Digital Forensics: Data Analysis

Thematic CERN School of Computing, 2026

Daniel Kouřil
kouril@cesnet.cz

MUNI

cesnet

Analysis of storage images

Analysis aims

- The aim is to analyze collected evidence
  - Imagine you have a large (GBs) image and need to do its analysis (e.g. access files and recover deleted data)
- Clarify the objective before starting the actual analysis
  - Recovering deleted data vs. secure evidence about malicious activities

Analysis environment

- The analysis does not depend on the system where we got data from
  - Artifacts related to MS Windows architecture can be analyzed on Linux-based environment and vice versa
- A Linux environment based on CLI will be used thorough the course
  - Many tools are common commands or are available from distribution packages
- Always keep the primary data intact and work only with its copies

```
root@exam:/home/investigation/case_2020-09-30-01/primary_data# ls -lh
total 16G
-rw-r--r-- 1 root root 25G Sep 30 15:34 image.dd
```

exam

```
root@exam# xxd /dev/sda
00000000: fab8 0000 8ed0 bc00 7c8b f450 0750 1ffb  ........|..P.P..
00000010: fcbf 0006 b900 01f3 a5ea 1e06 0000 bebe  ................
00000020: 0780 3c80 7402 cd18 5653 06bb 007c b901  ..<.t...VS...|..
00000030: 00ba 0000 b801 02cd 1307 5b5e b280 720b  ..........[^..r.
00000040: bfbc 7d81 3d55 5375 02b2 00bf eb06 8815  ..}.=USu........
00000050: 8a74 018b 4c02 8bee eb15 be9b 06ac 3c00  .t..L.........<.
00000060: 740b 56bb 0700 b40e cd10 5eeb f0eb febb  t.V.......^.....
00000070: 007c b801 02cd 1373 05be b306 ebdf bed2  .|.....s........
00000080: 06bf fe7d 813d 55aa 75d3 bf24 7cbe eb06  ...}.=U.u..$|...
00000090: 8a04 8805 8bf5 ea00 7c00 0049 6e76 616c  ........|..Inval
000000a0: 6964 2070 6172 7469 7469 6f6e 2074 6162  id partition tab
000000b0: 6c65 0045 7272 6f72 206c 6f61 6469 6e67  le.Error loading
000000c0: 206f 7065 7261 7469 6e67 2073 7973 7465   operating syste
000000d0: 6d00 4d69 7373 696e 6720 6f70 6572 6174  m.Missing operat
000000e0: 5632 2e30 7379 7374 656d 0000 0000 0000  V2.0system......
000000f0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000100: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000110: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000120: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000130: 0000 0000 0000 0000 0000 0000 0000 0000  ................
                                                          1,1           Top
```

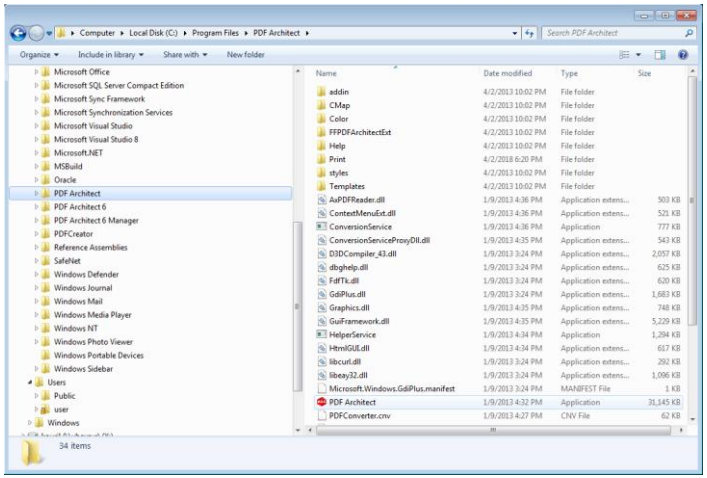
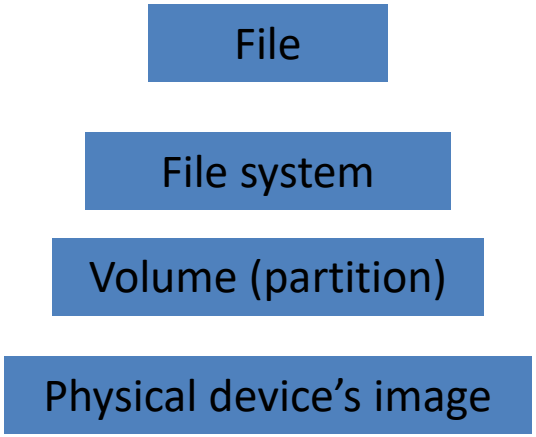


