

A long-exposure photograph of a volcano erupting at night. The central focus is a massive, bright yellow and orange plume of incandescent lava fountains that rises high into the dark sky. Below the main eruption, a thick, glowing lava flow cascades down the slope of the volcano, its surface punctuated by numerous smaller, bright orange and red spots of intense heat. The overall scene is dominated by the fiery colors of the eruption against the deep black background of the night.

Ring of Fire

Par Sébastien Dudek (FIUxluS)

Sommaire

- 1) Présentation des rootkits
- 2) Un peu d'architecture
- 3) Premiers pas sur le Ring0
- 4) La communication User<->kernel
- 5) Le DKOM
- 6) Références

Les Rootkits : Présentation

Un Rootkits : Virus, Exploit ? O_o

Un rootkit peut être
un virus
ou un exploit chef !

MON C*L !!!



Rootkit : Culture générale

- Une technologie (ensemble de programmes et codes)
- Reprend le principe des virus (années 80) :
 - Modifications logiques d'un programme
 - Clés des tables systèmes
 - Mémoire
 - ...
- Objectif : Rendre un code furtif et indétectable
- Ne se reproduit pas != virus

Le Rootkit première génération

- Très primitif = Backdoor
- Possibilités : Remplacer des binaires → cacher des fichiers et processus
- Exemple (remplacement de « /bin/lis ») :

```
$ ls  
helloworld.c  angelina.png
```

Nous avons caché à la vue de l'utilisateur notre fichier « Piraterlaplanet.bin »

Le Rootkit 2G

- Introduction aux Rootkits Kernel
- Utilisation de 2 modes : User + Kernel
- Hooking :
 - IAT (Import Address Table)
 - EAT (Export Address Table)
 - SSDT (System Service Dispatch Table)
 - IDT (Interrupt Descriptor Table)

Le Rootkit 3G

- Principe : modification des objets dynamiques de l'OS chargés en mémoire
- DKOM (Direct Object Manipulation) :
 - Exemple : FU Rootkit
- Les objets Kernel : Processus, threads, drivers, etc...
- Montre la faiblesse des systèmes de détections actuels

Le Rootkit « 3G+ » ou Firmware

- Firmware
- Ajout de fonctionnalités → reverse engineering
- Taille du firmware limitée → faire un rootkit adapté à la capacité de stockage.
- Vecteurs :
 - BIOS → Cible parfaite (Rappel: CIH)
 - PCI
 - PCMCIA
 - USB → Clés U3
- Plus : Présentation Hack.lu par Guillaume Delugré (Sogeti ESEC)

Motifs d'un attaquant

- Un attaquant → pénétrer un système → avoir des informations
- Détruire un système ou Garder un accès privilégié sur la machine → revenir sur la machine à un autre moment



Utilisations et légalité

- Utilisés pour attaquer (vue précédemment)
- Mais aussi pour défendre (Ex: LoJack for Laptops)
- Un rootkit en lui même n'est pas illégale
- Rootkit → concept de modification (Sources-code, patching, easter eggs, spyware, updater & packages)
- Une modification peut compromettre/violer le copyright → illégale

Composants attaqués

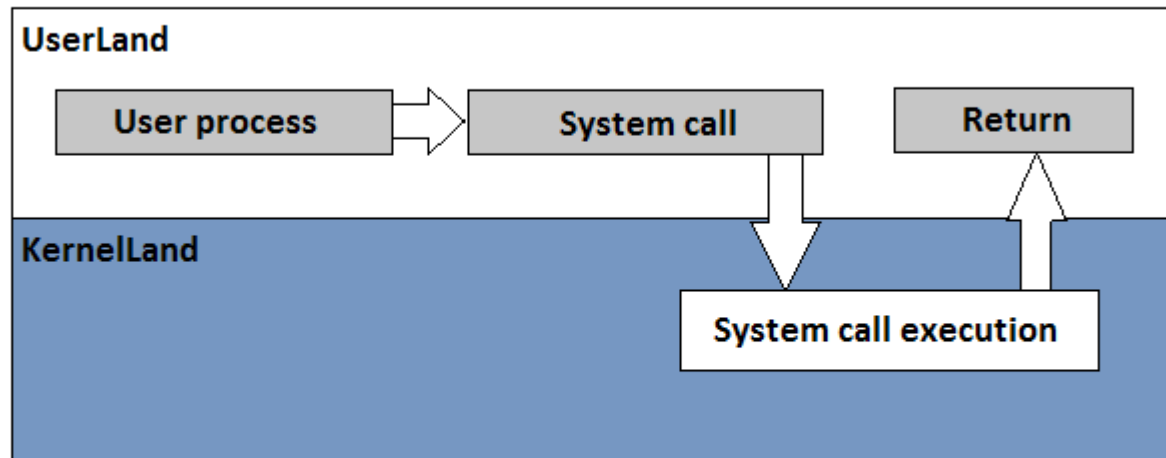
- Les composants du OS attaqués :
 - Gestion E/S (Log de frappes, activités réseaux)
 - Périphériques & systèmes de fichiers (Cacher des fichiers)
 - Gestion d'objets (Cacher des handles de processus/threads)
 - Moniteur de sécurité (Désactivation des règles de sécurité)
 - Gestion des threads et processus (Cacher processus et threads)
 - Gestion de la configuration (Cacher des entrées du registre)

