

# Rootkit Profiler LX

Overview and Documentation

Tobias Klein  
[www.trapkit.de](http://www.trapkit.de)

Version 0.12, 2007/04/22

## Table of contents

1	Overview .....	3
1.1	Features.....	3
1.2	Supported operating systems.....	3
2	Quick start guide.....	4
3	Rootkit Profiler Module.....	4
4	Rootkit Profiler Console.....	4
5	Interpretation of results .....	7
5.1	Example 1: Clean system.....	7
5.2	Example 2: Syscall address modification.....	10
5.3	Example 3: Syscall code modification .....	12
5.4	Example 4: VFS function pointer modification .....	14

# 1 Overview

Rootkit Profiler LX (RKProfiler LX) is an advanced kernel rootkit detection toolkit for Linux.

RKProfiler LX is divided into two parts: a data collection component called “Rootkit Profiler Module” (RKPmod) and a data interpretation component called “Rootkit Profiler Console” (RKPconsole).

**RKPmod** is a kernel module that gets loaded on the system that should be checked for the presence of a kernel rootkit. There are other ways to perform data collection, but currently only this approach is publicly available.

**RKPconsole** is a userland program that can be used to analyse the collected information.

## 1.1 Features

**Detection:** RKProfiler LX checks the whole kernel code as well as different kernel data sections and cpu registers regarding possible modifications and hidden components:

- Generic kernel code modification
- Syscall table address modification
- Syscall address modification
- Syscall code modification
- Interrupt handler address modification
- Interrupt handler code modification
- Page Fault Handler modification
- Kernel symbol modification
- SYSENTER register modification
- Virtual File System function pointer modification
- Hidden processes and threads
- Hidden kernel modules

**Self-protection:** RKPmod supports some rudimentary methods to ensure the integrity of itself as well as the integrity of the collected information. The data collection module gets a different name each time it is loaded into the kernel. The collected data is encrypted in the kernel so no unencrypted data will be accessible in userland. Furthermore, the data collection module checks sensitive code parts of itself in memory in order to spot possible runtime in-memory modifications.

**Separation of data collection and data interpretation:** It is possible to analyse the collected data on a different system than the one the data was collected on. Therefore the data interpretation phase is not manipulable by a possible rootkit. Of course but not advisable the data can also be analysed on the same system the data was collected on.

## 1.2 Supported operating systems

RKProfiler LX currently supports the following Linux Distributions:

- SUSE Linux Enterprise Server 10 (x86, 32-bit)
- SUSE Linux Enterprise Desktop 10 (x86, 32-bit)
- Ubuntu 7.04 (x86, 32-bit)
- openSUSE 10.2 (x86, 32-bit)

Only the standard kernels of these distributions are supported. Self compiled kernels are not supported with the public version of RKProfiler LX.

No longer supported Linux Distributions:

- Ubuntu 6.10 Edgy Eft (x86, 32-bit)

## 2 Quick start guide

1. Download the appropriate RKProfiler LX package  
Go to <http://www.trapkit.de/research/rkprofiler/>
2. Transfer the data collection module (RKPmod) on the system that should be analysed (e.g. via scp, USB stick, etc.)
3. Load RKPmod into the kernel by executing the RKPmod script
4. Transfer the dump.tar.bz2 archive to another system (e.g. via scp, USB stick, etc.) and analyse the collected data with RKPconsole. Alternately you can also analyse the data on the same system the data was collected on however this is not recommended.

## 3 Rootkit Profiler Module

The Rootkit Profiler module can be found in the `RKPmod_LX_version_distribution` directory of the appropriate RKProfiler LX package.

**Example:** SUSE Enterprise Server 10

```
# pwd
/home/tk/RKProfiler_LX_v0.1_SLES10/RKPmod_LX_v0.1_SLES10

# ls
RKPmod
```

To load the Rootkit Profiler module into the kernel just execute the RKPmod script:

```
# ./RKPmod
```

After the script has finished execution there should be a new archive named `dump.tar.bz2` in the same directory.

```
# ls
dump.tar.bz2  RKPmod
```

This archive contains all data that was collected by the Rootkit Profiler module.

## 4 Rootkit Profiler Console

The Rootkit Profiler console can be found in the `RKPconsole_LX_version_distribution` directory of the appropriate RKProfiler LX package.

