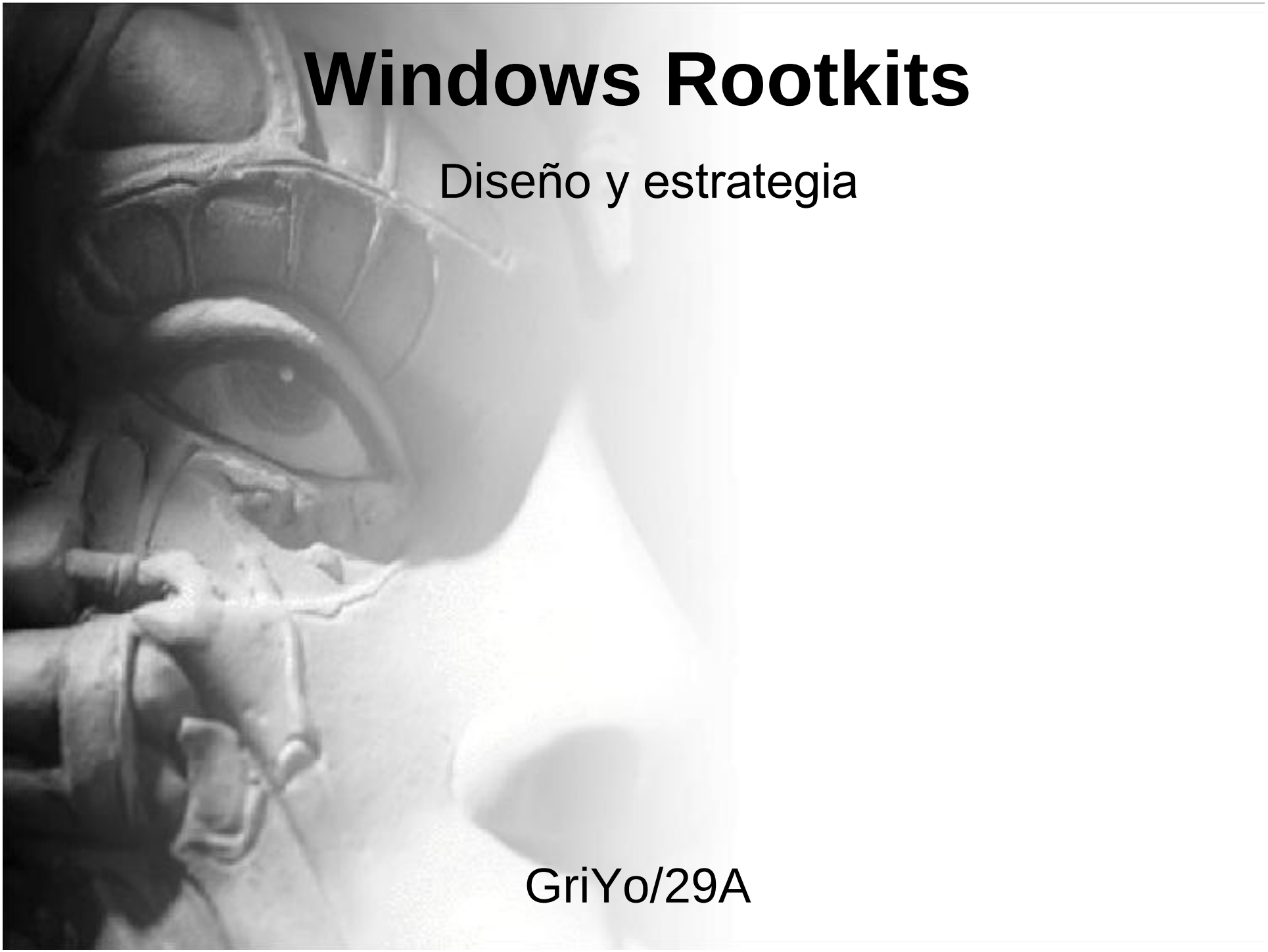
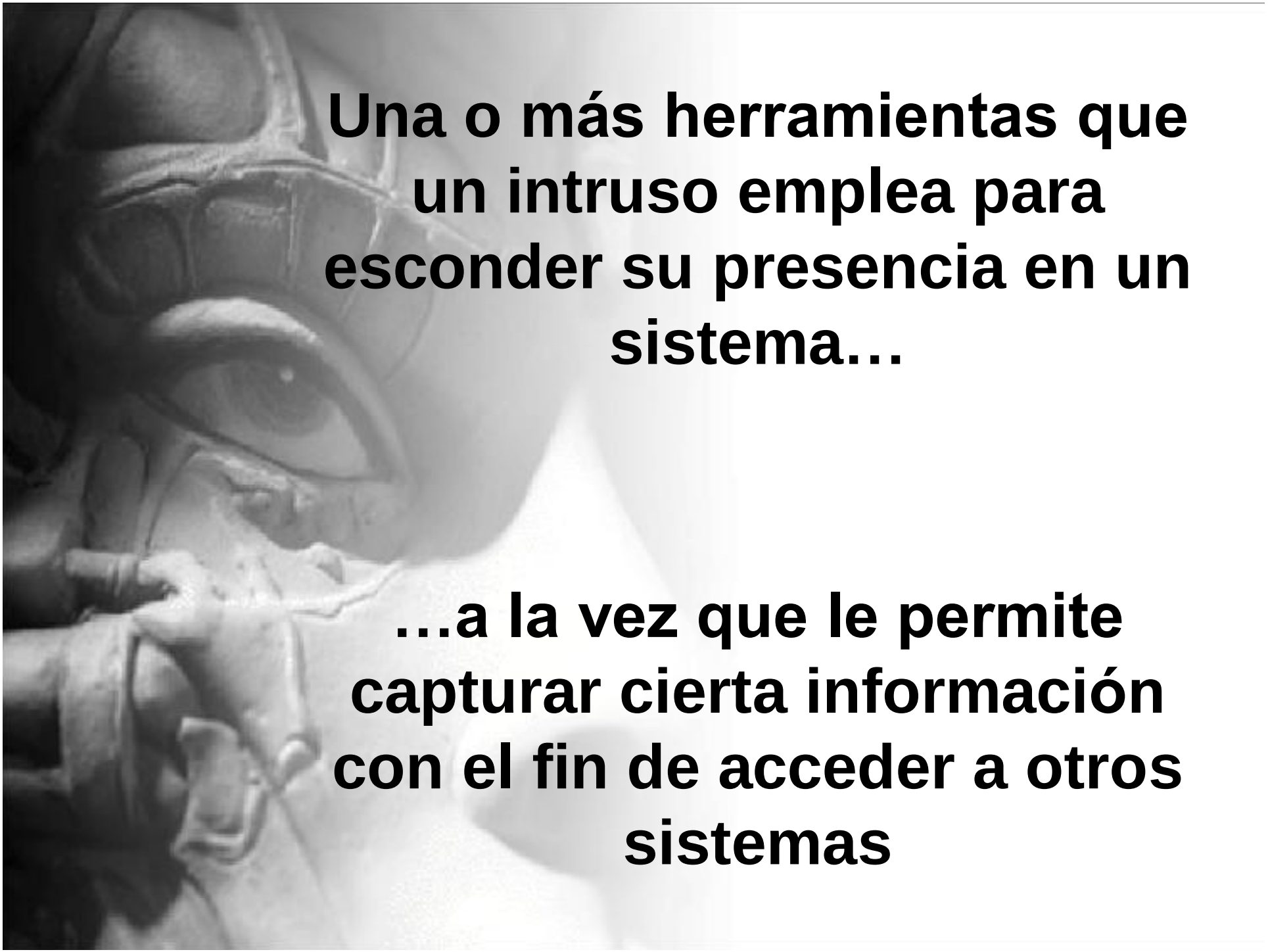


