Curso: (30227) Seguridad Informática

Fernando Tricas García

Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza
http://webdiis.unizar.es/~ftricas/
http://moodle.unizar.es/
ftricas@unizar.es

Tema Condiciones de carrera

Fernando Tricas García

Departamento de Informática e Ingeniería de Sistemas
Universidad de Zaragoza
http://webdiis.unizar.es/~ftricas/
http://moodle.unizar.es/
ftricas@unizar.es

Condiciones de carrera

- Una fuente (muy) común de errores
- Sólo son posibles en sistemas concurrentes, donde hay procesos que pueden interactuar
  - Pero casi todo es concurrente hoy en día
- Difíciles de encontrar (sobre todo si no se buscan)
- Difíciles de corregir

Condición de carrera

- Cuando una suposición debe ser cierta durante un cierto periodo de tiempo, pero puede no serlo.
  - Ventana de vulnerabilidad: si se viola la suposición, se obtinene un comporamiento incorrecto
- Hablando de programas, las ventanas pueden ser grandes, pero con frecuencia son muy pequeñas

Condición de carrera: ejemplo en java

```
import          java.io.*;
import          javax.servlet.*;
import          javax.servlet.http.*;

public class Counter extends HttpServlet {

  int             count = 0;

  public void     doGet(HttpServletRequest in,
                    HttpServletResponse out)
  throws          ServletException, IOException {

    out.setContentType("text/plain");
    Printwriter     p = out.getWriter();
                    count++;
                    p.println(count + " hits so far!");
  }
}
```

Condición de carrera

Usuario 1	Usuario 2	count
Solicita el 'servlet'	Solicita el 'servlet'	

Condición de carrera

Usuario 1	Usuario 2	count
Solicita el 'servlet'	Solicita el 'servlet'	
incrementa `count`		1

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

Condición de carrera

Usuario 1	Usuario 2	count
Solicita el 'servlet'	Solicita el 'servlet'	
incrementa `count`		1
	incrementa `count`	2

Condición de carrera

Usuario 1	Usuario 2	count
Solicita el 'servlet'	Solicita el 'servlet'	
incrementa `count`		1
	incrementa `count`	2
print 2		2

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

Condición de carrera

Usuario 1	Usuario 2	count
Solicita el 'servlet'	Solicita el 'servlet'	
incrementa `count`		1
	incrementa `count`	2
print 2		2
	print 2	2

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

Condición de carrera

Usuario 1	Usuario 2	count
Solicita el 'servlet'	Solicita el 'servlet'	
incrementa `count`		1
	incrementa `count`	2
print 2		2
	print 2	2

Incluso moviendo la actualización, no estaríamos seguros de hacerlo bien:

```
p. println (++count + " hits so far !");
```

Condición de carrera: no sólo mala suerte

- Si el atacante tiene acceso a suficientes recursos, puede mejorar sus posibilidades de éxito
- Sólo hace falta tener éxito una vez
- Automatizar las peticiones y sentarse a esperar
  ... si hay una posibilidad entre un millón, no tardará mucho en conseguirlo

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

Condición de carrera: soluciones

- Cerrar las ventanas: asegurarse de que las condiciones se cumplen tanto tiempo como sea necesario
- Hacer el código relevante atómico con respecto a los datos necesarios
  - Atómico $\longrightarrow$ se ejecuta como si fuera una sola operación, sin interrupción
- Se utilizan bloqueos, que la mayoría de los lenguajes tienen

Condición de carrera: El ejemplo

```
import   java.io.*; import   javax.servlet.*;
import   javax.servlet.http.*;

public class Counter extends HttpServlet {
  int   count = 0;
  public synchronized void
     doGet(HttpServletRequest in,
                HttpServletResponse out)
  throws   ServletException, IOException {
    out.setContentType("text/plain");
    Printwriter      p = out.getWriter();
       count++;
       p.println(count + " hits so far!");
  }
}
```

Condición de carrera: nada es perfecto

- Sólo un hilo cada vez
- Si se usa mucho, puede convertirse en un cuello de botella
- Solución:
  Mantener el código que ha de ser atómico (*sección crítica*) tan pequeño como sea posible

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

Condición de carrera: El ejemplo mejor

```
import              java.io.*;
import              javax.servlet.*;
import              javax.servlet.http.*;
public class Counter extends HttpServlet {
  int                 count = 0;
  public void
    doGet(HttpServletRequest in, HttpServletResponse out)
  throws              ServletException, IOException {
    int                 my_count;
                        out.setContentType("text/plain");
    Printwriter         p = out.getWriter();
                        synchronized(this) {
      my_count = ++count;
    }
            p.println(my_count + " hits so far!");
  }
}
```

println

fuera del bloque!

