

Quality Control and reporting on a disk image

DAY 2 →

What is Quality Control?

A process that verifies the quality, accuracy and consistency of digital files or data

Suggested Steps in Quality Control Workflow

- Regularly test, use and upgrade equipment and software for QC
- Design your workflow around 2 step-process
 - creation
 - verification
- Comparison to source material
- Fixity check
- Metadata; check that it exists and is stored in the correct format and location, that metadata is accurate, complete and valid
- Verify that technical information matches standards for the project (file format, file naming, sidecar files, etc)

An important part of any workflow is to build QC into a system

Equipment/System evaluations

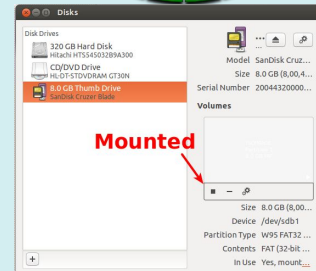
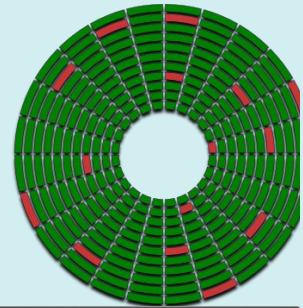
-scheduled service and calibrating protocols according to manufacturer recommendations, swapping out gear

Metadata

-embedded metadata guidelines

Quality Control of Disk Images

1. Checking for bad sectors
2. “Verifying” the image against source media
3. Mounting the image, checking partitions, exporting files
4. Metadata extraction

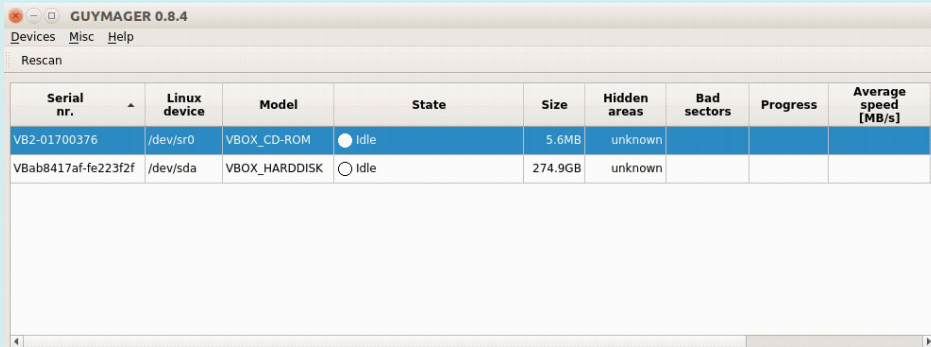


When thinking about a Quality Control workflow for disk images, here some points we thought were important to consider:

1. Bad sectors (image on top right) - a bad sector is a sector that can't be read due to physical causes like deterioration of the drive or a malfunction/failing hard drive. -if you get bad sectors try with a different drive, if you get the same bad sectors with different drives it's probably your media - isolate your variables, a la scientific method.
2. Verifying the image. If the image is supposed to be a bit-for-bit copy we have to have some way of making sure that it actually is. More on that in a second
3. Have to make sure the file is usable. Mounting the image means opening it up on another computer and taking a look at what's inside, just like when you connect an external hard drive to your computer and can browse the files and folders. We'll discuss in depth
4. Extract metadata to corroborate a successful imaging process and that it matches your source file

Initial Steps– verify source media

```
$ md5 /dev/disk1
```



The screenshot shows the GUYMAGER 0.8.4 application window. It has a menu bar with 'Devices', 'Misc', and 'Help'. Below the menu bar is a 'Rescan' button. The main area contains a table with the following data:

Serial nr.	Linux device	Model	State	Size	Hidden areas	Bad sectors	Progress	Average speed [MB/s]
VB2-01700376	/dev/sr0	VBOX_CD-ROM	● Idle	5.6MB	unknown			
VBab8417af-fe223f2f	/dev/sda	VBOX_HARDDISK	○ Idle	274.9GB	unknown			

when we talk about verification, we mean that the image matches the source volume. This can be done by re-hashing the source volume and comparing it to the image's hash. I would recommend doing this process - comparing the checksum of the image to the checksum of the original volume - manually if the application you are using doesn't automate this process.

