Digital Forensics: Data Analysis

Daniel Kouřil

Analysis of storage images

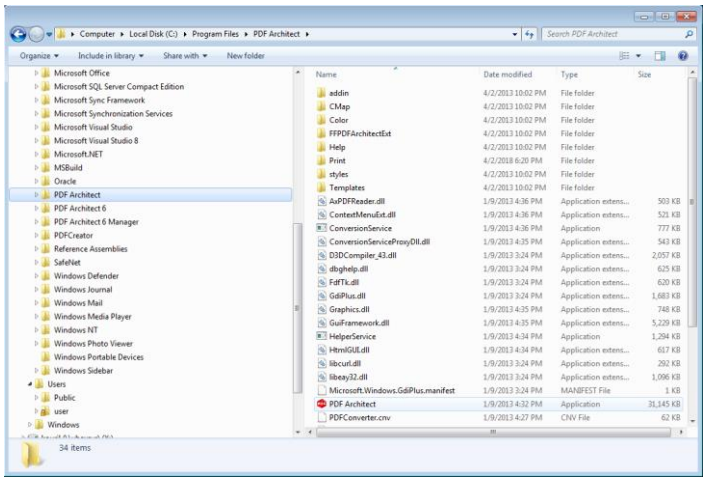
Analysis aims

- 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

```
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
```

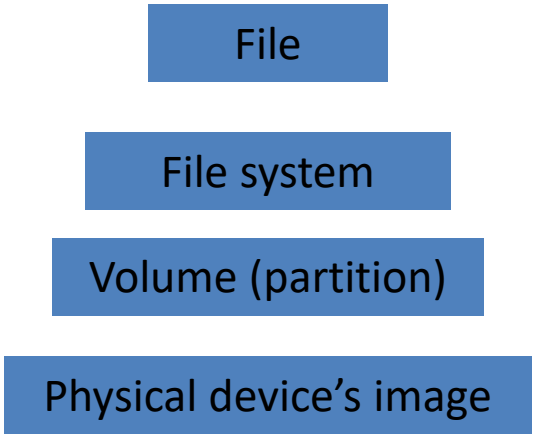


Analysis environment

- 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

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)

Procedure to analyze an image

- Take the input image as a single volume
- Break down the current volume to additional volumes (if any)
  - Detect all visible volumes and their types
  - Detect unallocated space
- Process identified volumes one by one
  - If the volume hosts a file system -> mark for subsequent analysis
  - Other (known) volumes (auxiliary) -> check if they contain other volumes (or their parts) and reconstruct them
  - Unknown volume type -> ad-hoc analysis
- Process file system volumes
  - Gather and evaluate information about files stored
  - Files can contain volumes also (ie. start again)
- Examine unallocated space

