CS 6V81-05
Metadata Extraction Tools

Junyuan Zeng
Department of Computer Science
The University of Texas at Dallas

September 23\textsuperscript{th}, 2011
Outline

1. Overview
2. Metadata Extraction Tool
   - Overview
   - Demo
3. Hachoir
   - Overview
   - Demo
4. Sleuthkit
   - Overview
   - Demo
5. Summary
Outline

1. Overview

2. Metadata Extraction Tool
   - Overview
   - Demo

3. Hachoir
   - Overview
   - Demo

4. Sleuthkit
   - Overview
   - Demo

5. Summary
What is Metadata

Metadata is described as "data about data." In technical disciplines, metadata can refer to "data about data structures" or "data about database systems." From a digital forensics perspective, metadata is also defined as "evidence, typically stored electronically, that describes the characteristics, origins, usage and validity of other electronic evidence."
What is Metadata

- described as "data about data"
What is Metadata

- described as "data about data"
- refer to "data about data structures" or "data about database systems" in technical disciplines
What is Metadata

- described as "data about data"
- refer to "data about data structures" or "data about database systems" in technical disciplines
- from a **digital forensics** perspective, metadata is also defined as "evidence, typically stored electronically, that describes the characteristics, origins, usage and validity of other electronic evidence."
## Metadata and Digital forensics

Data recorded by metadata tools include:

- **Register values**
- **Memory state**
- **Timers**
- **Network events**
- **Interrupt information**

**Need for large storage space**

**System snapshot** reflects the state of the system at a discrete point in time.

**Files** as indicators of system activity:

- **File accesses**
- **File modifications**
- **inode information** etc.
Metadata and Digital forensics

- record a system’s state (register values, memory, timers, network events, interrupt information etc) for every single clock step, one could replay all events that took place on the system
Metadata and Digital forensics

- record a system’s state (register values, memory, timers, network events, interrupt information etc) for every single clock step, one could replay all events that took place on the system
  - need large storage space
Metadata and Digital forensics

- record a system’s state (register values, memory, timers, network events, interrupt information etc) for every single clock step, one could replay all events that took place on the system
  - need large storage space
  - slow down the system
Metadata and Digital forensics

- record a system’s state (register values, memory, timers, network events, interrupt information etc) for every single clock step, one could replay all events that took place on the system
  - need large storage space
  - slow down the system
- snapshot of the system’s state
Metadata and Digital forensics

- record a system’s state (register values, memory, timers, network events, interrupt information etc) for every single clock step, one could replay all events that took place on the system
  - need large storage space
  - slow down the system

- snapshot of the system’s state
  - reflects the system state at a given discrete point in time
Metadata and Digital forensics

- record a system’s state (register values, memory, timers, network events, interrupt information etc) for every single clock step, one could replay all events that took place on the system
  - need large storage space
  - slow down the system
- snapshot of the system’s state
  - reflects the system state at a given discrete point in time
  - feasible for critical system
Metadata and Digital forensics

- record a system’s state (register values, memory, timers, network events, interrupt information etc) for every single clock step, one could replay all events that took place on the system
  - need large storage space
  - slow down the system
- snapshot of the system’s state
  - reflects the system state at a given discrete point in time
  - feasible for critical system
- **Files** are a good indicator of what actions took place on a system
Metadata and Digital forensics

- record a system’s state (register values, memory, timers, network events, interrupt information etc) for every single clock step, one could replay all events that took place on the system
  - need large storage space
  - slow down the system
- snapshot of the system’s state
  - reflects the system state at a given discrete point in time
  - feasible for critical system
- **Files** are a good indicator of what actions took place on a system
  - file accesses, file modification, inode info etc.
What does File Metadata include

File Metadata includes but is not limited to:
- the dates a file was created, modified, accessed or deleted
- the location of a file on a computer or network
- subfiles in one file
- file system format and inode info
- . . .
Outline

