

# Linux vs rootkits



- Adam 'pi3' Zabrocki



`/usr/bin/whoami`

- ADAM ZABROCKI

/usr/bin/whoami

- ADAM ZABROCKI



`/usr/bin/whoami`

- ADAM ZABROCKI



IT Security Reseracher  
HISPASEC



`/usr/bin/whoami`

- ADAM ZABROCKI



IT Security Reseracher  
HISPASEC



`/usr/bin/whoami`

- ADAM ZABROCKI



IT Security Reseracher  
HISPASEC



`/usr/bin/whoami`

- ADAM ZABROCKI



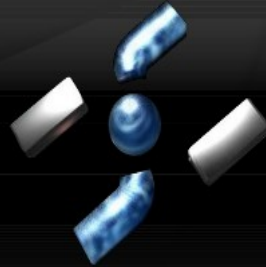
`/usr/bin/whoami`



• ADAM ZABROCKI



IT Security Reseracher  
HISPASEC



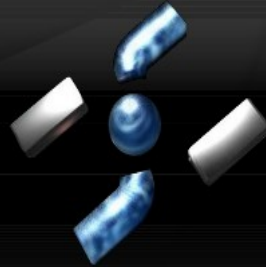
`/usr/bin/whoami`



- ADAM ZABROCKI



IT Security Reseracher  
HISPASEC



- <http://pi3.com.pl>
- [pi3@itsec.pl](mailto:pi3@itsec.pl) (lub oficjalnie [zabrocki@cern.ch](mailto:zabrocki@cern.ch))

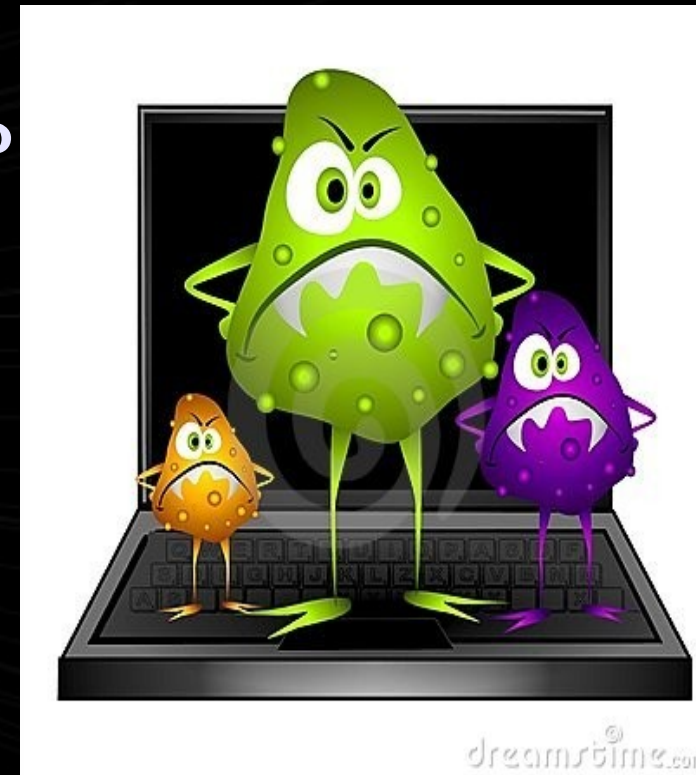
## Plan prezentacji:

- Wstęp (virus / malware / rootkit)
- Współczesne rootkit'y w skrócie
- Rootkity:
  - User Space
  - Kernel Space
    - LKM
    - Non-LKM
- Debug Register Rootkit
- Black Market...

Wstep:

## Wirus komputerowy:

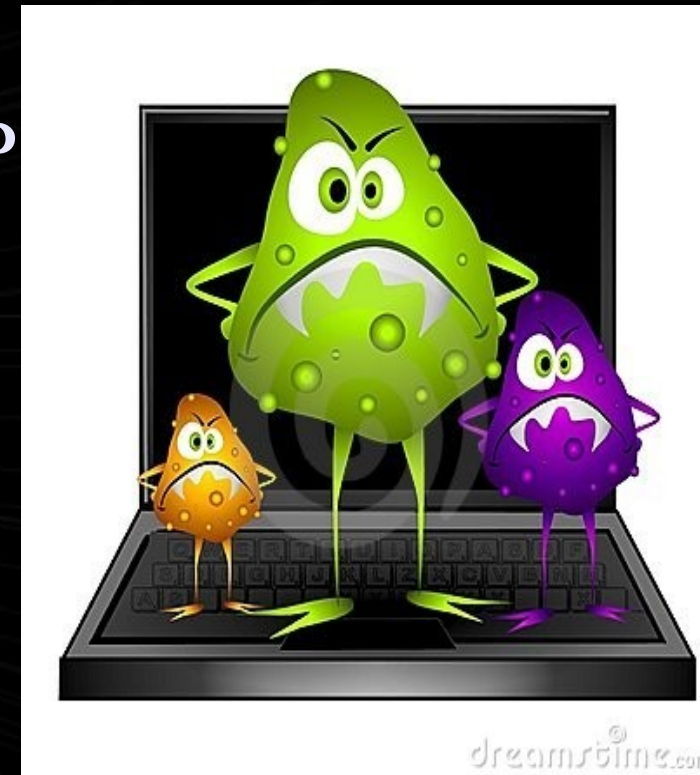
- 1970/71 – Creeper (TeneX OS) – DEC PDP



## Wstęp:

### Wirus komputerowy:

- 1970/71 – Creeper (TeneX OS) – DEC PDP
- Powielanie się...
- MUSI mieć nosiciela
- Słabość systemów...
- ... bądź beztroskę użytkowników



Wstęp:

Robak komputerowy:

- 2 listopada 1988 -  
“pogromca” UNIX'ów



## Wstęp:

### Robak komputerowy:

- 2 listopada 1988 -  
“pogromca” UNIX'ów
- Samoreplikujący się!
- Zazwyczaj wirus +  
dodatki (backdoor, koń  
trojański, itp.)



Wstęp:

Rootkit:

- User “root” + narzędzia “kit” = “rootkit”



## Wstęp:

### Rootkit:

- User “root” + narzędzia “kit” = “rootkit”
- Ukrywa procesy
- Ukrywa pliki
- Ukrywa połączenia sieciowe
- Backdoor
- (Key/Traffic/\*) Logger
- Ukrywa siebie...



# Teoria vs praktyka



# Rootkit'y dzisiaj

- Wirus + Trojan + Rootkit + Malware połączone w jedność => potocznie zwane malware

## Rootkit'y dzisiaj

- Virus + Trojan + Rootkit + Malware połączone w jedność => potocznie zwane malware
- ... a jakie skojarzenia mają ludzie “z branży” ?? ;)

## Rootkit'y dzisiaj

- Wirus + Trojan + Rootkit + Malware połączone w jedność => potocznie zwane malware
- ... a jakie skojarzenia mają ludzie “z branży” ?? ;)

14:13 <@pi3> powiedzcie mi swoje randomowe skojarzenia - pierwsze, które przyszły na myśl

14:13 <@pi3> na słowo rootkit / wirus komputerowy

# Rootkit'y dzisiaj

- Odpowiedzi:

14:13 <@ducer> piotr bania

# Rootkit'y dzisiaj

- Odpowiedzi:

14:13 <@ducer> piotr bania

14:14 <@mpp> Michał podatność bućko

# Rootkit'y dzisiaj

- Odpowiedzi:

14:13 <@ducer> piotr bania

14:14 <@mpp> Michał podatność bućko

14:14 <@bz> ./kodzik ;]

## Rootkit'y dzisiaj

- Odpowiedzi:

14:13 <@ducer> piotr bania

14:14 <@mpp> Michał podatność bućko

14:14 <@bz> ./kodzik ;]

14:14 <@Gynvael> na rootkit: subverting  
windows kernel

14:14 <@Gynvael> na wirus: mks\_vir

# Rootkit'y dzisiaj

- Co dalej? :)



# Rootkit'y dzisiaj

- Co dalej? :)

14:15 <@Gynvael> to pierwszy (i ostatni)  
AV ktorego uzywalem

14:15 <@Gynvael> ;>

# Rootkit'y dzisiaj

- Co dalej? :)

14:15 <@Gynvael> to pierwszy (i ostatni)  
AV ktorego uzywalem

14:15 <@Gynvael> ;>

14:15 <@mpp> a co teraz masz ? :>

## Rootkit'y dzisiaj

- Co dalej? :)

