CYBER502x
Computer Forensics

Unit 3: Linux/Unix Filesystems

Investigating Linux/Unix systems

- Four basic forensics steps
  - Collect
  - Preserve
  - *Analyze*
  - Present (report)

What is special about forensics analysis?

- Show deleted content and other data that is typically inaccessible
- Parse data directly from the image bypassing kernel’s support
- Work on both images and live systems

Lets talk about the filesystem…

- Recall the boot process…
- BIOS/MBR
    - BIOS starts POST
    - Finds DRIVES using BIOS sequence
    - Finds BOOTABLE MEDIA
        - MBR, located in sector zero, contains the initial boot code
- EFI (Extensible Firmware Interface)/GPT
    - EFI jumps to the EFI system partition (ESP) that begin the initial boot strapping

GUID Partition Table (GPT)

- GPT scheme and EFI boot framework
- Supports up to 128 primary partitions on a GPT disk
- Provides unique identification of both disks and partitions
- Tools for managing GPT
  - OSX – Disk Utility (or gpt)
  - Windows – DISKPART
  - Linux – parted /dev/xxx print
  - mmls –t gpt /dev/xxx

GPT Forensic Analysis (not required)

- Bruce J. Nikkel, *Forensic Analysis of GPT Disks and GUID Partition Tables,* The International Journal of Digital Forensics and Incident Response Vol. 6, No. 1-2
- Brian Carrier, *File System Forensic Analysis,* Addison Wesley, 2005

MBR structure

- MBR executable code starts at offset 0x0000, total 446 bytes
  - The MBR messages start at offset 0x008b
- The partition table starts at offset 0x01be
  - 64 bytes in four 16 bytes sections
  - 1st partition Entry starts at offset 0x01be
  - 2nd partition entry starts at offset 0x01ce
  - 3rd partition entry starts at offset 0x01de
  - 4th partition entry starts at offset 0x01ee
- The signature is at offset 0x01fe, 2 bytes (55AAh)
- Total 512 bytes

File systems in Unix/Linux

- Unix File System
  - UFS, or Solaris and BSD systems BSD derived (from Berkeley fast file system (FFS))
- Linux File System – derived from UFS
  - Extended file system (ext) in 1992
  - ext, ext2 (1993), ext3(2001), ext4 (2008)

Disk Structure

- Disks are partitioned, and filesystem is installed
- Each partition contains the following areas:
  - An optional boot block
  - A super block defining the boundaries of the other areas.
  - Block Bitmap
  - Inode Bitmap
  - A set of file information blocks known as I-nodes
  - The data blocks (free and used intermingled)

