Lessons learned

In a Forensics Lab

CIRCL
Computer Incident
Response Center
Luxembourg

Michael Hamm - *TLP:GREEN*

info@circl.lu

2019-05-30

1. Hiding data in HPA

- ATA-3: Hard disk password
- ATA-4: HPA - Host Protected Area
  - Vendor area - benefit system vendors
  - Recovery data. persistent data
  - Controlled by firmware not OS
- ATA-6: DCO - Device Configuration Overlay
  - Benefit system vendors
  - Control reported capacity and disk features
  - Use disk from different manufacturers
  - Use disk with different number of sectors
    $\rightarrow$ Makes disks looking uniq

$\rightarrow$ Exercise

1.1 Create HPA

- New disk

```
dmesg
    sd 1:0:0:0: [sdb] 3900000000 512-byte logical blocks: (2.00 TB/1.82 TiB)
```

- Create hidden data

```
echo -n 'MySecret 123456' | dd of=/dev/sdb seek=3500000000

dd if=/bin/dd of=/dev/sdb seek=3500000001
     148+1 records in
     148+1 records out
     76000 bytes (76 kB, 74 KiB) copied, 0,022659 s, 3,4 MB/s
```

- Review

```
dd if=/dev/sdb skip=3500000000 count=2 status=none | xxd | less
    00000000: 4d79 5365 6372 6574 2031 3233 3435 3600  MySecret 123456.
    00000010: 0000 0000 0000 0000 0000 0000 0000 0000  ................
```

- Create HPA

```
hdparm --yes-i-know-what-i-am-doing -N p3000000000 /dev/sdb
    setting max visible sectors to 3000000000 (permanent)
    max sectors   = 3000000000/3907029168, ACCESSIBLE MAX ADDRESS enabled

Power cycle your device after every ACCESSIBLE MAX ADDRESS
```

1.2 Put disk into production

- Create partition and format

```
dmesg
    sd 1:0:0:0: [sdb] 3000000000 512-byte logical blocks: (1.54 TB/1.40 TiB)

fdisk /dev/sdb
    primary
    2048
    2999999999

mkfs.ntfs -L CIRCL.DFIR -f /dev/sdb1
    Creating NTFS volume structures.
    mkntfs completed successfully. Have a nice day.
```

- Investigate disk layout

```
fdisk -l /dev/sdb
    Device     Boot Start        End    Sectors  Size Id Type
    /dev/sdb1        2048 2999999999 2999997952  1,4T  7 HPFS/NTFS/exFAT
```

- Investigate last accessible sector + more

```
dd if=/dev/sdb skip=2999999999 count=2 status=none| xxd
    00000000: eb52 904e 5446 5320 2020 2000 0208 0000  .R.NTFS    .....
      ........
    000001f0: 0000 0000 0000 0000 0000 0000 0000 55aa  ..............U.
```

1.3 Recover hidden data

- Try to access hidden message

```
dd if=/dev/sdb skip=3500000000 count=2 | xxd
    dd: /dev/sdb: cannot skip: Invalid argument
    0+0 records in
```

- Resize HPA

```
hdparm --yes-i-know-what-i-am-doing -N p3900000000 /dev/sdb
    max sectors   = 3900000000/3907029168, ACCESSIBLE MAX ADDRESS enabled

Power cycle your device after every ACCESSIBLE MAX ADDRESS
```

- Investigate disk layout and last sector

```
fdisk -l /dev/sdb
    Device     Boot Start        End    Sectors  Size Id Type
    /dev/sdb1        2048 2999999999 2999997952  1,4T  7 HPFS/NTFS/exFAT

dd if=/dev/sdb skip=2999999999 count=2 status=none | xxd | less
    00000000: eb52 904e 5446 5320 2020 2000 0208 0000  .R.NTFS    .....
    ......
```

1.3 Recover hidden data

- Recover hidden message

```
dd if=/dev/sdb skip=3500000000 count=1 status=none
    00000000: 4d79 5365 6372 6574 2031 3233 3435 3600  MySecret 123456.
```