Image analysis

- Image is a sequence of bytes (just a file)
  - Internal structure needs to be established



- Some objects may be embedded
  - Files containing other images (VM disks)

File system

File system

- Organization of data on storage
- Data kept in files
  - File content
  - File metadata
- File systems differ from forensics view
  - Different features
  - Different support of tools

File Content

- File is a logical sequence of bytes
  - The type is determined by the content, not by name, location or extension
- File analysis is dictated by the objectives
  - User data (data content)
  - System files (logs, configurations, installed SW)
  - Executables

File metadata

- Metadata information
  - Owner identifier
  - Group identifier
  - Permissions / ACL
  - Addresses of data blocks (content)
  - Important timestamps
- No need to access content
  - Smaller space needed
  - Less privacy issues

Analysis using timelines

Timelines

- Common analysis technique important for many objectives
- Timeline provides a simple overview of events that happened on the system
- Can be constructed from any data where timestamps is recorded
  - Logs, events (*users logins*)
  - Application data (*mail/document manipulations*)
  - File metadata (*file utilizations*)

File timestamps

- Common types of timestamps (POSIX)
  - m-time (modification time) – the last time the content was changed
  - a-time (access time) – the time of last access (content)
  - c-time (change time) – the last time metadata was changed
- Additional timestamps on some file systems
  - d-time (deletion time)
  - b-time (creation time) (sometimes cr-time)
- Timestamps only refer to the very last action performed

```
File:  /var/www/files/shell.php
Access:  2017-06-07 07:00:05 +0200
Modify:  2017-06-03 17:59:28 +0200
Change:  2017-06-03 18:01:40 +0200
```

```
File:  /var/www/archive.php
Access:  2017-06-01 11:01:07 +0200
Modify:  2017-06-01 10:59:54 +0200
Change:  2017-06-01 10:59:54 +0200
```

```
File:  /var/www/upload.php
Access:  2017-06-03 17:45:50 +0200
Modify:  2017-06-01 10:59:54 +0200
Change:  2017-06-03 17:40:10 +0200
```

```
File:  /var/www/gallery.php
Access:  2017-06-03 17:59:28 +0200
Modify:  2017-06-01 10:59:55 +0200
Change:  2017-06-01 10:59:55 +0200
```

A malicious PHP file (backdoor) **shell.php** was found on a web server. Examine the time stamps of **archive.php, upload.php, gallery.php** and determine which file was likely used to store the malicious payload and select timestamp when the backdoor was used for the last time.

