ISGC 2017 Security Workshop

Vincent Brillault

Computer Security forensics

www.egi.eu

EGI-Engage is co-funded by the Horizon 2020 Framework Programme
of the European Union under grant number 654142

Introduction

Live Analysis

Analysing the data collected

Offline Analysis

Preparing for next time

Credits & Resources

- Credits:
  - ”Quick & Dirty forensics” by Leif Nixon
  - EGI’s Forensic Howto by Heiko Reise
- New EGI guide: https://wiki.egi.eu/wiki/Forensic

EGI-ENGAGE

CERTIFIED BY TI TRUSTED INTRODUCER

Introduction

Why bother with forensics?

We can simply reinstall can’t we?

Why bother with forensics?

We can simply reinstall can’t we? NO!

- What if attack was before last backup?
- What if the hole that was used is still there?
- Which credentials were stolen? Is there a backdoor?
- What if other systems were compromised?

Why bother with forensics?

We can simply reinstall can’t we? NO!

- What if attack was before last backup?
  → When did the compromise happened?
- What if the hole that was used is still there?
  → How was it compromised?
- Which credentials were stolen? Is there a backdoor?
  → What was done?
- What if other systems were compromised?
  → What else was impacted?

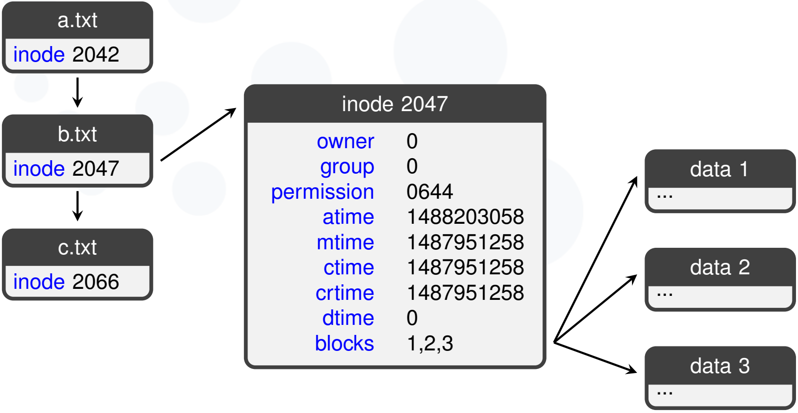
Otherwise attackers might still be in or will come back!

Most of your action will have consequences

- Observation changes the observed object:
  - List folder content: change folder atime
  - Read a file: change file atime
  - Run a command: change binary atime
- Writing on disk destroy evidences:
  - Write to file: Write on *free* sectors
  - Leave syslog running: Write on *free* sectors
- Killing process releases open file & destroy memory
- Rebooting kills all processes & destroy memory

→ Do the least to keep the most

Inode-based filesystem 101



Filesystem timestamps

- Which timestamps?

atime	Last file read or folder listed
mtime	Last modification
ctime	Last data or metadata (inode) change
crtime	Creation time. Only on ext4
dtime	Deletion time

- Can you trust them?

atime/mtime	Changed by *touch*, *tar*, *wget*
*time	Based on local time, bits on drive
dtime/crtime	Not shown by stat, less accessible

Live Analysis

Danger from live analysis

- Common traps:
  - Malicious kernel module/rootkit
  - Malicious libraries (ld-preload or replacement)
  - Malicious binaries
- Are you alone?
  - Attacker might still be around
  - If possible, isolate system

Let’s get started...

- ... go get a coffee/tee/... ;)
  - Also pickup paper & pen, to keep tracks
  - Find a colleague
- Use *script* to record what you do:

```
script -t$CASEID.timing $CASEID.log
```

- Get a live shell
  - If VM, work from a snapshot/clone
  - Otherwise use local credentials or revoke them afterwards

Setup Work environment

- Avoid writing to disk: `export HISTFILE=/dev/null`
- Find a tmpfs or network-mount large enough to work in
  → Create a folder and work from there!
- Later (after timestamps): put static copies of tools here

Data Volatility

Registers, peripheral memory, caches, etc.	nanoseconds
Main memory	nanoseconds
Network state	milliseconds
Running processes	seconds
Disk	minutes
Backup media, etc.	years
Printouts, etc.	tens of years

Table borrowed from Forensic Discovery, Farmer & Venema, Addison-Wesley 2005)

However, you might want to over-prioritize file-system metadata

Filesystem metadata collection

- Your choice:
  - Using specialized software (TSK)
  - With `stat`, after remounting read-only
  - With `stat`, directly
- If you have a snapshot, ignore this, you will do it off-line
- Should be done for all mounted filesystems:
  - Root filesystem: as soon as possible
  - Home, log, ...: later (no bin)

Filesystem metadata collection

Using specialized software (TSK)

- Not obvious to use, experience required:

  ```
  fls -m '/' -r /dev/sda1 > sda.files
  mactime -b sda.files > sda.timeline
  ```