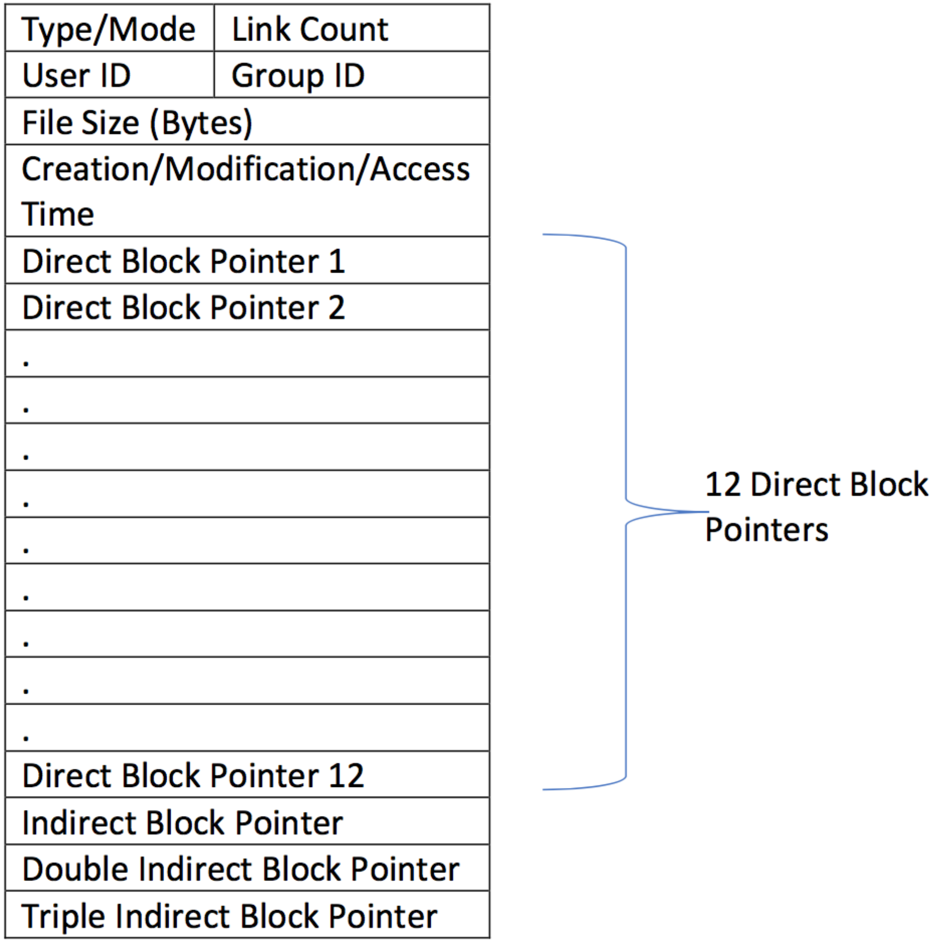
superblock

- Superblock contains
  - Magic Number
    - For EXT2, it is 0xEF53
  - Mount Count and Maximum Mount Count
  - Block Size, for example 4096B
  - Inode count and Block count
  - Number of free disk blocks
  - Number of free inodes on the system
  - First inode
    - This is the inode number of the first inode in the file system. The first inode in an EXT2 root file system would be the directory entry for the ‘/’ directory

Inodes

- Contain META data (info about files)
  - type
  - Owner
  - Permission bits
  - MAC times
  - Links to the file (link count)
  - Data block addresses

Inode structure (each row is 32 bits)



Type/Mode	Link Count
User ID	Group ID
File Size (Bytes)	
Creation/Modification/Access Time	
Direct Block Pointer 1	
Direct Block Pointer 2	
.	
.	
.	
.	
.	
.	
.	
.	
.	
Direct Block Pointer 12	
Indirect Block Pointer	
Double Indirect Block Pointer	
Triple Indirect Block Pointer	