Condiciones de carrera: más posibilidades

- También ocurren en sistemas con múltiples procesos que comparten datos
- Como mínimo, ficheros
- Principalmente en Unix
- De todas formas, la API de Windows hace que sea mucho más difícil

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

Otro ejemplo

El atacante	El programa
Crea	

`/tmp/unFichero`

Otro ejemplo

El atacante | El programa

Crea

`/tmp/unFichero`

Comprueba antes de borrar:

access("/tmp/unFichero");

Éxito (todo está correcto)

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

Lo que puede pasar. Ejemplo

El atacante | El programa

Borra

`/tmp/unFichero`

Lo que puede pasar. Ejemplo

El atacante | El programa

Borra

`/tmp/unFichero`

Crea un enlace

`/tmp/unFichero` $\rightarrow$ `/etc/shadow`

Lo que puede pasar. Ejemplo

El atacante | El programa

Borra

`/tmp/unFichero`

Crea un enlace

`/tmp/unFichero` $\rightarrow$ `/etc/shadow`

Abre:

open("/tmp/unFichero")

Éxito

Lo que puede pasar. Ejemplo

El atacante | El programa

Borra

`/tmp/unFichero`

Crea un enlace

`/tmp/unFichero` → `/etc/shadow`

Abre:

open("/tmp/unFichero")

Éxito

Pero está usando ...

`/etc/shadow`

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

Lo que puede pasar

- Crear un recurso temporal con permisos débiles
- Crear un recurso temporal en un directorio con permisos débiles
- Crear un recurso en un directorio que creó un atacante
- El propio recurso fue creado por un atacante
- Se crea el recurso y se modifican los permisos después
- Se crea el recurso en un directorio y luego se mueve a otro
- Los permisos del proceso se modifican temporalmente

Condiciones de carrera: esquema común

1. Comprobación de cierta propiedad de un fichero
2. La comprobación necesita mantenerse válida hasta que se accede al fichero
   Fallos **tiempo de comprobación, tiempo de uso**
   TOCTOU $\longrightarrow$ *'time–of–check, time–of–use flaws'*

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

Condición de carrera: ejemplo

```
if(!access(file, W_OK)) {

    f = fopen(file, "wb+");

    write_to_file(f);

...
```

access comprueba si UID tiene permiso para escribir y devuelve 0 en caso positivo
El problema aparece cuando alguien cambia el fichero entre la comprobación y la apertura

Condición de carrera: evitando TOCTOU

Consejos:

- **Evitar** llamadas al sistema que utilizan **nombres de fichero**, en lugar de sus 'identificadores'
  `fstat()` en lugar de `stat()`
- En lugar de usar `access()`, cambiar el EUID y el EGID al usuario adecuado (abandonando los privilegios que se tengan con `setgroups(0,0)`)

Condición de carrera: evitando TOCTOU

Comprobación de propiedades

- `lstat()`, guardar los datos
- `open()`
- `fstat()`, guardar los datos
- Comparar `st_mode, st_ino, st_dev`

Creación

Si el fichero no existe lstat lo dice, pero alguien puede crear uno antes del open (usar `O_CREAT` y `O_EXCL`).

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

Ejemplo ...

```
if (lstat(fname, &stb1) >= 0 && S_ISREG(stb1.st_mode)) {
        fd = open(fname, O_RDWR);
        if (fd < 0 || fstat(fd, &stb2) < 0
          || ino_or_dev_mismatch(&stb1, &stb2))
            raise_big_stink()
    } else {
      fd = open(fname, O_RDWR | O_CREAT | O_EXCL, FMODE);
      if (fd < 0)
        raise_big_stink();

        }
```

http://seclists.org/bugtraq/2000/Jan/0016.html

Acceso seguro a ficheros

- No siempre es posible acceder a los ficheros por su puntero -manejador, identificador- (`link`, `mkdir`, `mknod`, `rmdir`, `symlink`, `unmount`, `unlink`, `utime`)
- ¿Y entonces?
- Almacenar los ficheros en su propio directorio, que sólo sea accesible para la UID del programa que hace las operaciones.

