Ungraded Pre-Lecture Quiz

- In digital forensics field, what's the **difference** between disk *copying*, *cloning* and *imaging*?
- According to the **Order of Volatility (OOV) theory**, the first 3 most volatile evidence categories are:
  (1) CPU, cache and register content;
  (2) Routing table, ARP cache, process table, kernel statistics;
  (3) Memory.
  Why isn't "Memory" the most volatile category?

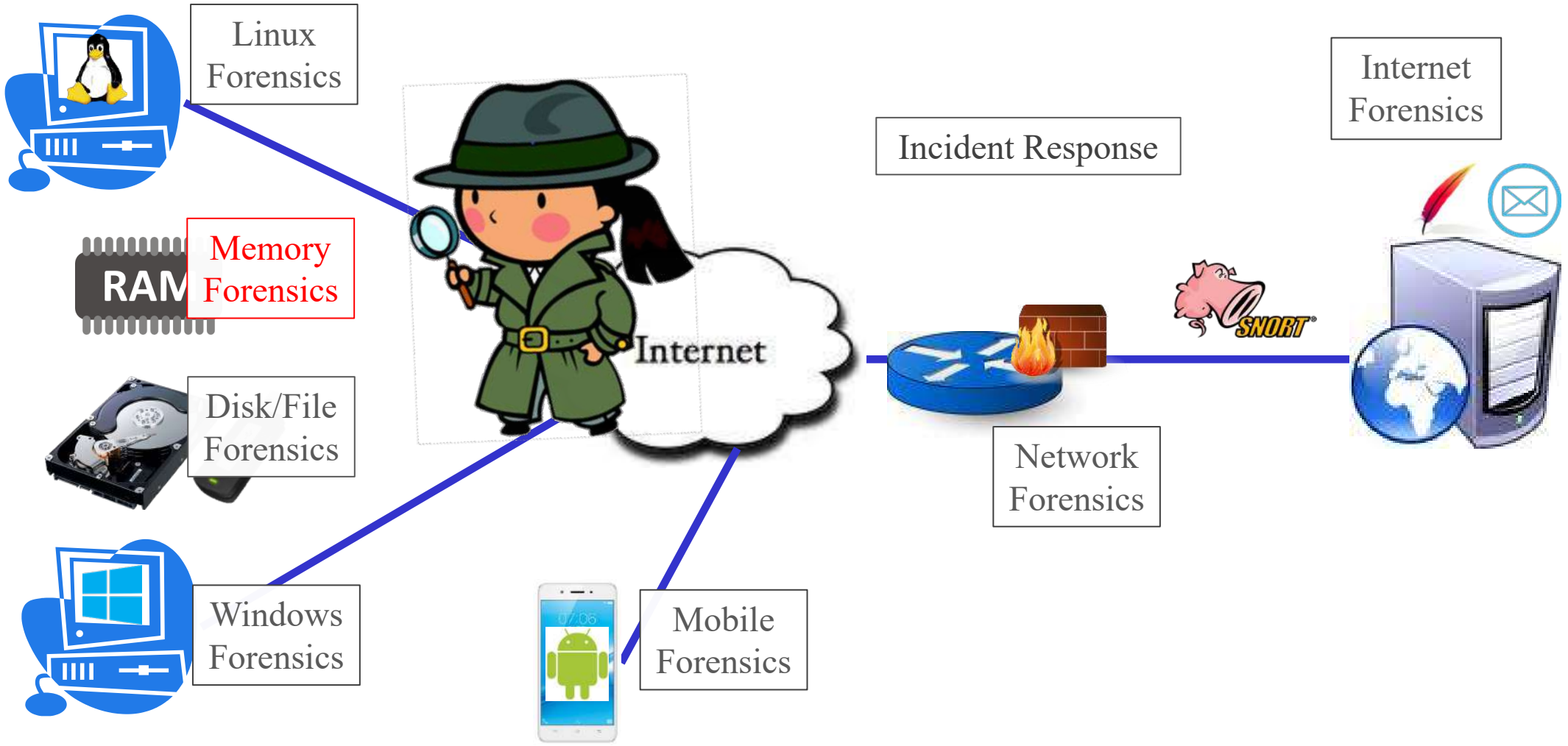
IFS4102: Digital Forensics

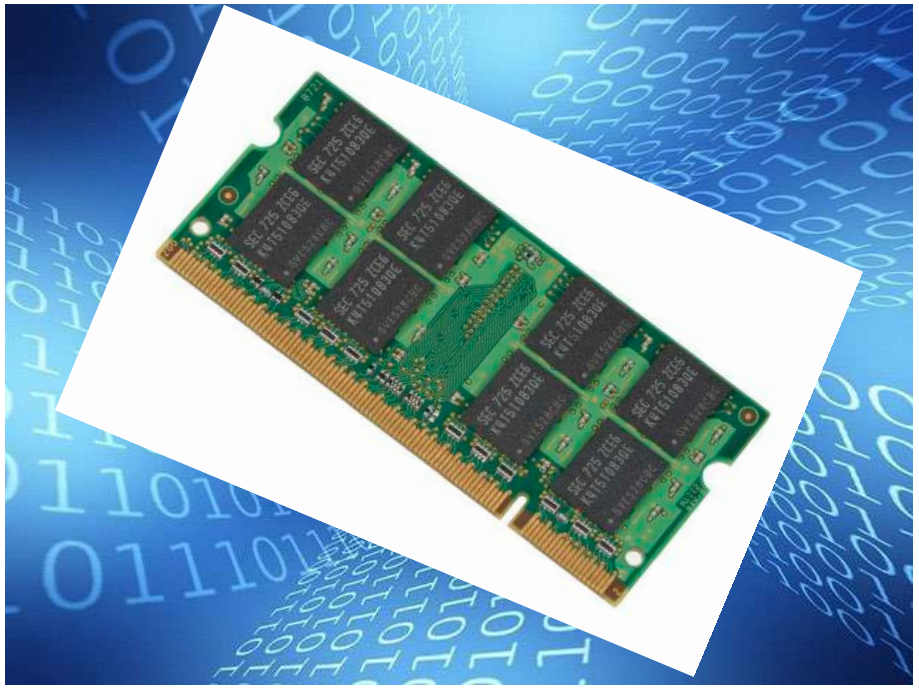
Lecture 3: Live/Volatile Acquisition, Image Analyses

Outline

- Live/volatile acquisition
- Memory image analysis
- Further on disk image acquisition & analysis
- Static acquisition challenges
- Forensic analysis using Autopsy
- Lab 3 exercises

Memory Forensics





Live/Volatile Acquisition

Live/Volatile Acquisition

- **Live/volatile acquisition**:
  - Will dump a target machine's **physical memory** (RAM)
  - May be considered **necessary** nowadays, since many users **encrypt** their hard drives or use SSDs (*more on SSDs later*)
- General **steps**:
  - Prepare the acquisition software's **executable files** in a **thumb drive**
  - **Invoke** the acquisition software from the drive (*as administrator*)
  - Output the memory image file into the **thumb drive** or another connected **external storage** device, *not* the machine's HDD!
  - Your Lab 3 Task-1 exercises!

Live Acquisition of a Windows Machine

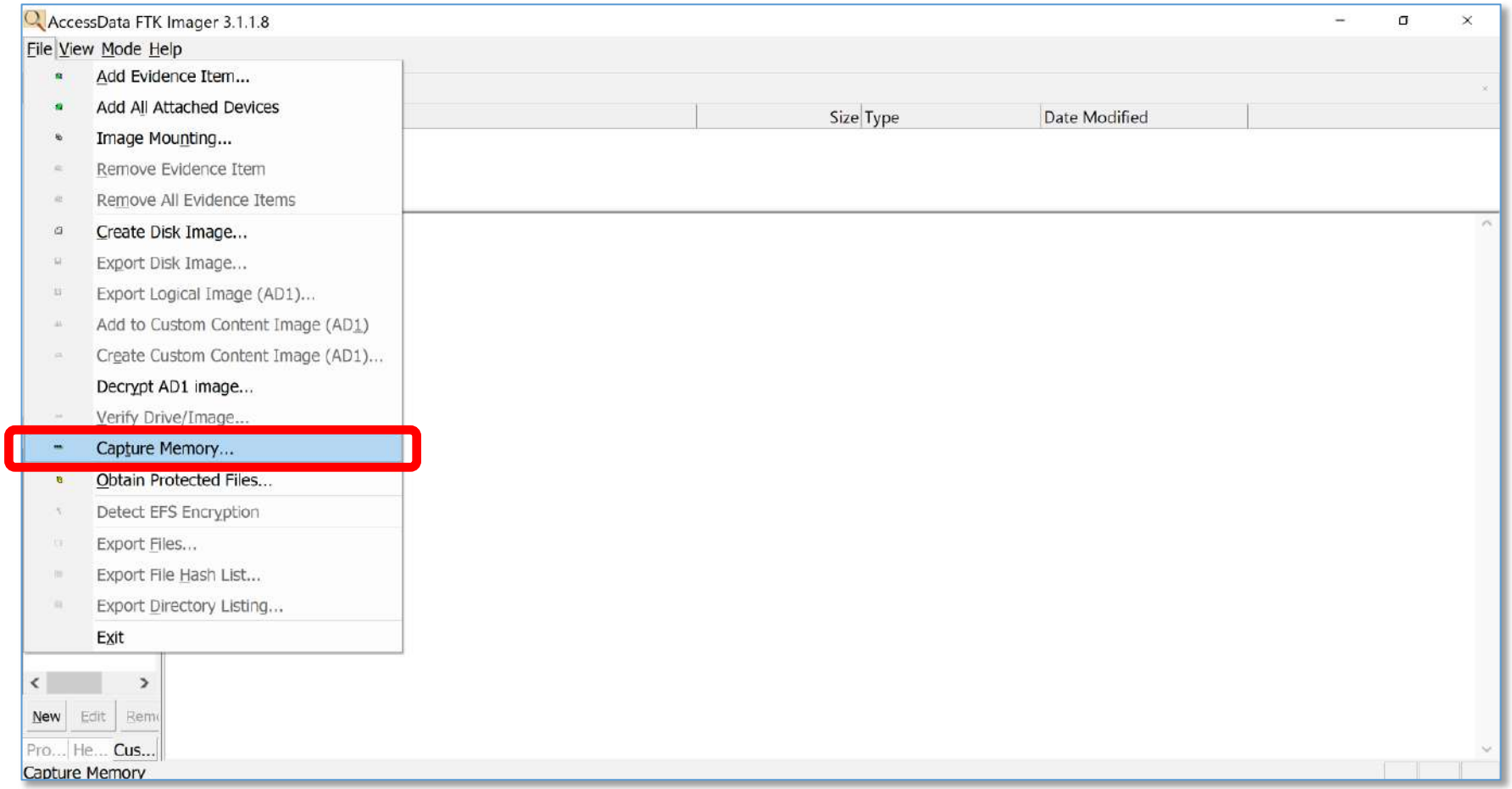
- Useful acquisition software:
  - **FTK Imager Lite**:
    - A stand-alone imaging tool/set-up from AccessData: *no* installation is required
    - See Lab 3, Task 1-A
  - **FireEye/Mandiant Memoryze**:
    - A CLI-based acquisition executable: see Lab 3, Task 1-B
    - Can also perform various **live analysis tasks** on a running computer
  - **MoonSols' DumpIt**:
    - Another CLI-based acquisition tool (check out https://www.comae.com/)

Live Acquisition of a Windows Machine

- **WinPmem** (part of the Pmem Suite):
  - See https://winpmem.velocidex.com/
- (Commercial) **Live Response** from E-fense: uses a **USB key**
  - See http://malwarefieldguide.com/LinuxChapter1.html

FTK Imager Lite

- FTK Imager Lite: a Windows-based tool



FireEye/Mandiant Memoryze

- A powerful set of ***memory analysis tools***: “find evil in live memory”
- Some features:
  - **Acquire** and/or **analyze** memory images
  - Enumerate all running processes (including those hidden by rootkits)
  - Identify all loaded kernel modules by walking a linked list
  - …
- Memoryze takes **XML** documents that define what to do, and then outputs the result in **XML** format
- Each XML script has been wrapped by a corresponding **batch file**

FireEye/Mandiant Memoryze

- Some included **batch** files:
  - ***MemoryDD.bat***: acquires an image of **physical memory**
  - **ProcessDD.bat**: acquires an image of the process' address space
  - **DriverDD.bat**: acquires an image of a driver
  - **Process.bat**: enumerates everything about a process including handles, virtual memory, network ports, and strings
  - **HookDetection.bat**: looks for hooks within the OS
  - **DriverSearch.bat**: finds drivers
  - **DriverWalkList.bat**: enumerates all modules and drivers in a linked list

FireEye/Mandiant Memoryze in Action

```
F:\Memoryze\Audits> cd EXADATA

F:\Memoryze\Audits\EXADATA> dir

 Volume in drive F is SAMSUNG
 Volume Serial Number is 3243-30C2

 Directory of F:\Memoryze\Audits\EXADATA

11/06/2014  13:51    <DIR>          .
11/06/2014  13:51    <DIR>          ..
11/06/2014  13:51    <DIR>          20140611165146
               0 File(s)              0 bytes
               3 Dir(s)  535.223.562.240 bytes free

F:\Memoryze\Audits\EXADATA> cd 20140611165146

F:\Memoryze\Audits\EXADATA\20140611165146> dir
 Volume in drive F is SAMSUNG
 Volume Serial Number is 3243-30C2

 Directory of F:\Memoryze\Audits\EXADATA\20140611165146

11/06/2014  13:51    <DIR>          .
11/06/2014  13:51    <DIR>          ..
11/06/2014  13:51            20.056 BatchResults.xml
11/06/2014  13:51               283 Issues.BatchResults.xml
11/06/2014  13:55             2.172
issues.memory.4d021d38.img.xml
11/06/2014  13:55    17.951.621.120 memory.4d021d38.img
               4 File(s) 17.951.643.631 bytes
               2 Dir(s)  535.223.562.240 bytes free
```

From: A. Borges, "*Memory Acquisition for Forensic Memory Analysis on Windows and Linux*", 2014

FireEye Memoryze: Possible Acquisition Issues

- Possible **issues** during live acquisition:
  - Memoryze requires **loading a kernel-level driver** that gives access to raw memory: *no* driver, *no* memory image
  - Several things can **prevent** the driver from being loaded:
    - Not being run with an **Admin account** (or an Admin-level command prompt): the most common cause
    - Anti-virus software running on the target system
- You may see some **error/warning messages** during your acquisition
- Reference: https://www.sans.org/blog/digital-forensics-how-to-memory-analysis-with-mandiant-memoryze/

MoonSols' DumpIt in Action

```
F:\DumpIt> DumpIt.exe

  DumpIt - v1.3.2.20110401 - One click memory memory dumper
  Copyright (c) 2007 - 2011, Matthieu Suiche
<http://www.msuiche.net>
  Copyright (c) 2010 - 2011, MoonSols <http://www.moonsols.com>


    Address space size:       17951621120 bytes (  17120 Mb)
    Free space size:         571126833152 bytes ( 544668 Mb)

    * Destination = \??\F:\DumpIt\EXADATA-20140611-164112.raw

    --> Are you sure you want to continue? [y/n] y
    + Processing... Success.

F:\DumpIt>dir

   Volume in drive F is SAMSUNG
   Volume Serial Number is 3243-30C2

   Directory of F:\DumpIt

  11/06/2014  13:41    <DIR>          .
  11/06/2014  13:41    <DIR>          ..
  03/05/2011  02:41               207.496 DumpIt.exe
  11/06/2014  13:44    17.951.621.120 EXADATA-20140611-164112.raw
  18/07/2011  08:29               743 README.txt
                 3 File(s) 17.951.829.359 bytes
                 2 Dir(s)  535.191.465.984 bytes free

  F:\DumpIt>
```

From: A. Borges, "*Memory Acquisition for Forensic Memory Analysis on Windows and Linux*", 2014

WinPmem

- Download it from: https://github.com/Velocidex/WinPmem/releases
- It used to be under the **Rekall project**:
  - A fork of Volatility: http://blog.rekall-forensic.com/
  - The project is discontinued now
- To write a raw image to `physmem.raw`: `winpmem_1.6.0.exe physmem.raw`

```
c:\..> winpmem_1.6.0.exe -h
Winpmem - A memory imager for windows.
Copyright Michael Cohen (scudette@gmail.com) 2012-2014.

Version 1.6.0 May 15 2014
Usage:
  winpmem_1.6.0.exe [option] [output path]

Option:
  -l    Load the driver and exit.
  -u    Unload the driver and exit.
  -d [filename]
        Extract driver to this file (Default use random name).
  -h    Display this help.
  -w    Turn on write mode.
  -0    Use MmMapIoSpace method.
  -1    Use \\Device\PhysicalMemory method (Default for 32bit OS).
  -2    Use PTE remapping (AMD64 only - Default for 64bit OS).
  -3    Use PTE remapping with PCI introspection (AMD64 Only).
  -e    Produce an ELF core dump.

NOTE: an output filename of - will write the image to STDOUT.

Examples:
winpmem_1.6.0.exe physmem.raw
Writes an image to physmem.raw

winpmem_1.6.0.exe -e - | nc 192.168.1.1 80
Writes an elf coredump to netcat for network transport.
```

From: https://rekall.readthedocs.io/en/gh-pages/Tools/pmem.html

Live Acquisition of a *Linux* Machine

- Useful acquisition software:
  - **LiME (Linux Memory Extraction, formerly DMD)**:
    - A **Loadable Kernel Module (LKM)**, which allows for volatile memory acquisition from Linux & Linux-based devices (e.g. Android)
    - Utilizes the **insmod** command to load the module, passing required arguments for its execution
    - Important **requirement**: ***OS type and kernel dependant*** → it must be built for the target machine's **specific kernel version**
    - Reference: https://markuta.com/live-memory-acquisition-on-linux-systems/
    - Watch LiME in action: https://www.youtube.com/watch?v=_7Tq8dcmP0k

LiME in Action

```
root@hacker:~/LiMe/src# make

make -C /lib/modules/3.7-trunk-amd64/build M=/root/LiMe/src
modules
make[1]: Entering directory `/usr/src/linux-headers-3.7-trunk-
amd64'
  CC [M]  /root/LiMe/src/tcp.o
  CC [M]  /root/LiMe/src/disk.o
  CC [M]  /root/LiMe/src/main.o
/root/LiMe/src/main.c: In function '__check_dio':
/root/LiMe/src/main.c:56:1: warning: return from incompatible
pointer type [enabled by default]
  LD [M]  /root/LiMe/src/lime.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC      /root/LiMe/src/lime.mod.o
  LD [M]  /root/LiMe/src/lime.ko
make[1]: Leaving directory `/usr/src/linux-headers-3.7-trunk-
amd64'
strip --strip-unneeded lime.ko
mv lime.ko lime-3.7-trunk-amd64.ko
make tidy
make[1]: Entering directory `/root/LiMe/src'
rm -f *.o *.mod.c Module.symvers Module.markers modules.order
\.*.o.cmd \.*.ko.cmd \.*.o.d
rm -rf \.tmp_versions
make[1]: Leaving directory `/root/LiMe/src'

root@hacker:~/LiMe/src#
```

From: A. Borges, "*Memory Acquisition for Forensic Memory Analysis on Windows and Linux*", 2014

LiME in Action

```
root@hacker:/media/pendrive# insmod /media/pendrive/lime-3.7-
trunk-amd64.ko "path=/media/external_drive/kali_memory_dump.bin
format=lime"

root@hacker:/media/pendrive# cd /media/external_drive

root@hacker:/media/external_drive# ls -lh kali_memory_dump.bin

  -r--r--r-- 1 root root 18G Jun 11 01:55 kali_memory_dump.bin
```

From: A. Borges, "*Memory Acquisition for Forensic Memory Analysis on Windows and Linux*", 2014

Live Acquisition of a Linux Machine

- **LinPmem** (part of the Pmem Suite):
  See https://winpmem.velocidex.com/,
  https://github.com/google/rekall/tree/master/tools/pmem
- **Fmem**: https://github.com/NateBrune/fmem
- Older & limited tool: **Memdump**
  (http://www.porcupine.org/forensics/tct.html)

- For a **comparison** of Linux live-acquisition tools, see:
  Carbone and Bourdon-Richard, "*The definitive guide to Linux-based live memory acquisition tools*", Defence R&D Canada, 2013
  (https://publications.gc.ca/collections/collection_2016/rddc-drdc/D68-6-319-2012-eng.pdf)

Memory Image Analysis

Analysis of Memory Image File

- We have completed a live/volatile memory acquisition of a target machine
- *Question: How can we **analyze** the **memory dump/image* file**?*
- Some available tools:
  - **Volatility**: See Lab 3 Task 2-A
  - Other **memory analyzer tools**, e.g. **FireEye Redline**
  - **FTK Imager**: See Lab 3 Tasks 2-B (for string searching)
  - A **hex editor**: See Lab 3 Tasks 2-C (for string searching)

*__Note__: Some forensics experts do *not* like the term memory "**image**" since the RAM memory was in constant flux during its acquisition, and prefer the term "**dump**" instead

Volatility

- A popular open-source CLI-based **memory image analysis tool**
- Can analyse **Windows, Linux, Mac** memory images
- Extensible and scriptable API
- Various available **plugins**: https://github.com/volatilityfoundation/volatility/wiki/Command-Reference

- **Note**: Volatility is *not* a memory acquisition tool
- See Lab 3 Task 2-A, and also your Assignment 1 (*see also the next slide*)
- **Rekall**: a fork of Volatility, but the project is discontinued

Volatility Versions

- **Version 3** (released in 2020):
  - A complete rewrite of the framework
  - Released under the Volatility Software License (VSL)
  - New **features**: see Volatility 3 Public Beta: Insider's Preview, https://www.volexity.com/wp-content/uploads/2020/11/Volexity-Cyber-Session-April-2020-Volatility3-Public-Beta.pdf
  - **Usage**: see "Introduction to Memory Forensics with Volatility 3" video (https://www.youtube.com/watch?v=Uk3DEgY5Ue8)
- **Version 2.6**:
  - Simple **standalone executable** for Windows
  - Can analyze Windows, Linux (`linux_*`), Mac (`mac_*`) memory images
  - Used in our **lab** and possibly **mid-term** practical test!

Volatility 2.6: General Commands

- **Syntax** and typical **command components**:
  `# vol.py -f [image] --profile=[profile] [plugin]`
- Display profiles, address spaces, plugins:
  `# vol.py –info`
- Display global command-line options:
  `# vol.py –help`
- Display plugin-specific arguments:
  `# vol.py [plugin] --help`

Volatility 2.6: Some Commands (Windows)

- **Image identification**:
  - `imageinfo`: gets a high-level summary of the memory sample, including **profile suggestions** (OS and architecture)

```
$ python vol.py -f ~/Desktop/win7_trial_64bit.raw imageinfo
Volatility Foundation Volatility Framework 2.4
Determining profile based on KDBG search...

          Suggested Profile(s) : Win7SP0x64, Win7SP1x64, Win2008R2SP0x64, Win2008R2SP1x64
                     AS Layer1 : AMD64PagedMemory (Kernel AS)
                     AS Layer2 : FileAddressSpace (/Users/Michael/Desktop/win7_trial_64bit.raw)
                      PAE type : PAE
                           DTB : 0x187000L
                          KDBG : 0xf80002803070
          Number of Processors : 1
     Image Type (Service Pack) : 0
                KPCR for CPU 0 : 0xfffff80002804d00L
             KUSER_SHARED_DATA : 0xfffff78000000000L
           Image date and time : 2012-02-22 11:29:02 UTC+0000
     Image local date and time : 2012-02-22 03:29:02 -0800
```

Source: https://github.com/volatilityfoundation/volatility/wiki/Command-Reference

Volatility 2.6: Some Commands (Windows)

- **Image identification**:
  - `kdbgscan`: positively identifies the correct profile and the correct **KDBG** (Kernel Debugger Block/_KDDEBUGGER_DATA64) address

```
$ python vol.py -f Win2K3SP2x64-6f1bedec.vmem --profile=Win2003SP2x64 kdbgscan
Volatility Foundation Volatility Framework 2.4
**************************************************
Instantiating KDBG using: Kernel AS Win2003SP2x64 (5.2.3791 64bit)
Offset (V)                    : 0xf80001172cb0
Offset (P)                    : 0x1172cb0
KDBG owner tag check          : True
Profile suggestion (KDBGHeader): Win2003SP2x64
Version64                     : 0xf80001172c70 (Major: 15, Minor: 3790)
Service Pack (CmNtCSDVersion) : 0
Build string (NtBuildLab)     : T?
PsActiveProcessHead           : 0xfffff800011947f0 (0 processes)
PsLoadedModuleList            : 0xfffff80001197ac0 (0 modules)
KernelBase                    : 0xfffff80001000000 (Matches MZ: True)
Major (OptionalHeader)        : 5
Minor (OptionalHeader)        : 2

**************************************************
Instantiating KDBG using: Kernel AS Win2003SP2x64 (5.2.3791 64bit)
Offset (V)                    : 0xf80001175cf0
Offset (P)                    : 0x1175cf0
KDBG owner tag check          : True
Profile suggestion (KDBGHeader): Win2003SP2x64
Version64                     : 0xf80001175cb0 (Major: 15, Minor: 3790)
Service Pack (CmNtCSDVersion) : 2
Build string (NtBuildLab)     : 3790.srv03_sp2_rtm.070216-1710
PsActiveProcessHead           : 0xfffff800011977f0 (37 processes)
PsLoadedModuleList            : 0xfffff8000119aae0 (116 modules)
KernelBase                    : 0xfffff80001000000 (Matches MZ: True)
Major (OptionalHeader)        : 5
Minor (OptionalHeader)        : 2
KPCR                          : 0xfffff80001177000 (CPU 0)
```

Source:
https://github.com/volatilityfoundation/volatility/wiki/Command-Reference

Volatility 2.6: Some Commands (Windows)

- **Processes & listings**:
  - `pslist`: lists the **processes** of a system

```
$ python vol.py -f ~/Desktop/win7_trial_64bit.raw --profile=Win7SP0x64 pslist
Volatility Foundation Volatility Framework 2.4
Offset(V)          Name                    PID   PPID   Thds     Hnds   Sess  Wow64 Start                 Exit
------------------ -------------------- ------ ------ ------ -------- ------ ------ ------------------- -------------
0xfffffa80004b09e0 System                    4      0     78      489 ------      0 2012-02-22 19:58:20
0xfffffa8000ce97f0 smss.exe                208      4      2       29 ------      0 2012-02-22 19:58:20
0xfffffa8000c006c0 csrss.exe               296    288      9      385      0      0 2012-02-22 19:58:24
0xfffffa8000c92300 wininit.exe             332    288      3       74      0      0 2012-02-22 19:58:30
0xfffffa8000c06b30 csrss.exe               344    324      7      252      1      0 2012-02-22 19:58:30
0xfffffa8000c80b30 winlogon.exe            372    324      5      136      1      0 2012-02-22 19:58:31
0xfffffa8000c5eb30 services.exe            428    332      6      193      0      0 2012-02-22 19:58:32
0xfffffa80011c5700 lsass.exe               444    332      6      557      0      0 2012-02-22 19:58:32
0xfffffa8000ea31b0 lsm.exe                 452    332     10      133      0      0 2012-02-22 19:58:32
0xfffffa8001296b30 svchost.exe             568    428     10      352      0      0 2012-02-22 19:58:34
0xfffffa80012c3620 svchost.exe             628    428      6      247      0      0 2012-02-22 19:58:34
0xfffffa8001325950 sppsvc.exe              816    428      5      154      0      0 2012-02-22 19:58:41
0xfffffa80007b7960 svchost.exe             856    428     16      404      0      0 2012-02-22 19:58:43
0xfffffa80007bb750 svchost.exe             880    428     34     1118      0      0 2012-02-22 19:58:43
0xfffffa80007d09e0 svchost.exe             916    428     19      443      0      0 2012-02-22 19:58:43
0xfffffa8000c64840 svchost.exe             348    428     14      338      0      0 2012-02-22 20:02:07
0xfffffa8000c09630 svchost.exe             504    428     16      496      0      0 2012-02-22 20:02:07
0xfffffa8000e86690 spoolsv.exe            1076    428     12      271      0      0 2012-02-22 20:02:10
0xfffffa8000518b30 svchost.exe            1104    428     18      307      0      0 2012-02-22 20:02:10
```

Source: https://github.com/volatilityfoundation/volatility/wiki/Command-Reference

Volatility 2.6: Some Commands (Windows)

- **Processes & listings**:
  - `pstree`: views the **process** listing in **tree form**

```
$ python vol.py -f ~/Desktop/win7_trial_64bit.raw --profile=Win7SP0x64 pstree
Volatility Foundation Volatility Framework 2.4
Name                                                  Pid   PPid   Thds   Hnds Time
-------------------------------------------------- ------ ------ ------ ------ ----------
 0xfffffa80004b09e0:System                              4      0     78    489 2012-02-22 19:58:20
. 0xfffffa8000ce97f0:smss.exe                         208      4      2     29 2012-02-22 19:58:20
 0xfffffa8000c006c0:csrss.exe                         296    288      9    385 2012-02-22 19:58:24
 0xfffffa8000c92300:wininit.exe                       332    288      3     74 2012-02-22 19:58:30
. 0xfffffa8000c5eb30:services.exe                     428    332      6    193 2012-02-22 19:58:32
.. 0xfffffa8000aa0b30:SearchIndexer.                 1800    428     12    757 2012-02-22 20:02:26
.. 0xfffffa80007d09e0:svchost.exe                     916    428     19    443 2012-02-22 19:58:43
.. 0xfffffa8000a4f630:svchost.exe                    1432    428     12    350 2012-02-22 20:04:14
.. 0xfffffa800094d960:wlms.exe                       1264    428      4     43 2012-02-22 20:02:11
.. 0xfffffa8001325950:sppsvc.exe                      816    428      5    154 2012-02-22 19:58:41
.. 0xfffffa8000e86690:spoolsv.exe                    1076    428     12    271 2012-02-22 20:02:10
.. 0xfffffa8001296b30:svchost.exe                     568    428     10    352 2012-02-22 19:58:34
... 0xfffffa8000a03b30:rundll32.exe                  2016    568      3     67 2012-02-22 20:03:16
...
```

Source: https://github.com/volatilityfoundation/volatility/wiki/Command-Reference

Volatility 2.6: Some Commands (Windows)

- **Processes & listings**:
  - `psscan`: finds processes that previously **terminated** (inactive) and processes that have been **hidden** or **unlinked** by a rootkit

```
$ python vol.py --profile=Win7SP0x86 -f win7.dmp psscan
Volatility Foundation Volatility Framework 2.0
 Offset     Name             PID    PPID   PDB        Time created             Time exited
---------- ---------------- ------ ------ ---------- ------------------------ ------------------------
0x3e025ba8 svchost.exe        1116    508 0x3ecf1220 2010-06-16 15:25:25
0x3e04f070 svchost.exe        1152    508 0x3ecf1340 2010-06-16 15:27:40
0x3e144c08 dwm.exe            1540    832 0x3ecf12e0 2010-06-16 15:26:58
0x3e145c18 TPAutoConnSvc.     1900    508 0x3ecf1360 2010-06-16 15:25:41
0x3e3393f8 lsass.exe           516    392 0x3ecf10e0 2010-06-16 15:25:18
0x3e35b8f8 svchost.exe         628    508 0x3ecf1120 2010-06-16 15:25:19
0x3e383770 svchost.exe         832    508 0x3ecf11a0 2010-06-16 15:25:20
0x3e3949d0 svchost.exe         740    508 0x3ecf1160 2010-06-16 15:25:20
0x3e3a5100 svchost.exe         872    508 0x3ecf11c0 2010-06-16 15:25:20
0x3e3f64e8 svchost.exe         992    508 0x3ecf1200 2010-06-16 15:25:24
0x3e45a530 wininit.exe         392    316 0x3ecf10a0 2010-06-16 15:25:15
0x3e45d928 svchost.exe        1304    508 0x3ecf1260 2010-06-16 15:25:28
0x3e45f530 csrss.exe           400    384 0x3ecf1040 2010-06-16 15:25:15
0x3e4d89c8 vmtoolsd.exe       1436    508 0x3ecf1280 2010-06-16 15:25:30
0x3e4db030 spoolsv.exe        1268    508 0x3ecf1240 2010-06-16 15:25:28
0x3e50b318 services.exe        508    392 0x3ecf1080 2010-06-16 15:25:18
0x3e7f3d40 csrss.exe           352    316 0x3ecf1060 2010-06-16 15:25:12
0x3e7f5bc0 winlogon.exe        464    384 0x3ecf10c0 2010-06-16 15:25:18
0x3eac6030 SearchProtocol     2448   1168 0x3ecf15c0 2010-06-16 23:30:52      2010-06-16 23:33:14
0x3eb10030 SearchFilterHo     1812   1168 0x3ecf1480 2010-06-16 23:31:02      2010-06-16 23:33:14
[snip]
```

Source: https://github.com/volatilityfoundation/volatility/wiki/Command-Reference

Volatility 2.6: Some Commands (Windows)

- **Process & information** (with `-o|--offset` or `-p|--pid`):
  - `dlllist`: displays **DLLs**

```
$ python vol.py -f ~/Desktop/win7_trial_64bit.raw --profile=Win7SP0x64 dlllist
**************************************************************************
wininit.exe pid:    332
Command line : wininit.exe

Base                             Size          LoadCount Path
------------------ ------------------ ------------------ ----
0x00000000ff530000            0x23000              0xffff C:\Windows\system32\wininit.exe
0x0000000076d40000           0x1ab000              0xffff C:\Windows\SYSTEM32\ntdll.dll
0x0000000076b20000           0x11f000              0xffff C:\Windows\system32\kernel32.dll
0x000007fefcd50000            0x6b000              0xffff C:\Windows\system32\KERNELBASE.dll
0x0000000076c40000            0xfa000              0xffff C:\Windows\system32\USER32.dll
0x000007fefd7c0000            0x67000              0xffff C:\Windows\system32\GDI32.dll
0x000007fefe190000             0xe000              0xffff C:\Windows\system32\LPK.dll
0x000007fefef80000            0xca000              0xffff C:\Windows\system32\USP10.dll
0x000007fefd860000            0x9f000              0xffff C:\Windows\system32\msvcrt.dll
[snip]
```

