498.4
Non-Traditional and Cloud Acquisition

FOR498.4 | Battlefield Forensics & Data Acquisition

Non-Traditional and Cloud Acquisition

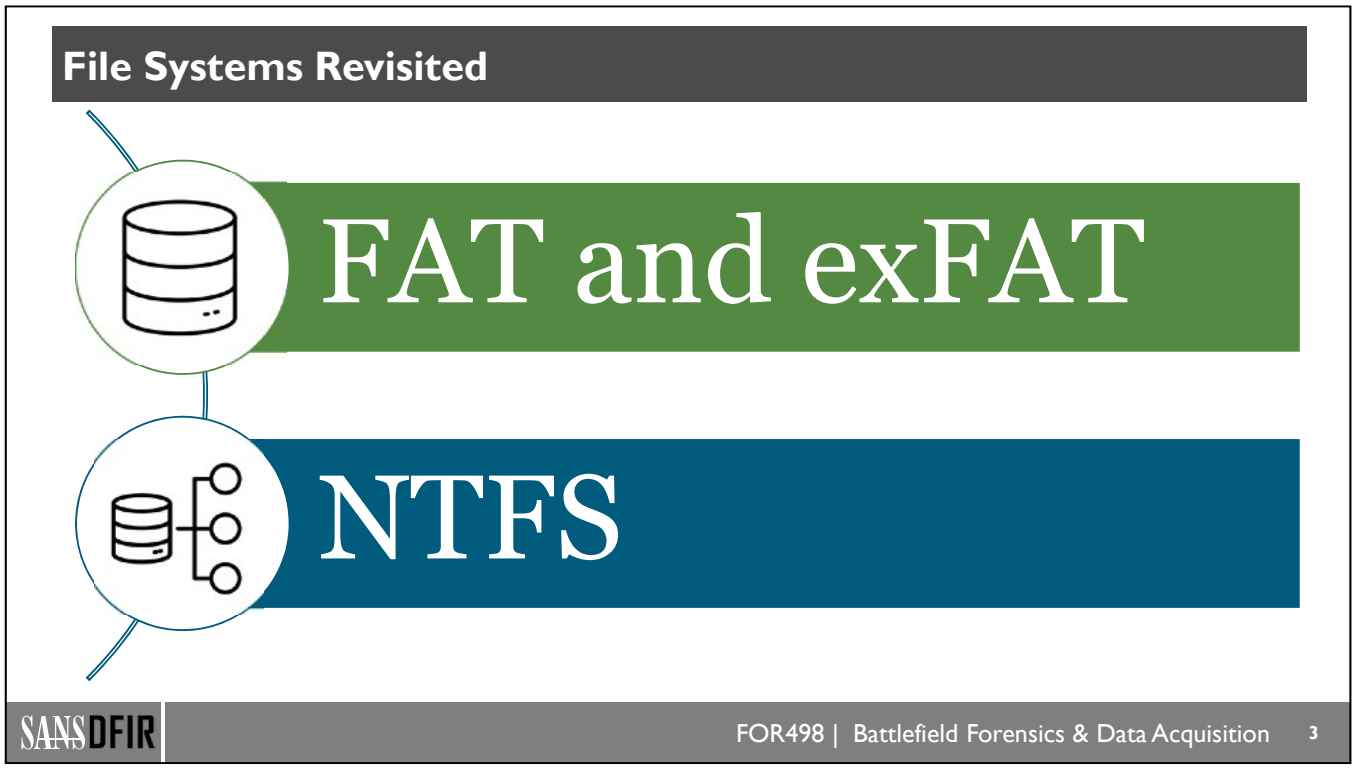
Authors:
Eric Zimmerman – saericzimmerman@gmail.com
Kevin Ripa – kevin.ripa@gmail.com

https://twitter.com/ericrzimmerman
https://twitter.com/kevinripa

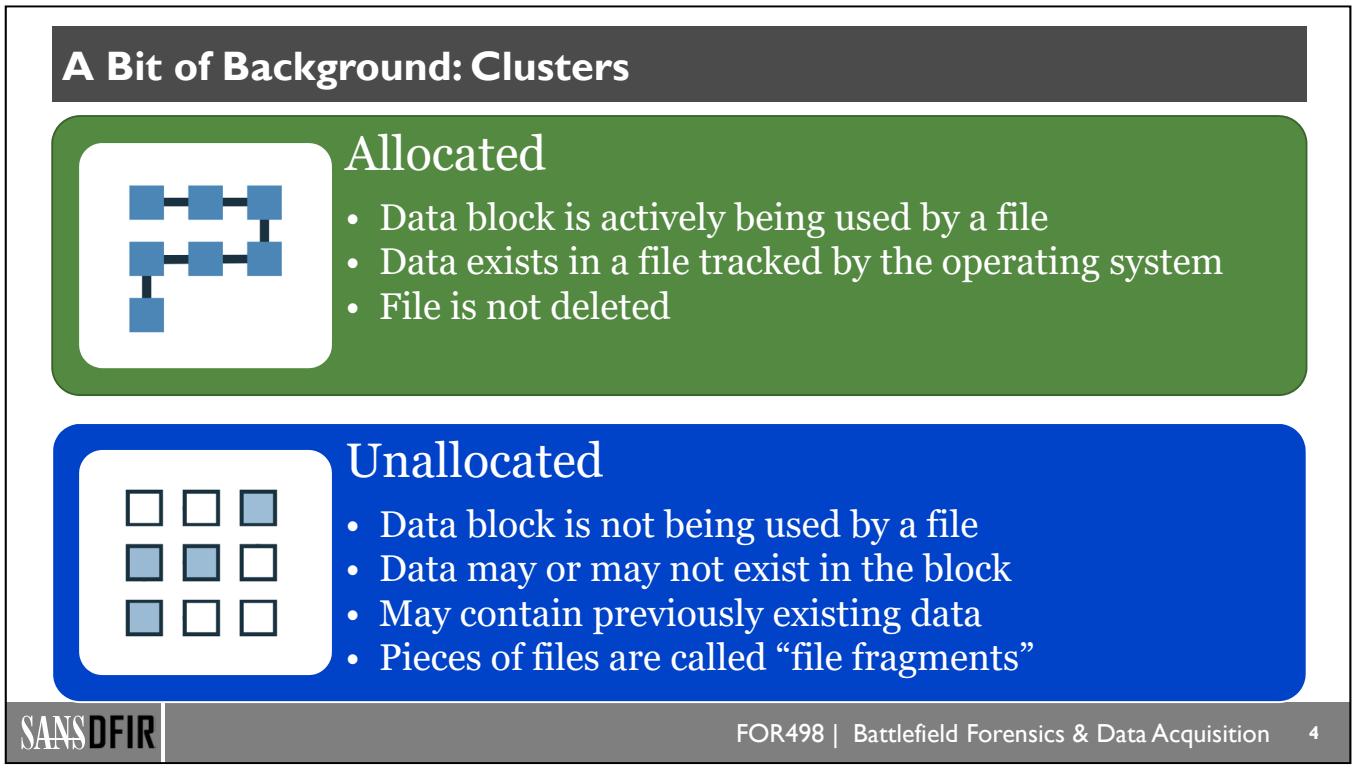
FOR498.4: Non-Traditional & Cloud Acquisition Agenda

- 4.1 File Systems Revisited
- 4.2 Battlefield Forensics with KAPE
- 4.3 Multi-Drive Storage
- 4.4 Remote Acquisition

This page intentionally left blank.



This page intentionally left blank.



At the lowest level, a storage device is made up of sectors. When a drive is partitioned and formatted however, groups of sectors are organized into clusters. When formatting a drive, one of the parameters selected is the cluster size. The cluster size determines how many sectors will be assigned to a cluster. For example, if sectors are 512 bytes and a cluster is 4096 bytes, there will be 8 sectors in a single cluster. Once a drive is formatted and arranged into clusters, the cluster becomes the smallest addressable unit of storage from the operating system's perspective. The operating system needs a way to track each cluster and its state.

There are two states that a cluster can have: allocated, or unallocated. By this we mean that a cluster will either be assigned to a file being tracked by the file system, or it is available for the operating system to store data as needed. When a cluster is not being used by a file, it is sometimes referred to as "free space", but this does not mean that all free space is empty.

When a file is deleted, the operating system simply marks the clusters that were being used as free again, but the data sitting in those clusters is still the same as it was before the file was deleted. In this way, data persists after file deletion unless special care is taken to delete the data before file deletion. Consider the case where a file was using three clusters and was deleted. At this point the data is intact in the three clusters. But if the operating system created a new file and used the first cluster of the three, this cluster would be overwritten with the data from the new file. The second and third clusters however would remain unaffected. Since the first cluster is no longer available, a full recovery of the file that used to exist is no longer possible. With that said, it is still possible to recover a fragment of the original file as it exists in clusters two and three. Most operating systems try to keep file clusters contiguous (that is, one right after the other, without gaps), but if a file isn't stored this way, it is considered fragmented. This situation makes finding file fragments more difficult, especially if the structure tracking the original three cluster file has been reused by another file.

Windows writes file information in sector sized-chunks. For example, if a file is 1280 bytes in length and the cluster size is 2048 bytes, Windows will write data into the first three sectors (a sector in this case being 512 bytes, so three of them equals 1536 bytes). Because the three sectors contain more bytes than are required to save the file, the third sector would only be partially written. When this occurs, Windows uses the null byte

(\x00) as a filler until the end of the **sector**. Any extra sectors in the cluster not used in writing data for the file are then considered slack space. Slack space could contain data from the previous file that was stored at that location.

Other operating systems handle slack space in different ways. For example, slack space still exists on Unix-based filesystems, but slack space is overwritten to the end of the block with the null byte. This means that naturally occurring data in slack space is rare. However, data hiding tools, such as BMAP, might utilize slack space in blocks to hide files.

A Bit of Background: Deleting a File

Recovery is (usually) possible!

As we just saw, when a file is deleted, the contents are not overwritten immediately. Deleting a file normally is like throwing away a piece of paper in a garbage can. Anyone that looks inside the can would be able to locate and recover the file by simply taking it out of the garbage can.