Un peu d'architecture

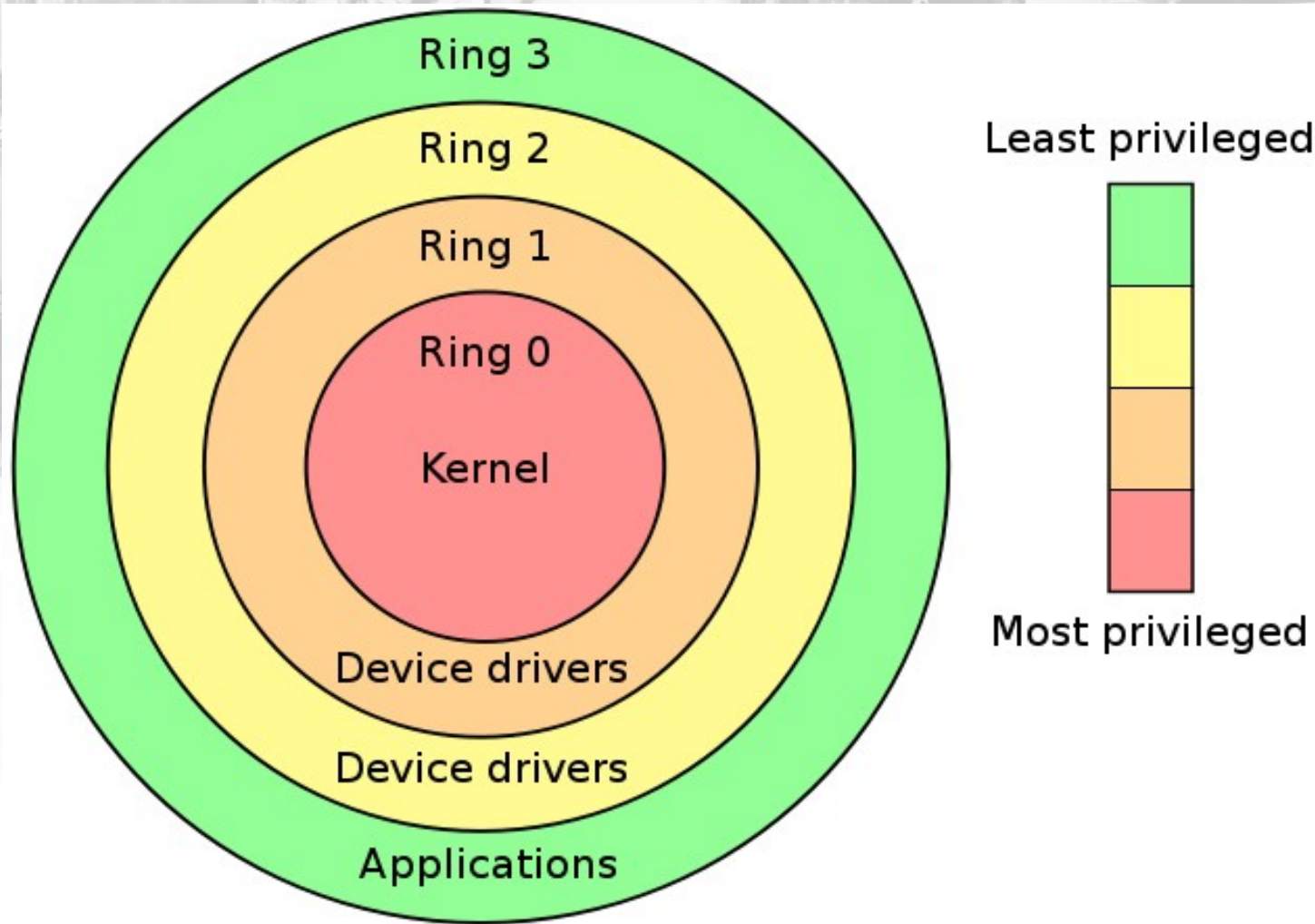
Intel x86 : Interruptions (1)