14:15 <@Gynvael> to pierwszy (i ostatni)  
AV ktorego uzywalem

14:15 <@Gynvael> ;>

14:15 <@mpp> a co teraz masz ? :>

14:16 <@Gynvael> teraz? teraz mam IDA i  
OllyDbg

14:16 <@Gynvael> ;p

## Rootkit'y dzisiaj

- Co dalej? :)

14:15 <@Gynvael> to pierwszy (i ostatni)  
AV ktorego uzywalem

14:15 <@Gynvael> ;>

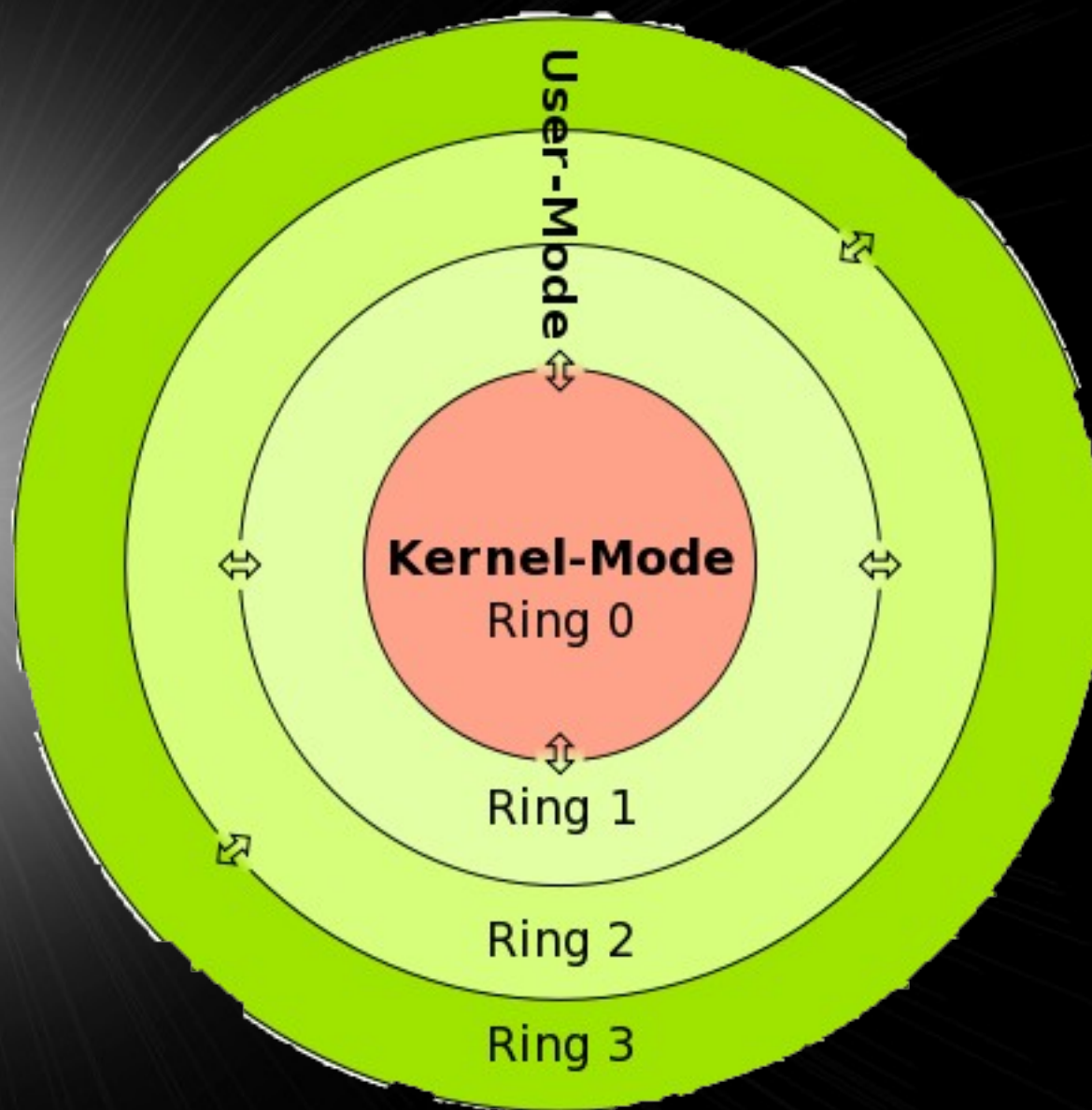
14:15 <@mpp> a co teraz masz ? :>

14:16 <@Gynvael> teraz? teraz mam IDA i  
OllyDbg

14:16 <@Gynvael> ;p

14:16 <@pi3> az zapisze te logi i pokaze  
na SecDay ;]

# Rootkit'y:



# Rootkit'y:

## User Space rootkits:

- Ukrywa procesy
- Ukrywa pliki
- Ukrywa połączenia sieciowe
- Backdoor
- (Key/Traffic/\*) Logger
- Ukrywa siebie...



# Rootkit'y:

## User Space rootkits:

- Ukrywa procesy (ps/... )
- Ukrywa pliki
- Ukrywa połączenia sieciowe
- Backdoor
- (Key/Traffic/\*) Logger
- Ukrywa siebie...



# Rootkit'y:

## User Space rootkits:

- Ukrywa procesy (ps/... )
- Ukrywa pliki (ls/dir/...)
- Ukrywa połączenia sieciowe
- Backdoor
- (Key/Traffic/\*) Logger
- Ukrywa siebie...



# Rootkit'y:

## User Space rootkits:

- Ukrywa procesy (ps/... )
- Ukrywa pliki (ls/dir/...)
- Ukrywa połączenia sieciowe (netstat/...)
- Backdoor
- (Key/Traffic/\*) Logger
- Ukrywa siebie...



# Rootkit'y:

## User Space rootkits:

- Ukrywa procesy (ps/... )
- Ukrywa pliki (ls/dir/...)
- Ukrywa połączenia sieciowe (netstat/...)
- Backdoor (ssh[d]/ftp/...)
- (Key/Traffic/\*) Logger
- Ukrywa siebie...



# User Space Rootkits:

Zaity:

- Prostota



# User Space Rootkits:

## Zalety:

- Prostota
- “Bezpieczeństwo”



# User Space Rootkits:

## Zalety:

- Prostota
- “Bezpieczeństwo”
- Stabilność



# User Space Rootkits:

## Zalety:

- Prostota
- “Bezpieczeństwo”
- Stabilność
- Przenośność



# User Space Rootkits:

## Wady:

- Bardzo proste do wykrycia (checksum/'czyste' narzędzia/...)



# User Space Rootkits:

## Wady:

- Bardzo proste do wykrycia (checksum/'czyste' narzędzia/...)
- Wiele programów



# User Space Rootkits:

## Wady:

- Bardzo proste do wykrycia (checksum/'czyste' narzędzia/...)
- Wiele programów
- Upgrade może zabić