1. Overview

2. Metadata Extraction Tool
   - Overview
   - Demo

3. Hachoir
   - Overview
   - Demo

4. Sleuthkit
   - Overview
   - Demo

5. Summary
Outline

1. Overview

2. Metadata Extraction Tool
   - Overview
   - Demo

3. Hachoir
   - Overview
   - Demo

4. Sleuthkit
   - Overview
   - Demo

5. Summary
Introduction

- general developed by the National Library of New Zealand to extract preservation metadata from a range of file formats
Introduction

- general developed by the National Library of New Zealand to extract preservation metadata from a range of file formats
- initially developed in 2003 and released as open source software in 2007
Introduction

- general developed by the National Library of New Zealand to extract preservation metadata from a range of file formats
- initially developed in 2003 and released as open source software in 2007
- Microsoft Windows interface and UNIX command line interface
Introduction

- developed by the National Library of New Zealand to extract preservation metadata from a range of file formats
- initially developed in 2003 and released as open source software in 2007
- Microsoft Windows interface and UNIX command line interface
- written in Java and XML and is distributed under the Apache Public License
Supported File Formats

- **Images:** BMP, GIF, JPEG and TIFF
- **Office documents:** MS Word (version 2, 6), Word Perfect, Open Office (version 1), MS Excel, MS PowerPoint, and PDF
- **Audio and Video:** WAV, MP3 (normal and with ID3Tags), BFW, FLAC
- **Markup languages:** HTML and XML
- **Internet files:** ARC
Each adapter registered in the system (config.xml) is responsible for parsing a specific input file type. Each adapter must extend the DataAdapter class, which defines several methods:
Adapters

Each adapter registered in the system (config.xml) is responsible for parsing a specific input file type. Each adapter must extend the DataAdapter class, which defines several methods:

- `acceptsFile(File)`
Adapters

Each adapter registered in the system (config.xml) is responsible for parsing a specific input file type. Each adapter must extend the DataAdapter class, which defines several methods:

- `acceptsFile(File)`
- `adapt(File, DataAdapter)`
Adapters

Each adapter registered in the system (config.xml) is responsible for parsing a specific input file type. Each adapter must extend the DataAdapter class, which defines several methods:

- `acceptsFile(File)`
- `adapt(File, DataAdapter)`
- `getOutputType()`
Adapters

Each adapter registered in the system (config.xml) is responsible for parsing a specific input file type. Each adapter must extend the DataAdapter class, which defines several methods:

- acceptsFile(File)
- adapt(File, DataAdapter)
- getOutputType()
- getInputType()
Outline

1. Overview
2. Metadata Extraction Tool
   - Overview
   - Demo
3. Hachoir
   - Overview
   - Demo
4. Sleuthkit
   - Overview
   - Demo
5. Summary
Demo

- use Metadata Extraction Tool in Windows Interface
  - doc, xls, ppt, pdf
  - mp3, wav,
  - jpeg, gif
Outline

1. Overview
2. Metadata Extraction Tool
   - Overview
   - Demo
3. Hachoir
   - Overview
   - Demo
4. Sleuthkit
   - Overview
   - Demo
5. Summary
Outline

1. Overview

2. Metadata Extraction Tool
   - Overview
   - Demo

3. Hachoir
   - Overview
   - Demo

4. Sleuthkit
   - Overview
   - Demo

5. Summary
Introduction

Hachoir

- a Python library that allows to view and edit a binary stream field by field
Introduction

Hachoir

- a Python library that allows to view and edit a binary stream field by field
- "browse" any binary stream just like you browse directories and files
Introduction

Hachoir

- a Python library that allows to view and edit a binary stream field by field
- "browse" any binary stream just like you browse directories and files
- a file is split in a tree of fields, where the smallest field is just one bit
Introduction