- Un système attend toujours quelque chose à faire (Exécution de processus, Périphérique E/S, Répondre aux utilisateurs).
- Les événements → signalés par des interruptions ou traps (Exceptions).
- Trap : Causé par une erreur (Ex: Division par zéro ou accès mémoire invalide).
- Besoin de privilèges ? → Interfacé le kernel grâce aux « system calls ».

Intel x86 : Interruptions (2)



Intel x86 : Rings (1)



Intel x86 : Rings (2)

- 4 Rings utilisés pour le contrôle d'accès
- Ring0 → le plus privilégié
- Ring3 → le moins privilégié
- Windows et Linux utilise le ring0 et ring3 (pas besoin des autres modes)
- Ring0 → accès à la mémoire virtuelle, matériel, etc...
- Ring3 → Ring0 = interruption (anormal)



Emmènes-moi au Ring0

JAKOB EKLUND / JAN MYBRAND
IDA ENGVOLL / HELENA AF SANDEBERG
PETRONELLA BARKER / ÅKE LUNDOVIST
SAMUEL FRÖLER «IBSEN 2010»

KATARINA EWERLÖF «MOSKVA 7 OKTOBER»

META VELANDER / INGVAR KJELLSON
«RUT OCH RAGNAR»

ANJA LUNDOVIST / LENNART JÄHKE
«KVINNA KLÄDD I SOLEN»

HELENA BERGSTRÖM / KALLE HÄLMBERG
«GENGÅNGARE»

ANSLAGSTAVLA

HÖJ PULS
OCH SPA

VÄRLDENS
FRISKASTE
STAD

Histoire de Kernel Driver

- Extension Kernel
- Code tournant en kernel → possibilité de tout modifier
- Un module comporte un point d'entrée et une routine de nettoyage (utile pour les mises à jours de notre rootkit).

Kernel Driver : Exemple Windows

```
#include "ntddk.h"

VOID CleanUP(IN PDRIVER_OBJECT pDriverObject);

NTSTATUS DriverEntry(IN PDRIVER_OBJECT TheDriverObject, IN
PUNICODE_STRING TheRegistryPath)
{
    DbgPrint("Hello wolrd"); // Fonction Print vu en debuggage

    return STATUS_SUCCESS;
}

VOID CleanUP(IN PDRIVER_OBJECT pDriverObject)
{
    DbgPrint("CleanUp routine called")
}
```

Kernel Driver : Exemple Linux

```
#include <linux/init.h>
#include <linux/module.h>
MODULE_LICENSE("HzV");

static int hello_init(void)
{
    printk(KERN_ALERT "Hello, world\n"); // print présent en
/var/log/messages
    return 0;
}
static void hello_exit(void)
{
    printk(KERN_ALERT "CleanUp routine called\n");
}
module_init(hello_init);
module_exit(hello_exit);
```

Chargement du Kernel Driver : Linux

- Makefile bien configuré
- Exemple de chargement :

```
$ make
```

```
make[1]: Entering directory `/usr/src/linux-2.6.32'
```

```
CC [M] /home/ldd3/src/misc-modules/hello.o
```

```
Building modules, stage 2.
```

```
MODPOST
```

```
CC /home/ldd3/src/misc-modules/hello.mod.o
```

```
LD [M] /home/ldd3/src/misc-modules/hello.ko
```

```
make[1]: Leaving directory `/usr/src/linux-2.6.32'
```

```
$ sudo -s
```

```
root# insmod ./hello.ko
```

```
Hello, world
```

```
root# rmmod hello
```