1994-05-26T00:00:00.000

m type a type c type b type

Interactive List View Total: 27862

Timestamp	Size	Type	Mode	UID	GID	File Name
2016-05-23 07:35:01	577009	...b	r/rrw-r--r	0	0	/var/backups/dpkg.status.6.gz
2016-05-23 07:35:01	222	...b	r/rrw-r--r	0	0	/var/backups/dpkg.statoverride.6.gz
2016-05-24 07:35:01	577009	...b	r/rrw-r--r	0	0	/var/backups/dpkg.status.5.gz
2016-05-24 07:35:02	222	...b	r/rrw-r--r	0	0	/var/backups/dpkg.statoverride.5.gz
2016-05-25 00:41:43	0	mac.	-rw-r--r--	0	0	/home/martin/.ssh/.s.c
2016-05-25 00:41:43	0	mac.	-rw-------	1	1	/home/martin/.ssh/authorized_keys
2016-05-25 00:43:22	0	m.c.	drwx------	1	1	/home/martin/.ssh
2016-05-25 02:41:43	6992	m...	r/rrwsr-xr	0	100	/var/lib/.s
2016-05-25 02:43:12	12400	...b	r/rrwxr-xr	0	0	/lib/libselinux.so.4
2016-05-25 02:43:22	21	m.cb	r/rrw-r--r	0	0	/etc/ld.so.preload
2016-05-25 02:43:22	6992	..cb	r/rrwsr-xr	0	100	/var/lib/.s
2016-05-25 07:35:01	577009	...b	r/rrw-r--r	0	0	/var/backups/dpkg.status.4.gz
2016-05-25 07:35:01	222	...b	r/rrw-r--r	0	0	/var/backups/dpkg.statoverride.4.gz
2016-05-25 19:20:02	0	m.c.	-rw-r--r--	1	1	/home/roberto/.ssh/authorized_keys
2016-05-25 19:27:49	0	m.c.	drwx------	1	1	/home/roberto/.ssh
2016-05-25 21:24:07	35	..cb	l/lrwxrwxr	0	0	/usr/lib/x86_64-linux-gnu/libsepol.so - /lib/x86_64-linux-gnu/libsepol.so.1
2016-05-25 21:24:07	37	..cb	l/lrwxrwxr	0	0	/usr/lib/x86_64-linux-gnu/libselinux.so - /lib/x86_64-linux-gnu/libselinux.so.1
2016-05-25 21:25:31	12400	..c.	r/rrwxr-xr	0	0	/lib/libselinux.so.4
2016-05-25 22:08:38	1239	.acb	r/rrwxr-xr	1000	1000	/var/tmp/.../autoasnfs/m/.git/hooks/prepare-commit-msg.sample
2016-05-25 22:08:38	896	.acb	r/rrwxr-xr	1000	1000	/var/tmp/.../autoasnfs/m/.git/hooks/commit-msg.sample
2016-05-25 22:08:38	1234	.acb	r/rrw-r--r	1000	1000	/var/tmp/.../autoasnfs/m/vs10/masscan.sln
2016-05-25 22:08:38	160	.acb	r/rrwxr-xr	1000	1000	/var/tmp/.../autoasnfs/m/.git/hooks/post-commit.sample
2016-05-25 22:08:38	548	.acb	r/rrwxr-xr	1000	1000	/var/tmp/.../autoasnfs/m/.git/hooks/post-receive.sample
2016-05-25 22:08:38	4951	.acb	r/rrwxr-xr	1000	1000	/var/tmp/.../autoasnfs/m/.git/hooks/pre-rebase.sample
2016-05-25 22:08:38	3611	.acb	r/rrwxr-xr	1000	1000	/var/tmp/.../autoasnfs/m/.git/hooks/update.sample
2016-05-25 22:08:38	398	.acb	r/rrwxr-xr	1000	1000	/var/tmp/.../autoasnfs/m/.git/hooks/pre-applypatch.sample
2016-05-25 22:08:38	189	.acb	r/rrwxr-xr	1000	1000	/var/tmp/.../autoasnfs/m/.git/hooks/post-update.sample
2016-05-25 22:08:38	1578	.acb	r/rrwxr-xr	1000	1000	/var/tmp/.../autoasnfs/m/.git/hooks/pre-commit.sample

Working with file timestamps

- Executing a file changes its Atime
  - The precision of a-time depends on configuration
- m-time and a-time can be easily changed by file owner
  - happens when copying/moving files, or deploying software from packages
- c-time can’t be changed easily
- Pay attention to time zones and granularity
  - FAT uses system time, NTFS uses UTC
  - Precision is among days (FAT), secs (ext3), and nanosecs (ext4)

Obtaining metadata

- `fls` and `mactime` commands (only for supported FS)
- Simple ’find’ command (recursive walk through the filesystem)
  - `find / -xdev -print0 |`
    `xargs -0 stat -c "%Y %X %Z %A %U %G %n"`
  - Leif Nixon’s timeline-decorator.py to format
- Be prepared for a lot of data (hundreds thousands of records)

Live Analysis

Live analysis

- Access to volatile information kept by OS
- Some crucial aspects to consider
  - Reliability of the collected data
  - Modifications to the system done during the process
- The goal is to capture information for off-line analysis, not doing analysis on the host

Areas of Live Analysis

- Obtaining volatile information available from kernel and applications
- Obtaining content of memory
  - A complete host memory or memory of selected processes
- Recovering data that would be lost
  - Deleted, still open files on Linux

Obtaining OS information

- Network status
  - Open/established connections, listening/bound processes
  - allocated IP (4/6) addresses
  - VPN connections, routing tables, neighbors
  - Firewall state
- Information on the system setup
  - Available devices
  - Mounted file systems, mapped drives, shares
    - Data and “auxiliary” (RAIDs,…)