12 Direct Block Pointers

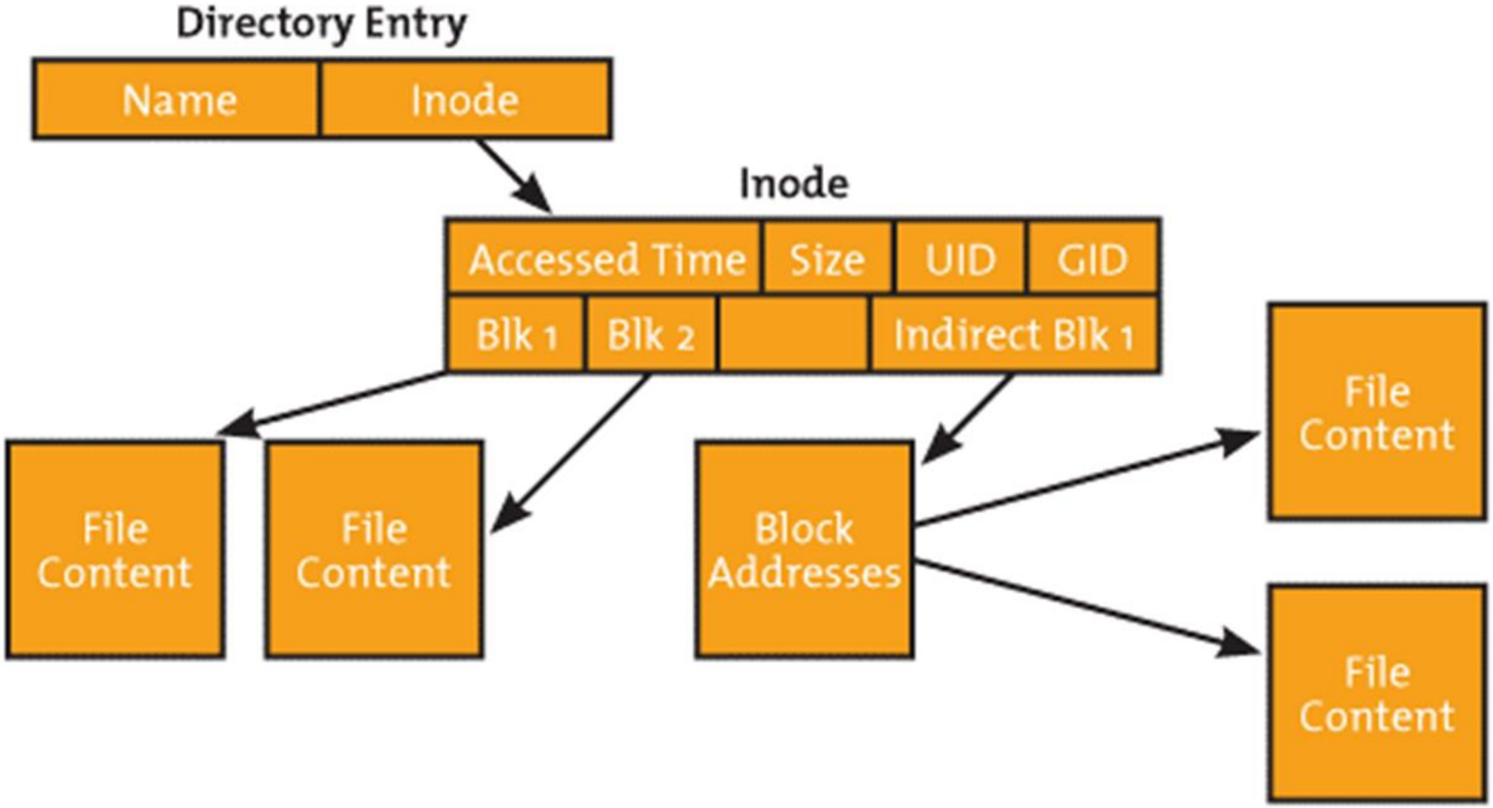
Directories

- Everything in Linux is a file
- Directory is a simple file whose data is a sequence of file entries which contains:
  - Inode number
  - Byte offset in directory (or the length of this entry)
  - File name

Directory example

Byte offset in directory	Inode number	File name
0	70	.
16	35	..
32	123	file1
48	345	file2
64	90	file3

Directory entry



What happened when we create a file

- Each file has data stored in **blocks, inodes and directory entries**
  - A free inode is chosen from inode bitmap
  - The superblock free-inode values are decremented
  - Add an entry in the parent directory
  - Choose a free data block from data bitmap to contain the file contents
- Fill in the inode contents

How files are deleted…

- http://www.porcupine.org/forensics/forensic-discovery/chapter4.html
- When the link-count in the inode reaches zero (0):
  - The Data blocks in the Block Bitmap are marked as free
  - The inode in the Inode Bitmap is marked as free
  - The deletion time is set in the inode.
  - The directory entry is marked as unused.

What happens when a file is deleted in ext3/ext4?

- The file size in the inode is set to zero.
- The data blocks info in the inode is cleared.
- http://linux.sys-con.com/node/117909 (not required)
- *An analysis of Ext4 for digital forensics*, Kevin Fairbanks, 2012, in dfrws.org proceedings (not required)

Why do I care about deletion?

- Data still exists on disk
  - Recoverable until space is overwritten
  - Larger disks less likely to overwrite formerly-used space
- Therefore, you can (usually) recover deleted files
  - Unless the disk blocks are “wiped” before the file is deleted (e.g. “srm” or “shred”) (dd if=/dev/zero or /dev/random)