CS 6V81-05: System Security and Malicious Code Analysis
An Overview of Linux (with Source) and Windows Kernel

Zhiqiang Lin

Department of Computer Science
University of Texas at Dallas

February 20th, 2012
Outline

1. Linux Kernel Architecture
2. Linux Kernel Source Code
3. Windows Kernel Architecture
4. OS Comparison
5. Summary
In the following a few weeks: OS Kernel

User program → Compiler → Executable binary → Linker → System library → Operating system → Device drivers → Architecture

user
compiler
OS interface
OS
Hardware
Unix history

1970
1980
1990
2000

Time

BSD family

BSD (Berkeley Software Distribution)

SunOS 4.1.3

Xenix OS

BSD (Berkeley Software Distribution)

NetBSD

OpenBSD

FreeBSD 7.2

Darwin

MacOS X 5.7

GNU/Hurd K16

GNU Project

Richard Stallman

Minix

Linus Torvalds

Andrew S. Tanenbaum

Research Unix (Bell Labs) 10

Commercial Unix (AT&T)

UnixWare (Univel/SCO)

Solaris (Sun Microsystems) 10 5/09

HP-UX

AIX (IBM) 7.1.4 MP46.1

IRIX (SGI) 6.5.30

System III & V family
Outline

1. Linux Kernel Architecture
2. Linux Kernel Source Code
3. Windows Kernel Architecture
4. OS Comparison
5. Summary
"Core" Kernel

- Applications
- System Libraries (libc)
- System Call Interface
- Hardware
  - Architecture-Dependent Code
    - I/O Related
      - File Systems
      - Networking
      - Device Drivers
    - Process Related
      - Scheduler
      - Memory Management
      - IPC
Linux Kernel Architecture

- Why use monolithic kernel?
  - begin from ‘slow’ 386 architecture
  - micro-kernel depends on careful design
- Linux use module approach to make use of the advantages of micro-kernel
Source Tree Layout

```
Documentation

/arch
/drivers
/fs
/include
/init
/ipc
/kernel
/lib
/mm
/net
/scripts
```

```
alpha
arm
i386
ia64
m68k
mips
mips64
ppc
s390
sh
sparc
sparc64

adfs
affs
autofs
autofs4
bfs
code
cramfs
devfs
devpts
efs
ext2
fat
hfs
hpfs
...

asm-alpha
asm-arm
asm-generic
asm-i386
asm-ia64
asm-m68k
asm-mips
asm-mips64
...

linux
math-emu
net
pcmcia
scsi
video

adfs
affs
autofs
autofs4
bfs
code
cramfs
devfs
devpts
efs
ext2
fat
hfs
hpfs
...

802
appletalk
atm
ax25
bridge
core
decnet
econet
ethernet
ipv4
ipv6
irda
khttpd
lapb
...
```
spotty but important collection of developer-generated documentation; you need to read what’s in here!

recent effort to produce javadoc-style documentation from source header comments using OpenDoc

an ambitious open-source kernel book effort has begun; see kernelbook.sourceforge.net for details

some especially interesting entries:
  - kernel-docs.txt (a bit out of date but good)
  - filesystems/ (very extensive)
  - networking/ (very extensive)
  - kmod.txt
  - oops-tracing.txt
  - spinlocks.txt (the official story from Linus)
Subdirectories for each current port

Each contains kernel, lib, mm, boot and other directories whose contents override code stubs in architecture independent code

Lib contains highly-optimized common utility routines such as memcpy, checksums, etc.