- Recover hidden dd command

```
dd if=/dev/sdb skip=$(( 3500000001*512 )) count=76000 bs=1 of=dd.exe

md5sum dd.exe
    36a70f825b8b71a3d9ba3ac9c5800683

md5sum /bin/dd
    36a70f825b8b71a3d9ba3ac9c5800683
```

- Feeback:

```
https://www.schneier.com/blog/archives/2014/02/swap_nsa_exploi.html
https://en.wikipedia.org/wiki/Host_protected_area
```

- How it works

```
IDENTIFY DEVICE
SET MAX ADDRESS
READ NATIVE MAX ADDRESS
—> HPA aware software (like the BIOS)
```

2. Block device I/O Error

- `https://github.com/adulau/dcfldd/issues/1`

Known issues #1

Open **msuhanov** opened this issue Oct 24, 2018 · 3 comments

msuhanov commented Oct 24, 2018 • edited

1. Data becomes misaligned when a faulty sector is encountered on a source drive. NIST report, upstream patch.

Github

2.1 Setup virtual block device

```
xxd a.raw | less

    00000000: 4265 6769 6e5f 6f66 5f73 6563 746f 725f  Begin_of_sector_
    00000010: 3120 2020 2041 4141 4141 4141 4141 4141  1    AAAAAAAAAAA
    ...
    ...
    000001e0: 4141 4141 4141 4141 4141 4141 4145 6e64  AAAAAAAAAAAAAEnd
    000001f0: 5f6f 665f 7365 6374 6f72 5f31 2020 2020  _of_sector_1
    00000200: 4265 6769 6e5f 6f66 5f73 6563 746f 725f  Begin_of_sector_
    00000210: 3220 2020 2041 4141 4141 4141 4141 4141  2    AAAAAAAAAAA
    ...
    ...
    009fffe0: 4141 4141 4141 4141 4141 4141 4145 6e64  AAAAAAAAAAAAAEnd
    009ffff0: 5f6f 665f 7365 6374 6f72 5f32 3034 3830  _of_sector_20480


losetup -f
losetup /dev/loop27 a.raw
losetup /dev/loop28 a.raw

# Table
# 0 10239 linear /dev/loop27 0
# 10239 1 error
# 10240 10240 linear /dev/loop28 10240

echo -e "0 10239 linear /dev/loop27 0\n10239 1 error\n10240 10240 linear /dev/loop28 10240
blockdev --getsize64 /dev/mapper/dcfldd
```

2.2 Results

- Create image files

```
dd      if=/dev/mapper/dcfldd of=a.dd      conv=noerror,sync
dcfldd  if=/dev/mapper/dcfldd of=a.dcfldd  conv=noerror,sync
dc3dd   if=/dev/mapper/dcfldd of=a.dc3dd
```

- Compare output

```
ls -l a.*
    10485760 Mai 30 05:31 a.dc3dd
    10518528 Mai 30 05:30 a.dcfldd
    10485760 Mai 30 05:25 a.dd
    10485760 Mai 30 05:04 a.raw
```

- What about `md5sum`

```
md5sum a.*
     174a9f3eaa53376c7a369ef9cb83665c  a.dc3dd
     b1c33fcf4f9564b7486d09f9c032f615  a.dcfldd
     41f4d594a9b7272391d6f52621d656a1  a.dd
     2b6eb69b776741cb9721524c62d34a5c  a.raw
```

2.3 Investigate results

- Investigate dc3dd output

```
xxd a.dc3dd | less
    004ffdd0: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
    004ffde0: 4141 4141 4141 4141 4141 4141 4145 6e64  AAAAAAAAAAAAAAEnd
    004ffdf0: 5f6f 665f 7365 6374 6f72 5f31 3032 3339  _of_sector_10239
    004ffe00: 0000 0000 0000 0000 0000 0000 0000 0000  ................
    004ffe10: 0000 0000 0000 0000 0000 0000 0000 0000  ................
    ...
    ...
    004ffff0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
    00500000: 4265 6769 6e5f 6f66 5f73 6563 746f 725f  Begin_of_sector_
    00500010: 3130 3234 3141 4141 4141 4141 4141 4141  10241AAAAAAAAAAA
    00500020: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
```