- Asegurarnos de que un posible atacante no tiene acceso a los directorios superiores.
  - Se crea el directorio y se entra `chdir()`
  - Luego, ir subiendo por el árbol de directorios, asegurándonos de que sólo root y el usuario pueden modificar (comprobar UID y GID) en cada paso.

Acceso seguro a ficheros

- Evitar los nombres predecibles. Algunos consejos
  1. Utilizar un prefijo `miPrograma.`
  2. Generar al menos 64 bits aleatorios, codificar en *base64*, reemplazar las / y concatenar.
  3. Poner una máscara adecuada `umask`, 0066
  4. Crear el fichero con `fopen()`
  5. Si el fichero no está en un disco montado por red, borrarlo (`unlink()`)
  6. Trabajar
  7. Cerrar el fichero

  Nunca cerrar y reabrir en directorios donde pueda haber 'carreras'

Borrado seguro de ficheros

- Sin usar un directorio seguro, no se puede hacer de forma segura
- Sólo se puede con `unlink()`!!, que usa un nombre.

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

Borrado seguro de ficheros

- Sin usar un directorio seguro, no se puede hacer de forma segura
- Sólo se puede con `unlink()`!!, que usa un nombre.

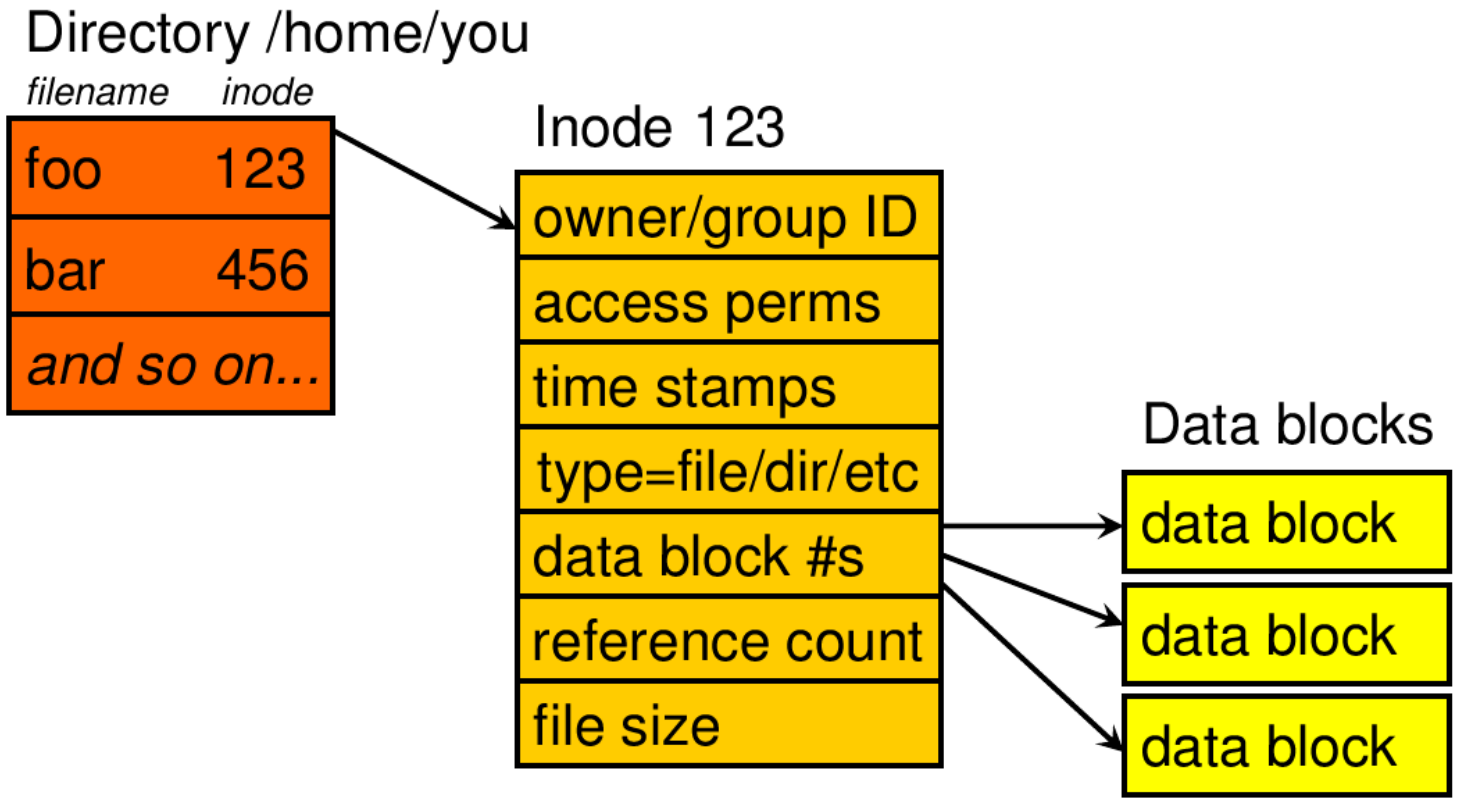
- Evitar condiciones de carrera no es lo único, en este caso

