriting the destination IP address of the packet

    --to-destination ipaddress
- MASQUERADE
  - Used to do Source Network Address Translation.
  - By default the source IP address is the same as that used by the firewall's interface

    [--to-ports <port>[-<port>]]

Ipchains packet traversal



Important Iptables Command Switch Operations 1/2

iptables command Switch	Description
`-t <table>`	If you don't specify a table, then the `filter` table is assumed. As discussed before, the possible built-in tables include: filter, nat, mangle
`-j <target>`	Jump to the specified target chain when the packet matches the current rule.
`-A`	Append rule to end of a chain
`-F`	Flush. Deletes all the rules in the selected table
`-p <protocol-type>`	Match protocol. Types include, `icmp`, `tcp`, `udp`, and `all`

Common TCP and UDP Match Criteria

Switch	Description
`-p tcp --sport <port>`	TCP source port Can be a single value or a range in the format: *start-port-number:end-port-number*
`-p tcp --dport <port>`	TCP destination port Can be a single value or a range in the format: *starting-port:ending-port*
`-p tcp --syn`	Used to identify a new TCP connection request ! --syn means, not a new connection request
`-p udp --sport <port>`	UDP source port Can be a single value or a range in the format: *starting-port:ending-port*

Common ICMP (Ping) Match Criteria

Matches used with ---icmp-type	Description
`--icmp-type <type>`	The most commonly used types are echo-reply and echo-request

- Allow ping request and reply
  - `iptables` is being configured to allow the firewall to send ICMP echo-requests (pings) and in turn, accept the expected ICMP echo-replies.

```
iptables -A OUTPUT -p icmp --icmp-type echo-request -j ACCEPT
iptables -A INPUT  -p icmp --icmp-type echo-reply   -j ACCEPT
```

- Put limit on ping to prevent flood pings

```
iptables -A INPUT -p icmp --icmp-type echo-request \
  -m limit --limit 1/s -i eth0 -j ACCEPT
```

Important Iptables Command Switch Operations 2/2

-s <ip-address>	Match source IP address
-d <ip-address>	Match destination IP address
-i <interface-name>	Match "input" interface on which the packet enters.
-o <interface-name>	Match "output" interface on which the packet exits

- **We try to define a rule that will accept all packages on interface eth0 that uses TCP and has destination address 192.168.1.1.**
- **We first define the MATCH criterias:**

  Use default filter table (absense of –t)

  Append a rule to end of INPUT chain (-A INPUT)

  Match on source address can be any 0/0 address (-s 0/0)

  Input interface used is eth0 (-i eth0)

  Match on destination address 192.168.1.1 (-d 192.168.1.1)

  Match Protocol TCP (-p TCP)

  If all matches is fulfilled, then jump to ACCEPT chain. (-j ACCEPT)

iptables syntax

```
iptables -I INPUT -i eth1 -p tcp -s 192.168.56.1 \
--sport 1024:65535 -d 192.168.56.2 --dport 22 \
-j ACCEPT

iptables -I OUTPUT -o eth1 -p tcp ! --syn \
-s 192.168.56.2 --sport 22 -d 192.168.56.1 \
--dport 1024:65535 -j ACCEPT
```

Forwarding Packets

```
iptables -A FORWARD -i <internal interface> \
-o <external interface> -s 192.168.56.1/32 --sport \
1024:65535 -m state --state \ NEW,ESTABLISHED,RELATED -j ACCEPT

iptables -A FORWARD -i <external interface> \
-o <internal interface> -m state --state \
ESTABLISHED,RELATED -j ACCEPT
```