Source: https://github.com/volatilityfoundation/volatility/wiki/Command-Reference

Volatility 2.6: Some Commands (Windows)

- **Process & information** (with `-o|--offset` or `-p|--pid`):
  - `dlllist`: displays DLLs of a **specific process**

```
$ python vol.py -f ~/Desktop/win7_trial_64bit.raw --profile=Win7SP0x64 dlllist -p 1892
Volatility Foundation Volatility Framework 2.4
************************************************************************
iexplore.exe pid:   1892
Command line : "C:\Program Files (x86)\Internet Explorer\iexplore.exe"
Note: use ldrmodules for listing DLLs in Wow64 processes

Base                             Size          LoadCount Path
------------------ ------------------ ------------------ ----
0x0000000000080000           0xa6000             0xffff C:\Program Files (x86)\Internet Explorer\iexplore.exe
0x0000000076d40000          0x1ab000             0xffff C:\Windows\SYSTEM32\ntdll.dll
0x00000000748d0000           0x3f000                0x3 C:\Windows\SYSTEM32\wow64.dll
0x0000000074870000           0x5c000                0x1 C:\Windows\SYSTEM32\wow64win.dll
0x0000000074940000            0x8000                0x1 C:\Windows\SYSTEM32\wow64cpu.dll
```

Source: https://github.com/volatilityfoundation/volatility/wiki/Command-Reference

Volatility 2.6: Some Commands (Windows)

- **Process & information** (with `-o|--offset` or `-p|--pid`):
  - `handles`: displays **open handles** (files, registry keys, mutexes, named pipes, events, window stations, desktops, threads, …)

```
$ python vol.py -f ~/Desktop/win7_trial_64bit.raw --profile=Win7SP0x64 handles
Volatility Foundation Volatility Framework 2.4
Offset(V)          Pid             Handle             Access Type            Details
------------------ ------ ---------------- ---------------- ---------------- -------
0xfffffa80004b09e0      4                0x4           0x1fffff Process          System(4)
0xfffff8a0000821a0      4               0x10            0x2001f Key              MACHINE\SYSTEM\CONTROLSET001\CONTROL\F
0xfffff8a00007e040      4               0x14            0xf003f Key              MACHINE\SYSTEM\CONTROLSET001\CONTROL\S
0xfffff8a000081fa0      4               0x18            0x2001f Key              MACHINE\SYSTEM\SETUP
0xfffffa8000546990      4               0x1c           0x1f0001 ALPC Port        PowerMonitorPort
0xfffffa800054d070      4               0x20           0x1f0001 ALPC Port        PowerPort
0xfffff8a0000676a0      4               0x24            0x20019 Key              MACHINE\HARDWARE\DESCRIPTION\SYSTEM\MU
0xfffffa8000625460      4               0x28           0x1fffff Thread           TID 160 PID 4
0xfffff8a00007f400      4               0x2c            0xf003f Key              MACHINE\SYSTEM\CONTROLSET001
0xfffff8a00007f200      4               0x30            0xf003f Key              MACHINE\SYSTEM\CONTROLSET001\ENUM
0xfffff8a000080d10      4               0x34            0xf003f Key              MACHINE\SYSTEM\CONTROLSET001\CONTROL\C
0xfffff8a00007f500      4               0x38            0xf003f Key              MACHINE\SYSTEM\CONTROLSET001\SERVICES
0xfffff8a0001cd990      4               0x3c                0xe Token
0xfffff8a00007bfa0      4               0x40            0x20019 Key              MACHINE\SYSTEM\CONTROLSET001\CONTROL\W
0xfffffa8000cd52b0      4               0x44           0x120116 File             \Device\Mup
0xfffffa8000ce97f0      4               0x48               0x2a Process          smss.exe(208)
0xfffffa8000df16f0      4               0x4c           0x120089 File             \Device\HarddiskVolume2\Windows\System
0xfffffa8000de37f0      4               0x50           0x12019f File             \Device\clfsTxfLog
0xfffff8a000952fa0      4               0x54            0x2001f Key              MACHINE\SYSTEM\CONTROLSET001\CONTROL\V
0xfffffa800078da20      4               0x58           0x12019f File             \Device\Tcp
0xfffff8a002e17610      4               0x5c                0x9 Key              MACHINE\SOFTWARE\MICROSOFT\WINDOWS NT\
0xfffff8a0008f7b00      4               0x60               0x10 Key              MACHINE\SYSTEM\CONTROLSET001\CONTROL\L
0xfffffa8000da2870      4               0x64           0x100001 File             \Device\KsecDD
0xfffffa8000da3040      4               0x68                0x0 Thread           TID 228 PID 4
...
```

Source: https://github.com/volatilityfoundation/volatility/wiki/Command-Reference

Volatility 2.6: Some Commands (Windows)

- **Process & information** (with `-o|--offset` or `-p|--pid`):
  - `handles`: displays open handles **in a process** [based on object type]

```
$ python vol.py -f ~/Desktop/win7_trial_64bit.raw --profile=Win7SP0x64 handles -p 296 -t Process
Volatility Foundation Volatility Framework 2.4
Offset(V)          Pid             Handle             Access Type             Details
------------------ ------ ------------------ ------------------ --------------- -------
0xfffffa8000c92300    296               0x54           0x1fffff Process         wininit.exe(332)
0xfffffa8000c5eb30    296               0xc4           0x1fffff Process         services.exe(428)
0xfffffa80011c5700    296               0xd4           0x1fffff Process         lsass.exe(444)
0xfffffa8000ea31b0    296               0xe4           0x1fffff Process         lsm.exe(452)
0xfffffa8000c64840    296              0x140           0x1fffff Process         svchost.exe(348)
0xfffffa8001296b30    296              0x150           0x1fffff Process         svchost.exe(568)
0xfffffa80012c3620    296              0x18c           0x1fffff Process         svchost.exe(628)
0xfffffa8001325950    296              0x1dc           0x1fffff Process         sppsvc.exe(816)
...
```

Source: https://github.com/volatilityfoundation/volatility/wiki/Command-Reference

Volatility 2.6: Some Commands (Windows)

- **Process & information** (with `-o|--offset` or `-p|--pid`):
  - `cmdscan`: shows commands entered through a **console shell** (cmd.exe) by scanning for COMMAND_HISTORY

```
$ python vol.py -f VistaSP2x64.vmem --profile=VistaSP2x64 cmdscan
Volatility Foundation Volatility Framework 2.4

**************************************************
CommandProcess: csrss.exe Pid: 528
CommandHistory: 0x135ec00 Application: cmd.exe Flags: Allocated, Reset
CommandCount: 18 LastAdded: 17 LastDisplayed: 17
FirstCommand: 0 CommandCountMax: 50
ProcessHandle: 0x330
Cmd #0 @ 0x135ef10: cd \
Cmd #1 @ 0x135ef50: cd de
Cmd #2 @ 0x135ef70: cd PerfLogs
Cmd #3 @ 0x135ef90: cd ..
Cmd #4 @ 0x5c78b90: cd "Program Files"
Cmd #5 @ 0x135fae0: cd "Debugging Tools for Windows (x64)"
Cmd #6 @ 0x135efb0: livekd -w
Cmd #7 @ 0x135f010: windbg
Cmd #8 @ 0x135efd0: cd \
Cmd #9 @ 0x135fd20: rundll32 c:\apphelp.dll,ExportFunc
Cmd #10 @ 0x5c8bdb0: rundll32 c:\windows_apphelp.dll,ExportFunc
Cmd #11 @ 0x5c8be10: rundll32 c:\windows_apphelp.dll
Cmd #12 @ 0x135ee30: rundll32 c:\windows_apphelp.dll,Test
Cmd #13 @ 0x135fd70: cd "Program Files"
Cmd #14 @ 0x5c8b9e0: dir
Cmd #15 @ 0x5c8be60: cd "Debugging Tools for Windows (x64)"
Cmd #16 @ 0x5c8ba00: dir
Cmd #17 @ 0x135eff0: livekd -w

[snip]
```

Source:
https://github.com/volatilityfoundation/volatility/wiki/Command-Reference

Volatility 2.6: Some Commands (Windows)

- **Process & information** (with `-o|--offset` or `-p|--pid`):
  - `consoles`: shows commands that attackers typed into cmd.exe or executed via backdoors by scanning **for CONSOLE_INFORMATION**

```
$ python vol.py -f xp-laptop-2005-07-04-1430.img consoles
Volatility Foundation Volatility Framework 2.4

[csrss.exe @ 0x821c11a8 pid 456 console @ 0x4e23b0]
  OriginalTitle: '%SystemRoot%\\system32\\cmd.exe'
  Title: 'C:\\WINDOWS\\system32\\cmd.exe - dd if=\\\\.\\PhysicalMemory of=c:\\xp-2005-07-04-1430.img conv=noerror'
  HistoryBufferCount: 2
  HistoryBufferMax: 4
  CommandHistorySize: 50
[history @ 0x4e4008]
  CommandCount: 0
  CommandCountMax: 50
  Application: 'dd.exe'
[history @ 0x4e4d88]
  CommandCount: 20
  CommandCountMax: 50
  Application: 'cmd.exe'
  Cmd #0 @ 0x4e1f90: 'dd'
  Cmd #1 @ 0x4e2cb8: 'cd\\'
  Cmd #2 @ 0x4e2d18: 'dr'
  Cmd #3 @ 0x4e2d28: 'ee:'
  Cmd #4 @ 0x4e2d38: 'e;'
  Cmd #5 @ 0x4e2d48: 'e:'
  Cmd #6 @ 0x4e2d58: 'dr'
  Cmd #7 @ 0x4e2d68: 'd;'
  Cmd #8 @ 0x4e2d78: 'd:'
  Cmd #9 @ 0x4e2d88: 'dr'
  Cmd #10 @ 0x4e2d98: 'ls'
  Cmd #11 @ 0x4e2da8: 'cd Docu'
  Cmd #12 @ 0x4e2dc0: 'cd Documents and'
  Cmd #13 @ 0x4e2e58: 'dr'
  Cmd #14 @ 0x4e2e68: 'd:'
  Cmd #15 @ 0x4e2e78: 'cd dd\\'
```

Source:
https://github.com/volatilityfoundation/volatility/wiki/Command-Reference

Volatility 2.6: Some Commands (Windows)

- **Process & information** (with `-o|--offset` or `-p|--pid`):
  - `envars`: displays **environment variables**

```
$ python vol.py -f ~/Desktop/win7_trial_64bit.raw --profile=Win7SP0x64 envars
Volatility Foundation Volatility Framework 2.4
Pid      Process              Block              Variable                       Value
-------- -------------------- ------------------ ------------------------------ -----
     296 csrss.exe            0x00000000003d1320 ComSpec                        C:\Windows\system32\cmd.exe
     296 csrss.exe            0x00000000003d1320 FP_NO_HOST_CHECK               NO
     296 csrss.exe            0x00000000003d1320 NUMBER_OF_PROCESSORS           1
     296 csrss.exe            0x00000000003d1320 OS                             Windows_NT
     296 csrss.exe            0x00000000003d1320 Path                           C:\Windows\system32;C:\Windows;C:\Wind
     296 csrss.exe            0x00000000003d1320 PATHEXT                        .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE
     296 csrss.exe            0x00000000003d1320 PROCESSOR_ARCHITECTURE         AMD64
     296 csrss.exe            0x00000000003d1320 PROCESSOR_IDENTIFIER           Intel64 Family 6 Model 2 Stepping 3, G
     296 csrss.exe            0x00000000003d1320 PROCESSOR_LEVEL                6
     296 csrss.exe            0x00000000003d1320 PROCESSOR_REVISION             0203
     296 csrss.exe            0x00000000003d1320 PSModulePath                   C:\Windows\system32\WindowsPowerShell\
     296 csrss.exe            0x00000000003d1320 SystemDrive                    C:
     296 csrss.exe            0x00000000003d1320 SystemRoot                     C:\Windows
     296 csrss.exe            0x00000000003d1320 TEMP                           C:\Windows\TEMP
     296 csrss.exe            0x00000000003d1320 TMP                            C:\Windows\TEMP
     296 csrss.exe            0x00000000003d1320 USERNAME                       SYSTEM
     296 csrss.exe            0x00000000003d1320 windir                         C:\Windows
```

Source: https://github.com/volatilityfoundation/volatility/wiki/Command-Reference

Volatility 2.6: Some Other Commands (Windows)

- **Logs & histories**:
  - Recover event logs (XP/2003): `evtlogs`
  - Recover IE cache/Internet history: `iehistory`
  - Show running services: `svcscan`
- **Networking** information:
  - Active info (XP/2003): `connections` and `sockets`
  - Scan for residual info (XP/2003): `connscan` and `sockscan`
  - Network info for Vista, 2008, and 7: `netscan`
- **Kernel & objects**:
  - Scan for driver objects: `driverscan`

Volatility 2.6: Some Commands (Linux)

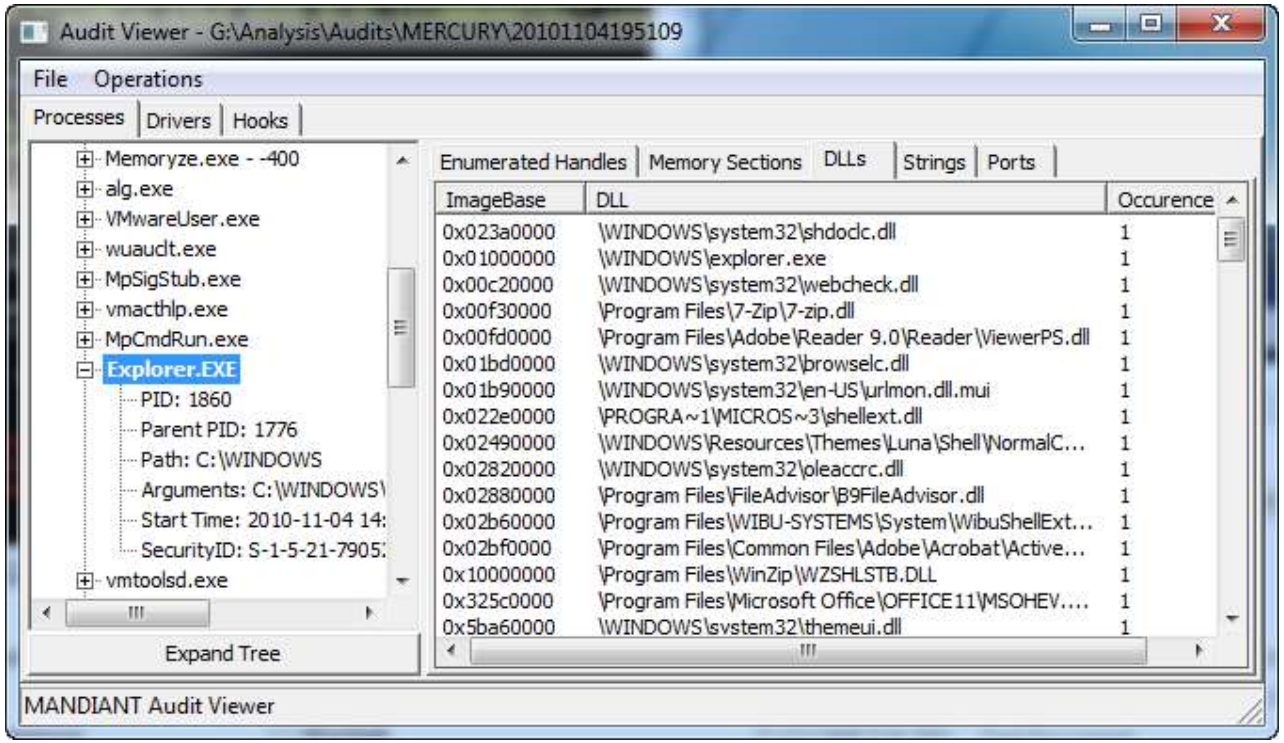
- **Processes listings**:
  - Basic active process listing: `linux_pslist`
  - Show processes in parent/child tree: `linux_pstree`
- **Process information** (with `-o|--offset` or `-p|--pid`):
  - Display shared libraries: `linux_library_list`
  - Show command line arguments: `linux_psaux`
  - Display open handles: `linux_lsof`
  - Display environment variables: `linux_psenv` and `linux_bash_env`
- **Networking** information:
  - Active info: `linux_netstat`
  - Interface information: `linux_ifconfig`

Volatility: Resources

- Volatility **references**:
  - Uploaded Volatility **cheat sheet** on LumiNUS
  - Command reference: https://github.com/volatilityfoundation/volatility/wiki/Command-Reference
- Some **videos**:
  - https://www.youtube.com/watch?v=1Cl69Bj9T5s
  - https://www.youtube.com/watch?v=1PAGcPJFwbE
- A sample interesting Volatility **plugin**: **CryptoScan**:
  - Written by Jesse Kornblum
  - Finds TrueCrypt passphrases
  - Source: https://github.com/binglot/Cryptoscan
  - Demo (interesting!): https://www.youtube.com/watch?v=1oEcqYZpNvs

Other Memory Analyzers: FireEye Redline

- You can also use other memory analyzer tools, e.g. (GUI-based) **FireEye Redline**
- FireEye used to have **AuditViewer**



Ref: https://www.sans.org/blog/digital-forensics-how-to-memory-analysis-with-mandiant-memoryze/

Other Memory Analyzers: FireEye Redline

- **Redline** can inspect running processes and drivers from memory
- Reference: https://fireeye.market/apps/211364
- A video demo: https://www.youtube.com/watch?v=tCIEYCWTdk4

- See also for a **comparison** of Redline and Volatility + other memory analysis tools: https://soshace.com/comparative-analysis-of-free-tools-for-physical-memory-dumps-parsing/

Hex Editor for File Inspection & Analysis

- Examine and operate on data at the ***binary/raw*** level
- Various available **tools**:
  - Commercial: Hex Workshop (BreakPoint Software), 010 Editor, **WinHex (X-Ways Software Technology)**, …
  - Free: various tools
- **Good features** of a good hex editor:
  - Open big files or the entire local drives
  - Compute a hash value
  - Search/replace byte patterns or sectored data
  - Make changes on data

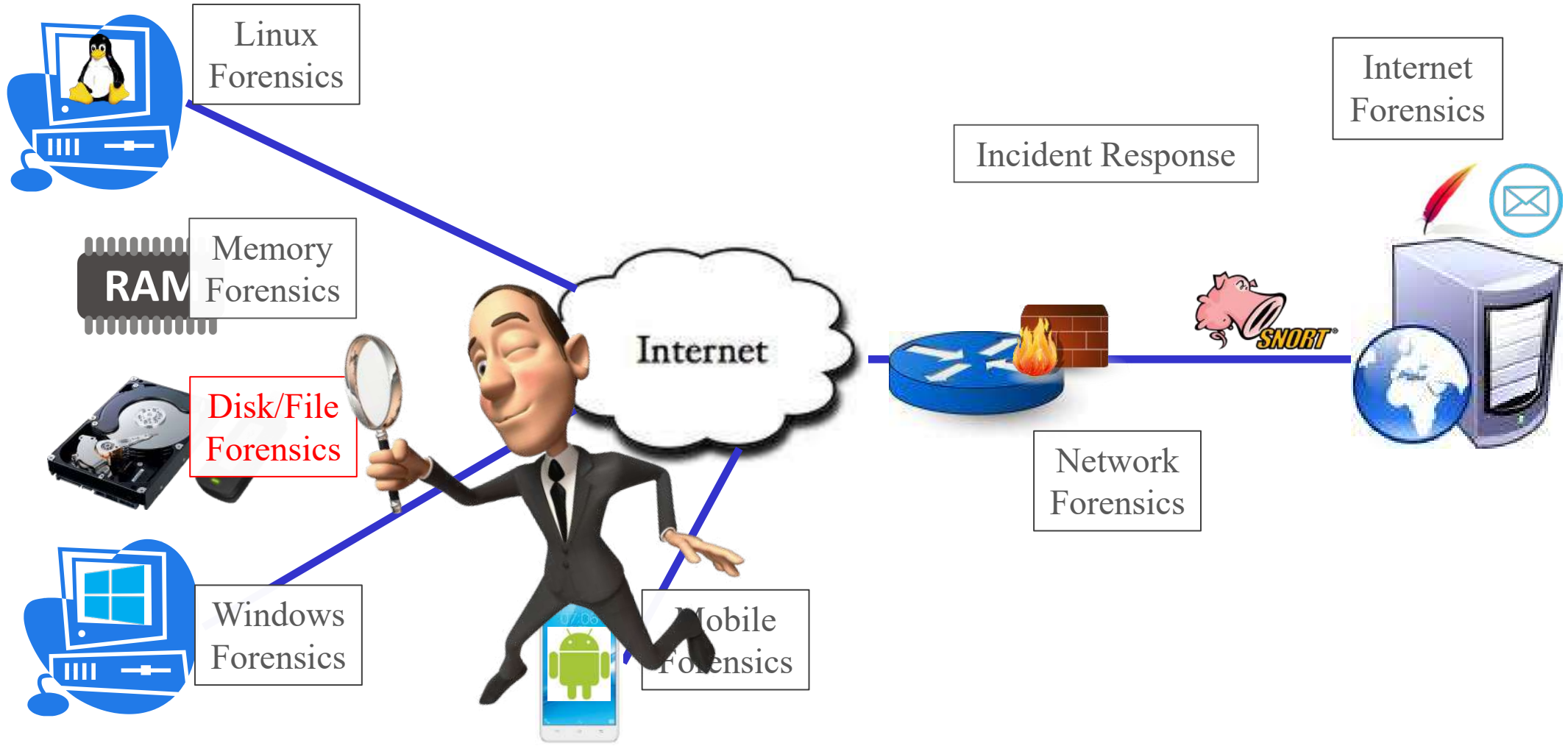
Hex Editor for File Inspection & Analysis

- Perform bitwise arithmetic operations: e.g. shifting bits
- Convert number base
- Recover data
- Other disk utilities: disk backup, low-level erase, …

- Some **forensics use cases** in your labs:
  - String searching: **Lab 3 Task 2-C**
  - Inspect a file with modified extension: *Lab 4*
- Video:
  - https://www.youtube.com/watch?v=L3BwXbRDQM4

Further on Disk Image Acquisition & Analysis

Disk Forensics: Acquisition Challenges & Analysis



Accessing Memory Disk Image (Review)

- How can we analyze the dumped **disk image**?
- We have used **FTK Imager** (Lab 2):
  - **Add image** as **evidence**:
    - Allow you to browse file system, and extract/export files of interest
  - **Mount image** as a **drive**:
    - Make a disk visible to the OS
    - Allows external tools to inspect the files, e.g. anti-virus software

Challenges during Static Acquisition?

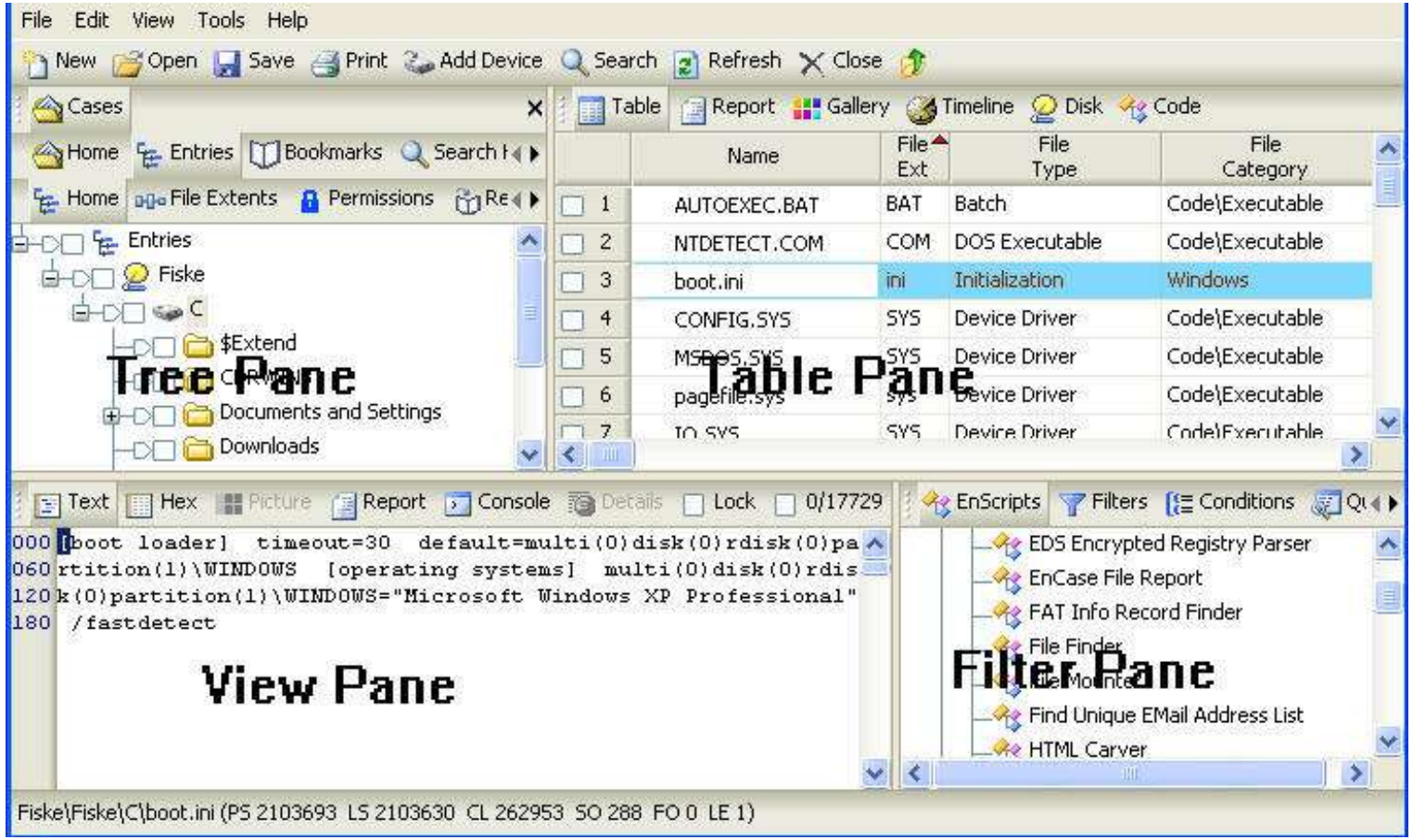
- We haven't discussed some **challenges** that you may face during your static acquisition on non-volatile storage media
- It is ***not* always possible** to do a static acquisition:
  - Size of the media (storage density) and time available
  - RAID
  - Cloud-based storage
  - SSDs
  - Anti-forensics techniques

Acquired Disk Image: How about Its Analysis??

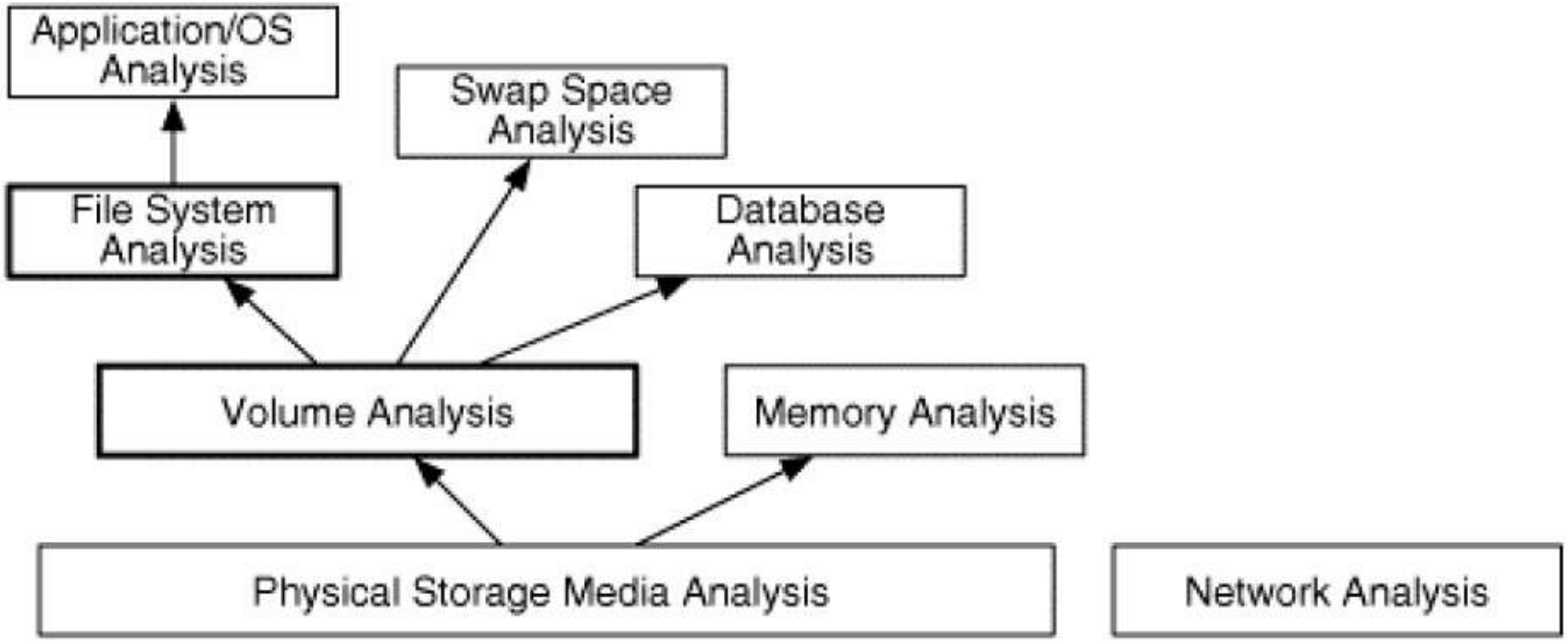
- Analysis using **forensic suites**:
  - **Autopsy**: Discussed later in this lecture, Lab 3 Task 3 (file and deleted file extractions), additional tasks in the next two labs
  - **Other** forensics suites: Links to video demos are given in this lecture
- More **specialized** analysis tools:
  - **Disk** and **file analyses**: in Lecture 4 & Lab 4 (e.g. TSK tool)
  - **OS** forensics: Windows and Linux forensics
  - **Network** forensics
  - **Internet** forensics: web and email forensics
  - **Application/media** forensics

Forensic Software Suites

- Typical **user interface**:



Static Acquisition & Volume+File-System Analysis



From: Brian Carrier, “File System Forensic Analysis”

Lab 3 Exercises: Preview

- Task 1: Perform a **live acquisition** of a Windows machine using:
  - FTK Imager Lite
  - (Optional) FireEye's Memoryze
- Task 2: **Inspect and analyze** a **memory image** file using:
  - Volatility: various commands
  - FTK Imager: string searching
  - (Optional) Hex Editor (WinHex): string searching
- Task 3: **Inspect and analyse** a **disk image** file using Autopsy

Break!

Lab 3 Exercises: Things to Do *before* the Lab!

During the break or while listening to the 2nd part of the lecture, **you can already start doing**:

- For Task 2: **Download** the given sample image file
- For Task 1-A: Run FTK Imager Lite to do a **live acquisition**

Static Acquisition Challenges

Potential Challenges during Static Acquisition

- Let's discuss the **potential challenges**:
  - **Size** of the media (storage density) and time available
  - **RAID**
  - **Cloud-based storage**
  - **SSDs**
  - **Anti-forensics** techniques

Acquisition Challenge #1: RAID

- **RAID:** Redundant Array of Independent Disks, originally Redundant Array of Inexpensive Disks
- Combines **multiple physical disk drive** components into **one or more logical units**: for data redundancy, performance improvement, or both
- Common **techniques**: mirroring, striping, parity
- Various RAID **levels**:
  - **Standard** levels: RAID 0, 1, 2, 3, 4, 5, 6
  - **Nested (hybrid)** levels: RAID 0+1, 1+0 (10), …
  - See: https://en.wikipedia.org/wiki/RAID

Understanding RAID

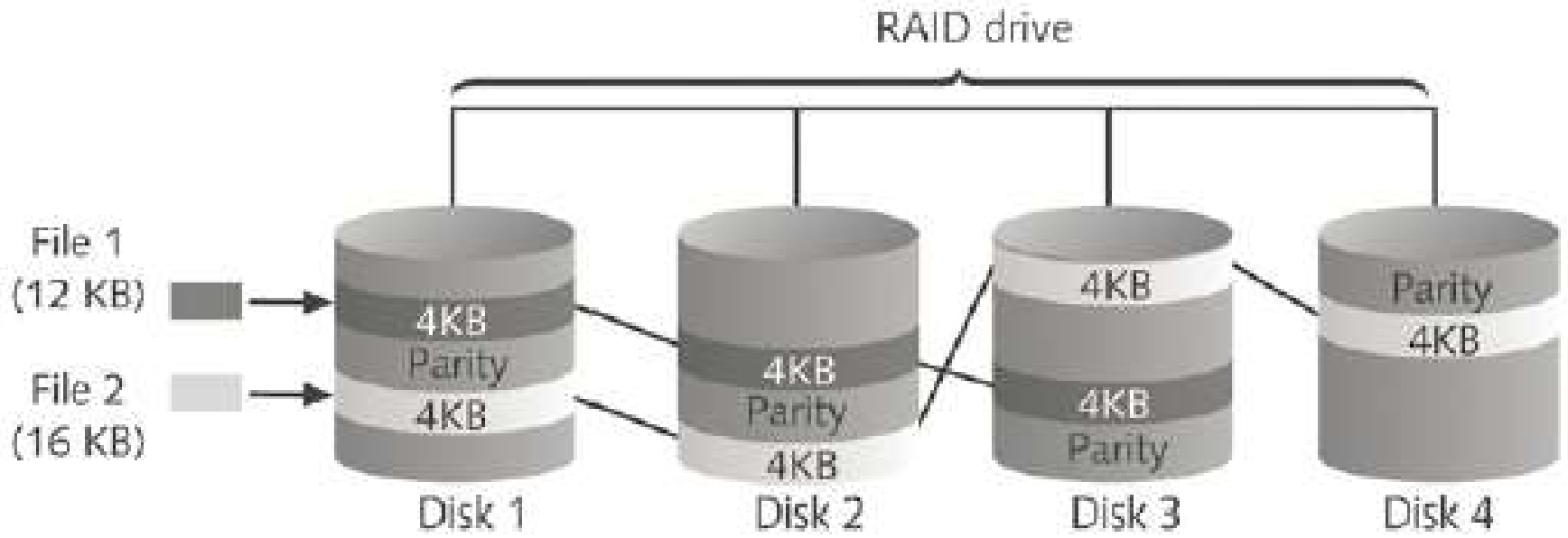


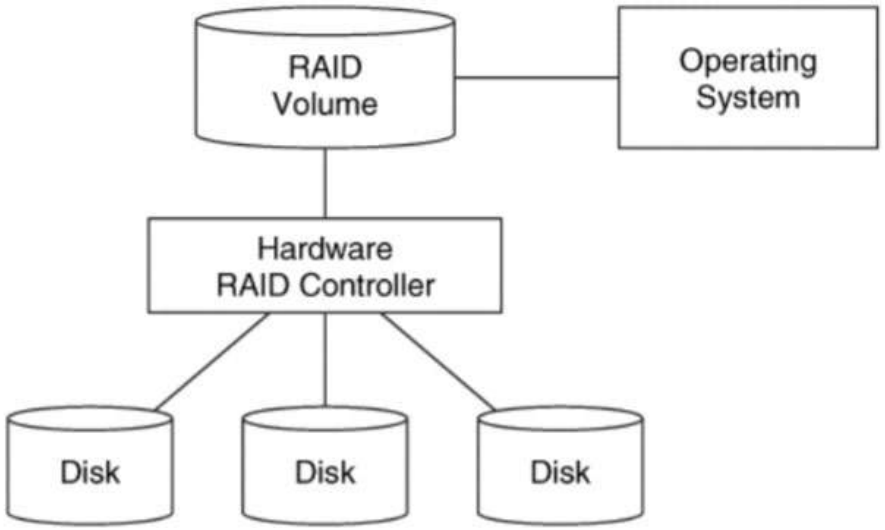
Figure 3-13 RAID 5: Block-level striping with distributed parity

Handling Computer Hardware (cont.)