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

Borrado de ficheros (los datos; 'excursión')

¿Qué es borrar? (I)

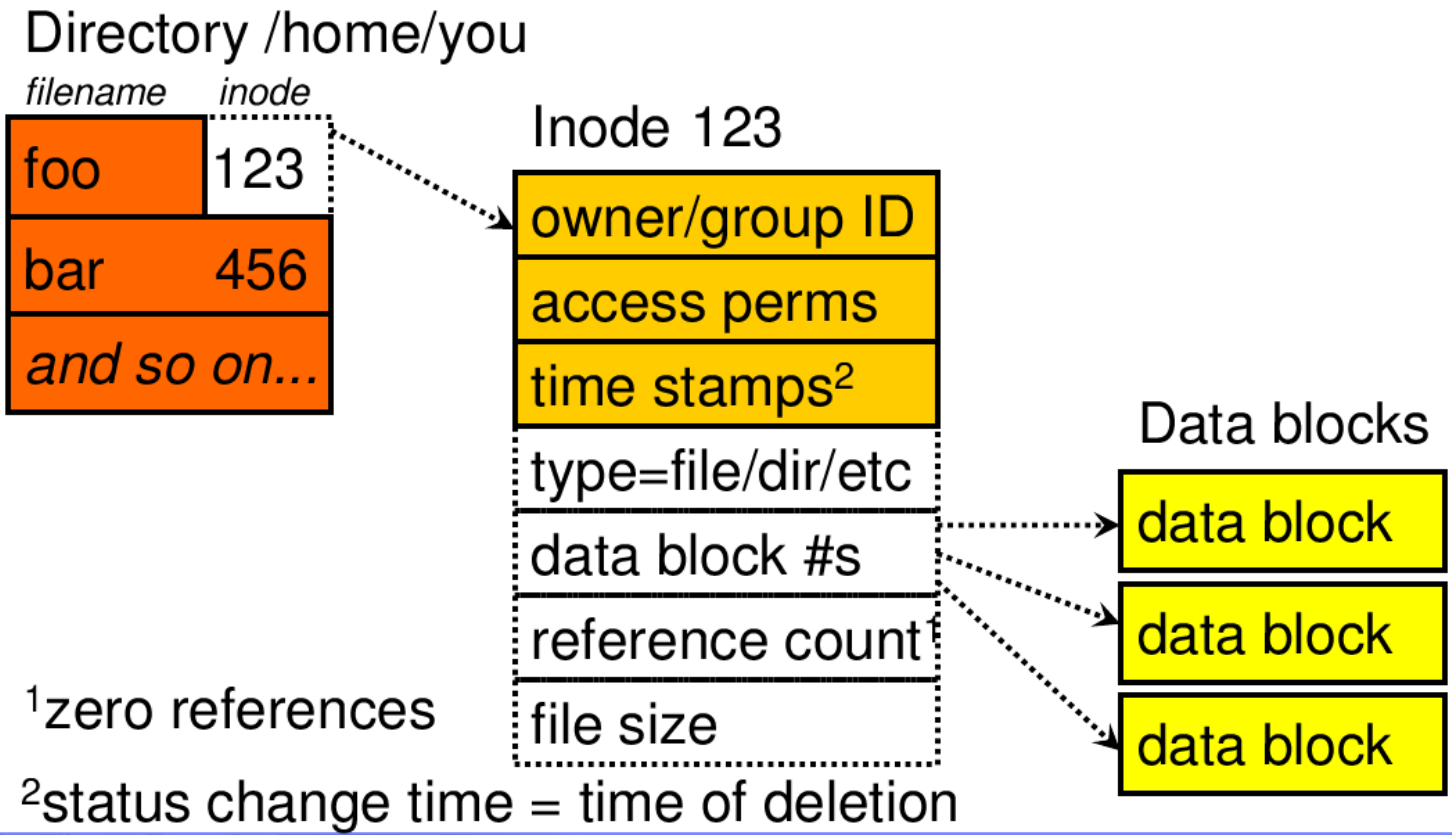