The Tableau Imager software and FTK Imager do not automate this process. Keep in mind that a checksum mismatch is not a very granular measurement of difference. A difference of one or two bytes in a 500 GB disk image is in most cases negligible, but that's enough to change the checksum.

Just as you can create a checksum of a file, you can create a checksum of a mounted volume. When a drive is mounted to your OS, it is assigned an identifier, so in this example "disk1." You can create a checksum of the entire volume, just as you would with a file, from the command line. OR you can automate it. The only application that I'm aware of that automatically verifies the source volume and the disk image are identical is guymager.

This is a little in the weeds, but FTK imager has a verification option in its GUI, which simply verifies the contents of the disk image. It creates a checksum during the process of disk imaging, and then verifies that the resulting disk image is the same as the data read during acquisition. This protects against an error in writing the disk image. BUT if the data was misread, then FTK's verification process doesn't help you. Guymager re-reads the source and compares that with the disk image to assure that

there wasn't an error in the reading OR writing process.

Initial Steps– verifying source and disk image

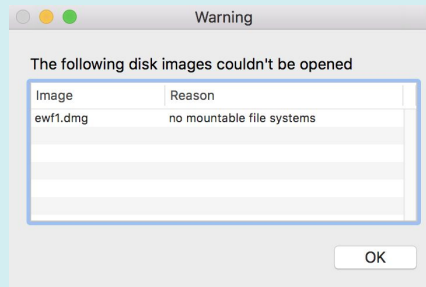
- Create a duplicate copy of the disk image for the purpose of testing in a staging area
- Verify that the duplicate copy is exactly the same as the source image by using the diff command
 - `$ diff Source1 Source2`
- If the image was not created in Guymager (which automatically runs this process), you can also run the following commands to verify the integrity of the image against the original source media:
 - `$ md5sum mydiskimage.E01`
 - `$ md5sum [device ID of original media such as /dev/sr0]`

You may also use the diff (stands for difference) command by creating a duplicate copy of the disk image for the purpose of testing and locate it on a staging area

Type: `diff source 1 source 2` on terminal to compare both images, This command is used to display the differences in the files by comparing the files line by line.

You can also run a md5 checksum on the newly created disk image and compare that with the source md5 checksum

Mounting Disk Images– libewf on macOS



The following is the procedures taken in an attempt to install libewf on a Mac OS High Sierra 10.13

Update August 2019

New instructions from Twitter [thread](#)

1. Open text editor (I'm using Sublime Text) and paste code from this [formula](#)

2. On line 38 of the code, edit line to "--with-libfuse=yes"

3. Save file locally as libewf.rb

3. brew install --HEAD [locally saved libewf.rb]

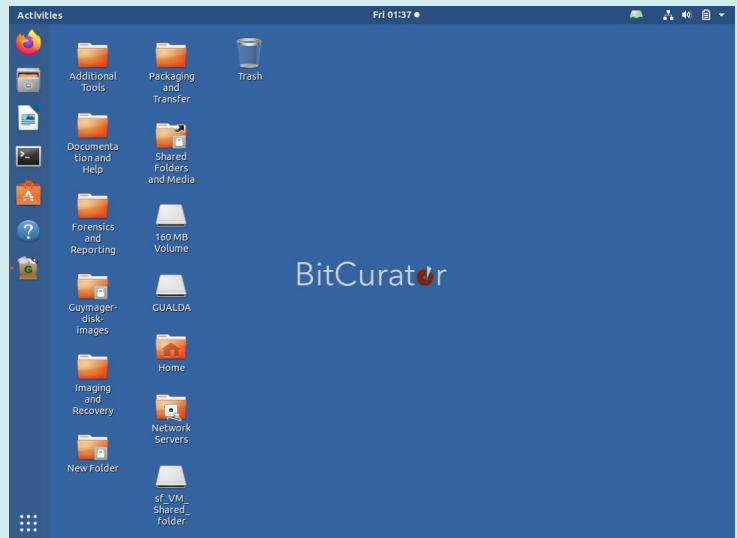
4. now you can use the command "ewfmount -X allow_other"

