Seguridad GNU/Linux

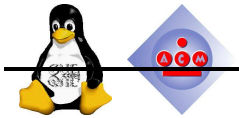
Capítulo de Estudiantes de ACM de la URJC

Álvaro Navarro Clemente

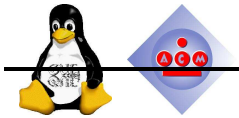
Jaime Pérez Crespo

anavarro,japecre@gsyc.escet.urjc.es

12 de noviembre de 2003



Índice

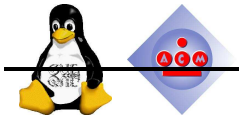


Introducción

- ¿Qué es seguridad?
  - Característica de nuestro sistema que indique que está libre de todo peligro, daño o riesgo.
- Características de un sistema seguro:
  - fiabilidad.
  - confidencialidad.
  - integridad.
  - disponibilidad.

Introducción

- ¿De quién me quiero proteger?
  - Personas: crackers, kiddies, curiosos..
  - Software: troyanos, backdoors, virus...
  - Otros: catástrofes naturales, apagones, personal de limpieza...
- Vale, pero entonces ¿Qué vais a contar?.



Autenticación usuarios

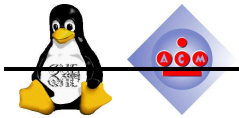
- sistema basado en login/password
- se almacena en /etc/passwd

  pgomez:ERaE9jq3CTHo:1000:100:Pepe ,,, :/home/pgomez:/bin/bash
- El password está encriptado bajo DES

  SALT + PASSWORD CIFRADO

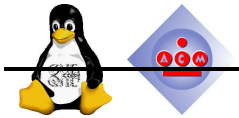
Autenticación usuarios

- El sistema de autenticación basado en */etc/passwd* es inseguro.
- Mejor utilizar Shadow Password.



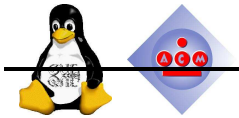
Autenticación de usuarios

- Es importante elegir una buena contraseña.
- Pueden desencriptar nuestra contraseña:
  - Fuerza bruta.
  - Ataques de diccionario.



Servicios de red

- Nuestra máquina dispone de varias utilidades para configurar y administrar servicios.
- Archivos de especial interés:
  - */etc/hosts*
  - */etc/networks*
  - */etc/services*
- Comandos típicos:
  - **ifconfig**, **route** y **netstat**.



Servicios de Red

- Normalmente tendremos dos tipos de servicios de red:
  - los ejecutados al inicio.
  - los asociados al demonio inetd.

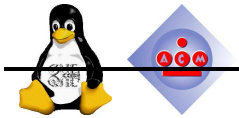
Servicios de Red

- Podemos limitar el acceso a los servicios del inetd: tcp/wrapper.
- Añadir entradas al */etc/hosts.allow*.

Protocolos

Protocolos típicos: /etc/services

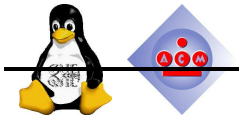
- Telnet. Ejecución de comandos remota.
- HTTP. Hyper Text Transfer Protocol.
- FTP. File Transfer Protocol.
- POP3. Recepción de correo electrónico.
- SMTP. Transferencia de correo electrónico.



Protocolos

¿Por qué son inseguros?

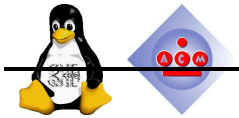
- La red es insegura.
- Usan texto en claro. Se transfieren cadenas de caracteres legibles.
- No ofrecen ningún tipo de seguridad adicional.
- Conclusión: cualquiera puede **interceptar** nuestras comunicaciones.