UNIX file system architecture



Borrado de ficheros (los datos; 'excursión')

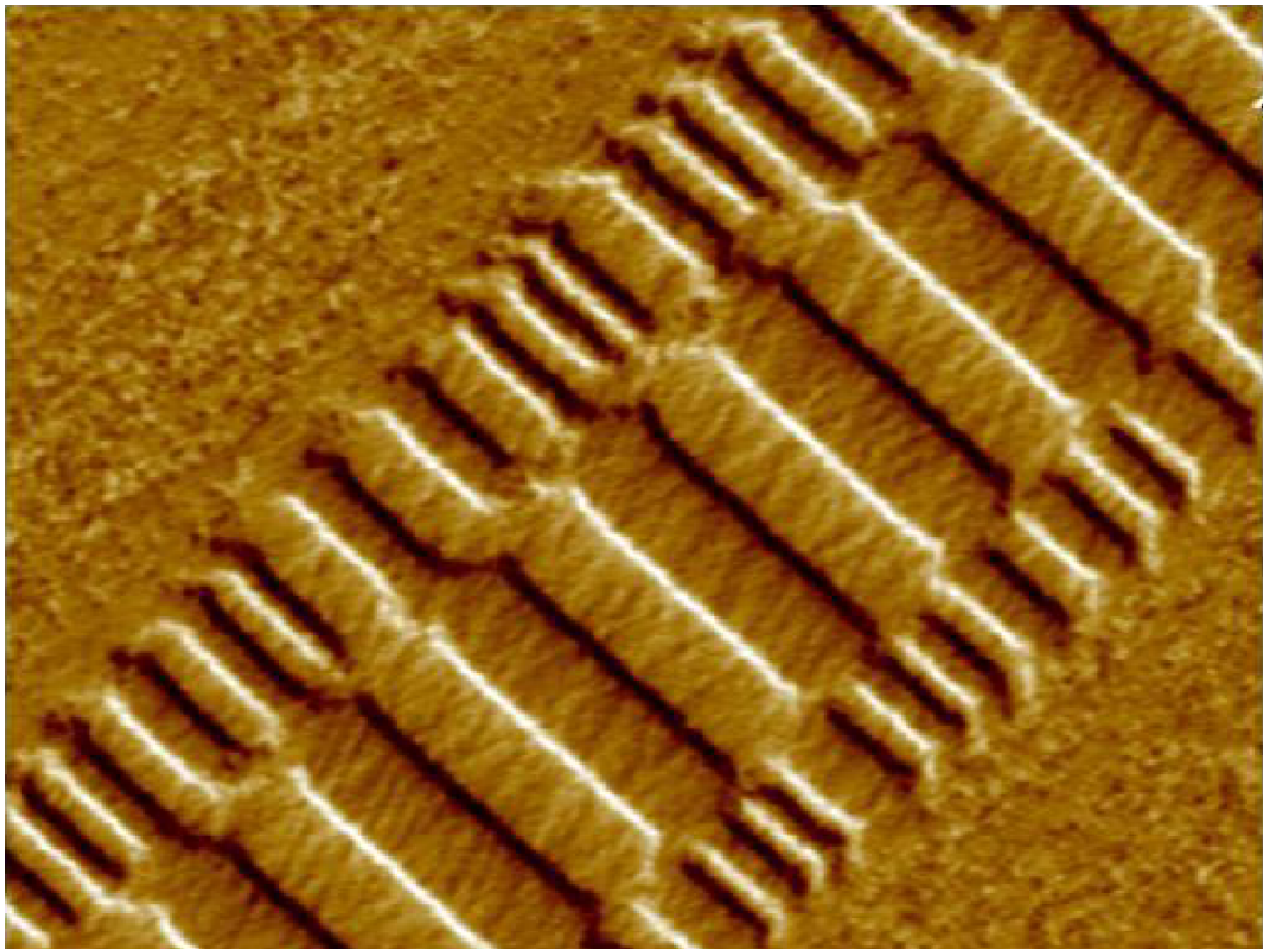
¿Qué es borrar? (II)