Another way to verify a newly created disk image is to mount it and peruse through the contents of the image and independently examine any artwork related files to make sure that the disk image file you created is usable.

Even though a disk image is not a physical disk, it can be mounted in the same way. "mounting" here refers to allowing an image to be accessible for reading and writing on a computer.

Sometimes you'll run into issues like the one on this screen, which can be solved with the assistance of open-source tool devs and colleagues.

BitCurator Environment



Directories:

Forensics and reporting contains launchers for forensics tools, disk and file system analysis tools, and report generation utilities

Imaging and recovery folder contains tools to create raw and forensically packaged disk images from physical media

Packaging and transfer folder points to transfer and accession tools including bagger, python- bagit, grsync (gui rsync)

Additional tools- includes other useful software tools that may be used to inspect and process disk images and files

You can create a folder on the desktop as your destination folder for disk images,

Read-only mode- mounting files in read only mode via a loop device, a pseudo device used to enforce the read-only mount, because of this loop device the contents of the mount will not appear on the files window itself. You must double click the disk icon that appears on the desktop to browse any available file system. no eject button is present for this type of mount, right-click on the disk and select 'unmount to unmount;

libewf on BitCurator

The **libewf** package contains the following tools:

- * **ewfacquire**; which writes storage media data from devices and files to EWF files.
- * **ewfexport**; which exports storage media data in EWF files to (split) RAW format or a specific version of EWF files.
- * **ewfinfo**; which shows the metadata in EWF files.
- * **ewfmount**; which FUSE mounts EWF files.
- * **ewfrecover**; special variant of ewfexport to create a new set of EWF files from a corrupt set.
- * **ewfverify**; which verifies the storage media data in EWF files.

```
ewfmount 20140608
Missing EWF image file(s).
Use ewfmount to mount the EWF format (Expert Witness
Compression Format)

Usage: ewfmount [ -f format ] [ -X extended_options ] [ -hvV ]
      ewf_files mount_point

ewf_files:  the first or the entire set of EWF segment files
mount_point: the directory to serve as mount point

-f:        specify the input format, options: raw (default),
           files (restricted to logical volume files)
-h:        shows this help
-v:        verbose output to stderr
           ewfmount will remain running in the foreground
-V:        print version
-X:        extended options to pass to sub system
```

EWF disk images can't be mounted directly on most OSs. They need special tools. You can either extract a raw disk image from the EWF or mount it directly using an libewf.

Obviously, needing a special tool to mount EWF disk images is a disadvantage, but since this tool is open source and fairly widely adopted, I think it is kind of a low risk. You could equate it to the risks inherent to ffv1 mkv video files. There aren't too many players or editing software that can interact with the file type, but given the compression and built in fixity checks it contains, I'm willing to accept that trade off.

Once libewf is installed it's fairly straightforward to use.