# User Space Rootkits:

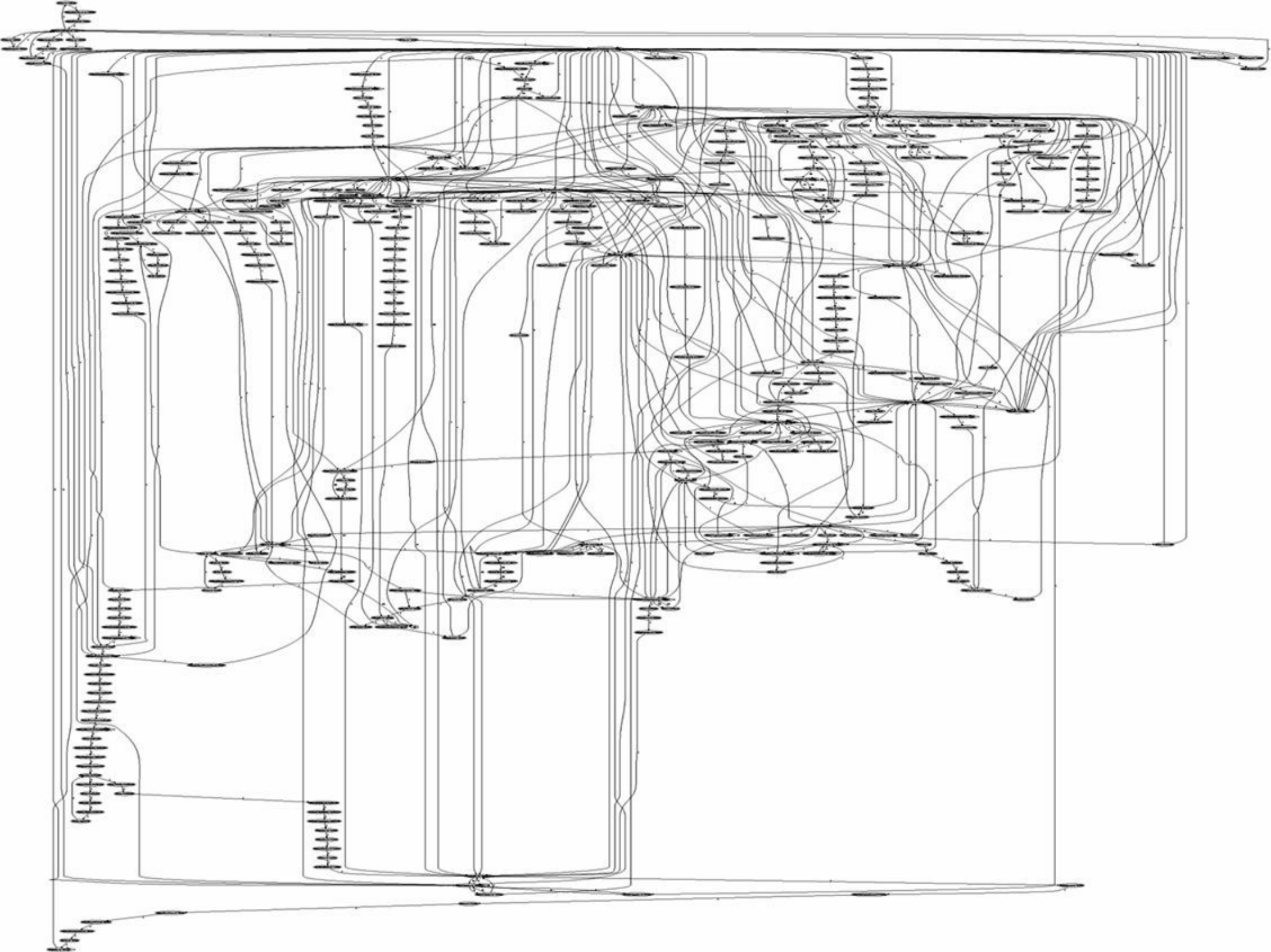
## Wady:

- Bardzo proste do wykrycia  
(checksum/'czyste'  
narzędzia/...)
- Wiele programów
- Upgrade może zabić
- Mało edukacyjne ;)



# Kernel Space Rootkits:

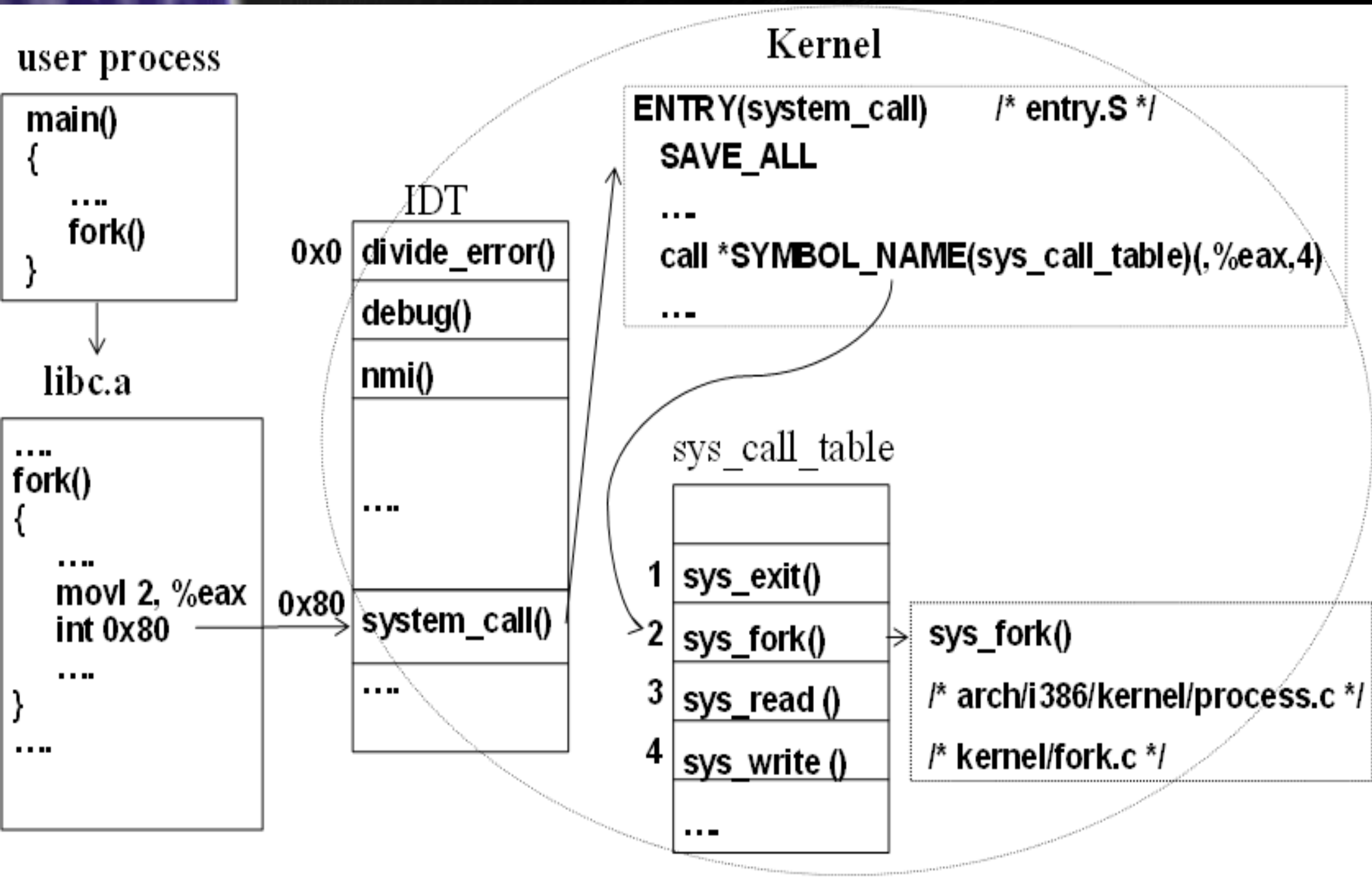
Analiza wywołania systemowego na  
przykładzie serwera Apache...



# Kernel Space Rootkits:

To był kiepski żart ;)

# Kernel Space Rootkits:



# Loadable Kernel Module Rootkits:

- Lekcja historii:

```
sys_call_table[__NR_open] = (void *)  
my_func_ptr;
```

# Loadable Kernel Module Rootkits:

- Lekcja historii:

```
sys_call_table[__NR_open] = (void *)  
my_func_ptr;
```

- *sys\_call\_table* nie jest już eksportowane

# Loadable Kernel Module Rootkits:

- Lekcja historii:

```
sys_call_table[__NR_open] = (void *)  
my_func_ptr;
```

- *sys\_call\_table* nie jest już eksportowane
- Zczytać:

```
[root@pi3book files]# cat /proc/kallsyms |grep  
sys_call_table
```

```
c0732320 R sys_call_table
```

```
[root@pi3book files]#
```

# Loadable Kernel Module Rootkits:

- Lekcja historii:

```
sys_call_table[__NR_open] = (void *)  
my_func_ptr;
```

- *sys\_call\_table* nie jest już eksportowane
- Zczytać:

```
[root@pi3book files]# cat /proc/kallsyms |grep  
sys_call_table
```

```
c0732320 R sys_call_table
```

```
[root@pi3book files]#
```

# Loadable Kernel Module Rootkits:

- Opcje

*CONFIG\_KALLSYMS=y*

*CONFIG\_KALLSYMS\_ALL=y*

*CONFIG\_KALLSYMS\_EXTRA\_PASS=y*

# Loadable Kernel Module Rootkits:

- Opcje

*CONFIG\_KALLSYMS=y*

*CONFIG\_KALLSYMS\_ALL=y*

*CONFIG\_KALLSYMS\_EXTRA\_PASS=y*

- “arch/x86/kernel/entry\_32.S”:

*530:ENTRY(system\_call):*

*...*

*542:     call \***sys\_call\_table**(,%eax,4)*

# Loadable Kernel Module Rootkits:

- Gdb...

```
0xc040352e:  call  *-0x3f8cdce0(,%eax,4)
```

```
0xc0403535:  mov   %eax,0x18(%esp)
```

```
(gdb) p/x 0xc0403535-0xc040352e
```

```
$1 = 0x7
```

```
(gdb) x/2x 0xc040352e
```

```
0xc040352e:  0x208514ff 0x89c07323
```

```
(gdb)
```