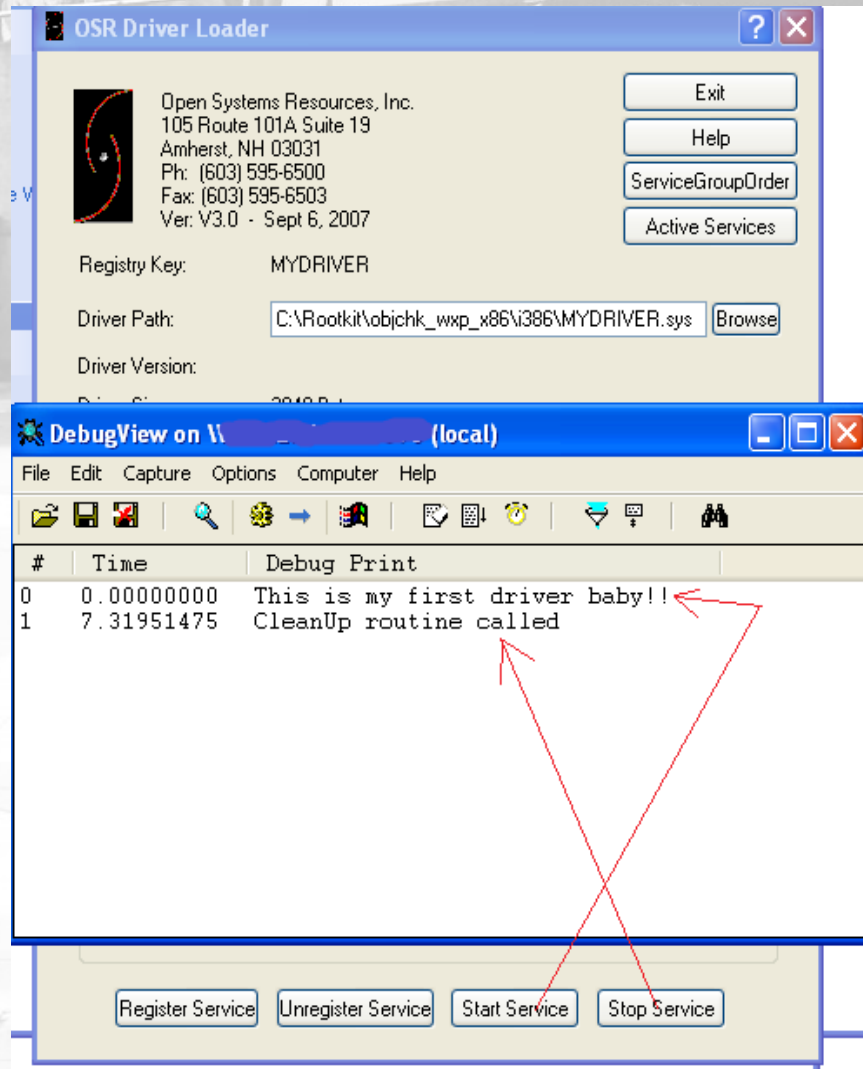
```
CleanUp routine called
```

```
root#
```



Windows : chargement simple

- Utilisation de **OSR Driver Loader** (gratuit)
- Démo :



The image shows two overlapping windows from a Windows operating system. The top window is titled "OSR Driver Loader" and contains the following information:

- Company: Open Systems Resources, Inc.
- Address: 105 Route 101A, Suite 19, Amherst, NH 03031
- Phone: (603) 595-6500
- Fax: (603) 595-6503
- Version: V3.0 - Sept 6, 2007
- Registry Key: MYDRIVER
- Driver Path: C:\Rootkit\objchk_wxp_x86\386\MYDRIVER.sys
- Buttons: Exit, Help, ServiceGroupOrder, Active Services

The bottom window is titled "DebugView on \\... (local)" and displays a table of debug prints:

#	Time	Debug Print
0	0.00000000	This is my first driver baby!!
1	7.31951475	CleanUp routine called

Red arrows point from the "CleanUp routine called" message in the DebugView window to the "Active Services" button in the OSR Driver Loader window. Below the DebugView window are buttons for "Register Service", "Unregister Service", "Start Service", and "Stop Service".