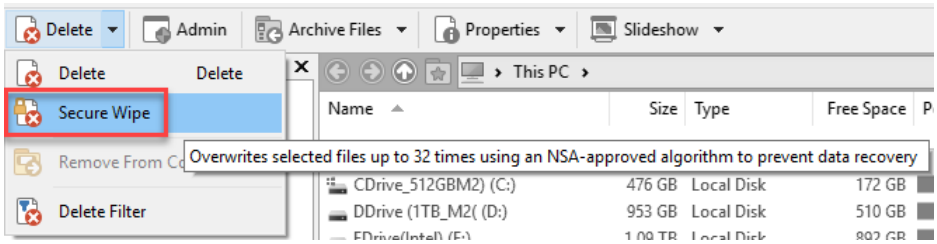
When a file on a computer is deleted, the data still exists on disk and can be recovered until the free space is reallocated by the operating system and overwritten. This involves several techniques such as file carving, examining file system structures to look for recoverable files, and so on.

It may seem that the clusters that once held a file's contents would be quickly used again, decreasing our ability to recover deleted files, but this is not necessarily the case. In fact, because of the way operating systems allocate and use free space, this sort of thing happens infrequently.

The bottom line is that we can usually recover most (if not all) of a deleted file. Studies have been done that indicate that roughly 80% of the time you will be able to recover deleted files unless they have been deleted using a tool purpose built to wipe files, making them unrecoverable.

So how does deleting a file differ from wiping a file? We will look at that scenario next.

A Bit of Background: Wiping a File



Wiping a file is synonymous with shredding a paper file...

...then setting it on fire!

So how does a wiping tool work in comparison to file deletion? When a wiping tool is used to delete a file, the first thing it does BEFORE deleting the file, is to overwrite the data in the file one or more times. Once the data has been overwritten; the file is deleted like we saw in our previous example. The total number of times the data is overwritten is usually configurable, but in general, overwriting the data one time is enough. Some tools also allow a pattern (or even a changing pattern), to be defined to use when overwriting data.

One example of such a tool is srm[1]. srm is a secure replacement for rm, the standard deletion program on Unix/Linux systems. srm defeats recovery techniques by wiping (or overwriting) the contents of a file prior to the file being deleted. On the Windows side (and seen in the screen shot above), tools like Directory Opus allow for secure deletion of files [2].

As we mentioned earlier, only one wipe is necessary to stop any forensic tool from being able to recover the data that once existed. The NIST guideline, Guidelines for Media Sanitization (September 2006), states that "Studies have shown that most of today's media can be effectively cleared by one overwrite. [3]" So while many tools offer overwriting with more than a single pass, this will have diminishing returns and in most cases, just make the wiping process take longer.

One final thought to consider is how to handle sensitive vs non-sensitive data. For non-sensitive data, wiping the drives one or more times is enough to prevent recovery, but for more sensitive data, it is recommended to destroy the drives themselves to fully prevent data recovery.

[1] srm - secure file deletion for posix systems | `https://for498.com/2d-co`
[2] Directory Opus | `https://for498.com/qacil`
[3] Guidelines for Media Sanitization | `https://for498.com/9ah4m`

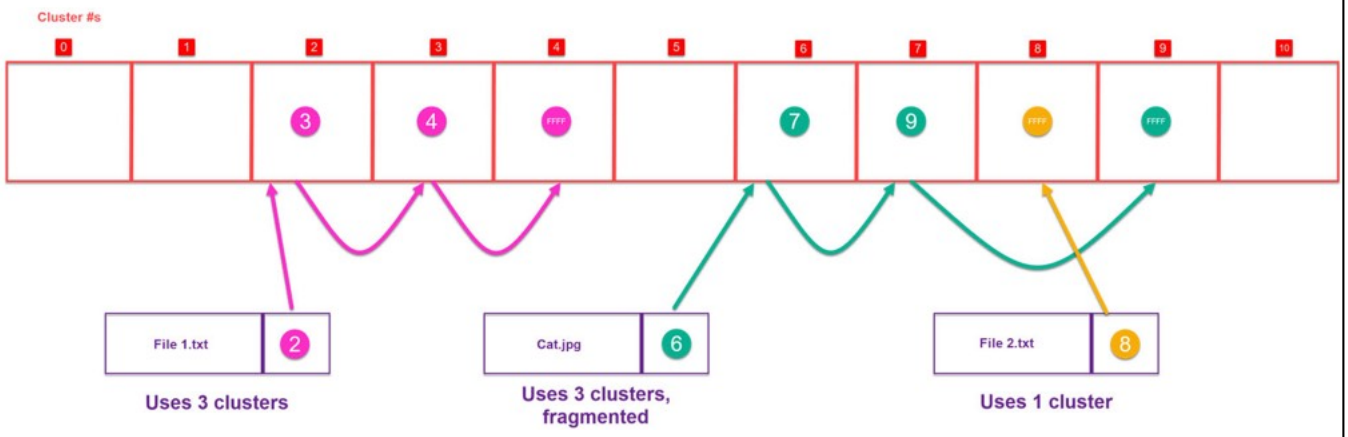
File Systems Capability Overview

File system	Max file size	Max volume size	Max files per volume	Compatibility	Security
FAT	4 GB (FAT32) 2 GB (FAT16)	32 GB (FAT32) 4 GB (FAT16)	4,177,920 65,536	Just about everywhere	None
exFAT	Almost as large as volume	16 TB	4,294,967,285	Most operating systems, some devices	None
NTFS	~16 TB	256 TB	4,294,967,295	Windows, with varying support on Mac and Linux	Robust

All three file systems support long filenames

This page intentionally left blank.

FAT/exFAT: How File Data Is Stored



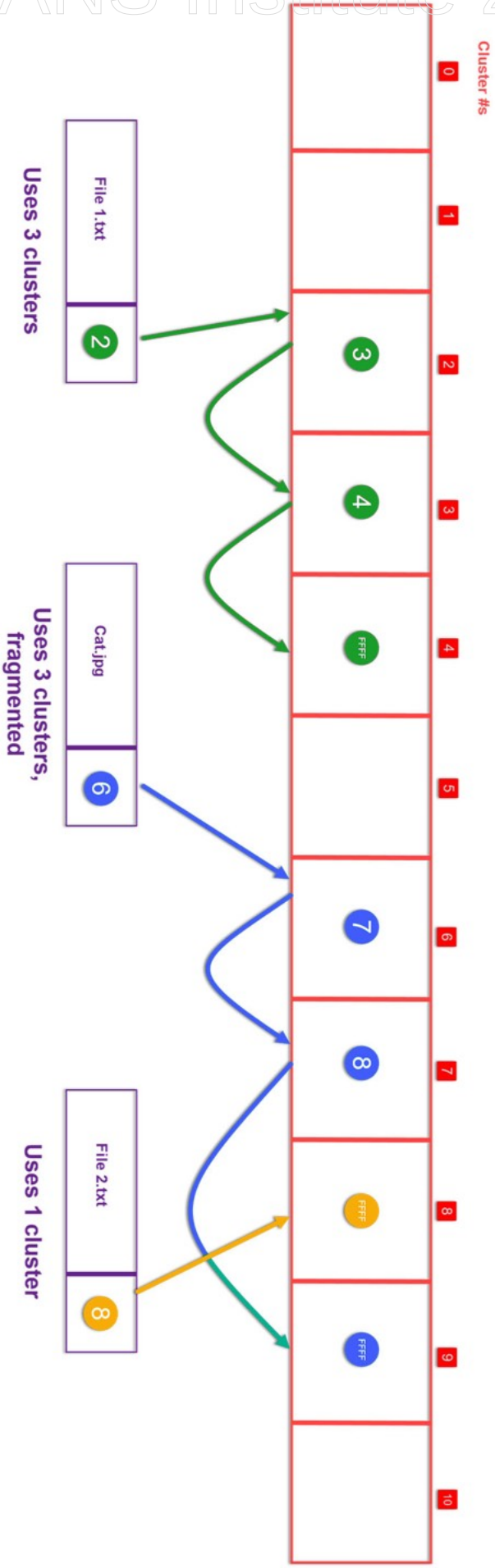
FAT and exFAT use the concept of a File Allocation Table (FAT)[1], which is a table that provides a "map" of the clusters a file has been stored in. Windows creates a FAT entry for a new file that records where each cluster is located and its sequential order. When you read a file, the operating system reassembles the file from clusters and makes it available when you want to read it. For example, a long Word document or video file may very well be stored in more than one cluster on your hard disk.

To illustrate this, three example files are described above, with "File 1.txt" and "Cat.jpg" taking up three clusters. "Cat.jpg" however, is fragmented, because its cluster numbers are not all in a row (cluster 8 is in use by "File 2.txt").

Each file is stored in a chain of clusters. For example, for "File 1.txt" the starting cluster is 2. This is the first chunk of the file. Looking at the map at position 2, we see a 3, which means the next chunk of the file is in cluster 3. Looking at the map at position 3, we see a 4, which means the next chunk of the file is in cluster 4. In map position 4 however, we see FFFF, which means that the last chunk of the file is in cluster 4 and there are no additional clusters to follow. The FFFF essentially serves as an indicator that we are at the last cluster of the file.

A primary difference between FAT and exFAT is the size of addressable entries that exist in the FAT for each file system. Because exFAT can address so many more clusters, it can store much larger files when compared to FAT (in any of its variations).

[1] What is File Allocation Table (FAT) | `https://for498.com/nptgy`



FAT/exFAT Timestamps

File system	Time zone	Time Resolution	Modified	Accessed	Born (Created)
FAT	Local	January 1, 1980	Modified (Rounded to even second)	Accessed (no time stored, only the date)	Created (accurate to 10ms)
exFAT	UTC	10 ms intervals since January 1, 1601	Modified	Accessed	Created

The chart above summarizes the differences between timestamps on a FAT and exFAT volume. There are a few peculiar things to notice when it comes to the FAT timestamps, but the most peculiar is the differences in granularity between the created, modified, and last accessed timestamps.

Notice the accessed timestamp does not even store a time at all! Internally, the only data that is stored is for the date the file was last accessed. The time then is inferred as "0" since it was not supplied. Therefore you will see Windows show last accessed timestamps on FAT volumes as 12 midnight for the last accessed column in File Explorer. Things get a little better for the modified timestamp, but it too makes compromises in that it can only store the last even second of the modified timestamp. Finally, the most accurate timestamp is the created timestamp, which is accurate to 10 milliseconds.

