Digital Forensics for SSC Solvers

Daniel Kouril
EGI CSIRT

Digital Forensics

- Methods to collect and analyze (digital) evidence
- Three basic phases
  - Data collection, Analysis, Reporting
- Various sources of information
  - Host, network
- Online (live) vs. Offline analysis
- https://wiki.egi.eu/wiki/Forensic_Howto

Triage – is there an incident or not?

- Minimize actions
  - Every contact leaves a trace
- Quickly examine the system
  - Looking for anomalies
  - Even minor things may matter
- If incident is confirmed- isolate services/machines
  - Proceed to contain incident

Starting investigation

- Leif Nixon’s cup of tea/coffee
- Don’t try to fix the system now
- Any applicable policies?
- Security contact(s), teams, …
- Do some documentation, note times
  - Save outputs
- Prepare for communication
- Isolate the system

Live analysis

- Part of triage
- Checking live system is often important
  - Access to memory and working system
- Memory can be gathered
  - Hard to hide something
  - Processes are “unlocked”
  - Data can only be available from memory
  - Independent view on OS structures
- But the system may not be yours anymore!

Performing live analysis

- Before you start, secure evidence that could be changed
  - Snapshot(s), take FS metadata, (RAM)
- Start with introspection of the whole system
  - Network connections, running processes, ….
  - Note processes for additional analysis
- After that, examine suspicious processes
  - resources used
  - recover files
  - obtain memory dumps

Processes

- A *process* is an instance of a *program*
  - Program is usually an executable file on a disk
- Process keeps data in memory, uses system resources
  - Sometimes released only during termination
- Processes form a hierarchy

System examination

- Closer look at processes
  - Strange names, executables
  - Distributions of PIDs, relationships, CPU consumption
- Resources in use
    - Memory, open sockets (files, networks), shared memory
- Investigations
  - User-space commands (common commands)
  - Check kernel structures
    - Correlation of command outputs, access lower-level info

Commands needed

- Commands
  - `ps`, `netstat`, `lsof`
- Kernel structures
  - `/proc/$PID`
- Document/record the process
  - Keep track of issued commands
  - Save outputs
    - Ramdisks (`/dev/shm`) might be an option

/proc records

/proc/31418

```
-r--r--r-- 1 kouril kouril 0 May  5 18:46 cmdline
lrwxrwxrwx 1 kouril kouril 0 May  5 18:46 cwd -> /tmp
-r-------- 1 kouril kouril 0 May  5 18:46 environ
lrwxrwxrwx 1 kouril kouril 0 May  5 18:46 exe -> /usr/bin/wget

dr-x------ 2 kouril kouril 0 May  5 18:46 fd
lrwx------ 1 kouril kouril 64 May  5 18:46 0 -> /dev/pts/47
lrwx------ 1 kouril kouril 64 May  5 18:46 1 -> /dev/pts/47
lrwx------ 1 kouril kouril 64 May  5 18:46 2 -> /dev/pts/47
lrwx------ 1 kouril kouril 64 May  5 18:46 3 -> socket:[3097580]
l-wx------ 1 kouril kouril 64 May  5 18:46 4 -> /tmp/ubuntu-19.04-
desktop-amd64.iso?_ga=2.213675796.1604966281.1557074696-
1247976767.1557074696
```

Deleted files

- Unix keeps deleted files open until they are closed
- `ls /proc/$PID/exe`:

```
lrwxrwxrwx 1 kouril kouril 0 May  4 07:31 exe ->
                                        /tmp/wget (deleted)
```

- Proc's "symbolic links" can be used for easy recovering the data
  - `cp/cat/… /proc/$PID/exe /tmp/dest`
  - The process must be still running!
- Both executable and open files (see the fd directory)