Hachoir

- a Python library that allows to view and edit a binary stream field by field
- "browse" any binary stream just like you browse directories and files
- a file is split in a tree of fields, where the smallest field is just one bit
- operating system independent and has many text/graphic user interfaces (ncurses, wxWidget, Gtk+)
Introduction

Hachoir

- a Python library that allows to view and edit a binary stream field by field
- "browse" any binary stream just like you browse directories and files
- a file is split in a tree of fields, where the smallest field is just one bit
- operating system independent and has many text/graphic user interfaces (ncurses, wxWidget, Gtk+)
- supports more than sixty file formats
Introduction

Hachoir
- a Python library that allows to view and edit a binary stream field by field
- "browse" any binary stream just like you browse directories and files
- a file is split in a tree of fields, where the smallest field is just one bit
- operating system independent and has many text/graphic user interfaces (ncurses, wxWidget, Gtk+)
- supports more than sixty file formats
- File format recognition is based on the headers and footers
### Supported File Formats

- **Archives:** bzip2, gzip, zip, tar
- **Audio:** MP3, WAV, Sun/NeXT audio, OGG, MIDI, AIFF, AIFC, RA
- **Video:** WMV video, AVI, MKV, MOV, Ogg/Theora, RM
- **Image:** BMP, CUR, EMF, ICO, GIF, JPEG, PCX, PNG, TGA, TIFF, WMF, XCF
- **Program:** EXE
- **Misc:** Torrent
Modules

- hachoirl-core: parser core
- hachoirl-parser: various file format parsers
- hachoirl-regex: regular expression (regex) manipulation library
Sample Program

- **hachoir-metadata**: extract metadata from video, music and other files
- **hachoir-grep**: find substring in a binary file (don’t parse non-string data)
- **hachoir-subfile**: find all subfiles in a file
- **hachoir-strip**: remove metadata and other "useless" informations
- **hachoir-urwid**: text user interface to explore a binary file
- **hachoir-wx**: graphical user interface to explore a binary file
- ...
1. Overview

2. Metadata Extraction Tool
   - Overview
   - Demo

3. Hachoir
   - Overview
   - Demo

4. Sleuthkit
   - Overview
   - Demo

5. Summary
use `hachoir-metadata` to extract metadata from some file formats (mp3, mov, jpeg etc.)

use `hachoir-subfile` to find specified files in iso image and disk

use `hachoir-grep` to find substrings in one file

use `hachoir-urwid` and `hachoir-wx` to explore binary files in text and graphics interface
**hachoir-metadata**

Usage: hachoir-metadata [options] files

Options:
- `-h, --help`  show this help message and exit
- `--type`  Only display file type (description)
- `--mime`  Only display MIME type
- `--level=LEVEL`  Quantity of information to display from 1 to 9 (9 is the maximum)
- `--raw`  Raw output
- `--bench`  Run benchmark
- `--force-parser=FORCE_PARSER`  List all parsers then exit
- `--parser-list`  List all parsers then exit
- `--profiler`  Run profiler
- `--version`  Display version and exit
- `--quality=QUALITY`  Information quality (0.0=fastest, 1.0=best, and default is 0.5)
- `--maxlen=MAXLEN`  Maximum string length in characters, 0 means unlimited (default: 300)
- `--verbose`  Verbose mode
- `--debug`  Debug mode
hachoir-subfile

Usage: hachoir-subfile [options] filename [output_directory]

Options:
-h, --help       show this help message and exit

hachoir-subfile:
  Option of hachoir-subfile

  --offset=OFFSET Skip first bytes of input file
  --size=SIZE     Maximum size of input file
  --category=CATEGORY Parser category list (separated with a comma)
  --parser=PARSER Parser identifier list (separated with a comma)
  --version       Display version and exit
  --quiet         Be quiet
  --profiler      Run profiler
  --debug         Enable debug mode
**hachoir-grep**

Usage: hachoir-grep [options] pattern filename [filename2 ...]