Obtaining OS information

- Information on processes
  - List of active processes and their attributes
    - The full path of the program, command line parameters, running time
  - List of files open by processes
  - Information on inter-process communication (shared memory, queues)
- Information on the OS
  - Loaded kernel modules/drivers, OS messages (dmesg), running OS version
  - Configured time-zone, uptime
  - Clipboards contents
- Auxiliary info (partially available also offline)
  - Logged users

Extracting information on processes

- Processes may contain important information
  - Resources used (e.g. network connections, files being processed, IPC)
  - Memory contains pristine information, including sensitive data
    - Encryption keys, passwords

Linux specifics

Getting process information

- Process may have multiple file-descriptors opened
  - Used executable, libraries
  - Particular files on file system
  - Network sockets
- Information on processes can be accessed using standard system commands
  - `lsof -p PID`

/proc filesystem

- Linux kernel exposes some internal structures in the /proc virtual file
- System commands mostly use data from /proc
- /proc can be useful to access data that is not available through commands (or spot anomalies)

```
forensics# ps axuf
kouril    926258  0.0  0.0   7884  4768 pts/19   Ss   14:06   0:00          |   \_ bash
kouril    930511  1.8  0.8 2656792 293272 pts/19 Sl+  15:04   0:02          |   |   \_ /usr/lib/thunderbird/thunderbird
kouril    926309  0.0  0.0   7884  4832 pts/20   Ss   14:07   0:00          |   \_ bash
kouril    930802  0.0  0.0   2608   612 pts/22   Ss+  15:06   0:00          |   |       \_ sh -c
kouril    930803  0.0  0.0   7024  3688 pts/22   S+   15:06   0:00          |   |           \_ bash
kouril    930806  5.3  0.0  12836  7060 pts/22   S+   15:06   0:00          |   |               \_ wget -q -i -
kouril    930814  0.0  0.0   9744  4192 pts/22   R+   15:06   0:00          |   |               \_ ps axuf
kouril    929030  0.0  0.0   7368  4056 pts/21   Ss   14:55   0:00          |   \_ bash
kouril    929048  0.0  0.0  21756  9700 pts/21   S+   14:55   0:00          |       \_ vi README.txt
forensics# lsof -p 930806
COMMAND    PID   USER   FD   TYPE   DEVICE  SIZE/OFF     NODE NAME
wget    930806 kouril  cwd    DIR    253,0      4096   131074 /tmp
wget    930806 kouril  rtd    DIR    253,0      4096        2 /
wget    930806 kouril  txt    REG    253,0    548568  1313290 /usr/bin/wget