Windows Rootkits

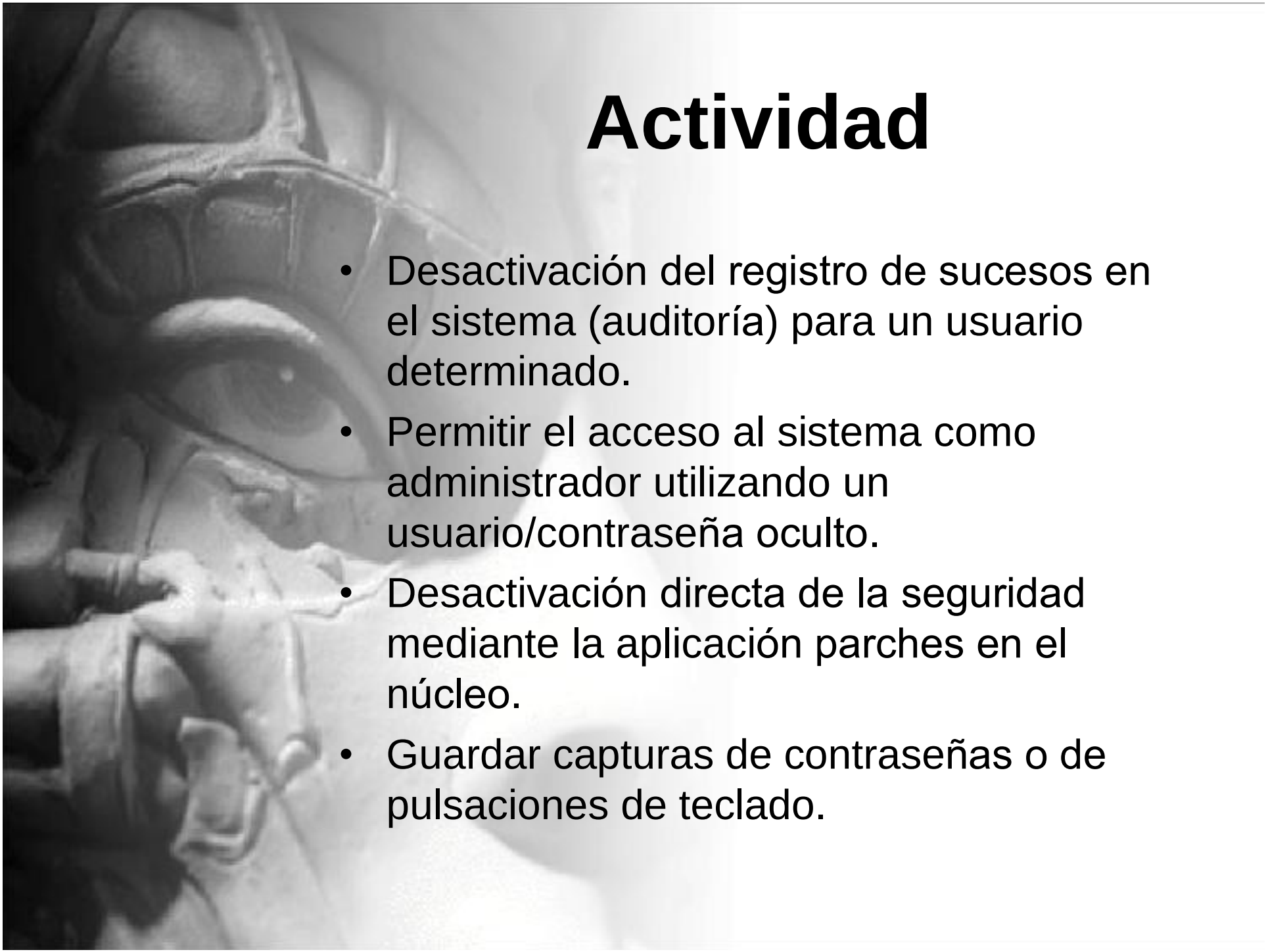
Diseño y estrategia

GriYo/29A



Una o más herramientas que un intruso emplea para esconder su presencia en un sistema…

…a la vez que le permite capturar cierta información con el fin de acceder a otros sistemas



Actividad

- Desactivación del registro de sucesos en el sistema (auditoría) para un usuario determinado.
- Permitir el acceso al sistema como administrador utilizando un usuario/contraseña oculto.
- Desactivación directa de la seguridad mediante la aplicación parches en el núcleo.
- Guardar capturas de contraseñas o de pulsaciones de teclado.

Ocultación

- **Carpetas y ficheros.**
  - El propio ejecutable del rootkit.
  - Ficheros de contraseñas/teclas capturadas.
  - Otras herramientas.
- **Claves del registro.**
  - Definición de un driver o un servicio estándar.
  - Claves pertenecientes a herramientas instaladas por el intruso.
- **Usuarios/Grupos.**
- **Conexiones.**
  - Acceso remoto.
  - IRC Bots.
- **Procesos.**
- **Retrospectiva: Virus Full-Stealth en MSDOS**

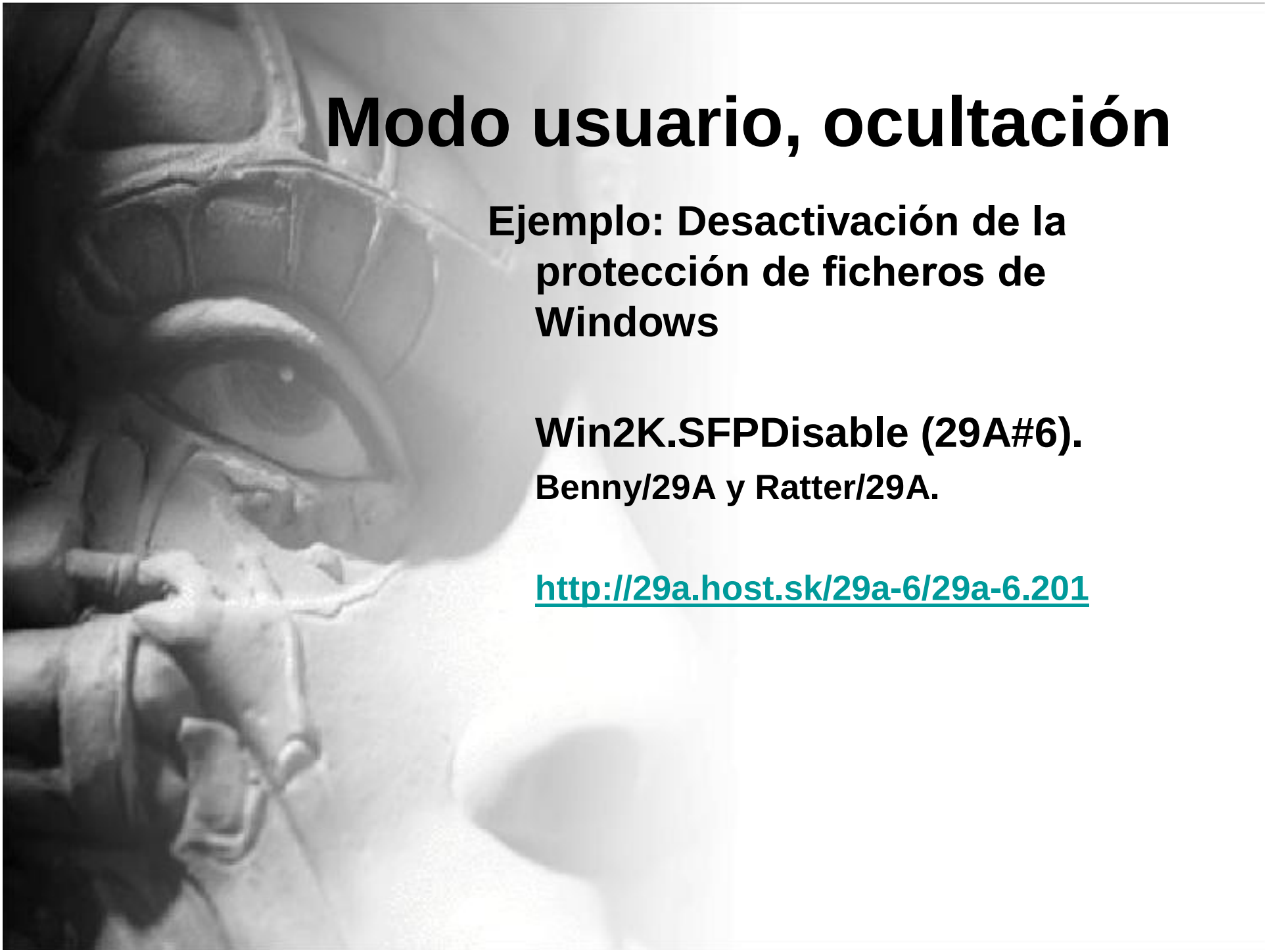
Modo usuario, residencia

- Modificación de binarios del sistema (netstat…).
- Servicio (Stigma).
- Inyección dentro de procesos del sistema (trusted processes, Hacker Defender, Vanquish).
  - OpenProcess.
  - WriteProcessMemory.
  - CreateRemoteThread.
  - /device/physicalmemory.
- Protección del sistema: Windows File Protection.