Options:
- `-h, --help` show this help message and exit

Grep:
- `--percent` Display percent
- `--no-addr` Don’t display address
- `--no-value` Don’t display value
- `--case` Search is case sensitive
- `--path` Display path
- `--all` Match all (just extract strings)
- `--bench` Run benchmark
- `--version` Display version and exit

Hachoir library:
- `--verbose` Verbose mode
- `--log=LOG` Write log in a file
- `--quiet` Quiet mode (don’t display warning)
- `--debug` Debug mode
hachoir-urwid

Options:
- `--help` show this help message and exit

Urwid:
Option of urwid explorer

- `--preload=PRELOAD` Number of fields to preload at each read
- `--path=PATH` Initial path to focus on
- `--parser=PARSER` Use the specified parser (use its identifier)
- `--offset=OFFSET` Skip first bytes of input file
- `--parser-list` List all parsers then exit
- `--profiler` Run profiler
- `--profile-display` Force update of the screen between each event
- `--size=SIZE` Maximum size of bytes of input file
- `--hide-value` Don’t display value
- `--hide-size` Don’t display size
- `--version` Display version and exit

Hachoir library:
Configure Hachoir library

- `--verbose` Verbose mode
- `--log=LOG` Write log in a file
- `--quiet` Quiet mode (don’t display warning)
- `--debug` Debug mode
Demo

hachoir-wx

/home/junysan/Desktop/Demo/hachoir/my.iso/

<table>
<thead>
<tr>
<th>address</th>
<th>name</th>
<th>type</th>
<th>size</th>
<th>data</th>
<th>descriptor</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>padding[0]</td>
<td>NullBytes</td>
<td>00001024</td>
<td>&lt;null&gt;</td>
<td></td>
</tr>
<tr>
<td>00000400</td>
<td>superblock/</td>
<td>SuperBlock</td>
<td>00000433</td>
<td></td>
<td>Super...</td>
</tr>
<tr>
<td>000005B1</td>
<td>padding[1]</td>
<td>NullBytes</td>
<td>00000591</td>
<td>&lt;null&gt;</td>
<td></td>
</tr>
<tr>
<td>00008800</td>
<td>group_desc/</td>
<td>GroupDescriptors</td>
<td>00001472</td>
<td></td>
<td>Group...</td>
</tr>
<tr>
<td>0000D00C</td>
<td>padding[2]</td>
<td>NullBytes</td>
<td>00262720</td>
<td></td>
<td>\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0{...}</td>
</tr>
<tr>
<td>00041000</td>
<td>group[0]</td>
<td>Group</td>
<td>00122360</td>
<td></td>
<td>Group...</td>
</tr>
<tr>
<td>00080000</td>
<td>group[1]</td>
<td>Group</td>
<td>00388608</td>
<td></td>
<td>Group...</td>
</tr>
<tr>
<td>01000000</td>
<td>group[2]</td>
<td>Group</td>
<td>00388608</td>
<td></td>
<td>Group...</td>
</tr>
<tr>
<td>01080000</td>
<td>group[3]</td>
<td>Group</td>
<td>00388608</td>
<td></td>
<td>Group...</td>
</tr>
<tr>
<td>02000000</td>
<td>group[4]</td>
<td>Group</td>
<td>00388608</td>
<td></td>
<td>Group...</td>
</tr>
<tr>
<td>02800000</td>
<td>group[5]</td>
<td>Group</td>
<td>00388608</td>
<td></td>
<td>Group...</td>
</tr>
<tr>
<td>03000000</td>
<td>group[6]</td>
<td>Group</td>
<td>00388608</td>
<td></td>
<td>Group...</td>
</tr>
<tr>
<td>03800000</td>
<td>group[7]</td>
<td>Group</td>
<td>00388608</td>
<td></td>
<td>Group...</td>
</tr>
<tr>
<td>04000000</td>
<td>group[8]</td>
<td>Group</td>
<td>00388608</td>
<td></td>
<td>Group...</td>
</tr>
<tr>
<td>04800000</td>
<td>group[9]</td>
<td>Group</td>
<td>00388608</td>
<td></td>
<td>Group...</td>
</tr>
<tr>
<td>05000000</td>
<td>group[10]</td>
<td>Group</td>
<td>00388608</td>
<td></td>
<td>Group...</td>
</tr>
</tbody>
</table>
Outline