But why is it this way? If you think about the time when FAT was invented, hard drive storage was very expensive. With more modern file systems, each one may occupy 8 bytes each, but not so with FAT. The developers were given a certain number of bytes to store all the timestamps and had to compromise in order to work with what they had.

exFAT, being a more modern file system, corrects the mistakes that FAT made in regard to timestamps, in that it does store a much more granular and accurate timestamp (including milliseconds) for each of the three timestamps. In this way, it is related to NTFS in that the beginning of time according to NTFS is also January 1, 1601, but NTFS has an even better resolution of a millionth of a second.

FAT/exFAT: What Still Exists on Deletion

FAT	exFAT
Filename preserved, but first character overwritten with 0xE5	Entire filename is preserved
Created, modified, last accessed times preserved	Created, modified, last accessed times preserved
Starting cluster and size preserved	Starting cluster and size preserved
Clusters used for storage marked as free, but data unchanged	Clusters used for storage marked as free, but data unchanged

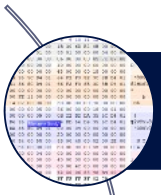
When a file is deleted on a FAT or exFAT volume, the file allocation data structures that keep track of files are adjusted to show that the file has been deleted. The underlying data within the data structures is not changed until a new file is created which reuses the file allocation entry. Because of this, it is possible to recover nearly all the information about a deleted file as it existed before the file was deleted. For the most part, both file systems do the same thing (albeit in slightly different ways under the hood) when it comes to deleting files.

Notice however, that when a file is deleted on a FAT volume, the first character of the filename is overwritten with a new character. In many cases you can use the rest of the filename to determine what the overwritten character is, but this may or may not be useful.

Since the timestamps, starting clusters, and other metadata are preserved, it is possible to recover deleted files from these file systems rather easily. However, one must always take care to check whether the blocks that were assigned to a file have been reassigned to another file since the original file's deletion. Consider what could happen if this check did not take place, and data from a currently active file was assumed to be the data in a previously deleted file. Depending on what data is in the clusters now, this could lead to a lot of confusion. Sadly more than once the author has been involved in criminal cases where these mistakes were made by the prosecution. Remember when we said the stakes are high?

It is the job of data recovery tools to do this kind of validation, and different tools use different terms to describe it. Some tools may show the deleted files but note that the first cluster is no longer available. What this means is that the first cluster, rather than being on vacation or otherwise indisposed, has simply been reassigned to store the data for another file.

File Systems Revisited: NTFS



MFT and FILE Record Overview

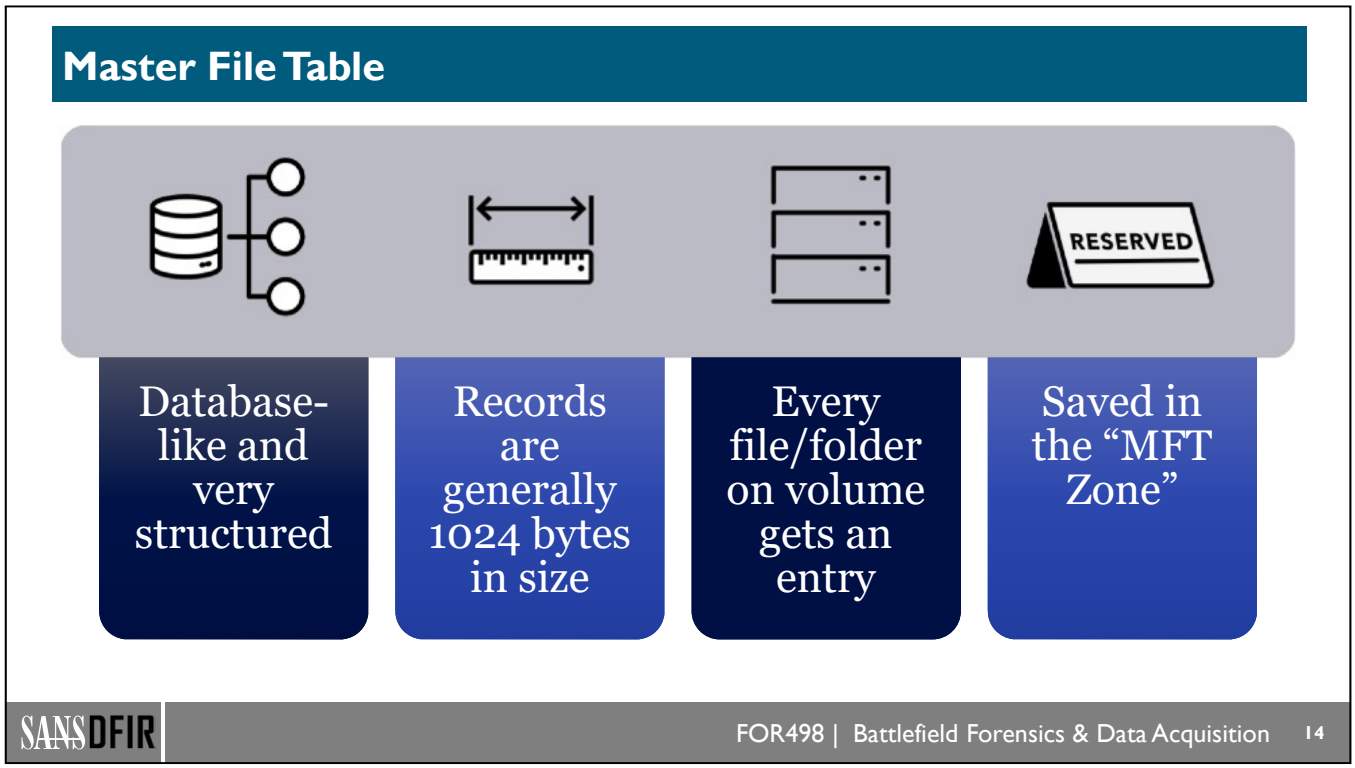
- Header and attributes

NTFS Features Overview

Notable Artifacts

- Alternate data streams and Volume Shadow Copies

This page intentionally left blank.



The Master File Table (MFT) [1] is a very structured database that tracks all the objects on an NTFS volume. Every object gets a FILE record within the MFT. Each FILE record contains a series of Attributes that contain the various data and metadata related to that file, such as timestamps, size, name, parent directory, and so on. When we say every object on a volume gets a FILE record, we mean a normal file gets a record, a directory gets a record, and even the volume name gets its own record.

Each record is, in most cases, 1024 bytes long, but in some circumstances, you may see larger record sizes such as 4096 bytes each. This is due to Microsoft traditionally making a FILE record a multiple of the sector size. This can, however, lead to a lot of wasted space, so Microsoft has relaxed this rule so FILE records smaller than the sector size can exist (i.e. 1024-byte records even when the sector size is 4096 bytes). If a file is small enough, the file's contents will be held within the MFT record along with its metadata. Otherwise, there is a pointer to which clusters contain the file's contents, known as a data run.

The MFT can become quite large. For example, a 1TB drive with over 400K files on it will produce an MFT that is over 485MB. If the MFT becomes fragmented, and the system must seek all over the drive to get to its various parts, the speed of the whole system will become very noticeably degraded. To prevent this, NTFS drivers traditionally created an "MFT Zone" for the MFT to reside in. This reserved the first 12.5% of the drive, and user files were then created after this zone so that the MFT had free space to grow into.

If the rest of the free space in the other 87.5% of the drive became full, then the MFT Zone would be cut in half. This would happen again when that other half got full, and so on until the drive was full. Once the MFT becomes fragmented, it cannot be defragmented by normal means. (Handy tip from a seasoned sysadmin: Quickest and most thorough way to defrag a drive is to copy all the files off a volume, format the volume, then copy them all back. The files will be created one at a time and NTFS will lay them down in contiguous order.)

We said the MFT was 'traditionally' created in the aforementioned manner. Today it would be a waste of drive space to pre-reserve so much space. Instead, the MFT now pre-allocates in 200 MB chunks.

The first 24 MFT entries are reserved for special use by the NTFS volume. The first 12 entries are used by system files that make NTFS work. These files are all named starting with a $ and are hidden from view unless using specialized tools. For our purposes, we want to focus on the MFT file itself, so we will only look at the first two entries.