**Example:** SUSE Enterprise Server 10

```
# pwd
/home/tk/RKProfiler_LX_v0.1_SLES10/RKPconsole_LX_v0.1_SLES10

# ls
RKPconsole  SLES10_data
```

The binary of the Rootkit Profiler console is called `RKPconsole`. The directory `SLES10_data` contains the SUSE Enterprise Server 10 kernel signature file, the kernel binary file and the `System.map` file of the kernel.

To analyse the collected data from RKPmod it is necessary to unpack the `dump.tar.bz2` archive.

```
# mkdir data
```

```
# cd data
```

Now copy the dump.tar.bz2 archive into this new data directory and unpack it.

```
# mv ../../RKPmod_LX_v0.1_SLES10/dump.tar.bz2 .
# tar jxvf dump.tar.bz2
```

To analyse the data use the following syntax:

```
# ../RKProfil console -k ../SLES10_data/vmlinuz-2.6.16.27-0.6-default -s ../SLES10_data/signature.xml -m
../SLES10_data/System.map-2.6.16.27-0.6-default
RKProfiler Console v0.1 public edition, Tobias Klein, www.trapkit.de
```

```
[+] Performing checks
```

```
[+] No Rootkit found.
```

```
[+] Done.
```

This example shows a management summary like output. To get more detailed information just add the “-v” option:

```
# ../RKProfil console -k ../SLES10_data/vmlinuz-2.6.16.27-0.6-default -s ../SLES10_data/signature.xml -m
../SLES10_data/System.map-2.6.16.27-0.6-default -v
RKProfiler Console v0.1 public edition, Tobias Klein, www.trapkit.de
```

```
[+] Dump information
```

```
*** Operating system: Linux
*** Distribution      : SUSE Enterprise Server 10
*** Kernel           : 2.6.16.27-0.6-default
```

```
[+] Performing checks
```

```
[+] Checking RKPmod integrity
```

```
*** CRC check: OK
```

```
[+] Checking Interrupt Descriptor Table
```

```
*** No IDT Handler Code discrepancies found.
*** No IDT Handler Address discrepancies found.
```

```
[+] Checking System Call Table address
```

```
*** Method 1: Syscall Symbols
    SCT address from memory      : 0xc0276480
```

```
*** Method 2: INT80
    SCT address from memory      : 0xc0276480
    SCT address from kernel file : 0xc0276480
```

```
System Call Table address of INT80 handler is NOT manipulated!
```

```
*** Method 3: SYSENTER
    SCT address from memory      : 0xc0276480
    SCT address from kernel file : 0xc0276480
```

```
System Call Table address of SYSENTER handler is NOT manipulated!
```

```
[+] Checking for syscall address and code discrepancies [Method 1]
```

```
*** No Syscall Code discrepancies found.
*** No Syscall Address discrepancies found.
```

```
[+] Checking for syscall address and code discrepancies [Method 2]
```

```
*** No Syscall Code discrepancies found.
*** No Syscall Address discrepancies found.
```

```
[+] Checking for syscall address and code discrepancies [Method 3]
```

```
*** No Syscall Code discrepancies found.
```

```
*** No Syscall Address discrepancies found.

[+] Checking for Page Fault Handler Address discrepancies
*** No Page Fault Handler Address discrepancies found.

[+] Checking for kernel code discrepancies
*** No kernel code discrepancies found.

[+] Checking for hidden kernel modules
*** No hidden kernel modules found.

[+] Checking for hidden processes and threads
[*] List comparing:
*** No hidden processes or threads found.

[*] Memory scanning:
*** No hidden processes or threads found.

[+] Checking Symbol Codes
*** No Symbol Code discrepancies found.

[+] Checking VFS function pointers
[*] VFS pointer into kernel modules
*** VFS pointer [reiserfs]_dir_operations (0xd0dd5a00) points into
loaded kernel module: reiserfs (0xd0da4000-0xd0dd7300)

[*] VFS pointer into kernel modules
*** VFS pointer [reiserfs]_readdir (0xd0db14b6) points into
loaded kernel module: reiserfs (0xd0da4000-0xd0dd7300)

[*] VFS pointer into kernel modules
*** VFS pointer [reiserfs]_ioctl (0xd0dc4d6e) points into
loaded kernel module: reiserfs (0xd0da4000-0xd0dd7300)

[*] VFS pointer into kernel modules
*** VFS pointer [reiserfs]_fsync (0xd0db12b8) points into
loaded kernel module: reiserfs (0xd0da4000-0xd0dd7300)

[*] VFS pointers into kernel text and data
*** No VFS function pointer discrepancies found.

[+] Done.
```