1. Overview

2. Metadata Extraction Tool
   - Overview
   - Demo

3. Hachoir
   - Overview
   - Demo

4. Sleuthkit
   - Overview
   - Demo

5. Summary
1. Overview
2. Metadata Extraction Tool
   - Overview
   - Demo
3. Hachoir
   - Overview
   - Demo
4. Sleuthkit
   - Overview
   - Demo
5. Summary
Introduction

- the Sleuth Kit (TSK) is open source digital investigation tools
- run on Windows and Unix systems (such as Linux, OS X, Cygwin, FreeBSD, OpenBSD, and Solaris)
- analyze NTFS, FAT, HFS+, Ext2, Ext3, UFS1, and UFS2 file systems and several volume system types
- C library and a collection of command line tools
TSK library Layers

- **Base Layer** contains common programming and data structure functions that can be applied to all layers.
Overview

**TSK library Layers**

- **Base Layer** contains common programming and data structure functions that can be applied to all layers
- **Disk Image Layer** allows disk images in various formats to be opened and processed
TSK library Layers

- **Base Layer** contains common programming and data structure functions that can be applied to all layers.
- **Disk Image Layer** allows disk images in various formats to be opened and processed.
- **Volume System Layer** processes data as a volume system, such as DOS partition tables or BSD disk label structures.
TSK library Layers

- **Base Layer** contains common programming and data structure functions that can be applied to all layers
- **Disk Image Layer** allows disk images in various formats to be opened and processed
- **Volume System Layer** processes data as a volume system, such as DOS partition tables or BSD disk label structures
- **File System Layer** processes data as a file system, such as FAT or NTFS
TSK library Layers

- **Base Layer** contains common programming and data structure functions that can be applied to all layers.
- **Disk Image Layer** allows disk images in various formats to be opened and processed.
- **Volume System Layer** processes data as a volume system, such as DOS partition tables or BSD disk label structures.
- **File System Layer** processes data as a file system, such as FAT or NTFS.
- **Hash Database Layer** handles hash databases, such as NSRL and md5sum outputs.
TSK library Layers

- **Base Layer** contains common programming and data structure functions that can be applied to all layers
- **Disk Image Layer** allows disk images in various formats to be opened and processed
- **Volume System Layer** processes data as a volume system, such as DOS partition tables or BSD disk label structures
- **File System Layer** processes data as a file system, such as FAT or NTFS
- **Hash Database Layer** handles hash databases, such as NSRL and md5sum outputs
- **Automation Layer** integrates all of the previous layers in an automated fashion
Tools

- blkcalc
- blkcat
- blkls
- blkstat
- ffind
- fls
- fsstat
- hfind

- icat
- ifind
- ils
- img_cat
- img_stat
- istat
- jcat
- jls
- mactime

- mmcat
- mmstat
- sigfind
- sorter
- tsk_comparedir
- tsk_gettimes
- tsk_loaddb
- tsk_recover

www.sleuthkit.org/sleuthkit/man/
Outline

1. Overview
2. Metadata Extraction Tool
   - Overview
   - Demo
3. Hachoir
   - Overview
   - Demo
4. Sleuthkit
   - Overview
   - Demo
5. Summary
TSK Tools

- blkcat: display the contents of file system data unit in a disk image
- blkls: output the contents of unallocated data blocks by default
- blkcalc: converts between unallocated disk unit numbers and regular disk unit numbers
- ffind: find the name of the file or directory using a given inode
- fls: list file and directory names in a disk image
**TSK Tools**