RAID

RAID Disk Acquisition: Hardware RAID

- *Question: how should we do a **static acquisition** on **RAID disk**?*
- Some **guidelines** below (Brian Carrier, "File System Forensic Analysis")
- If a **hardware RAID** is used:
  - The computer sees **only** the RAID **volume** and not the individual disks



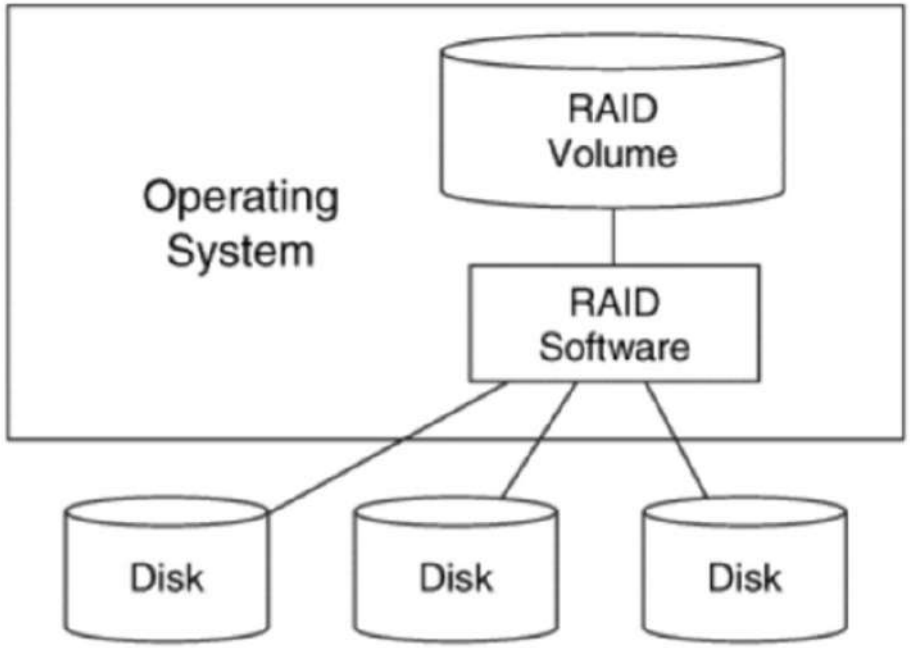
From: Brian Carrier, "File System Forensic Analysis"

RAID Disk Acquisition: Hardware RAID

- First, acquire the **final RAID *volume*** as though it were a normal single disk: to be analyzed using partition & file system analysis tools
  - ***How?*** Boot the suspect system with a **bootable OS** that **has drivers for the RAID controller**, then use dd*/FTK Imager/…
  - **Issue**: a **large amount** of disk space (GBs to TBs) can be required to store the image?!
- Then, try to possibly acquire the contents of **each *physical* disk** to check any hidden data in the disk's unused sectors
- In summary: **logical acquisition** + (possible) **physical acquisition**

RAID Disk Acquisition: Software RAID

- If a **software RAID** is used:
  - The OS has special **drivers** that merge the individual disks
  - The OS **sees** the individual disks,
    but may **show** only the RAID volume to the user



From: Brian Carrier, "File System Forensic Analysis"

RAID Disk Acquisition: Software RAID

- Yet, the **individual disks** can typically be accessed through **raw devices** (UNIX system) or **device objects** (Microsoft Windows)
- Analysis & acquisition of software RAID is similar to a hardware RAID: acquire the **RAID *volume*** and then ***physical* disks**
- In summary:
  logical acquisition + **physical acquisition** (which can be done)
- Unlike hardware RAID, there could be **some tools** that **can merge** the individual disks together: *more possible analysis & results?*

RAID Disk Acquisition

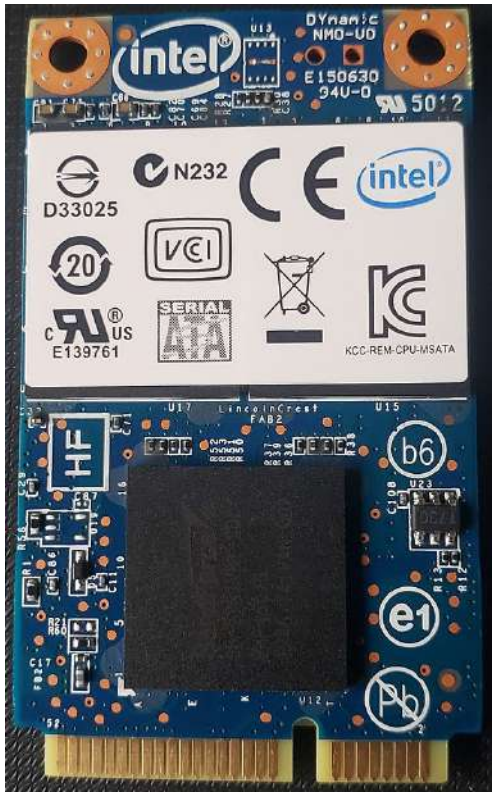
- RAID disk acquisition can be rather **challenging** in practice, especially the ***physical*** acquisition
- More doable **alternative techniques**:
  - **Logical acquisition**:
    - Copies the **directories and files** of a **logical volume**
    - Pros: Faster & easier than bit-stream imaging
    - Cons: Does not capture other data that may be present on the media, e.g. deleted files or residual data stored in slack space
  - **Sparse acquisition**:
    - **Logical** + **fragments** of unallocated/deleted data blocks

Acquisition Challenge #2: Cloud-based Storage

- Various popular **cloud-based storage systems**:
  Google Drive, DropBox, OneDrive, …
- **Inaccessible** underlying physical disks: completely shielded!

- *So, how?*
- Acquisition is typically limited to **logical acquisition** only *(duh!)*
- *Any other possibilities??*

Acquisition Challenge #3: Solid-State Drives



From: Wikipedia

Solid-State Drives (SSDs)

- SSDs store data in semiconductor cells: ***no* moving parts**!
- **Strengths**: They are more resistant to physical shock, run silently, and have quicker access time and lower latency
- **Drawbacks**: More expensive, a **limited number of writes**, and will be slower the more filled up they are
- An SSD uses a ***SSD controller***:
  - An embedded **processor** that executes firmware-level code: one of the most important factors of SSD performance
  - It handles various **functions**, e.g. wear leveling, garbage collection (*more on these later*)
- See: https://en.wikipedia.org/wiki/Solid-state_drive

SSDs: Data Overwrite = Erase + Write

- **Data overwrite** process:
  - In magnetic tapes and disks: can write over old data
  - In SSDs: has to completely **clear each data block** before writing new data to drives
- In SSDs:
  - Data-block **overwrite** = ***erase*** then **write** the data-block
  - See also: https://datarecovery.com/rd/file-deletion-different-solid-state-drives-hard-drives/

SSDs: Wear Levelling Policy

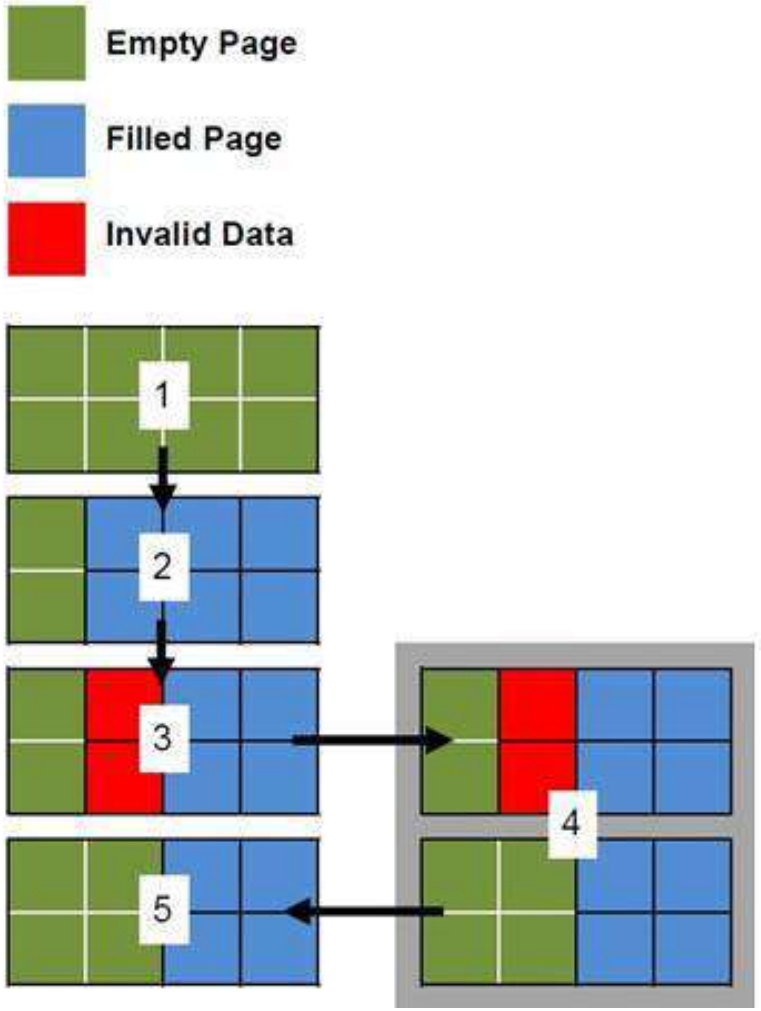
- SSDs have constraints on **limited** number of writes on blocks
  - ***Endurance rating***: number of block erasure (in cycles)
  - *Question: any consequences on SSD operations?*

- ***Wear levelling* policy**:
  - Suppose a block is erased repeatedly without writing to any other blocks: the block will **wear out** before all the other blocks
    → a premature ending of life
  - To prevent that, SSDs distribute blocks around endurance rating limits: to ensure **even** wear of erasures and re-writes for **all blocks**
  - A **data-block update** is done by writing into a ***new*** **data block**
  - The **old** data block is to be **erased** by the SSD controller and put into free space: **TRIM command** and **garbage collector**

SSDs: TRIM Command & Garbage Collector

- **TRIM command** (enabled by default): issued by the OS to the SSD controller at the time the user deletes a file, formats the disk, or deletes a partition
- Background **garbage collector**:
  - Responds to TRIM command
  - Built into SSD controller to **physically erases** deleted blocks and put them back to free space
- The block-deletion **workflow** is explained in the next slide
  - See also: https://www.youtube.com/watch?v=vLoYduckmuo

SSDs: TRIM Command & Garbage Collector



1.) SSD pages contain no data

2.) User writes data to SSD pages

3.) User deletes some data. Pages are marked as 'not in use' by the host OS, but data remains on SSD.

4.) TRIM command tells SSD controller that pages contain invalid data. Pages with invalid data are cleaned.

5.) Data is written back to SSD memory cells. The invalid data has been cleaned and data is able to be written to the pages at full speed.

https://articles.forensicfocus.com/2014/09/23/recovering-evidence-from-ssd-drives-in-2014-understanding-trim-garbage-collection-and-exclusions/

Acquisition Challenge #3: Solid-State Drives

- **Acquisition challenge**:
  - The TRIM command + garbage collector ***zeroes*** all deleted data blocks, whole range of a deleted file
  - Most SSDs execute the TRIM function **very quickly**
  - See also: https://en.wikipedia.org/wiki/Solid-state_drive#Data_recovery_and_secure_deletion
- In conclusion, for acquisition of SSDs:
  - **Logical** acquisition
  - **Physical** acquisition: possible, but finding residual data and performing file carving become much harder
    → the TRIM function *will eventually (even soon)* zero deleted data blocks

Acquisition Challenge #4: Anti-Forensics Techniques

- **Why/goal**:
  - To **avoid discovery** of information related to a suspect's illegal activities
- **How**:
  - By making forensics investigations difficult, significantly more time-consuming, or impossible
- Possible different **techniques**:
  - Data **transformation**:
    - By **manipulating** data
    - E.g. encryption
  - Data **hiding**:
    - By **obfuscating** data
    - E.g. steganography

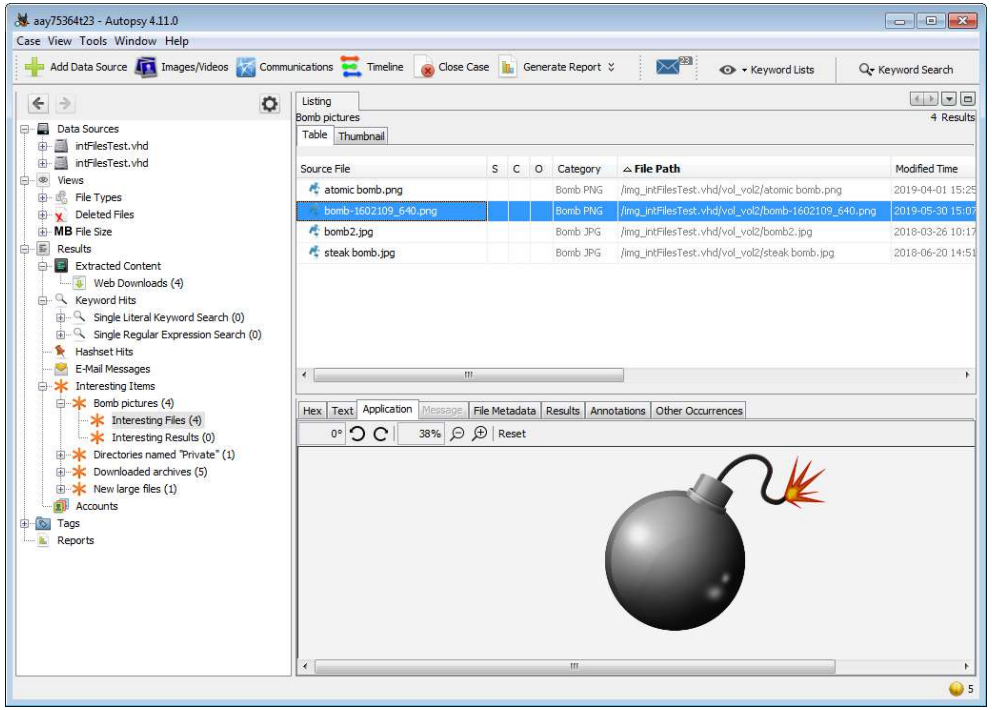
Other Possible Anti-Forensics Techniques

- Data **destruction**:
  - By **erasing** data
  - Some tools: BCWipe, Eraser, MS Sdelete, shred, Secure Empty Trash
- Data **fabrication**
  - By **creating** bogus data
- File-system **obstruction/alteration**:
  - By **blocking** data access on file systems
  - E.g. manual file-system metadata **alteration**

Forensic Analysis using Autopsy Forensic Suite

Forensic Analysis

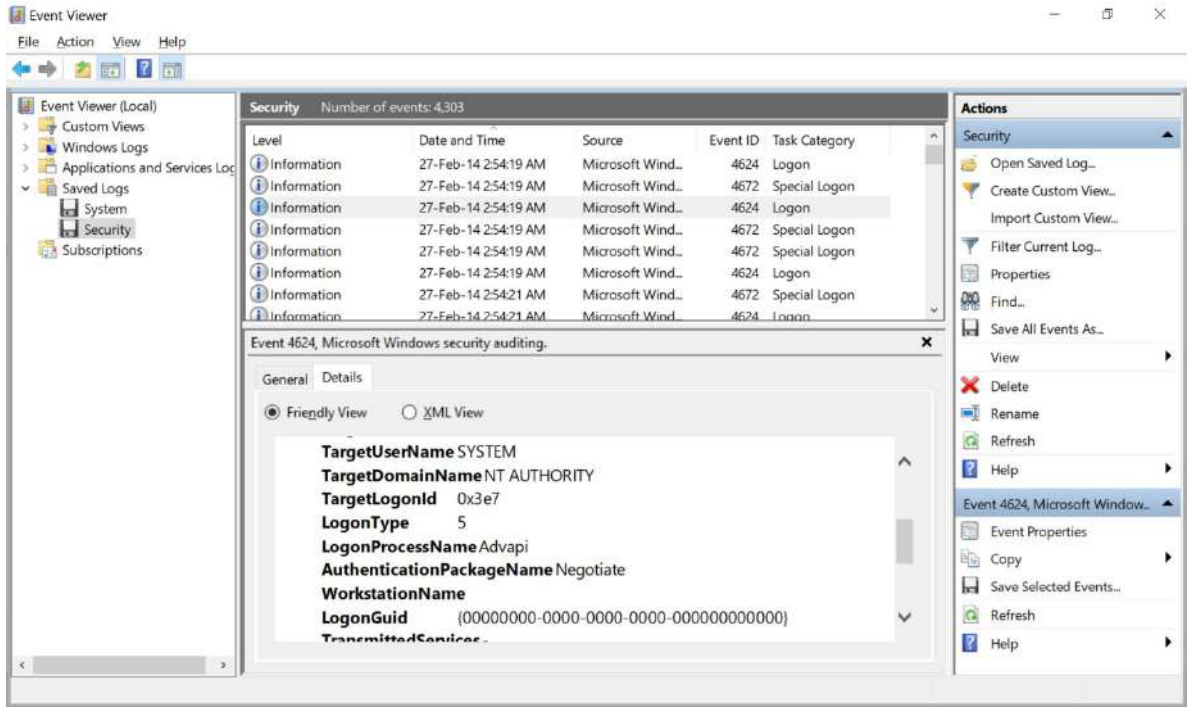
- The **analysis** of forensics images is normally the **most time-consuming** task:
  - Which tools to use
  - What techniques/features to use
  - How to identify and correlate important pieces of evidence
- Forensic **software suites**:
  - They can do acquisitions, interpretation, analysis and report writing
  - While this is useful, it does **not** mean they are the only tools needed



From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

Supporting Specialized Software Tools

- Numerous **specialized tools** exist:
  - Drive mounters
  - Registry editors
  - Event viewers
  - Password crackers
  - Cache extractors
  - …
- Other **diagnostic & analysis tools** not specifically for forensics are often also used



Autopsy Forensic Suite: Some History

- **2001**: First open source release
  - Interface to **The Sleuth Kit (TSK)**: powerful volume + FS analysis tools
  - Perl/HTML, Linux and OS X only
  - Usability limitations: See D.J. Bennett and P. Stephens, "*A Usability Analysis of the Autopsy Forensic Browser*", 2nd International Symposium on Human Aspects of Information Security & Assurance (HAISA), 2008
- **2010**: Started **v3** from scratch as a platform
  - Windows-based & automated
  - v3.0.0 released in September 2012
- **2015**: Autopsy **v4.0** was launched

Autopsy v3+

- Targeted **features**:
  - Simple UI concepts
  - Easy to install and use
  - Several frameworks and plug-in **modules**
  - **Extensible**
  - Updated by community
  - **Free** *(yay!)*
- See:
  - Brian Carrier, "*Autopsy 3.0*", Basis Technology, 2013
  - Julia Keffer, "*Autopsy Forensic Browser User Guide*", 2013

Autopsy v4+

- Allows for a **collaboration**:
  - **Multiple examiners** can work on the same case at the same time, and see what other examiners are doing
  - Examiners have access to real-time information via **network-based services**, including a central database and keyword index
  - See: https://www.autopsy.com/collaborative-autopsy-how-it-works/
- Additionally implemented **new features**:
  - See https://www.autopsy.com/blog/
- **References**:
  - Quick start guide (latest version): https://sleuthkit.org/autopsy/docs/user-docs/4.19.3/quick_start_guide.html
  - User Documentation (latest version): https://sleuthkit.org/autopsy/docs/user-docs/4.19.3/

Autopsy Workflow (Process Flow)

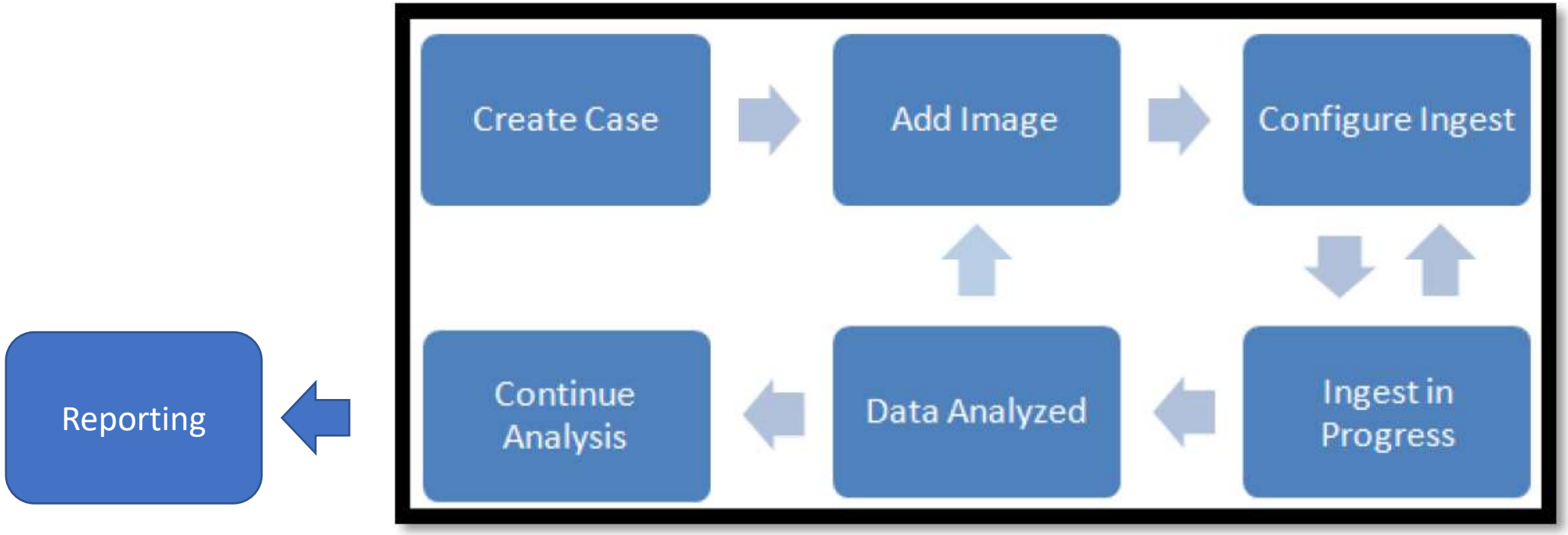


Figure 1: Autopsy Process Flow

From: Julia Keffer, "*Autopsy Forensic Browser User Guide*", 2013

Autopsy Workflow

1.Create a case:

- A ***case*** is a container for one or more data sources (one must be created)

2.Adding data sources:

- One or more ***data sources*** (e.g. disk images, VM files, local files) are added

3.Analyze with ingest modules:

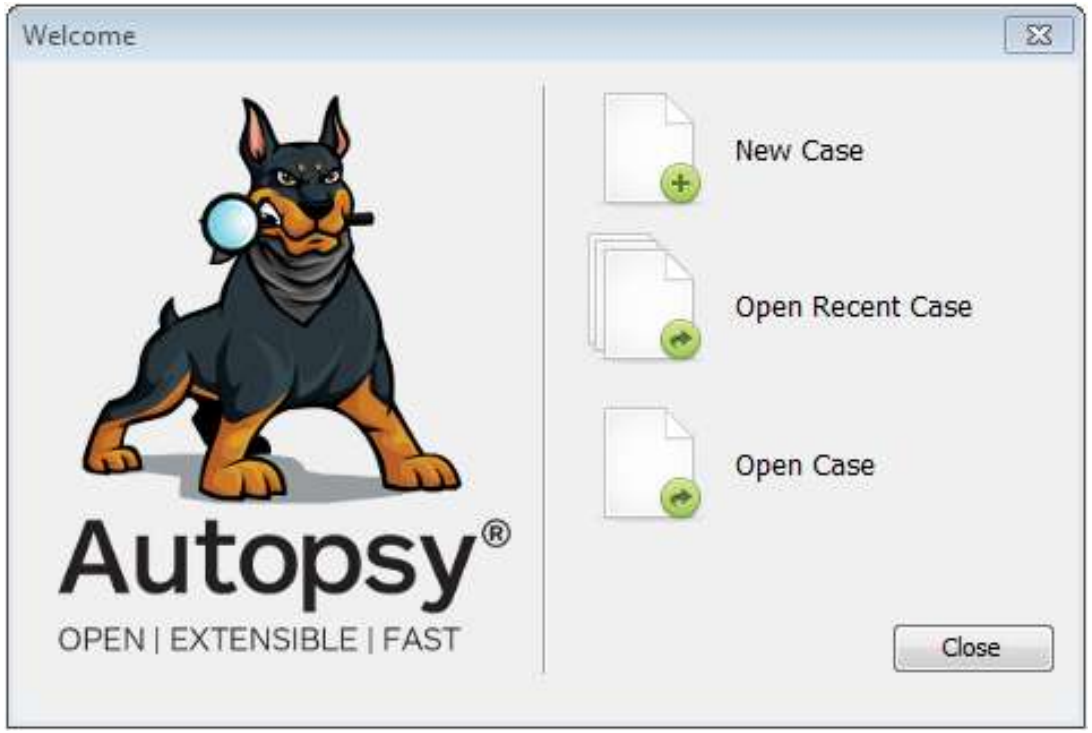
- Selected ***ingest modules*** operate in the **background** to analyze the data
- **Results** are posted to the interface in real time and provide alerts as necessary

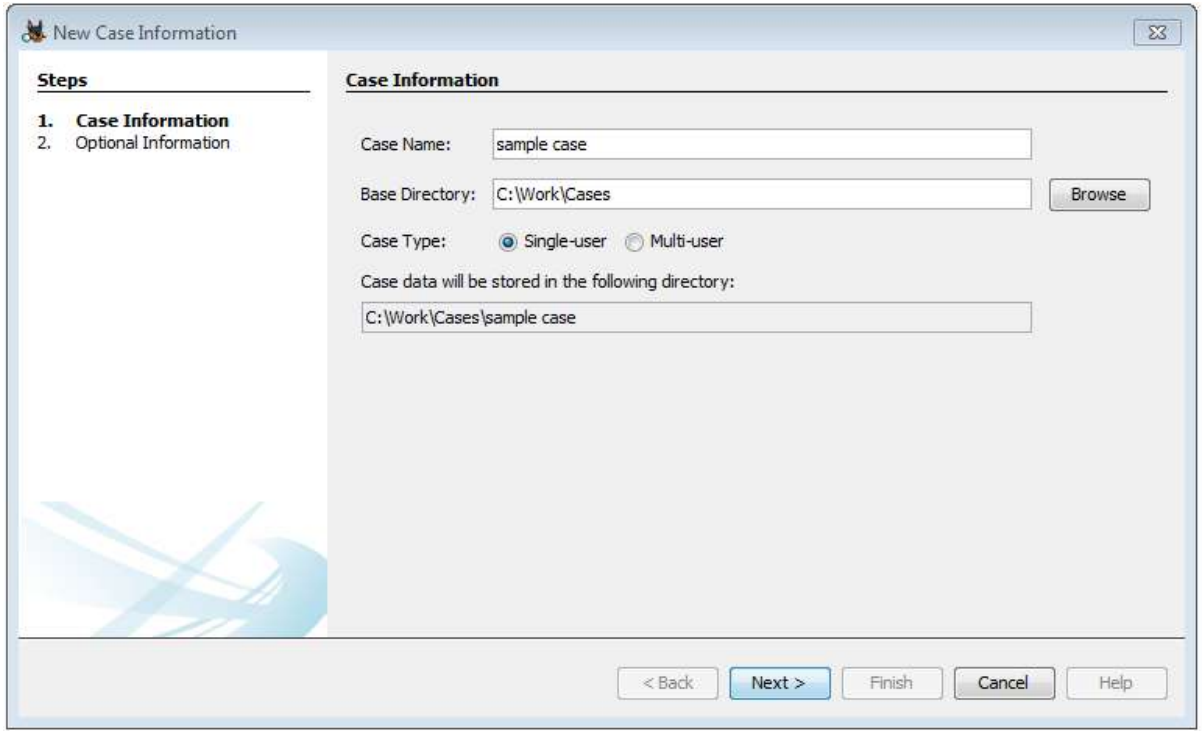
Autopsy Workflow

3. **Manual Analysis**:
   - The user navigates the interface, file contents, and ingest module results for ***evidence identification***
   - Item ***tagging*** is possible for reporting & analysis

4. **Report Generation**:
   - The user initiates a ***final report*** based on selected tags or results