To get a description of the different command line options of RKPconsole just execute it without any option:

#### **#!/RKPconsole**

RKProfiler Console v0.1 public edition, Tobias Klein, [www.trapkit.de](http://www.trapkit.de)

Usage: ./RKPconsole [options]

#### Options:

```
-h           - This help
-k [kernel] - The kernel reference binary
-s [infile]  - Check kernel code section (needs reference
               xml file with clean kernel code signature)
-m [System.map] - Check symbol entry code bytes (needs '-s')
-n           - Turn off shell colors
-v           - Be verbose
```

## 5 Interpretation of results

The following examples are used to describe the functionality as well as the interpretation of the results of RKPconsole.

### 5.1 Example 1: Clean system

On a clean system where no rootkit could be found, the output of RKPconsole should look like the following:

Management summary:

RKProfiler Console v0.1 public edition, Tobias Klein, [www.trapkit.de](http://www.trapkit.de)

[+] Performing checks

**[+] No Rootkit found.**

[+] Done.

Verbose technical summary:

RKProfiler Console v0.1 public edition, Tobias Klein, [www.trapkit.de](http://www.trapkit.de)

[+] Dump information

```
*** Operating system: Linux
*** Distribution      : SUSE Enterprise Server 10
*** Kernel           : 2.6.16.27-0.6-default
```

[+] Performing checks

[+] Checking RKPmod integrity

```
*** CRC check: OK
```

[+] Checking Interrupt Descriptor Table

```
*** No IDT Handler Code discrepancies found.
*** No IDT Handler Address discrepancies found.
```

[+] Checking System Call Table address

```
*** Method 1: Syscall Symbols
    SCT address from memory      : 0xc0276480
```

```
*** Method 2: INT80
    SCT address from memory      : 0xc0276480
    SCT address from kernel file : 0xc0276480
```

System Call Table address of INT80 handler is NOT manipulated!

```
*** Method 3: SYSENTER
    SCT address from memory      : 0xc0276480
    SCT address from kernel file : 0xc0276480
```

System Call Table address of SYSENTER handler is NOT manipulated!

[+] Checking for syscall address and code discrepancies [Method 1]

```
*** No Syscall Code discrepancies found.
*** No Syscall Address discrepancies found.
```

[+] Checking for syscall address and code discrepancies [Method 2]

```
*** No Syscall Code discrepancies found.
*** No Syscall Address discrepancies found.
```

[+] Checking for syscall address and code discrepancies [Method 3]

```
*** No Syscall Code discrepancies found.
```

```
*** No Syscall Address discrepancies found.

[+] Checking for Page Fault Handler Address discrepancies
*** No Page Fault Handler Address discrepancies found.

[+] Checking for kernel code discrepancies
*** No kernel code discrepancies found.

[+] Checking for hidden kernel modules
*** No hidden kernel modules found.

[+] Checking for hidden processes and threads
[*] List comparing:
*** No hidden processes or threads found.

[*] Memory scanning:
*** No hidden processes or threads found.

[+] Checking Symbol Codes
*** No Symbol Code discrepancies found.

[+] Checking VFS function pointers
[*] VFS pointers into kernel modules
*** VFS pointer [reiserfs]_dir_operations (0xd0dd5a00) points into
loaded kernel module: reiserfs (0xd0da4000-0xd0dd7300)

[*] VFS pointers into kernel modules
*** VFS pointer [reiserfs]_readdir (0xd0db14b6) points into
loaded kernel module: reiserfs (0xd0da4000-0xd0dd7300)

[*] VFS pointers into kernel modules
*** VFS pointer [reiserfs]_ioctl (0xd0dc4d6e) points into
loaded kernel module: reiserfs (0xd0da4000-0xd0dd7300)

[*] VFS pointers into kernel modules
*** VFS pointer [reiserfs]_fsync (0xd0db12b8) points into
loaded kernel module: reiserfs (0xd0da4000-0xd0dd7300)

[*] VFS pointers into kernel text and data
*** No VFS function pointer discrepancies found.

[+] Done.
```