Modo usuario, ocultación

Ejemplo: Vanquish Rootkit

- CreateProcessW
- FindFirstFileExW
- FindNextFileW
- LogonUserA
- LogonUserW
- RegCloseKey
- RegEnumKeyW
- RegEnumKeyA
- RegEnumKeyExW
- RegEnumKeyExA
- RegEnumValueW
- RegEnumValueA
- RegQueryMultipleValuesW
- SetLocalTime
- SetSystemTime
- SetTimeZoneInformation
- SetSystemTimeAdjustment
- DeleteFileA
- DeleteFileW
- RemoveDirectoryA
- RemoveDirectoryW



Modo usuario, ocultación

Ejemplo: Desactivación de la protección de ficheros de Windows

Win2K.SFPDisable (29A#6).

Benny/29A y Ratter/29A.

http://29a.host.sk/29a-6/29a-6.201

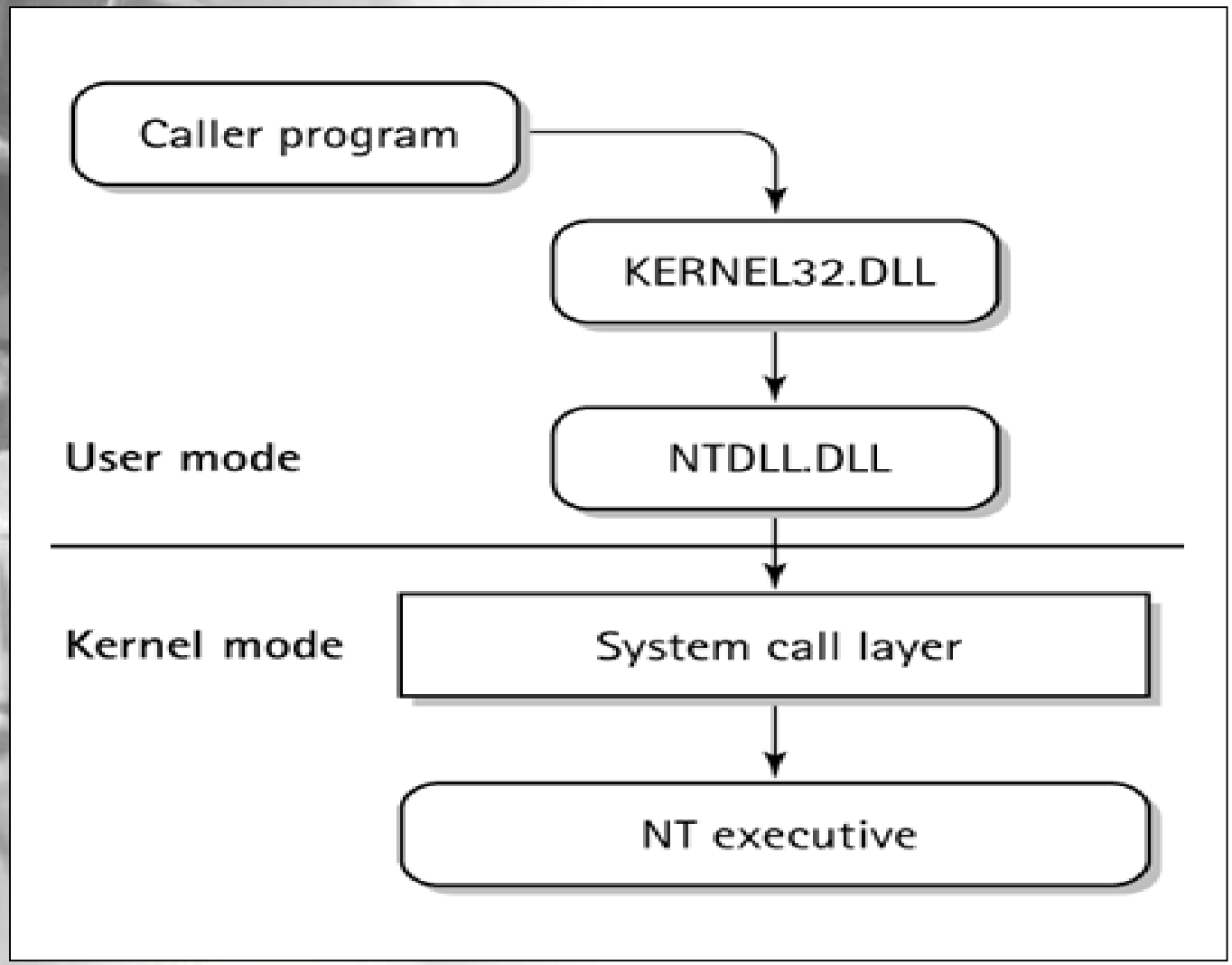
Modo usuario, detección

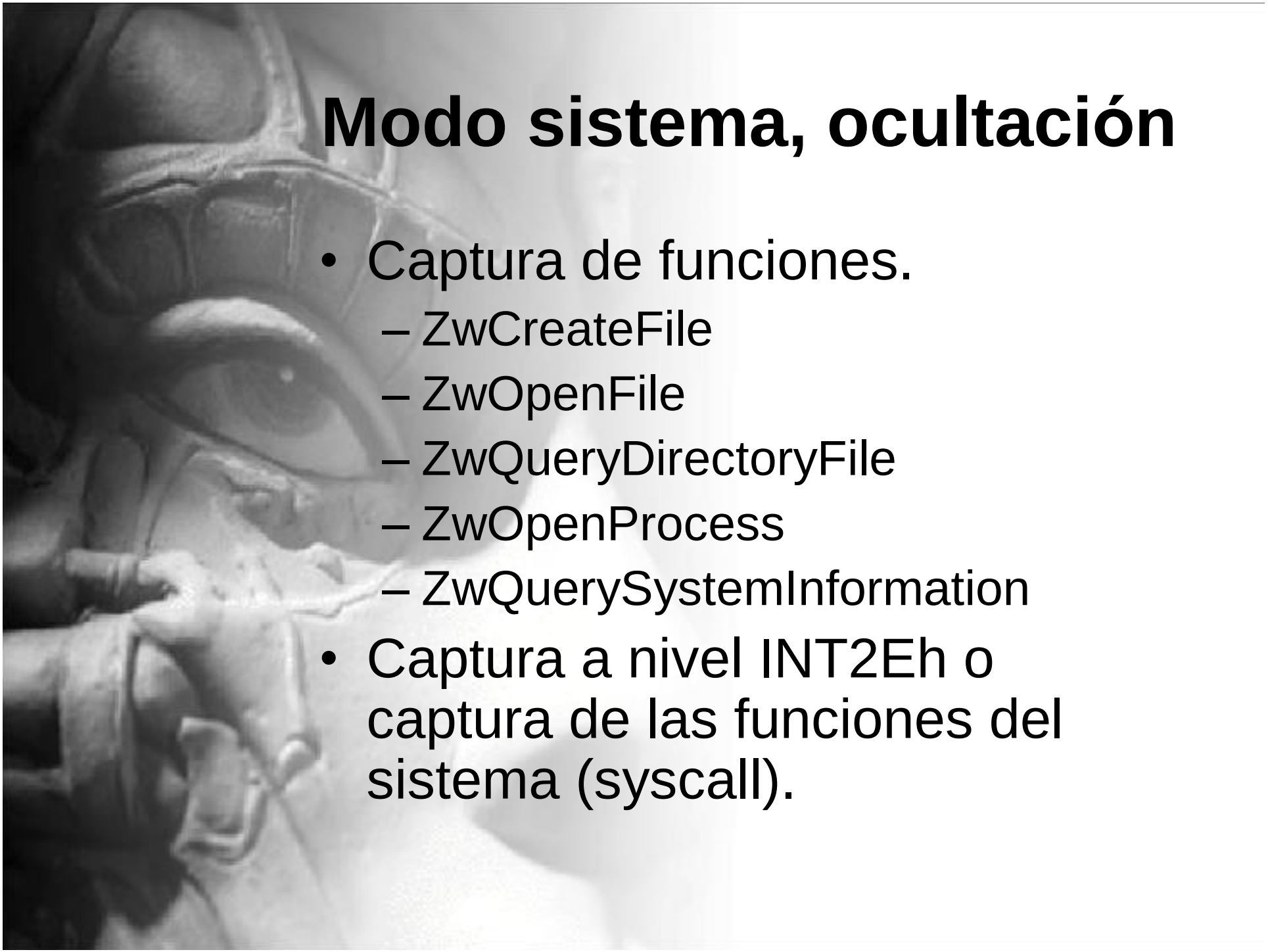
- Llamadas a bajo nivel no controladas por el rootkit.