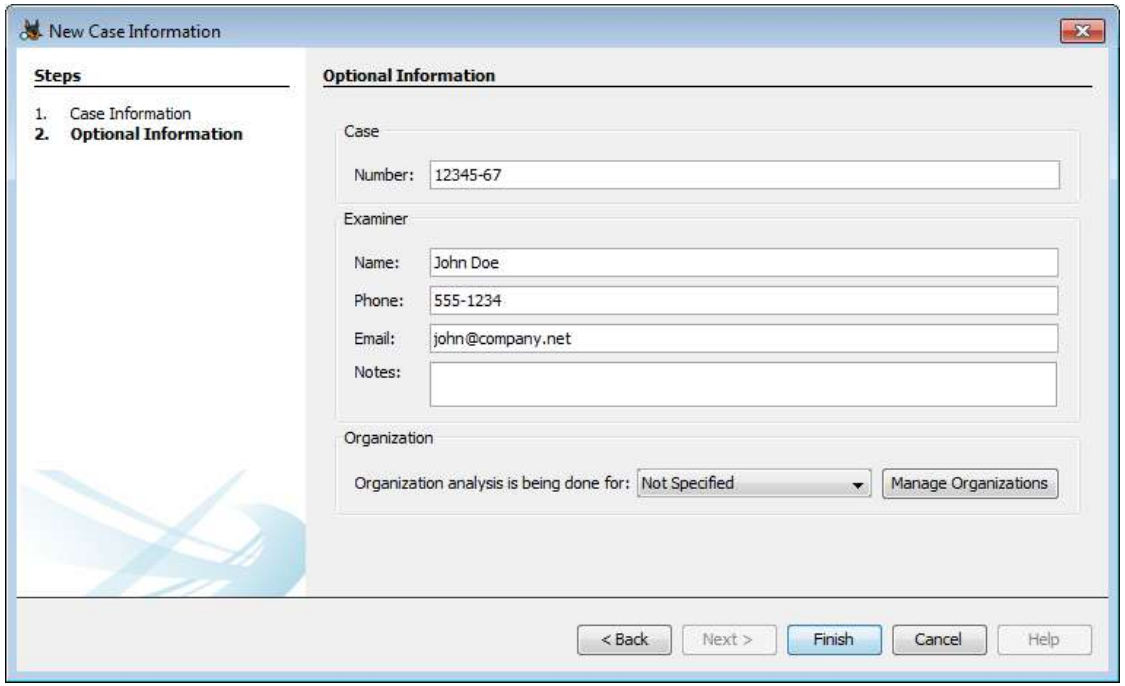
Autopsy v4+: Creating a Case

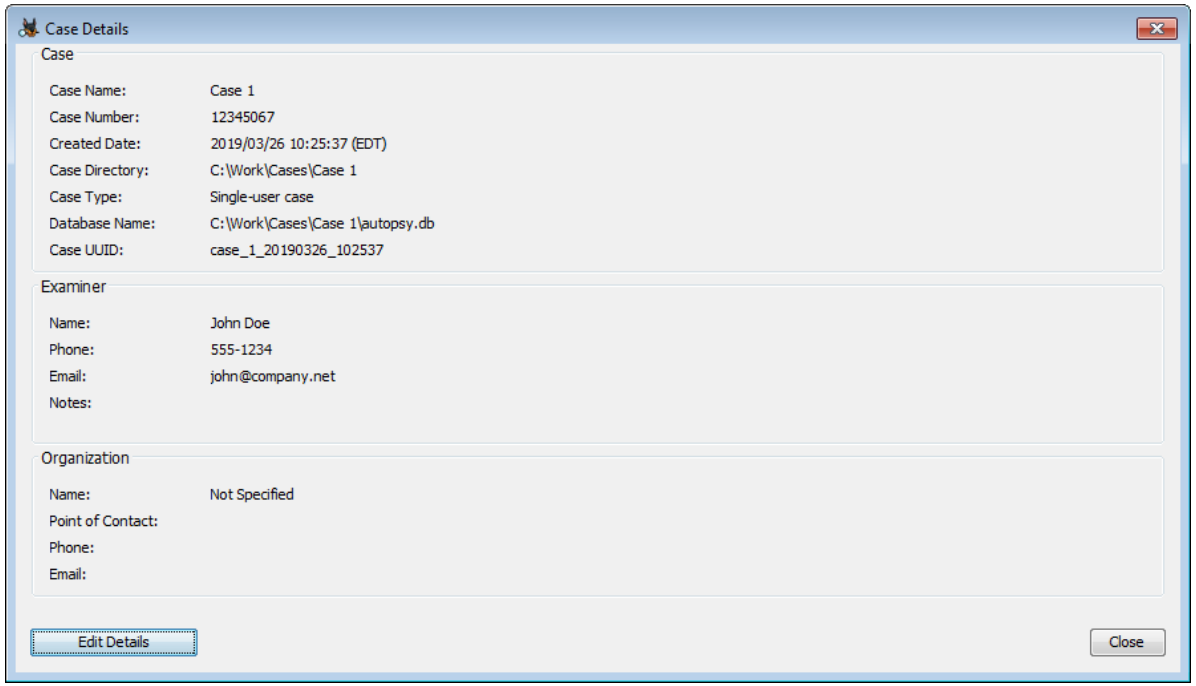




From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

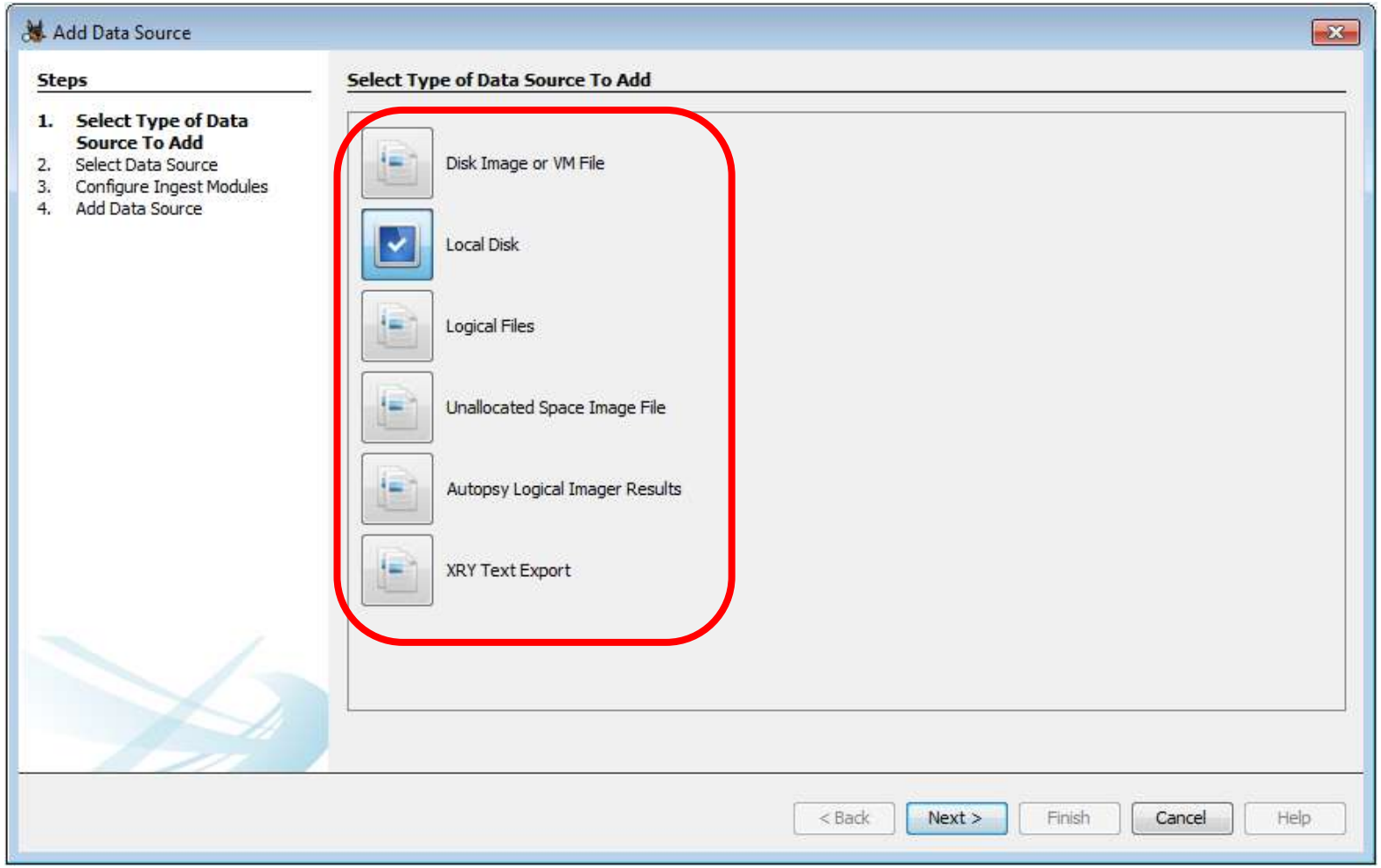
Autopsy v4+: Creating a Case





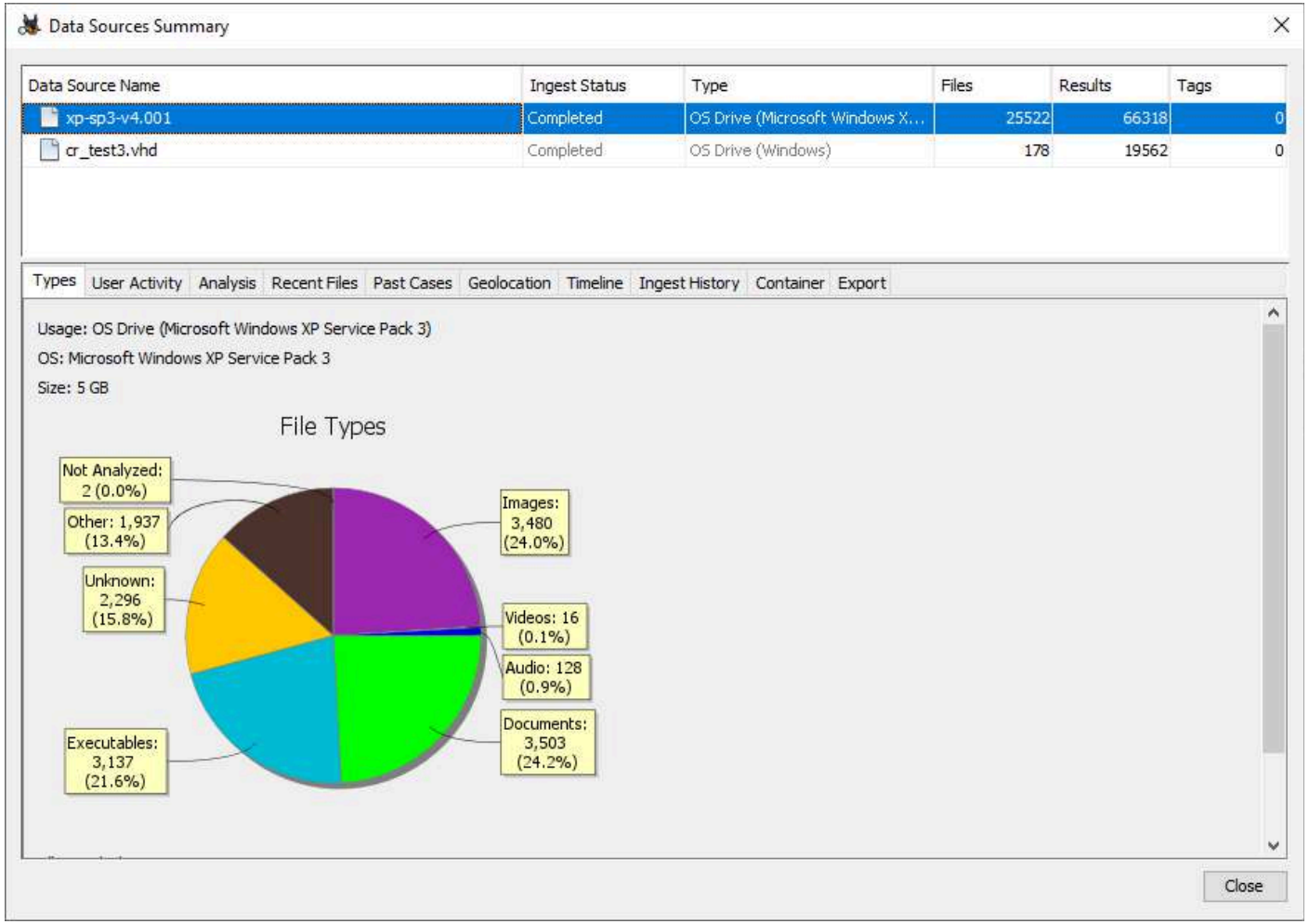
From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

Autopsy v4+: Adding Data Sources



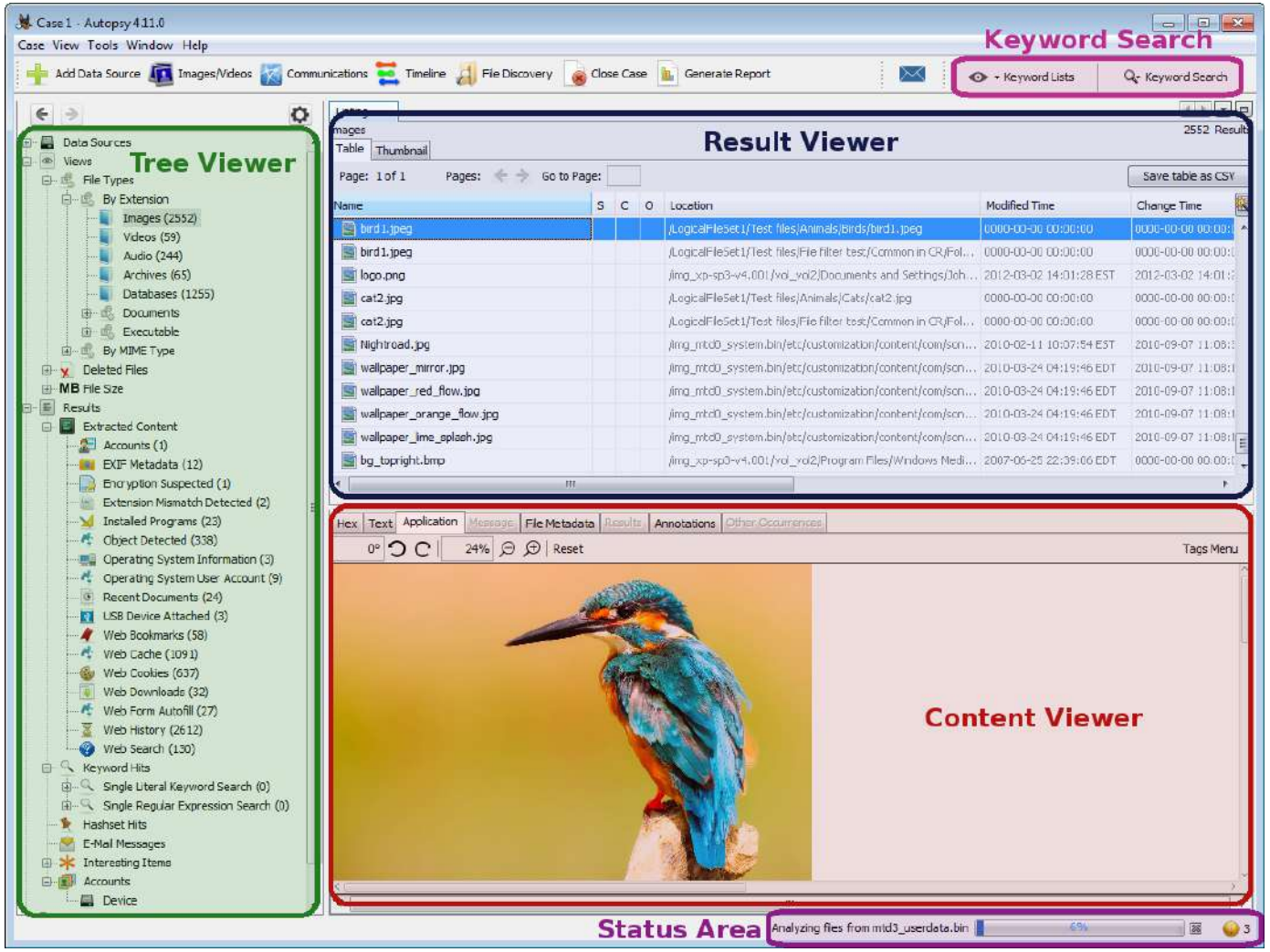
From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

Autopsy v4+: Case's "Data Source Summary"



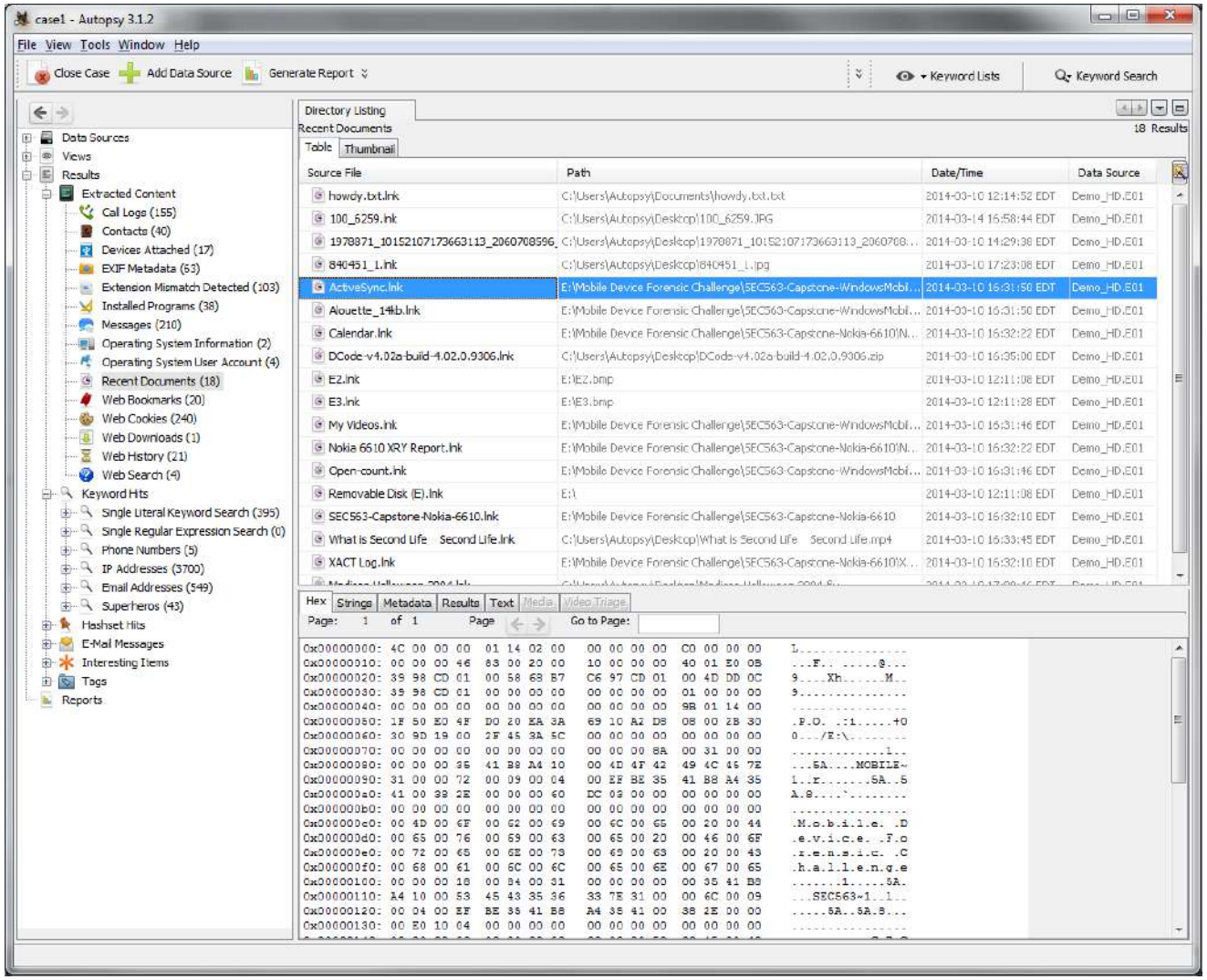
From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

Autopsy v4+: Interface Window



From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

Autopsy v4+: Interface Window

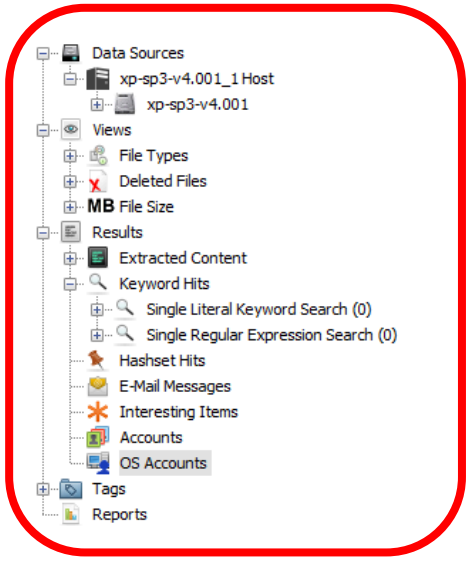


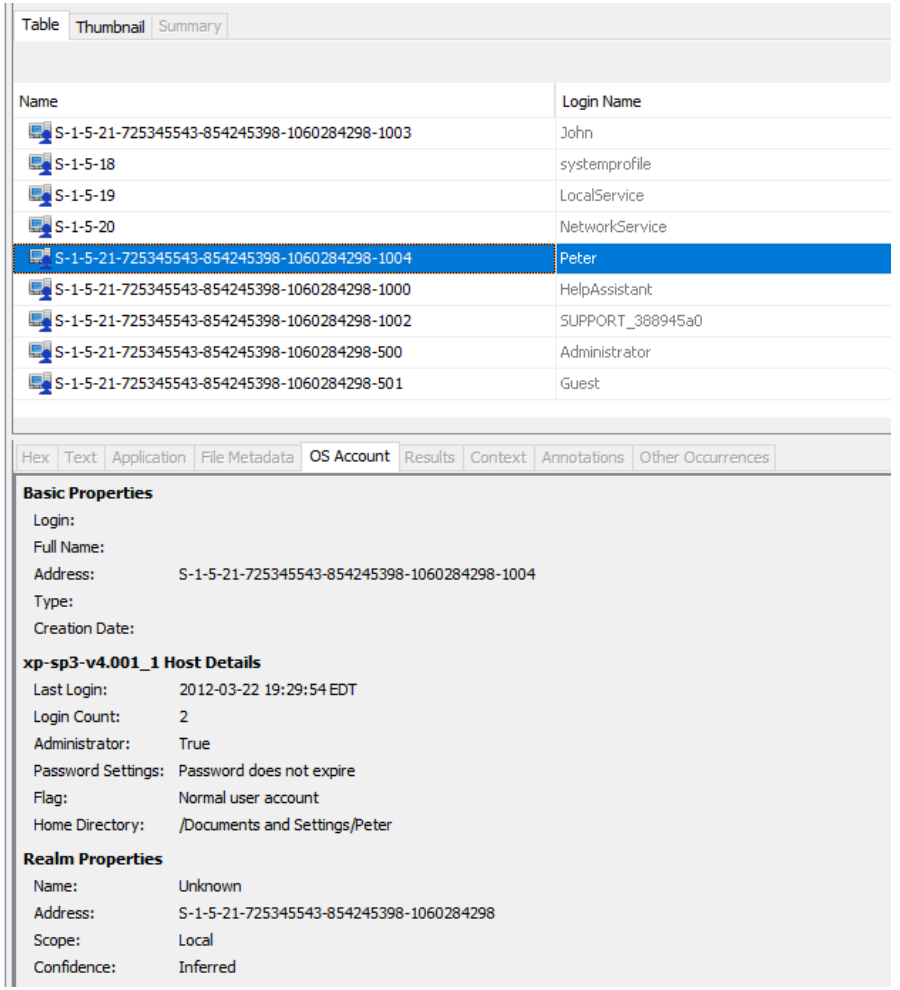
From: https://sleuthkit.org/autopsy/docs/user-docs/3.1/quick_start_guide.html

Autopsy v4+ Interface: Tree Viewer

- Seven main areas/sections:
  - **Persons / Hosts / Data Sources**
  - **Views**: shown aggregately from more than one data source by **file type**, **deleted files**, **file size**
  - **Results:**
    - **Extracted Content**: data artifacts created by running ingest, e.g. call logs and messages from communication logs
    - **Analysis Results**: results from running ingest, e.g. **Keyword Hits**, **Hashset Hits**
    - **OS Accounts**: associated with a host, and the host information is displayed
  - **Tags**: files and results that have been tagged are shown
  - **Reports**: Reports that have been generated

Autopsy v4+ Interface: Tree Viewer

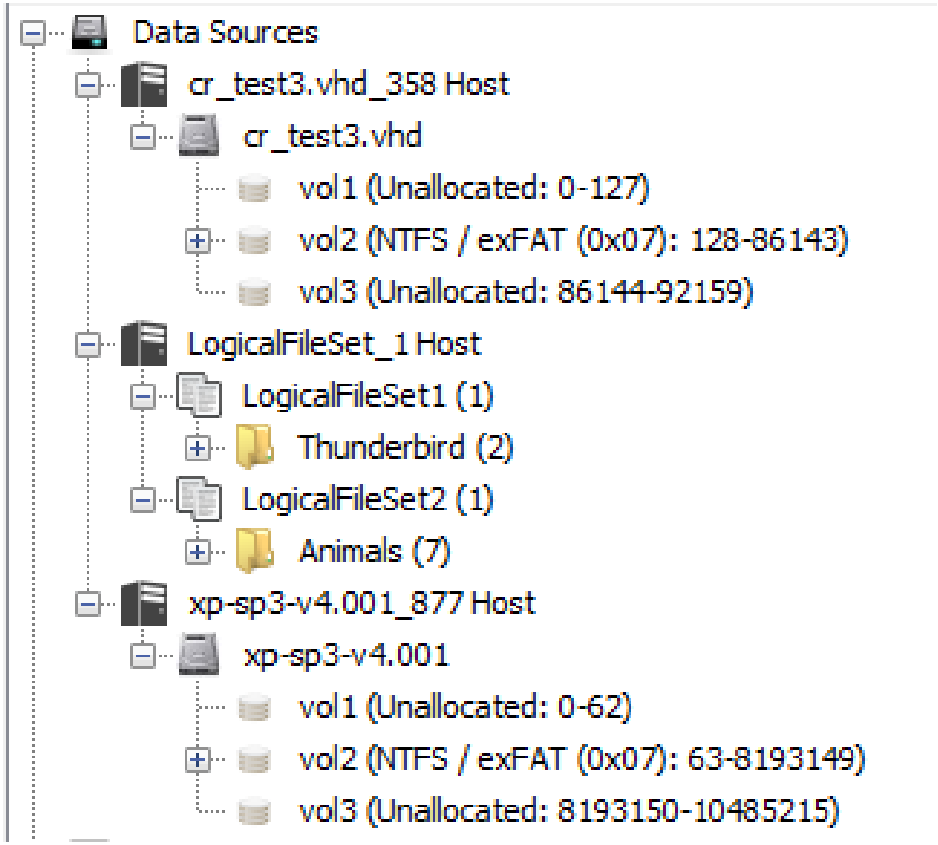




From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

Autopsy v4+ Interface: Tree Viewer

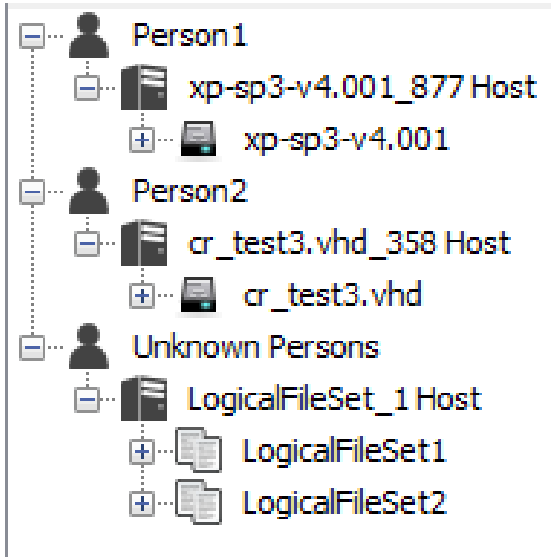
The default mode of "Persons / Hosts / Data Sources" section:

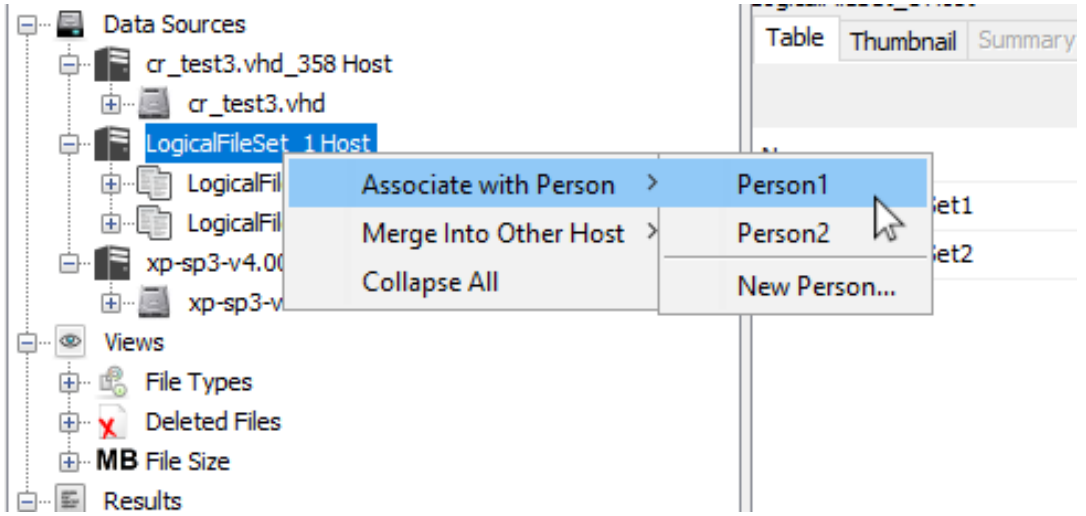


From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

Autopsy v4+ Interface: Tree Viewer

After the **"Group by Person/Host" option** in the View Options is selected:

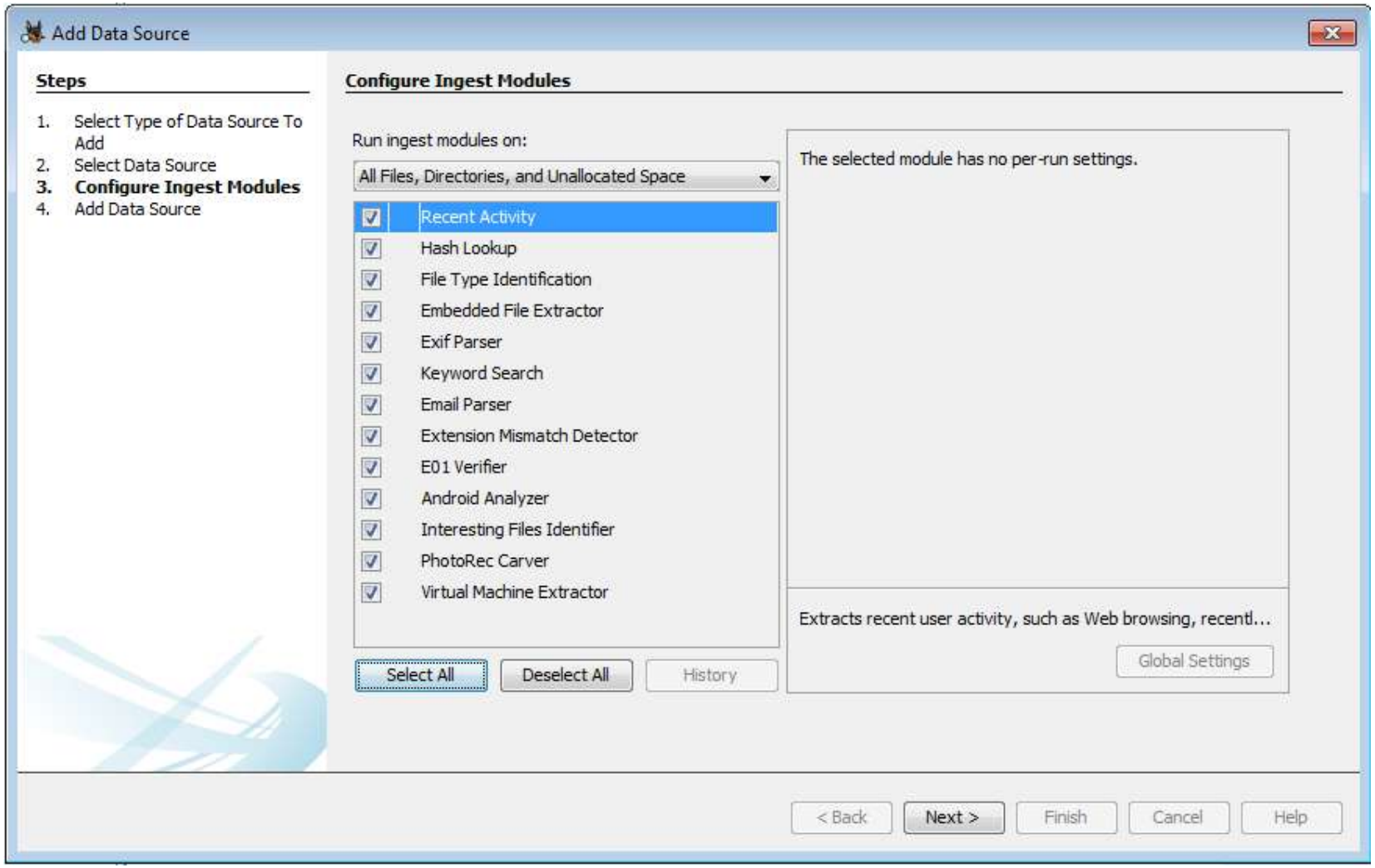




Persons are **manually created** and can be **associated** with one or more hosts

From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

Autopsy v4+: Ingest Modules



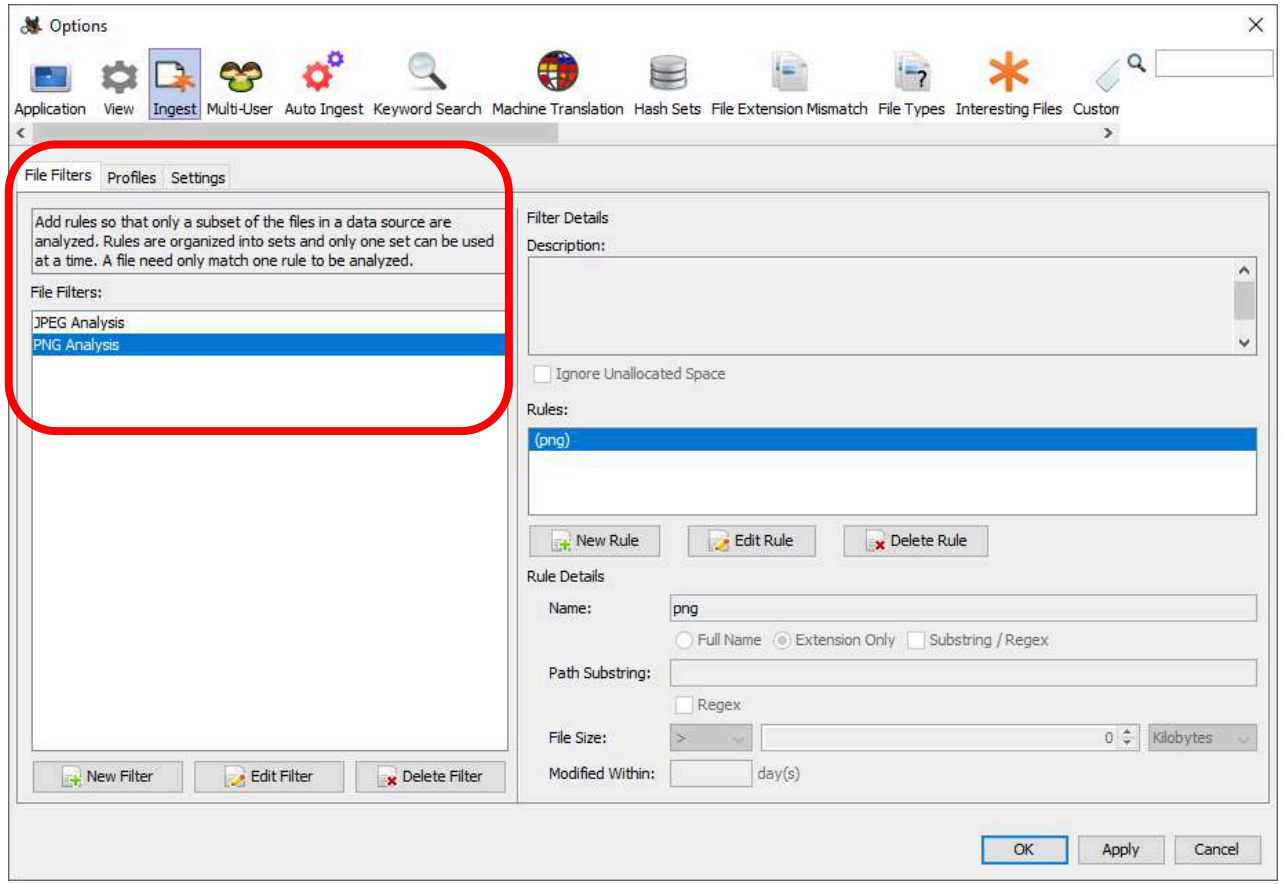
From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//ingest_page.html