In the following, the output of the technical summary will be described in more detail.

```
[+] Dump information
*** Operating system: Linux
*** Distribution      : SUSE Enterprise Server 10
*** Kernel            : 2.6.16.27-0.6-default
```

Shows information of the system the data was collected on.

```
[+] Checking RKPmod integrity
*** CRC check: OK
```

Shows if RKPmod was modified in memory while collecting the data.

```
[+] Checking Interrupt Descriptor Table
*** No IDT Handler Code discrepancies found.
*** No IDT Handler Address discrepancies found.
```

Shows if the code or the address of an Interrupt Descriptor Table (IDT) handler was modified.

```
[+] Checking System Call Table address
*** Method 1: Syscall Symbols
```

```
SCT address from memory      : 0xc0276480

*** Method 2: INT80
SCT address from memory      : 0xc0276480
SCT address from kernel file : 0xc0276480

System Call Table address of INT80 handler is NOT manipulated!

*** Method 3: SYSENTER
SCT address from memory      : 0xc0276480
SCT address from kernel file : 0xc0276480

System Call Table address of SYSENTER handler is NOT manipulated!
```

RKPMOD uses three different methods to get the address of the System Call Table (SCT). All methods should provide the same address for the SCT. Furthermore the addresses of the kernel memory should match the ones in the kernel binary. Method 2 and 3 are also used to check if the interrupt handler code of INT80 and the SYSENTER handler code are modified.

```
[+] Checking for syscall address and code discrepancies [Method 1]
    *** No Syscall Code discrepancies found.
    *** No Syscall Address discrepancies found.

[+] Checking for syscall address and code discrepancies [Method 2]
    *** No Syscall Code discrepancies found.
    *** No Syscall Address discrepancies found.

[+] Checking for syscall address and code discrepancies [Method 3]
    *** No Syscall Code discrepancies found.
    *** No Syscall Address discrepancies found.
```

Every SCT address found (see "Checking System Call Table address") is then used as a basis to enumerate and check all the syscall addresses and code areas regarding malicious modifications.

```
[+] Checking for Page Fault Handler Address discrepancies
    *** No Page Fault Handler Address discrepancies found.
```

Shows if a kernel Page Fault Handler Address was modified.

```
[+] Checking for kernel code discrepancies
    *** No kernel code discrepancies found
```

This information shows if there are any modifications of the kernel code.

```
[+] Checking for hidden kernel modules
    *** No hidden kernel modules found.
```

Shows if a hidden kernel module was found.

```
[+] Checking for hidden processes and threads
    [*] List comparing:
    *** No hidden processes or threads found.

    [*] Memory scanning:
    *** No hidden processes or threads found.
```

Shows if a hidden process or thread was found.

```
[+] Checking Symbol Codes
    *** No Symbol Code discrepancies found.
```

Shows if the code area of a kernel symbol was modified.

```
[+] Checking VFS function pointers
```

```
[*] VFS pointers into kernel modules
*** VFS pointer [reiserfs]_dir_operations (0xd0dd5a00) points into
    loaded kernel module: reiserfs (0xd0da4000-0xd0dd7300)

[*] VFS pointers into kernel modules
*** VFS pointer [reiserfs]_readdir (0xd0db14b6) points into
    loaded kernel module: reiserfs (0xd0da4000-0xd0dd7300)

[*] VFS pointers into kernel modules
*** VFS pointer [reiserfs]_ioctl (0xd0dc4d6e) points into
    loaded kernel module: reiserfs (0xd0da4000-0xd0dd7300)

[*] VFS pointers into kernel modules
*** VFS pointer [reiserfs]_fsync (0xd0db12b8) points into
    loaded kernel module: reiserfs (0xd0da4000-0xd0dd7300)

[*] VFS pointers into kernel text and data
*** No VFS function pointer discrepancies found.
```

Shows if function pointers of the Virtual File System (VFS) were modified.

## 5.2 Example 2: Syscall address modification

The following output of RKPconsole shows that the address of a syscall was modified.

Management summary:

RKProfiler Console v0.1 public edition, Tobias Klein, www.trapkit.de

[+] Performing checks

```
*** Syscall Address discrepancies found (Method 1).
*** Syscall Address discrepancies found (Method 2).
*** Syscall Address discrepancies found (Method 3).
```

[+] Done.