Raw iptables log output

```
Jun 25 09:05:11 hebe kernel: IN=eth1 OUT= MAC=00:00:92:a7:df:05:02:07:01:23:5e:29:08:00
SRC=10.90.10.112 DST=10.90.10.116 LEN=44 TOS=0x00
PREC=0x00 TTL=60 ID=7276 PROTO=TCP SPT=47785 DPT=10003 WINDOW=16384 RES=0x00 SYN URGP=0
Jun 25 09:05:11 hebe kernel: IN=eth1 OUT= MAC=00:00:92:a7:df:05:02:07:01:23:5e:29:08:00
SRC=10.90.10.112 DST=10.90.10.116 LEN=44 TOS=0x00
PREC=0x00 TTL=60 ID=7276 PROTO=TCP SPT=47785 DPT=10003 WINDOW=16384 RES=0x00 SYN URGP=0
Jun 25 09:05:12 hebe kernel: IN=eth1 OUT= MAC=ff:ff:ff:ff:ff:ff:00:06:5b:d1:24:bb:08:00
SRC=10.90.50.251 DST=10.90.255.255 LEN=241 TOS=0x00 PREC=0x00 TTL=128 ID=547 PROTO=UDP
SPT=138 DPT=138 LEN=221
Jun 25 09:05:12 hebe kernel: IN=eth1 OUT= MAC=ff:ff:ff:ff:ff:ff:00:06:5b:d1:24:bb:08:00
SRC=10.90.50.251 DST=10.90.255.255 LEN=241 TOS=0x00 PREC=0x00 TTL=128 ID=547 PROTO=UDP
SPT=138 DPT=138 LEN=221
Jun 25 09:05:12 hebe kernel: IN=eth1 OUT= MAC=ff:ff:ff:ff:ff:ff:00:50:04:74:0b:81:08:00
SRC=10.90.10.6 DST=10.90.255.255 LEN=78 TOS=0x00 PREC=0x00 TTL=64 ID=44852 PROTO=UDP SPT=137
DPT=137 LEN=58
Jun 25 09:05:12 hebe kernel: IN=eth1 OUT= MAC=ff:ff:ff:ff:ff:ff:00:50:04:74:0b:81:08:00
SRC=10.90.10.6 DST=10.90.255.255 LEN=78 TOS=0x00 PREC=0x00 TTL=64 ID=44852 PROTO=UDP SPT=137
DPT=137 LEN=58
Jun 25 09:05:15 hebe kernel: IN=eth1 OUT= MAC=ff:ff:ff:ff:ff:ff:00:60:cf:20:2d:37:08:00
SRC=10.90.10.104 DST=10.90.255.255 LEN=78 TOS=0x00 PREC=0x00 TTL=1 ID=60733 DF PROTO=UDP
SPT=137 DPT=137 LEN=58
```

log_analysis output

```
3          Chain: input   Interface: eth0 >> 211.39.225.244  1559  => 192.168.56.2    TCP   27374
4          Chain: input   Interface: eth0 >> 211.44.96.76    1659  => 192.168.56.2    TCP   27374
4          Chain: input   Interface: eth0 >> 24.209.129.7    2846  => 192.168.56.2    TCP   27374
4          Chain: input   Interface: eth0 >> 4.41.13.124     1537  => 192.168.56.2    TCP   27374
3          Chain: input   Interface: eth0 >> 61.255.229.7    3714  => 192.168.56.2    TCP   27374
3          Chain: input   Interface: eth0 >> 64.231.21.254   2361  => 192.168.56.2    TCP   27374
4          Chain: input   Interface: eth0 >> 65.24.46.200    1992  => 192.168.56.2    TCP   27374
4          Chain: input   Interface: eth0 >> 65.33.176.170   1328  => 192.168.56.2    TCP   27374
4          Chain: input   Interface: eth0 >> 65.43.103.123   3672  => 192.168.56.2    TCP   27374
4          Chain: input   Interface: eth0 >> 66.188.158.191  3064  => 192.168.56.2    TCP   27374
3          Chain: input   Interface: eth0 >> 80.224.203.178  4697  => 192.168.56.2    TCP   27374
3          Chain: input   Interface: eth0 >> 12.220.98.42    1380  => 192.168.56.2    TCP   27374
3          Chain: input   Interface: eth0 >> 193.205.135.94  2498  => 192.168.56.2    TCP   1433
3          Chain: input   Interface: eth0 >> 198.83.120.42   1711  => 192.168.56.2    TCP   1433
3          Chain: input   Interface: eth0 >> 202.108.234.155 3877  => 192.168.56.2    TCP   1433
3          Chain: input   Interface: eth0 >> 202.140.162.42  19914 => 192.168.56.2    TCP   1433
3          Chain: input   Interface: eth0 >> 205.158.95.87   1367  => 192.168.56.2    TCP   1433
3          Chain: input   Interface: eth0 >> 208.2.225.43    3818  => 192.168.56.2    TCP   1433
3          Chain: input   Interface: eth0 >> 212.118.71.3    1429  => 192.168.56.2    TCP   1433
4          Chain: input   Interface: eth0 >> 61.85.33.8      2113  => 192.168.56.2    TCP   27374
4          Chain: input   Interface: eth0 >> 61.99.45.198    4515  => 192.168.56.2    TCP   27374
3          Chain: input   Interface: eth0 >> 62.90.204.2     3798  => 192.168.56.2    TCP   1433
3          Chain: input   Interface: eth0 >> 63.231.101.56   61428 => 192.168.56.2    TCP   1433
3          Chain: input   Interface: eth0 >> 66.28.45.209    4268  => 192.168.56.2    TCP   1433
```

Firewall Optimization

- Place loopback rules as early as possible.
- Place forwarding rules as early as possible.
- Use the state and connection-tracking modules to bypass the firewall for established connections.
- Combine rules to standard TCP client-server connections into a single rule using port lists.
- Place rules for heavy traffic services as early as possible.