Autopsy Ingest Modules: Common Information

- **Execution**:
  - Run **automatically** as a data source is added to a case
  - Can also be run later by right-clicking on a data source and choosing "**Run Ingest Modules**"
- **Configuration**:
  - You can enable/disable each module and choose which type of files to analyze
- **Progress** and **completion** information:
  - Notification of ingest already run
  - Completed ingests
  - Ongoing ingest activity

Autopsy v4+: Ingest Module's Configuration

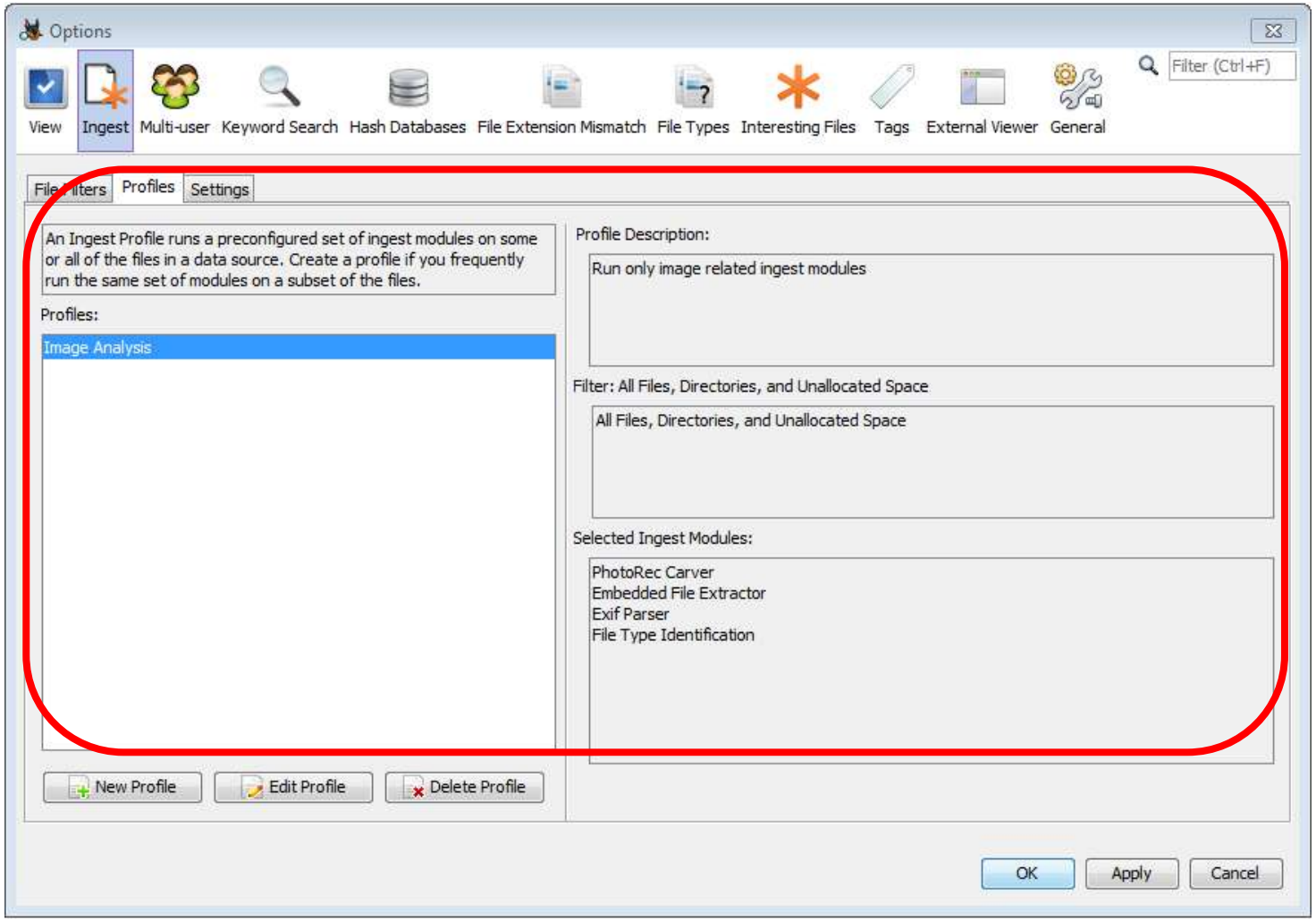
File Filters panel: can be opened from the ingest module's **selection panel** or through the **Ingest tab** on the main options panel



From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//ingest_page.html

Autopsy v4+: Ingest Module's Configuration

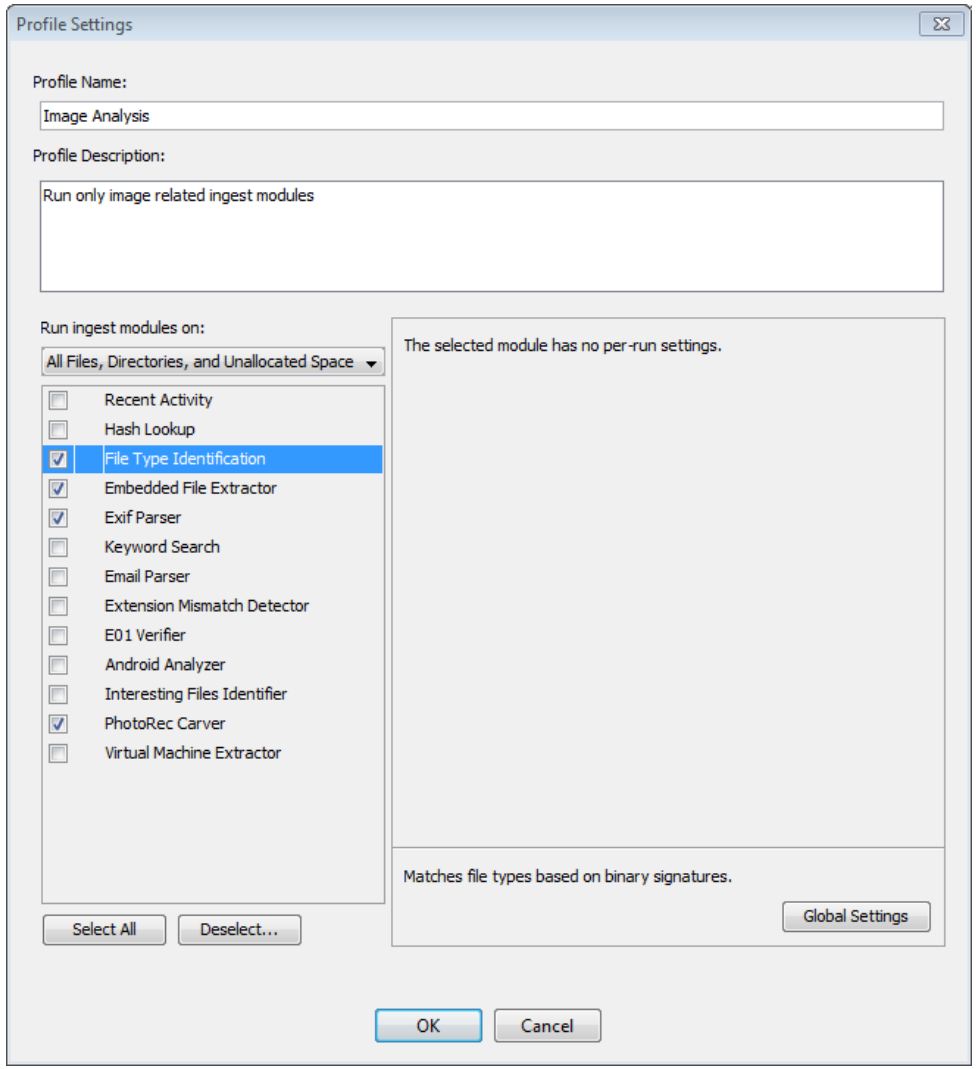
Ingest Profiles



From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//ingest_page.html

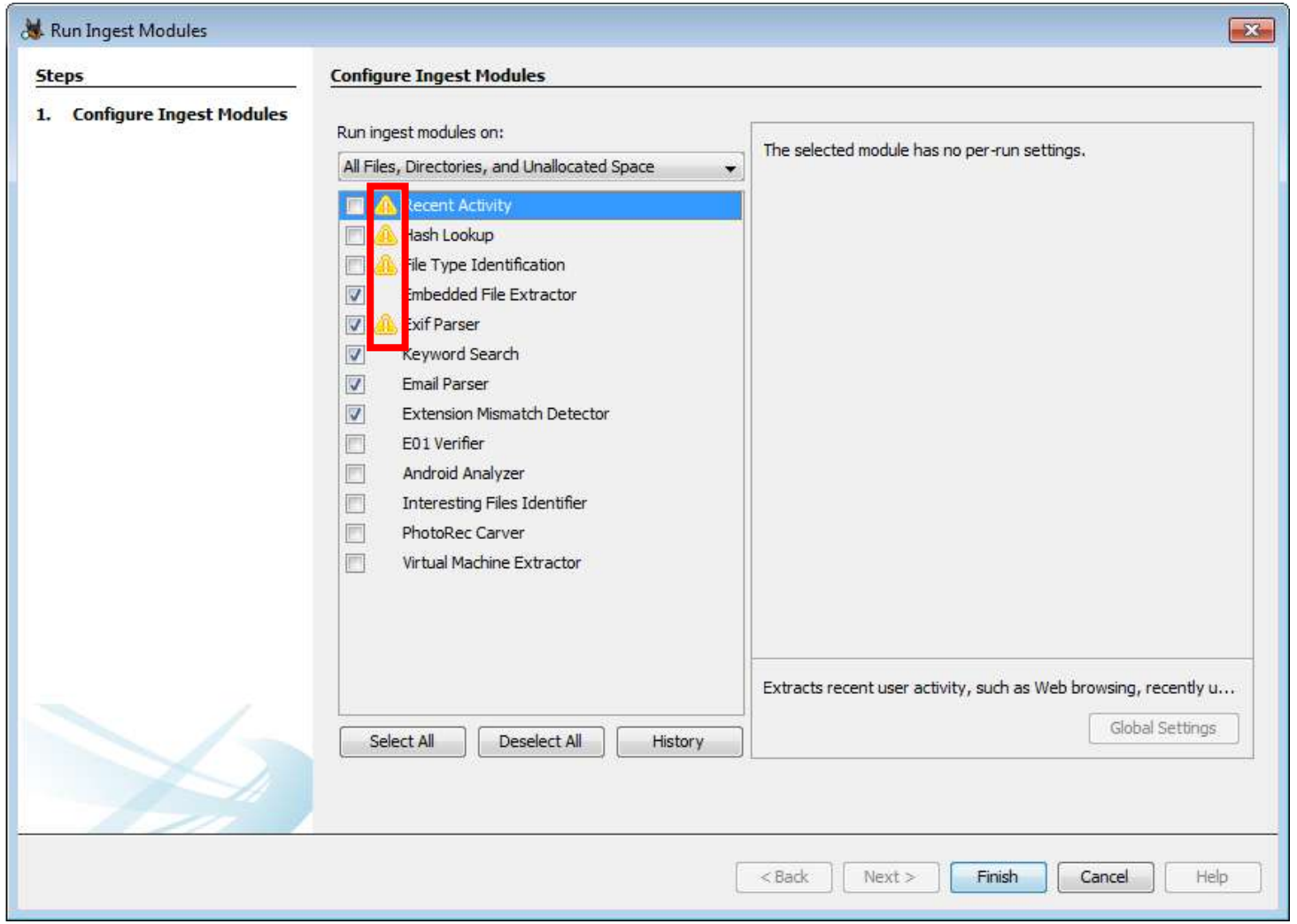
Autopsy v4+: Ingest Module's Configuration

Ingest Profiles



From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//ingest_page.html

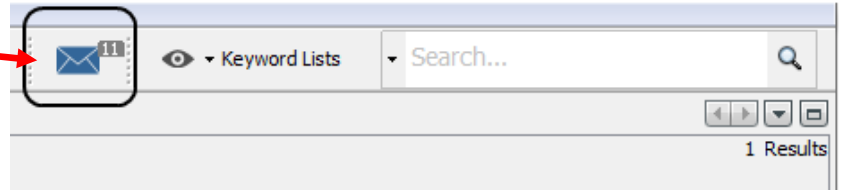
Autopsy v4+: Ingest Already-Run Alert

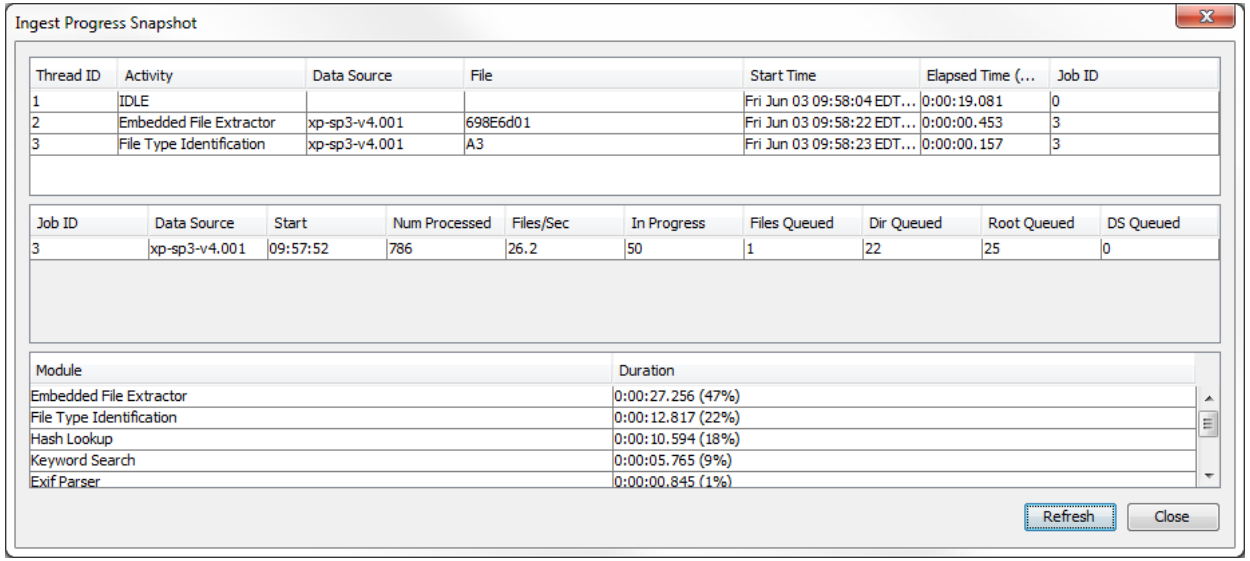


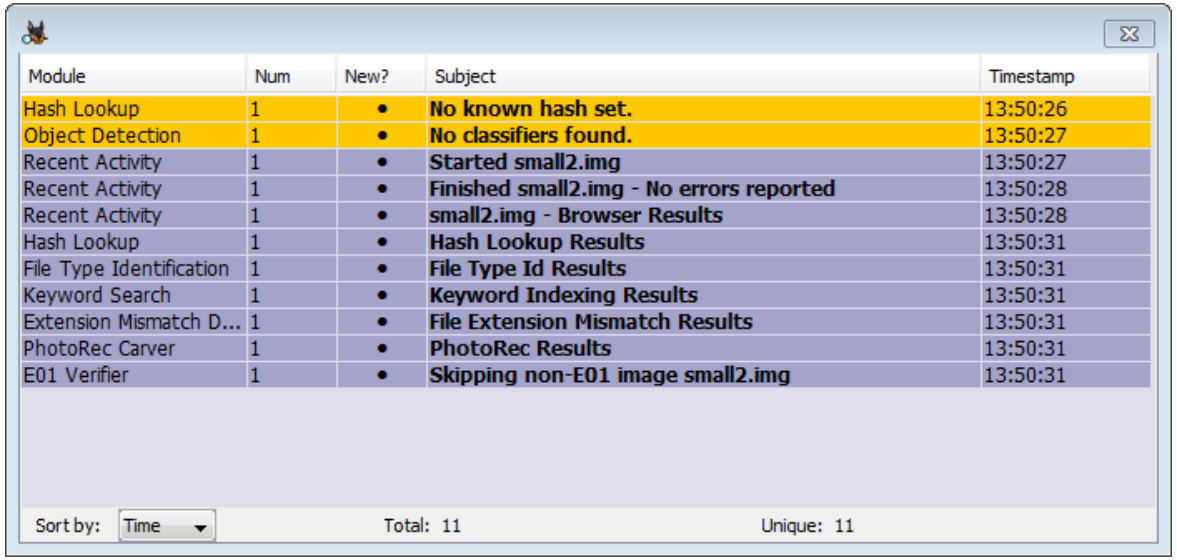
From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//ingest_page.html

Autopsy v4+: Ingest Progress/Completion Statuses

The Ingest Inbox





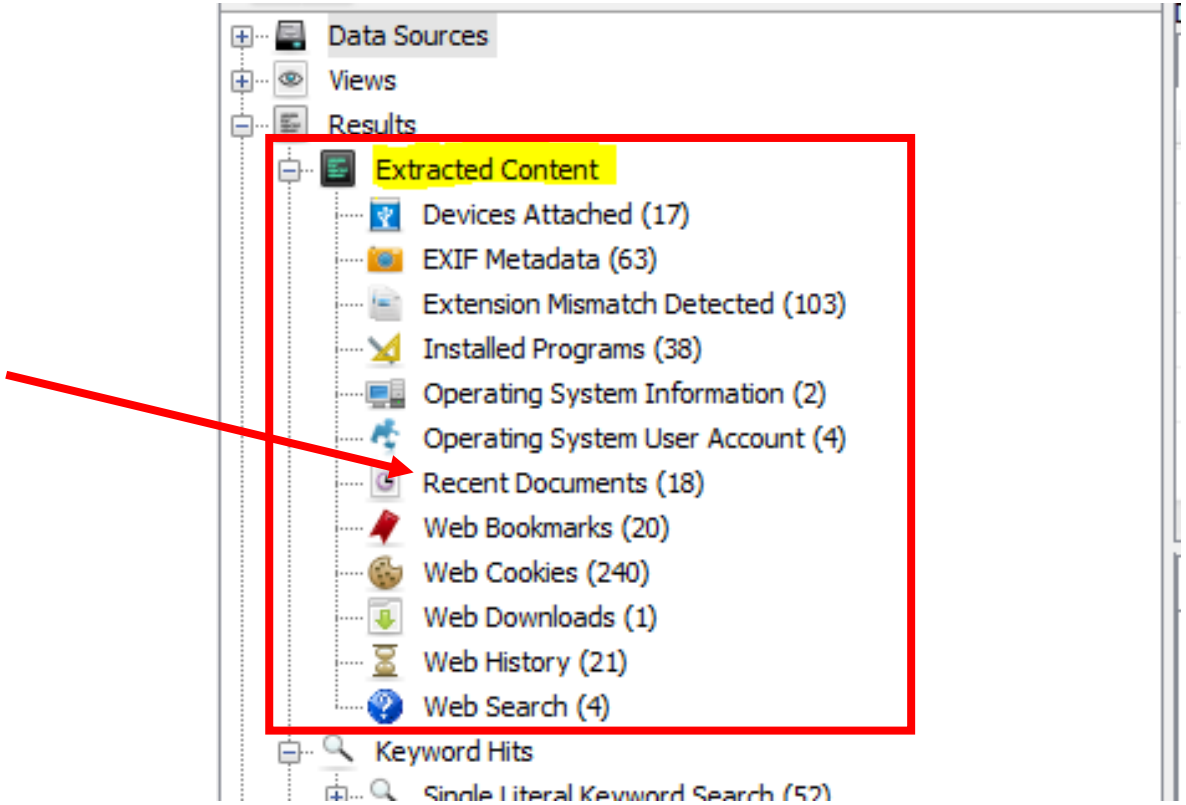


From:
http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//ingest_page.html

Some Useful Ingest Modules

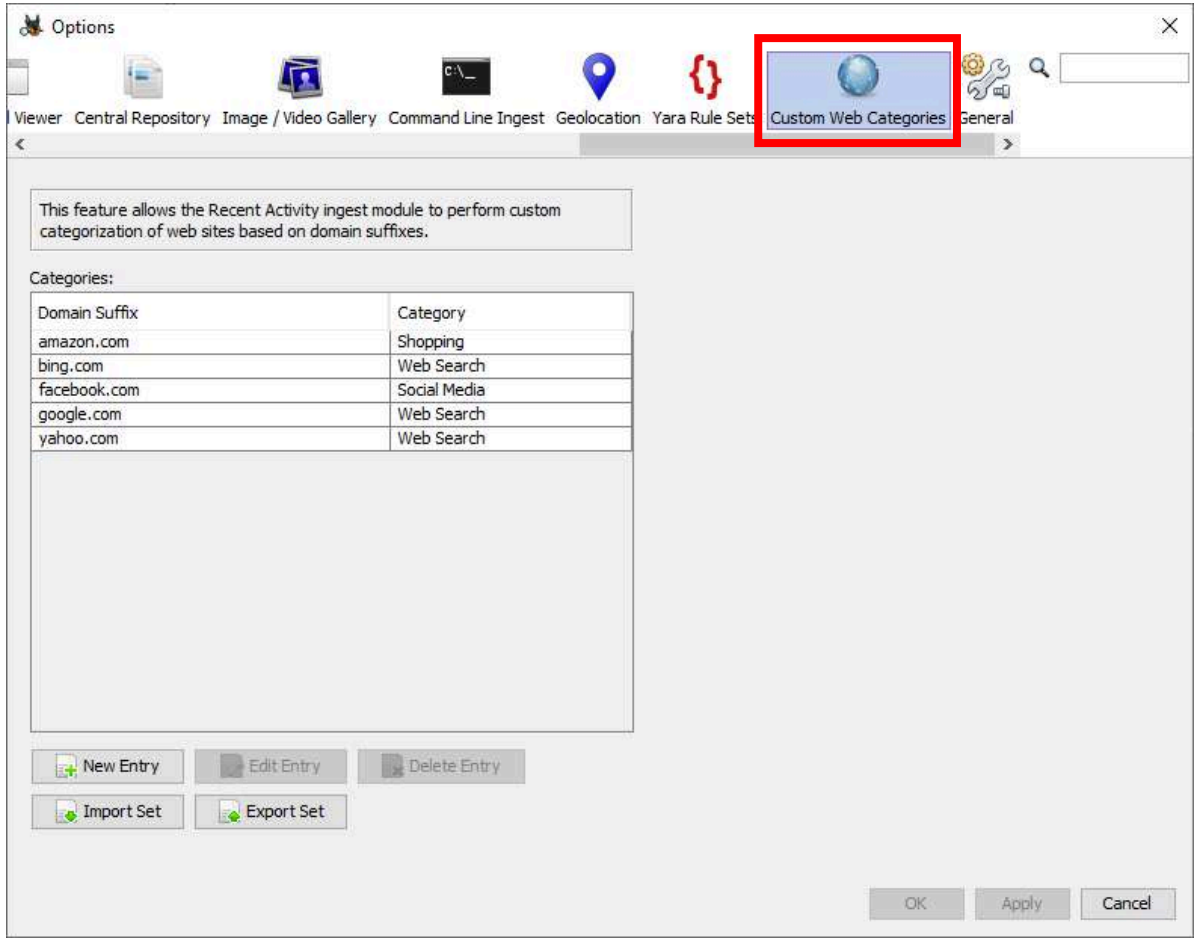
- ***Recent Activity***:
  - Extracts **user activity** as saved by web browsers (including web searches), installed programs, and OS
  - Runs **Regripper** on the **Registry** hive
  - Enables to see what activity has occurred in **the last 7 days of usage**, what web sites were visited, what the machine did, and what it connected to

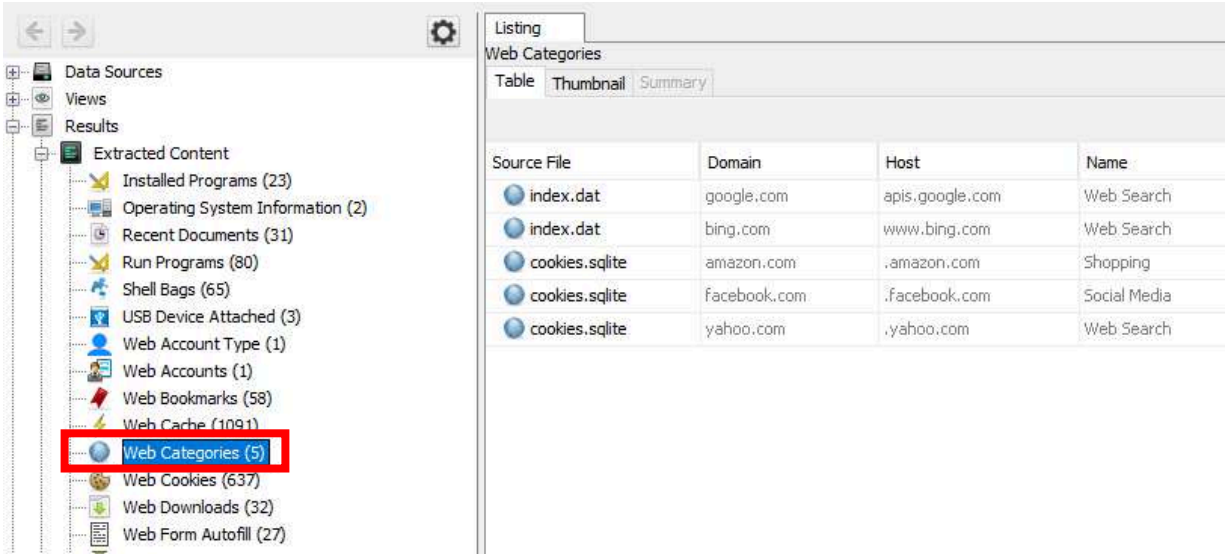
Autopsy v4+ Module: Recent Activity



From: https://sleuthkit.org/autopsy/docs/user-docs/3.1/quick_start_guide.html

Autopsy v4+ Module: Recent Activity



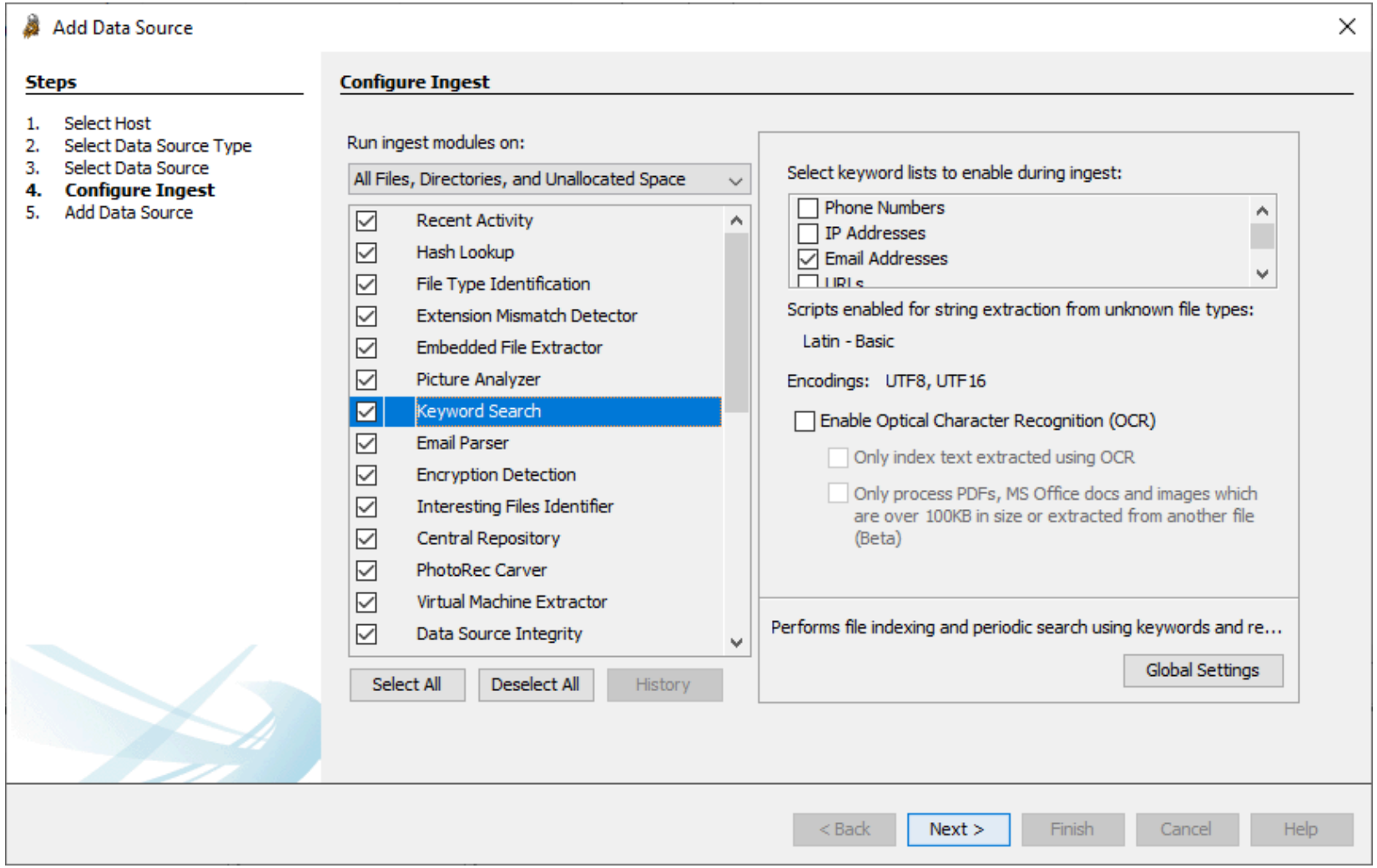


From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

Some Useful Ingest Modules

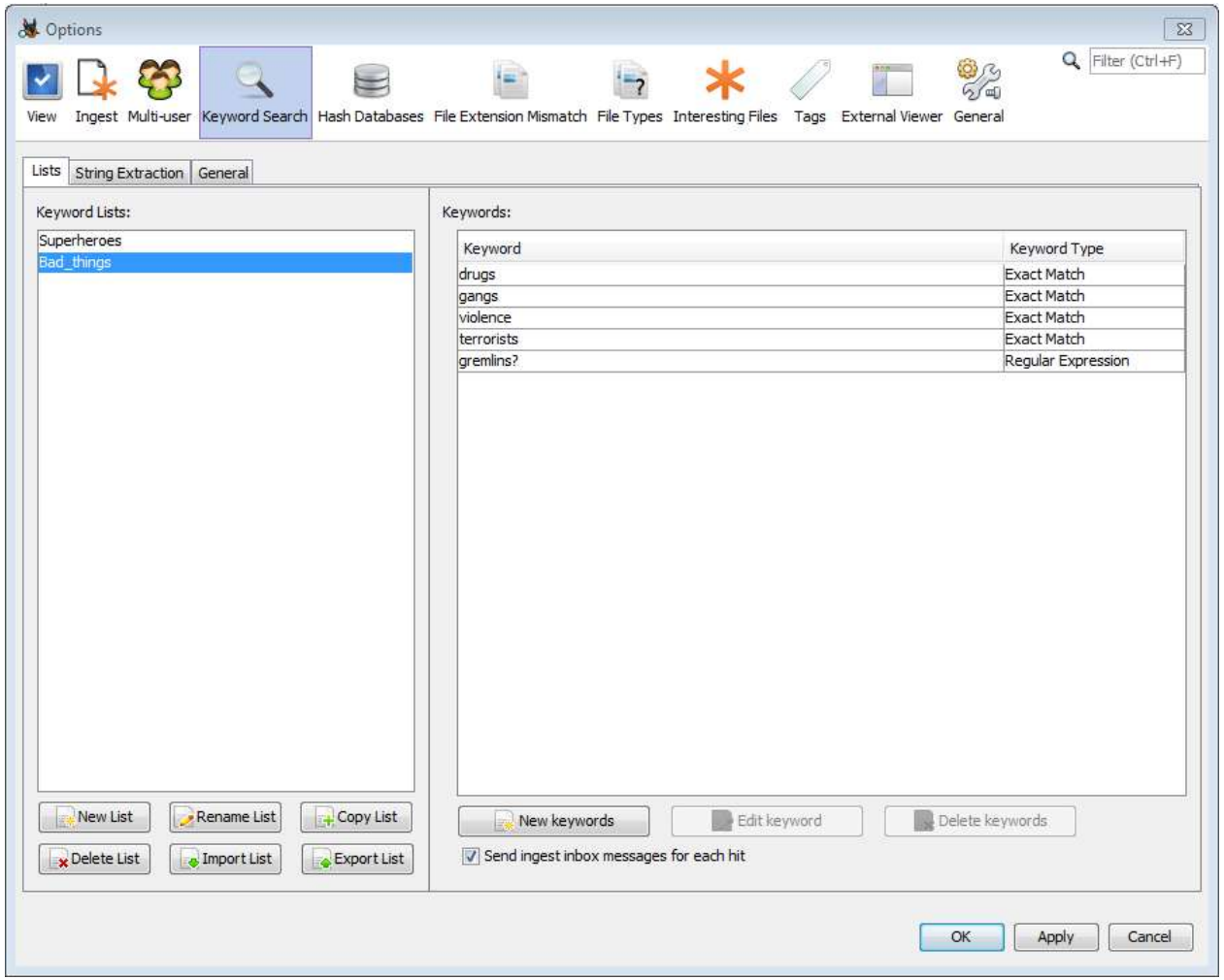
- ***Keyword Search***:
  - Facilitates both the **ingest portion of searching**, and also supports **manual text searching** after ingest has completed
  - **Extracts text** from: files being ingested, selected reports generated by other modules, and results generated by other modules
  - **Techniques** used: indexing, string extraction algorithm
  - Built-in lists of **regular expressions** for searching Phone Numbers, IP addresses, URLs and E-mail addresses:
    - This can potentially take **a long time** to complete
    - A possibility of **false positives**