Arch as of 2.6:
- alpha, arm, i386, ia64, m68k, mips, mips64
- ppc, s390, sh, sparc, sparc64
linux/drivers

- largest amount of code in the kernel tree (1.5M)
- device, bus, platform and general directories
- drivers/char - n_tty.c is the default line discipline
- drivers/block - elevator.c, genhd.c, linear.c, ll_rw_blk.c, raidN.c
- drivers/net - specific drivers and general routines Space.c and net_init.c
- drivers/scsi - scsi_* .c files are generic; sd.c (disk), sr.c (CD-ROM), st.c (tape), sg.c (generic)
- general:
  - cdrom, ide, isdn, parport, pcmcia,
  - pnp, sound, telephony, video
- buses - fc4, i2c, nubus, pci, sbus, tc, usb
- platforms - acorn, macintosh, s390, sgi
**linux/fs**

- contains:
  - virtual filesystem (VFS) framework
  - subdirectories for actual filesystems

- vfs-related files:
  - exec.c, binfmt_* .c - files for mapping new process images
  - devices.c, blk_dev.c - device registration, block device support
  - super.c, filesystems.c
  - inode.c, dcache.c, namei.c, buffer.c, file_table.c
  - open.c, read_write.c, select.c, pipe.c, fifo.c
  - fcntl.c, ioctl.c, locks.c, dquot.c, stat.c
**linux/include**

- **include/asm-***
  - architecture-dependent include subdirectories

- **include/linux**
  - header info needed both by the kernel and user apps
  - usually linked to `/usr/include/linux`
  - kernel-only portions guarded by `#ifdef`
    - `#ifdef __KERNEL__`
    - /* kernel stuff */
    - `#endif`

- **other directories:**
  - math-emu
  - net
  - pcmcia
  - scsi
  - video
just two files: version.c, main.c
version.c - contains the version banner that prints at boot
main.c - architecture-independent boot code
start_kernel is the primary entry point
System V IPC facilities

if disabled at compile-time, util.c exports stubs that simply return -ENOSYS

one file for each facility:

- sem.c - semaphores
- shm.c - shared memory
- msg.c - message queues
the core kernel code
  - sched.c - “the main kernel file”
  - scheduler, wait queues, timers, alarms, task queues
process control
  - fork.c, exec.c, signal.c, exit.c
  - acct.c, capability.c, exec_domain.c
kernel module support
  - kmod.c, ksymbs.c, module.c
other operations
  - time.c, resource.c, dma.c, softirq.c, itimer.c
  - printk.c, info.c, panic.c, sysctl.c, sys.c
kernel code cannot call standard C library routines

files:
- brlock.c - “Big Reader” spinlocks
- cmdline.c - kernel command line parsing routines
- errno.c - global definition of errno
- inflate.c - “gunzip” part of gzip.c used during boot
- string.c - portable string code
- usually replaced by optimized, architecture-dependent routines
- vsprintf.c - libc replacement
Linux Kernel Architecture

Linux Kernel Source Code

Windows Kernel Architecture

OS Comparison

Summary

---

**Linux/mm**

- **paging and swapping**
  - swap.c, swapfile.c (paging devices), swap_state.c (cache)
  - vmscan.c - paging policies, kwapd
  - page_io.c - low-level page transfer

- **allocation and deallocation**
  - slab.c - slab allocator
  - page_alloc.c - page-based allocator zone allocator
  - vmalloc.c - kernel virtual-memory allocator

- **memory mapping**
  - memory.c - paging, fault-handling, page table code
  - filemap.c - file mapping
  - mmap.c, mremap.c, mlock.c, mprotect.c
scripts for:

- menu-based kernel configuration
- kernel patching
- generating kernel documentation
Where to start looking

- **System startup and initialization**
  - `arch/i386/kernel/head.S`
  - Some arch-dependent setup and then jumps to the main() in `init/main.c`

- **Memory management**
  - Page fault handling: `mm/memory.c`
  - Memory mapping and page caching: `mm/filemap.c`
  - Buffer cache: `mm/buffer.c`
  - Swap cache: `mm/swap_state.c, mm/swapfile.c`

- **Kernel**
  - Scheduler: `kernel/sched.c`
  - Fork: `kernel/fork.c`
  - Bottom half of int handling: `include/linux/interrupt.h`
  - Proc table: `include/linux/sched.h`
Where to start looking

- Interrupt handling
  - Almost architecture specific
  - `arch/i386/kernel/irq.c`