Deleting a UNIX file destroys structure, not content



Borrado de ficheros (los datos; 'excursión')

¿Qué es borrar? (III)



'The broken file shredder. Programming traps and pitfalls'

Wietse Venema. IBM T.J. Watson Research Center.

Borrado de ficheros (los datos; 'excursión')

- ¿Qué es borrar? Habitualmente, en los sistemas operativos, dejar inaccesible
- Incluso sobre-escritos pueden recuperarse, con las herramientas adecuadas
- Sobreescribir varias veces. (hasta 7!)
  1. Con unos
  2. Con ceros
  3. Con ceros y unos alternando
  4. 4 veces con basura (más o menos) aleatoria
     Podría no ser suficiente

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

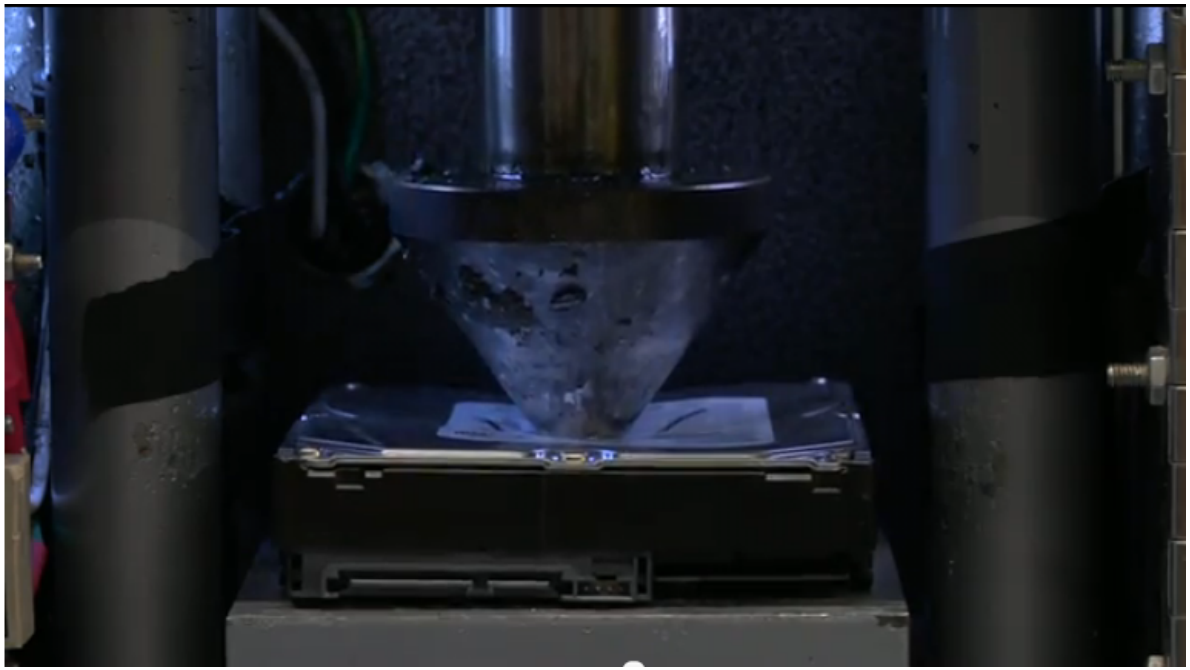
Borrado de ficheros (los datos; 'excursión')

Pero ...

- ¿Habremos escrito donde queríamos?
- ¿Habremos escrito todas las veces? (cachés, ...)
- ¿Habremos escrito en todos los sitios que debíamos? (copias temporales, ...)
- ¿Y el sistema operativo no habrá hecho nada en medio?
- ...