- Raw drive access: not fooled by rootkits
- Raw drive access: Kernel cache?
- Obtains deleted files & creation time
- Requires TSK binaries:
  - Copy them from remote
  - Via existing network mount
  - Via USB key

Filesystem metadata collection

With `stat`, after remounting read-only

- Basic shell commands[1]:

```
mkdir newmount && mount --bind / newmount && mount -o remount,ro newmount && cd newmount
find . -print0 | xargs -0 stat -c "%Y %X %Z %A %U %G %n" -- > ../root.files
```

- Using kernel: fooled by rootkits
- Can't see deleted files or creation time
- Calls local binaries: altered atimes

[1] Decorator: https://wiki.egi.eu/wiki/Forensic#TimelineDecorator

Filesystem metadata collection

With `stat`, directly

- Basic shell commands[2]:

```
find / -xdev -exec stat -c "%Y %X %Z %A %U %G %n" -- ’’  > ../root.files
```

- Will modify all atime on folders
- Using kernel: fooled by rootkits
- Can’t see deleted files or creation time
- Calls local binaries: altered atimes

[2] Decorator: https://wiki.egi.eu/wiki/Forensic#TimelineDecorator

Collect all relevant live data: Network

- Network sockets and connections:

```
netstat -apn | tee netstat_apn.txt
```

- Network environment:

```
ip -4 neigh show | tee ip6_neigh_show.txt
ip -4 route show | tee ip6_route_list.txt
ip -4 link  show | tee ip6_link_show.txt
ip -6 neigh show | tee ip6_neigh_show.txt
ip -6 route show | tee ip6_route_list.txt
ip -6 link  show | tee ip6_link_show.txt
```

Collect all relevant live data: Users

- User connections

```
w > w.txt
last | tee last.txt
lastlog | tee lastlog.txt
```

Collect all relevant live data: Processes

- Running processes

```
ps -auxwwwe | tee ps_auxwwwe.txt
pstree -lap | tee pstree_lap.txt
```

- Files open

```
lsof -b -l -P -X -n -o -R -U | tee lsof_blPXnoRU.txt
lsof -b -l -P -X -n -o -R > tee lsof_blPXnoR.txt
```

Collect all relevant live data: System

- Mounted devices:

```
cat /proc/mounts | tee proc_mounts.txt
```

- Kernel modules:

```
cat /proc/modules | tee proc_modules
ls /sys/modules |tee sys_modules
```

Dump interesting processes

- Stop it: `kill -STOP $PID`
- Dump it: `gcore $PID`
- Find interesting open files with `lsof -p $PID`
- Save them, e.g.:

```
cp /proc/$PID/exe $PID.exe
cp /proc/$PID/fd/$FDNUM $FILENAME
cp /dev/shm/$FILENAME $FILENAME
```

- Keep process information:

```
tar cvf proc_$PID.tar /proc/$PID/{auxv,cgroup,cmdline,comm,environ,limits,maps,sched,
schedstat,sessionid,smaps,stack,stat,statm,status,syscall,wchan}
```

[Optional] Dumping whole memory

- Specialized dumper: LIME
- Analysis: Volatility

[Optional] Automated scans

- Automated scanners
  - Package integrity: `rpm -Va`, `debsum`,...
  - Rootkit detection: chkrootkit, rkhunter, ossec-rootcheck
- Installing and running such tool will temper evidences

Analysing the data collected

Analysis rules

- Never trust data from compromised system
- Corroborate between different local sources
- Check if actually possible (paradox?)
- Corroborate with external sources

It’s a hide & seek game/war!

Filesystem timeline analysis

- Rule of thumb:
  - atime/mtime usually manipulated
  - ctime less often manipulated
  - crtime/dtime rarely manipulated
- Look for weird folders (e.g. `...`, in `/var/tmp`...)
- Check mtime/ctime/crtime on binaries (`{/usr,}/{s,}{bin}`): Most likely malicious if updated without package update
- Idem for libraries
- Look for incoherences:
  - File created after last modification on folder
  - File created after its last modification
- Check compilation traces: atime in */usr/include*

Checking processes & network

- Process name irrelevant: easily faked
- Weird parent/child relationship?
- Weird open network sockets?
- Raw socket?
- Duplicated system process
- Check pid ranges:
  kernel/system pids usually packed together

Check user accesses

- Connection logs can be in 3 places
  - `/var/log/wtmp`: used by `last`
  - `/var/log/secure`: SSHD logs
  - `/var/log/audit/audit.log*`: audit logs, incl. auth.