Modo sistema, residencia

- Driver.
  - Driver estándar.
  - Imagen cargada en el núcleo mediante ZwLoadDriver.
  - ZwSetSystemInformation (SystemLoadImage / SystemLoadAndCallImage).
- Modificación del núcleo o de sus estructuras internas.
  - /dev/physicalmemory.
- Desbordamiento en alguna función del núcleo.
- Protección del sistema: Drivers firmados.

Modo sistema, ocultación





Modo sistema, ocultación

- Captura de funciones.
  - ZwCreateFile
  - ZwOpenFile
  - ZwQueryDirectoryFile
  - ZwOpenProcess
  - ZwQuerySystemInformation
- Captura a nivel INT2Eh o captura de las funciones del sistema (syscall).

Captura a nivel INT2Eh

```
NtCreateFile:
        mov    eax, 0x0000001A
        lea    edx, [esp+04]
        int    0x2E
        ret    0x2C
```

EAX > Identificador de la funcion.

EDX > Puntero a los parámetros en la pila.

Captura a nivel INT2Eh

- IDTR > Deposita la dirección base de la IDT en la localización especificada en el segundo parámetro.
- A partir de la base podemos indexar el descriptor de una interrupción, manipulando su selector y su desplazamiento.

Captura de las funciones del sistema (syscall)

- Las APIs de KERNEL32 son un envoltorio entorno a las APIs de NTDLL, mucho más primitivas.
- Cada servicio del sistema tiene un identificador.
  - 12bit > Identificador del servicio.
  - 2bit > Tabla de servicios a emplear.
    - 0 > Servicios del núcleo (NTDLL).
    - 1 > Servicios del entorno gráfico (WIN32K).
    - 2 y 3 > No se emplean.
  - 18bit > Reservados

Captura de las funciones del sistema (syscall)

- KeServiceDescriptorTable contiene la tabla de funciones.

```
typedef struct {
PVOID *rgpfnHandlerTable;
PULONG rgulCounterTable;
ULONG cServices;
PUCHAR rguchParamTable;
} sst ;
```

Captura de las funciones del sistema (syscall)

- Necesitamos obtener el identificador del servicio que queremos interceptar.
- El código de una función comienza cargando el identificador del servicio en EAX.
  - MOV EAX,id ; B8h XX XX XX XX

Captura de las funciones del sistema (syscall)

```
#define SSTForId(x) (((ULONG) x) >> 12)
#define IndexForId(x) (((ULONG) x) & 0xFFFUL)
#define ValidId(x) ((((ULONG) x) & ~0x3FFFUL) == 0UL)

ULONG ServiceIdFromFn (PVOID pfnHandler)
{
	PUCHAR		pbHandler ;
	ULONG	ulService ;

	pbHandler = (PUCHAR) pfnHandler ;

	if( *pbHandler != 0xB8)
	{
		DbgPrint(( "GKit: ServiceIdFromFn() Expected B8 got %02x\n", *pbHandler)) ;
		return 0UL ;
	}

	ulService = *( PULONG)( pbHandler + 1) ;

	if( !ValidId( ulService))
	{
		DbgPrint( ( "GKit: ServiceIdFromFn() Bogus service ID %08x\n", ulService)) ;
		return 0UL ;
	}

	return ulService ;
}
```

Captura de las funciones del sistema (syscall)

- **En un intento por proteger la tabla de servicios, los desarrolladores de Microsoft decidieron denegar el acceso de escritura sobre ella a partir de Windows XP. La solución más sencilla consiste en desactivar la protección de escritura en el registro de control CR0 antes de modificar la tabla.**

- **Creamos las siguientes macros:**

```
#define WPOFF() \
_asm mov eax, cr0 \
_asm and eax, NOT 10000H \
_asm mov cr0, eax

#define WPON() \
_asm mov eax, cr0 \
_asm or eax, 10000H \
_asm mov cr0, eax
```

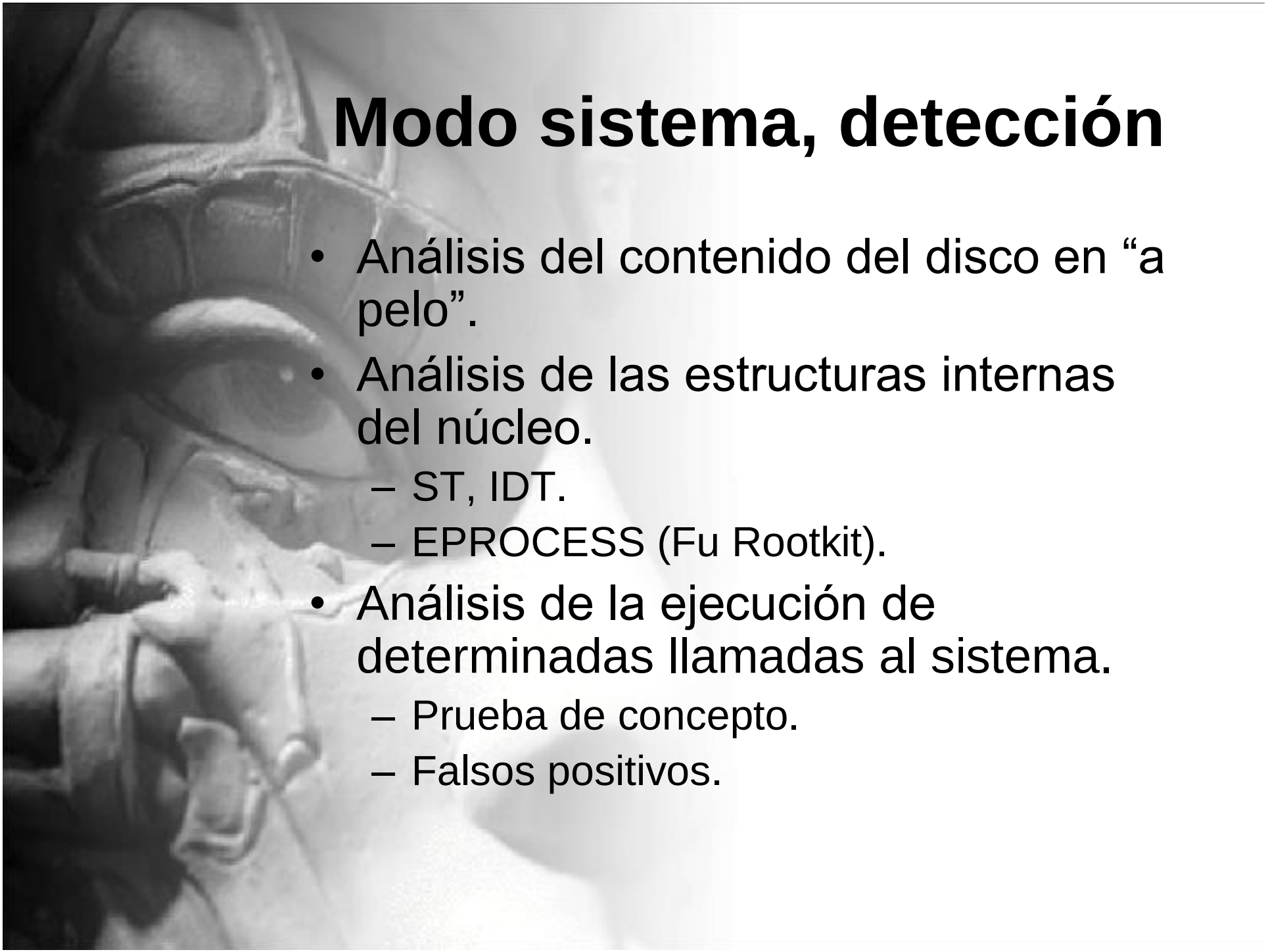
Captura de las funciones del sistema (syscall)

- **El código que parchea la tabla de servicios queda de la siguiente manera:**

```
DbgPrint( ( "GKit: HookSystemService() Hooking SST %x Index %03x Old %08x\n",
ulSST, ulIndex, ( ULONG) pfnOldHandler)) ;

__try
{
        WPOFF() ;

        pfnOldHandler = InterlockedExchangePointer( pfnHandler, pfnNewHandler) ;

        WPON() ;

        DbgPrint( ( "GKit: HookSystemService() Done\n"));

}
__except( EXCEPTION_EXECUTE_HANDLER)
{
        pfnOldHandler = pfnHandler ;

        DbgPrint( ( "GKit: HookSystemService() Hook failed - can't write\n")) ;
}
```

Captura de las funciones del sistema (syscall)