wget    930806 kouril  mem    REG    253,0     31176  1327174 /usr/lib/x86_64-linux-gnu/libnss_dns-2.31.so
wget    930806 kouril  mem    REG    253,0    101320  1327181 /usr/lib/x86_64-linux-gnu/libresolv-2.31.so
wget    930806 kouril  mem    REG    253,0     18504  1325883 /usr/lib/x86_64-linux-gnu/libnss_mdns4_minimal.so.2
wget    930806 kouril  mem    REG    253,0   3035952  1311418 /usr/lib/locale/locale-archive
wget    930806 kouril  mem    REG    253,0     18816  1327168 /usr/lib/x86_64-linux-gnu/libdl-2.31.so
wget    930806 kouril  mem    REG    253,0   1575112  1317178 /usr/lib/x86_64-linux-gnu/libunistring.so.2.1.0
wget    930806 kouril  mem    REG    253,0    157224  1327180 /usr/lib/x86_64-linux-gnu/libpthread-2.31.so
wget    930806 kouril  mem    REG    253,0   2029224  1327167 /usr/lib/x86_64-linux-gnu/libc-2.31.so
wget    930806 kouril  mem    REG    253,0     71672  1317135 /usr/lib/x86_64-linux-gnu/libpsl.so.5.3.2
wget    930806 kouril  mem    REG    253,0    108936  1317207 /usr/lib/x86_64-linux-gnu/libz.so.1.2.11
wget    930806 kouril  mem    REG    253,0   2954080  1314738 /usr/lib/x86_64-linux-gnu/libcrypto.so.1.1
wget    930806 kouril  mem    REG    253,0    598104  1327133 /usr/lib/x86_64-linux-gnu/libssl.so.1.1
wget    930806 kouril  mem    REG    253,0    129096  1317048 /usr/lib/x86_64-linux-gnu/libidn2.so.0.3.6
wget    930806 kouril  mem    REG    253,0     30936  1317194 /usr/lib/x86_64-linux-gnu/libuuid.so.1.3.0
wget    930806 kouril  mem    REG    253,0    584392  1317121 /usr/lib/x86_64-linux-gnu/libpcre2-8.so.0.9.0
wget    930806 kouril  mem    REG    253,0     51832  1327175 /usr/lib/x86_64-linux-gnu/libnss_files-2.31.so
wget    930806 kouril  mem    REG    253,0     27002  1450838 /usr/lib/x86_64-linux-gnu/gconv/gconv-modules.cache
wget    930806 kouril  mem    REG    253,0    191472  1327086 /usr/lib/x86_64-linux-gnu/ld-2.31.so
wget    930806 kouril    0r  FIFO     0,13       0t0 17501707 pipe
wget    930806 kouril    1u   CHR   136,22       0t2       25 /dev/pts/22
wget    930806 kouril    2u   CHR   136,22       0t2       25 /dev/pts/22
wget    930806 kouril    3u  IPv6 17518067       0t0      TCP odorn:57766->https-services.banjo.canonical.com:https (ESTABLISHED)
wget    930806 kouril    4w   REG    253,0 148881408   131663 /tmp/ubuntu-20.04.3-desktop-amd64.iso
forensics# ls -l /tmp/ubuntu-20.04.3-desktop-amd64.iso
-rw-r--r-- 1 kouril kouril 378060800 Nov 23 15:06 /tmp/ubuntu-20.04.3-desktop-amd64.iso
```