File system analysis

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

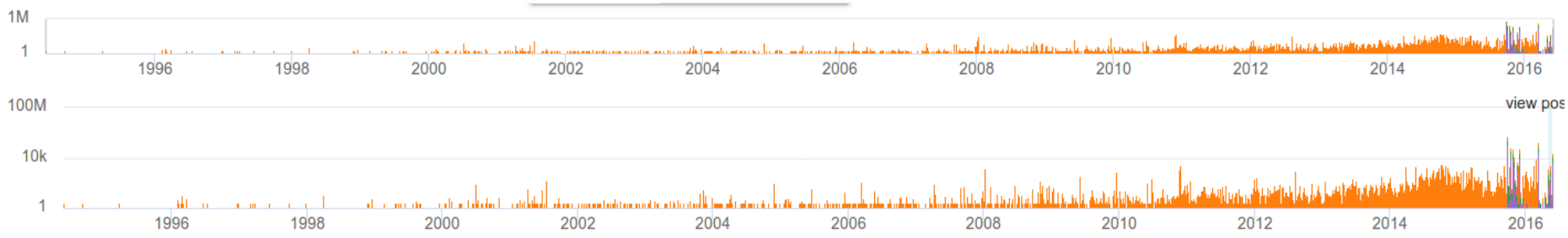
```
File:  /var/www/files/shell.php
Access:  2017-06-07 07:00:05 +0200
Modify:  2017-06-03 17:59:28 +0200
Change:  2017-06-03 18:01:40 +0200
```

```
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 2016-05-31T00:00:00.000

Interactive List View Total: 27862

.s

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/rwsr-xr	0	100	/var/lib/.s
2016-05-25 02:43:12	12400	...b	r/rwxr-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/rwsr-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

Fimetis by CSIRT-MU (https://github.com/CSIRT-MU/fimetis)

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)