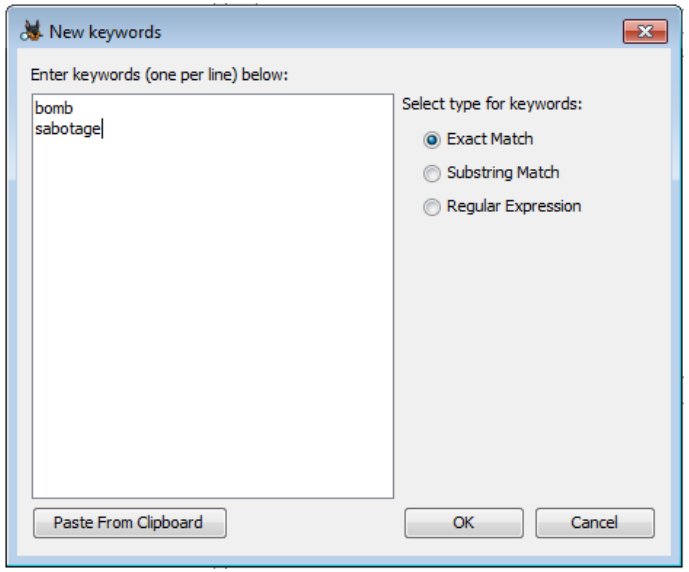
Autopsy v4+ Module: Keyword Search

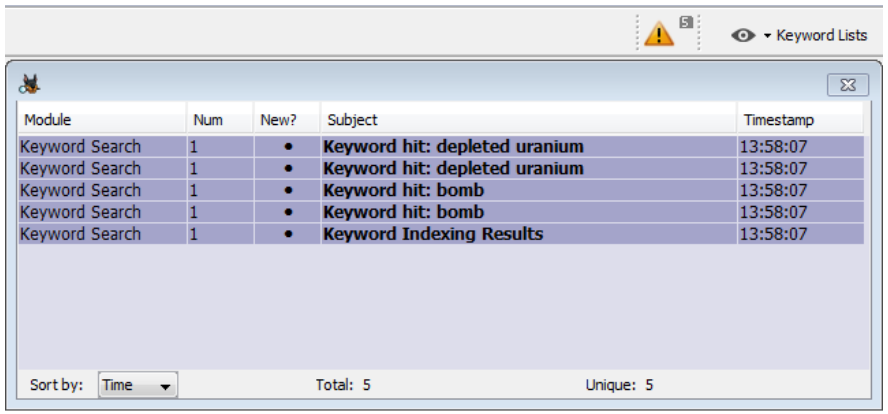


From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

Autopsy v4+ Module: Keyword Search's Options

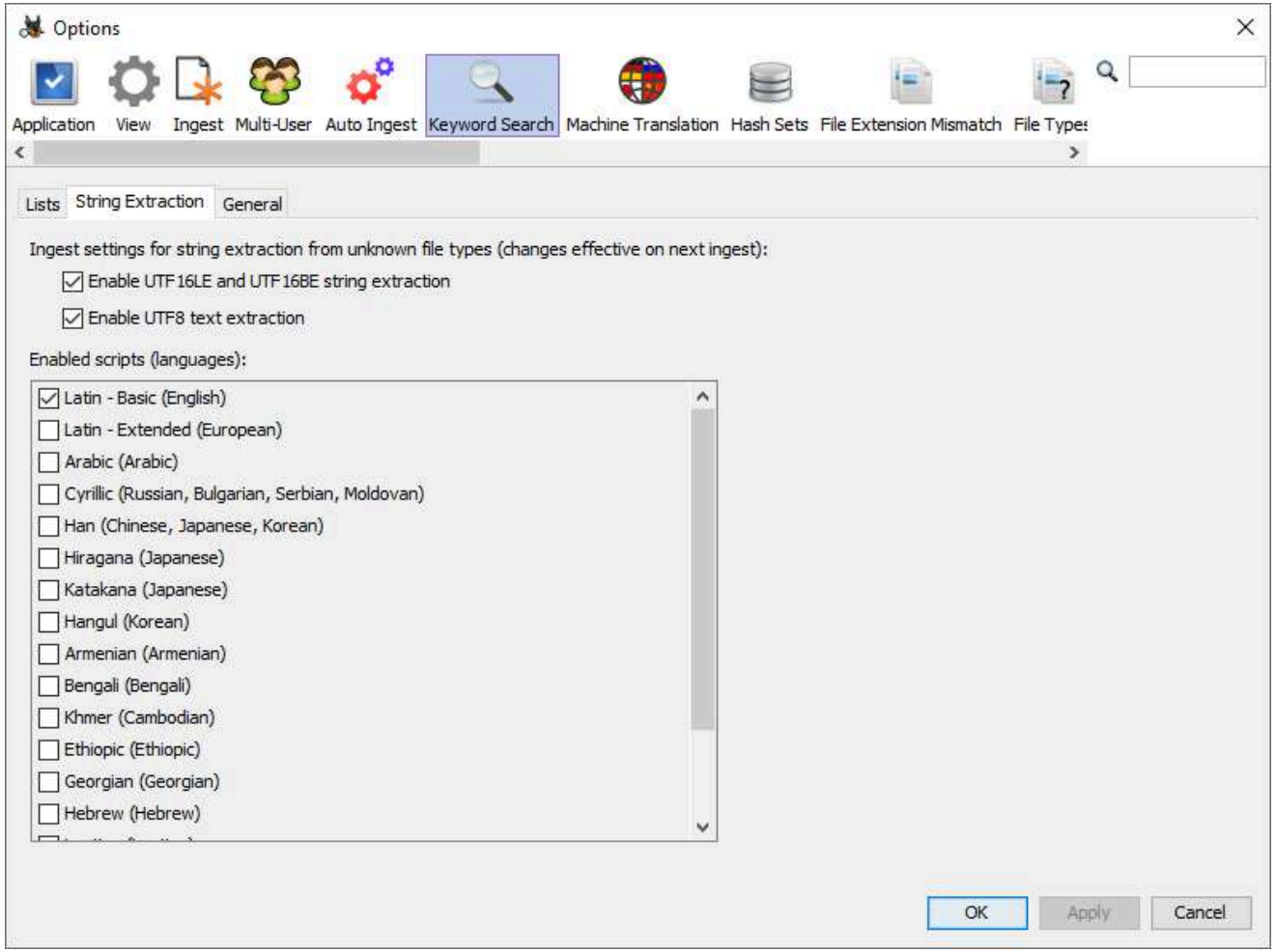






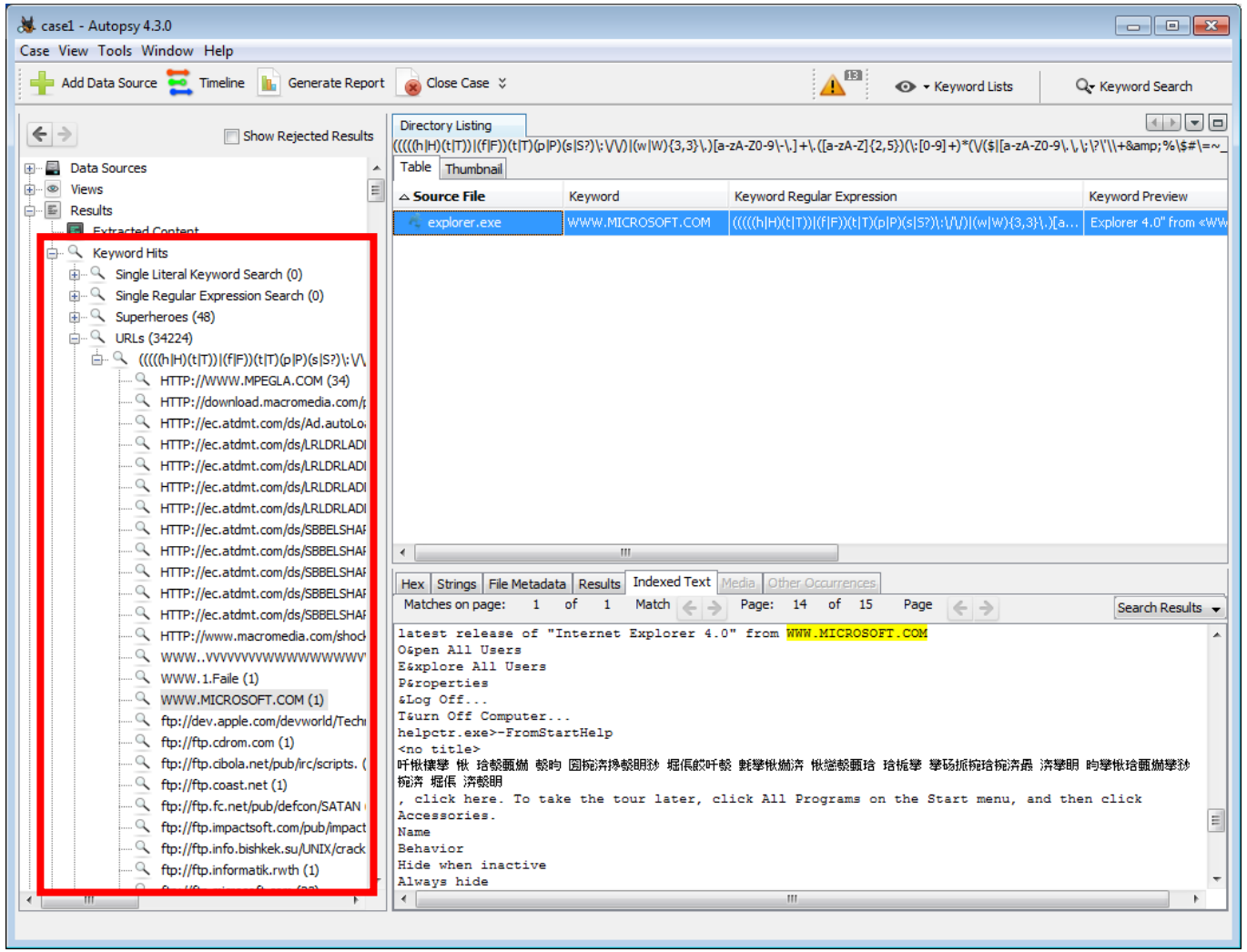
From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

Autopsy v4+ Module: Keyword Search's Options



From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2

Autopsy v4+ Module: Keyword Search's Results

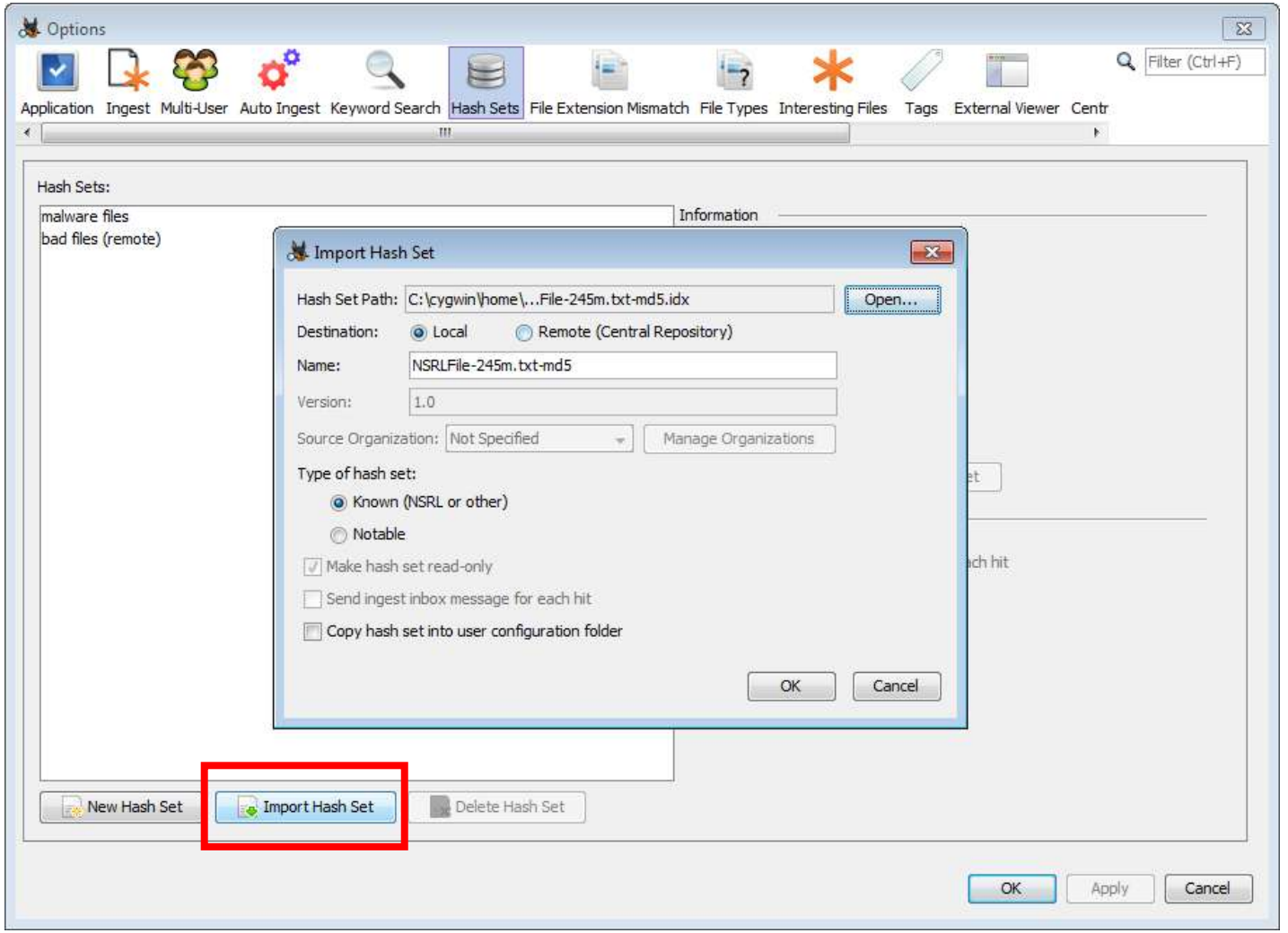


From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2

Some Useful Ingest Modules

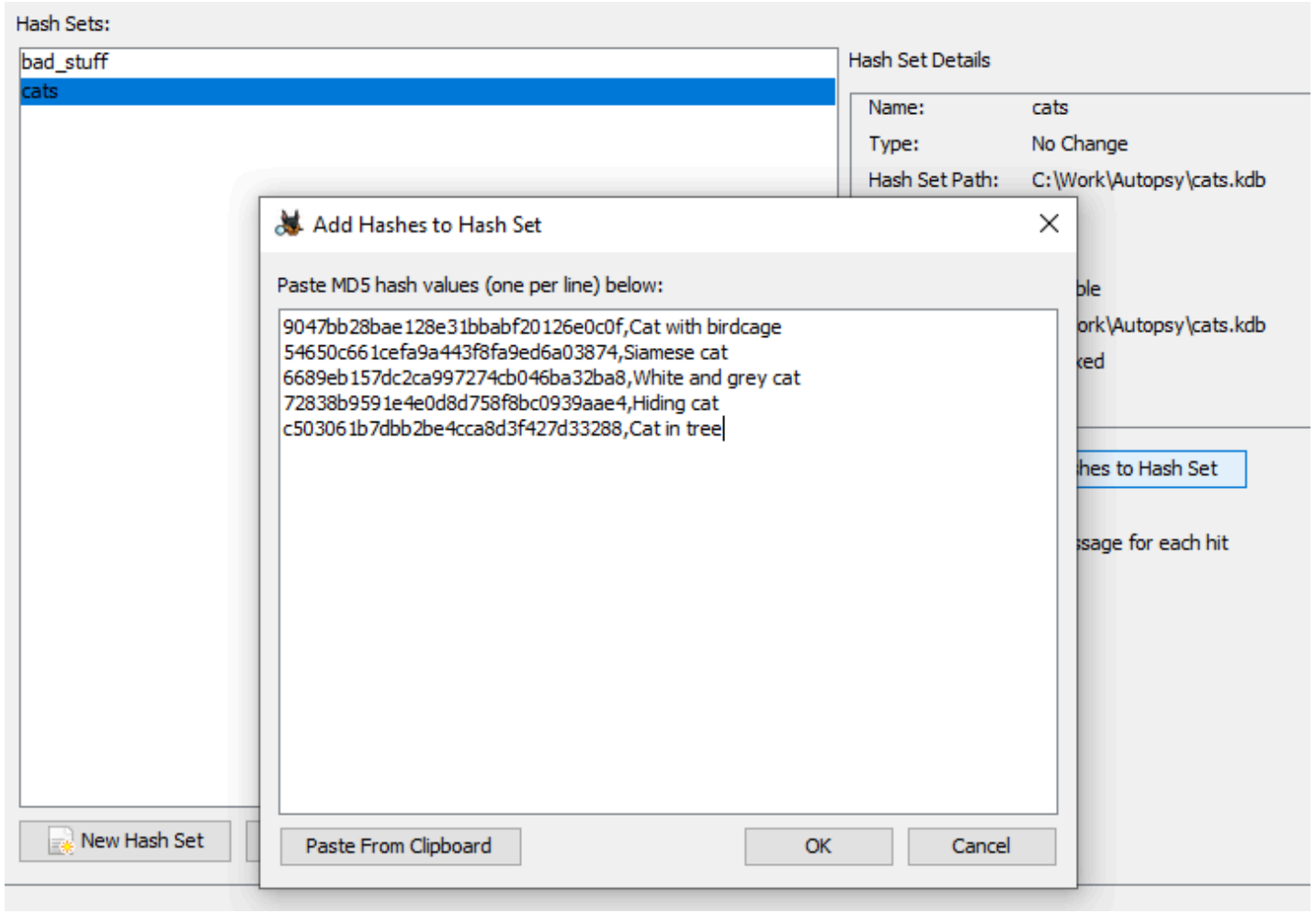
- ***Hash Lookup***:
  - Calculates MD5 hash values for files,
    and looks them up in a database (existing hash set)
  - Determines if the files are **known good files**, ***notable*** **(known bad) files**, **included/contained** in a specific set of files, or **unknown files**
- **Steps**:
  - Import a **database**/hash-set, e.g. **NIST National Software Reference Library (NSRL)** for **known** good/bad files
  - Add **hashes** into a hash set
  - **Index** the files (lookup processing)
  - Inspect the reported **hashset hits**
  - Create a **new hash set**

Autopsy v4+ Module: Hash Lookup



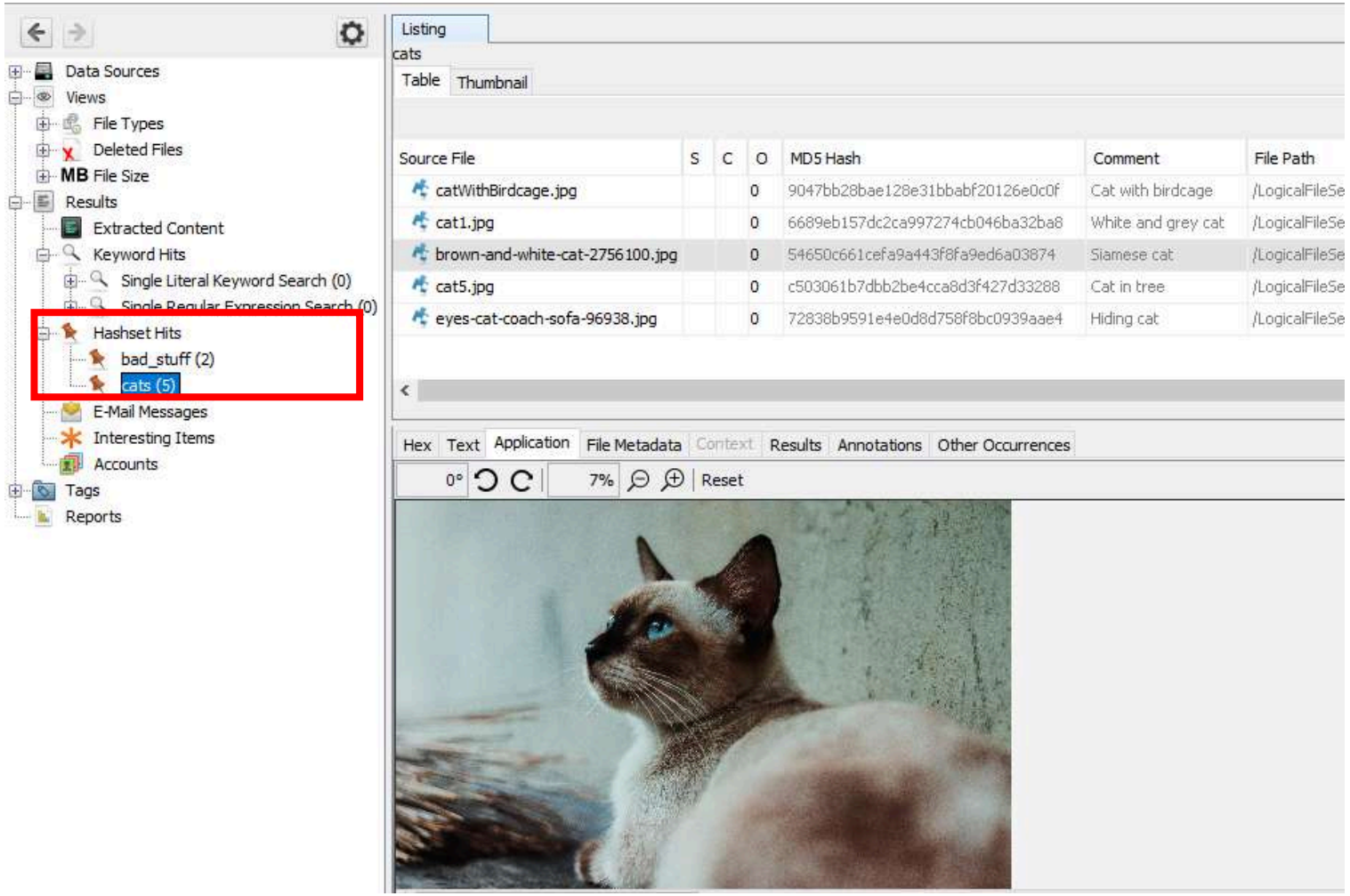
From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

Autopsy v4+ Module: Hash Lookup



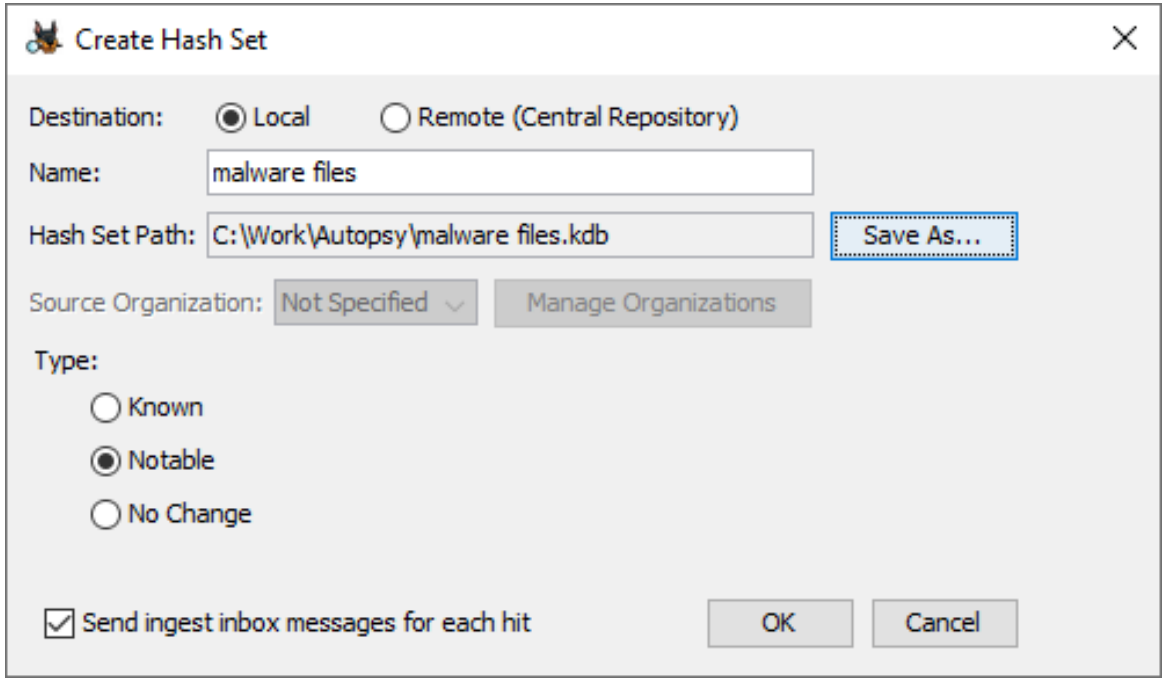
From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

Autopsy v4+ Module: Hash Lookup



From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

Autopsy v4+ Module: Hash Lookup

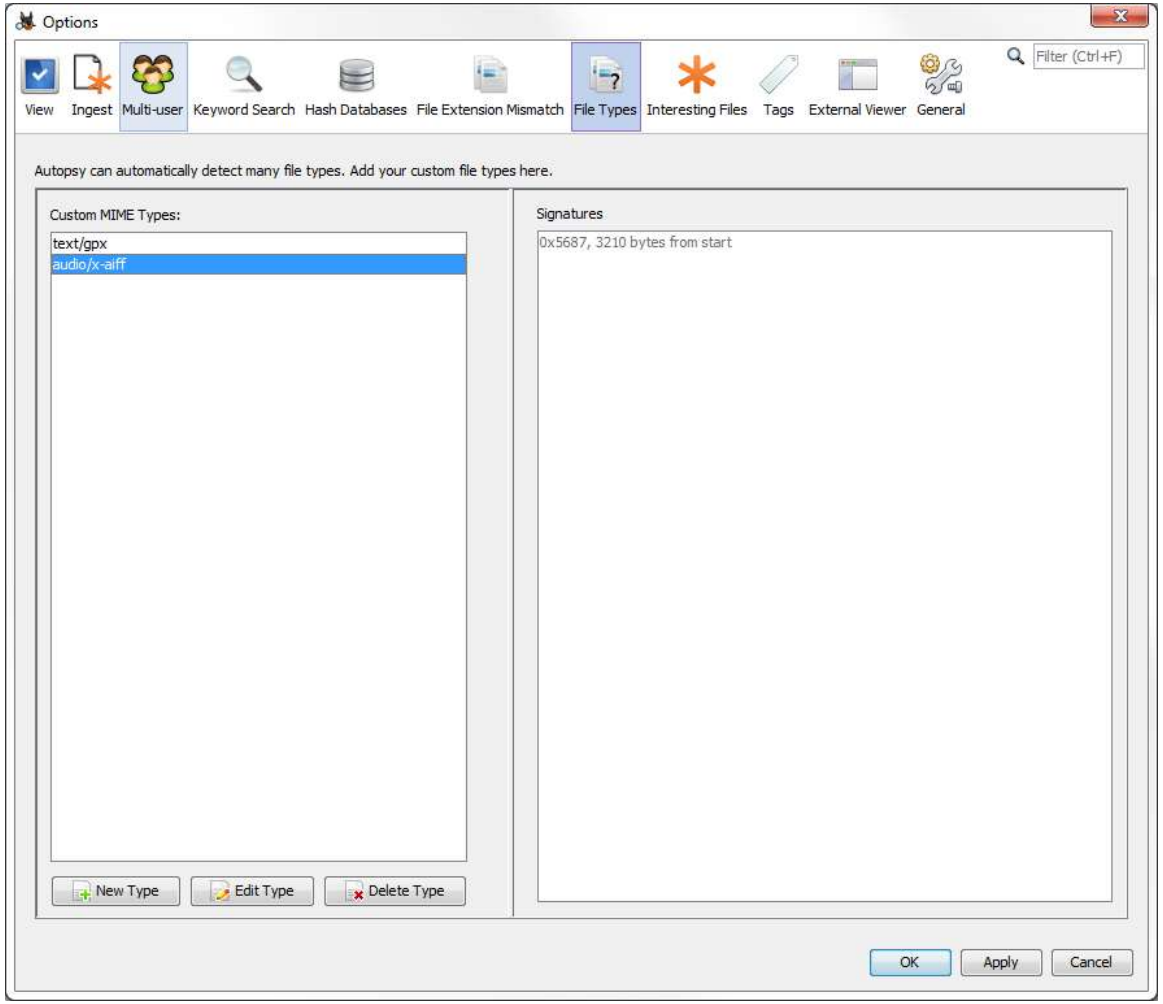


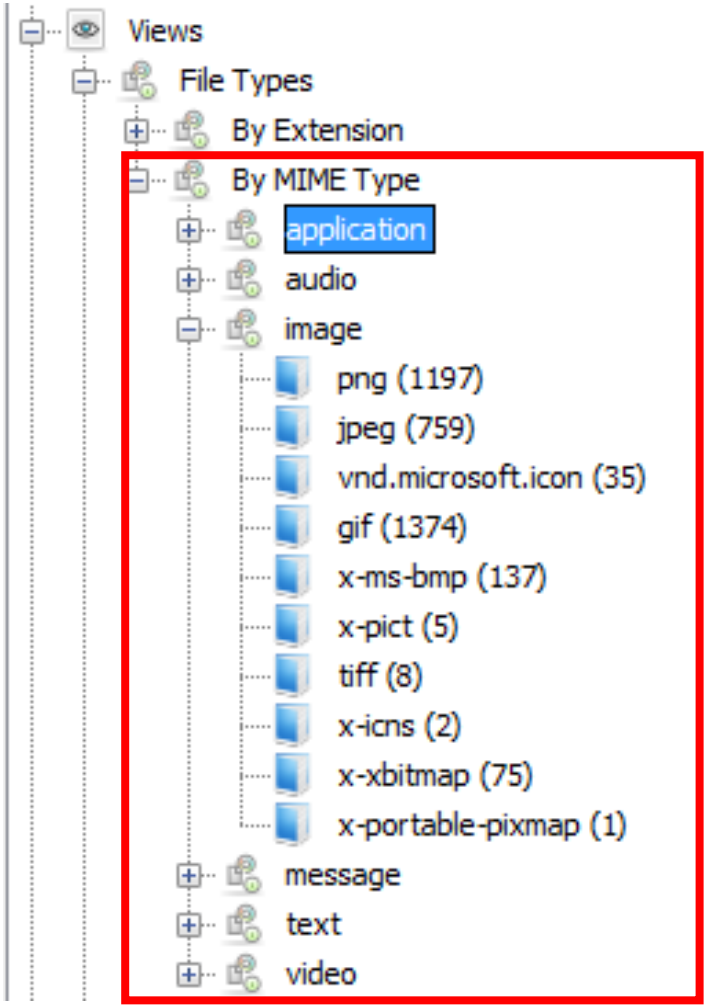
From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

Some Useful Ingest Modules

- ***File Type Identification***:
  - Does ***not*** rely on file extensions
  - Identifies files based on their ***internal signatures***
  - Uses the **Tika library** for its primary file detection, which can be customized with user-defined rules
- **Do enable this ingest module** as other modules depend on its results, including:
  - Extension Mismatch Detector module
  - Keyword Search module

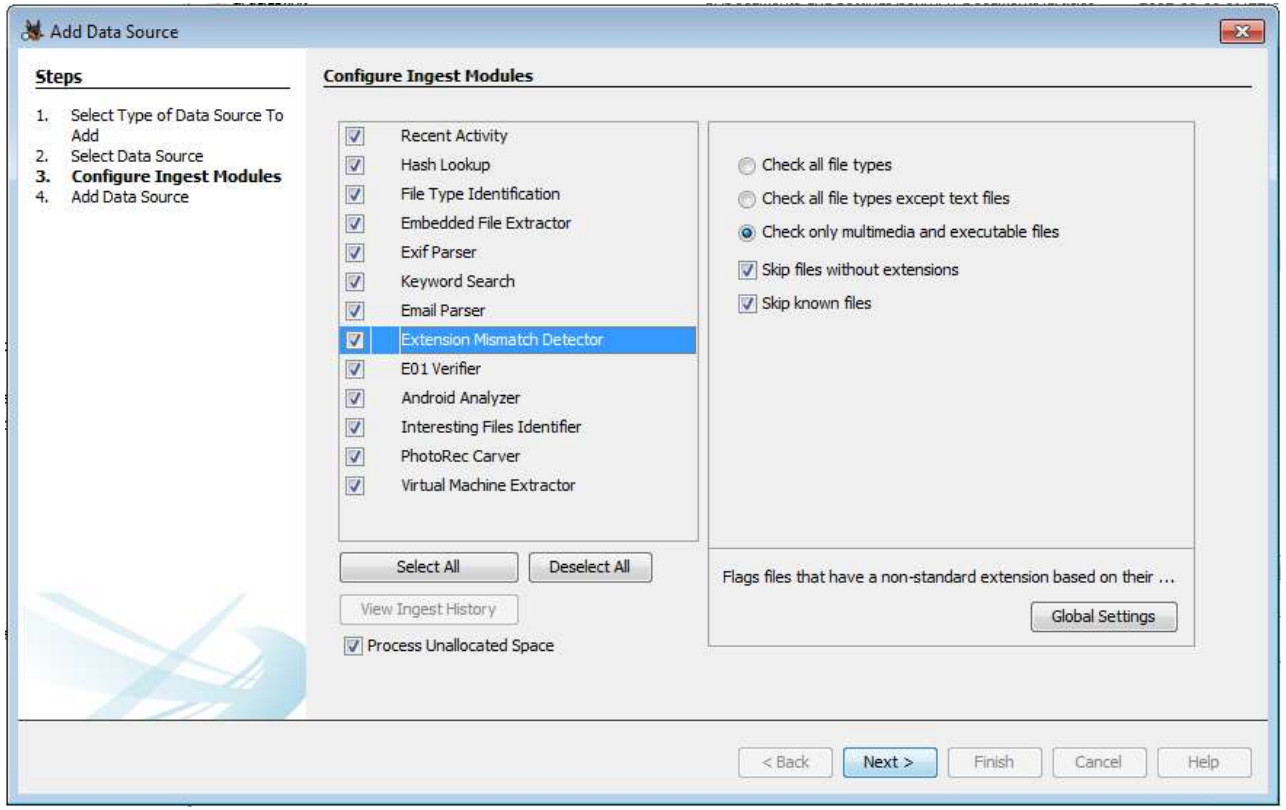
Autopsy v4+ Module: File Type Identification