# Loadable Kernel Module Rootkits:

- Gdb...

```
0xc040352e:  call  *-0x3f8cdce0(,%eax,4)
```

```
0xc0403535:  mov   %eax,0x18(%esp)
```

```
(gdb) p/x 0xc0403535-0xc040352e
```

```
$1 = 0x7
```

```
(gdb) x/2x 0xc040352e
```

```
0xc040352e:  0x208514ff 0x89c07323
```

```
(gdb)
```

# Loadable Kernel Module Rootkits:

- Gdb...

```
0xc040352e:  call  *-0x3f8cdce0(,%eax,4)
```

```
0xc0403535:  mov   %eax,0x18(%esp)
```

```
(gdb) p/x 0xc0403535-0xc040352e
```

```
$1 = 0x7
```

```
(gdb) x/2x 0xc040352e
```

```
0xc040352e:  0x208514ff 0x89c07323
```

```
(gdb)
```

# Loadable Kernel Module Rootkits:

- Gdb...

```
0xc040352e:  call  *-0x3f8cdce0(,%eax,4)
```

```
0xc0403535:  mov   %eax,0x18(%esp)
```

```
(gdb) p/x 0xc0403535-0xc040352e
```

```
$1 = 0x7
```

```
(gdb) x/2x 0xc040352e
```

```
0xc040352e:  0x208514ff 0x89c07323
```

```
(gdb)
```

# Loadable Kernel Module Rootkits:

- Gdb...

```
0xc040352e:  call  *-0x3f8cdce0(,%eax,4)
```

```
0xc0403535:  mov   %eax,0x18(%esp)
```

```
(gdb) p/x 0xc0403535-0xc040352e
```

```
$1 = 0x7
```

```
(gdb) x/2x 0xc040352e
```

```
0xc040352e:  0x208514ff 0x89c07323
```

```
(gdb)
```

# Loadable Kernel Module Rootkits:

- Gdb...

```
0xc040352e:  call  *-0x3f8cdce0(,%eax,4)
```

```
0xc0403535:  mov   %eax,0x18(%esp)
```

```
(gdb) p/x 0xc0403535-0xc040352e
```

```
$1 = 0x7
```

```
(gdb) x/2x 0xc040352e
```

```
0xc040352e:  0x208514ff 0x89c07323
```

```
(gdb)
```

# Loadable Kernel Module Rootkits:

- Gdb...

```
0xc040352e:  call  *-0x3f8cdce0(,%eax,4)
```

```
0xc0403535:  mov   %eax,0x18(%esp)
```

```
(gdb) p/x 0xc0403535-0xc040352e
```

```
$1 = 0x7
```

```
(gdb) x/2x 0xc040352e
```

```
0xc040352e:  0x208514ff  0x89c07323
```

```
(gdb)
```

```
c0732320 R sys_call_table
```

# Loadable Kernel Module Rootkits:

- Gdb...

```
0xc040352e:  call  *-0x3f8cdce0(,%eax,4)
```

```
0xc0403535:  mov   %eax,0x18(%esp)
```

```
(gdb) p/x 0xc0403535-0xc040352e
```

```
$1 = 0x7
```

```
(gdb) x/2x 0xc040352e
```

```
0xc040352e:  0x208514ff 0x89c07323
```

```
(gdb)
```

```
c0732320 R sys_call_table
```

# Loadable Kernel Module Rootkits:

- Skąd wziąć adres początkowy? (system\_call)

## Loadable Kernel Module Rootkits:

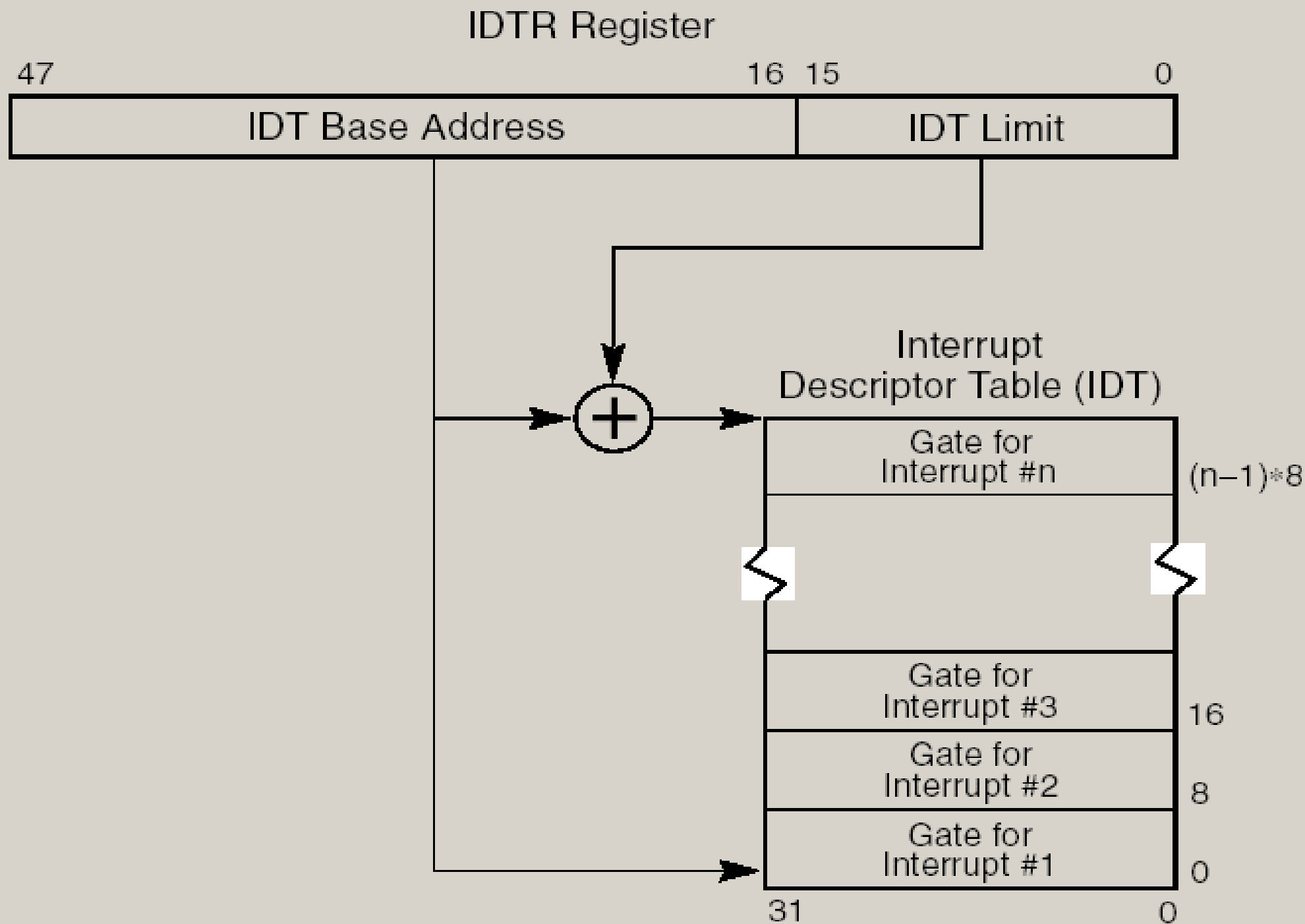
- Skąd wziąć adres początkowy? (system\_call)
- “arch/i386/kernel/traps.c:889”  
set\_system\_trap\_gate(SYSCALL\_VECTOR,  
&system\_call);
- x86 => tablica przerwań (IDT)

## Loadable Kernel Module Rootkits:

- Skąd wziąć adres początkowy? (system\_call)
- “arch/i386/kernel/traps.c:889”  
set\_system\_trap\_gate(SYSCALL\_VECTOR,  
&system\_call);
- x86 => tablica przerwań (IDT)
- Skąd wziąć adres tablicy IDT?

