SISTEMAS INFORMÁTICOS

SOFTWARE Y LICENCIAS

LENGUAJES DE PROGRAMACIÓN

SEGURIDAD

1. CONCEPTOS BÁSICOS

Informática

Conjunto de conocimientos científicos y técnicas que hacen posible el tratamiento automático de la información por medio de ordenadores (RAE).

Conceptualmente, se puede entender como aquella disciplina encargada del estudio de métodos, procesos, técnicas, desarrollos y su utilización en ordenadores (computadoras), con el fin de almacenar, procesar y transmitir información y datos en formato digital[1].

Ordenador, computador o computadora

Máquina electrónica dotada de una memoria de gran capacidad y de métodos de tratamiento de la información, capaz de resolver problemas aritméticos y lógicos gracias a la utilización automática de programas registrados en ella (RAE).

Un ordenador es una máquina compuesta de elementos físicos tanto mecánicos como electrónicos. Los primeros permiten que se ponga en marcha una unidad de disco, por ejemplo. Los segundos nos permiten realizar trabajos con gran velocidad y precisión siempre, claro está, que le demos el programa adecuado.

Tratamiento de la información

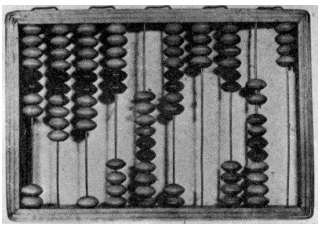
Para poder ejecutar los programas precisamos de un **sistema informático** que es el conjunto de elementos que nos permiten introducir la información, tratarla y obtener los resultados deseados. Este conjunto de operaciones con la información se denomina **tratamiento de la información**:

Entrada:	Proceso:	Salida:
• recogida de los datos	• tratamiento de los datos	• recogida de resultados
• depuración de datos	• almacenamiento de los resultados	• distribución de los resultados
• almacenamiento de datos	(no siempre necesario)	a quien los necesite

El programa (o algoritmo) que utilizamos para transformar los datos en resultados requiere un análisis previo a su introducción y ejecución en el ordenador:

Planteamiento de un problema + Análisis del mismo = Algoritmo de resolución (programa) ⇒ Ejecución en él ordenador

2. EVOLUCIÓN HISTÓRICA DE LA INFORMÁTICA Y LOS ORDENADORES.



El término calcular procede del latín **calculus** que eran unas piedrecitas que se usaban para realizar cuentas. A partir de esta primera ayuda se desarrollo el primer aparato conocido para calcular: el **ábaco** que data de, aproximadamente, 3500 a.C. Hoy en día siguen usándose, sobre todo en los países asiáticos. Consiste en una serie de cuentas de madera ensartadas en varillas. Estas últimas están montadas sobre un armazón

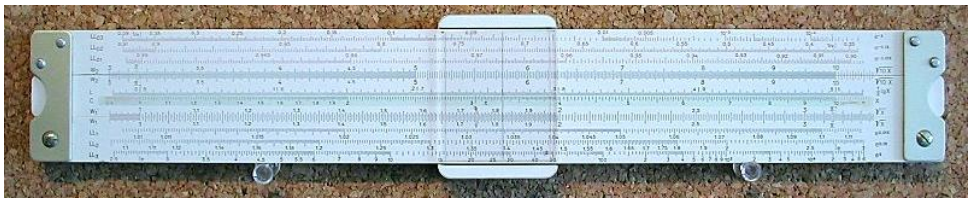
[1] Codificadación de la información mediante dos únicos estados. A dichos estados se les puede llamar "verdadero" o "falso", o más comúnmente 1 y 0, refiriéndose a que en un circuito electrónico digital hay dos niveles de tensión

dividido en dos partes. Aunque parezca increíble, en muchos sitios se fían más de los resultados obtenidos con el ábaco que de los del ordenador.

Hasta finales del siglo XVI la única forma de realizar operaciones matemáticas era a mano o con un ábaco. Fue entonces cuando John Napier ideó un dispositivo, basado en palillos, capaz de multiplicar y dividir automáticamente. En 1623 Wilhelm Schickard construyó una calculadora mecánica basada en ruedas dentadas capaz de multiplicar.

En 1642 **Blaise Pascal** inventó la primera máquina automática de calcular basada en ruedas dentadas y que simulaba el funcionamiento del ábaco, realizando sumas y restas. Se la llamo **Máquina aritmética de Pascal**.

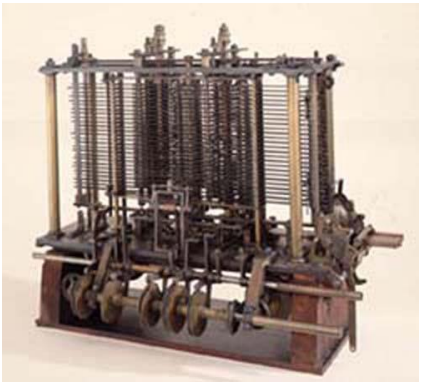
En 1650 Patridge inventó la **regla de cálculo** que se ha utilizado hasta los años 60 en que fue sustituida por las calculadoras. Con ellas se podían realizar cálculos altamente sofisticados. Baste con pensar que los ordenadores se crearon a partir de cálculos realizados con ellas.



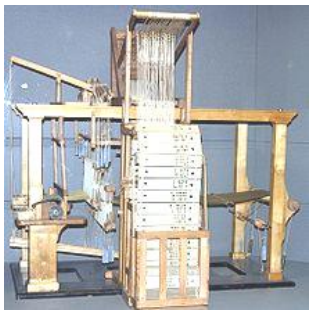
En 1672 Gottfried Wilhelm von **Leibnizt** mejoró la máquina de Pascal obteniendo la **calculadora universal** que además de sumar y restar, multiplicaba, dividía y extraía raíces cuadradas.

A finales del siglo XVIII, Joseph Marie **Jacquard** construyó un telar que controlaba el tejido y los dibujos con tarjetas perforadas. Se la considera la primera máquina mecánica programada.

En 1822 **Charles Babbage** diseñó la **máquina de diferencias** que no llegó a fabricarse por faltar la tecnología necesaria para elaborar sus piezas. En 1833 diseñó la **máquina analítica** que, por lo mismo que la anterior, tampoco se construyó. Sin embargo, era similar al ordenador actual ya que disponía de todos los elementos del gráfico del apartado 1.3. excepto las unidades de E/S. Fue construida posteriormente (1854) por Scheutz, funcionando con bastante éxito. Se considera a Babbage el **padre de la informática**. Se considera a lady **Augusta Ada Byron** (hija de Lord Byron) la **primera programadora** por los trabajos que realizó en la prueba de la máquina analítica.



En 1854 George **Boole** desarrolló la teoría del **Álgebra de Boole** que sentó las bases para el desarrollo de circuitos lógicos.

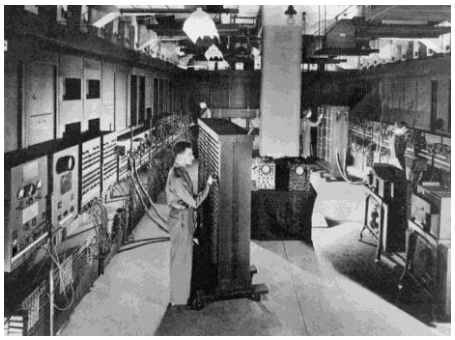


Entre 1885-90 Herman **Hollerit** utilizó las ta rjetas perforadas para elaborar el censo estadounidense. Partiendo de las tarjetas de Jacquard, diseño unas nuevas y una máquina capaz de leer y tabular dicha información, la **máquina censora** o **tabuladora**, que realizó el censo de 1890 en sólo tres años (antes se tardaba 10) y procesó las tarjetas de 56 millones de personas. Posteriormente, incorporó a la máquina la operación de sumar de cara a su utilización en contabilidad. En 1896 fundo la empresa Tabulating Machines Company, que derivó en 1924 en International Business Machines (**I.B.M**.)