Verbose technical summary:

RKProfiler Console v0.1 public edition, Tobias Klein, www.trapkit.de

[+] Dump information

```
*** Operating system: Linux
*** Distribution      : SUSE Enterprise Server 10
*** Kernel           : 2.6.16.27-0.6-default
```

[+] Performing checks

[+] Checking RKPmod integrity  
\*\*\* CRC check: OK

[+] Checking Interrupt Descriptor Table  
\*\*\* No IDT Handler Code discrepancies found.  
\*\*\* No IDT Handler Address discrepancies found.

[+] Checking System Call Table address  
\*\*\* Method 1: Syscall Symbols  
 SCT address from memory : 0xc0276480  
  
\*\*\* Method 2: INT80  
 SCT address from memory : 0xc0276480  
 SCT address from kernel file : 0xc0276480

System Call Table address of INT80 handler is NOT manipulated!

```
*** Method 3: SYSENTER
    SCT address from memory      : 0xc0276480
    SCT address from kernel file : 0xc0276480

    System Call Table address of SYSENTER handler is NOT manipulated!

[+] Checking for syscall address and code discrepancies [Method 1]
    *** Syscall Address discrepancy found.
        Syscall   : sys_mkdir [Address in SCT: 0xd0f16000]
        Kernel code: 0xc0100000-0xc0274ad0 (size: 00174ad0)

    *** No Syscall Code discrepancies found.

[+] Checking for syscall address and code discrepancies [Method 2]
    *** Syscall Address discrepancy found.
        Syscall   : sys_mkdir [Address in SCT: 0xd0f16000]
        Kernel code: 0xc0100000-0xc0274ad0 (size: 00174ad0)

    *** No Syscall Code discrepancies found.

[+] Checking for syscall address and code discrepancies [Method 3]
    *** Syscall Address discrepancy found.
        Syscall   : sys_mkdir [Address in SCT: 0xd0f16000]
        Kernel code: 0xc0100000-0xc0274ad0 (size: 00174ad0)

    *** No Syscall Code discrepancies found.

[+] Checking for Page Fault Handler Address discrepancies
    *** No Page Fault Handler Address discrepancies found.

[+] Checking for kernel code discrepancies
    *** No kernel code discrepancies found.

[+] Checking for hidden kernel modules
    *** No hidden kernel modules found.

[+] Checking for hidden processes and threads
    [*] List comparing:
    *** No hidden processes or threads found.

    [*] Memory scanning:
    *** No hidden processes or threads found.

[+] Checking Symbol Codes
    *** No Symbol Code discrepancies found.

[+] Checking VFS function pointers
    [*] VFS pointer into kernel modules
    *** VFS pointer [reiserfs]_dir_operations (0xd0dd5a00) points into
        loaded kernel module: reiserfs (0xd0da4000-0xd0dd7300)

    [*] VFS pointer into kernel modules
    *** VFS pointer [reiserfs]_readdir (0xd0db14b6) points into
        loaded kernel module: reiserfs (0xd0da4000-0xd0dd7300)

    [*] VFS pointer into kernel modules
    *** VFS pointer [reiserfs]_ioctl (0xd0dc4d6e) points into
        loaded kernel module: reiserfs (0xd0da4000-0xd0dd7300)

    [*] VFS pointer into kernel modules
    *** VFS pointer [reiserfs]_fsync (0xd0db12b8) points into
        loaded kernel module: reiserfs (0xd0da4000-0xd0dd7300)

    [*] VFS pointers into kernel text and data
    *** No VFS function pointer discrepancies found.
```

[+] Done.

The bold lines of the RKPconsole output show that the address of the `sys_mkdir` syscall was modified. The addresses of all syscalls should normally point into the kernel code area. The output shows, that the kernel code area starts at address `0xc0100000` and ends at `0xc0274ad0`. The address of the `sys_mkdir` syscall points to code at the address `0xd0f16000` that's definitely not in the kernel code area.

### 5.3 Example 3: Syscall code modification

The following output of RKPconsole shows that the code of a syscall within the System Call Table was modified.