Windows : Chargement embarqué

- Utilisation de Service Control Manager (SCM)
 - Créé les clés de registre
- SCM → Driver non-pageable → pas de soucis avec le « page out »! (donc accès à l'intégralité du driver → pas de BSOD)
- Démo avec exemple de code et base de registre.

```
''  
SC_HANDLE sh = OpenSCManager(NULL, NULL,  
SC_MANAGER_ALL_ACCESS);  
''
```


Communication entre User et Kernel

I/O Request Packets

- Rootkit actuellement : UserLand + KernelLand
- Communication la plus utilisé : IOCTL sur Windows
- On a besoin de I/O Request Packets (IRPs)
- IRP = Buffer de donnée → un utilisateur peu traiter un fichier (lire et écrire)

Structure de DRIVER_OBJECT

```
nt!RtlpBreakWithStatusInstruction:
```

```
80527bdc cc          int    3
```

```
kd> dt nt!_DRIVER_OBJECT
```

```
”
```

```
+0x00c DriverStart      : Ptr32 Void
```

```
+0x010 DriverSize      : Uint4B
```

```
+0x014 DriverSection   : Ptr32 Void
```

```
+0x018 DriverExtension : Ptr32
```

```
_DRIVER_EXTENSION
```

```
+0x01c DriverName      :
```

```
_UNICODE_STRING
```

```
”
```

```
+0x034 DriverUnload    : Ptr32 void
```

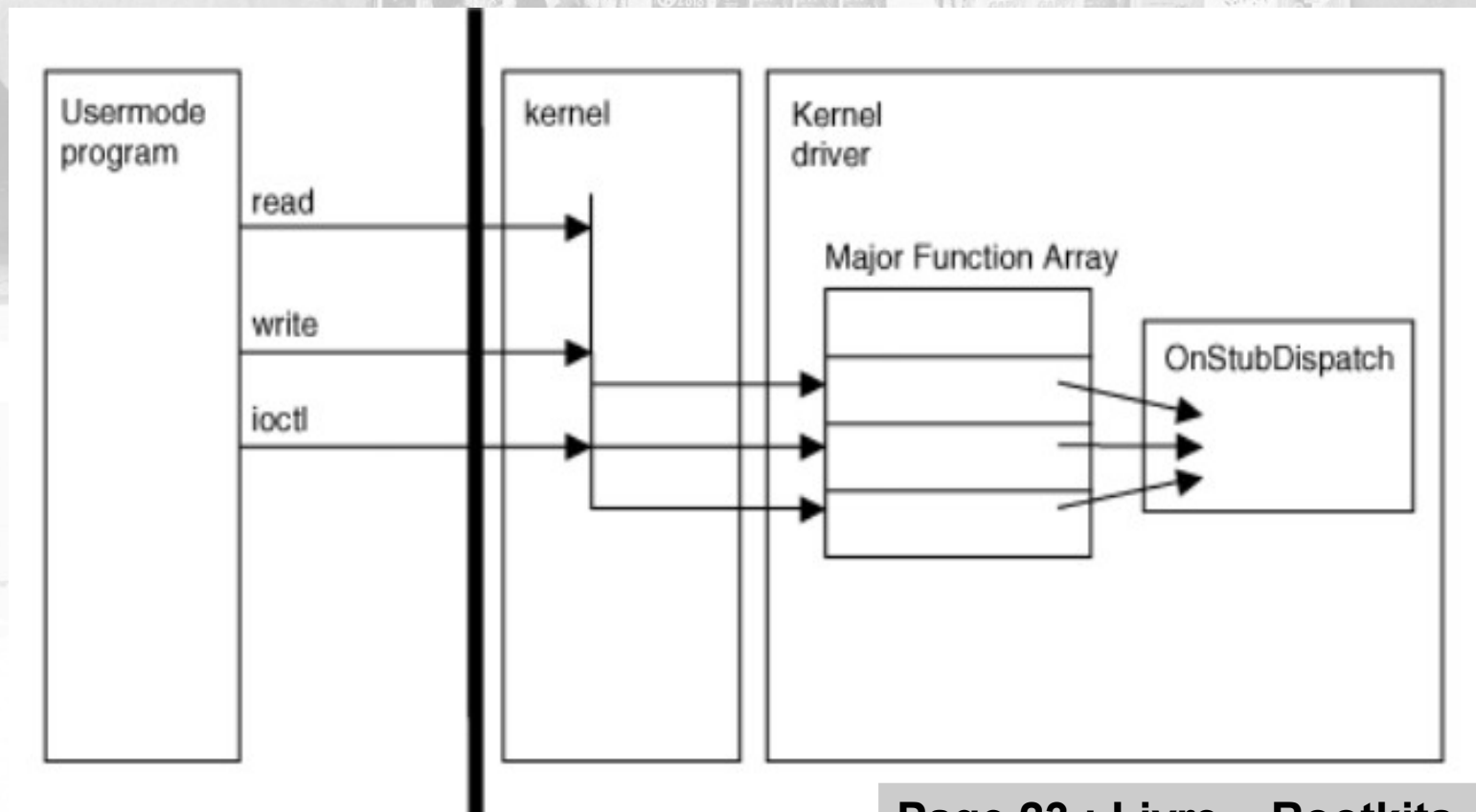
```
+0x038 MajorFunction   : [28] Ptr32 long
```

