Case Project 3-6

Your supervisor has asked you to list the acquisition tools available on a forensic Linux Live

CD. Download the current ISO version of Deft (www.deftlinux.net), CAINE (www.caine-live.net),

Kali Linux (www.kali.org), or Penguin Sleuth (www.linux-forensics.com), and then create a

bootable CD or DVD of it. Start it on your workstation and survey its tools. Then write a oneto

two-page report containing a brief description of each acquisition utility on the CD or DVD.

Case Project 3-2

At a murder scene, you have started making an image of a computer’s drive. You’re in the

back bedroom of the house, and a small fire has started in the kitchen. If the fire can’t be

extinguished, you have only a few minutes to acquire data from a 10 GB hard disk. Write one

to two pages outlining your options for preserving the data.

Case Project 3-3

You need to acquire an image of a disk on a computer that can’t be removed from the scene,

and you discover that it’s a Linux computer. What are your options for acquiring the image?

Write a brief paper specifying the hardware and software you would use.

Case Project 3-4

A bank has hired your firm to investigate employee fraud. The bank uses four 20 TB

machines on a LAN. You’re permitted to talk to the network administrator, who is familiar

with where the data is stored. What diplomatic strategies should you use? Which acquisition

method should you use? Write a two-page report outlining the problems you expect to

encounter, explaining how to rectify them, and describing your solution. Be sure to address

any customer privacy issues.

Description

Simply turning on the machine can already alter/destroy data that can be potential evidence, let alone booting its original OS.

The best solution is to remove the hard drive, connect it to another computer through a [hardware write blocker](http://www.forensicswiki.org/wiki/Write_Blockers) and then grab a complete image of it with dd or some equivalent. The write blocker is necessary to prevent accidental write to the device that may alter potential evidence (although a basic Linux installation that doesn't mount drives automatically won't do any writes by itself, Mac OS and Windows will definitely do).

If that's not possibly then you need to go with the forensics Live-CD route, but since it involves booting the computer you may already trigger some firmware-embedded code designed to alter/destroy potential evidence (hard but not impossible to do, it depends on what kind of criminal you're after).

Note that in the end, it doesn't matter what OS was installed on the computer since you shouldn't boot that OS anyway, and a bit-by-bit copy doesn't care about the data or even the filesystem that's on the drive (you can copy encrypted drives, but that won't help you much if you don't have the key since the copy is still encrypted).

In a forensic investigation, you need to retrieve data from a possibly hostile system, and you need to provide strong evidence that the data you retrieved is genuine. Acquiring a disk image from the live computer is extremely bad in both respects.

The system software may be some non-standard software that's programmed to lie to you, and provide the “interesting” data only upon presentation of credentials that you don't have. For example, you may be interacting with a virtual machine that masquerades as the real hardware. System executables or the kernel may have been modified to not report genuine data. This is both a problem for you because you want genuine data, and a problem for any legal proceedings because the suspect's lawyer will have a field day showing that you can't prove that what you grabbed was genuine.

The normal procedure for any forensics is to disconnect the storage medium and connect it to your own laptop which is running software that you can precisely account for.

Furthermore, dd is the wrong tool for the job because [it does not reliably copy its complete input](https://unix.stackexchange.com/questions/17295/when-is-dd-suitable-for-copying-data-or-when-are-read-and-write-partial). (If you're reading from a block device and not from a pipe, it might work, but good luck proving that with any measure of confidence.) At least use a proper tool like dcfldd or at least cat.

Not only can you not trust the software running on the computer, but you can't trust the hardware either. So it isn't safe to attempt to boot your own media on the suspect's computer. For example, your OS could end up running in a hypervisor that you can't detect but that returns corrupted data. You need to bring your own computer and connect the hard disk to it.

It's also important that you don't write to the suspect's computer (because you need the suspect's data, not your own data derived from the suspect's). So standard procedure is to connect the hard disk through a write blocker — a dedicated hardware component that enforces at the protocol level (e.g. USB storage protocol) that the disk will not be written to. Then you keep that grab as a reference copy — on physically write-once media if possible, on a hard disk that you never write to otherwise — and never work directly off it, only from secondary copies.

Preferably, grab an image of the whole disk, and reconstruct filesystems, etc. from there. Beware that the mere act of assembling volumes or mounting a filesystem (even read-only!) can modify it: setting a dirty bit, updating a last use date, replaying a journal, … When mounting an ext3/ext4 filesystem which wasn't cleanly unmounted, the noload mount option (in addition to ro) skips the journal update and causes nothing to be written to the disk; this has the downside that the data you'll read may be inconsistent (which can have a silver lining in making freshly deleted files still present).

If you need to get something from the live system, for example because the data is encrypted, then grab what you can the way you can. Document what you did as precisely as possible (taking videos may help). But grab the real underlying data in addition. Use what you got from the live system (typically keys) to reconstruct, in an easily traceable manner, the interesting data from the direct dump.