- Conclusion:

```
--> 1 Sector broken 10239
```

2.3 Investigate results

- Investigate dd output

```
xxd a.dd | less
    004fefd0: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
    004fefe0: 4141 4141 4141 4141 4141 4141 4145 6e64  AAAAAAAAAAAAAAEnd
    004feff0: 5f6f 665f 7365 6374 6f72 5f31 3032 3332  _of_sector_10232
    004ff000: 0000 0000 0000 0000 0000 0000 0000 0000  ................
    004ff010: 0000 0000 0000 0000 0000 0000 0000 0000  ................
    004ff020: 0000 0000 0000 0000 0000 0000 0000 0000  ................
    ...
    ...
    004fffe0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
    004ffff0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
    00500000: 4265 6769 6e5f 6f66 5f73 6563 746f 725f  Begin_of_sector_
    00500010: 3130 3234 3141 4141 4141 4141 4141 4141  10241AAAAAAAAAAA
    00500020: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
    ...
    ...
```

- Conclusion:

```
--> 8 Sectors empty
--> First broken sector: 10233
--> Last broken sector:  10240
```

2.3 Investigate results

- Investigate `dcfldd` output

```
xxd a.dcfldd
    004fefd0: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
    004fefe0: 4141 4141 4141 4141 4141 4141 4145 6e64  AAAAAAAAAAAAAAEnd
    004feff0: 5f6f 665f 7365 6374 6f72 5f31 3032 3332  _of_sector_10232
    004ff000: 0000 0000 0000 0000 0000 0000 0000 0000  ................
    004ff010: 0000 0000 0000 0000 0000 0000 0000 0000  ................
    ...
    00507fe0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
    00507ff0: 0000 0000 0000 0000 0000 0000 0000 0000  ................
    00508000: 4265 6769 6e5f 6f66 5f73 6563 746f 725f  Begin_of_sector_
    00508010: 3130 3239 3741 4141 4141 4141 4141 4141  10297AAAAAAAAAAA
    00508020: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
    ...
    ...
    00a00fd0: 4141 4141 4141 4141 4141 4141 4141 4141  AAAAAAAAAAAAAAAA
    00a00fe0: 4141 4141 4141 4141 4141 4141 4145 6e64  AAAAAAAAAAAAAAEnd
    00a00ff0: 5f6f 665f 7365 6374 6f72 5f32 3034 3830  _of_sector_20480
    00a01000: 0000 0000 0000 0000 0000 0000 0000 0000  ................
    00a01010: 0000 0000 0000 0000 0000 0000 0000 0000  ................
```

- Conclusion:

```
--> 72 empty sectors
--> 64 missing sectors
--> 56 additional empty sectors at end of file
```

2.4 Feedback

- Feedback D. Byers

  David Byers and Nahid Shahmehri.
  Contagious errors:
  Understanding and avoiding issues with imaging drives containing faulty sectors.
  The International Journal of Digital Forensics and Incident Response,
  ISSN 1742-2876, E-ISSN 1873-202X, Vol. 5, no 1, p. 29-33

  - READ call goes through the kernel page cache
    → Linux reads 4096 bytes
    → 1 faulty sector → fail to read 8 sectors
  - Reading a device in direct I/O mode solve the problem
    → `dd if=/dev/mapper/dcfldd of=addIO iflag=direct conv=noerror,sync`

- MD5

```
md5sum a.*
    174a9f3eaa53376c7a369ef9cb83665c  a.dc3dd
    174a9f3eaa53376c7a369ef9cb83665c  a.ddIO
    .....
```

3. Modify data on a "Read Only" mounted device

- Message in a forensic book

  "Nothing will prevent your Linux system to modify data on a read only mounted device"

- Leads to a new exercise for CIRCL DFIR 1.0.1 training
- I will
  - Targeted tamper of evidences
  - Only use on-board-tools
  - Be `root`
  - Cheat (A little bit)
- I will not
  - Remount the device in RW mode
- Any ideas?