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#

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

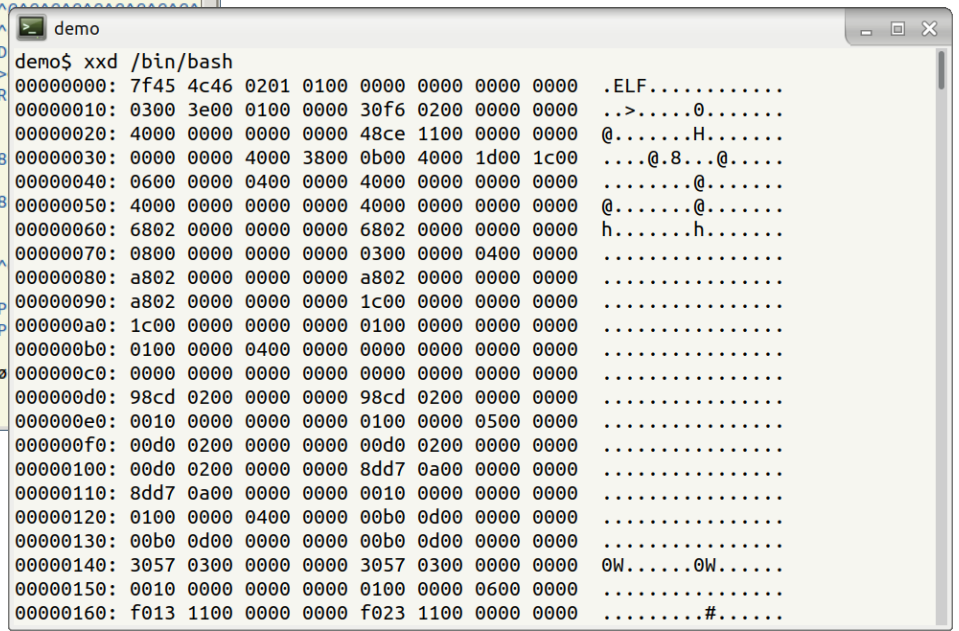
Analysis of executable files

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

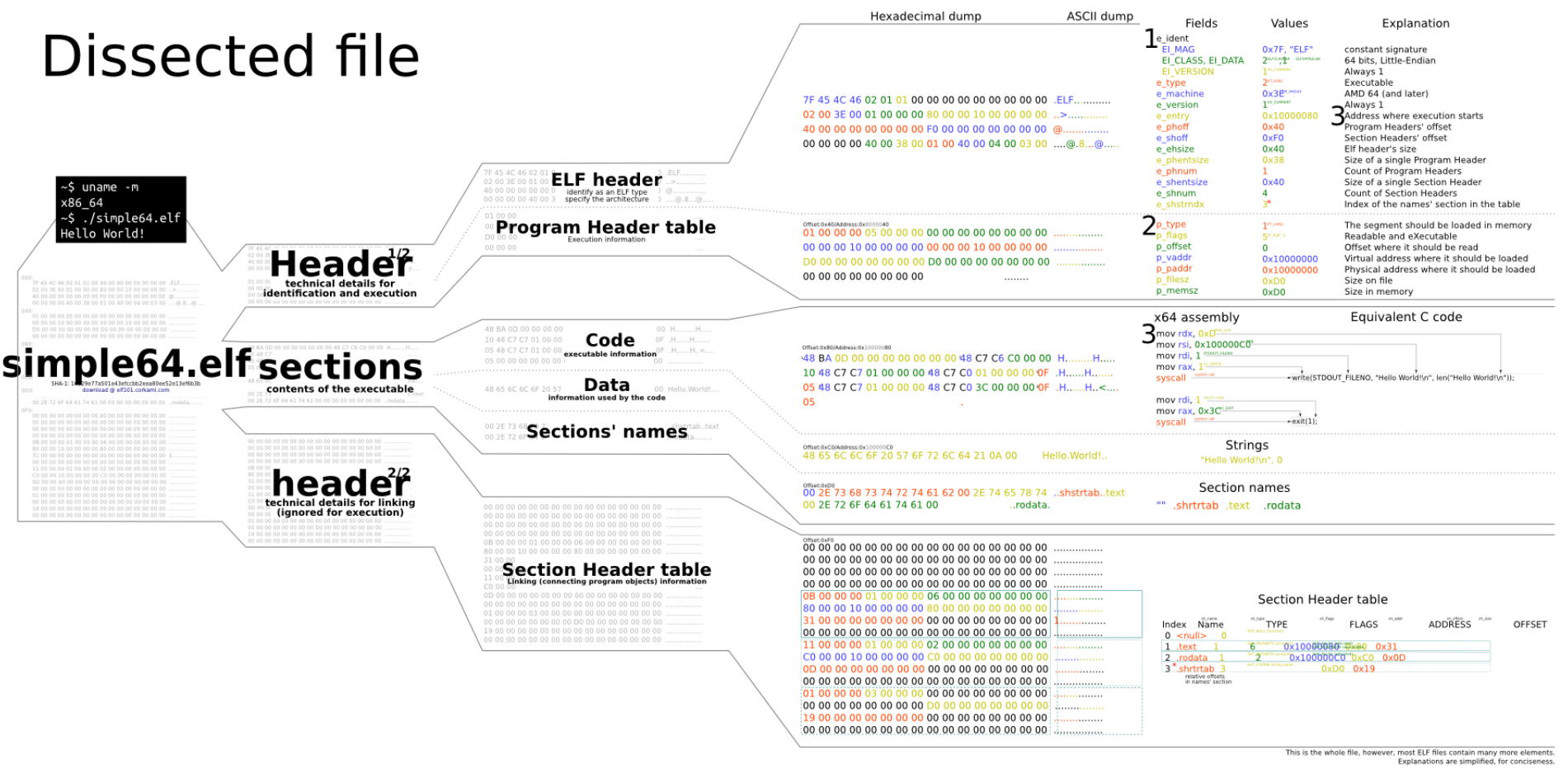


ELF

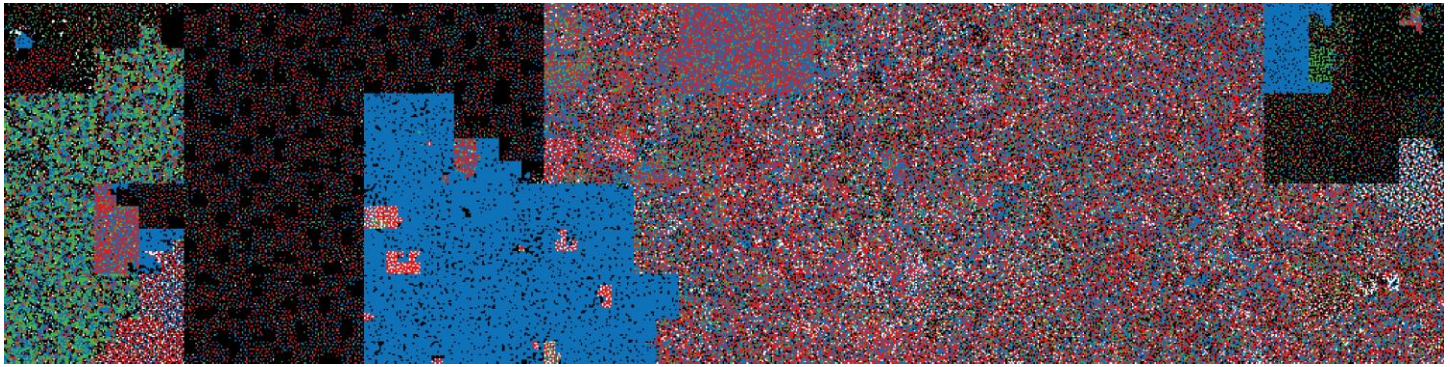
ELF101 a Linux executable walkthrough

Ange Albertini
corkami.com

Dissected file



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
```

Quick examination

- `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
- A number of other tools is available

Useful links

- https://confluence.egi.eu/display/EGIBG/Forensics+Howto
- https://www.dfn-cert.de/en/Trainings.html#ITForensics