Les Major Functions

- Traitent les IRPs
- Ont différentes valeurs :
 - IRP_MJ_CLEANUP
 - IRP_MJ_CLOSE
 - IRP_MJ_CREATE
 - IRP_MJ_READ
 - ... (voir MSDN)
- Si on gère CREATE, READ et CLOSE par exemple → en userLand on devra faire CreateFile(), ReadFile() et CloseHandle()

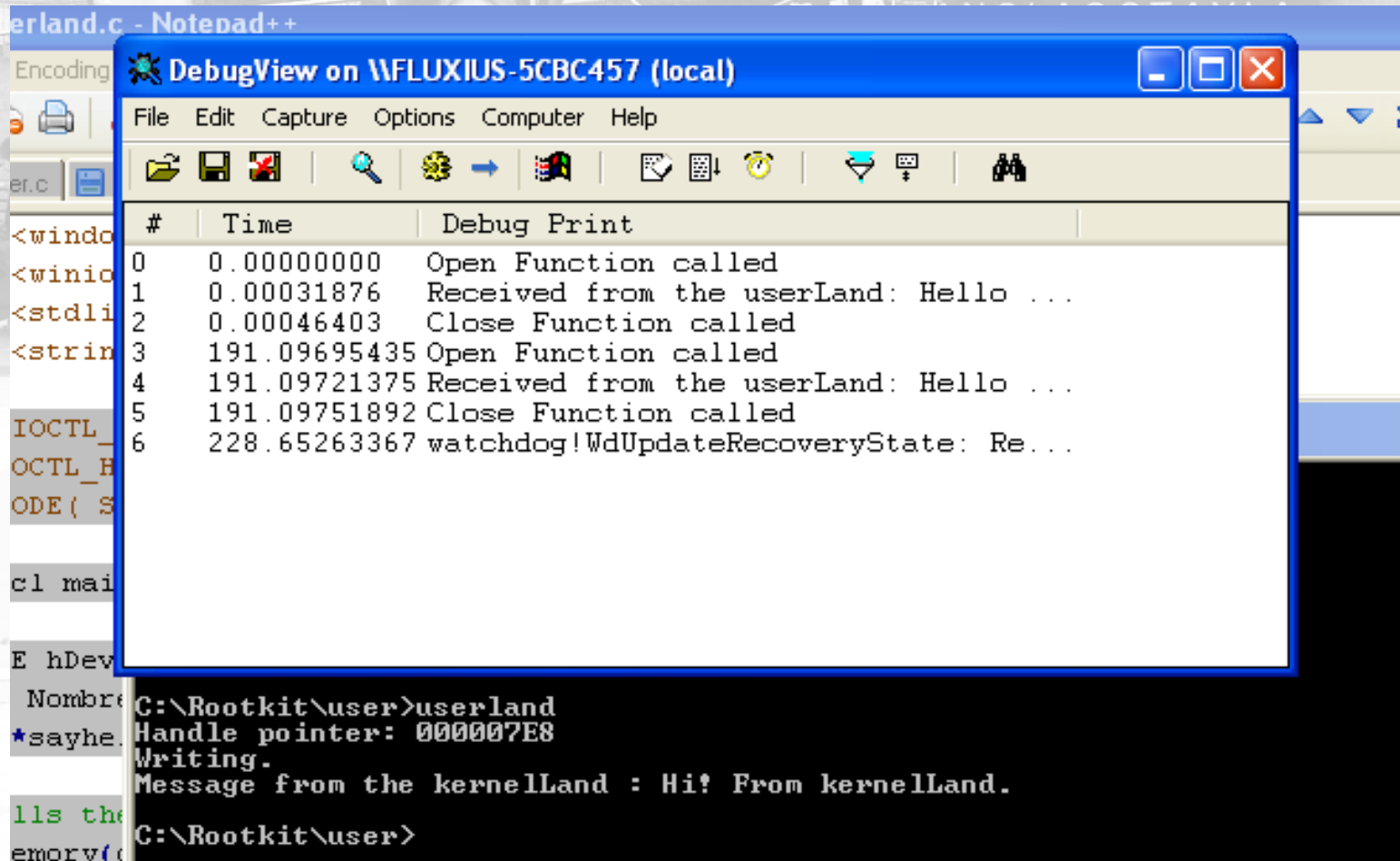
User + Kernel mode (1)