User Defined Chains

```
iptables -A INPUT -i $INTERNET -d <public address> \
-j EXT-input

iptables -A EXT-input -p udp --sport 53 \
--dport 53 -j EXT-dns-server-in
iptables -A EXT-input -p tcp ! --syn --sport 53 \
--dport 1024:65535 -j EXT-dns-server-in

iptables -A EXT-dns-server-in -s $NAMESERVER_1 \
-j ACCEPT
```

Applications

- Complex Network Applications
- Volatile environments
- Internal Security
- System Segregation
- Local Host Protection

Using Firewalls

- Know your system
- Principle of Least Privilege
- Defense in Depth
- Protection is key
- Detection is must
- Identify the enemy

Avoiding Locking Yourself

- Scenario: You are going to make changes to the IPTables policy rules. You want to avoid locking yourself, and potentially everybody else out too (this costs time and money).
- Tips #1: Take a backup of your IPTables configuration before you ever start working on it.
  - `/sbin/iptables-save > /root/iptables-works`
- Even better, include a timestamp as part of the file name:
  - `/sbin/iptables-save > /root/iptables-works-`date +%F`
  - You get a file with a name like `/root/IPTablesworks-2014-04-14`.
- If you do something that prevents your system from working, you can quickly restore it.
  - `/sbin/iptables-restore < /root/iptables-works-2014-04-14`

- Tip #2: Put specific rules at the top of the policy and generic rules at the bottom.
- The more criteria you specify in the rule, the less chance you will have of locking yourself out.
  - `iptables -A INPUT -p tcp --dport 22 -s 10.0.0.0/8 -d 192.168.100.101 -j DROP`
- Avoid generic rules like this at the top of the policy rules:
  - `iptables -A INPUT -p tcp --dport 22 -j DROP`
- There are plenty of ways that you can be more specific.
- For example, using "-i eth0" will limit the processing to a single NIC in your server.
- This way, it will not apply the rule to eth1.

- Tip #3: Whitelist your IP address at the top of your policy rules.
- This is a very effective method of not getting locked out.
- Everybody else, not so much.
  - `iptables -I INPUT -s <your IP> -j ACCEPT`
- You need to put this as the FIRST rule in order for it to work properly.
- Remember, "-I" inserts it as the first rule; "-A" appends it to the end of the list.

- Tip #4: Know and understand all of the rules in your current policy.
- Not making the mistake in the first place is half the battle.
- If you understand the inner workings behind your IPTables policy, it will make your life easier.
- Draw a flow chart if you must.
- Also remember: What the policy does and what it is supposed to do can be two different things.

Restricting an IP Address Range

- Scenario: You're employees are spending too much time on Facebook and not getting their work done.
- You want to block access to Facebook.
- Tip: Use this process to block access to Facebook.
  - Find out all ip addresses of facebook.com:

```
host -t a www.facebook.com
www.facebook.com is an alias for star.c10r.facebook.com.
star.c10r.facebook.com has address 31.13.65.17
whois 31.13.65.17 | grep inetnum
inetnum:        31.13.64.0 - 31.13.127.255
```

Regulating by Time

- Scenario: The backlash from your employees over denying access to Facebook is causes you to relent (a little). You decide to allow access to facebook.com only at lunch time (1200 to 1300).
- Tip: Use the time features of IPTables to open up the access.
  - `iptables -A OUTPUT -p tcp -m multiport -dport http,https -i eth0 -o eth1 -m time --timestart 12:00 --timestop 13:00 -d 31.13.64.0/18  -j ACCEPT`
- This presumes a default policy of DROP.