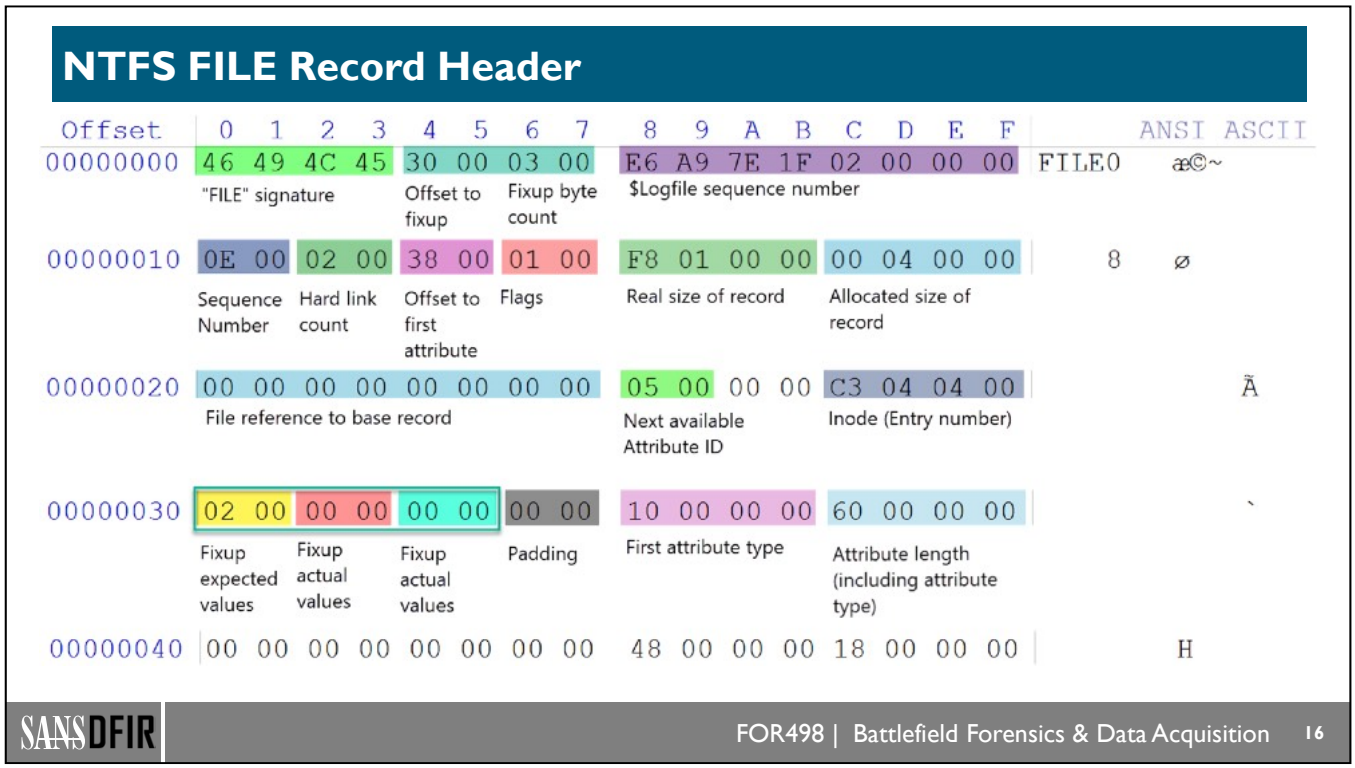
$MFT

The first record, record number 0, describes the MFT. This record provides us the name, $MFT, and information necessary to find all the other clusters containing the rest of the MFT. The Volume Boot Record (VBR) contains a pointer to the cluster this record will lie in, and the records within the MFT contain the pointers to the clusters for every other object. Unlike FAT, in NTFS the VBR is the only object that is tied to a specific sector on the disk and cannot be relocated elsewhere.

$MFTMIRR

The second record contains a backup of the $MFT record above in case the above record cannot be read due to physical damage to the disk. The information in record 0 that the system needs to read in the rest of the $MFT file is all we are really needing backed up, but because we are allocating space on the disk for an entire cluster, an entire cluster of MFT records will get backed up. Because the default cluster size is 4K and FILE records are usually 1K, this usually works out to be the first four records.

[1] Master File Table | `https://for498.com/9jldv`



Each MFT entry will begin with a signature of "FILE" (0x46 0x49 0x4C 0x45) [1]. If the system has detected an error in the entry, you may see a signature of "BAAD" (0x42 0x41 0x41 0x44). Looking for "FILE" at the start of a sector is a good way of locating MFT fragments in unallocated space.

The most important property we want to understand is the entry number, found at offset 0x2C, which serves as part of the identifier for a FILE record. This is often accompanied by the Sequence Number, found at offset 0x10. The sequence number is a counter which tracks the number of times an MFT record has been reused [2]. When an MFT record is allocated for the first time, its Sequence Number will be set to 1. When a file is deleted and the MFT record becomes unallocated, the Sequence Number is incremented again (and will only be incremented on subsequent deallocations). In other words, the sequence number goes up when a file is deleted and not when a new FILE record is created or reused.

Offset 0x14 contains a pointer (relative to the beginning of the FILE record) where the FILE record's attributes start. We will look at an attribute next.

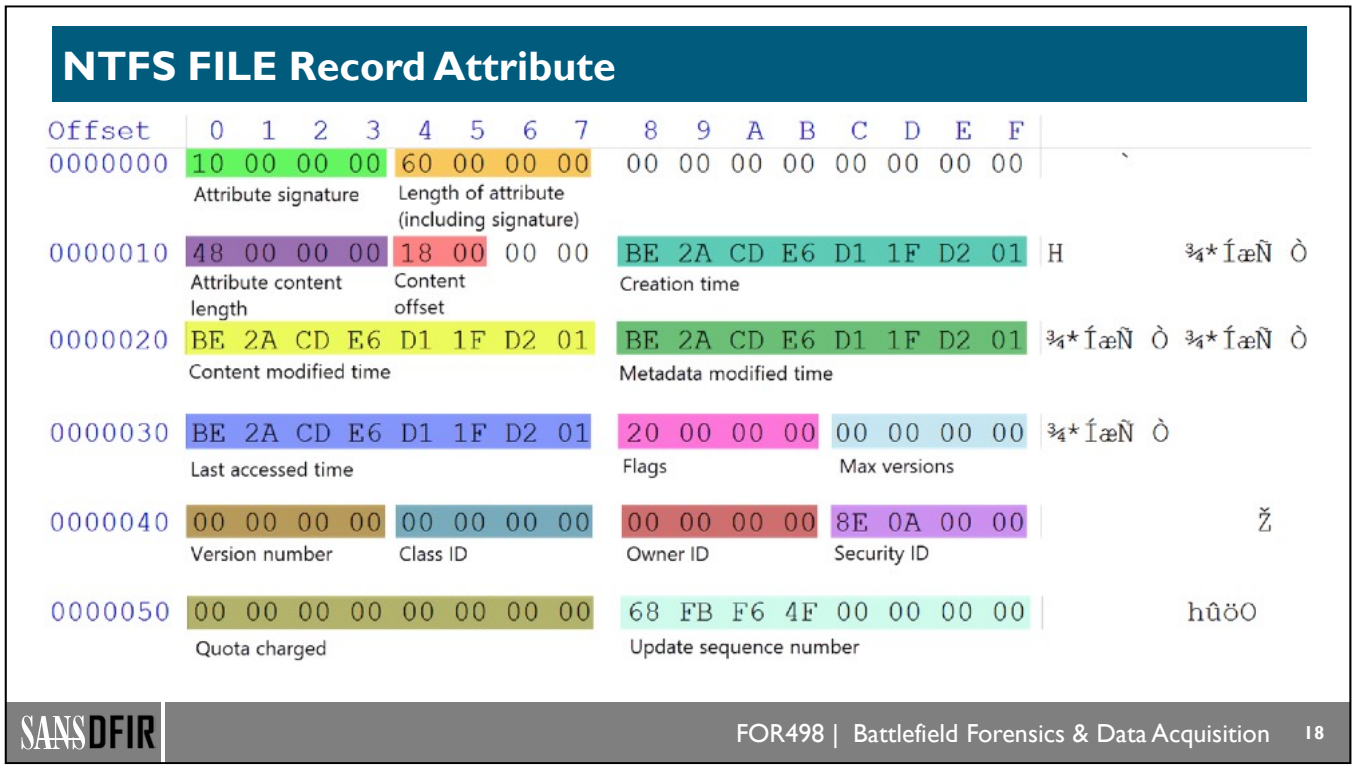
The Flags at offset 0x16 can have the following values (hex, binary, meaning):

0x00	0000	Not in use
0x01	0001	File in use
0x02	0010	Directory that has been deleted
0x03	0011	Directory in use

Deleting a file changes the "in use" bit of the flag to 0 but does nothing to clear out the rest of the data. Thus, many of the deleted file's metadata is still recoverable as long as the MFT record hasn't been recycled for a different file.

Finally, the offsets at 0x18 and 0x1C reflect the current size of the bytes being used by the FILE record and the total number of bytes allocated to the FILE record (0x0400 == 1024 bytes). The difference between these two values would be the slack space in the FILE record.

[1] FILE_RECORD_SEGMENT_HEADER structure | https://for498.com/kfw8h
[2] MFT_SEGMENT_REFERENCE structure | https://for498.com/tq7k5



FILE records store most of their data in the form of attributes. These attributes contain such things as timestamp, file names, where to find a file's data, and so on. The most common attributes are:

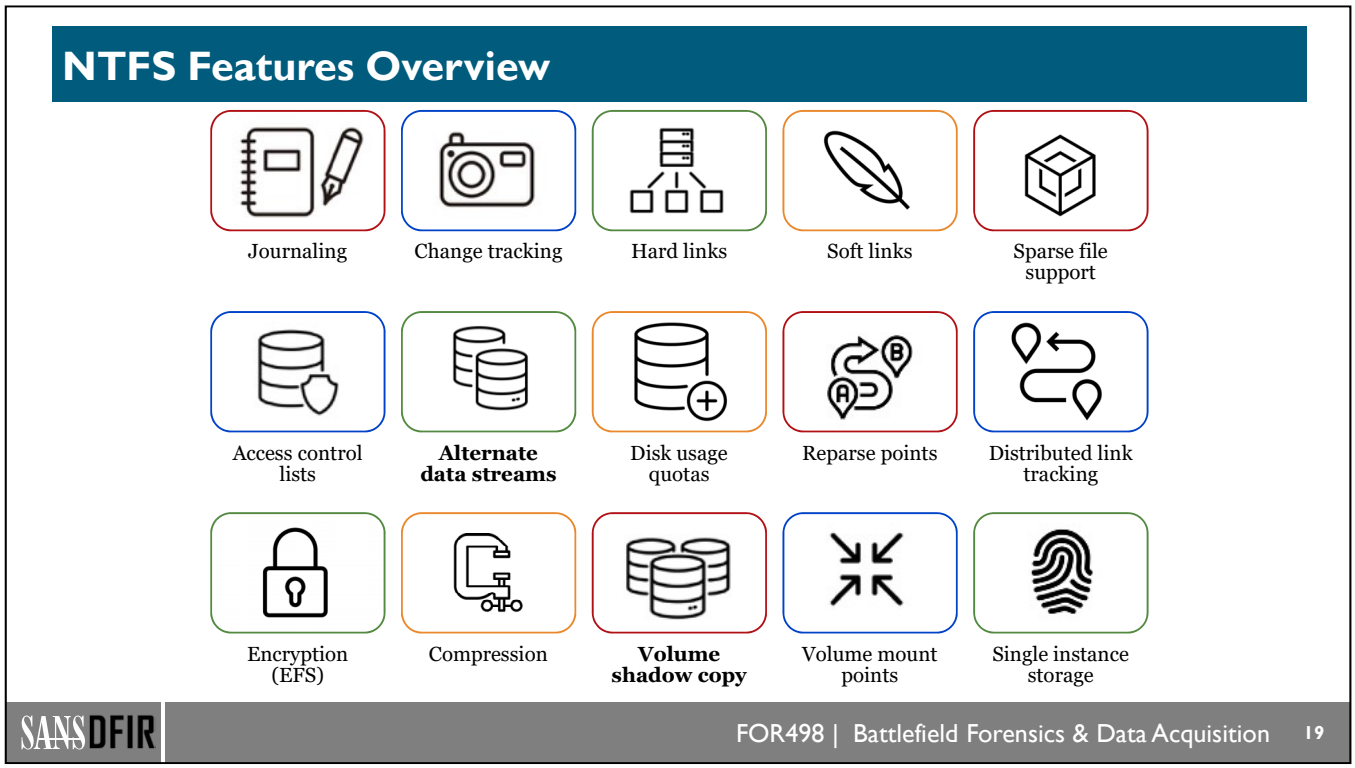
$STANDARD_INFORMATION: Contains timestamps, flags, and tracking information.
$FILE_NAME: Contains the parent directory, timestamps, size, flags, and file's name.
$DATA: Information about file data and either the contents of the file or pointers to the clusters containing file contents
$INDEX_ROOT: Contains the header of an index which tells system how to sort entries, size of entries, and so on.
$INDEX_ALLOCATION: Like $DATA, but for Indexes.