La primera calculadora fue construida en 1893 por Otto **Steiger**. Se utilizó en los grandes negocios y en algunas aplicaciones científicas.

Con los estudios de Alan M. **Turing** de 1936 sobre una máquina capaz de resolver todo tipo de problemas, surge la **teoría matemática de la computación**, que define un **algoritmo** como la representación formal y sistemática de un proceso. En ella se demuestra que no todos los problemas tienen solución algorítmica. De ella deriva la **teoría de la computabilidad** que engloba el conjunto de estudios encaminados a encontrar formas de descripción y representación de algoritmos.

En 1937 Howard H. Aiken, junto con científicos de la I.B.M., desarrolla la idea de Babbage construyendo la primera computadora electromecánica basada en relés, ruedas dentadas, etc. Fue la Calculadora Automática de Secuencia Controlada y se la llamo **MARK-I**. Tenia 17 m de largo por 2 m de alto, pesaba unas 70 Tm y tenía unos 800000 m de **cableado** (en el que estaba la programación)



En 1945 se pone en marcha el **ENIAC** (Electronic Numerical Integrator and Calculator), primer ordenador electrónico. A pesar de su aparatosidad (pesaba 30 Tm, ocupaba 111 m^3, tenía 17000 **válvulas de vacío** y consumía tanta energía que las luces de Filadelfia disminuían cuando lo ponían en marcha) el ENIAC fue creado con el único fin de calcular trayectorias de proyectiles para el ejército.

En 1944 John von **Neumann**, basándose en los trabajos de Augusta Ada, desarrolla la idea de una máquina capaz de ser programada. A raíz de sus estudios, se creó en 1952 el EDVAC, que era una modificación del ENIAC.

En 1951 sale al mercado el **UNIVAC-I** que permitía su programación mediante **cintas magnéticas** (hasta la fecha se utilizaban tarjetas y cinta de papel perforada). Un año más tarde se construyen el MANIAC-I, MANIAC-II y el UNIVAC-II, con lo que termina la prehistoria de los ordenadores. A partir de este momento dejan de utilizarse las válvulas de vacío y empiezan a utilizarse **transistores**, **diodos** y **núcleos de ferrita** (para la memoria).

Evolución de los componentes electrónicos

La posterior aparición de los **circuitos integrados (chips)** aceleró la evolución de los ordenadores. En sólo 45 años se ha pasado de ordenadores mastodónticos pero de muy poca potencia a los modernos portátiles de alta velocidad y gran capacidad de almacenamiento. Esta evolución se ha dividido en las denominadas generaciones:

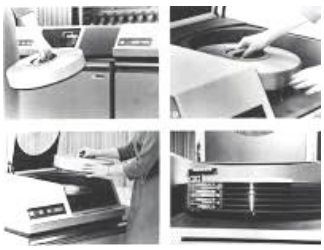
1ª generación: 1940-1952. Formada por los ordenadores de válvulas de vacío y con uso restringido al mundo militar y científico. Usaban **lenguaje máquina** para programarlas y no poseían dispositivos de almacenamiento de datos.

2ª generación: 1952-1964. Se sustituyen las válvulas por transistores. Las memorias se hacen con núcleos de ferrita. Los ordenadores ganan potencia y fiabilidad y pierden tamaño y consumo. Se amplia el uso a las empresas. Surgen los lenguajes evolucionados como el **ensamblador** y los pioneros de **alto nivel** (COBOL para el mundo empresarial y ALGOL y FORTRAM para el científico).

3ª generación: 1964-1971. Aparece el circuito integrado. Las memorias se hacen con semiconductores. Se amplia la miniaturización a los todos los componentes, apareciendo los miniordenadores y aumentando la potencia y disminuyendo el tamaño y consumo aun más que en la generación precedente. Se extiende el uso de los ordenadores, lo que origina la evolucionan los S.O. que permiten ya la multiprogramación, el tiempo real y el modo interactivo. Aparecen los discos magnéticos.

4º generación: 1971-1981. Aparece el **microprocesador** (toda la U.C.P. está en un único chip. La miniaturización permite que los ordenadores entren en el ámbito domestico con la aparición de los PC (Personal Computer). Estos caben perfectamente en una mesa, disponen de unidades de casete o de disquete que les permite modificar la programación a voluntad. Aparecen los S.O. genéricos que permiten intercambiar programas entre ordenadores de distinta marca y/o modelo. Surgen gran cantidad de lenguajes de alto nivel y las redes.





Olivetti P6060 (1975) y P6066 (en la imagen) fue el primer ordenador con todo integrado: unidad de disquete de 8" de 256KB, display de 32 caracteres, impresora térmica integrada de 80c y teclado. Tenía RAM dinámica, con una capacidad de 32 o 64 KB y un lenguaje propio, basado en BASIC, con funciones matématicas y graficas integradas. Se le podía incorporar un monitor monocromo, impresora de carro ancho o plotter y unidades de disco duro.

Osborne 1, el primer "portátil". Salió al mercado en 1981 y sólo pesaba algo mas de 10 kilos. Eso sí, tenía 64 Kb de RAM

3. SISTEMA INFORMÁTICO

Es un conjunto formado por las 3 partes siguientes interrelacionadas:

1. Física: hardware
2. Lógica: software
3. Humano: desarrolladores (analistas y programadores), implementadores y usuarios

3.1. TIPOS DE SISTEMAS INFORMÁTICOS

- Según su uso:
  - De uso general
  - De uso específico (robots industriales, videoconsolas…)
- Según sus prestaciones:
  - **Supercomputadoras**: gran capacidad de cálculo. Uso técnico-científico o simulaciones (Cray Jaguar posee en total 224.162 núcleos de procesamiento). Actualmente suelen ser cluster de ordenadores "blade". Se suele dar su rendimiento en gigaFLOPS.
  - **Mainframes o macrocomputadoras** (computadoras centrales): una computadora grande y potente usada principalmente por una compañía para el procesamiento de una gran cantidad de datos o dar soporte a grandes redes de comunicaciones. Su principal diferencia con las supercomputadoras es el manejo de grandes volúmenes de E/S
  - **Terminales "tontos**" dependen del mainframe o servidor para la ejecución de procesos ya que no tienen capadidad de procesamiento ni de almacenamiento.
  - **Miniordenadores** : servidores con terminales inteligentes.
  - **Microordenadores**: equipos monousuario (PCs, sobremesa, portátiles).

- **Estaciones de trabajo**: equipos monousuario de gran potencia y grandes prestaciones. Se diferencian de los microordenadores en que su hardware esta optimizado para ciertas tareas (diseño y CAD).

Nota: Actualmente, el uso general y específico está bastante difuso en algunos casos ya que, por ejemplo, una videoconsola permite conectarse a Internet entre otras cosas. Lo mismo pasa con las estaciones de trabajo y los microordenadores.

Investiga 1:

1. Nombra y explica las principales características y usos de 3 supercomputadoras.
2. Busca información sobre la Red española de supercomputación.
3. Nombra tres empresas/procesos que se realicen con mainframes.