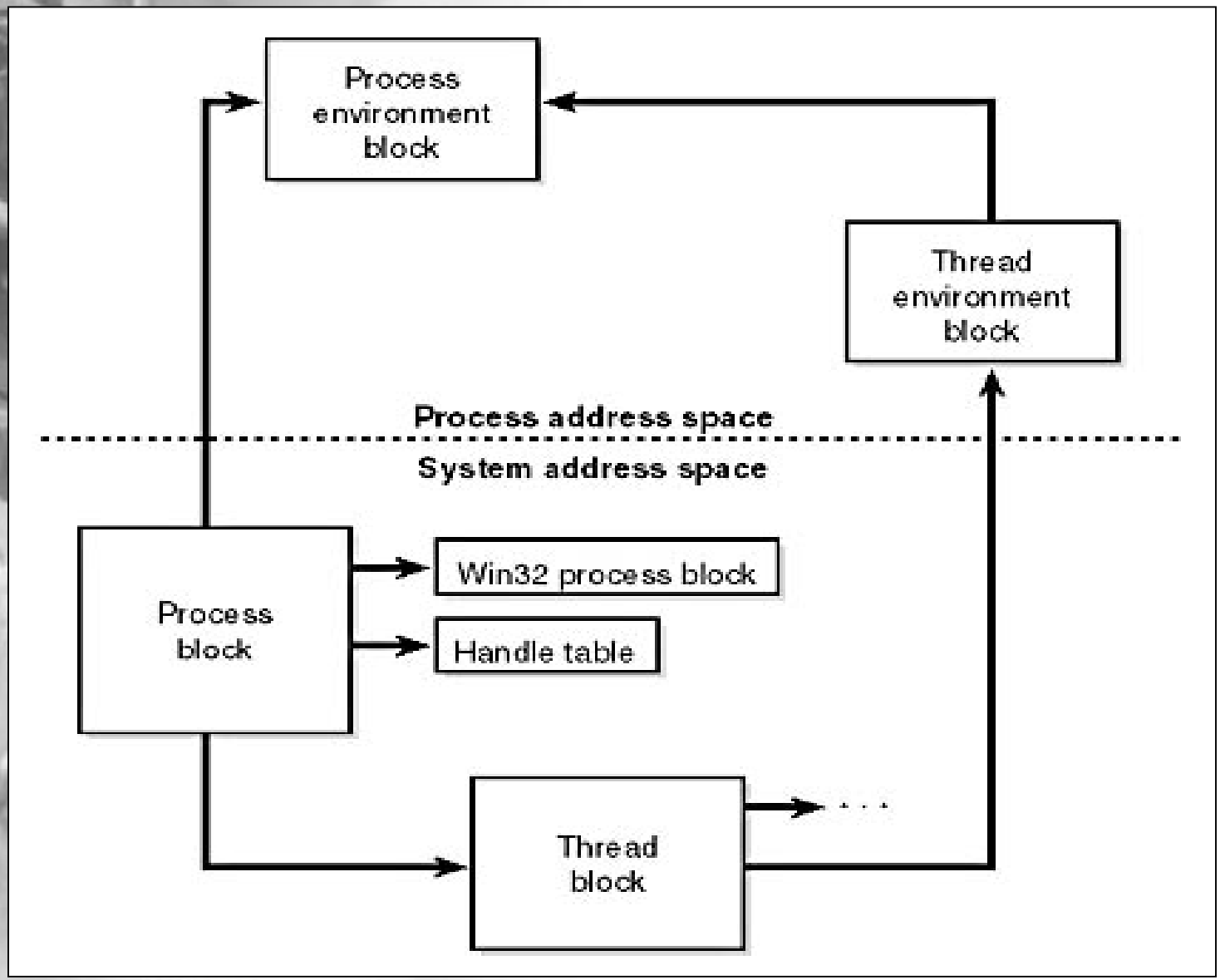
- **Referencias:**
  - **Mark Russinovich and Bryce Cogswell, "Windows NT System-Call Hooking", Dr. Dobb's Journal January 1997.**
  - **Nishad P. Herath, "Adding Native APIs", post to microsoft.public.win32.programmer.kernel newsgroup dated 11 October 1998.**
  - **Prasad Dabak, Sandeep Phadke, Milind Borate, "Undocumented Windows NT", M&T Books, October 1999.**
  - **Tim J. Robins, "Windows NT System Service Table Hooking", May 2003.**