Mounting Disk Images

- Mount as “read only” in BitCurator
- Mount with the command
“mount -o loop,ro /path/to/disk.img /mountpoint/”
- Create a “Work Copy” and mount that one instead.
 - Need change the file extension of the disk image to something the OS will recognize, like “.iso”



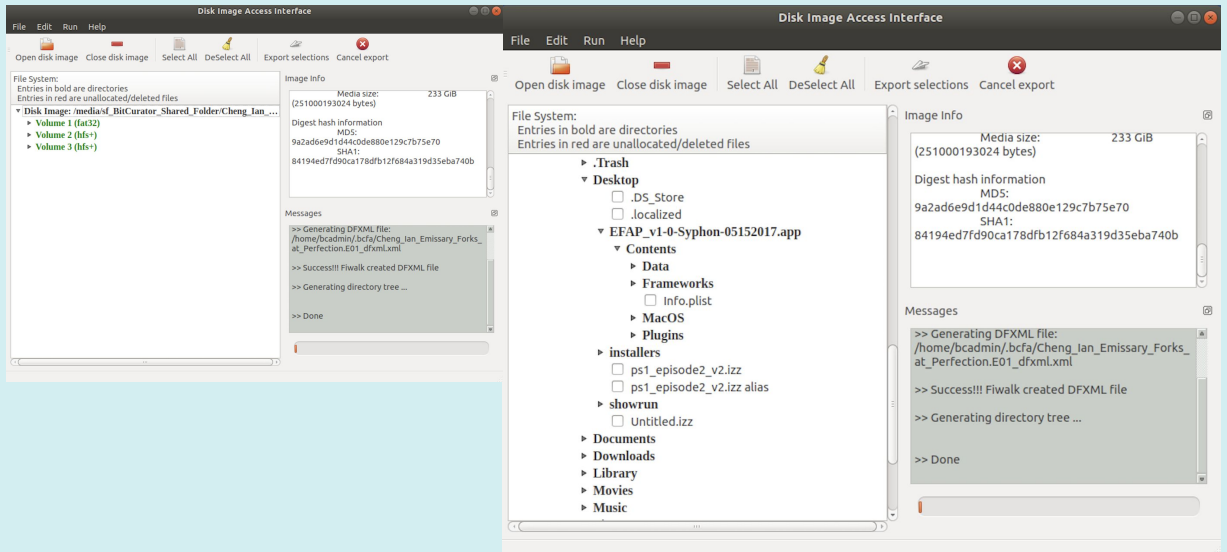
Created by Philip Sheffield
from Noun Project

A more straightforward way...

Safety first! Mounting a disk image can expose it to risk if the mounted volume is not write protected. BitCurator allows you to mount a disk image as read only by simply right clicking on the disk image and selecting that option. here's also a Linux command line tool that allows for read only mounting. I feel it would be remiss to not mention it, but it's a really pain in the ass.

For raw disk images, I've been doing the last option. I've created a copy of the original disk image, in order to ensure that I don't alter that copy, and then I make a “work copy” and I mount that one in the macOS. The macOS won't see a disk image with a .dd or .raw file extension as a disk image, so you just change the extension, and it will mount.

Disk Image Access in BitCurator



BitCurator Disk Image Access: A GUI interface to browse raw and forensically-packaged disk images, export files and deleted items, and view disk image metadata.

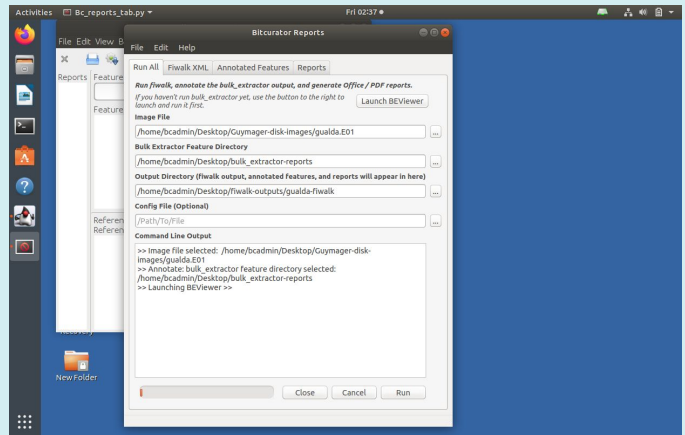
Located in the forensics and reporting directory

Disk Image Access is a tool that is specific to BitCurator. It's a GUI interface to browse raw and forensically-packaged disk images, export files and deleted items, and view disk image metadata.