4. EL SOFTWARE

4.1. TIPOS DE SOFTWARE

- **Software de base** (firmware y sistema operativo):
  - **Firmware**: programa para propósitos específicos que controla los circuitos electrónicos de un dispositivo de cualquier tipo. Al estar integrado en la electrónica del dispositivo (en una memoria ROM) es en parte hardware, pero también es software. Es el intermediario (interfaz) entre las órdenes externas que recibe el dispositivo y su electrónica.
    - BIOS: firmware de la placa base del ordenador que se ejecuta al encenderlo. Una parte, el POST, reconoce y chequea los distintos componentes (procesador, memoria, teclado, discos duros, etc.). Otras partes se cargan en memoria para permitir al SO el acceso a los componentes. También contiene un programa de arranque (boot) que busca el programa de inicio del sistema operativo en los distintos dispositivos de almacenamiento (disco duro, CD-ROM,..)
  - **Sistema operativo (SO):** conjunto de aplicaciones que gestionan los recursos del sistema y optimiza su uso, actuando como intermediario entre el usuario y el hardware.
- **Software de aplicación**: programas de usuario, aplicaciones y suites
  - **Programa**: Conjunto de instrucciones que permite a un ordenador realizar funciones diversas para la resolución de una tarea.
  - una **aplicación** es un tipo de programa informático diseñado como herramienta para **permitir** a un usuario realizar uno o diversos tipos de trabajo.
  - Una **suite** (o paquete integrado) es una agrupación diversos programas y/o aplicaciones de distinta naturaleza

4.2. LICENCIAS

Una licencia es un contrato mediante el cual una persona recibe de otra el derecho de uso, de copia, de distribución, de estudio y de modificación (en el caso del Software Libre) de varios de sus bienes, normalmente de carácter no tangible o intelectual, pudiendo darse a cambio el pago de un monto determinado por el uso de los mismos. Asi pues, lo que compramos no es el software en si mimos, sino su derecho de uso.

Resumiendo, en cuanto al software, una **licencia** es un contrato mediante el cual un usuario recibe de una persona o empresa el derecho de uso de su software cumpliendo una serie de términos y condiciones establecidas dentro de sus cláusulas.

4.2.1 TIPOS DE LICENCIAS

En función del tipo de software las aplicaciones pueden ser:

a) **Gratuitas (*freeware*)** o **comerciales**.

b) **Libres** o **propietarias**. Libres se basan en la distribución del código fuente junto con el programa, así como en las cuatro premisas del software libre. Que una aplicación sea libre no implica que sea gratuita.

   El software propietario es aquel en que los usuarios tienen limitadas las posibilidades de usarlo, modificarlo o redistribuirlo (con o sin modificaciones).

c) ***Opensource*** **(**código abierto) o **privativas** (código fuente no disponible o restringido).

Algunos tipos de licencias privativas del software son:

- **OEM**: La venta del software forma parte de un equipo nuevo
- **Retail**: El programa en este caso es enteramente del usuario que puede cederlo a terceros o venderlo.
- **Por volumen**: un número determinado de equipos pueden usar el mismo código de licencia. No se puede ceder a terceros.

El software libre está sujeto a su vez a una serie de licencias, cada una de ellas con sus respectivas normativas.

Investiga 3:

¿Qué implican los siguientes términos en cuanto a tipos de licencia?

GPL

BSD

Copyleft

Licencia de software libre

Licencia de software propietario

Freeware

Shareware

Opensource

EULA

Indica cuales de estos términos pueden referirse a un mismo producto y cuales son contradictorios.

5. LENGUAJES DE PROGRAMACIÓN

Se considera lenguaje de programación cualquier lenguaje artificial que puede utilizarse para definir una secuencia de instrucciones (programa) para su procesamiento por un ordenador. Existen dos niveles básicos de lenguajes, dependiendo del nivel de acercamiento al lenguaje humano (normalmente, el inglés):

5.1. LENGUAJES DE BAJO NIVEL.

Son aquellos en los que las instrucciones están codificadas en un lenguaje más próximo a la máquina que al hombre. Los programas en lenguajes de bajo nivel, al ser órdenes directamente inteligibles por el procesador, son muy rápidos pero el lenguaje en sí es, a menudo, difícil de aprender.

Además, un programa en lenguaje de bajo nivel raramente puede ser ejecutado en otra máquina, ya que las órdenes son específicas de un procesador en concreto.

5.1.1 CÓDIGO MÁQUINA.

Son aquellos en los que las órdenes se codifican de forma directamente entendible por el ordenador, es decir, en binario. Tanto las ordenes como los datos deben introducirse en binario. Además, las órdenes deben corresponderse con aquellas que comprende directamente el procesador, por lo que una orden simple, como sumar dos números almacenados en memoria, debe descomponerse en todos sus pasos:

leer la dirección correspondiente al primer dato,
llevarlo al circuito sumador,
leer el segundo dato,
llevarlo al circuito sumador,
asignar una dirección de memoria para el nuevo dato,
trasladar el dato resultante del registro temporal del procesador a memoria.

Todo esto codificado a base de 0 y 1.

Al no necesitar traducirse, son muy rápidos en la ejecución aunque el tiempo necesario para escribir el programa y depurarlo es mucho mayor que en otros lenguajes.

Hoy en día este tipo de lenguajes está en desuso aunque era el único que entendían los primeros ordenadores, como el ENIAC.

5.1.2 LENGUAJE ENSAMBLADOR.

Segundo paso en el desarrollo de los lenguajes. Funcionan prácticamente igual que el código máquina, salvo que se sustituye el binario en las órdenes por código alfanumérico mnemotécnico (generalmente unas pocas letras) y los datos por hexadecimal. Además, algunas operaciones básicas se reagrupan en un sólo código por lo que la orden anterior quedaría resumida a:

```
CAR A, M(H0007)
SUM A, M(H0009)
MEM M(H000C), A
```

Este lenguaje simplifica claramente el proceso de escritura y depuración de los programas, ya que es más fácil aprenderse los códigos de las instrucciones y, al estar los datos en hexadecimal, también resulta más fácil la conversión a decimal.

Sin embargo, presenta el mismo problema que el código máquina en cuanto a su utilización en diversos ordenadores ya que los lenguajes ensambladores son específicos de cada procesador e incompatibles entre si.

Además, es necesario traducir el programa generado por el usuario (código o programa fuente) a código máquina. Esto se hace con los ensambladores, que generan a partir del fuente el programa ejecutable.

Este programa ejecutable no necesita ser ensamblado cada vez que se ejecuta pero si cada vez que realicemos cambios en el fuente.

5.2. LENGUAJES DE ALTO NIVEL.

Son aquellos en los que el programa fuente se aproxima más al lenguaje humano que al código máquina. Entre sus características podemos destacar:

- Portabilidad: Son independientes del hardware, lo que permite al programador abstraer el problema del entorno en que será utilizado posteriormente. Un programa puede generarse en un tipo de máquina y ejecutarse posteriormente en otras de distinta arquitectura.
- Utilizan ordenes englobantes (una sola orden puede equivaler a cientos de ordenes máquina). Siguiendo el ejemplo anterior, la única orden necesaria es `A = B + C`.
- Precisan ser traducidos a código máquina antes de ejecutarlos. Para ello se puede recurrir a la interpretación o a la compilación que explicaremos más adelante.
- Sus ordenes, aunque codificadas, suelen utilizar palabras claves, en inglés normalmente, (SORT ordenar) o acronimos (CLS CLear Screen)
- Los programas en lenguaje de alto nivel (módulo fuente) requieren ser traducidos a código ejecutable mediante compiladores (con los que obtenemos un módulo ejecutable) o interpretes.

El primer compilador para lenguajes de alto nivel fue desarrollado a principio de los 50 por Grace Hopper mientras desarrollaba el FLOWMATIC, lenguaje orientado al mundo empresarial.