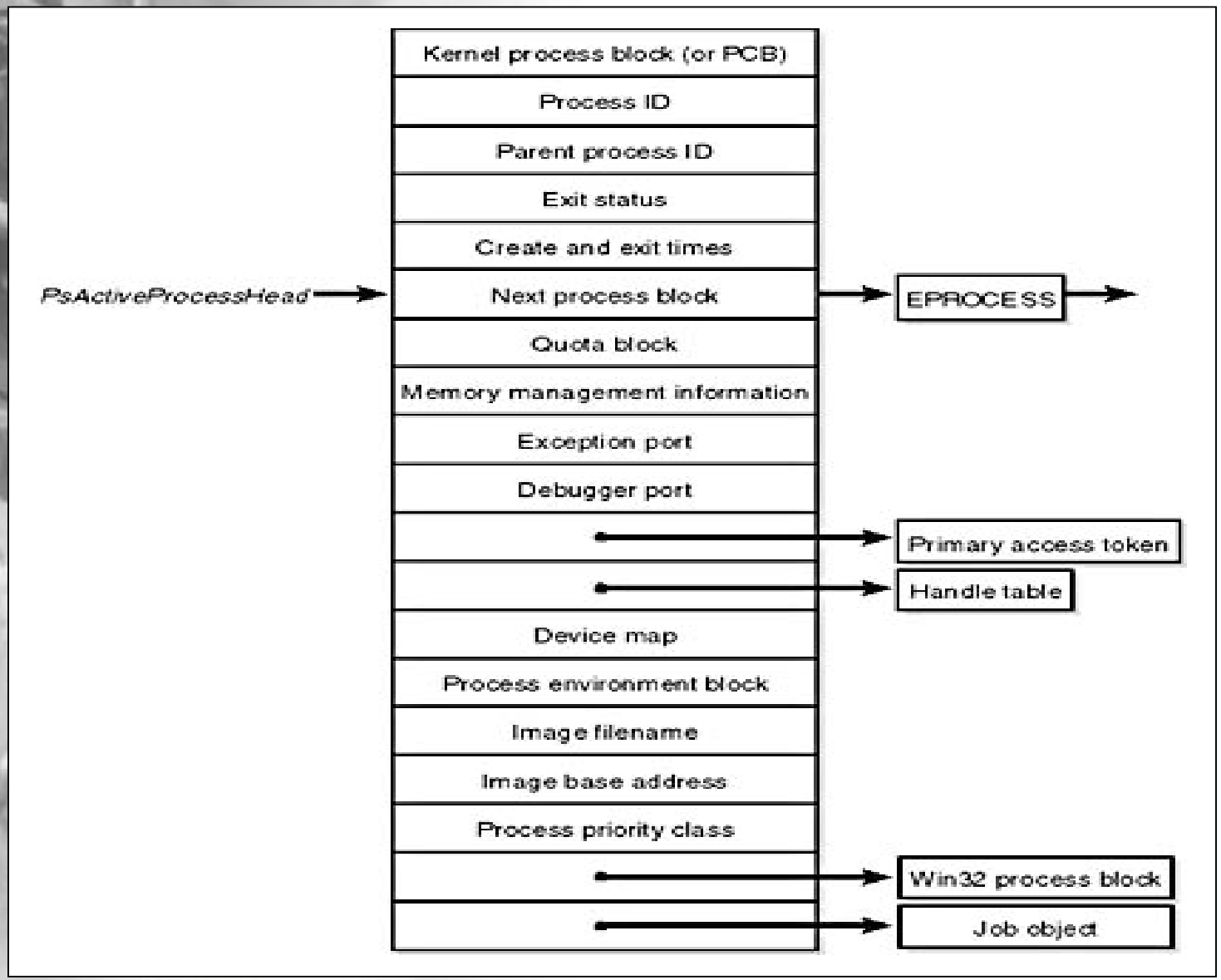
Modo sistema, detección

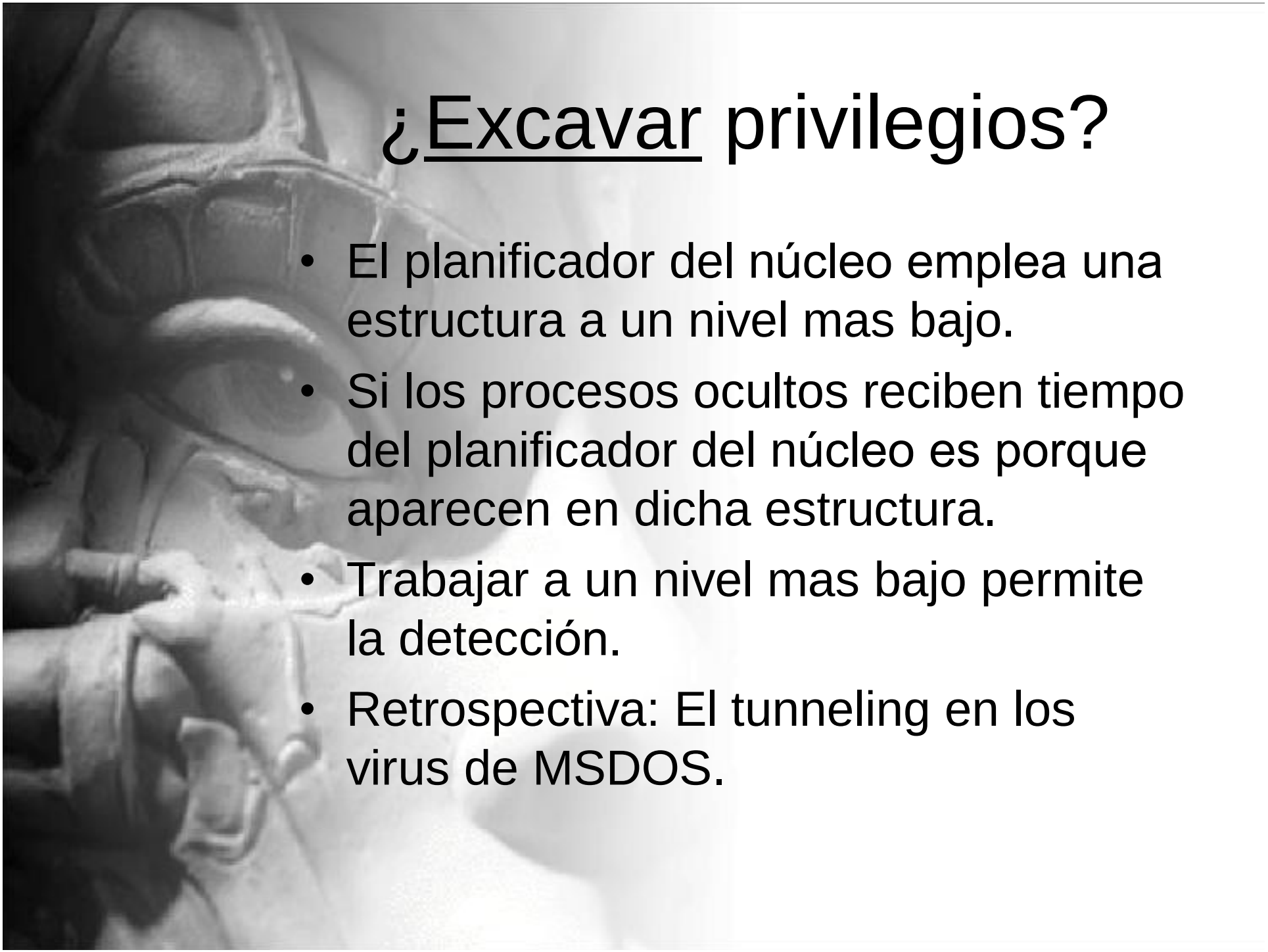
- Análisis del contenido del disco en “a pelo”.
- Análisis de las estructuras internas del núcleo.
  - ST, IDT.
  - EPROCESS (Fu Rootkit).
- Análisis de la ejecución de determinadas llamadas al sistema.
  - Prueba de concepto.
  - Falsos positivos.

EPROCESS


Process environment block
Thread environment block
Process address space
System address space
Process block
Win32 process block
Handle table
Thread block
. . .

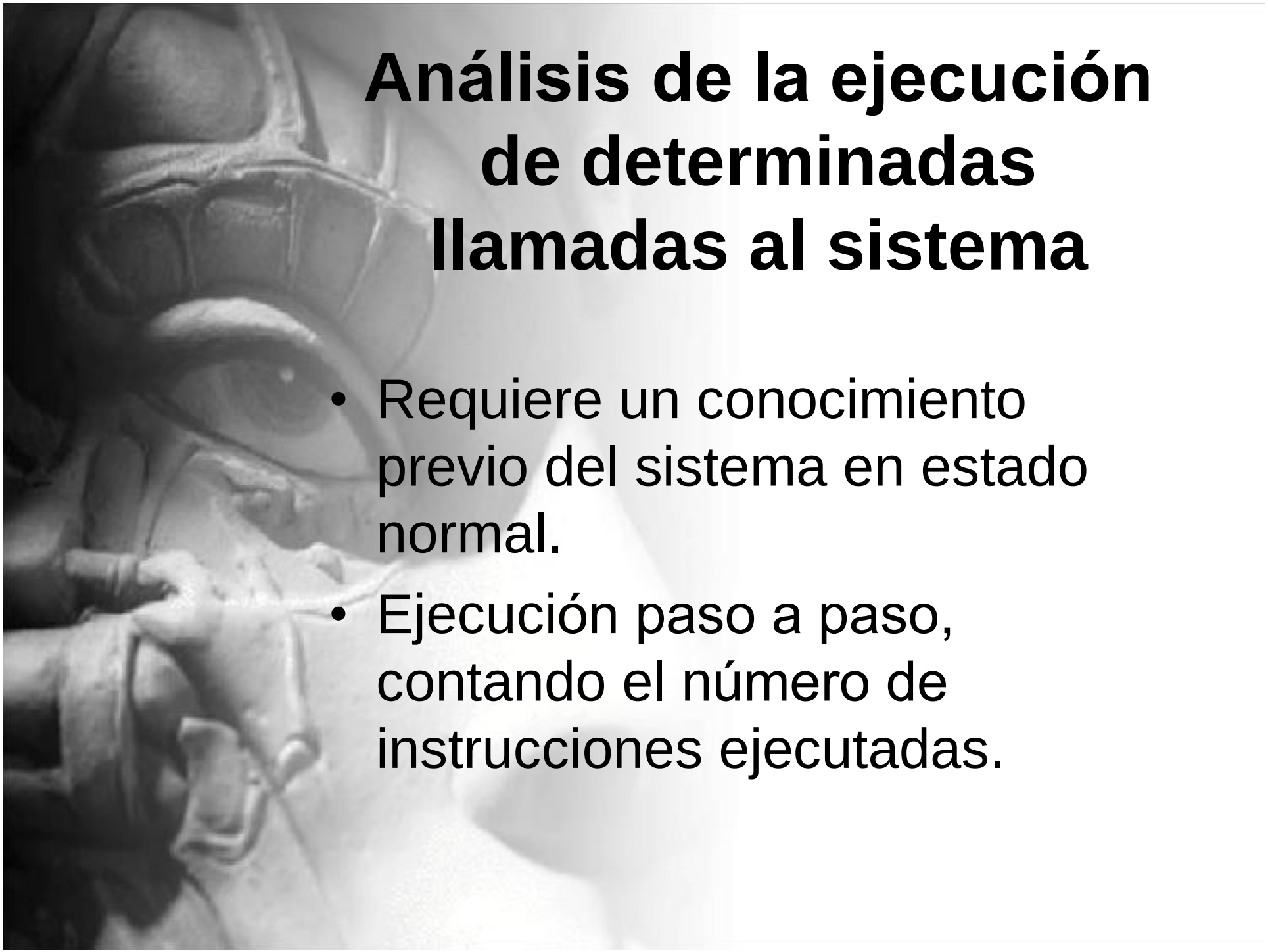
EPROCESS


Kernel process block (or PCB)
Process ID
Parent process ID
Exit status
Create and exit times
PsActiveProcessHead
Next process block
EPROCESS
Quota block
Memory management information
Exception port
Debugger port
Primary access token
Handle table
Device map
Process environment block
Image filename
Image base address
Process priority class
Win32 process block
Job object