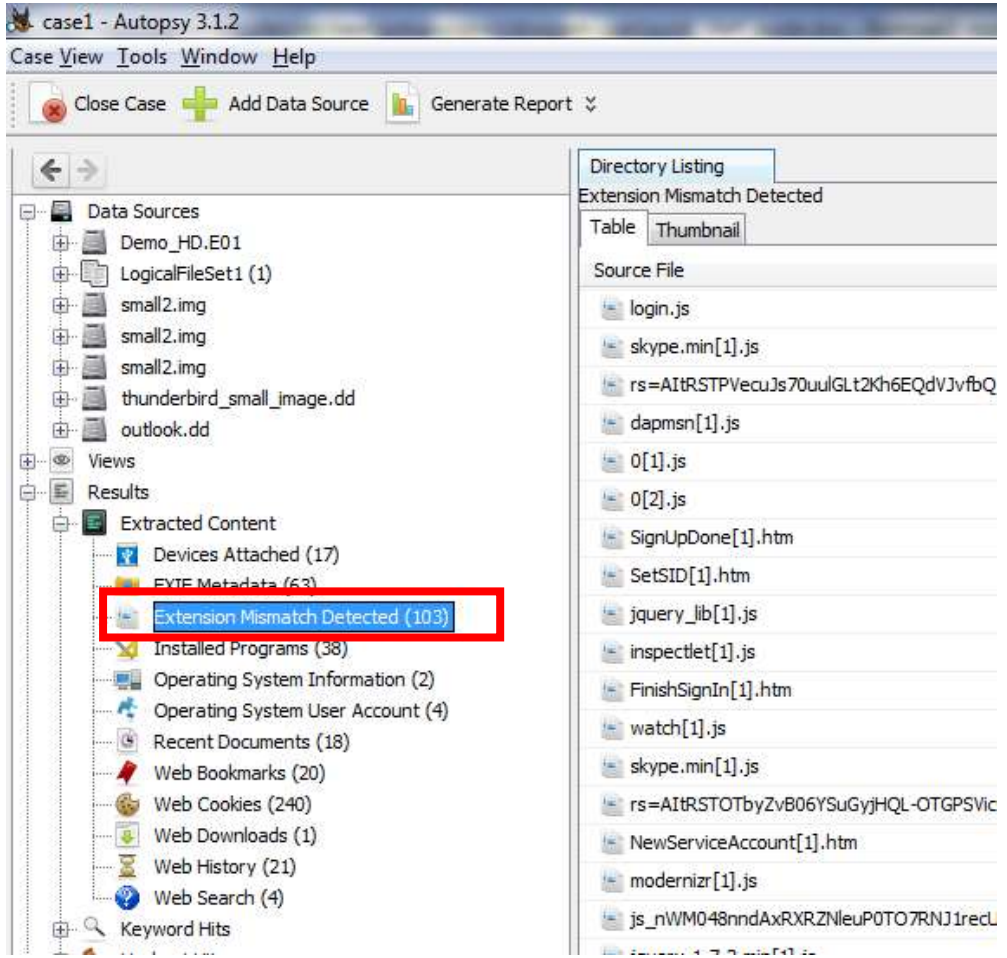




From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

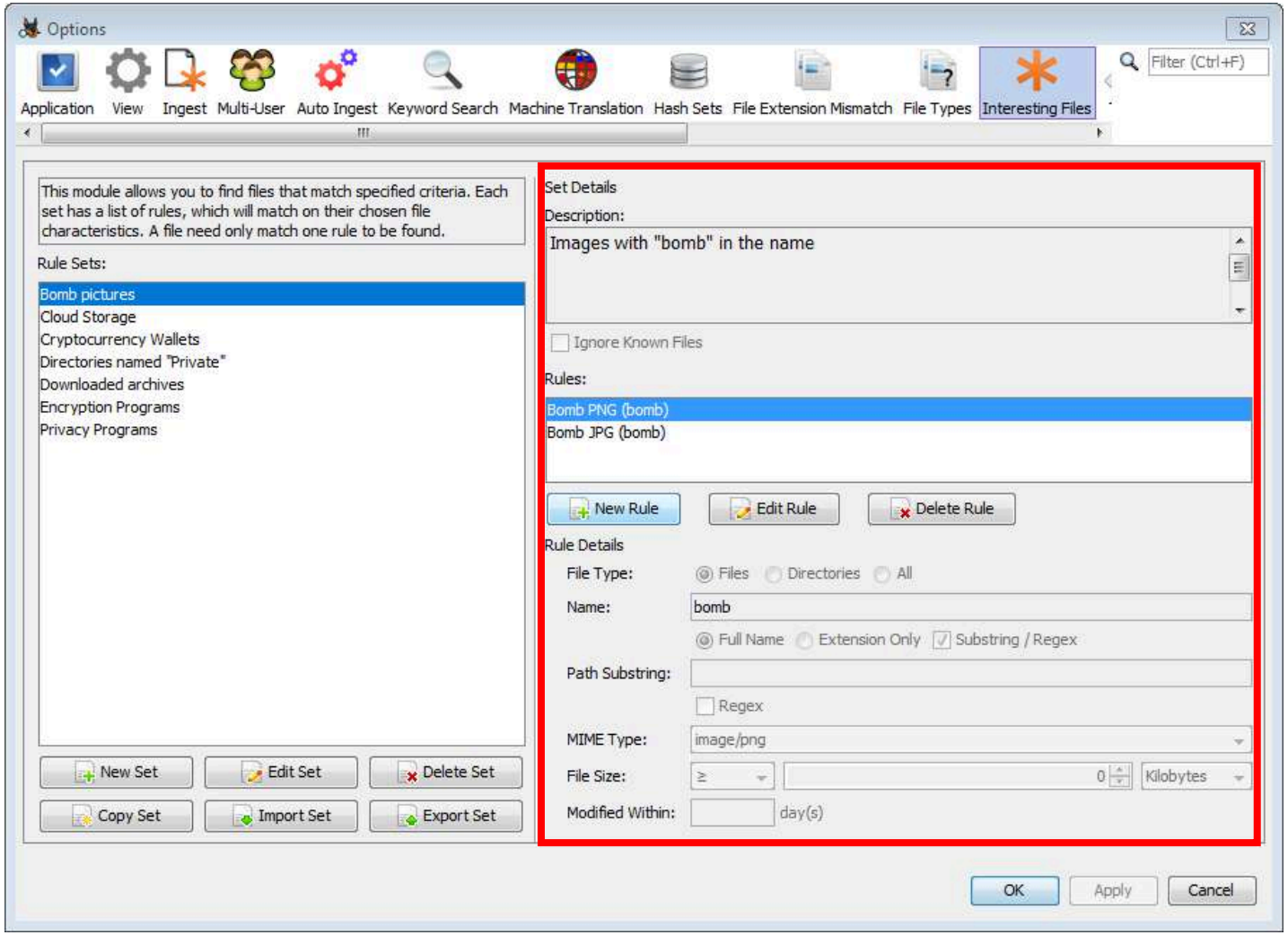
Autopsy v4+ Module: Extension Mismatch Detector





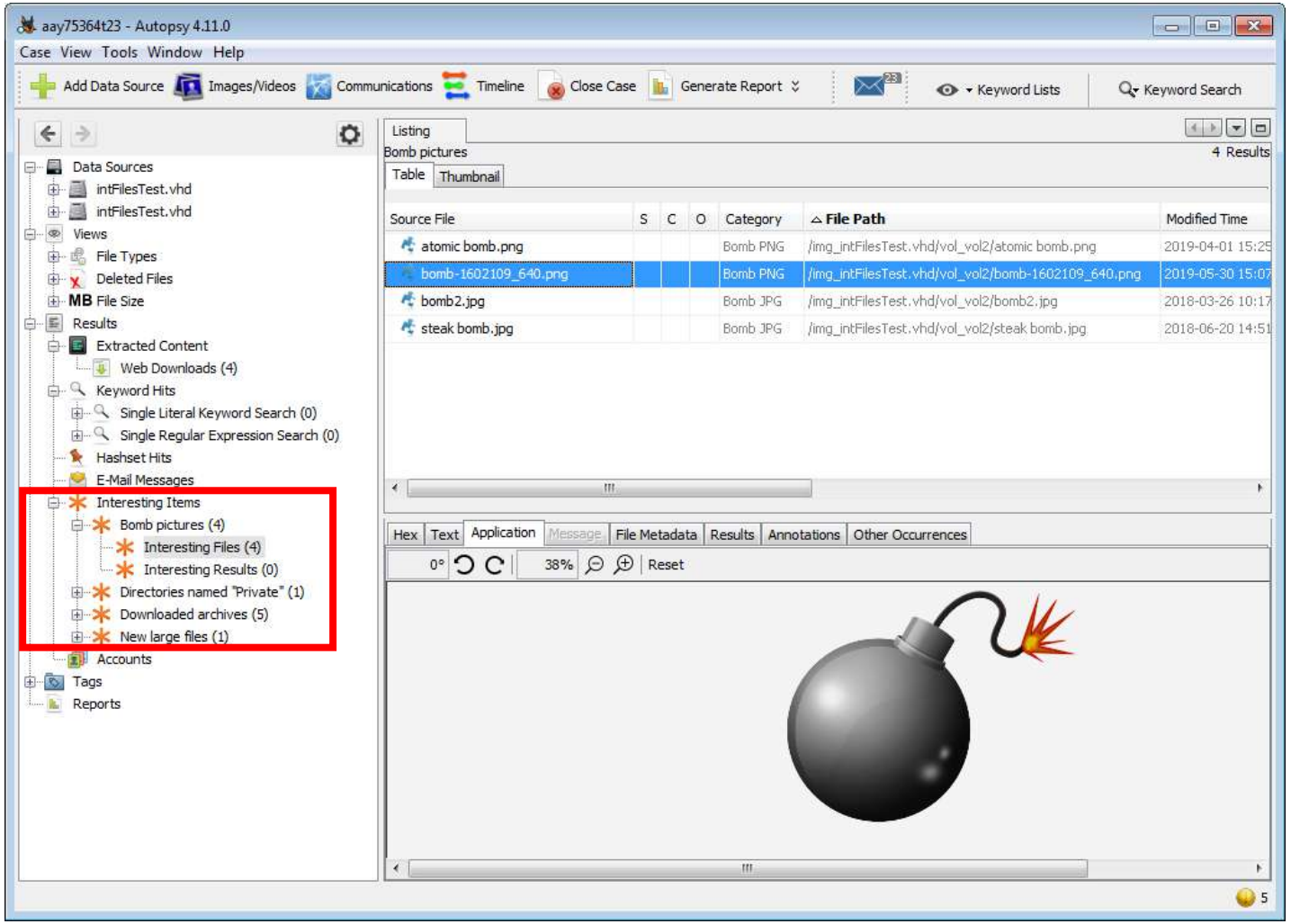
From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

Autopsy v4+ Module: Interesting Files Identifier



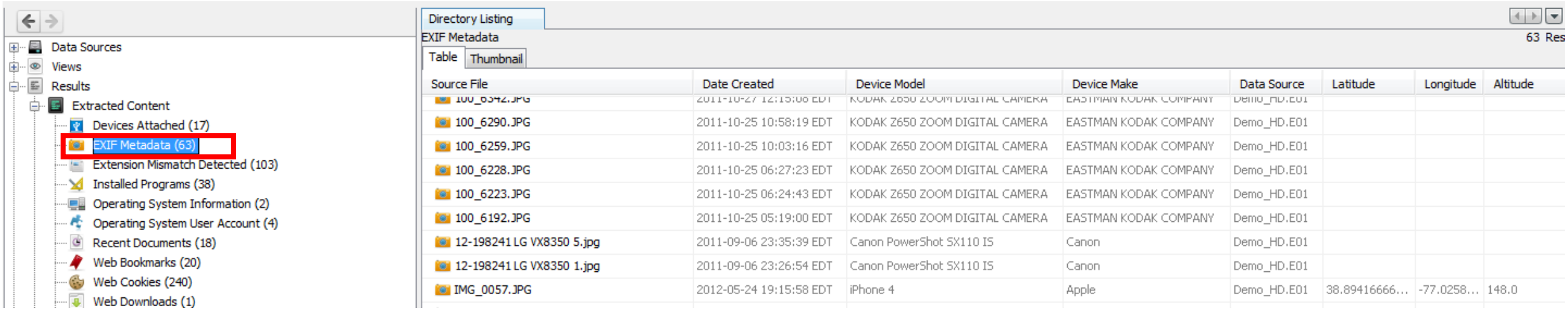
From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

Autopsy v4+ Module: Interesting Files Identifier



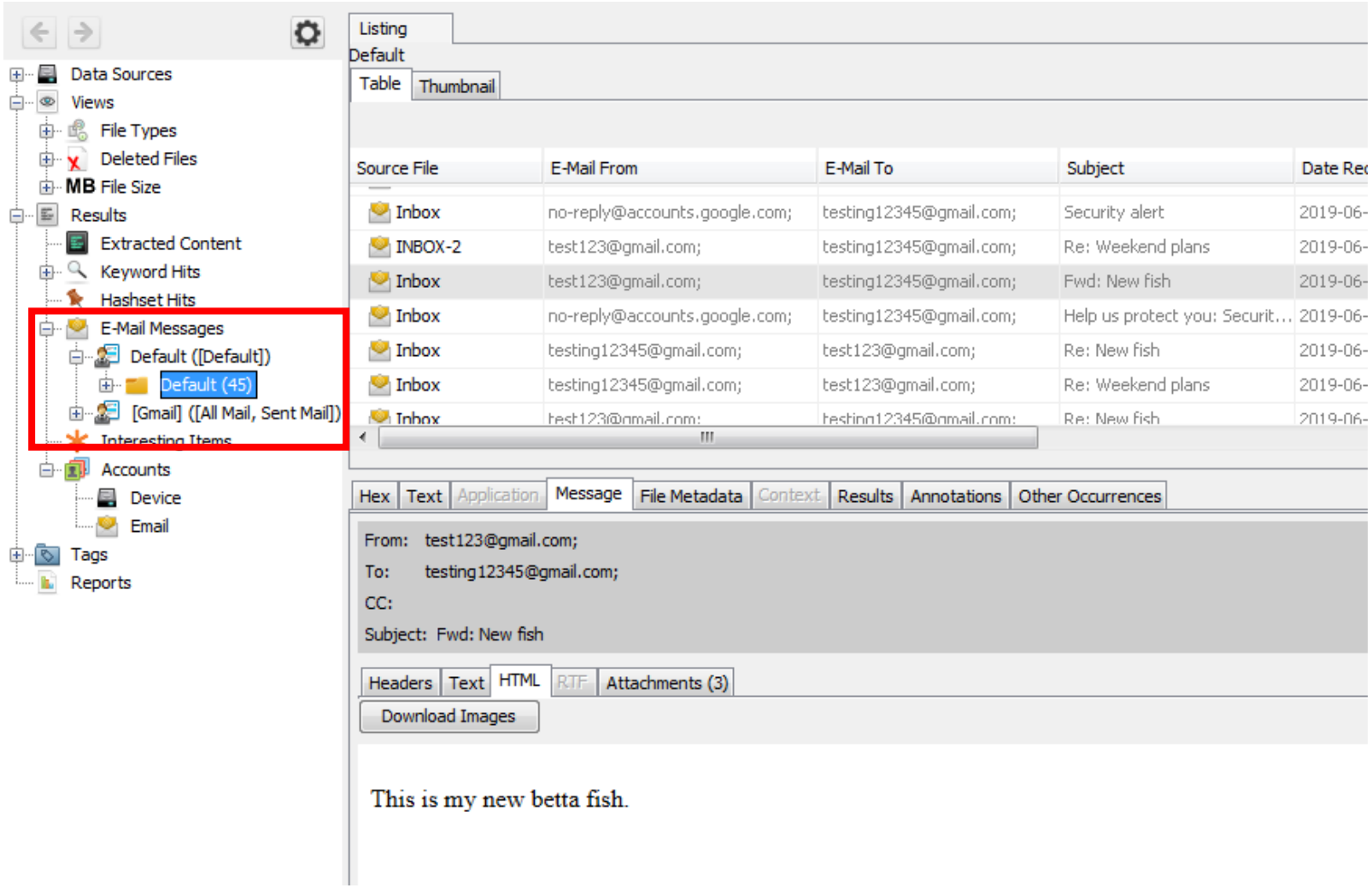
From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

Autopsy v4+ Module: Picture Analyzer



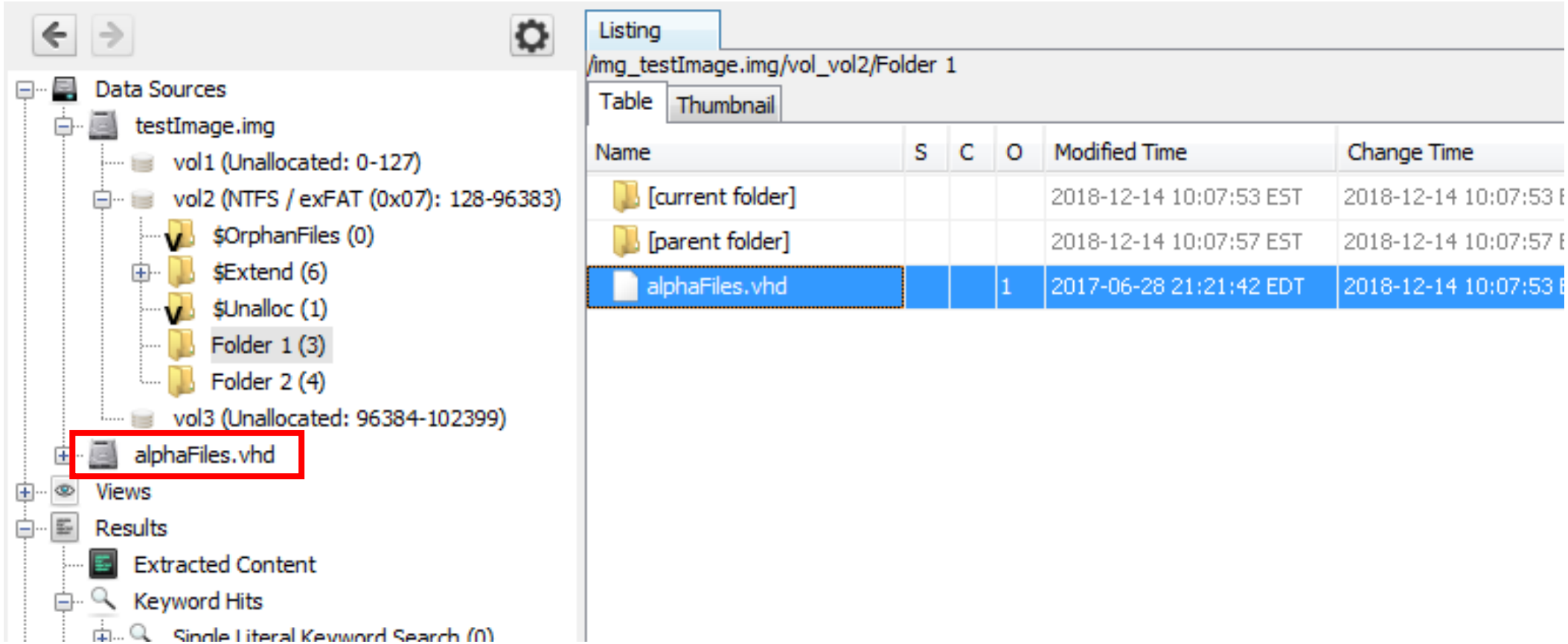
From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

Autopsy v4+ Module: Email Parser



From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

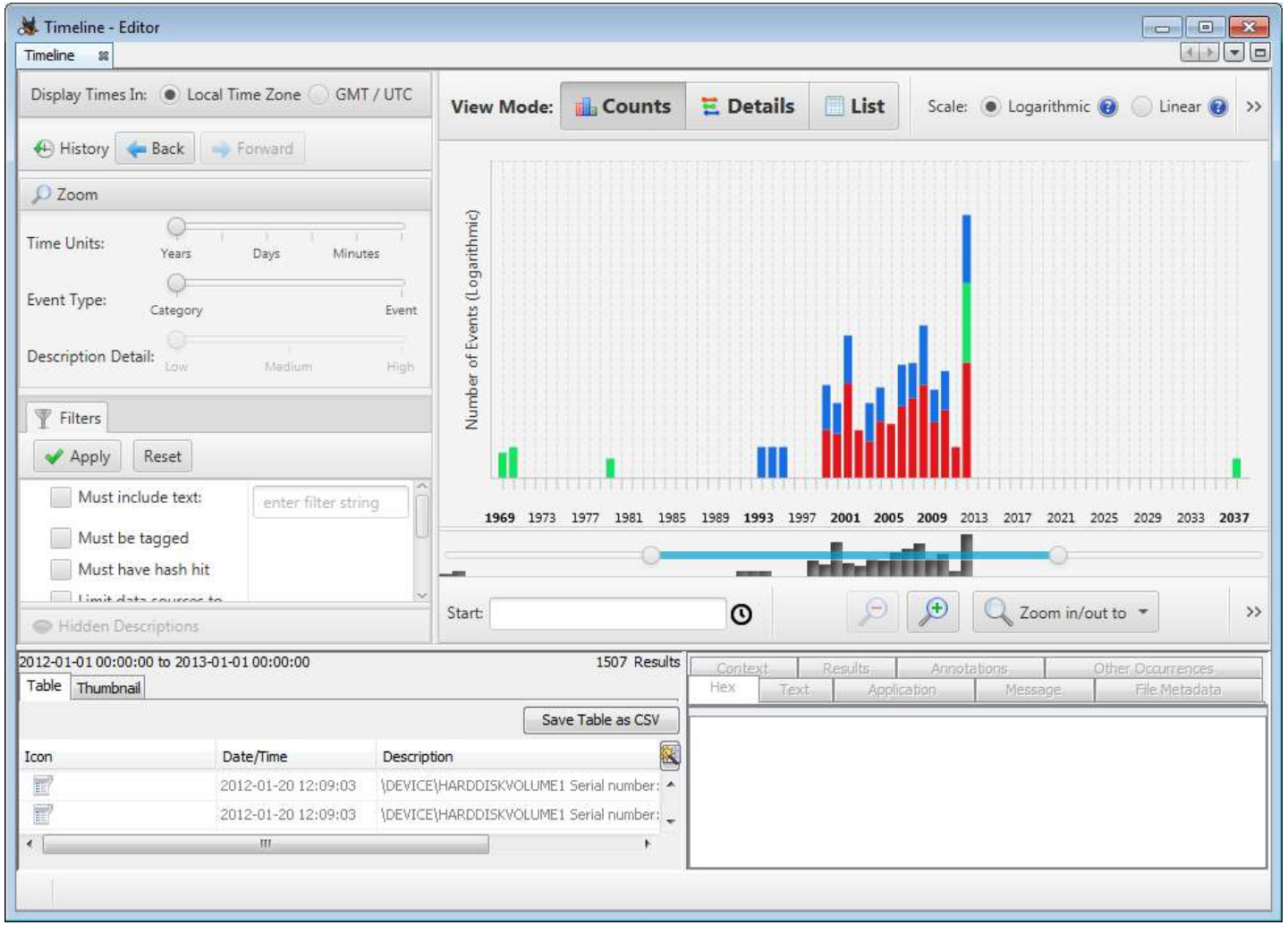
Autopsy v4+ Module: Virtual Machine Extractor



From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

Autopsy v4+ Module: Plaso

Extract **timestamps** for various file types, and create a **timeline diagram**

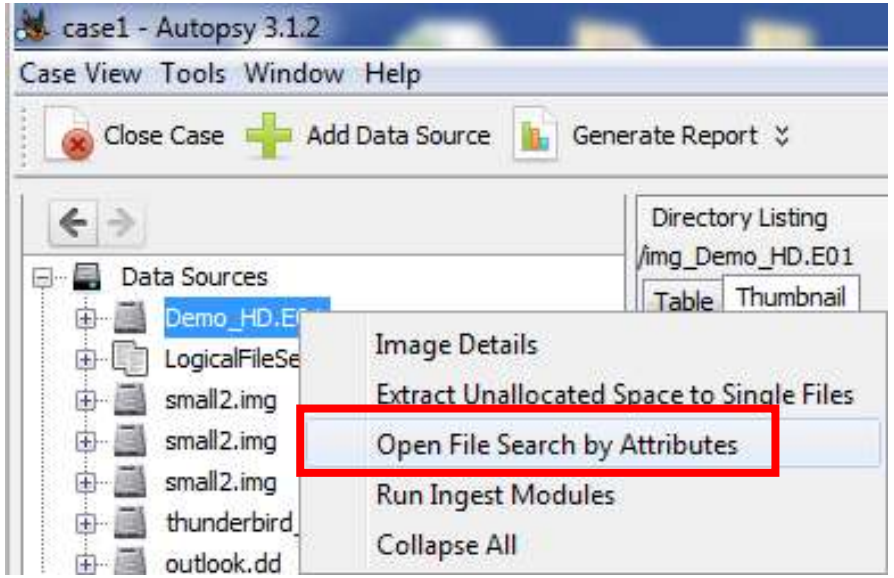


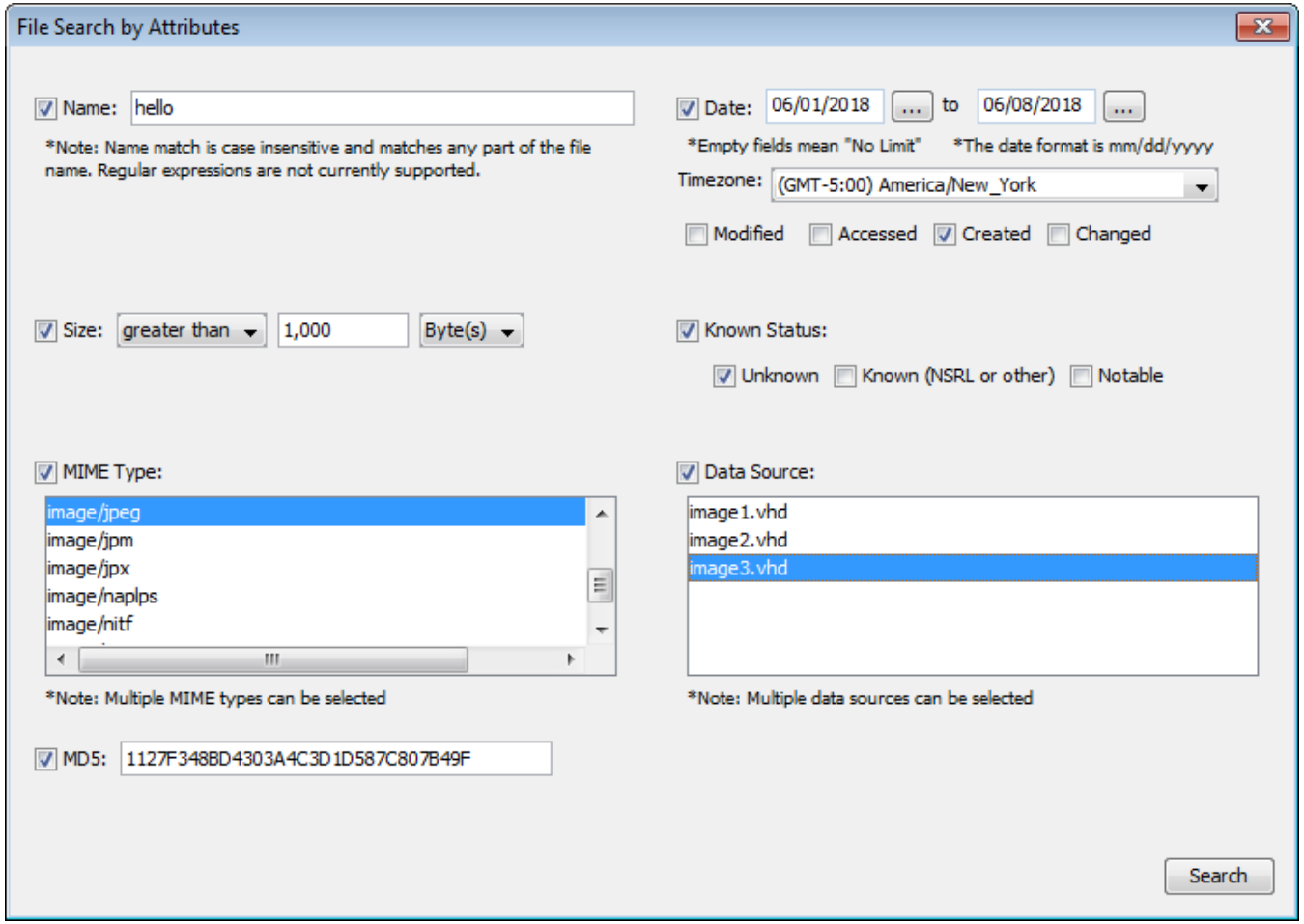
From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

Some Useful Ingest Modules

- Some **other modules** to consider:
  - ***Embedded File Extraction*:** opens ZIP, RAR, other archive formats
  - ***YARA Analyzer***: useful for malware analysis, but can be used to search for any type of files
  - ***Android Analyzer***: analyzes SQLite and other files from an Android device
  - ***Android Analyzer (aLEAPP)***: runs aLEAPP (https://github.com/abrignoni/aLEAPP), and converts the results into results that can be viewed in Autopsy
- **Third party** add-on modules:
  - Child_Exploitation_Hashsets, Chrome_Passwords, GoogleDrive, …
  - See: https://github.com/sleuthkit/autopsy_addon_modules/tree/master/IngestModules

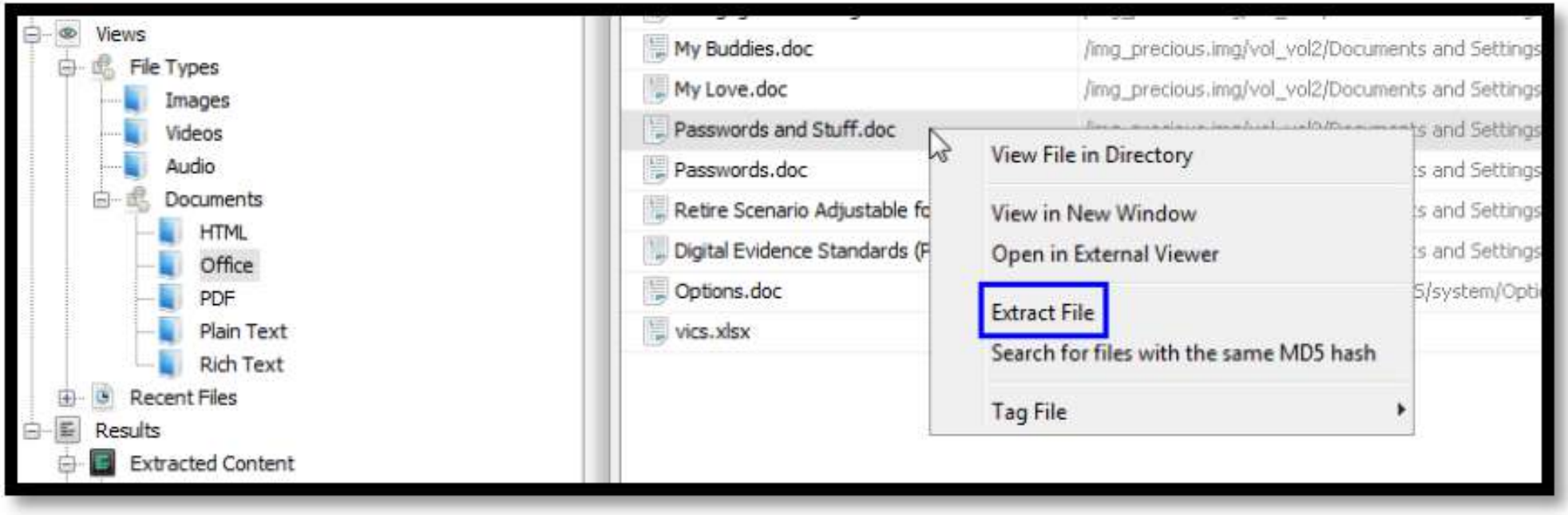
Autopsy v4+ Feature: File Searching





From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

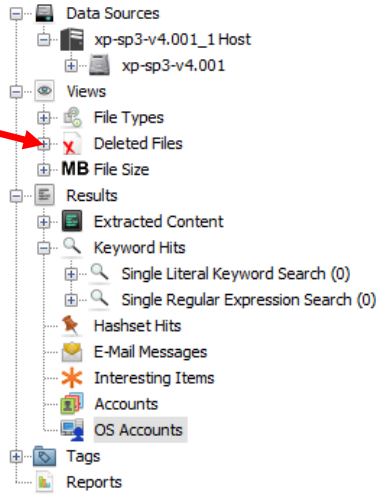
Autopsy v4+ Feature: File Extraction (See Lab 3)