- Network
  - Networking code in `net`, include files in `include/net`
  - BSD socket code: `net/socket.c`
  - IP version 4 INET code: `net/ipv4/af_inet.c`
  - Generic protocol support code: `net/core`
  - TCP/IP networking code: `net/ipv4`

- How to start looking
  - Few tools available
  - `vi`, `ctags`, `cflow`
Outline

1. Linux Kernel Architecture
2. Linux Kernel Source Code
3. Windows Kernel Architecture
4. OS Comparison
5. Summary
Windows Kernel Architecture

Applications

Subsystem servers
- DLLs
- System Services
- Login/GINA
- Kernel32
- Critical services
- User32 / GDI

User-mode
- ntdll / run-time library

Kernel-mode
- Trap interface / LPC

Security
- refmon

IO Manager
- Virtual memory
- Procs & threads
- Win32 GUI

- File filters
- File systems
- Volume mgrs
- Device stacks

Virtual memory
- FS run-time
- Cache mgr

Object Manager / Configuration Management

Kernel run-time / Hardware Adaptation Layer

© Microsoft Corporation 2004
Windows Kernel Organization

Kernel-mode organized into

- **NTOS (kernel-mode services)**
  - Run-time Library, Scheduling, Executive services, object manager, services for I/O, memory, processes

- **Hal (hardware-adaptation layer)**
  - Insulates NTOS & drivers from hardware dependencies
  - Provides facilities, such as device access, timers, interrupt servicing, clocks, spinlocks

- **Drivers**
  - kernel extensions (primarily for device access)
**Major Kernel Services**

- Process management
  - Process/thread creation
  - Schedules thread execution on each processor
- Security reference monitor
  - Access checks, token management
- Memory manager
  - Pagefaults, virtual address, physical frame, and pagefile management
  - Services for sharing, copy-on-write, mapped files, GC support, large apps
- Lightweight Procedure Call (LPC)
  - Native transport for RPC and user-mode system services.
- I/O manager (plug-and-play, power)
  - Maps user requests into IRP requests, configures/manages I/O devices, implements services for drivers
- Cache manager
  - Provides file-based caching for buffer file system I/O
  - Built over the memory manager
Windows Kernel Organization

Kernel-mode organized into

- **NTOS (kernel-mode services)**
  - Run-time Library, Scheduling, Executive services, object manager, services for I/O, memory, processes

- **Hal (hardware-adaptation layer)**
  - Insulates NTOS & drivers from hardware dependencies
  - Providers facilities, such as device access, timers, interrupt servicing, clocks, spinlocks

- **Drivers**
  - kernel extensions (primarily for device access)
OS Providers

- Microsoft, IBM, Apple, Oracle (Sun), Google...
History of Windows


- 1975-1981: Microsoft boots up (Microsoft co-founders Paul Allen (left) and Bill Gates)
- 1982-1985: Introducing Windows 1.0
- 1995-2001: Windows 95—the PC comes of age (and don’t forget the Internet)
- 2001-2005: Windows XP—Stable, usable, and fast
- 2006-2008: Windows Vista—Smart on security
- 2009-Today: Windows 7 and counting...
Mac OS

- Mac OS is a super-modern operating system that combines the power and stability of UNIX with the legendary elegance of the Macintosh.
- Mac OS features a stunning new user interface, making work and play on the Mac even more intuitive for new users, while providing powerful, customizable tools for professionals. At the foundation of Mac OS lies an industrial-strength UNIX-based core operating system that delivers unprecedented stability and performance.
Comparing Operating Systems

Comparing:
- Windows XP/Vista/7
- Macintosh OS X
- Linux

- Price?
- Hardware platform?
- Included Software?
- Ease of Use?
- Pretty?
- Software Availability?
Summary

- Linux kernel
  - Linux kernel source code
- Windows kernel
- Comparison of the common OSes

Book Recommendation
Understanding the Linux Kernel, 3rd Edition
http://voinici.ceata.org/tct/resurse/utlk.pdf