## Loadable Kernel Module Rootkits:

- Skąd wziąć adres początkowy? (system\_call)
- “arch/i386/kernel/traps.c:889”  
set\_system\_trap\_gate(SYSCALL\_VECTOR,  
&system\_call);
- x86 => tablica przerwań (IDT)
- Skąd wziąć adres tablicy IDT? => instrukcja assemblera 'sidt'



**Figure 5-1. Relationship of the IDTR and IDT**

# Loadable Kernel Module Rootkits:

- Plan:
  - 'sidt'

# Loadable Kernel Module Rootkits:

- Plan:
  - 'sidt'
  - 0x80 wpis w tablicy IDT => *system\_call*

# Loadable Kernel Module Rootkits:

- Plan:
  - 'sidt'
  - 0x80 wpis w tablicy IDT => *system\_call*
  - Szukamy 'naszej' instrukcji (fingerprint)

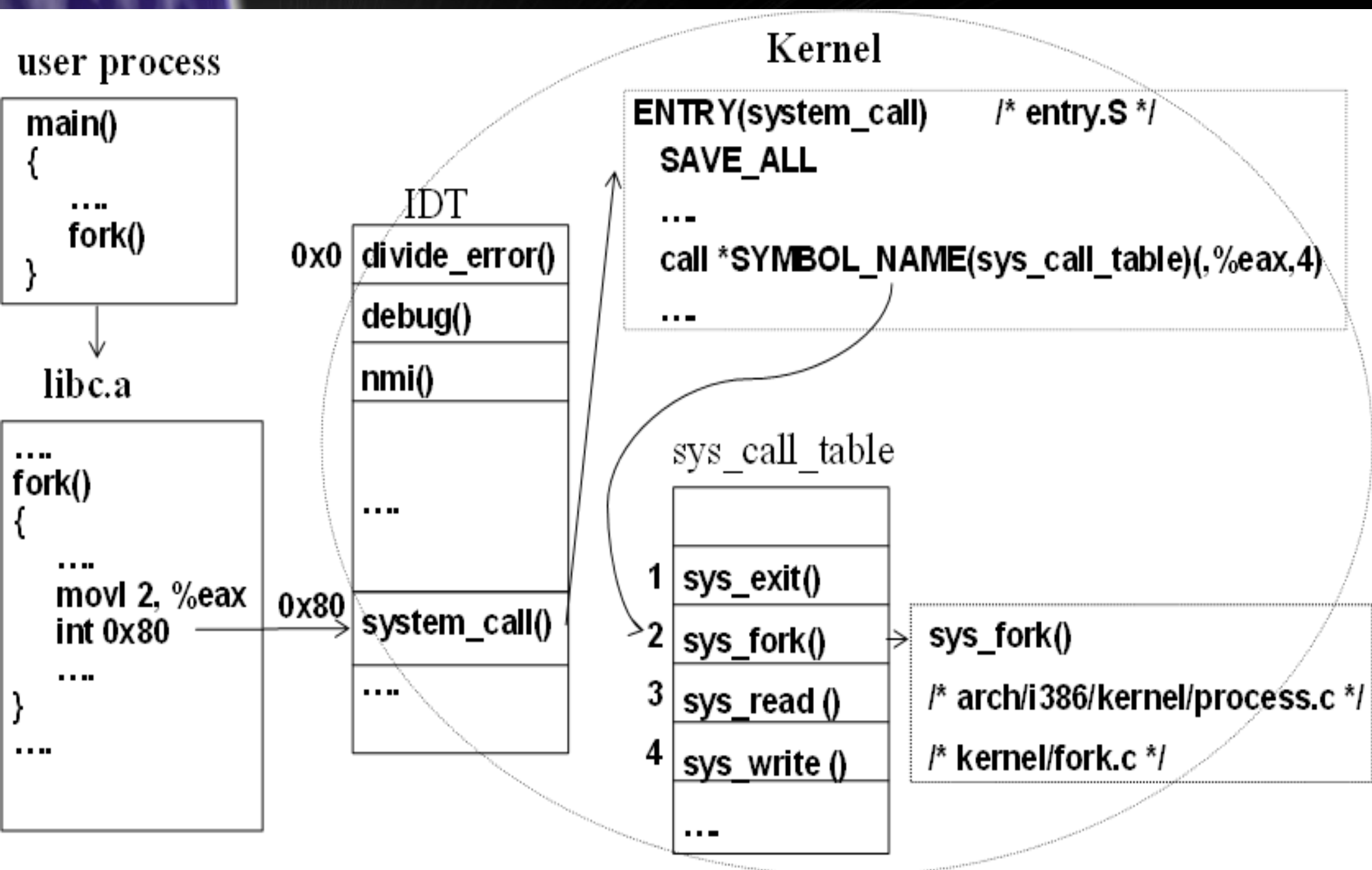
# Loadable Kernel Module Rootkits:

- Plan:
  - 'sidt'
  - 0x80 wpis w tablicy IDT => *system\_call*
  - Szukamy 'naszej' instrukcji (fingerprint)
  - Odkodowujemy adres *sys\_call\_table*

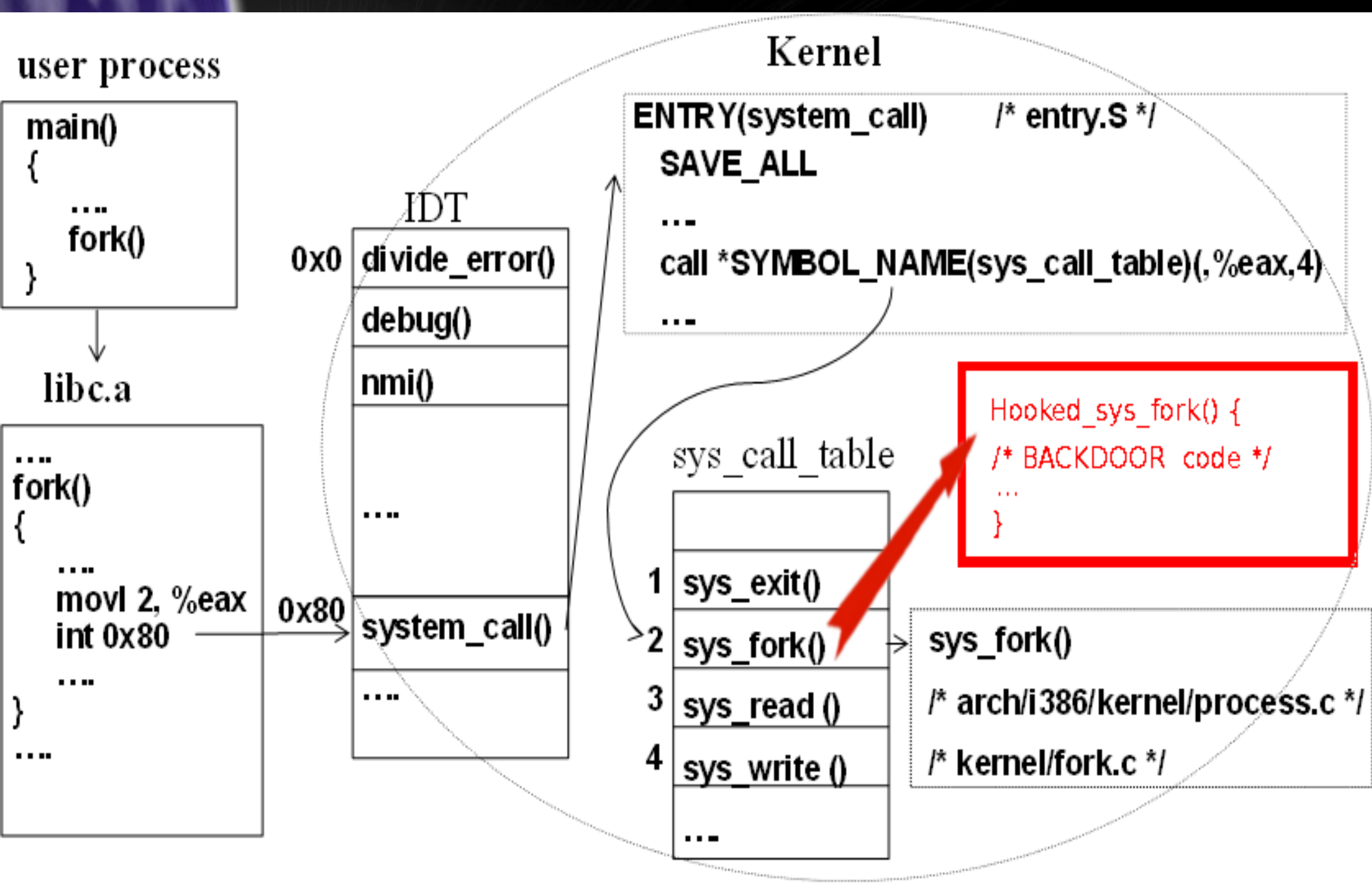
# Loadable Kernel Module Rootkits:

- Plan:
  - 'sidt'
  - 0x80 wpis w tablicy IDT => *system\_call*
  - Szukamy 'naszej' instrukcji (fingerprint)
  - Odkodowujemy adres *sys\_call\_table*
  - Voila! ;)