It's essentially a way of exploring the file system of a disk image without having to mount it. So it's a lower risk way of looking at the contents of a disk image. I'll admit that I've had mixed results with this tool, it will fail on me sometimes when some of the other methods I have described here work fine, BUT, when it does work, it will show you all of the file systems and partitions inside of a disk image, all of the directories, and all of the files inside of those directories are, and lets you select specific files or directories to export from the disk image into a new directory.

BitCurator Reporting Tools

- **features (directory):** The annotated features generated by bulk extractor;
- **bc_format_bargraph.pdf (file):** Histogram showing file formats;
- **bulk_extractor_report.pdf (file):** High-level overview of bulk extractor feature locations on disk;
- **fiwalk_deleted_files.pdf (file):** File documenting paths to any deleted materials found in a given partition;
- **fiwalk-output.xml.xlsx (file):** DFXML output (file system metadata) converted to an Excel spreadsheet;
- **fiwalk_report.pdf (file):** High-level overview of file system characteristics;
- **format_table.pdf (file):** Long-form file format names for formats shown in **bc_format_bargraph.pdf**;
- **premis.xml (file):** PREMIS preservation metadata.



BitCurator gives you the option to run a number of different reporting tools against their disk images. Although each of these tools can be run individually, users may alternatively use the "Run All" tab of the **BitCurator Reporting Tool** in order to simultaneously execute [fiwalk](#), the [annotated features report](#), and the [BitCurator forensic reports](#).

- **features (directory):** The annotated features generated by bulk extractor;
- **bc_format_bargraph.pdf (file):** Histogram showing file formats;
- **bulk_extractor_report.pdf (file):** High-level overview of bulk extractor feature locations on disk;
- **fiwalk_deleted_files.pdf (file):** File documenting paths to any deleted materials found in a given partition;
- **fiwalk-output.xml.xlsx (file):** DFXML output (file system metadata) converted to an Excel spreadsheet;
- **fiwalk_report.pdf (file):** High-level overview of file system characteristics;
- **format_table.pdf (file):** Long-form file format names for formats shown in **bc_format_bargraph.pdf**;
- **premis.xml (file):** PREMIS preservation metadata.

BC Reporting tool in the forensics and reporting directory

The run all tab will allow to carve the raw disk contents for features of interest, generate a dFXML listing of the file system hierarchy, match features to files within the file system and generate high-level reports. Click on Launch BEViewer to run

bulk_extractor before proceeding

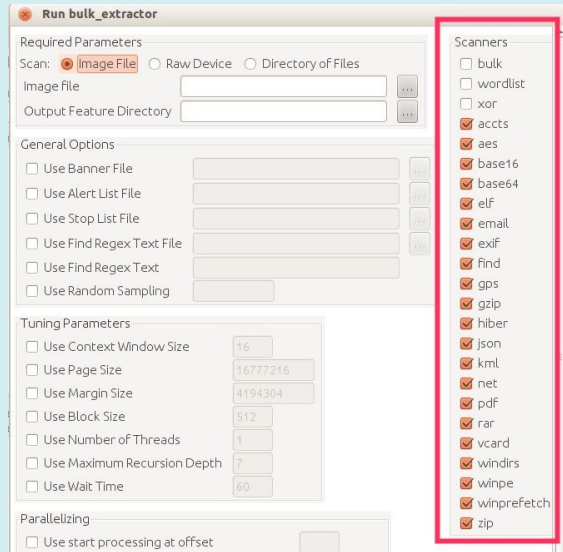
viewer is the graphical front end to bulk_extractor. together these tools allow you to identify various features of interest contained within the bitstream extracted from source media such as ssn, email addresses, exif metadata, and others

You may run that first and then the 'run all' selection on bc reports or choose to run different reports individually on their respective tab

Bulk_extractor

bulk_extractor is a program that extracts features such as email addresses, credit card numbers, URLs, and other types of information from disk image files, file directories or files.