Se considera al FORTRAN (*FORmula TRANslation*, 1954-57) como el primer lenguaje de alto nivel de uso generalizado. Está diseñado para el tratamiento de formulas matemáticas complejas. Con él aparecen los conceptos de variable, instrucciones condicionales y subrutinas independientes del programa principal.

En el año 59 aparece el COBOL (*COmmon Business-Oriented Language*) orientado a aplicaciones comerciales. Su diseño hace hincapié en las estructuras de datos de gran volumen, permitiendo un magnifico manejo para la organización de datos y manipulación de archivos. Además, presenta la característica de ser el lenguaje más similar al inglés ya que sus ordenes pueden escribirse como frases comunes (`TOTAL = IMPORTE + IVA`, en BASIC, `ADD IMPORTE TO IVA GIVING TOTAL`, en COBOL).

El lenguaje BASIC (*Beginners All-purpose Symbolic Instruction Code*) fue desarrollado a principio de los 60 y estaba dirigido a la enseñanza de la programación. Fue el primer lenguaje popularizado debido a la expansión de los microprocesadores en el ámbito doméstico que incorporaban el intérprete de BASIC en la ROM ya que es un lenguaje interpretado, lo que facilita la escritura y depuración de los programas por gente con pocos conocimientos metodológicos.

En la actualidad, los lenguajes más extendidos son el C (1972) y sus derivados, tanto en el entorno de desarrollo de sistemas (en él están desarrollados los sistemas operativos UNIX y LINUX) como en el de las aplicaciones, y los lenguajes orientados a objeto, con un cambio radical en la filosofía de la programación. Ya no se trata de dar una serie de ordenes consecutivas, sino que se manipulan objetos, que actuarán o no ante determinados eventos. Un programa orientado a objetos consta de un conjunto de objetos (ventanas, botones, zonas de entrada de texto, tablas, ficheros, etc.) interrelacionados a través de los eventos que soportan (implementados en el lenguaje) y los métodos desarrollados mediante programación para cada uno de ellos.

En la década de los 90 el C++ es el lenguaje orientado a objeto más popular, seguido por el JAVA (evolucionado del C++) que fue creado específicamente para programar en Internet, aunque puede ser usado para aplicaciones de propósito general.

Los lenguajes visuales (para programar en S.O. visuales como Windows) son el siguiente paso. Con ellos, y mediante programación orientada a objetos, se está programando en la actualidad (VisualBasic, Delphy, Objetive Cobol, C++ Builder, etc.)

5.3. LA TRADUCCIÓN.

Consiste en convertir un programa fuente (en texto ASCII) en ejecutable (en código máquina). Dependiendo del lenguaje, se utilizarán unas herramientas u otras para realizar el proceso. Estas herramientas (ensambladores, compiladores, linkadores o encuadernadores) forman parte del sistema operativo aunque vengan integradas en el paquete del lenguaje ya que el resultado que se obtiene con ellas es específico para un determinado procesador (ya que cada uno tiene su propio código máquina) y S.O.

En los lenguajes ensambladores, la traducción se realiza en un sólo paso ya que las ordenes del código fuente son equivalentes a las del código máquina, excepto que las primeras utilizan una notación mnemotécnica para facilitar el trabajo del programador.

En los lenguajes de alto nivel, la traducción puede hacerse con dos modalidades.

5.3.1 INTERPRETACIÓN.

Un interprete hace que el programa fuente vaya, sentencia a sentencia, traduciéndose y ejecutándose. El intérprete coge una sentencia, la analiza y traduce a código máquina y la ejecuta antes de proceder con la siguiente. La traducción no se almacena por lo que si la orden debe volver a ejecutarse, la traducirá de nuevo.

Entre los inconvenientes de este método cabe destacar:

- Si una orden ha de ejecutarse 100000 veces (bucles) la traducirá 100000 veces.
- Un simple error sintáctico sólo será detectado al ejecutarse la línea correspondiente. Si el error está en una bifurcación condicional, por ejemplo, en un bloque de instrucciones que sólo se ejecutarán en una determinada fecha, sólo lo detectaremos cuando se cumpla esa condición. Por ello, un programa nos puede dar un error a pesar de haberse ejecutado correctamente cientos de veces.
- Necesitan ejecutarse a través del interprete por lo que requieren el llamado entorno del lenguaje. Si trasladamos el programa a otra máquina tendremos que trasladar también el entorno.
- El número de órdenes y funciones que manipulan es bastante limitado

Las ventajas que presenta son:

- Los programas suelen ser cómodos de depurar ya que la ejecución se para al detectar cualquier error (sintáctico o de ejecución), permitiendo al programador depurarlo.
- Podemos parar la ejecución en cualquier punto, analizar y/o modificar los valores de los datos y reanudarla.
- Permiten ejecutar la mayoría de las órdenes aisladamente, desde la línea de comandos del entorno.
- El entorno suele ocupar poca memoria.

Todas estas características hacen que los lenguajes interpretados sean muy cómodos para el aprendizaje aunque engorrosos para la programación profesional.

Entre los lenguajes interpretados destacan el dBASE y el BASIC (que incluso iba incorporado en la ROM en los primeros ordenadores domésticos).

5.3.2 COMPILACIÓN.

Consiste en traducir la totalidad del fuente en una sola operación, generando como resultado un nuevo fichero que contendrá el programa ejecutable. Este último podrá ejecutarse en cualquier máquina con el mismo tipo de procesador, sin necesidad de que las herramientas utilizadas para la generación del programa estén presentes.

Aunque genéricamente se usa el término compilación para el proceso de convertir un fuente en ejecutable, el proceso consta de dos fases claramente diferenciadas:

COMPILACIÓN DEL FUENTE

Primero se hace la revisión léxica, sintáctica y semántica de cada una de las ordenes, es decir, comprobar que las ordenes corresponden a las que admite el lenguaje, que están estructuradas correctamente y que el orden y tipo de los parámetros sea el correcto.

A continuación, convierte las ordenes en texto ASCII a un código intermedio (similar al ensamblador en algunos casos o a código máquina en otros) generando con él el llamado programa objeto.

En cualquier caso, el objeto no es ejecutable directamente ya que le hace falta que se le incorporen las rutinas externas a las que hace llamadas, bien sean estas pertenecientes al usuario o al sistema. Las primeras suelen ser otros programas del usuario (que contienen rutinas o procesos requeridos por el programa) mientras las segundas forman parte de las llamadas librerías del lenguaje y contienen, entre otros elementos, las llamadas a las funciones básicas del sistema operativo o funciones del propio lenguaje.

Dependiendo de lenguaje, el objeto puede ser almacenado en forma de fichero en el disco o quedarse en memoria a la espera de que se inicie la segunda fase.

ENCUADERNACIÓN O LINKAJE.

En esta fase, se añaden al objeto los módulos objeto requeridos, ya sean de usuario o de las librerías del lenguaje. Así, de un conjunto de programas objeto se obtiene un único ejecutable.

Esta fase del proceso es la que determina en que tipos de procesador y/o sistema operativo será ejecutable el programa ya que es en este punto donde se incorporan las ordenes especificas del sistema para el manejo del hardware y del sistema de archivos.

Este proceso se estudiará detalladamente en el apartado 4.2 del tema Como funciona el ordenador.

EL DEBUGGER

Es un módulo objeto que, linkado junto con el programa, permite hacer un seguimiento paso a paso de la ejecución del programa compilado. Las características y forma de funcionamiento del debugger dependen del lenguaje en si, pero todos presentan determinadas similitudes: parar un programa en un punto determinado, examinar los valores de los datos, análisis de la ejecución (trazo o *trace*).

En resumen, el debugger nos permite ejecutar un programa compilado de forma similar a como lo hace un programa en un lenguaje interpretado.