3.1 Play Script

1. Identify how the device is connected
2. Review mount options
3. Re-mount the device in "Read Only" mode
4. Review mount options
5. Open file, modify data and try to safe
6. Use `strings` to identify offest of the data
7. Calculate sector number
8. `dd` sector on to local disk
9. Modify local stored sector with a hexeditor
10. `dd` sector back on RO mounted block device
11. Validate results

3.1 SCRIPT

```
dmesg
    sd 1:0:0:0: [sdb] Write Protect is off
    sdb: sdb1
    sd 1:0:0:0: [sdb] Attached SCSI removable disk
mount
    /dev/sdb1 on /media/michael/CIRCL-DFIR type vfat (rw,nosuid,

mount -o remount,ro /dev/sdb1 /media/michael/CIRCL-DFIR/
mount
    /dev/sdb1 on /media/michael/CIRCL-DFIR type vfat (ro,relatime,

strings -td /dev/sdb1 | grep Hello
    298897 Hello World!
    299106 Hello World!
echo $((299106/512))
    584

dd if=/dev/sdb1 bs=512 skip=584 count=1 of=584.raw
ls -l 584.raw
    -rw-r--r-- 1 root root 512 Mai 30 08:33 584.raw

hexer 584.raw

dd if=584.raw seek=584 of=/dev/sdb1
```

3.2 Countermeasures

- Try on board methods:
  - `hdparm -r1 /dev/sdb`
  - `blockdev --setro /dev/sdb`
  - `udev` rules

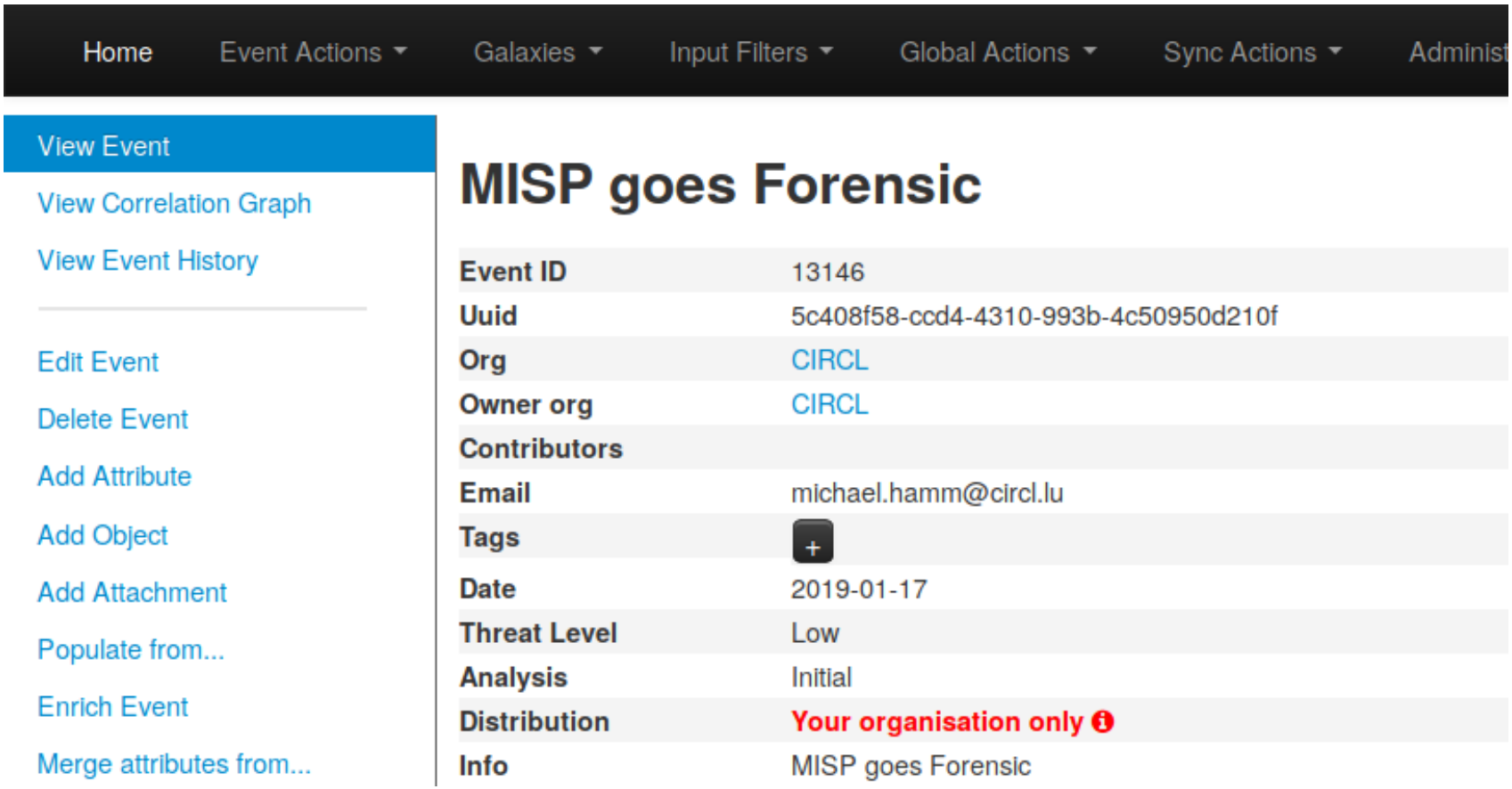
    → Attack on block device still possible
- Try Forensics Linux Distributions:
  - Live Kali 2018_4 in forensic mode
  - SANS SIFT Workstation 3.0
  - DEFT X 8.2 DFIR Toolkit
    - Some distributions do not auto mount