Open files
(lsof -p 31418 –n)

```
COMMAND   PID   USER   FD   TYPE  DEVICE SIZE/OFF    NODE NAME
wget    31418 kouril  cwd    DIR   252,0    36864  524291 /tmp
wget    31418 kouril  rtd    DIR   252,0     4096       2 /
wget    31418 kouril  txt    REG   252,0   407696  393524 /usr/bin/wget
wget    31418 kouril  mem    REG   252,0    43616 1049154 /lib/x86_64-linux-gnu/
libnss_files-2.19.so
wget    31418 kouril  mem    REG   252,0  3165552  394658 /usr/lib/locale/locale-arc
wget    31418 kouril  mem    REG   252,0    14664 1064472 /lib/x86_64-linux-gnu/libd
wget    31418 kouril  mem    REG   252,0  1857312 1064478 /lib/x86_64-linux-gnu/libc
wget    31418 kouril  mem    REG   252,0    18936 1050745 /lib/x86_64-linux-gnu/libu
wget    31418 kouril  mem    REG   252,0   207128  397935 /usr/lib/x86_64-linux-gnu/
wget    31418 kouril  mem    REG   252,0   100728 1048634 /lib/x86_64-linux-gnu/libz
wget    31418 kouril  mem    REG   252,0  1938752 1050644 /lib/x86_64-linux-gnu/libc
wget    31418 kouril  mem    REG   252,0   387272 1050636 /lib/x86_64-linux-gnu/libs
wget    31418 kouril  mem    REG   252,0   149120 1064460 /lib/x86_64-linux-gnu/ld-2
wget    31418 kouril  mem    REG   252,0    26258  671480 /usr/lib/x86_64-linux-gnu/
wget    31418 kouril    0u   CHR  136,47      0t0      50 /dev/pts/47
wget    31418 kouril    1u   CHR  136,47      0t0      50 /dev/pts/47
wget    31418 kouril    2u   CHR  136,47      0t0      50 /dev/pts/47
wget    31418 kouril    3u  IPv4 3101285      0t0     TCP 127.0.0.1:44280->127.0.0.1
wget    31418 kouril    4w   REG   252,0 14777874  573900 /tmp/ubuntu-19.04-desktop-
```

Open network connections (netstat –tnp)

```
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address       Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:9050          127.0.0.1:34902         ESTABLISHED -
tcp        0      0 127.0.0.1:44280         127.0.0.1:8118          ESTABLISHED 31418/wget
```