# Loadable Kernel Module Rootkits:



# Loadable Kernel Module Rootkits:



# Loadable Kernel Module Rootkits:

... ale:

- Od wersji 2.6.24  $\geq$  sys\_call\_table RO

# Loadable Kernel Module Rootkits:

... ale:

- Od wersji 2.6.24  $\geq$  sys\_call\_table RO
- ... więc zmieniamy atrybut strony na RW

# Loadable Kernel Module Rootkits:

... ale:

- Od wersji 2.6.24  $\geq$  sys\_call\_table RO
- ... więc zmieniamy atrybut strony na RW
- Od wersji 2.6.27.8  $\geq$  zmieniony mechanizm zmiany atrybutów strony ;)

# Loadable Kernel Module Rootkits:

... ale:

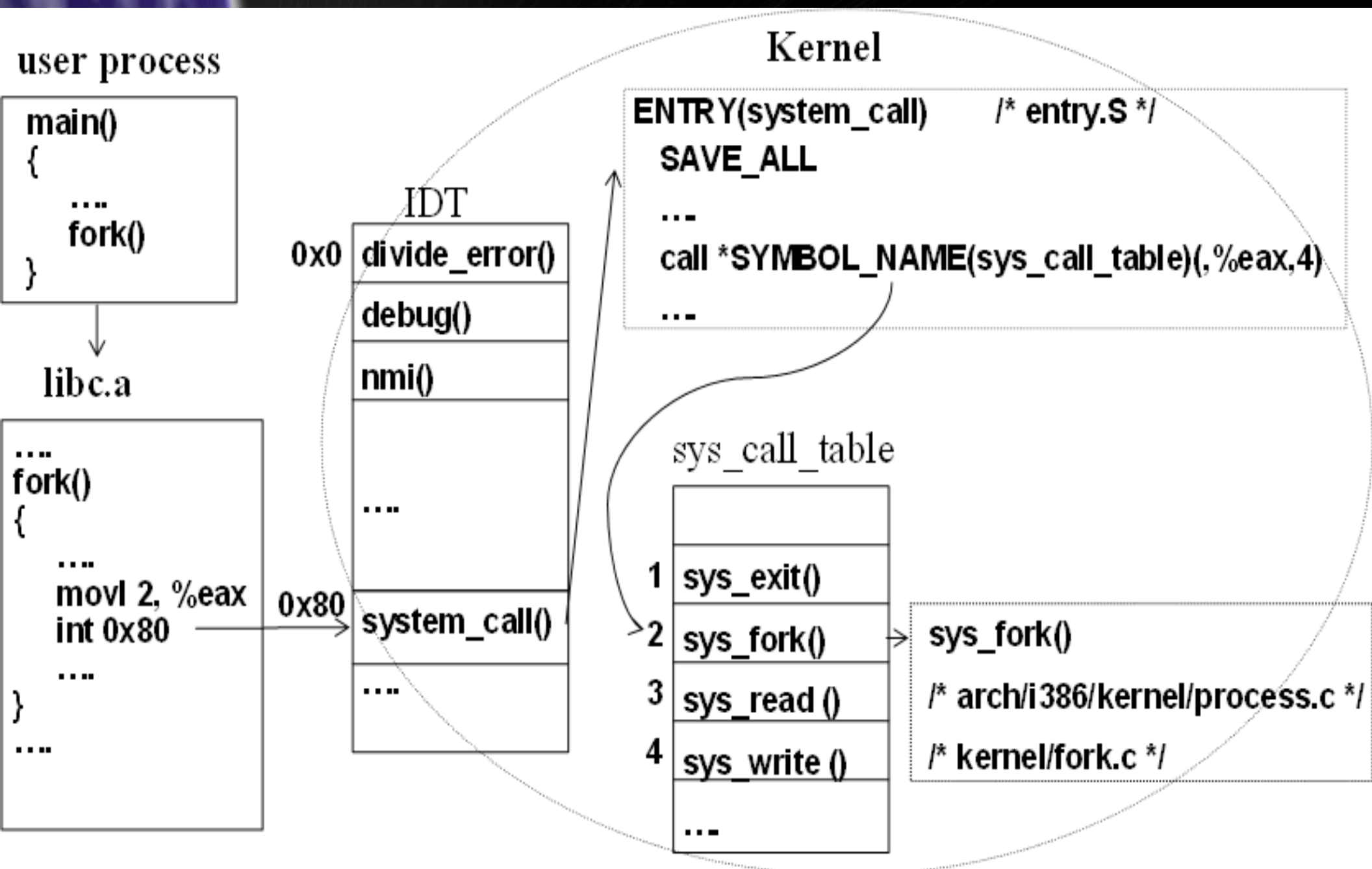
- Od wersji 2.6.24  $\geq$  sys\_call\_table RO
- ... więc zmieniamy atrybut strony na RW
- Od wersji 2.6.27.8  $\geq$  zmieniony mechanizm zmiany atrybutów strony ;)
- Nadal da się to zrobić, ale to bardzo kłopotliwe

# Loadable Kernel Module Rootkits:

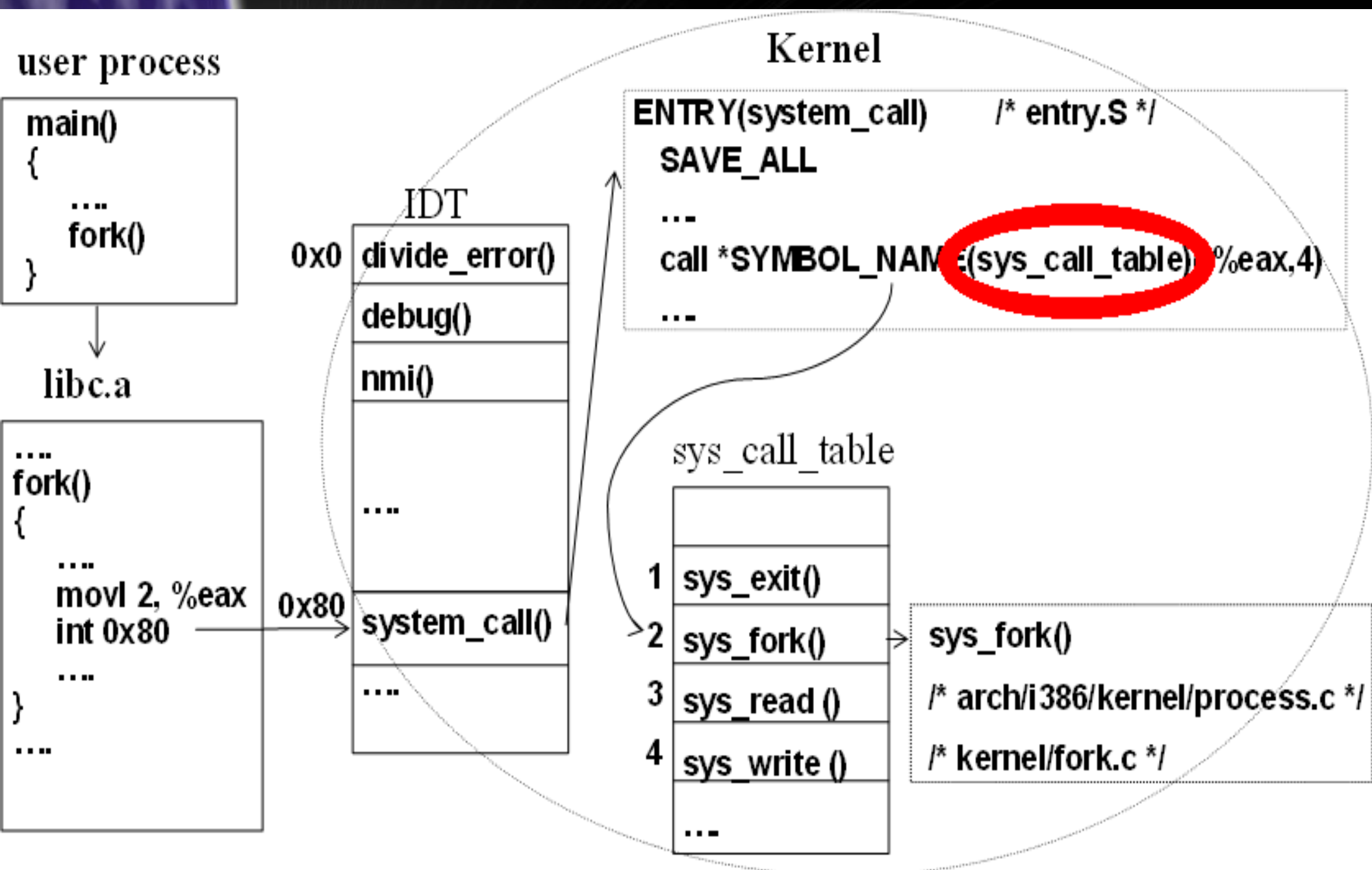
... ale:

- Od wersji 2.6.24  $\geq$  sys\_call\_table RO
- ... więc zmieniamy atrybut strony na RW
- Od wersji 2.6.27.8  $\geq$  zmieniony mechanizm zmiany atrybutów strony ;)
- Nadal da się to zrobić, ale to bardzo kłopotliwe
- Można sobie życie uprościć...