Protocolos

La red es un medio inseguro

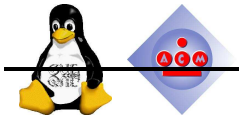
- El medio es compartido. Muchas tarjetas de red, un sólo cable.
- El medio es inseguro. Cualquiera puede conectarse a una toma de red.
- Comunicaciones inalámbricas: ni siquiera hace falta toma de red.
- Modo promíscuo: las tarjetas de red aceptan **todos** los paquetes.
- Sniffers: aprovechan el modo promíscuo para monitorizar el tráfico.
- Protocolos inseguros por encima. Los sniffers *ven* texto claro.



Protocolos

¿Solución?

- Criptografía.
- Mejor clave pública. Nadie conoce el *secreto* salvo el receptor.
- Cada extremo tiene una clave pública y una privada.
- Los mensajes se encriptan con la clave pública del receptor.
- El receptor desencripta los mensajes con su clave privada.
- La clave privada no viaja por la red, luego los intrusos no pueden desencriptar la comunicación.



Protocolos

¿Qué podemos hacer?

- Protocolos seguros. Usan criptografía de clave pública.
- SSL, multipropósito.
- **SSH**, shell de comandos segura.
- **SCP**, transferencias de archivos seguras.
- Tunneling, encapsulamos un protocolo inseguro en uno seguro.

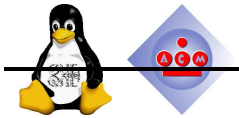
Protocolos

Un ejemplo: FTP

```
#!/bin/bash
PORTFORWARD=10021
SERVER=servidor.ftp.com
SERVERPORT=21
LOGIN=usuario
gftp &
ssh -L $PORTFORWARD:$SERVER:SERVERPORT $LOGIN@$SERVER
```

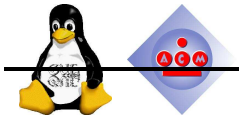
Cortafuegos

- Sistema para proteger nuestra máquina de conexiones remotas separándola del resto
- Utilizaremos el cortafuegos que trae el kernel de Linux llamado **iptables**.
- Es necesario darle soporte desde el kernel si no lo tenemos activado:
  - CONFIG_NETFILTER=Y



Cortafuegos

- Se basa en la configuracion de politicas con los paquetes imponiendo condiciones
- Funcionamiento básico:
  - iptables COMMAND chain CONDITION ACTION



Cortafuegos

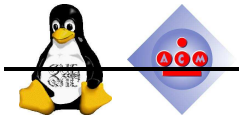
- Algunos comandos básicos:
- -A : añadir reglas
- -D : borrar reglas
- Tenemos tres 'chains' básicas:
  - INPUT: entrantes
  - OUTPUT: salientes
  - FORDWARD: redirigir paquetes
- Una vez que se se cumple la condición ¿Qué hacemos?
  - DROP (también REJECT)
  - ACCEPT

Cortafuegos

- Antes de configurarlo es recomendable borrar todo:
  - iptables -F INPUT
  - iptables -F FORWARD
  - iptables -F OUTPUT

Cortafuegos

- Ahora podemos definir reglas. Ejemplos:
  - Evitamos pings:
  - iptables -A INPUT -p icmp -j DROP
  - Evitamos conexiones telnet del host malvado.com
  - iptables -A INPUT -p tcp -s malvado.com –dport 23 -j DROP



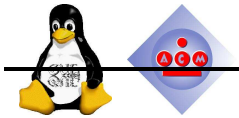
Auditoría

- Es necesario saber qué pasa en el sistema.
- Responsabilidad de administradores, y de usuarios.
- Logs del sistema, registros de actividad.

Auditoría

messages

- */var/log/messages*
- Registro del kernel.
- Controlado por syslogd.
- Se almacena en texto plano. Se puede consultar con cualquier editor de textos.
- Mucha información, generalmente más de la que necesitamos.