We will briefly look at a single attribute, $STANDARD_INFORMATION, to get an idea of what they look like, but we do not want to get stuck in the weeds on the low-level technical details.

The first four bytes of an attribute are the type. The $STANDARD_INFORMATION attribute is identified by a Signature value of 0x10, or 16 in decimal. Next is the size of the attribute, 0x60 in this example. Offset 0x10 tells us how long the attributes unique content is, and offset 0x14 tells us where that content starts. Each type of attribute's content will be different, and these values tell us where we can find the attributes payload.

Since the $STANDARD_INFORMATION's content starts at offset 0x18, let's see what is there. At offset 0x18 you will find a set of four timestamps: Created, Modified, MFT entry modified, and Last Accessed. These are the timestamps that are going to be used by the filesystem to display a file's date and time information. The MFT entry modified time is NOT displayed by Windows in File Explorer, but the other three are.

Again, this is just a brief example of what to expect in in an attribute and serves as an initial introduction to the kinds of structures that are in a FILE record.



As is evident from what we see above, NTFS has a wide range of features, and certainly far more than the previous file systems we looked at. Since you will most likely run into NTFS more than other file systems, we will dig into a few of the more important features such as Volume Shadow Copies (VSC) and Alternate Data Streams (ADS). Most of the other features are more likely to be found in an enterprise environment vs. an end user's system, but it is good to be aware of them regardless.

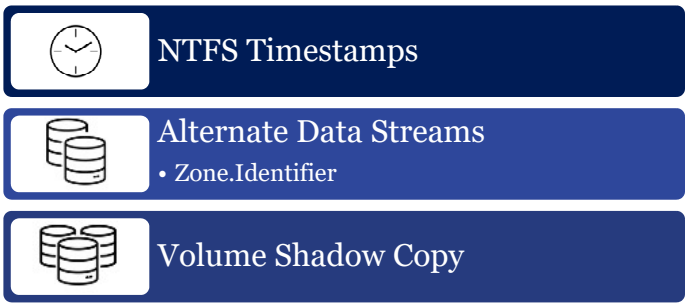
Some of the more useful features of NTFS include:

- NTFS tracks changes to metadata for objects in the file system to ensure it can always be recovered to a known good state in case of a system crash. While this does not guarantee data won't be lost, it does guarantee consistency of the file system. Other filesystems call this "journaling," and NTFS calls it "transaction logging."
- NTFS can track all files that have changed on a system via the USN Journal. This allows programs like a backup utility or virus scanner to know what files are new or changed since they last ran when they need to do an incremental pass over the drive.
- Portable Operating System Interface (POSIX) [1] compliance requires NTFS to support hard links and soft links. A hard link is when a single file's contents can be accessed via multiple file names often in different directories. At the user level, you may see two distinct files, but under the hood, there is only one copy of the data. Editing either of the hard-linked files would result in the data being changed. A soft link is where a second file is created but doesn't have any data at all. Instead, an alias or pointer to the other file is created, and opening it simply opens the other file's data without telling you.
- NTFS has incredibly robust security controls to prevent users from opening files and folders they do not have permission to access. (Of course, that can all be bypassed by mounting the drive in Linux, but that is a different discussion.)

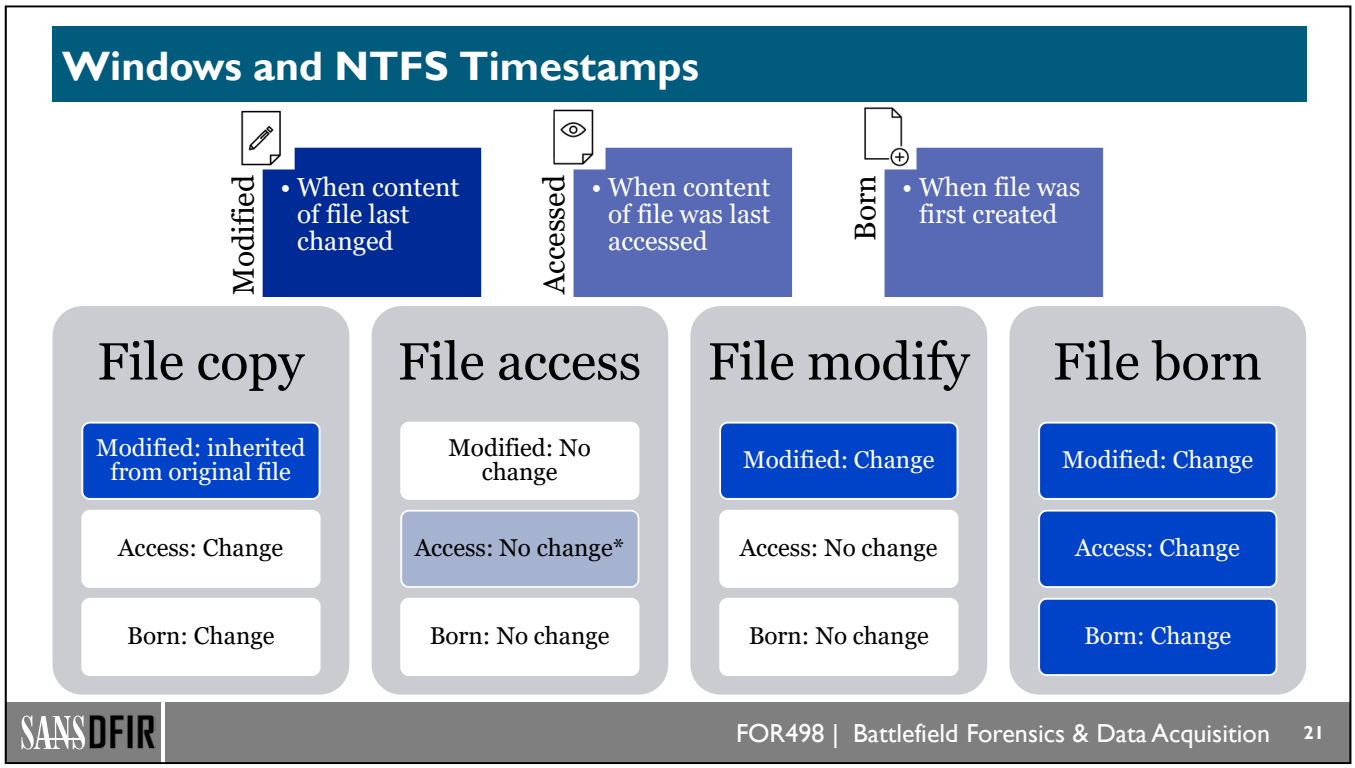
[1] POSIX information | `https://for498.com/x1ac8`

Notable NTFS Artifacts

- In FOR508, Advanced IR/Forensics, we go into NTFS in-depth
- Some notable artifacts are covered now because they are important in understanding some of the items you will see in this course



This page intentionally left blank.



NTFS records three sets of times referring to files/folders. The three include:

1. Last modification time of file data
2. Last access time of file data
3. The time when the file was born (created)

There is also a fourth timestamp that NTFS tracks, known as the Record change time, which is updated any time any of the data in the FILE record itself is modified (such as when moving a file from one directory to another, since the pointer to the new parent directory would be updated in the FILE record).

Collectively these are referred to the MACB times. Notice above we say a file is "born" vs. "created" and the reason for this is because the "C" in MACB stands for Change, which is the FILE record change timestamp. Furthermore, using MACB allows for keeping things consistent across file systems. To summarize:

M: Modified timestamp
A: Accessed timestamp
C: Change record timestamp
B: Born timestamp

These designators are also how many forensic tools denote them, so this is another reason to get comfortable with this naming convention.

All timestamps in NTFS are 64-bit Windows FILETIME structures. When these times are stored, they are stored as a number representing the number of 100 nanosecond intervals since January 1, 1601.

The rules above are the general rules when it comes to files being copied, accessed, modified, or created. The MFT record change timestamp is still being researched as it relates to the other timestamps changing.

One of the many mistakes investigators make is assuming that disabling the last access time will result in zero updates to a file's last access time. This is incorrect. The file last access time will update during a file copy or move. It would not update by merely opening the file, though.

It is worth noting that a file generally retains its Creation Date during a "cut-and-paste" operation via Windows Explorer. However, if the move happens via the command prompt, the Creation Date changes.