Deleted files

- Deleted files are available until they are closed
  - If a file is open by a process, it's removed from the filesystem but its content can still be still accessed
- "symbolic links" in `/proc` can be used to recover the data
  - `cp /proc/$PID/exe /tmp/exe`
  - The process must be alive (even stopped)
- Holds for both executable and open files (see the fd directory)

```
forensics# ps axuf
          ..........
kouril      9969  0.0  0.0   2576   944 pts/4    Ss+  20:15   0:00          |   |       \_ sh -c
kouril      9970  0.0  0.0   7064  3484 pts/4    S+   20:15   0:00          |   |           \_ bash
kouril      9974  3.5  0.0  16940  8880 pts/4    S+   20:15   0:00          |   |               \_ ./suspicious -q -i -
kouril      9992  0.0  0.0  11348  4968 pts/4    R+   20:15   0:00          |   |               \_ ps axuf
kouril      2911  0.0  0.0   7328  4028 pts/2    Ss+  19:47   0:00          |   \_ bash
kouril      3009  0.0  0.0   7328  3916 pts/5    Ss+  19:47   0:00          |   \_ bash
          ..........
forensics# lsof -p 9974
COMMAND    PID   USER   FD   TYPE DEVICE SIZE/OFF     NODE NAME
suspiciou 9974 kouril  cwd    DIR  253,1    61440 92274689 /tmp
suspiciou 9974 kouril  rtd    DIR  253,1     4096        2 /
suspiciou 9974 kouril  txt    REG  253,1   540016 92274792 /tmp/suspicious (deleted)
suspiciou 9974 kouril  mem    REG  253,1  3048928 34611608 /usr/lib/locale/locale-archive
suspiciou 9974 kouril  mem    REG  253,1   529216 34603517 /usr/lib/x86_64-linux-gnu/libgmp.so.10.4.1
suspiciou 9974 kouril  mem    REG  253,1  1261376 34603561 /usr/lib/x86_64-linux-gnu/libp11-kit.so.0.3.0
suspiciou 9974 kouril  mem    REG  253,1  1792040 34605285 /usr/lib/x86_64-linux-gnu/libunistring.so.2.2.0
suspiciou 9974 kouril  mem    REG  253,1  1922136 34619505 /usr/lib/x86_64-linux-gnu/libc.so.6
suspiciou 9974 kouril  mem    REG  253,1  2197240 34604810 /usr/lib/x86_64-linux-gnu/libgnutls.so.30.34.3
suspiciou 9974 kouril  mem    REG  253,1    43480 34603555 /usr/lib/x86_64-linux-gnu/libffi.so.8.1.2
suspiciou 9974 kouril  mem    REG  253,1   292856 34603544 /usr/lib/x86_64-linux-gnu/libhogweed.so.6.6
suspiciou 9974 kouril  mem    REG  253,1    83968 34605341 /usr/lib/x86_64-linux-gnu/libtasn1.so.6.6.3
suspiciou 9974 kouril  mem    REG  253,1    75616 34628718 /usr/lib/x86_64-linux-gnu/libpsl.so.5.3.4
suspiciou 9974 kouril  mem    REG  253,1   121280 34605309 /usr/lib/x86_64-linux-gnu/libz.so.1.2.13
suspiciou 9974 kouril  mem    REG  253,1   317544 34603527 /usr/lib/x86_64-linux-gnu/libnettle.so.8.6
suspiciou 9974 kouril  mem    REG  253,1   198776 34603021 /usr/lib/x86_64-linux-gnu/libidn2.so.0.3.8
suspiciou 9974 kouril  mem    REG  253,1    34872 34606742 /usr/lib/x86_64-linux-gnu/libuuid.so.1.3.0
suspiciou 9974 kouril  mem    REG  253,1   629384 34603678 /usr/lib/x86_64-linux-gnu/libpcre2-8.so.0.11.2
suspiciou 9974 kouril  mem    REG  253,1    18504 34641479 /usr/lib/x86_64-linux-gnu/libnss_mdns4_minimal.so.2
suspiciou 9974 kouril  mem    REG  253,1    27028 34619770 /usr/lib/x86_64-linux-gnu/gconv/gconv-modules.cache
suspiciou 9974 kouril  mem    REG  253,1   210968 34619502 /usr/lib/x86_64-linux-gnu/ld-linux-x86-64.so.2
suspiciou 9974 kouril    0r  FIFO   0,13      0t0    97668 pipe
suspiciou 9974 kouril    1u   CHR  136,4      0t0        7 /dev/pts/4
suspiciou 9974 kouril    2u   CHR  136,4      0t0        7 /dev/pts/4
suspiciou 9974 kouril    3w   REG  253,1 12976128 92274794 /tmp/ubuntu-20.04.3-desktop-amd64.iso
suspiciou 9974 kouril    4u  IPv4  99634      0t0      TCP 10.0.0.105:48080->https-services.banjo.canonical.com:https (ESTABLISHED)
forensics# ls -l /tmp/suspicious
ls: cannot access '/tmp/suspicious': No such file or directory
forensics# ls -l /proc/9974/exe
lrwxrwxrwx 1 kouril kouril 0 Mar 23 20:15 /proc/9974/exe -> '/tmp/suspicious (deleted)'
forensics# sha1sum /proc/9974/exe /bin/wget
45f34cf94fd48dfd81765f8fdb0bfd84a14c72f8  /proc/9974/exe
45f34cf94fd48dfd81765f8fdb0bfd84a14c72f8  /bin/wget
forensics#
```

Network

- Getting information on network status
  - Three different ways:
    - netstat -tnp
    - ss -tnp
    - Check the /proc virtual filesystem
  - All should yield the same information (in different formats though)
    - If not, some of the commands might be modified
- `tcpdump` might be handy to check live traffic

Dumping process memory

- `gcore -o dump`
  - Part of the GDB package
  - Some (soft) errors might be triggered
- Outputs an ELF file containing the process memory