- Pour communiquer nous devrions ouvrir un file handle.
- Schéma :



User + Kernel mode (2)

- Démonstration :



```
erland.c - Notepad++
Encoding
er.c
<windo
<winio
<stdli
<strin
IOCTL_
OCTL_B
ODE ( S
cl mai
E hDev
Nombre
*sayhe
lls the
emoryv(
```

```
DebugView on WFLUXIUS-5CBC457 (local)
File Edit Capture Options Computer Help
[Icons]
# Time Debug Print
0 0.00000000 Open Function called
1 0.00031876 Received from the userLand: Hello ...
2 0.00046403 Close Function called
3 191.09695435 Open Function called
4 191.09721375 Received from the userLand: Hello ...
5 191.09751892 Close Function called
6 228.65263367 watchdog!WdUpdateRecoveryState: Re...

C:\Rootkit\user>userland
Handle pointer: 000007E8
Writing.
Message from the kernelland : Hi! From kernelland.

C:\Rootkit\user>
```

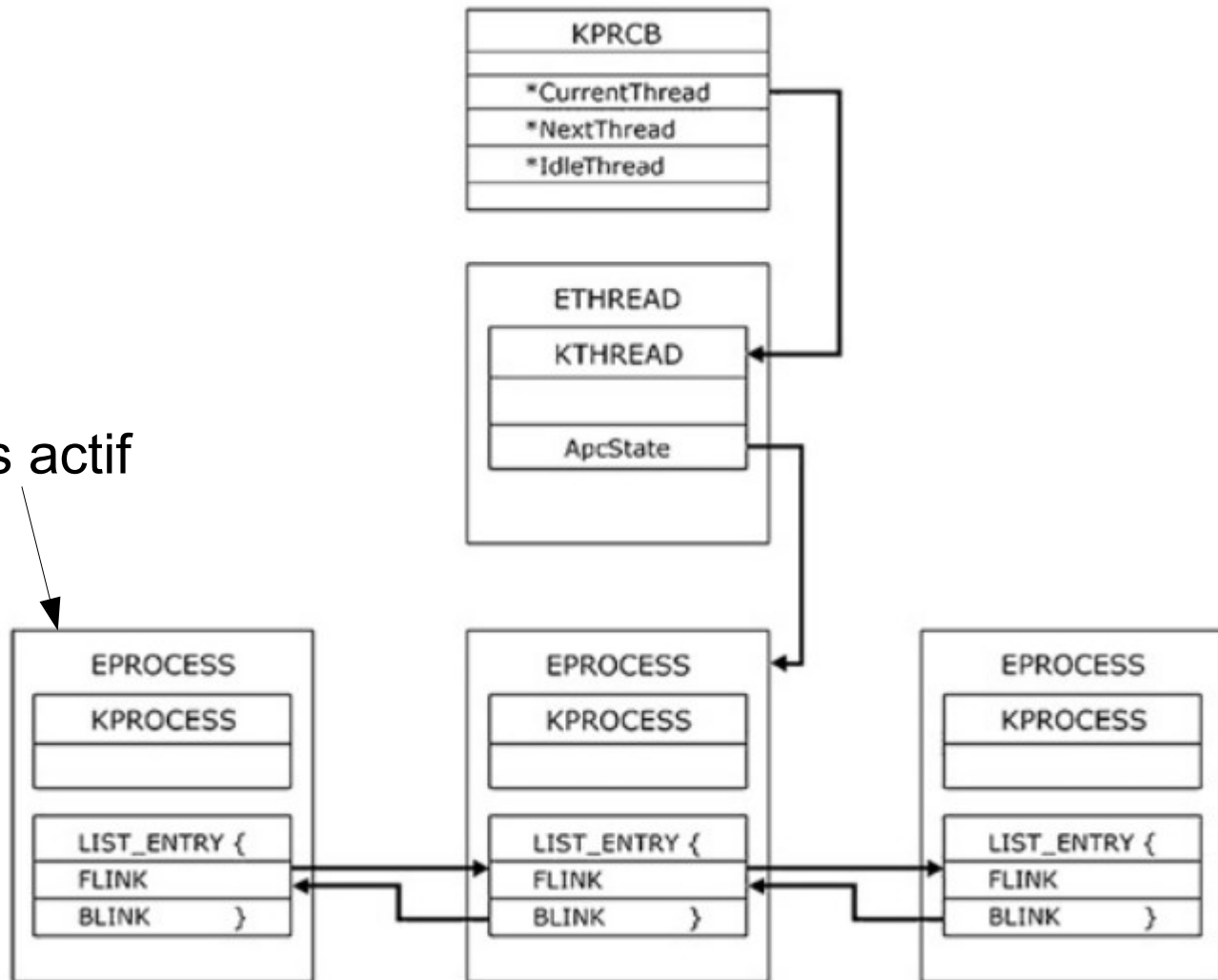
Direct Kernel Object Manipulation

DKOM : Spécificités

- Type d'attaque difficile à détecter