      → Attack on block device still possible
- Kernel Patch: Linux write blocker (not tested)

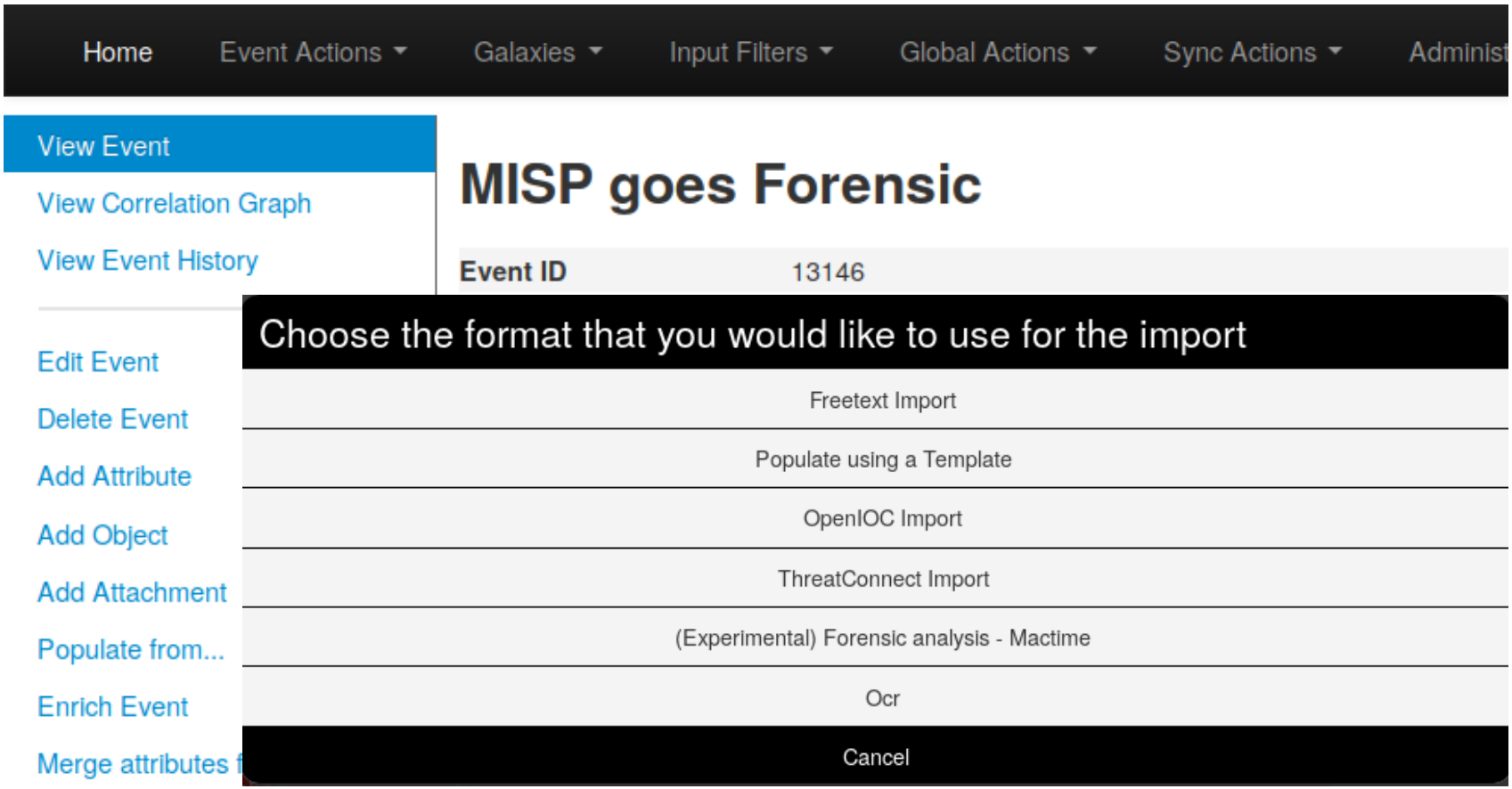
  → `https://github.com/msuhanov/Linux-write-blocker`
- Hardware Write Blocker