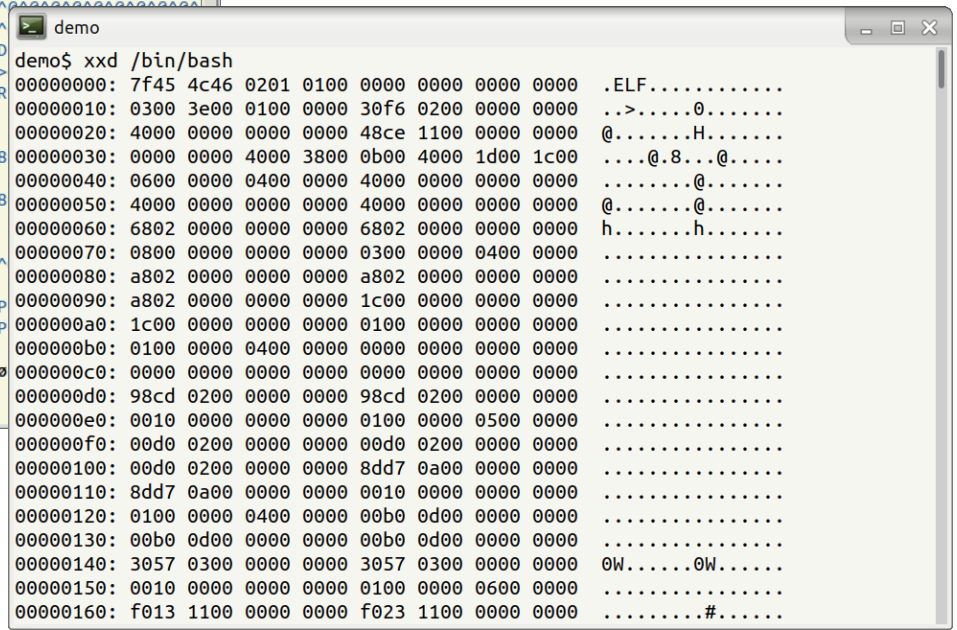
Analysis of executable files

Executable files

- Scripts
  - List of commands, script constructs
  - Easily readable by human (if not obfuscated)
- Binary executables
  - Machine code (produced by compiler)
  - byte code (Java)
  - ELF, PE formats
- Libraries
  - Static / dynamic
  - Library functions, variables (internal / exported)
  - Export interface (ABI/API)

Binary executable files

- System specific formats
- Dynamic vs. static analysis



ELF

ELF101 a Linux executable walkthrough

Ange Albertini
corkami.com

Dissected file

```
~$ uname -m
x86_64
~$ ./simple64.elf
Hello World!
```

simple64.elf

SHA-1: 1[illegible]29e77a501e43efccbb2eea80ee52e13ef6b3b
download @ elf101.corkami.com

Header 1/2
technical details for identification and execution

- **ELF header**
  identify as an ELF type
  specify the architecture
- **Program Header table**
  Execution information

sections
contents of the executable

- **Code**
  executable information
- **Data**
  information used by the code
- **Sections' names**

header 2/2
technical details for linking (ignored for execution)

- **Section Header table**
  Linking (connecting program objects) information

Hexadecimal dump | ASCII dump

```
7F 45 4C 46 02 01 01 00 00 00 00 00 00 00 00 00  .ELF............
02 00 3E 00 01 00 00 00 80 00 00 10 00 00 00 00  ..>.............
40 00 00 00 00 00 00 00 F0 00 00 00 00 00 00 00  @...............
00 00 00 00 40 00 38 00 01 00 40 00 04 00 03 00  ....@.8...@.....
```

1 Fields	Values	Explanation
e_ident		
EI_MAG	0x7F, "ELF"	constant signature
EI_CLASS, EI_DATA	2	64 bits, Little-Endian
EI_VERSION	1	Always 1
e_type	2	Executable
e_machine	0x3E	AMD 64 (and later)
e_version	1	Always 1
e_entry	0x10000080	3 Address where execution starts
e_phoff	0x40	Program Headers' offset
e_shoff	0xF0	Section Headers' offset
e_ehsize	0x40	Elf header's size
e_phentsize	0x38	Size of a single Program Header
e_phnum	1	Count of Program Headers
e_shentsize	0x40	Size of a single Section Header
e_shnum	4	Count of Section Headers
e_shstrndx	3	Index of the names' section in the table

```
Offset:0x40/Address:0x400040
01 00 00 00 05 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 10 00 00 00 00 00 00 00 10 00 00 00  ................
D0 00 00 00 00 00 00 00 D0 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00                          ........
```

2		
p_type	1	The segment should be loaded in memory
p_flags	5	Readable and eXecutable
p_offset	0	Offset where it should be read
p_vaddr	0x10000000	Virtual address where it should be loaded
p_paddr	0x10000000	Physical address where it should be loaded
p_filesz	0xD0	Size on file
p_memsz	0xD0	Size in memory

```
Offset:0x80/Address:0x10000080
48 BA 0D 00 00 00 00 00 00 00 48 C7 C6 C0 00 00  H.........H.....
10 48 C7 C7 01 00 00 00 48 C7 C0 01 00 00 00 0F  .H......H.......
05 48 C7 C7 01 00 00 00 48 C7 C0 3C 00 00 00 0F  .H......H..<....
05                                               .
```