Management summary:

RKProfiler Console v0.1 public edition, Tobias Klein, www.trapkit.de

[+] Performing checks

```
*** Syscall Code discrepancies found (Method 1).
*** Syscall Code discrepancies found (Method 2).
*** Syscall Code discrepancies found (Method 3).
*** Kernel code discrepancies found.
*** Symbol Code discrepancies found.
```

[+] Done.

Verbose technical summary:

RKProfiler Console v0.1 public edition, Tobias Klein, www.trapkit.de

[+] Dump information

```
*** Operating system: Linux
*** Distribution      : SUSE Enterprise Server 10
*** Kernel           : 2.6.16.27-0.6-default
```

[+] Performing checks

[+] Checking RKPmod integrity

```
*** CRC check: OK
```

[+] Checking Interrupt Descriptor Table

```
*** No IDT Handler Code discrepancies found.
*** No IDT Handler Address discrepancies found.
```

[+] Checking System Call Table address

```
*** Method 1: Syscall Symbols
    SCT address from memory      : 0xc0276480
```

```
*** Method 2: INT80
    SCT address from memory      : 0xc0276480
    SCT address from kernel file : 0xc0276480
```

System Call Table address of INT80 handler is NOT manipulated!

```
*** Method 3: SYSENTER
    SCT address from memory      : 0xc0276480
    SCT address from kernel file : 0xc0276480
```

System Call Table address of SYSENTER handler is NOT manipulated!

[+] Checking for syscall address and code discrepancies [Method 1]

```
*** Syscall Code discrepancy found.
    Syscall: sys_mkdir [Address in SCT: 0xc0157a80]
    Syscall entry bytes in memory      : bd1a60f1d0ffe5000000
```

**Syscall entry bytes in kernel file: ff742408ff7424086a9c**

\*\*\* No Syscall Address discrepancies found.

[+] Checking for syscall address and code discrepancies [Method 2]

\*\*\* **Syscall Code discrepancy found.**

**Syscall: sys\_mkdir [Address in SCT: 0xc0157a80]**

**Syscall entry bytes in memory : bd1a60f1d0ffe5000000**

**Syscall entry bytes in kernel file: ff742408ff7424086a9c**

\*\*\* No Syscall Address discrepancies found.

[+] Checking for syscall address and code discrepancies [Method 3]

\*\*\* **Syscall Code discrepancy found.**

**Syscall: sys\_mkdir [Address in SCT: 0xc0157a80]**

**Syscall entry bytes in memory : bd1a60f1d0ffe5000000**

**Syscall entry bytes in kernel file: ff742408ff7424086a9c**

\*\*\* No Syscall Address discrepancies found.

[+] Checking for Page Fault Handler Address discrepancies

\*\*\* No Page Fault Handler Address discrepancies found.

[+] Checking for kernel code discrepancies

**Discrepancy: memaddress=0xc0157a80 memvalue=bd filevalue=ff**

**Discrepancy: memaddress=0xc0157a81 memvalue=1a filevalue=74**

**Discrepancy: memaddress=0xc0157a82 memvalue=60 filevalue=24**

**Discrepancy: memaddress=0xc0157a83 memvalue=f1 filevalue=08**

**Discrepancy: memaddress=0xc0157a84 memvalue=d0 filevalue=ff**

**Discrepancy: memaddress=0xc0157a85 memvalue=ff filevalue=74**

**Discrepancy: memaddress=0xc0157a86 memvalue=e5 filevalue=24**

**Discrepancy: memaddress=0xc0157a87 memvalue=00 filevalue=08**

**Discrepancy: memaddress=0xc0157a88 memvalue=00 filevalue=6a**

**Discrepancy: memaddress=0xc0157a89 memvalue=00 filevalue=9c**

**Discrepancy: memaddress=0xc0157a8a memvalue=00 filevalue=e8**

**Discrepancy: memaddress=0xc0157a8b memvalue=00 filevalue=3b**

**Discrepancy: memaddress=0xc0157a8c memvalue=00 filevalue=ff**

**Discrepancy: memaddress=0xc0157a8d memvalue=00 filevalue=ff**

**Discrepancy: memaddress=0xc0157a8e memvalue=00 filevalue=ff**

**Discrepancy: memaddress=0xc0157a8f memvalue=00 filevalue=83**

[+] Checking for hidden kernel modules

\*\*\* No hidden kernel modules found.