  → Effectively block attack

4. MISP goes Forensic



4. MISP goes Forensic

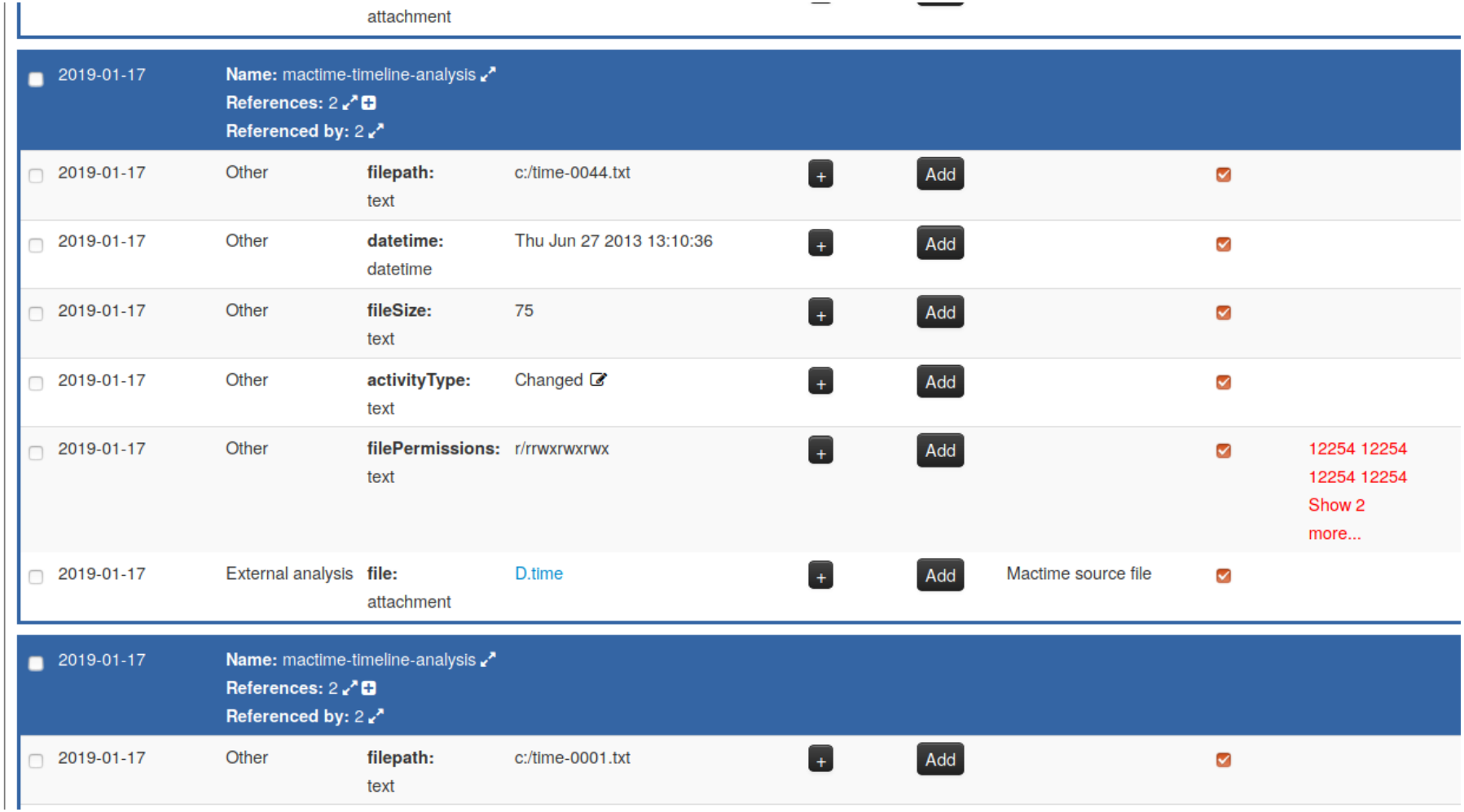


4. MISP goes Forensic

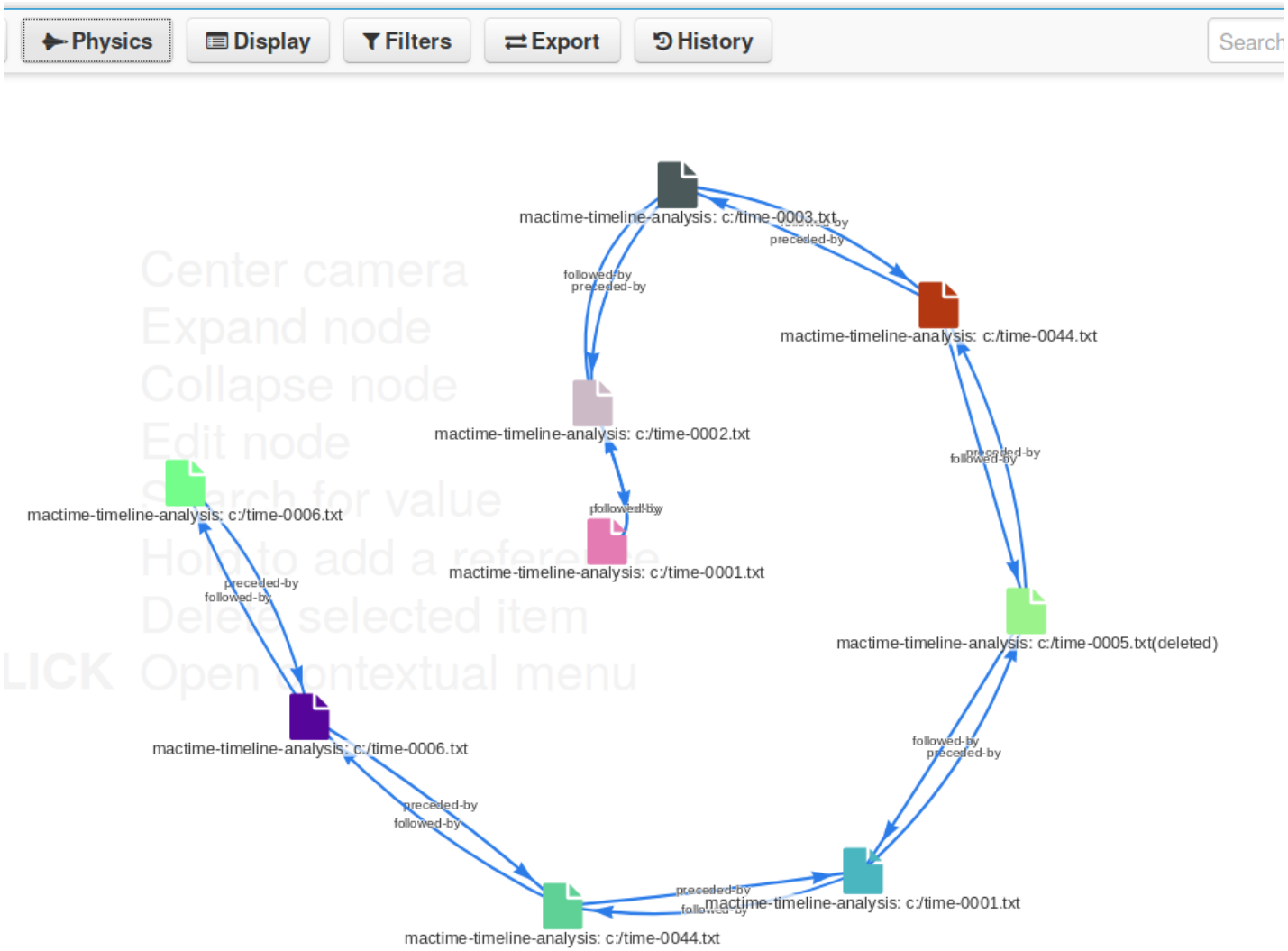
Select text for further analysis

Select	Filepath	File Size	Activity Type
☐	c:/$Extend/$RmMetadata	336	Accessed,Created,Changed,Mo
☐	c:/$Extend/$RmMetadata/$TxfLog/$TxfLog.blf	65536	Accessed,Created
☐	c:/$Extend/$RmMetadata/$TxfLog/$TxfLogContainer00000000000000000002	10485760	Accessed,Created