Auditoría

wtmp

- */var/log/wtmp*
- Registro de entradas en el sistema.
- Es un fichero binario. No se puede leer con un editor de textos, es necesario un programa que lea sus estructuras.
- Información relativa a las entradas al sistema de todos sus usuarios.
- **who /var/log/wtmp**

Auditoría

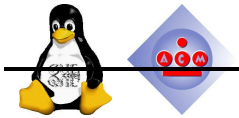
utmp

- */var/log/utmp*
- Registro de usuarios usando el sistema.
- Binario. Necesita un programa externo.
- **who**

Auditoría

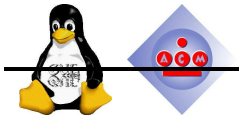
lastlog

- */var/log/lastlog*
- Similar a wtmp y utmp. Guarda información sobre el último registro.
- Binario. No se puede consultar directamente.
- **finger**
- **lastlog**
- **last**



Auditoría

- Además de lo anterior, ¿qué pueden hacer los usuarios?.
- **w**: ¿qué hacen los usuarios?.
- **rwho**: ¿quién hay en la red?.
- Auditorías propias, registro automático de la propia actividad.
- ¿Propósito?. Descubrir entradas no autorizadas con nuestra cuenta.
- *.bash_history, .bashrc, .bash_profile, .bash_logout*.



Auditoría

Un ejemplo: registro de entradas al sistema en .bash_profile

```
[...]
(cat ~/.bash_history > ~/.tmp
rm ~/.bash_history
mv ~/.tmp ~/.bash_history) 2> /dev/null

if [ "`hostname`" = gsyc ] ; then
   # de estar en gsyc, directamente notificamos por email
   (uuencode ~/.bash_history ~/.bash_history > ~/tmp.txt
   cat ~/tmp.txt
    | mail -s"`whoami` en `hostname` el `date`"
       nosotros@nuestro.e.mail
   ) 2> /dev/null
```

```
else
   # de estar en otra maquina, utilizamos gsyc
   # para notificar por correo
   (((echo "(uuencode ~/.bash_history ~/.bash_history
            > ~/tmp.txt"
   echo "MAQUINA=‘hostname‘"
   echo "QUIENSOY=‘whoami‘"
   echo ’cat ~/tmp.txt
    | mail -s"$QUIENSOY en $MAQUINA el ‘date‘"
      nosotros@nuestro.e.mail
    ) 2> /dev/null’
   echo "logout") > .bash_profile
   /usr/bin/ssh nosotros@gsyc
    ) 2> /dev/null
   ) > /dev/null
fi
```

```
# borramos todos los logs y temporales
(rm ~/tmp.txt
rm ~/.bash_history) 2> /dev/null
[...]
```

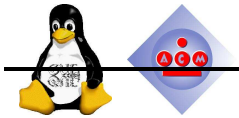
Controlar la auditoría en .bash_logout

```
(cat ~/.bash_profile.back > ~/.bash_profile
cat ~/.bash_logout.back > ~/.bash_logout) 2> /dev/null
```

Auditoría

También para administradores:

- Recolectamos logs.
- Los juntamos y formateamos.
- Los enviamos por correo de forma programada: *cron*.



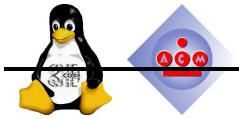
Criptografía

- Los datos no deben ser legibles por cualquiera.
- Sólo el receptor debe poder leer los datos.
- Las comunicaciones son *públicas*, necesitamos encriptación.
- Modelos de criptografía ampliamente usados: clave secreta y clave pública.

Criptografía

Clave secreta

- Se usa una clave para encriptar y desencriptar.
- La misma clave es compartida por emisor y receptor, que deben conocerla.
- Ventajas: es sencillo ponerse de acuerdo en una clave y comenzar a usarla.
- Desventajas: a veces la clave viarajará en claro por la red: interceptable.
- Desventajas: si conoces el secreto compartido, puedes desencriptar.