# Loadable Kernel Module Rootkits:



# Loadable Kernel Module Rootkits:



# Loadable Kernel Module Rootkits:

Co złego zrobiliśmy?

- Adres wejścia do *sys\_call\_table*

# Loadable Kernel Module Rootkits:

## Co złego zrobiliśmy?

- Adres wejścia do *sys\_call\_table*
- OMG! Kontrolujemy wszystkie wywołania systemowe

# Loadable Kernel Module Rootkits:

## Co złego zrobiliśmy?

- Adres wejścia do *sys\_call\_table*
- OMG! Kontrolujemy wszystkie wywołania systemowe
- ... wystarczy dobrze zaprojektować “własną” *sys\_call\_table* ;)

# Loadable Kernel Module rootkits podsumowanie:

## Zalety:

- Modyfikujemy TYLKO kernel

# Loadable Kernel Module rootkits podsumowanie:

## Zalety:

- Modyfikujemy TYLKO kernel
- Jesteśmy ograniczeni tylko wyobraźnią...  
i wiedzą (zawsze 'coś' się znajdzie ;))

# Loadable Kernel Module rootkits podsumowanie:

## Zalety:

- Modyfikujemy TYLKO kernel
- Jesteśmy ograniczeni tylko wyobraźnią... i wiedzą (zawsze 'coś' się znajdzie ;))

## Wady:

- Łatwo zniszczyć system

# Loadable Kernel Module rootkits podsumowanie:

## Zalety:

- Modyfikujemy TYLKO kernel
- Jesteśmy ograniczeni tylko wyobraźnią... i wiedzą (zawsze 'coś' się znajdzie ;))

## Wady:

- Łatwo zniszczyć system
- Mały błąd może zaburzyć działanie aplikacji

# Loadable Kernel Module rootkits

## podsumowanie:

### Zalety:

- Modyfikujemy TYLKO kernel
- Jesteśmy ograniczeni tylko wyobraźnią... i wiedzą (zawsze 'coś' się znajdzie ;))

### Wady:

- Łatwo zniszczyć system
- Mały błąd może zaburzyć działanie aplikacji
- “Słaby projekt” backdoora może skutkować wywołaniem w nieodpowiednim momencie

# Non LKM Rootkits:

- Czemu?

# Non LKM Rootkits:

- Czemu?
- Co zyskujemy?

## Non LKM Rootkits:



## Non LKM Rootkits:

- Czemu?
- Co zyskujemy?
- Jak działają?
  - /dev/[k]mem

## Non LKM Rootkits:

- Czemu?
- Co zyskujemy?
- Jak działają?
  - /dev/[k]mem
  - Statyczne łatanie jądra (vmlinuz)

## Non LKM Rootkits:

- Czemu?
- Co zyskujemy?
- Jak działają?
  - /dev/[k]mem
  - Statyczne łatanie jądra (vmlinuz)
  - Bootkit'y (np. Grub)

## Non LKM Rootkits:

- Czemu?
- Co zyskujemy?
- Jak działają?
  - /dev/[k]mem
  - Statyczne łatanie jądra (vmlinuz)
  - Bootkit'y (np. Grub)
  - BIOS rootkit

## Non LKM Rootkits:

### Zalety:

- Bardzo ciężkie do wykrycia
- Funkcjonalność taka sama jak w przypadku LKM

### Wady:

- Trzeba pomyśleć co po restarcie ;)
- Zmiana kernel'a może być krytyczna dla rootkit'a
- Nieprzenośne i trudne w pisaniu

Wykrywanie?



# Wykrywanie?

- No właśnie jak?
  - Porównywanie adresów
  - Porównywanie wyników
  - Każdy rootkit ma swoją własną “charakterystykę”

# Wykrywanie?

- No właśnie jak?
  - Porównywanie adresów
  - Porównywanie wyników
  - Każdy rootkit ma swoją własną “charakterystykę”
- Protekcja?:
  - Monitorowanie modułów
  - Nowe jądra zabraniają pisanie do /dev/[k]mem
  - Backup adresów i monitorowanie pamięci ;)

## Co dalej?

- Problemy:
  - Problemy z podmianą *sys\_call\_table*
  - Jądro ochrania *sys\_call\_table*
  - Moduły łatwo wykryć i się zabezpieczyć
  - Brak możliwości pisania do `/dev/[k]mem`
  - 'vmlinuz' prosto monitorować

# Debug Register Rootkit!



# Debug Register Rootkit!

- Debugować można software'owo...

# Debug Register Rootkit!

- Debugować można software'owo...
- ... ale architektura x86 ma wsparcie sprzętowe

# Debug Register Rootkit!

- Debugować można software'owo...
- ... ale architektura x86 ma wsparcie sprzętowe
- Ile osób na sali o tym wiedziało? :)

# Debug Register Rootkit!

- Debugować można software'owo...
- ... ale architektura x86 ma wsparcie sprzętowe
- Ile osób na sali o tym wiedziało? :)
- Rejestry DR\*:
  - DR[0..3] – breakpoint
  - DR6 – modyfikowany w czasie wystąpienia breakpoint'a
  - DR7 – konfigurowanie kiedy ma być wywołane przerwanie

# Debug Register Rootkit!

- Co możemy zrobić?
  - Breakpoint na *system\_call* i *sysenter\_entry*

# Debug Register Rootkit!

- Co możemy zrobić?
  - Breakpoint na *system\_call* i *sysenter\_entry*
  - Wygenerowanie przerwania 1

# Debug Register Rootkit!

- Co możemy zrobić?
  - Breakpoint na *system\_call* i *sysetner\_entry*
  - Wygenerowanie przerwania 1
  - Wpis w IDT wskazuje na funkcję *do\_debug...* podmieńmy ją!

# Debug Register Rootkit!

- Co możemy zrobić?
  - Breakpoint na *system\_call* i *sysetner\_entry*
  - Wygenerowanie przerwania 1
  - Wpis w IDT wskazuje na funkcje *do\_debug...* podmieńmy ją!
  - OMG! Znowu przejęliśmy kontrolę nad wszystkimi przerwaniem ;)

# Debug Register Rootkit!

## Zalety:

- Bardzo “cichy”
- Bardzo trudny do wykrycia
- Cały czas jesteśmy w przestrzeni jądra ;)

# Debug Register Rootkit!

## Zalety:

- Bardzo “cichy”
- Bardzo trudny do wykrycia
- Cały czas jesteśmy w przestrzeni jądra ;)

## Wady:

- Zaawansowany i trudny do napisania
- Działa TYLKO na architekturze x86

# Debug Register Rootkit!

Jak wykryć?

- Monitorować rejestry procesora DR\*...

# Debug Register Rootkit!

## Jak wykryć?

- Monitorować rejestry procesora DR\*...
- Konkluzja... ktoś ma prawa root?

# Debug Register Rootkit!

Jak wykryć?

- Monitorować rejestry procesora DR\*...
- Konkluzja... ktoś ma prawa root?

# GAME OVER!!!

Black Market...