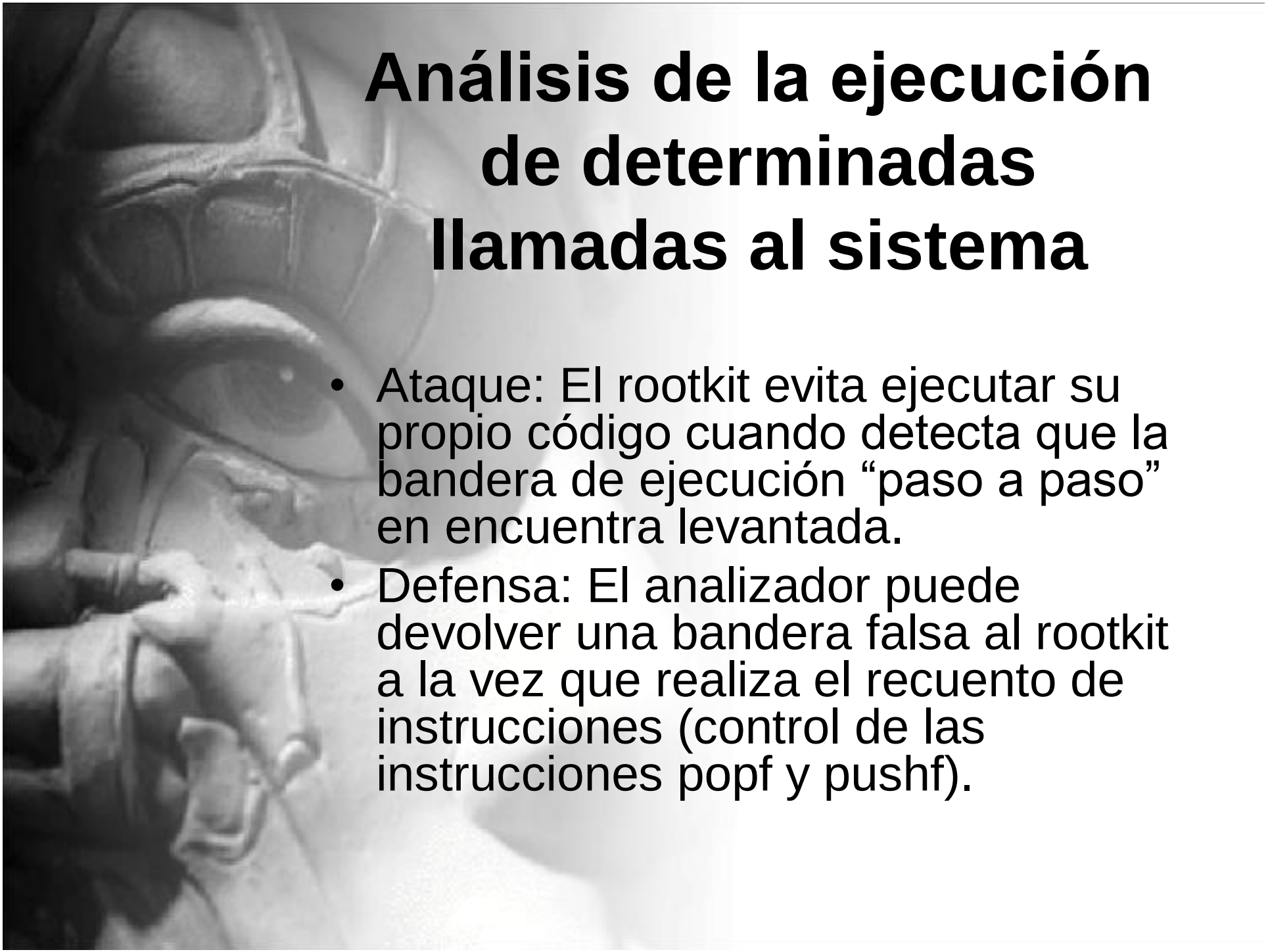
¿Excavar privilegios?

- El planificador del núcleo emplea una estructura a un nivel mas bajo.
- Si los procesos ocultos reciben tiempo del planificador del núcleo es porque aparecen en dicha estructura.
- Trabajar a un nivel mas bajo permite la detección.
- Retrospectiva: El tunneling en los virus de MSDOS.



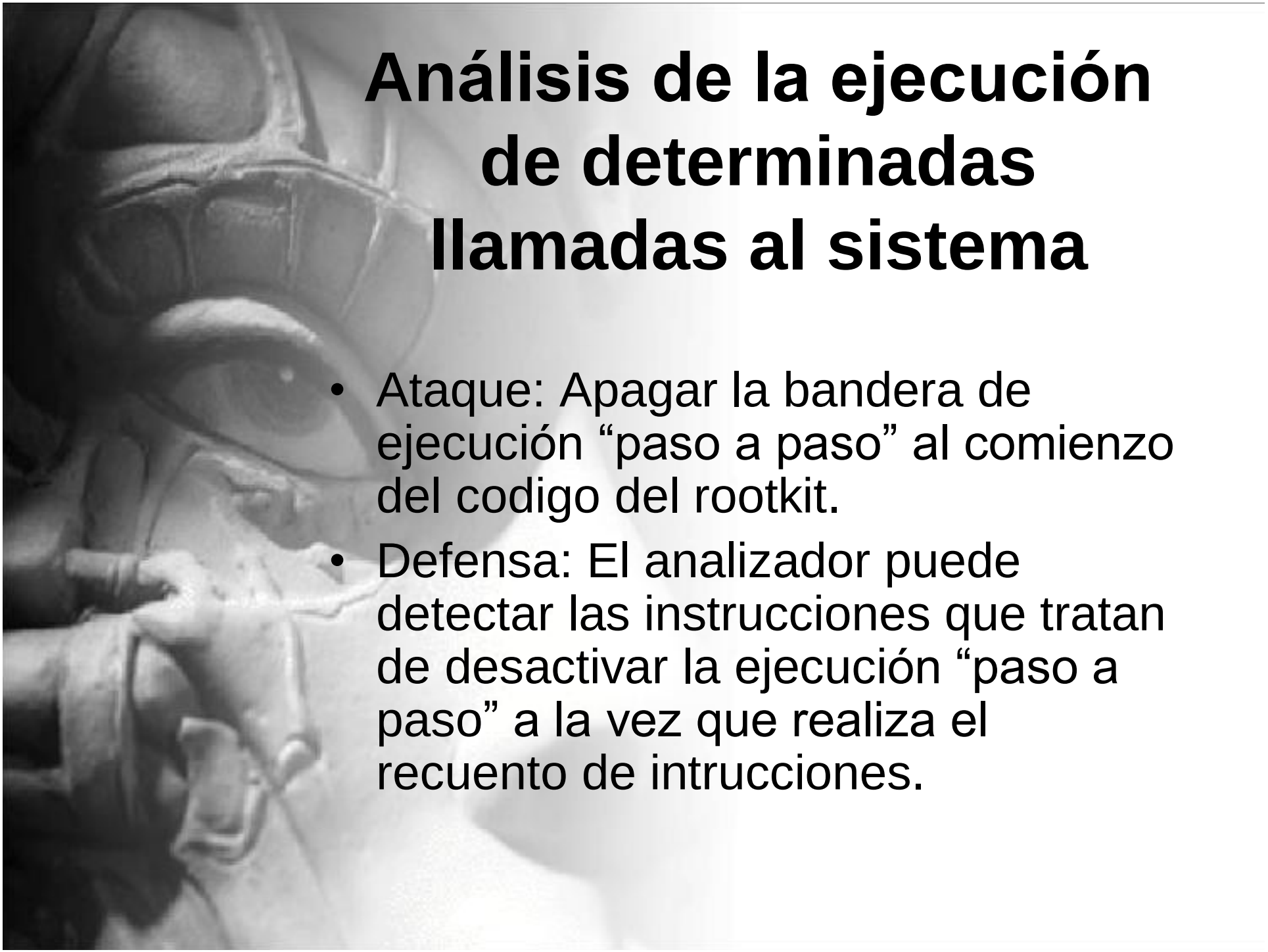
Análisis de la ejecución de determinadas llamadas al sistema

- Requiere un conocimiento previo del sistema en estado normal.
- Ejecución paso a paso, contando el número de instrucciones ejecutadas.