[+] Checking for hidden processes and threads

[\*] List comparing:

\*\*\* No hidden processes or threads found.

[\*] Memory scanning:

\*\*\* No hidden processes or threads found.

[+] Checking Symbol Codes

\*\*\* **Symbol Code discrepancy found.**

**Symbol name : sys\_mkdir**

**Symbol address : 0xc0157a80**

**Code bytes in memory: bd**

**Code bytes in file : ff**

[+] Checking VFS function pointers

[\*] VFS pointers into kernel modules

\*\*\* VFS pointer [reiserfs]\_dir\_operations (0xd0dd5a00) points into loaded kernel module: reiserfs (0xd0da4000-0xd0dd7300)

[\*] VFS pointers into kernel modules

\*\*\* VFS pointer [reiserfs]\_readdir (0xd0db14b6) points into loaded kernel module: reiserfs (0xd0da4000-0xd0dd7300)

```

[*] VFS pointers into kernel modules
*** VFS pointer [reiserfs]_ioctl (0xd0dc4d6e) points into
    loaded kernel module: reiserfs (0xd0da4000-0xd0dd7300)

[*] VFS pointers into kernel modules
*** VFS pointer [reiserfs]_fsync (0xd0db12b8) points into
    loaded kernel module: reiserfs (0xd0da4000-0xd0dd7300)

[*] VFS pointers into kernel text and data
*** No VFS function pointer discrepancies found.

```

[+] Done.

The bold lines of the RKPconsole output show that the code of the `sys_mkdir` syscall was modified.

```

[+] Checking for syscall address and code discrepancies [Method 1]
*** Syscall Code discrepancy found.
    Syscall: sys_mkdir [Address in SCT: 0xc0157a80]
    Syscall entry bytes in memory      : bd1a60f1d0ffe5000000
    Syscall entry bytes in kernel file: ff742408ff7424086a9c

```

This output snippet shows that the code bytes in kernel memory don't match the code bytes in the kernel binary. As there is a modification of the kernel code, the modified code bytes are also found by the generic check of the whole kernel code area:

```

[+] Checking for kernel code discrepancies
Discrepancy: memaddress=0xc0157a80 memvalue=bd filevalue=ff
Discrepancy: memaddress=0xc0157a81 memvalue=1a filevalue=74
Discrepancy: memaddress=0xc0157a82 memvalue=60 filevalue=24
Discrepancy: memaddress=0xc0157a83 memvalue=f1 filevalue=08
Discrepancy: memaddress=0xc0157a84 memvalue=d0 filevalue=ff
Discrepancy: memaddress=0xc0157a85 memvalue=ff filevalue=74
Discrepancy: memaddress=0xc0157a86 memvalue=e5 filevalue=24
Discrepancy: memaddress=0xc0157a87 memvalue=00 filevalue=08
Discrepancy: memaddress=0xc0157a88 memvalue=00 filevalue=6a
Discrepancy: memaddress=0xc0157a89 memvalue=00 filevalue=9c
Discrepancy: memaddress=0xc0157a8a memvalue=00 filevalue=e8
Discrepancy: memaddress=0xc0157a8b memvalue=00 filevalue=3b
Discrepancy: memaddress=0xc0157a8c memvalue=00 filevalue=ff
Discrepancy: memaddress=0xc0157a8d memvalue=00 filevalue=ff
Discrepancy: memaddress=0xc0157a8e memvalue=00 filevalue=ff
Discrepancy: memaddress=0xc0157a8f memvalue=00 filevalue=83

```

As the `sys_mkdir` syscall is also a (non-exported) code symbol of the kernel, the code discrepancies are also found while checking all the symbols of the kernel:

```

[+] Checking Symbol Codes
*** Symbol Code discrepancy found.
    Symbol name      : sys_mkdir
    Symbol address   : 0xc0157a80
    Code bytes in memory: bd
    Code bytes in file  : ff

```

## 5.4 Example 4: VFS function pointer modification

The following output of RKPconsole shows that two function pointers of the Virtual File System (VFS) were modified.

Management summary:

[+] Performing checks

\*\*\* VFS function pointer discrepancies found.  
\*\*\* VFS function pointer discrepancies found.

[+] Done.