Dumping process memory

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

Executable file analysis

- Static analysis
- Dynamic analysis

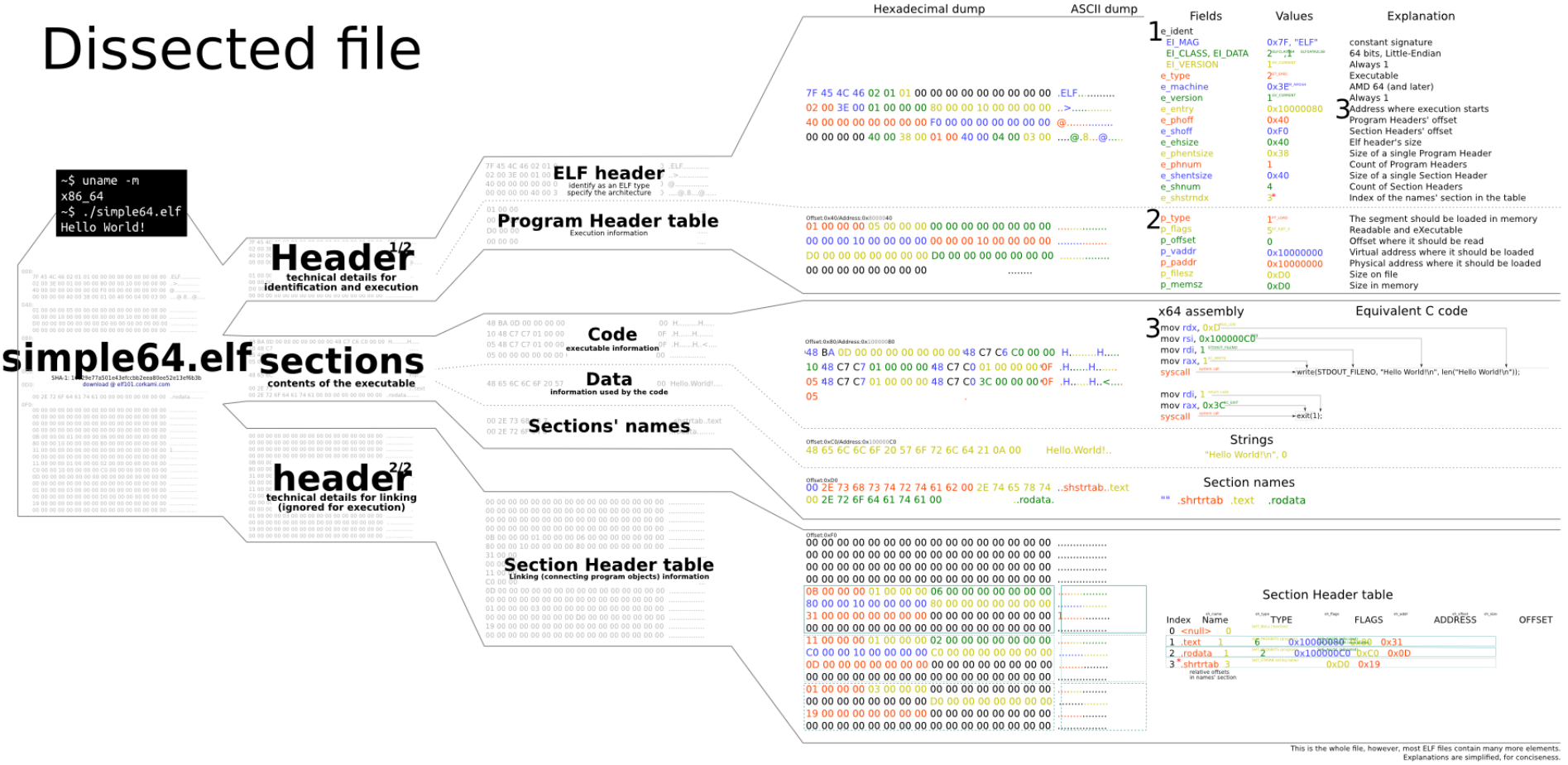
Binary executable analysis

ELF

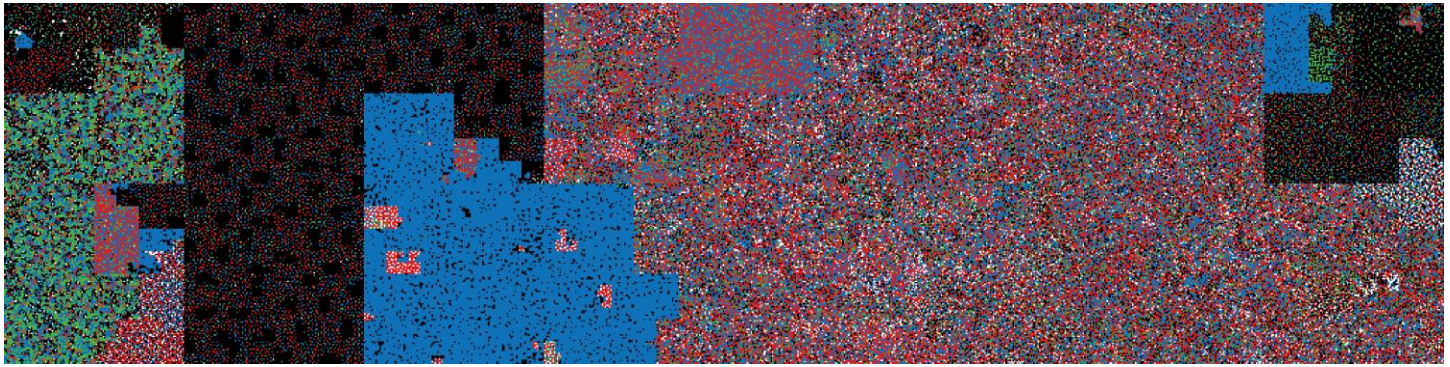
ELF101 a Linux executable walkthrough

Ange Albertini
corkami.com

Dissected file



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
```

Static analysis of binary files

- Determine the type

```
file /bin/bash
/bin/bash: ELF 64-bit LSB  executable, x86-64, version 1 (SYSV), dynamically linked (uses
shared libs), for GNU/Linux 2.6.24,
BuildID[sha1]=7e4c4de7a4d259aeb0896fd579609bb6c27fae8d, stripped
```

- Content analysis
  - Either break down individual ELF sections and analyse them
    - .rodata, .data - constants (strings), text - code
    - `readelf`, Python elftools
  - or do quick examination of the whole file
    - Human readable strings
      - `strings –a <binary>`
    - Strings often point to username, file paths, function names, . . .
    - Malware producers tend to obfuscate important strings
      - XOR, base64, . . .
      - Dynamic calls to library functions
      - dlopen(), dlsym()

Countermeasures

- Encoded (packed) binaries
  - Binary is encoded by a customized algorithm and gets unpacked during executions
  - Binary executables – in place extraction
  - Scripts – self-executable archives of files
- Obfuscated scripts
  - very often used for PHP or Javascript

Executable packer UPX

- LZMA-based compression applied on executable, which yields another executable
  - An unpacking routine at the beginning
  - Extraction to process memory
- Easily to detect
  - No human-readable strings
  - `This file is packed with the UPX executable packer http://upx.sf.net`
- Difficult to analyze directly
  - Contents of the binary is compressed
- `upx -d <binary>` decodes the original file

Next Session

- A joint walk-through the SSC malware
- VMs available for hands-on exercise
- SSH client necessary, access credentials will be circulated