Regulating by Time

- Scenario: Drop all TCP/UDP traffic during service hours (between 02:00 and 03:00), that is, for maintenance's tasks which should not be disrupted by incoming traffic.

```
iptables -A INPUT -p tcp -m time --timestart 02:00
--timestop 03:00 -j DROP
```

```
iptables -A INPUT -p udp -m time --timestart 02:00
--timestop 03:00 -j DROP
```

Limiting Connections with IPTables

- Scenario: You suspect a bad actor is attempting to DoS your webserver.
- Tip #1: You can restrict the number of connections a single IP address can have to your webserver.

```
iptables -A INPUT -p tcp -syn -m multiport --dport 80,443 -m connlimit --connlimit-above 20 -j REJECT     --reject-with-tcp-reset
```

Limiting Connections by Time

- Tip #2: You can drop incoming connections if the IP address makes more than 10 connections to port 80/443 in 100 seconds.

```
iptables -A INPUT -p tcp -m multiport --dport 80,443 -m
state d-state NEW -m recent --set
```

```
iptables -A INPUT -p tcp -m multiport --dport 80,443 -m
state --state NEW -m recent --update --seconds 100       -
hitcount 10 -j DROP
```

Log Management

Importance of Log Files

- Benefits
  - Logs provide clues about performance issues, application function problems, intrusion and attack attempts etc.
  - The logs provide vital inputs for managing the computer security incidents, both for Incident Prevention and Incident Response Benefits
  - When responding to computer security incident, logs provide leads to the activities performed over the system
  - Facilitates cyber crime investigation
    - Determine the activity
    - Determine the origin of attack

Log Sources

- System Logs
- Application Logs
- Firewall logs
- IDS/IPS logs
- Application Server Logs
  - Web server
  - Mail server
  - Database server

Types and Issues

- Format
  - W3C
  - NCSA
  - Common Log Format
  - Combined Log Format
  - Syslog
  - Event log
- Type, level
  - Access log, Error log
- Time stamps

```
127.0.0.1 user-identifier frank [10/Oct/2000:13:55:36 -0700] "GET /apache_pb.gif HTTP/1.0" 200 2326
```

Windows Logs

- Security Logs
  - valid and invalid logon attempts
  - resource use such as creating, opening, or deleting files or other objects
- Application Logs
  - events logged by applications or programs
  - Depends on developer
- System logs
  - events logged by system components
  - Ex: Functioning of drivers

Application Server Logs

- Web Server logs
  - Error Logs
  - Access Logs
- Mail Server logs
  - Connection Status
  - SMTP queues
  - Protocol Status (IMAP, POP3, SMTP)
- FTP Server Logs
  - Current logins
  - Commands executed
  - File uploaded and downloaded
- Database Server Logs
  - User activity
  - Objects accessed
  - Creation of new tables, databases, etc..

Challenges in Log File Management

- Non availability of proper logs
  - No auditing
  - Insufficient security
  - Poor management of Logs
- With logs available
  - Volume
  - Storage space, portability
  - Skills

http://www.iitg.ernet.in/cse/ISEA/isea_PPT/ISEA_02_09/Log%20Management.pdf

Local logon attempt failures

Event IDs 529, 530, 531, 532, 533, 534, and 537

Account Misuse

Events IDs 530, 531, 532, and 533

Account Lockouts

Event IDs 539

Terminal Services attacks

Terminal Services sessions can be left in a connected state that allows processes to continue running after the session is ended. Event ID 683 indicates when a user does not log out from the Terminal Services session, and Event ID 682 indicates when a connection to a previously disconnected session has occurred.

http://www.iitg.ernet.in/cse/ISEA/isea_PPT/ISEA_02_09/Log%20Management.pdf

Domain logon attempt failures

Event IDs 675 and 677

Creation of a user account. Event IDs 624 and 626

User account password changed

Event IDs 627 and 628

User account status changed

An attacker may attempt to cover their tracks by disabling or deleting the account used during an attack. All occurrences of Event IDs 629 and 630 should be investigated to ensure that these are authorized transactions. Also look for occurrences of Event ID 626 followed by Event ID 629 a short time later. This can indicate that a disabled account was enabled, used, and then disabled again.

Unix syslog

- A comprehensive logging system, used to manage information generated by the kernel and system utilities
- Allow messages to be sorted by their sources and importance, and routed to a variety of destinations:
  - log files, users' terminals, or even other machines
- Syslogd - the daemon that does the actual logging
- /etc/syslog.conf - configuration file

Example

- Dec 27 02:50:00 bruno ftpd [27876]: open of pid file failed: not a directory

- Dec 27 02:50:00 - Time stamp
- bruno - Terminal Name
- ftpd - Application Name
- 27876 - Process id of the Application
- open of pid file failed: not a directory
  - This is the message text

References

- Convert an address range to CIDR - www.ipaddressguide.com/cidr
- Real-time IPTables Monitor - www.perlmonks.org/?node_id=513732
- FWReport - http://fwreport.sourceforge.net
- Using Afterglow to Visualize IPTables Logs - http://lintut.com/use-afterglow-to-visualize-IPTables-logs-on-centos-rhel-fedora/
- IPTables - http://www.netfilter.org/