- **blkcat**: display the contents of file system data unit in a disk image
### TSK Tools

- **blkcat**: display the contents of file system data unit in a disk image
- **blkls**: output the contents of unallocated data blocks by default
TSK Tools

- **blkcat**: display the contents of file system data unit in a disk image
- **blkls**: output the contents of unallocated data blocks by default
- **blkcalc**: converts between unallocated disk unit numbers and regular disk unit numbers
Demo

**TSK Tools**

- **blkcat**: display the contents of file system data unit in a disk image
- **blkls**: output the contents of unallocated data blocks by default
- **blkcalc**: converts between unallocated disk unit numbers and regular disk unit numbers
- **ffind**: find the name of the file or directory using a given inode
TSK Tools

- **blkcat**: display the contents of file system data unit in a disk image
- **blkls**: output the contents of unallocated data blocks by default
- **blkcalc**: converts between unallocated disk unit numbers and regular disk unit numbers
- **ffind**: find the name of the file or directory using a given inode
- **fls**: list file and directory names in a disk image
TSK Tools

- `fsstat`: display general details of a file system
Demo

TSK Tools

- `fsstat`: display general details of a file system
- `ifind`: find the meta-data structure that has allocated a given disk unit or file name
Demo

**TSK Tools**

- `fsstat`: display general details of a file system
- `ifind`: find the meta-data structure that has allocated a given disk unit or file name
- `istat`: display details of a meta-data structure (i.e. inode)
Demo

**TSK Tools**

- **fsstat**: display general details of a file system
- **ifind**: find the meta-data structure that has allocated a given disk unit or file name
- **istat**: display details of a meta-data structure (i.e. inode)
- **icat**: output the contents of a file based on its inode number
Demo

Linux Tools

- **strings**: print the strings of printable characters in files
- **grep**: print lines matching a pattern
- **dd**: copy a file, converting and formatting according to the operands
Demo

Search a unallocated space of `linux-0.2.img` for the string "abcdefg" and look up some metadata information in file system:
Demo

Search a unallocated space of `linux-0.2.img` for the string "abcdefg" and look up some metadata information in file system:

- extract the unallocated disk units from `linux-0.2.img` using `blkls` tool
Demo

Search a unallocated space of `linux-0.2.img` for the string "abcdefg" and look up some metadata information in file system:

- extract the unallocated disk units from `linux-0.2.img` using `blkls` tool (`linux.blkls`)

...
Demo

Search a unallocated space of `linux-0.2.img` for the string "abcdefg" and look up some metadata information in file system:

- extract the unallocated disk units from `linux-0.2.img` using `blkls` tool (`linux.blkls`)
- use `strings` utility to extract all of the ASCII strings from `linux.blkls`
Demo

Search a unallocated space of `linux-0.2.img` for the string "abcdefg" and look up some metadata information in file system:

- extract the unallocated disk units from `linux-0.2.img` using `blkls` tool (`linux.blkls`)
- use `strings` utility to extract all of the ASCII strings from `linux.blkls` (`linux.blkls.str`)
Search a unallocated space of `linux-0.2.img` for the string "abcdefg" and look up some metadata information in file system:

- extract the unallocated disk units from `linux-0.2.img` using `blkls` tool (`linux.blkls`)
- use `strings` utility to extract all of the ASCII strings from `linux.blkls` (`linux.blkls.str`)
- use `grep` to search "abcdefg" in `linux.blkls.str`
Demo

Search a unallocated space of linux-0.2.img for the string "abcdefg" and look up some metadata information in file system:

- extract the unallocated disk units from linux-0.2.img using blkls tool (linux.blkls)
- use strings utility to extract all of the ASCII strings from linux.blkls (linux.blkls.str)
- use grep to search "abcdefg" in linux.blkls.str (offset M (byte))
Demo

Search a unallocated space of `linux-0.2.img` for the string "abcdefg" and look up some metadata information in file system:

- extract the unallocated disk units from `linux-0.2.img` using `blkls` tool (`linux.blkls`)
- use `strings` utility to extract all of the ASCII strings from `linux.blkls` (`linux.blkls.str`)
- use `grep` to search "abcdefg" in `linux.blkls.str` (offset M (byte))
- identify which block "abcdefg" belongs to in `linux.blkls`
Search a unallocated space of linux-0.2.img for the string "abcdefg" and look up some metadata information in file system:

- extract the unallocated disk units from linux-0.2.img using blkls tool (linux.blkls)
- use strings utility to extract all of the ASCII strings from linux.blkls (linux.blkls.str)
- use grep to search "abcdefg" in linux.blkls.str (offset M (byte))
- identify which block "abcdefg" belongs to in linux.blkls (block N)
Search a unallocated space of `linux-0.2.img` for the string "abcdefg" and look up some metadata information in file system:

- extract the unallocated disk units from `linux-0.2.img` using `blkls` tool (`linux.blkls`)
- use `strings` utility to extract all of the ASCII strings from `linux.blkls` (`linux.blkls.str`)
- use `grep` to search "abcdefg" in `linux.blkls.str` (offset M (byte))
- identify which block "abcdefg" belongs to in `linux.blkls` (block N)
- use `blkcalc` to do block number mapping between `linux-0.2.img` and `linux.blkls`
Demo

Search a unallocated space of `linux-0.2.img` for the string "abcdefg" and look up some metadata information in file system:

- extract the unallocated disk units from `linux-0.2.img` using `blkls` tool (`linux.blkls`)
- use `strings` utility to extract all of the ASCII strings from `linux.blkls` (`linux.blkls.str`)
- use `grep` to search "abcdefg" in `linux.blkls.str` (offset M (byte))
- identify which block "abcdefg" belongs to in `linux.blkls` (block N)
- use `blkcalc` to do block number mapping between `linux-0.2.img` and `linux.blkls` (block K)
Demo

Search a unallocated space of `linux-0.2.img` for the string "abcdefg" and look up some metadata information in file system:

- extract the unallocated disk units from `linux-0.2.img` using `blkls` tool (`linux.blkls`)
- use `strings` utility to extract all of the ASCII strings from `linux.blkls` (`linux.blkls.str`)
- use `grep` to search "abcdefg" in `linux.blkls.str` (offset M (byte))
- identify which block "abcdefg" belongs to in `linux.blkls` (block N)
- use `blkcalc` to do block number mapping between `linux-0.2.img` and `linux.blkls` (block K)
- look up inode, file information etc. which are related to the file in which "abcdefg" is located
Search a unallocated space of `linux-0.2.img` for the string "abcdefg" and look up some metadata information in file system:

- extract the unallocated disk units from `linux-0.2.img` using `blkls` tool (`linux.blkls`)
- use `strings` utility to extract all of the ASCII strings from `linux.blkls` (`linux.blkls.str`)
- use `grep` to search "abcdefg" in `linux.blkls.str` (offset M (byte))
- identify which block "abcdefg" belongs to in `linux.blkls` (block N)
- use `blkcalc` to do block number mapping between `linux-0.2.img` and `linux.blkls` (block K)
- look up inode, file information etc. which are related to the file in which "abcdefg" is located (file names, inode info etc.)
Search a unallocated space of `linux-0.2.img` for the string "abcdefg" and look up some metadata information in file system:
Search a unallocated space of `linux-0.2.img` for the string "abcdefg" and look up some metadata information in file system:

- extract the unallocated disk units from `linux-0.2.img` using `blkls` tool
Search a unallocated space of `linux-0.2.img` for the string "abcdefg" and look up some metadata information in file system:

- extract the unallocated disk units from `linux-0.2.img` using `blkls` tool (`linux.blkls`)
Search a unallocated space of `linux-0.2.img` for the string "abcdefg" and look up some metadata information in file system:

- extract the unallocated disk units from `linux-0.2.img` using `blkls` tool (`linux.blkls`)
- use `strings` utility to extract all of the ASCII strings from `linux.blkls`
Search a unallocated space of linux-0.2.img for the string "abcdefg" and look up some metadata information in file system:

- extract the unallocated disk units from linux-0.2.img using blkls tool (linux.blkls)
- use strings utility to extract all of the ASCII strings from linux.blkls (linux.blkls.str)
Search a unallocated space of `linux-0.2.img` for the string "abcdefg" and look up some metadata information in file system:

- extract the unallocated disk units from `linux-0.2.img` using `blkls` tool (`linux.blkls`)
- use `strings` utility to extract all of the ASCII strings from `linux.blkls` (`linux.blkls.str`)
- use `grep` to search "abcdefg" in `linux.blkls.str`
Demo

Search a unallocated space of `linux-0.2.img` for the string "abcdefg" and look up some metadata information in file system:

- extract the unallocated disk units from `linux-0.2.img` using `blkls` tool (`linux.blkls`)
- use `strings` utility to extract all of the ASCII strings from `linux.blkls` (`linux.blkls.str`)
- use `grep` to search "abcdefg" in `linux.blkls.str` (offset $M = 15392$ (byte))
Search a unallocated space of `linux-0.2.img` for the string "abcdefg" and look up some metadata information in file system:
Demo

Search a unallocated space of `linux-0.2.img` for the string "abcdefg" and look up some metadata information in file system:

- identify which block "abcdefg" belongs to in `linux.blkls`
Search a unallocated space of \texttt{linux-0.2.img} for the string "abcdefg" and look up some metadata information in file system:

- identify which block "abcdefg" belongs to in \texttt{linux.blkls} (block $N = 15$)
Search a unallocated space of linux-0.2.img for the string "abcdefg" and look up some metadata information in file system:

- identify which block "abcdefg" belongs to in linux.blkls (block N = 15)
- use blkcalc to do block number mapping between linux-0.2.img and linux.blkls
Search a unallocated space of `linux-0.2.img` for the string "abcdefg" and look up some metadata information in file system:

- identify which block "abcdefg" belongs to in `linux.blkls` (block $N = 15$)
- use `blkcalc` to do block number mapping between `linux-0.2.img` and `linux.blkls` (block $K = 17804$)
Search a unallocated space of `linux-0.2.img` for the string "abcdefg" and look up some metadata information in file system:

- identify which block "abcdefg" belongs to in `linux.blkls` (block N = 15)
- use `blkcalc` to do block number mapping between `linux-0.2.img` and `linux.blkls` (block K = 17804)
- look up inode, file information etc. which are related to the file in which "abcdefg" is located
Search a unallocated space of `linux-0.2.img` for the string "abcdefg" and look up some metadata information in file system:

- identify which block "abcdefg" belongs to in `linux.blkls` (block $N = 15$
- use `blkcalc` to do block number mapping between `linux-0.2.img` and `linux.blkls` (block $K = 17804$
- look up inode, file information etc. which are related to the file in which "abcdefg" is located (inode=1857 */$/OrphanFiles/OrphanFile-1857)
Outline

1. Overview

2. Metadata Extraction Tool
   - Overview
   - Demo

3. Hachoir
   - Overview
   - Demo

4. Sleuthkit
   - Overview
   - Demo

5. Summary
Summary

Similarity:
- open source library and can be extended
Summary

Similarity:

- open source library and can be extended
- used to extract metadata from files
Difference:

- single file, file systems and both
Difference:
- single file, file systems and both
- developing language: Java, Python and C
Summary

Difference:

- single file, file systems and both
- developing language: Java, Python and C
- platform independent: hachoir and Metadata Extraction Tool
Summary

- meta-extractor.sourceforge.net/
- bitbucket.org/haypo/hachoir/wiki/Home
- sleuthkit.org/