5.3.3 TRADUCCIÓN MIXTA.

Algunos lenguajes utilizan indistintamente la interpretación y la compilación. De esta forma, se obtienen las ventajas de la interpretación en la fase de desarrollo del programa y las de la compilación en la fase de ejecución. Este es el caso del BASIC, que puede ser compilado, el dBASE, compilable con Clipper, o de algunos lenguajes que presentan un entorno para el desarrollo (Visual Basic) que permiten interpretar el programa o compilarlo, a voluntad del programador.

6. SEGURIDAD.

El nombre genérico del conjunto de herramientas diseñadas para proteger los datos y frustrar a los piratas informáticos (*hackers*) es el de seguridad de ordenadores.

6.1. AMENAZAS A LA SEGURIDAD.

6.1.1 TIPOS DE AMENAZAS.

Los tipos de amenazas a la seguridad de un sistema de ordenadores o una red se caracterizan mejor contemplando la función del sistema como suministrador de información. En general, se produce un flujo de información desde un origen, como un archivo o una región de memoria principal, hacia un destino, como otro archivo o un usuario. Las categorías generales de las amenazas son:

- **Interrupción**: Se destruye un elemento del sistema o se hace inasequible o inútil. Ésta es una amenaza a la disponibilidad (los elementos de un sistema de ordenadores deben estar disponibles a grupos autorizados). Como ejemplos se incluyen la destrucción de una pieza de hardware, como un disco duro, el corte de una línea de comunicaciones o la inutilización del sistema de gestión de ficheros.
- **Intercepción**: Una parte no autorizada consigue acceder a un elemento. Esto es una amenaza a la confidencialidad. La parte no autorizada puede ser una persona, un programa o un ordenador. Como ejemplos se incluyen la interceptación de las conexiones telefónicas para capturar datos de una red y la copia ilícita de archivos o programas.
- **Modificación**: Una parte no autorizada no sólo consigue acceder, sino que falsifica un elemento. Ésta es una amenaza a la integridad. Se pueden citar como ejemplos el cambio de valores en un archivo de datos, la alteración de un programa para que se comporte de manera diferente o la modificación del contenido de los mensajes transmitidos en una red.
- **Invención**: Una parte no autorizada inserta objetos falsos en el sistema. Ésta es también una amenaza de integridad.

6.1.2 AMENAZAS A CADA UNO DE LOS ELEMENTOS DEL SISTEMA.

HARDWARE.

La amenaza principal al hardware se produce en el campo de la disponibilidad. Comprende daños accidentales y deliberados a los equipos, así como el hurto. Hacen falta medida de seguridad físicas para hacer frente a estas amenazas.

SOFTWARE.

Su amenaza principal es la disponibilidad. El software, en especial el de aplicación, es asombrosamente fácil de eliminar. Una gestión cuidadosa de la configuración del software, que incluye la realización de copias de reserva de las versiones más recientes, puede conservar una alta disponibilidad.

Un problema más difícil de afrontar es la modificación del software que provoca que un programa siga funcionando pero se comporte de forma diferente que antes. Los virus informáticos entran dentro de esta categoría y se tratarán posteriormente en este capítulo.

Un problema final es la confidencialidad del software. Aunque se facilitan ciertas contramedidas, por lo general el problema de la copia no autorizada de software no se ha resuelto.

DATOS.

El interés de la seguridad con respecto a los datos es amplio, abarcando la disponibilidad, la confidencialidad y la integridad. En el caso de la disponibilidad, la preocupación es la destrucción de los archivos de datos, lo que puede ocurrir accidentalmente o como consecuencia de una mala intención. Para asegurar la disponibilidad de los datos hay que procurar copiarlos para poder recuperarlos en caso de problema (puede hacerse en disquetes, discos removibles, cintas magnéticas, etc.). Existen varias estrategias de salvaguarda, que ofrecen distintos niveles de seguridad:

- Salvaguarda simple en un único soporte (juego de disquetes, por ejemplo): ofrece una seguridad limitada. En efecto, antes de hacer la salvaguarda hay que borrar el contenido de los disquetes. Si se produjera un problema durante la salvaguarda, se tienen bastantes probabilidades de perder todos los datos.
- La salvaguarda múltiple (en dos juegos de disquetes, por ejemplo) ofrece más seguridad. Si se produjera un problema durante la salvaguarda, el segundo juego estaría intacto (con la condición de que no se empiece por borrar los dos juegos antes de hacer la salvaguarda). Sin embargo, este sistema no protege ante errores lógicos que no suprimen los datos pero los contaminan. Así, si se hubiera borrado por error el contenido de un fichero sin borrar el propio fichero, en la salvaguarda se copiaría un fichero vacío. Cuando quisiéramos darnos cuenta, sería demasiado tarde.
- Para evitar el problema anterior, puede hacerse una salvaguarda los días pares y otra los días impares. El problema del borrado del contenido de un fichero no tendría entonces consecuencias si lo detectamos antes de 24 horas.
- Para aumentar la seguridad, pueden conservarse las salvaguardas durante más tiempo. Por ejemplo, puede hacerse una salvaguarda diferente para cada día de la semana. Se podrán así recuperar los datos en un estado anterior si fuera necesario. Todavía puede aumentarse la seguridad si se conservara la salvaguarda de un día (el viernes, por ejemplo) durante cierto tiempo, con el fin de incrementar las posibilidades de descubrir los problemas antes de que sea demasiado tarde.

La preocupación obvia de la confidencialidad es, si duda, la lectura no autorizada de archivos o bases de datos.

Finalmente, la integridad de los datos es una preocupación fundamental de la mayor parte de instalaciones. Las modificaciones de archivos de datos pueden tener consecuencias poco trascendentes o desastrosas.

REDES Y LÍNEAS DE COMUNICACIONES.

Los sistemas de comunicaciones se utilizan para transmitir datos. Por tanto, las importantes medidas de seguridad aplicadas a los datos (disponibilidad, seguridad e integridad) también se aplican a la seguridad de las redes.

En este contexto las amenazas se clasifican en pasivas (escuchas a escondidas o supervisión de las transmisiones de una organización con la intención de obtener la información que se está transmitiendo) y activas (alteración del flujo de datos o creación de un flujo falso por ejemplo alterando el contenido de los mensajes, inhibiendo el servicio de mensajería o fingiendo ser una entidad diferente suplantando a un usuario autorizado). Las amenazas pasivas son muy difíciles de detectar porque no acarrean alteración alguna de los datos. Hay que hacer frente a estas amenazas mediante la prevención. El objetivo sin embargo con respecto a los ataques activos es detectarlos y recuperarse de cualquier interrupción o retardo causado.

6.2. PRINCIPIOS DE DISEÑO DE LAS MEDIDAS DE SEGURIDAD.

- **Mínimo privilegio**: Todos los programas y usuarios del sistema deben operar utilizando el menor conjunto de privilegios necesarios para completar la labor. Los derechos de acceso deben adquirirse sólo por permiso explícito; por omisión deberían ser "sin acceso".
- **Ahorro de mecanismos**: Los mecanismos de seguridad deben ser tan pequeños y simples como sea posible y estar incluidos en los niveles más bajos del sistema.
- **Aceptación:** Los mecanismos de seguridad no deben interferir excesivamente en el trabajo de los usuarios, mientras cumplen al mismo tiempo las necesidades de aquellos que autorizan el acceso. Si los mecanismos no son fáciles de usar, probablemente no van a ser usados o lo serán de forma incorrecta.
- **Mediación total:** Cada acceso debe ser cotejado con la información de control de acceso. Hay que comprobar la autorización en cada momento.
- **Diseño abierto**: La seguridad del sistema no debe depender de guardar en secreto el diseño de sus mecanismos. La suposición de que los intrusos no conocen el funcionamiento del sistema sólo sirve para que los diseñadores se engañen a sí mismos.