# Black Market...

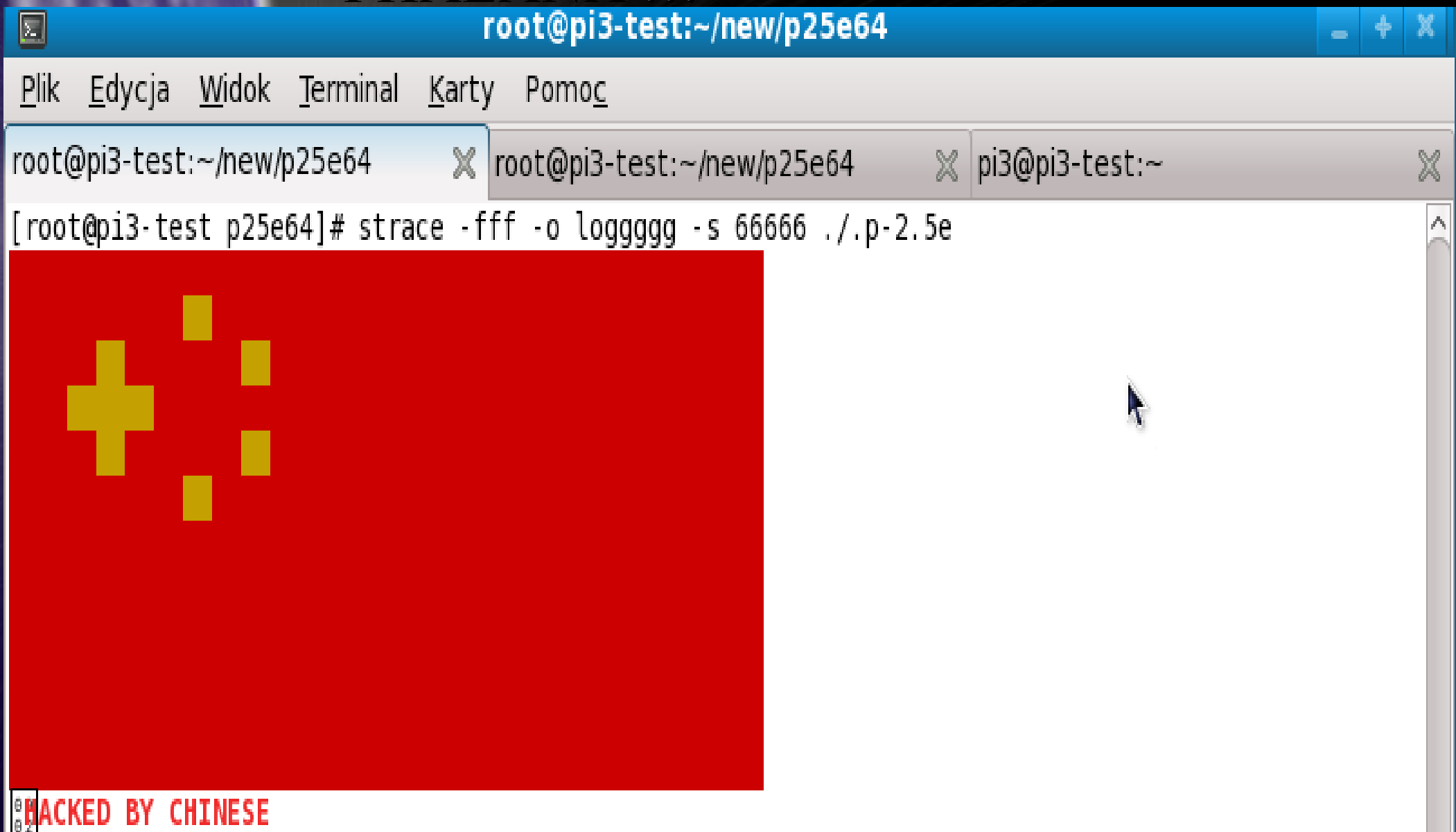


# Black Market...



# Black Market...

PHALANX !!!



# Black Market...

PHALANX !!!

```
[root@pi3-test p25e64]# strace -fff -o logggggg -s 66666 ./p-2.5e i
```

```
(_- phalanx 2.5e -_)
```

```
; mmap failed..bypassing /dev/mem restrictions
```

```
; locating sys_call_table..
```

```
; sys_call_table_phys = 0x5e6270
```

```
; phys_base = 0xe00000
```

```
; sys_call_table = 0xffffffff813e6270
```

```
; hooking.. [8=====D]
```

# Black Market...

PHALANX !!! - charakterystyka:

- Oficjalnie nie istnieje

# Black Market...

PHALANX !!! - charakterystyka:

- Oficjalnie nie istnieje
- Dostępne źródła są dla wersji phalanx 1 beta 6 (z roku 2005)

## Black Market...

PHALANX !!! - charakterystyka:

- Oficjalnie nie istnieje
- Dostępne źródła są dla wersji phalanx 1 beta 6 (z roku 2005)
- “Świat” dowiedział się o ciągłym rozwijaniu rookita w 2008 – automat do zabijania – SSH keys

## Black Market...

PHALANX !!! - charakterystyka:

- Oficjalnie nie istnieje
- Dostępne źródła są dla wersji phalanx 1 beta 6 (z roku 2005)
- “Świat” dowiedział się o ciągłym rozwijaniu rookita w 2008 – automat do zabijania – SSH keys
- Podobno 2.4 jest ostatnim wypustem...

## Black Market...

PHALANX !!! - charakterystyka:

- Oficjalnie nie istnieje
- Dostępne źródła są dla wersji phalanx 1 beta 6 (z roku 2005)
- “Świat” dowiedział się o ciągłym rozwijaniu rookita w 2008 – automat do zabijania – SSH keys
- Podobno 2.4 jest ostatnim wypustem... a ostatnią wersją którą analizowałem była 2.5e ;)

# Black Market...

PHALANX !!! - charakterystyka:

- Własny protokół komunikacji

# Black Market...

PHALANX !!! - charakterystyka:

- Własny protokół komunikacji
- Co najmniej “podwójne” szyfrowanie

# Black Market...

PHALANX !!! - charakterystyka:

- Własny protokół komunikacji
- Co najmniej “podwójne” szyfrowanie
- Omija restrykcje związane z urządzeniem /dev/[k]mem

# Black Market...

PHALANX !!! - charakterystyka:

- Własny protokół komunikacji
- Co najmniej “podwójne” szyfrowanie
- Omija restrykcje związane z urządzeniem /dev/[k]mem
- Przejmuje “prawie” wszystkie wywołania systemowe oraz uchwytów sygnałów

# Black Market...

PHALANX !!! - charakterystyka:

- Własny protokół komunikacji
- Co najmniej “podwójne” szyfrowanie
- Omija restrykcje związane z urządzeniem /dev/[k]mem
- Przejmuje “prawie” wszystkie wywołania systemowe oraz uchwytów sygnałów
- Doskonale się ukrywa

# Black Market...

PHALANX !!! - charakterystyka:

- Własny protokół komunikacji
- Co najmniej “podwójne” szyfrowanie
- Omija restrykcje związane z urządzeniem /dev/[k]mem
- Przejmuje “prawie” wszystkie wywołania systemowe oraz uchwyty sygnałów
- Doskonale się ukrywa
- Ma opcję 'uninstall' ;) ...

# Black Market...

PHALANX !!! - charakterystyka:

- Własny protokół komunikacji
- Co najmniej “podwójne” szyfrowanie
- Omija restrykcje związane z urządzeniem /dev/[k]mem
- Przejmuje “prawie” wszystkie wywołania systemowe oraz uchwytów sygnałów
- Doskonale się ukrywa
- Ma opcję 'uninstall' ;) ... ale nie usuwa ona rootkit'a całkowicie!!!

# Black Market...

PHALANX !!! - charakterystyka:

- Ma błędy ;p

# Black Market...

PHALANX !!! - charakterystyka:

- Ma błędy ;p
  - Można wykryć rootkit'a – prosty błąd = czas do załatwienia

# Black Market...

PHALANX !!! - charakterystyka:

- Ma błędy ;p
  - Można wykryć rootkit'a – prosty błąd = czas do załatania
  - Ma remote stack overflowa pre-auth ;)

# Black Market...

PHALANX !!! - charakterystyka:

- Ma błędy ;p
  - Można wykryć rootkit'a – prosty błąd = czas do załatania
  - Ma remote stack overflowa pre-auth ;)
    - Możliwość zdalnego wykrycia zainfekowanych maszyn

# Black Market...

PHALANX !!! - charakterystyka:

- Ma błędy ;p
  - Można wykryć rootkit'a – prosty błąd = czas do załatania
  - Ma remote stack overflowa pre-auth ;)
    - Możliwość zdalnego wykrycia zainfekowanych maszyn
    - Możliwość napisania remote exploita na zainfekowane maszyny ;)

# Black Market...

PHALANX !!! - charakterystyka:

- Ma błędy ;p
  - Można wykryć rootkit'a – prosty błąd = czas do załatania
  - Ma remote stack overflowa pre-auth ;)
    - Możliwość zdalnego wykrycia zainfekowanych maszyn
    - Możliwość napisania remote exploita na zainfekowane maszyny ;)
- Są rzeczy, o których mało kto wie ;)

## Black Market...

Upgrade Debug Register Rootkit:

- Break point na samego siebie – naprawdę trudny do wykrycia! ;)



DZIĘKUJĘ ZA UWAGĘ !