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

Vídeo ejemplo: "Google data center security"



http://www.google.com/about/datacenters/inside/data-security/

Mejor no escribir

- Cuando los datos sean importantes, lo mejor es que nunca se escriban en el disco
- Si hay que escribirlos, mejor cifrados
- Descifrarlos directamente en memoria, asegurándonos de que no son almacenados temporalmente (*swap*), con `mlock()`, o con un *ramdisk, sistemas de virtualización, ...*.
- Por supuesto, cuidado con la clave

Sobre los nombres de los ficheros temporales

Cuidado con ...

- `mktemp()` (dan un nombre, puede ser predecible, qué pasa si nos lo cambian?)
- `tmpnam()` y `tempnam()` (parecido)

Mejor...

- `mkstemp()` (da un nombre según la plantilla y devuelve el descriptor del fichero ya abierto)
  - int mkstemp(char *template);
- `tmpfile()` (binario) y `mkdtemp()` (directorio)
  - FILE *tmpfile(void);
  - char *mkdtemp(char *template);

Ficheros temporales

```c
#include <stdio.h>

int main() {
  FILE *fh = tmpfile();
  /* file is automatically deleted when program exits */
  /* do stuff with stream "fh" */
  fclose(fh);

  /* The C standard library also has a tmpnam()
   function to create a file for you to open
   later. But you should not use it because
   someone else might be able to open the file
   from the time it is created by this function
   to the time you open it. */

  return 0;
}
```

Ficheros temporales

```
import java.io.File;

try {
    // Create temp file
    File filename =
        File.createTempFile("prefix", ".suffix");

    // Delete temp file when program exits
    filename.deleteOnExit();

    System.out.println(filename);

} catch (IOException e) {
}
```

Bloqueo de ficheros

- Otra forma de evitar condiciones de carrera
- En algunos sistemas no es más que un indicativo, mejor asegurarse de que están en directorios no accesibles para los atacantes
- `open()` con `O_EXCL`.
  No funciona para sistemas montados con NFS (versiones antiguas)

  Además NFS tenía fama de inseguro

Simularlo

- Puede simularse, con otro fichero
  1. Crear un fichero con `open`, que contenga hostname y pid.
  2. Usar `link()` para enlazar al fichero de bloqueo
  3. Si devuelve 0, hemos tenido éxito
  4. Si no, comprobar con `stat()` si el número de enlaces pasó a ser 2, en cuyo caso todo fue bien.

Bloqueos en C

- Sistema V `lockf`
- BSD `flock`
- POSIX `fcntl`
- En muchos Unix `flock`

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

Bloqueos en C (ejemplo)

```c
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
int main(int argc, char *argv[]) {
    int fd; struct flock fl;
    fd = PrivoxyWindowOpen("testfile", O_RDWR);
    if (fd == -1) /* Handle error */;
    /* Make a non-blocking request to place a write lock on bytes
     * 100-109 of testfile */
    fl.l_type = F_WRLCK; fl.l_whence = SEEK_SET;
    fl.l_start = 100; fl.l_len = 10;
    if (fcntl(fd, F_SETLK, &fl) == -1) {
        if (errno == EACCES || errno == EAGAIN) {
            printf("Already locked by another process\n");
            /* We can't get the lock at the moment */
        } else {
            /* Handle unexpected error */;
        }
    } else { /* Lock was granted... */
        /* Perform I/O on bytes 100 to 109 of file */
        /* Unlock the locked bytes */

        fl.l_type = F_UNLCK;
        fl.l_whence = SEEK_SET;
        fl.l_start = 100;
        fl.l_len = 10;
        if (fcntl(fd, F_SETLK, &fl) == -1)
            /* Handle error */;
    }
    exit(EXIT_SUCCESS);
} /* main */
```

Bloqueos en Perl, PHP

- `flock`
- Igual que el `flock` del sistema
  - `LOCK_SH` `LOCK_EX`, `LOCK_UN`

En Perl

open(F, '> file.txt')
flock(F, LOCK_EX)

En PHP

$fp=fopen('file.txt', 'w')
flock($fp, LOCK_EX)

Departamento de Informática e Ingeniería de Sistemas
Universidad Zaragoza

Otras condiciones de carrera

- No sólo ocurren en el acceso a ficheros
- Bases de datos replicadas, por ejemplo
- Java 2 y el cambio de políticas 'al vuelo'

En la web, nuestros programas siempre son concurrentes!