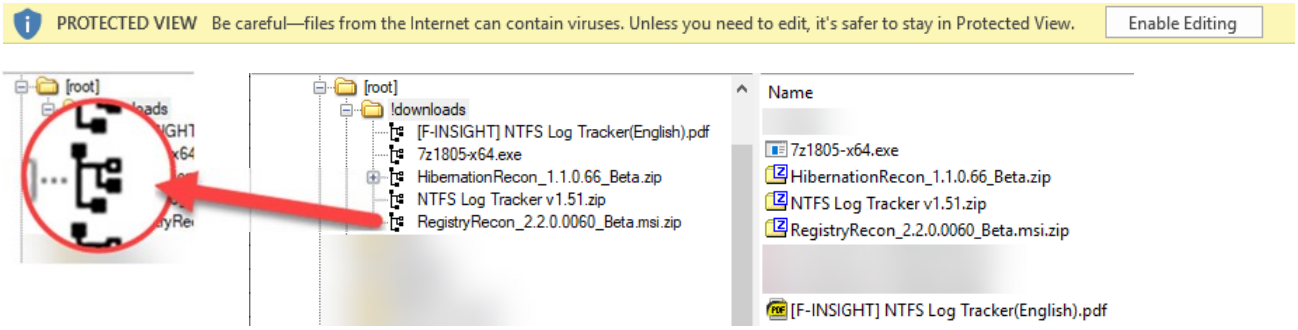
For a more in depth look at Windows Timestamps and how they change depending on what is happening to a file, see the SANS Red Poster. [1]

* No Change on NTFS in Win7+ unless specifically enabled via the Registry

[1] SANS Windows Forensic Analysis poster | `https://for498.com/a96im`

NTFS ADS: Zone.Identifier (1)

Ever wonder how Windows knows you downloaded a file from the Internet?



FTK imager showing downloaded files. Notice the icon on the files on the left.

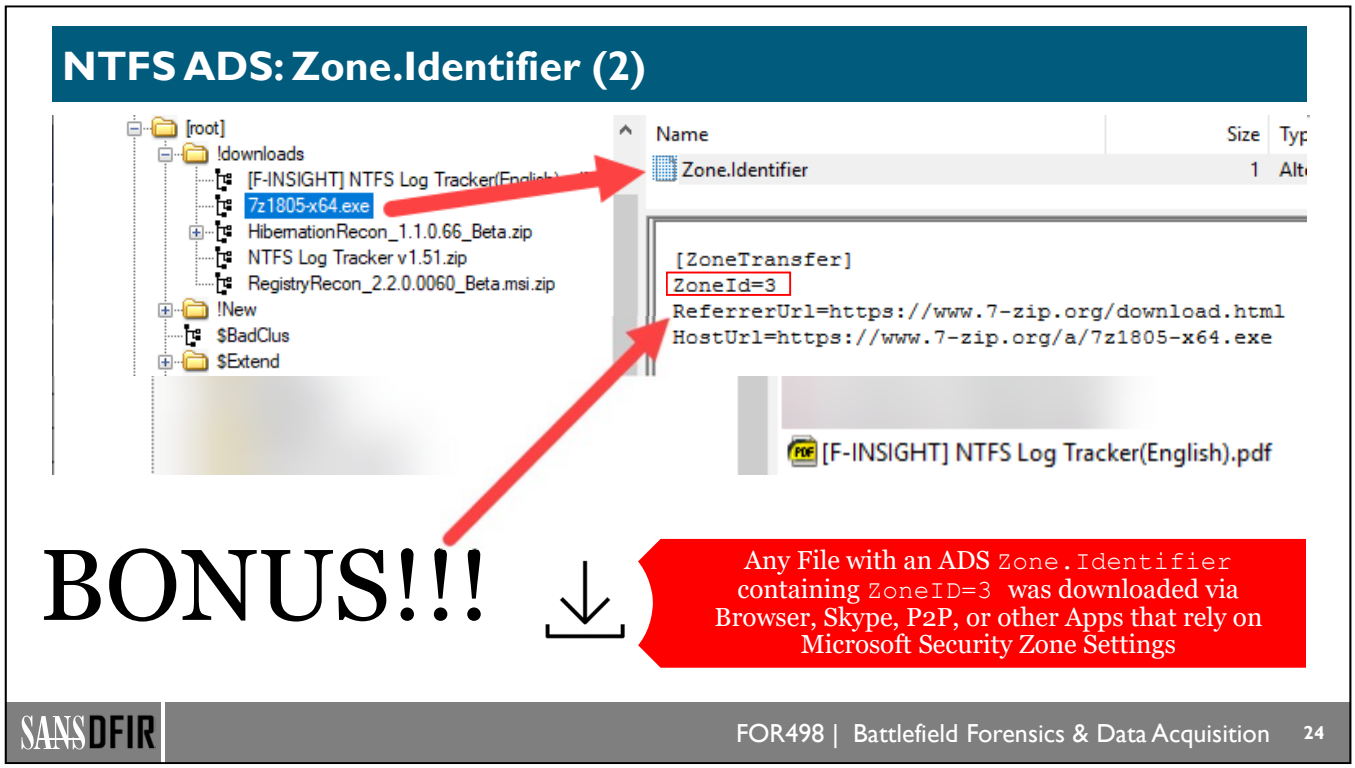
An alternate data stream (ADS) can be thought of as a file within a file. In other words, an ADS is a separate set of data that is attached to another file which can be referenced by name. An ADS's data is separate from the primary file and editing the primary file's data does not change the ADS's data, and vice versa. A file can have more than one ADS as well.

All files in NTFS will have a DATA attribute that tracks the content of the file. The main DATA attribute will not have a name, because its name is stored elsewhere in the FILE record (namely, in the FILE_NAME attribute). However, any DATA attribute after the first MUST have a name so that NTFS can differentiate between any ADSs that are present. The data structures for FILE records, including attributes and ADS, is discussed in detail in FOR508.

Because the rest of Windows is designed only to accommodate the primary data stream, most tools cannot report on the existence of an alternative data stream, let alone its size and contents. No other metadata, such as timestamps, about an alternative stream are maintained. This makes ADS an attractive place to hide illicit tools or data, or so it would seem. Each version of Windows from XP SP3 to 7 has placed more and more restrictions on the things you can do with an ADS, making them less useful to hackers. As we will see though, from a forensics point of view, detecting ADSs is trivial to do.

An example of a quick win in an investigation would be locating all files that were introduced to a computer from some external source, such as the Internet or an external drive. As we just saw, looking for specific Alternate Date Streams, ones named "Zone.Identifier", provides a means to quickly find all the files that came from one of these external sources. While this artifact will only be present for files created via a browser, Skype, P2P session, or an online application, we can leverage other means to determine if a file was copied from external storage, such as a USB device. In this case, files will not be tagged with a "Zone.Identifier" ADS.

Alternate data stream functionality was originally included in Windows NT as part of the Services for Macintosh functionality as a way a Windows file server could support the concept of a Resource Fork that exists in Mac filesystems. Ironically, Macintosh clients don't make use of it and opt instead to create a second hidden file that contains the second set of data.



In most cases, the contents of the Zone.Identifier ADS usually contains "ZoneID=3." A value of three means that the file was downloaded from the Internet. Three is not the only possible value however, so if you see something other than three, use this chart to determine where it originated [1]:

NoZone = -1 MyComputer = 0 Intranet = 1 Trusted = 2 Internet = 3 Untrusted = 4

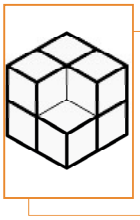
So going back to our original question, how does Windows know your Office document came from the Internet? When viewing a Word document or PowerPoint file has been tagged as "downloaded" from the Internet, Office prompts you to click "Enable" to edit it, all because of the Zone.Identifier ADS. The idea here is to give the user a warning that the document may have originated from somewhere outside their organization vs. simply opening the document, playing any embedded macros, etc. which could lead to the computer becoming compromised. Of course, getting users to do something other than clicking on “Enable” as fast as possible is another challenge!

From a forensic analysis perspective, ANY file that has the Zone.Identifier with ZoneID=3 value originated from the Internet and is direct evidence of "File Download.“ This can be useful because any file downloaded via a browser will be tagged with a Zone.Identifier ADS. This includes executables or DLLs as well. A quick scan of C:\Windows\System32 to see whether any files have the "Zone.Identifier" ADS is an easy way to find files that are out of the ordinary.

Finally, in Windows XP, only .exe files are tagged with this ADS. In Windows 7 and newer, any file downloaded will be tagged with this ADS.

[1] - Zone.Identifier Alternate Data Stream (ADS) information | `https://for498.com/-s0gt`

NTFS: Volume Shadow Copy



Tracks changes in NTFS volume:

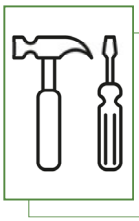
- Block/cluster level backup
- 16 kilobyte blocks

Enables a user to:

- Revert the file to any previous version
- Restore a previous version from backup
- Make a copy of previous version

Shadow copy limitations:

- Previous version is not stored every time a user changes a file
- Snapshots are staggered but typically one occurs each week
- Win7/8 uses 3-5% of a volume (can be larger) for VSCs



VSC tools:

- MagnetForensics IEF
- VSCMount
- Ubuntu SIFT Workstation with libvshadow
- Shadow Explorer

Windows Volume Shadow Copy is an exciting area for forensic investigators looking for an additional edge in computer forensics. Versions of windows prior to Windows Vista used restore points to back up certain kinds of files, but starting with Windows Vista, Volume Shadow Copies replaced restore points. Vista and newer now log changes to an entire volume and keep track of the specific clusters that are changed in the new Volume Snapshot Service or VSS. Most versions of Windows have Shadow Copy enabled by default.

Volume Snapshot Service (or Shadow Copy), being the new "System Restore" for Vista through 10, essentially acts as the equivalent of "Time Machine" for Apple's OS X operating system. VSS performs a cluster by cluster diffing/backup and stores that information into a new file. In a nutshell, you can rewind a file, a directory, or even an entire volume to a previous state—wonderful for recovering from a computer crash, and equally as wonderful for forensics!

Since VSS enables a user to essentially revert an entire volume, a folder, or a file back in time to a previous version, an investigator can also copy out of the Shadow Copy previous versions of files and examine the differences. On Windows Server, VSS takes a snapshot only once a day. It will not log continuous changes every time the user saves a file. On current, non-server versions of Windows, a VSC is generated once a week outside of special circumstances like a new driver being installed or user initiation of a new VSC.

The VSS monitors all changes made to a VSS-enabled volume. These changes are monitored in 16-KB "blocks." If a change is made to any data inside a 16-KB block, the entire block is copied to a volume shadow copy file. This happens regardless of the file system settings. All volume shadow copy files are stored in the **"System Volume Information"** folder on the root of the volume and are recognizable by their names. The file names for volume shadow copies look something like this:

{802c6ba4-300b-11df-a523-005056c00008}{3808876b-c176-4e48-b7ae-04046e6cc752}

You can tell that this is a volume shadow copy because the second set of braces contains the number 3808876b-c176-4e48-b7ae-04046e6cc752, which is a unique identifier specific to the volume shadow service.