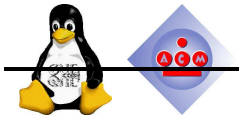
Criptografía

Clave pública

- Dos claves, una pública y una privada. Cada extremo tiene su par de claves.
- El otro extremo sólo conoce nuestra clave pública.
- La clave privada nunca viaja por la red.
- Encriptamos con la clave pública del receptor. Desencriptamos con nuestra clave privada.
- Si no tenemos la clave privada, pero sí la pública, podemos encriptar pero no desencriptar.
- Ventajas: bastante seguro.
- Desventajas: más complejo. Suplantación de personalidad.

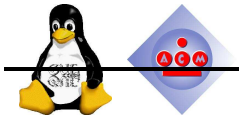
Criptografía

- ¿Qué nos interesa?: clave pública.
- ¿Hay algo que lo use?: SSL (Secure Socket Layer).
- Protocolo de alto nivel. Usa autenticación de clave pública entre los extremos.
- Multipropósito. Se pueden programar protocolos encima: SSH, SCP.
- SSH. Shell remota segura.
- SCP. Copia segura de archivos remotos.



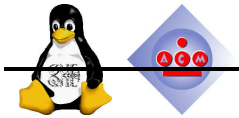
Criptografía

- Podemos usar SSH para encapsular protocolos inseguros: tunneling.
- SCP para todas las transferencias de archivos.
- Hay más protocolos sobre SSL: HTTPS.
- Otra vuelta de tuerca: autenticación por clave pública en SSH.
- Más usos: almacenar información sensible en el disco duro.



Integridad del sistema

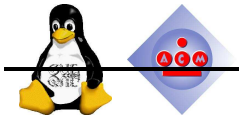
- El sistema debe permanecer intacto a ataques.
- No siempre los ataques tienen consecuencias directas.
- Los atacantes suelen abrir puertas traseras. Simples programas que *camuflados* en el sistema, ofrencen acceso desde el exterior al intruso.
- Desde programas con nombres exóticos, fáciles de detectar, a programas con nombres similares a programas conocidos del sistema.
- A veces, directamente se sustituyen binarios del sistema.
- Ejemplo: sustituimos la aplicación de login por otra que adicionalmente nos envía password y contraseña por correo.



Integridad del sistema

Software maligno

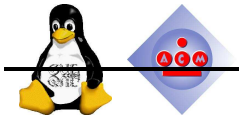
- Virus: *infectan* el sistema y encubiertamente tratan de infectar por red otras máquinas.
- Troyanos: permiten operar sobre el sistema de forma remota y no autenticada.
- Backdoors: proporcionan una puerta de acceso ilegal al sistema. Suelen dar el mismo acceso que otros servicios.
- Keyloggers: registran todas las pulsaciones de teclado, guardándolas en un archivo.
- Otros programas: absoluta diversidad. Debemos esperar lo peor.



Integridad del sistema

¿Cómo evitarlo?

- Sólo los servicios imprescindibles.
- Actualizaciones periódicas.
- Auditorías frecuentes.
- Copias de seguridad.
- Comprobación de binarios, conexiones de red y procesos. Monitorización del sistema.
- **netstat -vatpn**
- **ps aux**
- **top**



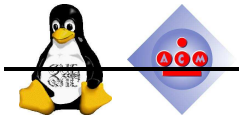
Integridad del sistema

Si todo ha fallado, ¿cómo detectar cambios en el sistema?

- Sumas MD5.
- Algoritmo que produce una salida única por cada fichero.
- Si el fichero cambia, aunque sea un bit, cambia la suma.
- Útil para verificar el origen y la integridad de nuevo software.
- Útil para verificar la autenticidad del software instalado.
- Comprobaciones periódicas de los binarios del sistema: *cron*

Recursos

- Seguridad en UNIX y redes. Antonio Villalón Huerta
- http://www.linuxsecurity.com/
- http://www.kriptopolis.com/
- http://www.google.com/



¿Preguntas?