☐	c:/$MFT	262144	Accessed,Created,Changed,Mo
☑	c:/time-0001.txt	113	Created
☑	c:/time-0002.txt	75	Created,Changed,Modified
☑	c:/time-0003.txt	75	Created,Changed,Modified
☑	c:/time-0044.txt	75	Created,Modified
☑	c:/time-0005.txt(deleted)	75	Accessed,Created,Changed,Mo
☑	c:/time-0001.txt	113	Changed,Modified
☐	c:/time-0003-	75	Accessed,Created,Changed
☑	c:/time-0044.txt	75	Changed
☑	c:/time-0006.txt	20	Modified
☑	c:/time-0006.txt	20	Accessed,Created,Changed
☐	c:/$Extend/$RmMetadata/$TxfLog/$TxfLogContainer00000000000000000002	10485760	Changed,Modified
☐	c:/time-0001.txt	113	Accessed
☐	c:/$Extend/$RmMetadata/$TxfLog/$TxfLog.blf	65536	Changed,Modified

4. MISP goes Forensic



4. MISP goes Forensic



Lessons learned in a Forensics lab

- Hidding data in HPA
- Block device I/O Error
- Modify data on "Read Only" mounted device
- MISP goes Forensic
- Further ideas
  - Recover data from broken ZIP archives
  - ...

Q & A

Thank you