Tools that can access and/or parse volume shadow copies include:

- Magnet Forensics IEF
- VSC-Toolset: `http://dfstream.blogspot.com/p/vsc-toolset.html`
- Ubuntu SIFT Workstation with libvshadow installed on it: `http://digital-forensics.sans.org/community/downloads`
- VSCMount: `https://ericzimmerman.github.io/#!index.md`

There are several ways to create a new volume shadow copy:

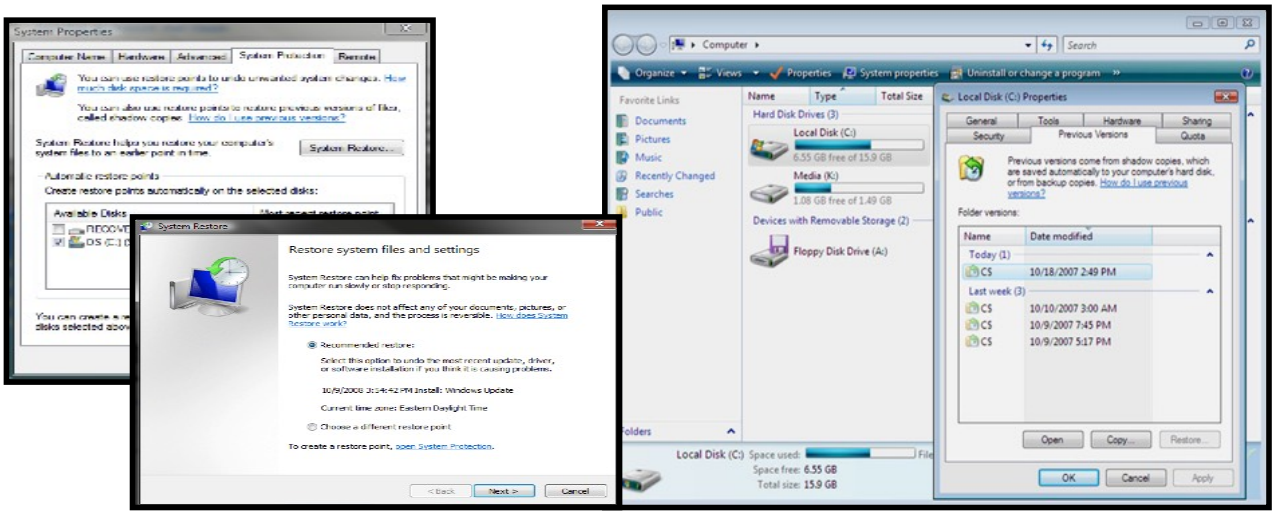
- System Snapshot
- Software Installation
- Manual Snapshot

On Windows Vista, the system snapshot is scheduled to take place every 24 hours. On Windows 7 and newer, it is scheduled to take place once every 7 days. You may notice that this is not exact. Windows Vista will not take volume snapshots exactly every 24 hours, but this is roughly the interval at which they occur. This is because the VSS will create new shadow copies only once the computer has been idle for a certain amount of time, the computer is being turned off or rebooted, and so on.

A VSS-enabled volume will always have a “live” volume shadow copy. This file saves all the 16-KB block changes and then, once a new volume shadow copy is created, these changes are committed to the current VSC and the shadow copy is archived. From this point, the contents of that volume shadow copy remain unchanged until it is deleted[1].

[1] Into The Shadows | `https://for498.com/5razy`

VSC: System Restore and Previous Versions

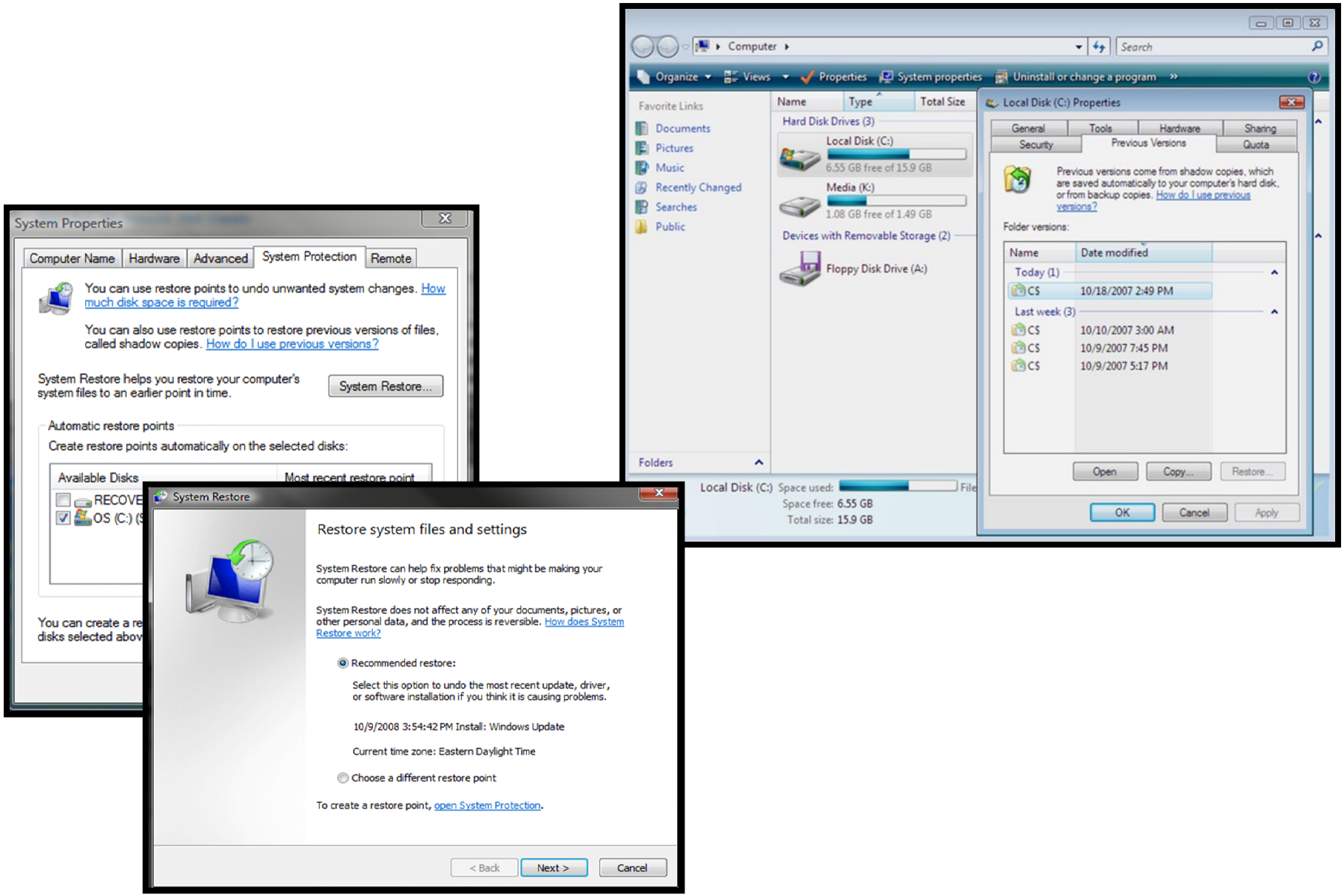


Here we see a snapshot created by VSS. Notice the entire C:\ volume can be recovered to earlier points in time. What this would mean to a forensic analyst is that if a user wipes a file today, it might still be recoverable from one of the previous snapshots. If an investigator examined the shadow volume created from yesterday's snapshot, the file, if it existed at that point in time, would be recoverable from that volume snapshot. The shadow volume is an exact duplicate/backup of the entire volume including unallocated space at the point in time the snapshot was created.

How many shadow volumes will an investigator have access to? It depends on disk size. Generally 3-5% of disk space is allocated for volume shadow copies. Like most other things however, upwards of 30% of disk space could be utilized should a user decide to manually increase the amount of reserved space.

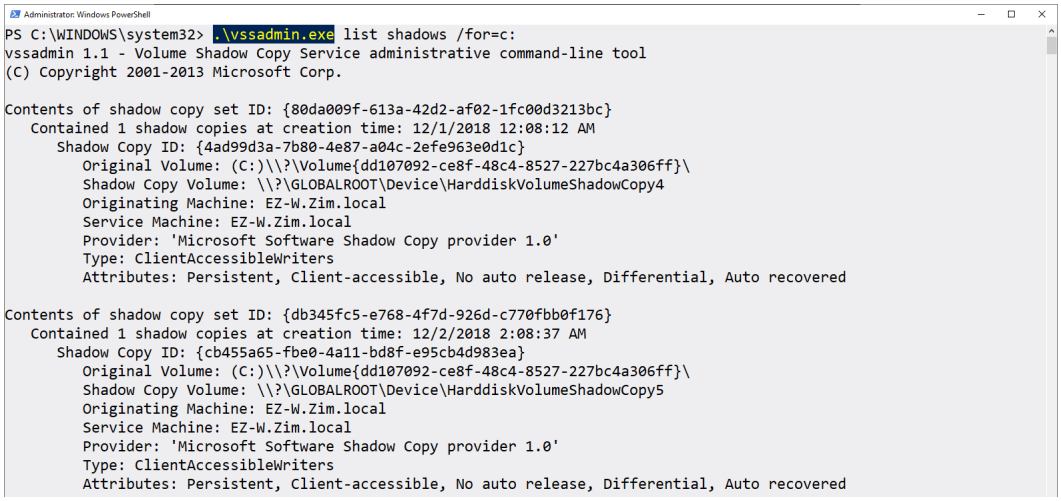
VSS is utilized in two different ways in Windows Vista and newer. For all versions, system restore will utilize the VSS in order to revert the computer to a previous snapshot. Business, enterprise, and ultimate versions, however, enable "Previous Versions", which will allow a user to "rewind" a file, a directory, or an entire volume back to a given point in time.

In Windows 8, the capability to browse previous versions of the volume shadow copy was removed but it is back again in Windows 10. In both cases however, the VSS service is still active and accessible to the forensic tools that you might use, either way.