Verbose technical summary:

RKProfiler Console v0.1 public edition, Tobias Klein, [www.trapkit.de](http://www.trapkit.de)

[+] Dump information

\*\*\* Operating system: Linux  
\*\*\* Distribution : SUSE Enterprise Server 10  
\*\*\* Kernel : 2.6.16.27-0.6-default

[+] Performing checks

[+] Checking RKPmod integrity

\*\*\* CRC check: OK

[+] Checking Interrupt Descriptor Table

\*\*\* No IDT Handler Code discrepancies found.  
\*\*\* No IDT Handler Address discrepancies found.

[+] Checking System Call Table address

\*\*\* Method 1: Syscall Symbols  
SCT address from memory : 0xc0276480

\*\*\* Method 2: INT80  
SCT address from memory : 0xc0276480  
SCT address from kernel file : 0xc0276480

System Call Table address of INT80 handler is NOT manipulated!

\*\*\* Method 3: SYSENTER  
SCT address from memory : 0xc0276480  
SCT address from kernel file : 0xc0276480

System Call Table address of SYSENTER handler is NOT manipulated!

[+] Checking for syscall address and code discrepancies [Method 1]

\*\*\* No Syscall Code discrepancies found.  
\*\*\* No Syscall Address discrepancies found.

[+] Checking for syscall address and code discrepancies [Method 2]

\*\*\* No Syscall Code discrepancies found.  
\*\*\* No Syscall Address discrepancies found.

[+] Checking for syscall address and code discrepancies [Method 3]

\*\*\* No Syscall Code discrepancies found.  
\*\*\* No Syscall Address discrepancies found.

[+] Checking for Page Fault Handler Address discrepancies

\*\*\* No Page Fault Handler Address discrepancies found.

[+] Checking for kernel code discrepancies

\*\*\* No kernel code discrepancies found.

[+] Checking for hidden kernel modules

\*\*\* No hidden kernel modules found.

[+] Checking for hidden processes and threads

[\*] List comparing:  
\*\*\* No hidden processes or threads found.

```
[*] Memory scanning:
*** No hidden processes or threads found.

[+] Checking Symbol Codes
*** No Symbol Code discrepancies found.

[+] Checking VFS function pointers
[*] VFS pointers into kernel text and data
*** VFS function pointer discrepancy found.
    Function pointer name: proc_root_readdir
    Address in memory      : 0xd0f1d05c
    Reference address      : 0xc0171c70

[*] VFS pointer into kernel modules
*** VFS pointer [reiserfs]_dir_operations (0xd0dd5a00) points into
    loaded kernel module: reiserfs (0xd0da4000-0xd0dd7300)
*** Pointer seems to be valid.

[*] VFS pointer into kernel modules
*** VFS pointer [reiserfs]_readdir (0xd0f1d098) points into
    loaded kernel module: vfs (0xd0f1d000-0xd0f1da10)
*** Possible pointer modification found.

[*] VFS pointer into kernel modules
*** VFS pointer [reiserfs]_ioctl (0xd0dc4d6e) points into
    loaded kernel module: reiserfs (0xd0da4000-0xd0dd7300)
*** Pointer seems to be valid.

[*] VFS pointer into kernel modules
*** VFS pointer [reiserfs]_fsync (0xd0db12b8) points into
    loaded kernel module: reiserfs (0xd0da4000-0xd0dd7300)
*** Pointer seems to be valid.

[+] Done.
```

The bold lines of the RKPconsole output show that the function pointers `proc_root_readdir` as well as `[reiserfs]_readdir` were modified.

```
[*] VFS pointers into kernel text and data
*** VFS function pointer discrepancy found.
    Function pointer name: proc_root_readdir
    Address in memory      : 0xd0f1d05c
    Reference address      : 0xc0171c70
```

This output snippet shows that the VFS function pointer `proc_root_readdir` points to the address `0xd0f1d05c`. On a clean unmodified system this function pointer should point to the address `0xc0171c70`.

```
[*] VFS pointer into kernel modules
*** VFS pointer [reiserfs]_readdir (0xd0f1d098) points into
    loaded kernel module: vfs (0xd0f1d000-0xd0f1da10)
*** Possible pointer modification found.
```

In this example the `[reiserfs]_readdir` function pointer points into a kernel module called `vfs`. On a clean unmodified system the pointer should normally point into a kernel module with the same name as the used filesystem. This should be `reiserfs` in this case.