6.2.1 PROTECCIÓN.

PROTECCIÓN DE MEMORIA.

En un entorno de multiprogramación, la protección de la memoria principal es fundamental. El interés no es sólo la seguridad, sino también el funcionamiento correcto de los diversos procesos que estén activos. Si un proceso puede escribir inadvertidamente en el espacio de memoria de otro proceso, este último puede no ejecutarse correctamente.

La separación del espacio de memoria de los diversos procesos se lleva a cabo fácilmente con un esquema de memoria virtual. La segmentación, paginación o la combinación de ambas proporciona un medio eficaz de gestión de la memoria principal.

Si se persigue un aislamiento total, el sistema operativo simplemente debe asegurar que cada segmento o cada página es accesible sólo para el proceso al que está asignada. Esto se lleva a cabo fácilmente exigiendo que no haya entradas duplicadas en las tablas de páginas o segmentos. Si se va a permitir la compartición, el mismo segmento o página puede ser referenciado en más de una tabla. Este tipo de compartición se consigue mejor en un sistema que soporta segmentación o una combinación de segmentación y paginación. En tal caso, la estructura del segmento es visible a la aplicación y la aplicación puede declarar segmentos individuales como compartibles o no compartibles.

CONTROL DE ACCESO ORIENTADO AL USUARIO.

La técnica más habitual de control de acceso al usuario, en un sistema de tiempo compartido o en un servidor, es en la conexión del usuario, que requiere un identificador de usuario (ID) y una contraseña. Este sistema no es un método muy fiable ya que los usuarios pueden olvidar sus contraseñas y pueden revelarlas accidental o deliberadamente. Los piratas informáticos son muy habilidosos en adivinar los ID de usuarios específicos, como el personal de control o de gestión del sistema.

El problema del control de acceso a los usuarios se complica en las redes de comunicaciones. El diálogo de conexión debe tener lugar a través del medio de comunicación y las escuchas se convierten en una amenaza potencial.

El control de acceso al usuario en sistemas distribuidos puede ser centralizado o descentralizado. Con un enfoque centralizado, la red proporciona un servicio de conexión para determinar a quién se le permite usar la red y a qué se le permite conectarse.

El control de acceso descentralizado considera la red como un enlace transparente de comunicaciones y el procedimiento usual de conexión lo lleva a cabo el servidor de destino. En muchas re-

des, pueden emplearse dos niveles de control de acceso. Los servidores individuales pueden estar provistos de un servicio de conexión que proteja las aplicaciones y recursos específicos del servidor.

Además, la red en conjunto puede ofrecer una protección para restringir el acceso a la red a los usuarios no autorizados.

CONTROL DE ACCESO ORIENTADO A LOS DATOS.

Asociado a cada usuario, puede haber un perfil de usuario que especifique las operaciones y los accesos a archivos permisibles. El sistema operativo puede hacer valer unas reglas en función del perfil del usuario. El sistema gestor de la base de datos, sin embargo, debe controlar el acceso a registros específicos o incluso partes de un registro. Por ejemplo, puede permitirse que cualquier administrador obtenga un listado del personal de una compañía, pero solamente unos individuos elegidos pueden tener acceso a la información de salarios.

Un modelo general de control de acceso, ejercido por un sistema gestor de archivos o bases de datos, es el de una matriz de acceso. Los elementos básicos del modelo son los siguientes:

- **Sujeto**: Una entidad capaz de acceder a los objetos (cualquier usuario o aplicación).
- **Objeto**: Cualquier elemento cuyo acceso debe controlarse (archivos, partes de archivos, programas y segmentos de memoria).
- **Derecho de acceso**: La manera en que un sujeto accede a un objeto (leer, escribir, ejecutar).

Una dimensión de la matriz de acceso consta de los sujetos identificados que pueden intentar acceder a los datos (usuarios, grupos de usuarios, aplicaciones, terminales...) y la otra dimensión enumera los objetos accesibles (campos, registros, archivos...). Cada entrada de la matriz indica los derechos de acceso del sujeto al objeto.

En la práctica las matrices de acceso se implementan por descomposiciones en columnas o filas. La descomposición en columnas da lugar a listas de control de acceso. Así pues, para cada objeto, una lista de control de acceso (ACL, *Access Control List*) expresa los usuarios y sus derechos de acceso permitidos. La descomposición por filas da lugar a etiquetas de capacidades (*capabilities*). Una etiqueta de capacidades especifica los objetos y las operaciones autorizadas para un usuario.

6.3. INTRUSOS.

Una de las dos amenazas más conocidas a la seguridad (la otra es la de los virus) es el intruso, conocido en general como pirata (*hacker* o *cracker*). El objetivo de los intrusos es obtener acceso a un sistema o aumentar el conjunto de privilegios accesibles en un sistema. En la mayoría de los casos pretenden obtener información en forma de una contraseña de usuario con la que conectarse al sistema y hacer ejercicio de los privilegios convenidos con el usuario legítimo.

Normalmente, un sistema debe mantener un archivo que asocia una contraseña a cada usuario autorizado. Este archivo debe protegerse de dos maneras: cifrando las contraseñas y limitando el acceso al archivo de contraseñas a una o muy pocas cuentas.

6.3.1 VIRUS.

En 1983, Fred Cohen (estudiante de la Universidad de California) empleó 8 horas en confeccionar un experimento para un seminario sobre seguridad informática. El diseño consistía en un programa que era capaz de modificar a otros, incluyéndose en ellos, y replicarse en el resto de los programas del ordenador. En un tiempo medio de media hora, el "experimento" se adueño de todos los equipos en que fue implantado. Poco después Len Adleman denominó a este tipo de programas **VIRUS**. Existen tres tipos básicos de virus:

- **Gusanos**: es un programa diseñado para reproducirse a través de las redes informáticas. Su misión principal es colapsar el sistema infectado por sobrecarga de sus recursos.

- **Caballos de Troya**: es un programa que modifica o destruye la información mientras simula realizar otra tarea. Suelen presentarse en forma de juegos y no se reproducen.
- **Virus**: combinan el poder reproductivo de los gusanos con el destructivo de los caballos de Troya. Su misión principal es reproducirse lo máximo posible antes de ser detectados o iniciar su fase destructiva.

Ya que la mayoría de los virus existentes son del tercer tipo, vamos a hablar de estos. Hay virus que tienen por misión eliminar otros virus, otros que simplemente muestran un mensaje o acción inofensiva en pantalla y otros que después de hacer su "gracia" se autodestruyen, pero la gran mayoría son más dañinos para el ordenador que el virus de la gripe para el ser humano.

CICLO DE LOS VIRUS.

Todos los virus nacen por obra y gracia de un programador, aunque hay virus que son mutaciones desarrolladas por otros programadores o mutaciones autoprovocadas por el propio virus con el fin de ocultarse. Están diseñados para propagarse y destruir la información de un determinado tipo de hardware ya que, para conseguir sus propósitos, deben manipular éste. Un 90% de los virus está diseñado para PCs.

Se transmiten de un ordenador a otro a través de la copia o uso de ficheros contaminados. Estos ficheros pueden estar en un disquete o ser adquiridos directamente de otros equipos mediante conexiones por red o Internet.

Dentro del mismo ordenador, al ejecutar un programa infectado, el virus se carga en la RAM y pasa a contaminar metódicamente todos los programas que ejecutemos a continuación. Al apagar el ordenador, el virus se pierde, pero volverá a cargarse en memoria en cuanto ejecutemos uno de los programas infectados.