Case Project 3-5

You’re investigating a case involving a 2 GB drive that you need to copy at the scene. Write

one to two pages describing the three types of acquisitions—physical, logical, and sparse—

you can use to copy the drive accurately. Be sure to include your software and media choices.

Case Project 3-6

Your supervisor has asked you to list the acquisition tools available on a forensic Linux Live

CD. Download the current ISO version of Deft (www.deftlinux.net), CAINE (www.caine-live.net),

Kali Linux (www.kali.org), or Penguin Sleuth (www.linux-forensics.com), and then create a

bootable CD or DVD of it. Start it on your workstation and survey its tools. Then write a oneto

two-page report containing a brief description of each acquisition utility on the CD or DVD.

Copyright 2019

Case Project 3-3

You need to acquire an image of a disk on a computer that can’t be removed from the scene,

and you discover that it’s a Linux computer. What are your options for acquiring the image?

Write a brief paper specifying the hardware and software you would use.

**Description**

In a forensic investigation, you need to retrieve data from a possibly hostile system, and you need to provide strong evidence that the data you retrieved is genuine. Acquiring a disk image from the live computer is extremely bad in both respects.

The system software may be some non-standard software that's programmed to lie to you, and provide the “interesting” data only upon presentation of credentials that you don't have. For example, you may be interacting with a virtual machine that masquerades as the real hardware. System executables or the kernel may have been modified to not report genuine data. This is both a problem for you because you want genuine data, and a problem for any legal proceedings because the suspect's lawyer will have a field day showing that you can't prove that what you grabbed was genuine.

The normal procedure for any forensics is to disconnect the storage medium and connect it to your own laptop which is running software that you can precisely account for.

Furthermore, dd is the wrong tool for the job because it does not reliably copy its complete input. (If you're reading from a block device and not from a pipe, it might work, but good luck proving that with any measure of confidence.) At least use a proper tool like dcfldd or at least cat.

Not only can you not trust the software running on the computer, but you can't trust the hardware either. So it isn't safe to attempt to boot your own media on the suspect's computer. For example, your OS could end up running in a hypervisor that you can't detect but that returns corrupted data. You need to bring your own computer and connect the hard disk to it.

It's also important that you don't write to the suspect's computer (because you need the suspect's data, not your own data derived from the suspect's). So standard procedure is to connect the hard disk through a write blocker — a dedicated hardware component that enforces at the protocol level (e.g. USB storage protocol) that the disk will not be written to. Then you keep that grab as a reference copy — on physically write-once media if possible, on a hard disk that you never write to otherwise — and never work directly off it, only from secondary copies.

Preferably, grab an image of the whole disk, and reconstruct filesystems, etc. from there. Beware that the mere act of assembling volumes or mounting a filesystem (even read-only!) can modify it: setting a dirty bit, updating a last use date, replaying a journal, … When mounting an ext3/ext4 filesystem which wasn't cleanly unmounted, the noload mount option (in addition to ro) skips the journal update and causes nothing to be written to the disk; this has the downside that the data you'll read may be inconsistent (which can have a silver lining in making freshly deleted files still present).

If you need to get something from the live system, for example because the data is encrypted, then grab what you can the way you can. Document what you did as precisely as possible (taking videos may help). But grab the real underlying data in addition. Use what you got from the live system (typically keys) to reconstruct, in an easily traceable manner, the interesting data from the direct dump.

While running a command like

# dd if=/dev/sda of=/path/to/external/medium/file.img

on a live system will work, it's going to result in a number of problems which you won't have if you boot into a separate OS and make the image(s) from there:

1. If you image an entire disk, it probably contains a boot loader and a partition table. Those are going to get in your way when you go to try and do forensics/recovery on the image.

What you really want is to image each filesystem independently:

# dd if=/dev/sda1 of=/path/to/external/medium/filesystem1.img

# dd if=/dev/sda2 of=/path/to/external/medium/filesystem2.img

...etc...

Doing it this way makes mounting the filesystems trivial:

# mount -oloop,ro filesystem1.img /mnt/fs1

(I show the mount done as root because on some Linuxes, [loop devices](http://en.wikipedia.org/wiki/Loop_device) are locked down, so regular users can't use them.)

1. You're snapshotting live, mounted filesystems, so when you mount them later, it's effectively no different than if you had power-cycled the machine. The partitions will be "dirty," which can make mounting them without forensically damaging them difficult.
2. You're using the suspect machine's copy of dd(1). If someone were trying to hide something from you, they could provide a sneaky or malicious copy of dd(1).

Now, all that having been said, there are some *good* reasons to do an on-line clone. The best reason is that the system is using some form of filesystem or whole-disk encryption, and rebooting it will erase the decryption keys for the mounted volumes.

dd is not the right tool for this job, however, since that will just get you a copy of the encrypted-at-rest data. A regular backup is a better idea. E.g.,

# tar -cvJf --exclude={'/proc/\*','/sys/\*','/tmp/\*'} \

/path/to/external/medium/everything.tar.xz /

That won't discover hidden partitions and such, but at least it will force the OS to decrypt every file accessible directly from the root of the filesystem.