3

x64 assembly | Equivalent C code

```
mov rdx, 0xD
mov rsi, 0x100000C0
mov rdi, 1
mov rax, 1
syscall                 → write(STDOUT_FILENO, "Hello World!\n", len("Hello World!\n"));

mov rdi, 1
mov rax, 0x3C
syscall                 → exit(1);
```

```
Offset:0xC0/Address:0x100000C0
48 65 6C 6C 6F 20 57 6F 72 6C 64 21 0A 00        Hello.World!..
```

Strings

```
"Hello World!\n", 0
```

```
Offset:0xD0
00 2E 73 68 73 74 72 74 61 62 00 2E 74 65 78 74  ..shstrtab..text
00 2E 72 6F 64 61 74 61 00                       ..rodata.
```

Section names

```
"" .shrtrtab .text .rodata
```

```
Offset:0xF0
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0B 00 00 00 01 00 00 00 06 00 00 00 00 00 00 00  ................
80 00 00 10 00 00 00 00 80 00 00 00 00 00 00 00  ................
31 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  1...............
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
11 00 00 00 01 00 00 00 02 00 00 00 00 00 00 00  ................
C0 00 00 10 00 00 00 00 C0 00 00 00 00 00 00 00  ................
0D 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
01 00 00 00 03 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 D0 00 00 00 00 00 00 00  ................
19 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

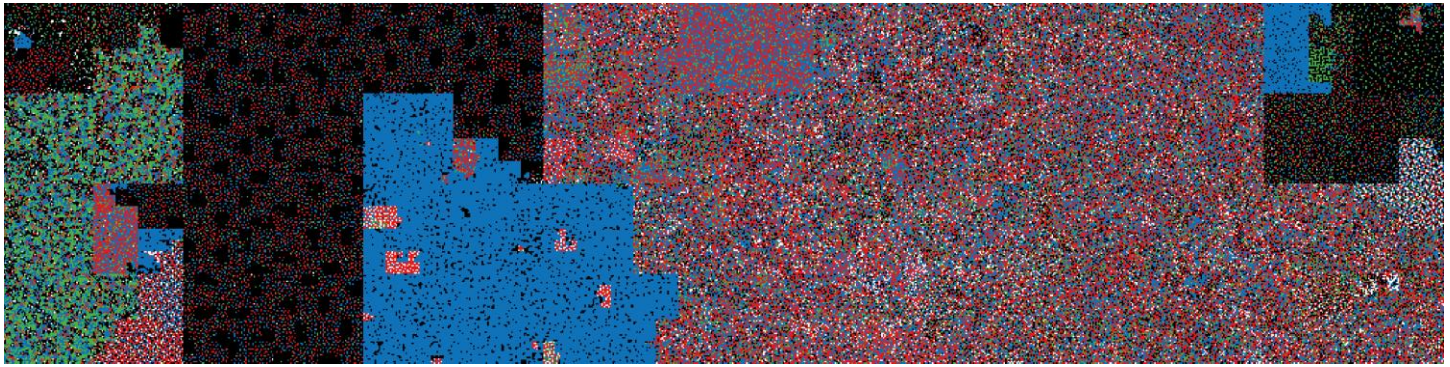
Section Header table

Index	Name	TYPE	FLAGS	ADDRESS	OFFSET	SIZ
0	<null>	0				
1	.text	1	6	0x10000080	0x80	0x31
2	.rodata	1	2	0x100000C0	0xC0	0x0D
3	*.shrtrtab	3			0xD0	0x19

relative offsets in names' section

This is the whole file, however, most ELF files contain many more elements.
Explanations are simplified, for conciseness.

A quick look inside an ELF executable



https://binvis.io/

- Statically vs. dynamically linked binaries
- `file exe`

```
exe: ELF 64-bit LSB  executable, x86-64, version
1 (GNU/Linux), for GNU/Linux 2.6.32, statically
linked, stripped
```

- `strings -a exe` (Reveals human-readable strings)