- It finds email addresses, URLs and credit card numbers that other tools miss because it can process compressed data (like ZIP, PDF and GZIP files) and incomplete or partially corrupted data. It can carve JPEGs, office documents and other kinds of files out of fragments of compressed data. It will detect and carve encrypted RAR files.
- It builds word lists based on all of the words found within the data, even those in compressed files that are in unallocated space. Those word lists can be useful for password cracking.
- It is multi-threaded; running bulk_extractor on a computer with twice the number of cores typically makes it complete a run in half the time.
- It creates histograms showing the most common email addresses, URLs, domains, search terms and other kinds of information on the drive.



Bulk_extractor, developed by Simson Garfinkel,⁴¹ is a forensics tool that can scan a disk image and look for particular types of content: strings that match a certain format or type, and metadata associated with certain image files and compressed files. It is very effective at identifying, for example, an email address, URL, or any specified string. It is not, however, as effective at helping an archivist evaluate a video or image, nor is it meant to act as a substitute for thoughtful and considerate evaluation of content. For example, according to bulk_extractor, the contents of a file might be benign due to absence of credit card numbers and Social Security Numbers (SSNs), but that doesn't mean those contents would not be harmful to its creator or others if kept. Bulk_extractor is designed to be an identification tool, not a redaction or deletion tool. Its purpose is to help find information, not remove it.

Virtualization & Emulation

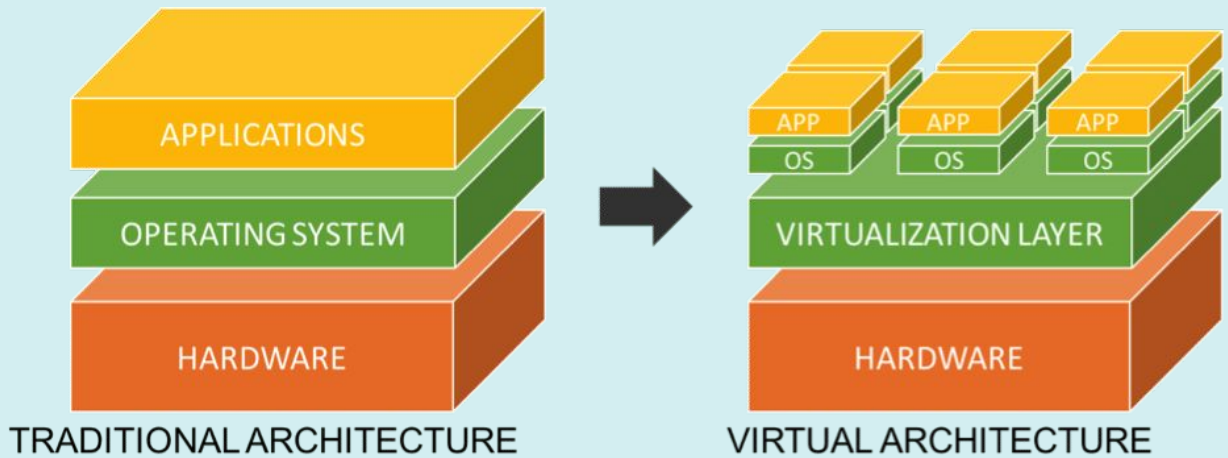


Image source: *11 Points to Consider When Virtualizing Security*
from resources.infosecinstitute.com