  → One of them might not have been cleaned!
- Check pattern change (password/key/kerberos)
- Check `/.ssh/authorized_keys` metadata

Offline Analysis

Stopping the system

- Only after obtaining live evidences
- Remember to get your evidences if in tmpfs
- Don’t go through shutdown:
  - Use Sysrq keys: mount read-only, sync, shutdown
  - In the worst case, unplug the cable

Imaging the disk

- Disable auto-mount before connecting hard-drive
- Identify each drive after connection
- Use basic old `dd`:

```
dd if=/dev/sdX of=file.img bs=65536 conv=noerror,sync status=progress
```

Access disk image: TSK

- Identify partition offset:

  `mmls file.img`

```
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

      Slot      Start        End          Length       Description
000:  Meta      0000000000   0000000000   0000000001   Primary Table (\#0)
001:  -------   0000000000   0000002047   0000002048   Unallocated
002:  000:000   0000002048   0020971220   0020969173   Linux (0x83)
003:  -------   0020971221   0020971519   0000000299   Unallocated
```

Access disk image: TSK

- Identify partition offset:

```
mmls file.img
```

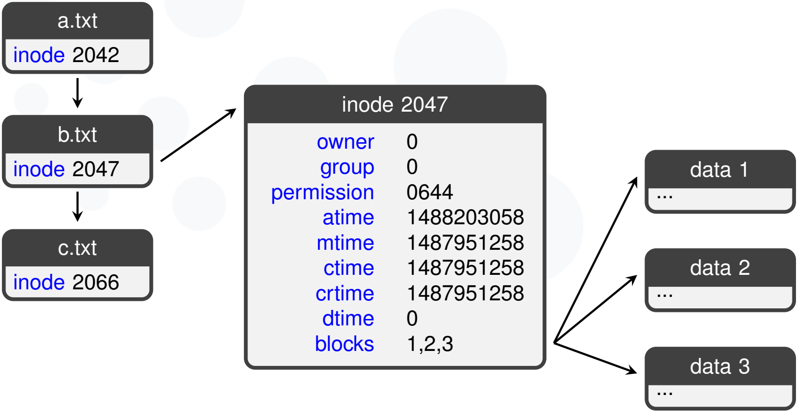
- Extract timeline: as before, with offset

```
fls -o 2048 -m '/' -r file.image > sda.files
mactime -b sda.files > sda.timeline
```

- Extract files (from inode, here 261257):

```
icat -o 2048 file.image 261257
```

Inode-based filesystem 102
File Deletion



a.txt
inode 2042

b.txt
inode 2047

c.txt
inode 2066

inode 2047

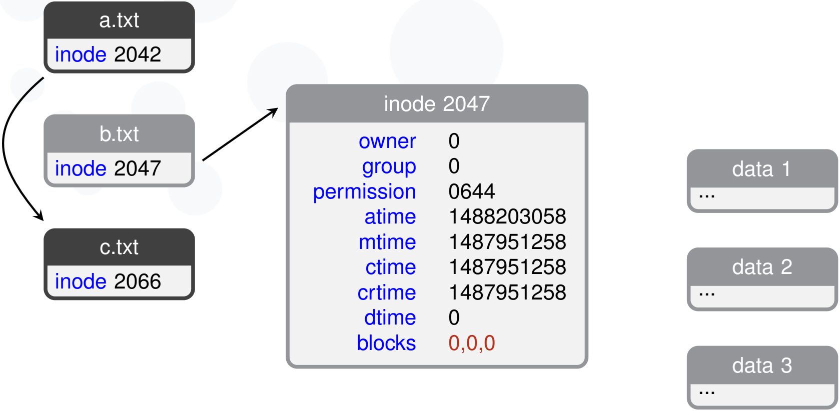
owner	0
group	0
permission	0644
atime	1488203058
mtime	1487951258
ctime	1487951258
crtime	1487951258
dtime	0
blocks	1,2,3

data 1
…

data 2
…

data 3
…

Inode-based filesystem 102
File Deletion



Deleted file/data recovery

- Recover recently deleted files:
  - ext2: Testdisk
  - ext3/4: extundelete
- *Carve* unallocated sectors using `photorec` from Testdisk
- `grep` image file directly
  → Will also find data in *slack* space

Basic malware analysis

- Run `strings -a` on it
- ’Dynamic’ analysis on isolated VM: `strace` & `ltrace`

Preparing for next time

Prepare some tools/hardware

- Download & test:
  - Testdisk for data recovery and carving
  - The Sleuth Kit for offline file-system analysis
  - extundelete for data recovery on ext3/ext4
- Prepare a USB key
  - A bootable Linux distribution, without auto-mount
  - The tools mentioned above
  - A README with commands that you might run & scripts
- Prepare some large storage for evidence/images
- (Optional) For offline forensics: a USB-SATA adapter

Prepare your systems

- Collect syslog remotely on a central server!
- Avoid losing evidences
  - Disable *prelink*
  - Avoid cronjob that read all files
  - Avoid mounting with *noatime*
- Install (& test) in advance basic debugging tools:
  - *netstat*: open sockets
  - *lsof*: open files
  - *pstree* (from *psmicc*): tree of processes
  - *gcore* (from *gdb*): generate *core* of running processes
- Enforce kernel module signature validation

EGI-ENGAGE

That’s it!

- Questions?

- Good luck for your next forensics

  You will be able to try a bit this afternoon!