Un virus está en latencia cuando está presente en un ordenador pero no en memoria ya que no puede replicarse. También llamamos latencia al periodo en que los virus se dedican exclusivamente a reproducirse.

Se activan bien sea en una fecha dada, como el Barrotes el 5 de enero, al reproducirse un determinado número de veces o al encender el ordenador un número dado de veces, como el Anti-Telefónica, que se activa al encender el ordenador 300 veces.

TIPOS DE VIRUS.

Virus de fichero.

Utilizan los ficheros ejecutables (`.EXE`, `.COM`, `.SYS`, `.OVL`, …) como medio de transmitirse y tomar el control. Si se instalan en un programa de poco uso, no se carga en RAM hasta que lo ejecutemos. Si es un virus de acción directa (tipo caballo de Troya), ya está hecho el daño.

Por el contrario, si es del tipo de virus residente, lo primero que hace es ver si debe activarse (si es 5 de enero en el caso del Barrotes). Si no es el momento adecuado, se queda en RAM dedicándose a añadirse a todos los programas que ejecutemos, entre ellos el intérprete de comandos (`COMMAND.COM`) que se carga en RAM nada mas encender el ordenador. A partir de aquí, la infección es exponencial.

Un síntoma de infección viral es el aumento del tamaño de los ficheros. Algunos virus de fichero se instalan en el fichero infectado sobreescribiendose, por lo que el programa queda con el mismo tamaño pero inutilizable. Son los llamados virus de sobreescritura.

Los virus de compañía crean un nuevo fichero con el mismo nombre que el que hemos ejecutado pero con la extensión `.COM` (los `.COM` tienen prioridad de ejecución sobre los `.EXE`) en el que instala una copia suya. A continuación le pone el atributo de oculto para que no se vea.

Los virus compresores se dedican a comprimir el fichero infectado. Estos pierden tamaño y, además, es más difícil limpiar el virus ya que éste también queda comprimido.

Un tipo raro de virus son los virus de enlace o de directorio que modifican el directorio cambiando el número del primer cluster del fichero ejecutado por el de la ubicación del virus y situando en un pseudodirectorio la dirección real. No se crean nuevas copias del virus, sino que hace que todos los ficheros infectados pasen primero por la única copia del virus y posteriormente, a través del pseudodirectorio, ejecuta el fichero.

Virus de BOOT.

Se instalan en el BOOT y la tabla de particiones de los discos. Se contagia al arrancar con un disquete contaminado (aunque no sea de arranque). Una vez instalado en el H.D., se carga en RAM nada mas encender el ordenador y se dedica a contagiar todos los disquetes que usemos. Son altamente dañinos pues al activarse suelen destruir la FAT.

Virus de macro.

Son los más abundantes hoy en día. No infectan programas sino ficheros de datos de OFFICE (suite de Microsoft) que, al contener macros, es como si contuviesen ficheros ejecutables. Dado que con el Word de Office se pueden crear paginas WEB de Internet y se suelen usar en el correo electrónico (además, el gestor de correo más popular, Outlook, también es de esta suite), su difusión se ha ampliado mucho en poco tiempo.

CAMUFLAJE DE LOS VIRUS.

Con el fin de dificultar su localización a los antivirus, los virus emplean algunos trucos como el ocultamiento (*stealth*) que consiste en suministrar al antivirus una copia del fichero que quiere examinar que no esté infectada (sólo si el virus está en RAM). Con el *tunneling* o sobrepasamiento intentan evitar que un antivirus residente les localice. El autoencriptamiento hace que cada copia del virus tenga un aspecto diferente con el fin de evitar su localización (pero tienen una rutina de encriptación común). El polimorfismo es un autoencriptamiento que cambia periódicamente la rutina de encriptación. El *armouring* incluye en el virus rutinas para evitar que pueda ser examinado de cara a la obtención de un antivirus.

6.3.2 LOS ANTIVIRUS.

Son programas que examinan la RAM, el sector de arranque y los ficheros susceptibles de ser infectados en busca de virus. Para ello utilizan métodos como búsqueda de determinadas cadenas de datos (el Viernes 13-B tiene la cadena `WARNING! THIS PROGRAM IS INFECTED WITH VIRUS-B`), búsquedas deductivas (examen del tamaño, fecha y hora de los ficheros) y la búsqueda heurística (anomalías o “cosas raras” en los ficheros) que, además, nos permite localizar virus nuevos (a cambio de un montón de falsas alarmas).

Salvo con la búsqueda heurística, sólo son capaces de detectar y eliminar (no siempre) los virus ya conocidos. Teniendo en cuenta que cada mes aparecen cientos de virus nuevos, es conveniente tener un antivirus lo mas actualizado posible (uno de hace 3 meses se considera obsoleto).

LAS VACUNAS.

Es un método por el que se toman datos de los ficheros que tenemos con el fin de poder determinar en un futuro si un fichero está infectado. El antivirus examina los datos actuales del fichero con los que contiene la vacuna. Si hay anomalías, procede a realizar un examen a fondo en búsqueda de virus.

ANTIVIRUS RESIDENTES O CENTINELAS.

Se cargan en memoria nada mas encender el ordenador. Examinan cualquier acceso a un disquete o a un fichero no vacunado. Si detectan un virus, bloquean el sistema o pasan a la ejecución del antivirus.

Presentan como inconvenientes la ocupación de RAM y el ralentizamiento del sistema. Además, previene contra los virus conocidos pero no contra los nuevos. Aun así, es conveniente tener uno instalado, sobre todo en los equipos que son manejados por varios usuarios o los conectados a redes o Internet.

LOS VIRUS E INTERNET.

El "contagio" de un virus de un ordenador a otro se puede dar por múltiples motivos: porque alguien le ha pasado un pendrive o CD con un fichero contaminado o bien porque accedió a algún lugar de Internet donde existía un fichero o recibió un e-mail contaminado. El mecanismo es exactamente igual: un fichero contaminado llega a nuestra máquina. Si no ejecutamos el programa no pasará nada, si lo ejecutamos o abrimos, nos contaminaremos.

En Internet existen muchos mecanismos para poder hacernos con un fichero contaminado, ya que existen múltiples formas de transmitirlo. El más inmediato es que alguien nos lo envíe junto con un mensaje de correo electrónico. Hay que tener cuidado con los ficheros que puedan venir adjuntados al mensaje. Sólo puede haber problema con los ficheros que tengan capacidad de ejecución, por lo que, en teoría, podemos abrir cualquier fichero que no tenga partes ejecutables y, por lo tanto, ver una foto si ésta viene en cualquiera de sus formatos estándar (`BMP`, `JPEG`, `GIF`, etc.), un vídeo (`.AVI`, `.MOV`, `.MPG`, etc.), un fichero de texto (`.TXT`). También hay que tener cuidado con los documentos de Office pues pueden contener macros vulnerables ante los virus. Este es el caso del famoso *Melissa* que era una macro dentro de un documento Word.

Como medida de protección es aconsejable no abrir ningún fichero procedente de ningún lugar que pueda poseer una parte activa, como los documentos Word (`.DOC`), las hojas de cálculo Excel (`.XLS`), los dibujos de CorelDRAW (`.DRW`) y, por supuesto los ejecutables directamente (`.EXE`, `.COM`).

También es fundamental no abrir ningún correo cuyo remitente nos sea desconocido.

Es cierto que, dada la diversidad de formas de transferir ficheros en Internet, los métodos de contaminación aumentan proporcionalmente. Hemos visto el peligro que puede existir con el correo electrónico pero, como no es la única vía para transferir un fichero desde Internet, tampoco es la única forma de posible contagio.