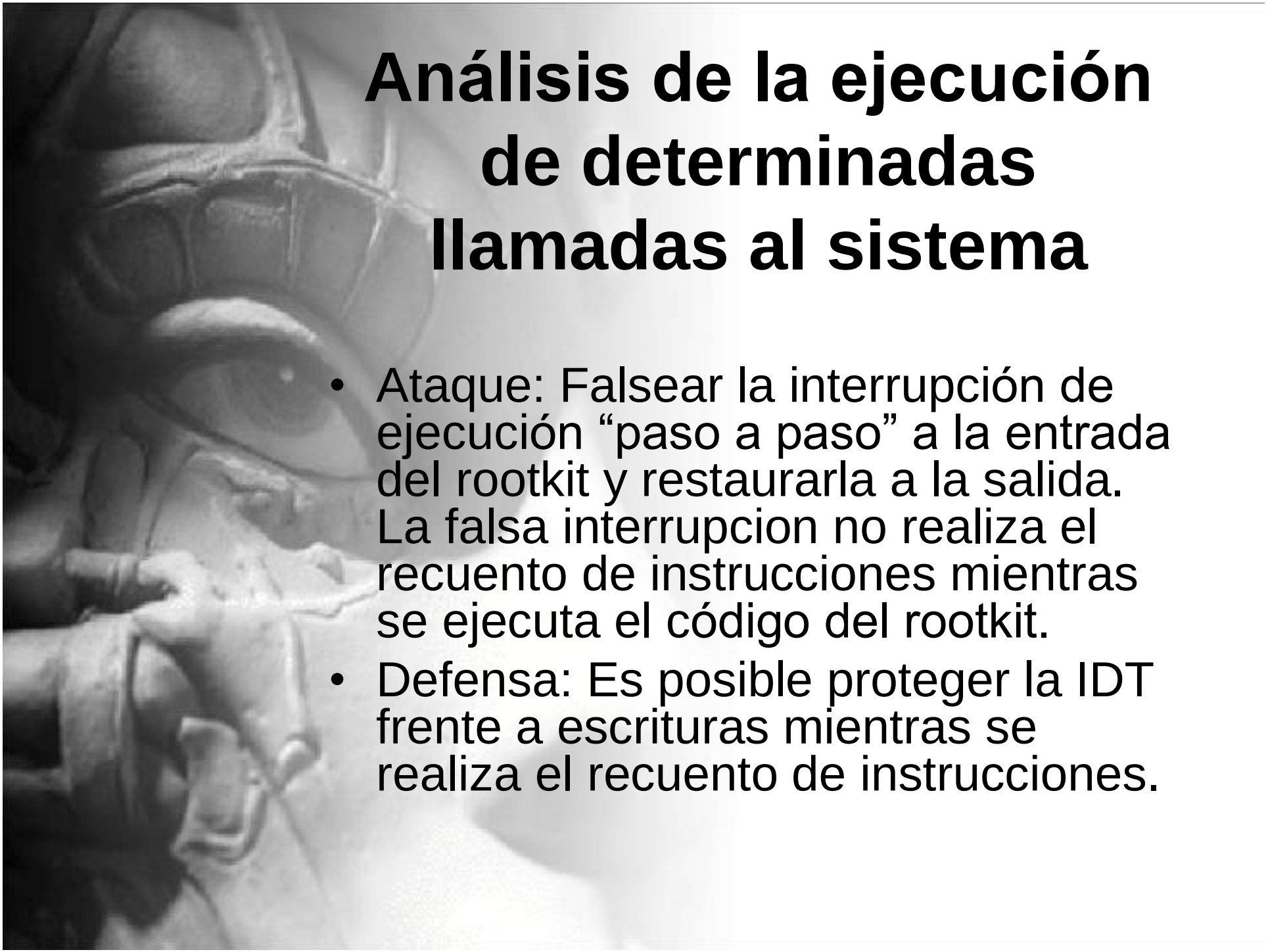
Análisis de la ejecución de determinadas llamadas al sistema

- Ataque: El rootkit evita ejecutar su propio código cuando detecta que la bandera de ejecución “paso a paso” en encuentra levantada.
- Defensa: El analizador puede devolver una bandera falsa al rootkit a la vez que realiza el recuento de instrucciones (control de las instrucciones popf y pushf).



Análisis de la ejecución de determinadas llamadas al sistema

- Ataque: Apagar la bandera de ejecución “paso a paso” al comienzo del codigo del rootkit.
- Defensa: El analizador puede detectar las instrucciones que tratan de desactivar la ejecución “paso a paso” a la vez que realiza el recuento de intrucciones.

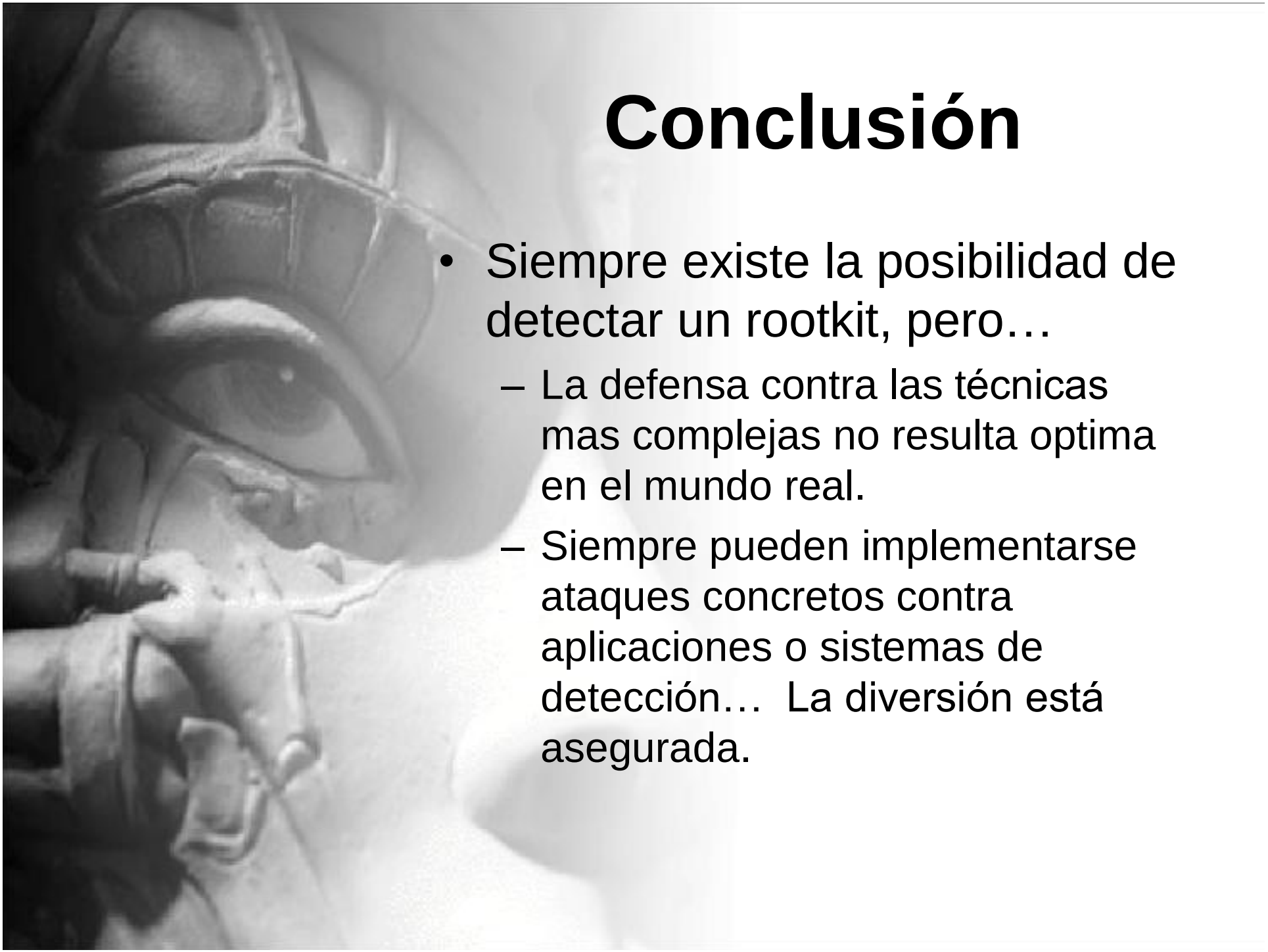


Análisis de la ejecución de determinadas llamadas al sistema

- Ataque: Falsear la interrupción de ejecución “paso a paso” a la entrada del rootkit y restaurarla a la salida. La falsa interrupcion no realiza el recuento de instrucciones mientras se ejecuta el código del rootkit.
- Defensa: Es posible proteger la IDT frente a escrituras mientras se realiza el recuento de instrucciones.

Acceso remoto

- Modo usuario.
  - Falsear WINSOCK para evitar la detección.
  - Emplear un sniffer e implementar un protocolo de comunicación “stateless” entre el maestro y el rootkit (Stigma).
- Modo sistema.
  - Implementación de una pila TCP/IP propia (ocultación completa de las comunicaciones, NTROOTKIT).
  - Captura del tráfico (NDIS Hook, IP Filter Hook) y comunicación “stateless”.



Conclusión

- Siempre existe la posibilidad de detectar un rootkit, pero…
  - La defensa contra las técnicas mas complejas no resulta optima en el mundo real.
  - Siempre pueden implementarse ataques concretos contra aplicaciones o sistemas de detección… La diversión está asegurada.