- Limitée → seul les objets pris en mémoire peuvent être manipulés (Ex: donc pas possible de cacher des fichiers)
- Ce qu'on peut faire :
 - Cacher des processus
 - Cacher des Kernel drivers
 - Cacher des ports
 - Élever le privilège d'un thread, d'un processus
 - Complexifier l'analyse forensic

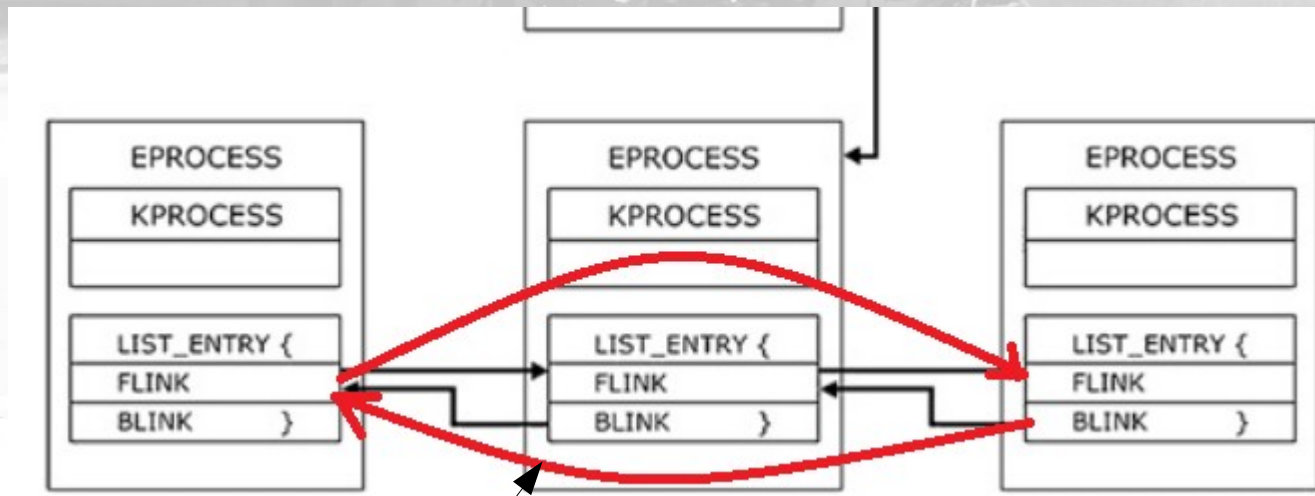
Cacher un processus avec DKOM (1)

Processus actif



Cacher un processus avec DKOM (2)

- Les phases :
 - Parcourir les structures jusqu'à EPROCESS (en suivant les pointeurs)
 - Trouver le PID qui nous intéresse dans EPROCESS



Ce qu'on veut faire disparaître

Références

- Rootkits – Subverting the windows kernel (Greg Hoggund & James Butler)
- « SHADOW WALKER » Raising The Bar For Rootkit Detection (Sherri Sparks & James Butler)
- Eye Digital Security White Paper – Remote Windows Kernel Exploitation
- Infond par t0ka7a
- Blog d'0verc0ck
- MSDN
- GoRing0 par Emilien Girault
- Blog de Ivanlef0u