Una forma usual de quedar contaminado con un virus es seguir un enlace en el navegador que nos lleva a la descarga de un fichero. Si éste está contaminado, podremos ser infectados. Evidentemente ese método no difiere en nada con respecto a la copia de un fichero desde un disquete, un CD-ROM o una unidad de red.

Una forma de contagio sofisticada y nueva consiste en hacer uso de los *scripts* que se añaden a las páginas web, que no es más que un fichero HTML que originariamente sólo contenían texto pero que en las últimas versiones se puede incluir texto ejecutable, es decir, *scripts* en Visual Basic Script, JavaScript o Jscript. Al llevar una parte ejecutable, volvemos a correr el riesgo de poder ser contaminados. El funcionamiento de este tipo de virus es algo complejo, aunque se basan en el mismo esquema que el resto de los virus. Básicamente existe una función que se autoejecuta cuando el navegador abre la página. A partir de ese momento se puede llegar a obtener un virus.

Por supuesto, no hay que descartar mecanismos como ICQ, IRC, FCT, etc., ya que todos ellos pueden efectuar cargas de ficheros y, además, en algunos casos los propios programas pueden almacenar *scripts* de automatización, aumentando así las posibilidades de obtener virus.

Los fanáticos del mundo de los virus no paran en su carrera por obtener cada día virus más difícilmente localizables y con mayor poder destructor. Incluso se dice que las guerras de este siglo que

viene podrían consistir en bombardeos electrónicos. Se prevé que en unos años absolutamente todos los dispositivos estén conectados a la Red, desde el ordenador del trabajo hasta el reloj de la muñeca, pasando por sistema de control de la vivienda, el coche y cualquier otra cosa que podamos imaginar. Así que esta posibilidad, que ahora parece absurda, quizá en un futuro pueda hacer más daño de lo que parece.

En este sentido se está trabajando y ya se habla de bombas "personalizadas". Éstas son virus pero que, en lugar de activarse en cualquier ordenador bajo algunas condiciones, sólo se activan en el equipo para el que han sido diseñadas. Si se programa de algún modo un virus que se expanda por la Red buscando el ordenador de "destino" y únicamente estalle en el que tiene el número para el que fue programado, tendríamos un virus personalizado, algo parecido a los misiles pero, en este caso, con formato electrónico.

DIRECCIONES ÚTILES.

Seguridad en la Red (http://www.seguridadenlared.es/)

WEB Instituto Nacional de Ciberseguridad (https://www.incibe.es/)

WEB Internet segura 4 kids (https://www.is4k.es/)

WEB Oficina de seguridad del internauta (https://www.osi.es/es)

Investiga 5:

1. Define malware y, al menos, tres tipos de malware.
2. Define crimeware y, al menos, tres tipos de crimeware.
3. Define fraude-on-line y, al menos, cuatro modalidades de fraude-on-line.

6.4. CONTRASEÑAS SEGURAS.

Por muy seguro que sea un sistema, no servirá de nada si un atacante consigue el nombre ycontraseña de un usuario legítimo.

Actualmente, el método más extendido para obtener acceso a información personal que hemos almacenado en nuestro equipo y/o servicios en línea es mediante contraseñas.

La mayoría de las veces una contraseña es la única barrera entre nuestros datos confidenciales y los ciberdelincuentes. Por lo que merece la pena invertir un poco de tiempo y esfuerzo para gestionarlas eficazmente.

¿QUÉ DEBE TENER UNA CONTRASEÑA PARA SER REALMENTE SEGURA?

Una buena contraseña debe cumplir, al menos, tres de estas cuatro características:

- Tener números.
- Tener letras mayúsculas y minúsculas.
- Tener símbolos ($, @, &, #, etc.)
- La longitud no debe ser inferior a siete caracteres. A mayor longitud más difícil de adivinar.
- No debe formarse con números y/o letras que estén adyacentes en el teclado. Ejemplos de malas contraseñas son: 123456, 1q2w3e o 123QWEasd.
- La contraseña no debe contener información que sea fácil de averiguar, por ejemplo, nombre de usuario de la cuenta, información personal (cumpleaños, nombres de hijos, etc.)
- No debe contener palabras existentes en algún idioma. Los ataques de diccionario prueban cada una de las palabras que figuran en el diccionario y/o palabras de uso común.

BUENAS PRÁCTICAS.

No uses la misma contraseña para diferentes cuentas. Sobre todo si son de alto riesgo, como las de los servicios bancarios o comerciales.

La contraseña es algo privado, no la dejes escrita en ningún sitio, y mucho menos al lado del ordenador.

Cambia las contraseñas que traen por defecto los dispositivos y servicios en línea. Un ejemplo es el de los router WiFi, que traen por defecto contraseñas públicamente conocidas, que un atacantepodría utilizar.

Limita el uso de las contraseñas almacenadas en el navegador para los servicios críticos. Si es posible el mejor sitio es la memoria de uno mismo.

Cambiar las contraseñas con cierta frecuencia, sobre todo si se usan en mas de un ordenador.

TRUCOS PARA CREAR CONTRASEÑAS SEGURAS.

Usar una frase fácil de memorizar. Una vez hecho esto, podemos hacer combinaciones con las distintas palabras que componen la frase: utilizar la primera letra de cada palabra, utilizar la última letra de cada palabra, etc.

Ejemplo: Utilizar la primera letra de cada palabra, en mayusculasn los verbos y sustantivos, sustituir/añadir cuando sea posible por símbolos especiales/números.

Frase: En un lugar de la Mancha de cuyo nombre no quiero acordarme.

Contraseña: e1LdlMdcNnQ?

Usar una «semilla» y aplicarle un «algoritmo»: En cada lugar donde debamos crear una contraseña, pensamos en una «semilla», que no es más que una palabra que ayude a recordar ese lugar. A la semilla se le aplica un «algoritmo» que es una combinación de pasos que utilizaremos para crear las contraseñas de cualquier sitio. La ventaja de utilizar este método es que sólo será necesario recordar el algoritmo.

Algoritmo: Quitarle las tres primeras letras, poner en mayúsculas la primera letra y la última, añadir al principio el número 92 (año del cumpleaños de mi hermano), añadir el final los símbolos *=.

Ejemplo: Recordar contraseña de Instagram y Facebook

Contraseña: 92AgraM*=, 92EbooK*=

APLICACIONES QUE NOS PUEDEN AYUDAR.

Comprobador de contraseñas:

Cuando no estés seguro de si la contraseña que has elegido es lo suficientemente segura, puedes utilizarun medidor de fortaleza de la contraseña:

https://password.kaspersky.com/es/.

Gestores de contraseñas:

Cuando manejamos muchas cuentas se vuelve complicado recordar la contraseña asociada a cada una de ellas. Lo peor que podemos hacer en ese caso es optar por utilizar la misma contraseña para todos los sitios, ya que si se descubre la contraseña de acceso a alguna de estas cuentas, un atacante podrá fácilmente acceder al resto de ellas. Para solucionar este problema, existen los gestores de contraseñas que son programas que se utiliza para almacenar contraseñas. Nos permiten recordar todas las contraseñas, claves de acceso y nombres de usuario que necesitamos para acceder a una cuenta o página de Internet. La información se almacena cifrada y sólo se puede acceder a travésde una clave que, por supuesto, deberárá ser segura.

7. MATERIAL RECOMENDADO

Videos del Profesor Alberto Prieto Espinosa (http://atc.ugr.es/APrieto_videoclases)

TEMA 1

L1.1 Terminología y conceptos básicos de Informática. (15:41)

L1 3 Tipos de computadores.(15:50)

TEMA 5.

L5.1 Tipos de lenguajes y estilos de programación. (25:22)

L5.2 Los procesos de traducción y ejecución de programas. (15:23)