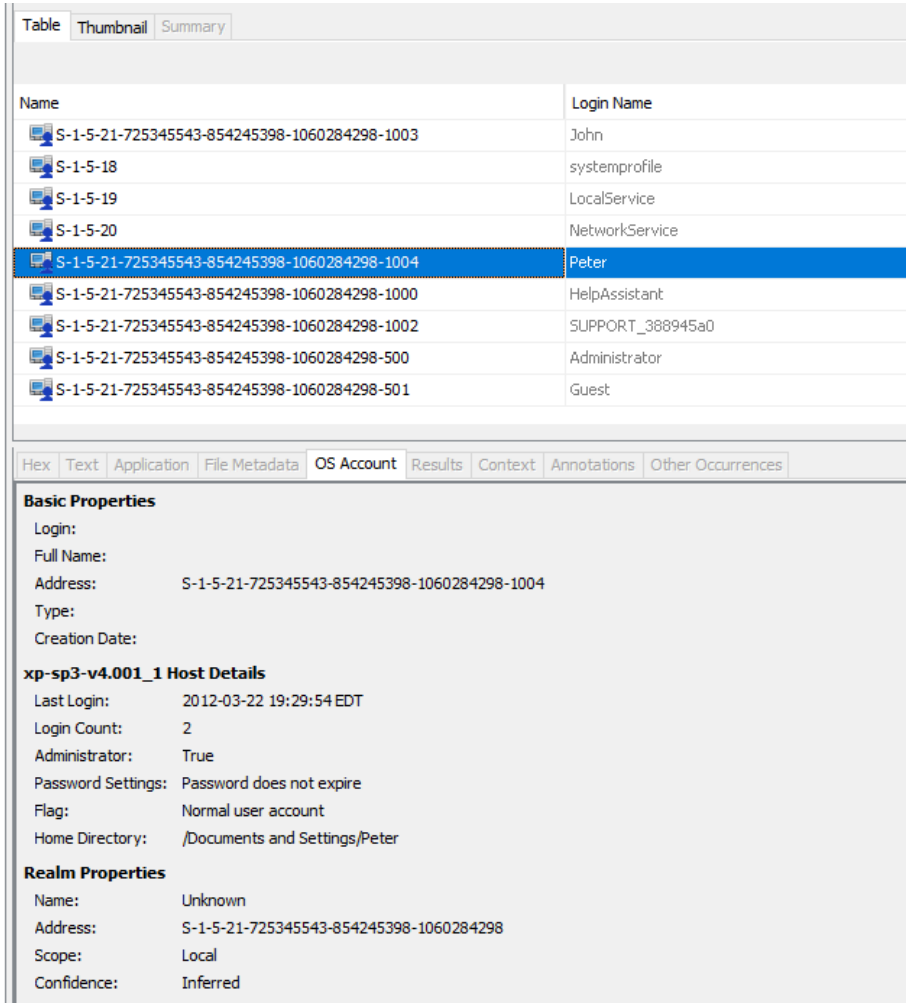


From: Julia Keffer, "*Autopsy Forensic Browser User Guide*", 2013

Autopsy v4+ Feature: File Carving

All deleted files

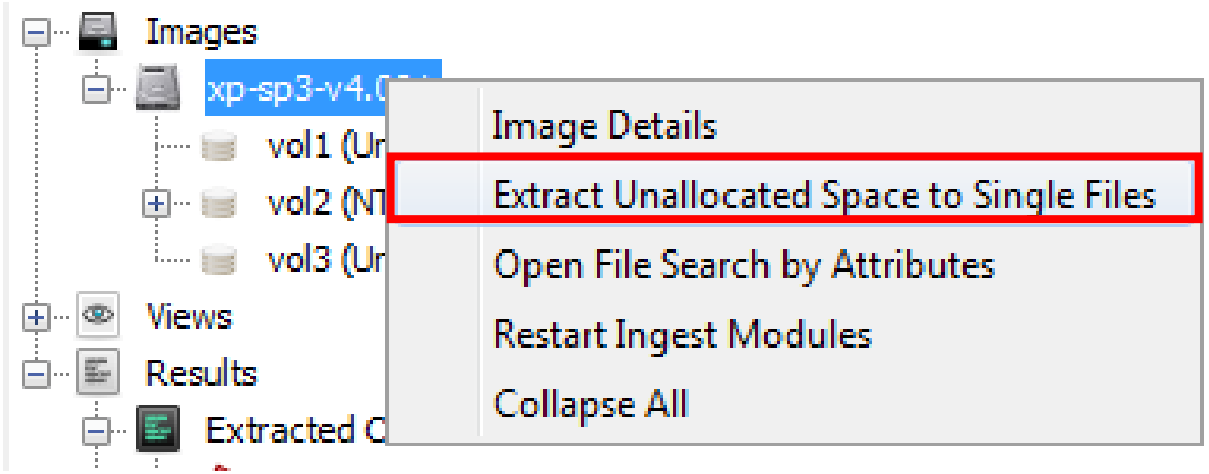




From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

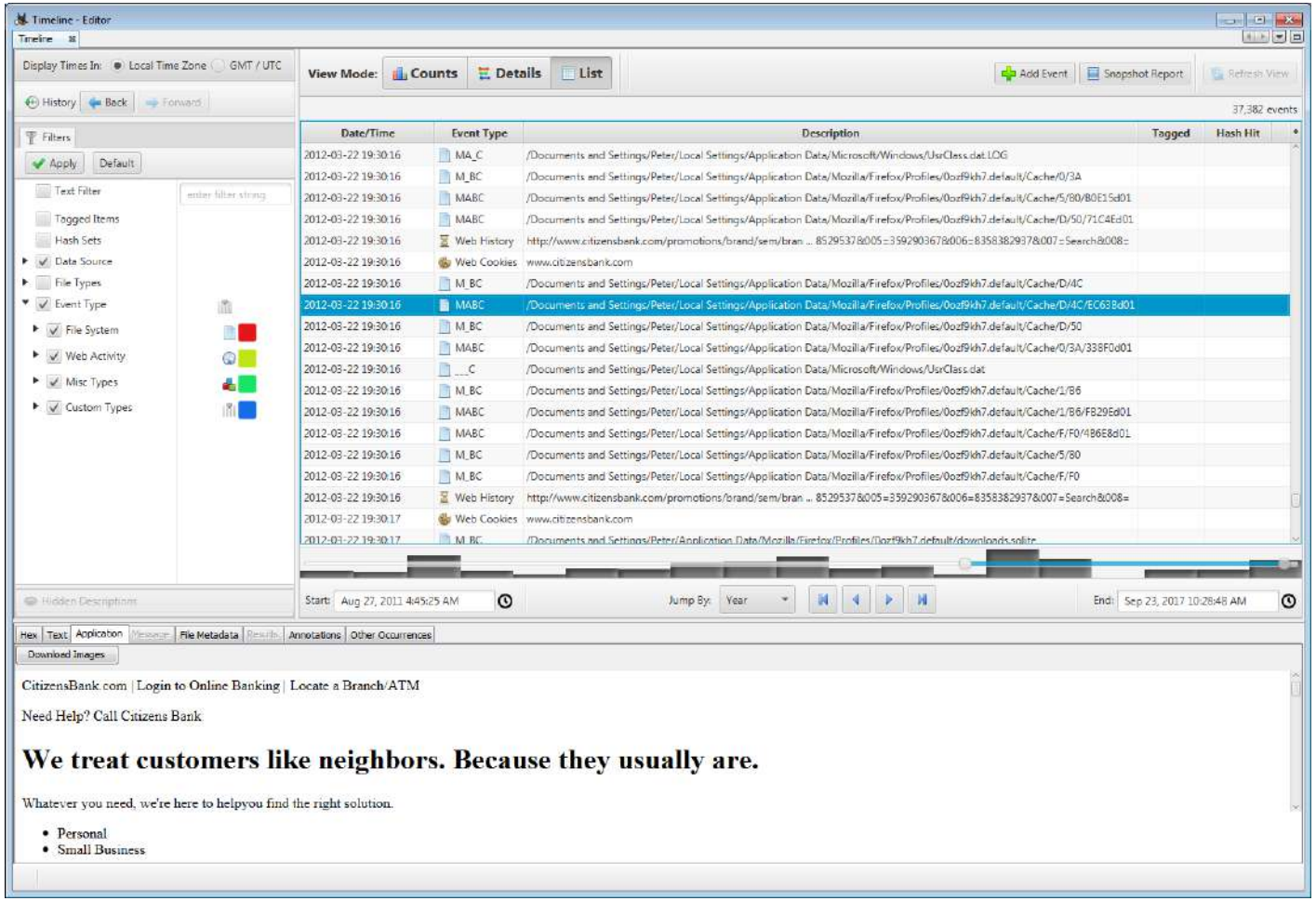
Autopsy v4+ Feature: File Carving

To extract **unallocated space** into a **single file**:



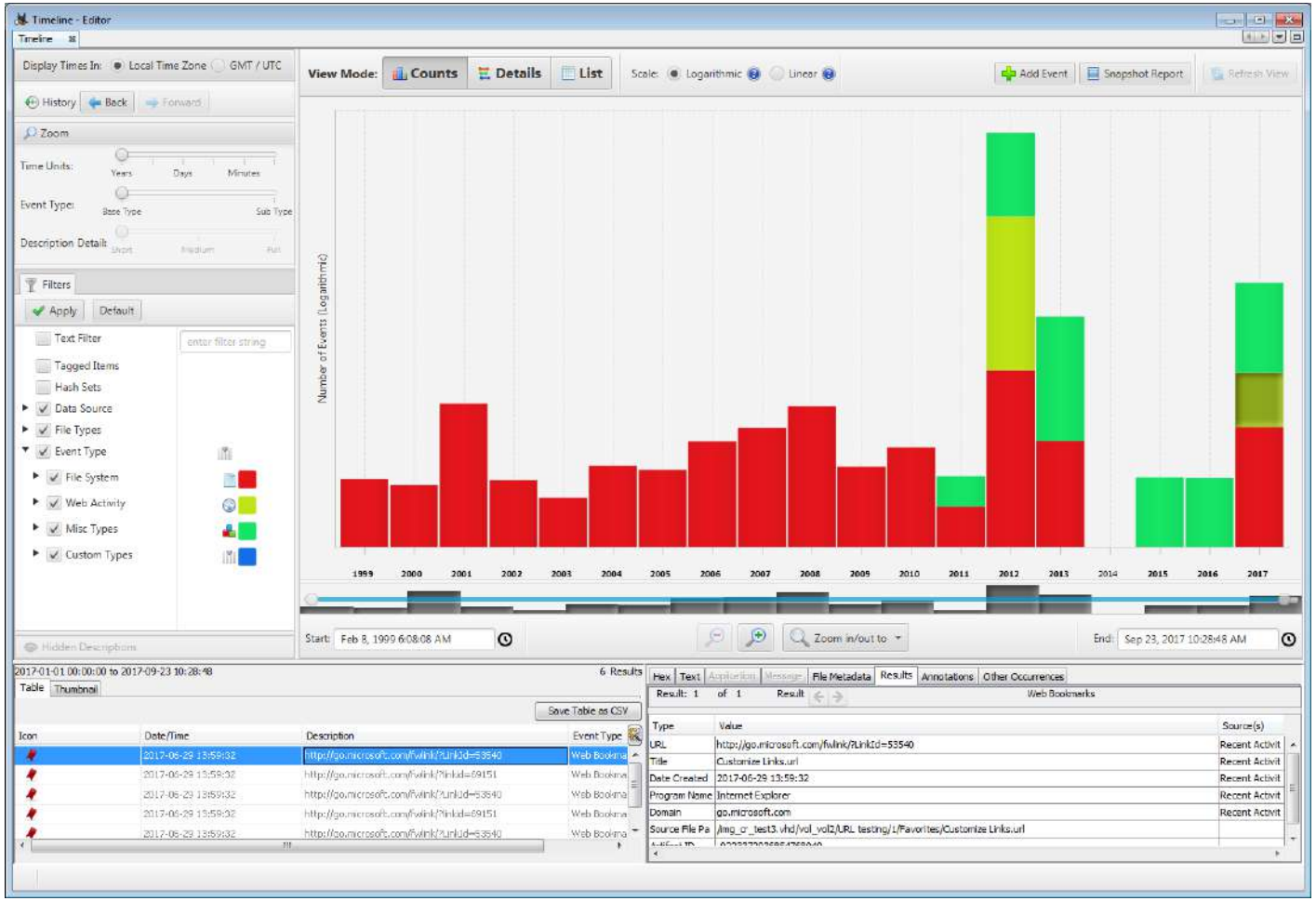
From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

Autopsy v4+ Feature: Timeline Analysis



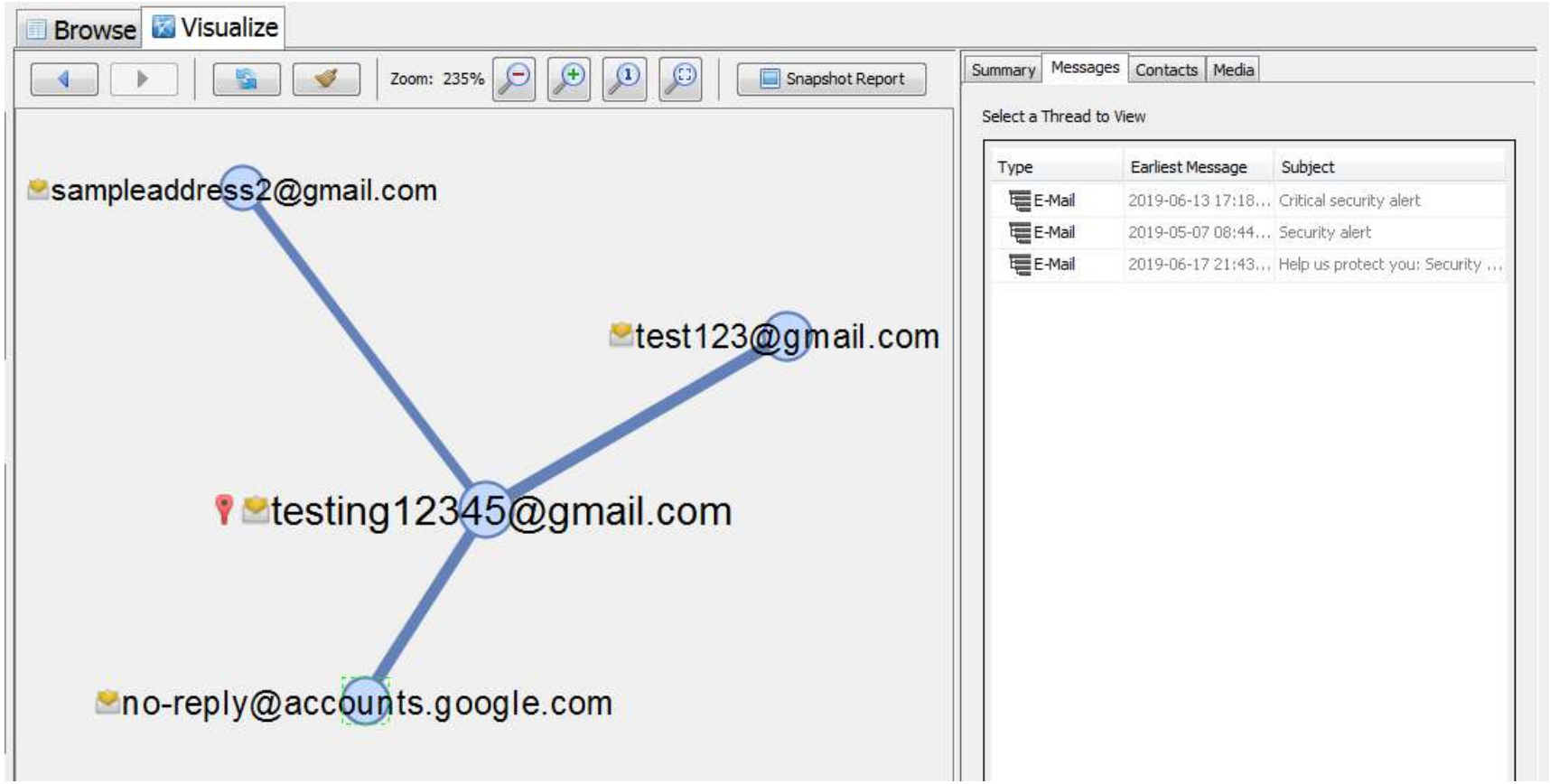
From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//timeline_page.html

Autopsy v4+ Feature: Timeline Analysis



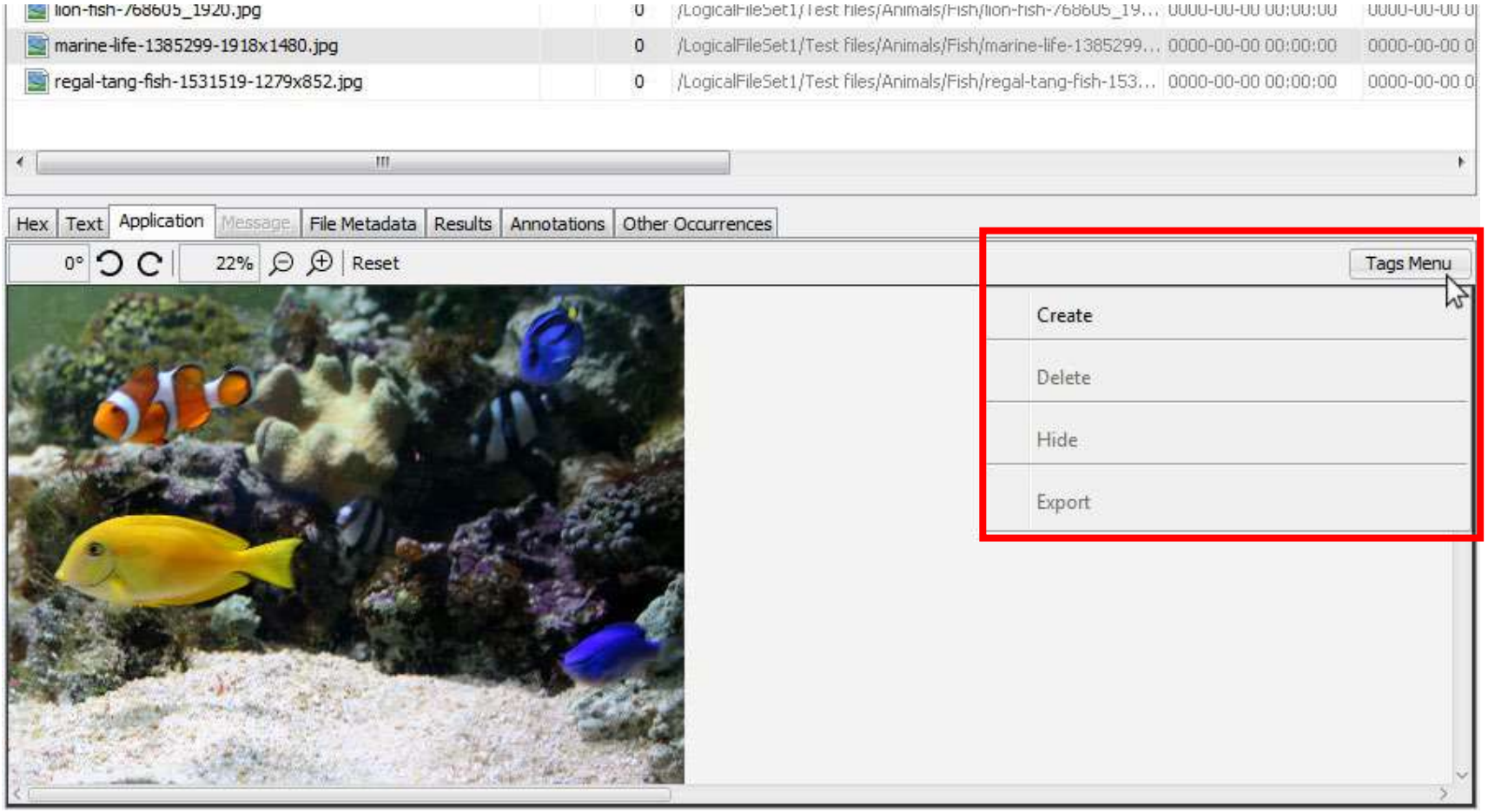
From: http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//timeline_page.html

Autopsy v4+ Feature: Communications Visualization Tool



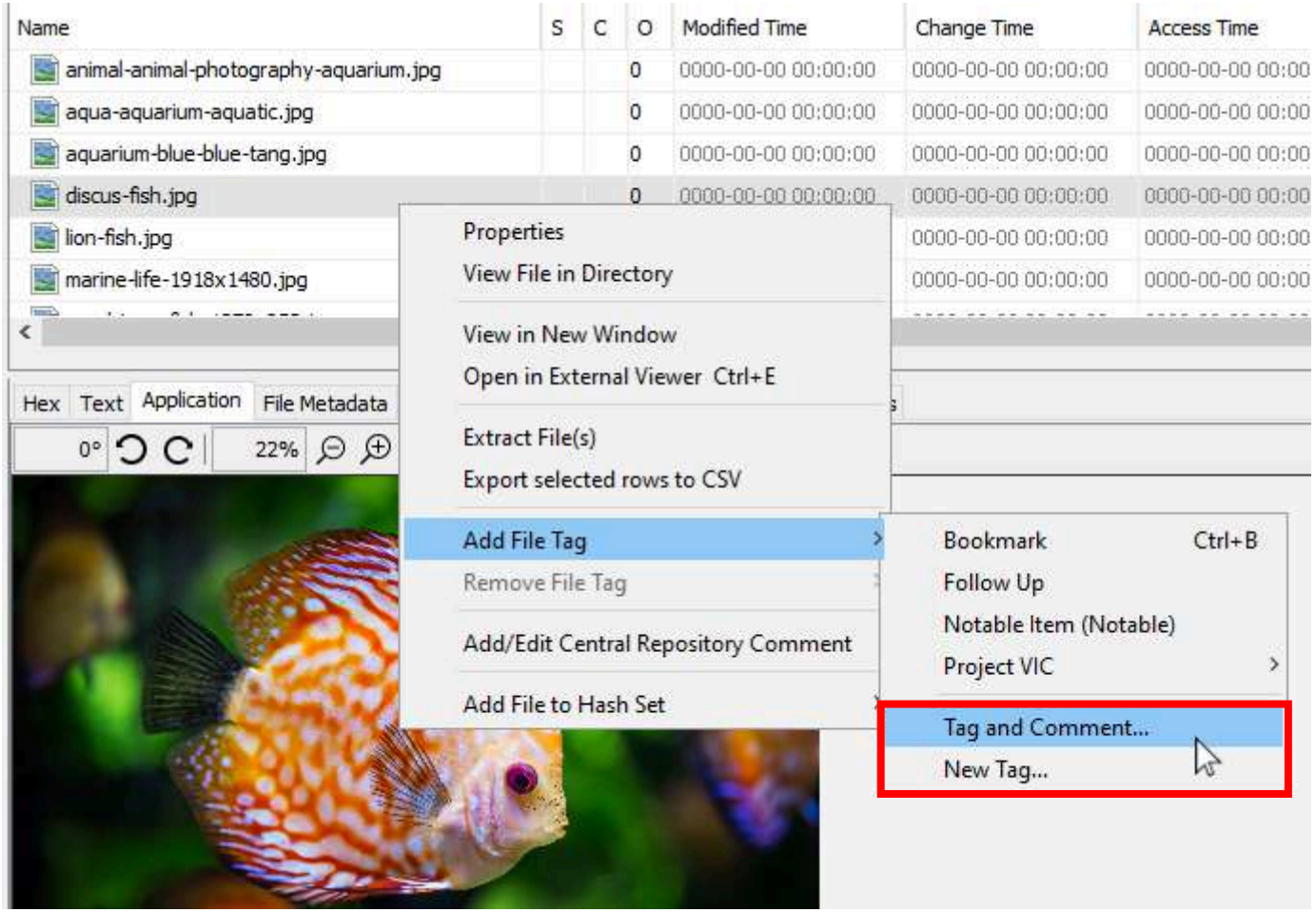
http://sleuthkit.org/autopsy/docs/user-docs/4.19.2/communications_page.html

Autopsy v4+ Feature: Tagging



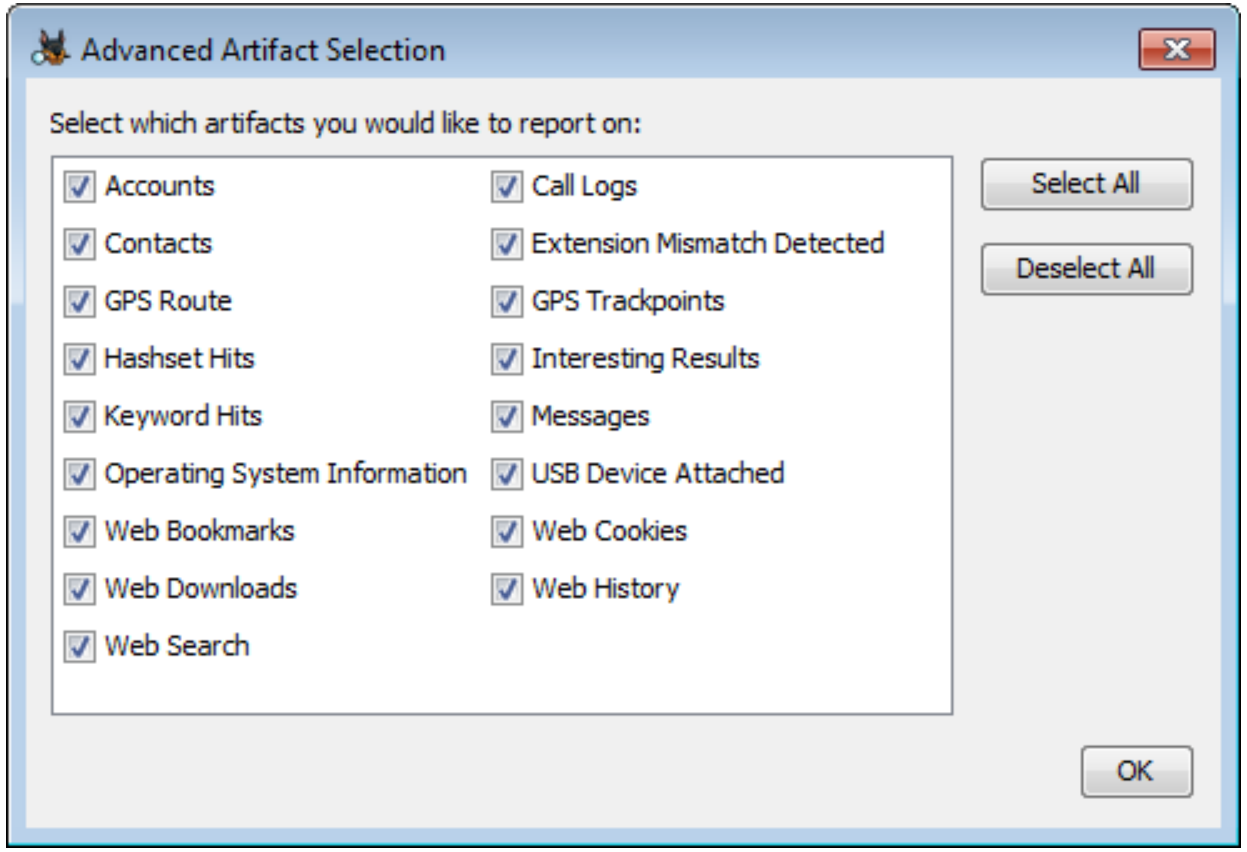
http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

Autopsy v4+ Feature: Tagging



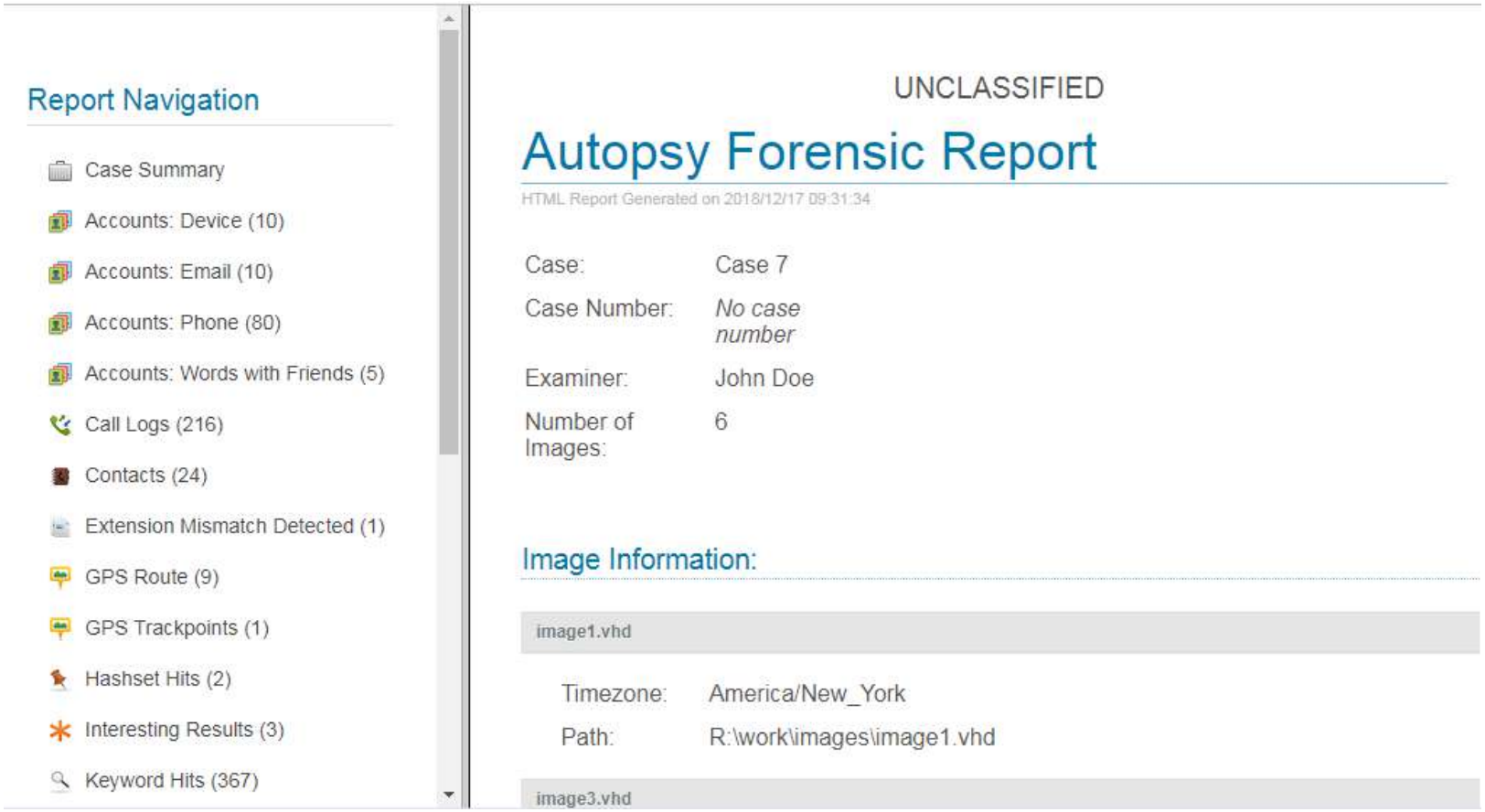
http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

Autopsy v4+ Feature: Reporting



http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

Autopsy v4+ Feature: Reporting



http://sleuthkit.org/autopsy/docs/user-docs/4.19.2//

Autopsy References

- **References**:
  - History (of features implemented): http://www.sleuthkit.org/autopsy/history.php
  - Autopsy User's Guide: https://wiki.sleuthkit.org/index.php?title=Autopsy_User%27s_Guide
- **Video** demo:
  - https://www.youtube.com/watch?v=Smy4mj293GE

Wait, Why *Not* Just Learn Autopsy in This Module?

- This is ***not*** about using tools, it is about **Digital Forensics** field
- By learning and understand the underlying **forensic concepts**, you **can**:
  - Understand deeper on **pieces of evidence** shown in Autopsy
  - Better use **Autopsy features** on the shown evidence
  - Better understand and use **other forensics suites** to complement/verify your Autopsy findings
  - Use other **specialized forensic tools/utilities** to find **more findings**: more manual control and greater flexibility
- In short: *deeper **understanding** of **forensic evidence & tasks** for more discovered findings!*

Wait, Why *Not* Just Learn Autopsy in This Module?

Furthermore…

- **Many** people could do some of what we do with a piece of forensic software suite
- But you **won't** stand the test of the law of evidence, nor your peers
- Understanding **more** and **deeper**: helps prepare you as a knowledgeable and well-rounded digital forensic investigator
- Similar approach taken in SoC modules: networking, OS, security, …

If You are Interested in Other Forensics Suites

- **Other** forensics suites:
  have **similar interface & features** with Autopsy's
- **AccessData FTK's** demo:
  - See e.g.: https://www.youtube.com/watch?v=k8QxpS8tM2I
- **Guidance Software Encase's** demo:
  - See e.g.:
    https://www.youtube.com/watch?v=QTV_vbCoYeU&list=PLO515IcRIbAxG2LLHBUVsxMnv3Du5JSEU
- **Magnet AXIOM's** demo:
  - See e.g.: https://www.youtube.com/watch?v=poKw8_NZ4ps

Lab 3 Exercises

- Task 1: Perform a **live acquisition** of a Windows machine using:
  - FTK Imager Lite
  - (Optional) FireEye's Memoryze
- Task 2: Inspect and analyze a **memory image** file using:
  - Volatility: various commands
  - FTK Imager: string searching
  - (Optional) Hex Editor (WinHex): string searching
- Task 3: Inspect and analyse a **disk image file** using Autopsy

Questions?
See you next week!

Don’t forget to submit your **Graded Lab Tasks 1**
via Canvas’ Assignment