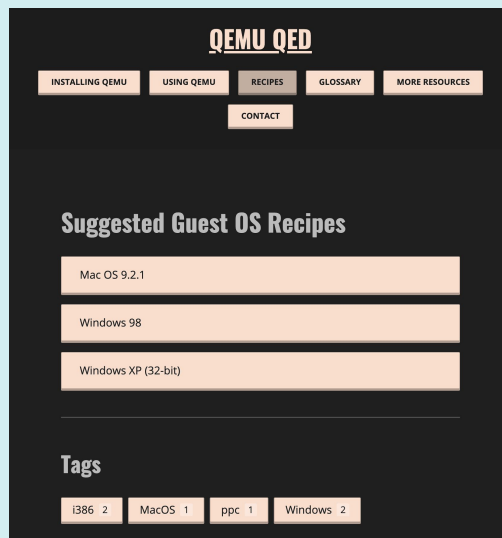
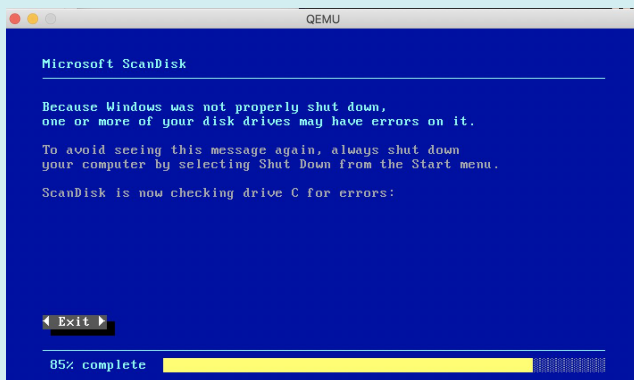
Virtualization and emulation, the process of running an operating system or other software within an isolated virtual environment, is one strategy for providing access to software based artworks, particularly those that are facing obsolescence. I imagine all of you are familiar with virtual machines, as BitCurator is distributed as is a virtual machine.

Practices for exhibiting a software-based artwork using a disk image are still in development. However, the exhibition of a work represents a key opportunity for research, developing more robust documentation and creating a deeper understanding of the work's needs. Disk imaging can be a powerful tool when conducting such research.

Disk images can also be used to create a new exhibition copy of the original device on a new machine. This process, referred to as creating a *replica* computer, was adopted by MoMA while exhibiting the *Long March: Restart* (2008), by Feng Mengbo (b. 1966) (Lewis and Fino-Radin 2015). Creating a replica computer can be an effective strategy for mitigating the risk of hardware failure, but it also allows an opportunity to evaluate the significance of certain properties of the original machine. If the replica does not contain the exact same

components or the exact same data as the original computer but still faithfully reproduces the work, it is still an effective replica. An ineffective replica—one that lacks certain data or components and does not faithfully reproduce the work—may still be instructive, helping conservators to understand what components are essential to the work.

A note about QEMU



A bit out of scope for this workshop... but wanted to briefly talk about qemu, why would you consider it, and point you to an excellent resource qemu-qed (a work in progress)...

QEMU QED began as a project during the iPRES 2019 conference, as part of the “Reading the Matrix: A Hackathon Linking Digital Forensics to User Access”, sponsored by the teams behind [EaaSI](#) and [BitCurator](#). it is based on [ffmprovisr](#) with code and content contributions from Ashley Blewer and Nick Krabbenhoeft.

QEMU is a command line application that is primarily used by developers and IT professionals for testing software in different environments, creating virtual environments on servers, and more.

There are many reasons why you would consider incorporating emulation, the chief reason among them is to apply emulation techniques as a preservation strategy. Emulation can remove an software based artwork’s dependency on ephemeral physical computer hardware, which we know is prone to obsolesce quickly. We can also use emulation as a preventive conservation strategy, when we create a backup of an artwork’s computer we can generate a safe copy from which to work from. Through actions like these, you can minimize the depreciation of equipment during an extensive exhibition schedule/.

Through emulation you may also replace technical components that are at risk of failure or that cannot be easily reinstated. Finally, you may use emulation as part of a workflow for disk imaging quality assurance. By running an application in a virtual

environment side by side with the non-emulated version of the work, you can certify that a disk image that you took on a computer (for example) is usable in the future.