VSC: List Available Shadows

C:\vssadmin list shadows /for=c:



How many shadow volumes are stored on the system you are examining? You can obtain a list of existing shadow volumes in the Volume Shadow Copy Service by executing the tool **`vssadmin`**.

To obtain a list of the shadows, execute:

```
vssadmin list shadows
```

This command will list the available shadows for all volumes on each device in the computer. In many cases however, you only want to see a specific drive's VSCs. In that case, we can use the /for switch, like this:

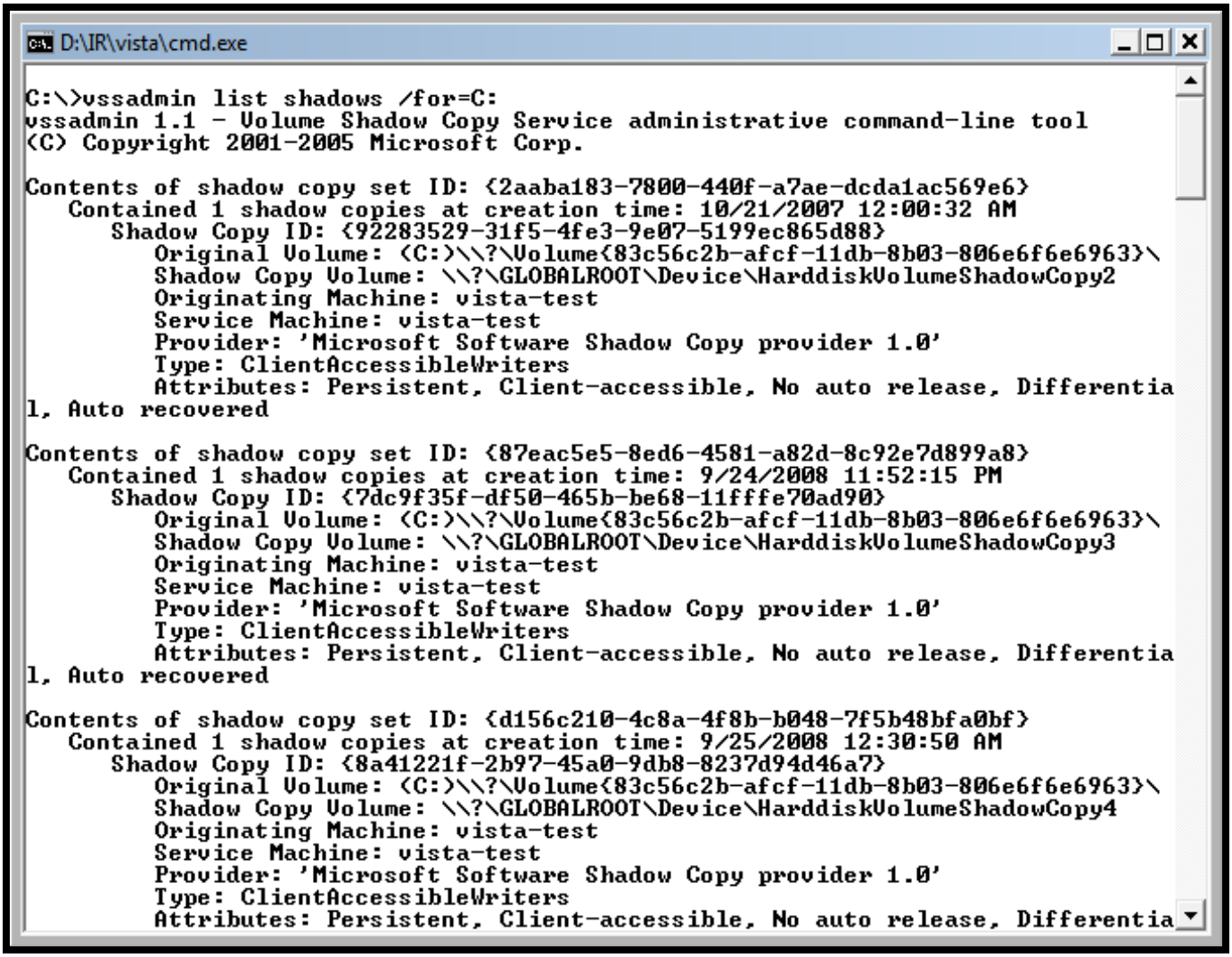
```
vssadmin list shadows /for=c:
```

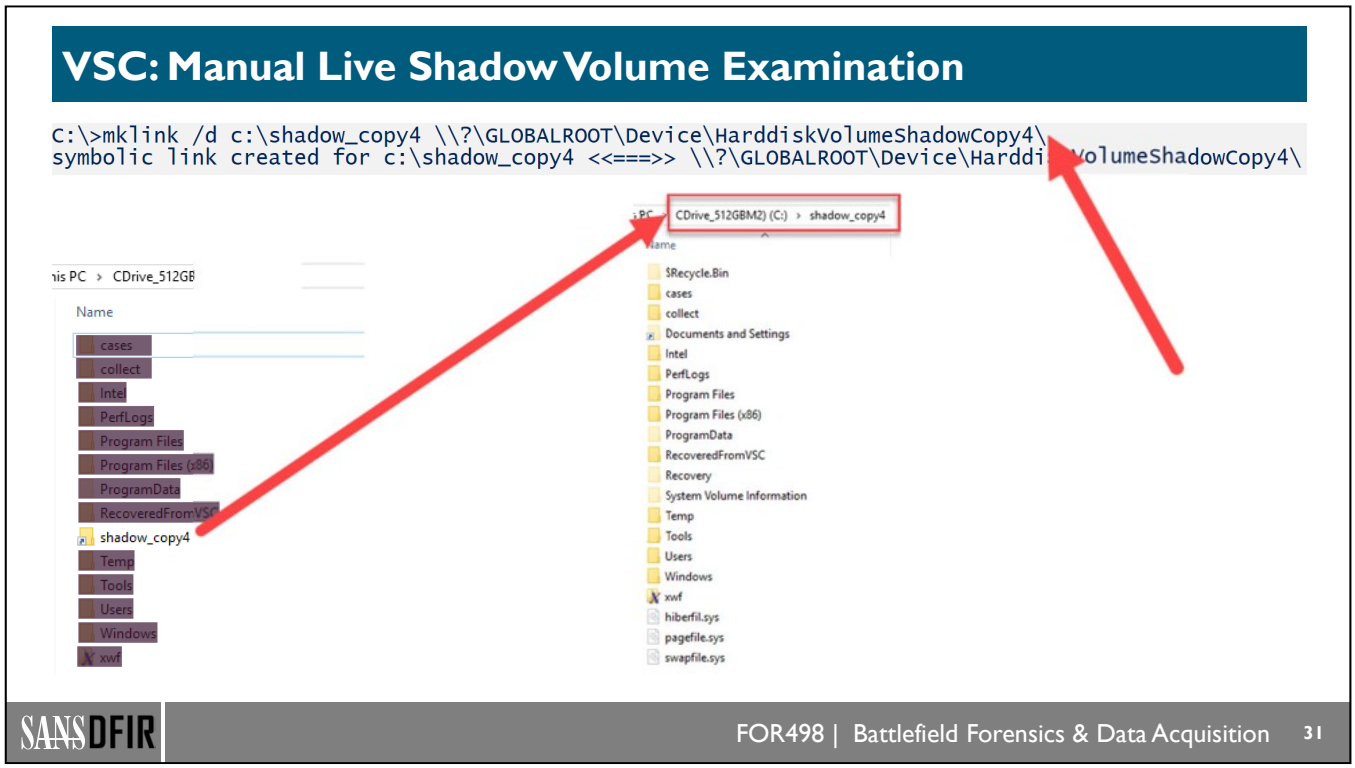
Where "c" is the drive letter you want to filter by.

Things to notice:

1. The **Shadow Copy Volume** property is the name of the volume that we will use to examine the contents of that specific volume. It is helpful to select the value and copy it to the clipboard for later use.
2. The **Originating Machine** would be noteworthy if you have plugged in an NTFS drive from another shadow copy-enabled machine.
3. The system time of the **creation time** of the VSC will tell you when the snapshot was created. This time is important, as you know which shadow copy volume might contain your data.

From the output of vssadmin, note the total number of shadow copy volumes from the machine. In this example, we are only looking at the first two, but there were six total shadow copy volumes that were listed as a result of running the vssadmin list shadows command shown above.





On a live machine, it might be useful to manually browse or scan a directory that contains a shadow copy volume. It is relatively easy to do this from an administrator-enabled command prompt using the tool **`mklink`**. **`mklink`** creates symbolic links from the command line.

Recall from the previous slide that the first VSC's Shadow Copy Volume property was "\\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy4"

To create a symbolic link to this VSC, open an administrator command prompt and issue the following command:

"mklink /d C:\shadow_copy4 \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy4\"

Pay careful attention to the Shadow Copy Volume value. Notice that it has an extra backslash at the end of the path. Without this backslash, the command will fail.

This technique can be adjusted to access any of the available VSCs, simply by selecting one of the "Shadow Copy Volume" names based on the date in time you would like to examine. Once selected, create the symbolic link using mklink as shown above.

This capability is incredibly useful in situations where a bad guy might have wiped or overwritten files with zeros or some other pattern to destroy data. Recovery is possible by going back a day or two into the appropriate VSC to retrieve the file from allocated space.

NOTE: You **must** do this using a Command Prompt Window, because the mklink command is built into cmd.exe. Using PowerShell will not work without a bit of trickery that we will see in the VSC exercise.

VSC: Automation with VSCMount (1)

```
PS D:\Tools> .\VSCMount.exe --dl c --mp C:\ShadowCopies --ud
VSCMount version [redacted]

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/VSCMount

Command line: --dl c --mp C:\ShadowCopies --ud

Creating directory 'C:\ShadowCopies_C'
Mounting VSCs to 'C:\ShadowCopies_C'

VSCs found on volume C: 5. Mounting...
        VSS 1    (Id {33724885-8a6d-4cd2-afa7-8ac41ba57428}, Created on: 2019-06-03 11:13:31.3040460 UTC) mounted OK!
        VSS 2    (Id {3e5a51ec-7736-4ea8-aae1-51bbfe49d1c4}, Created on: 2019-06-04 13:00:55.0852810 UTC) mounted OK!
        VSS 3    (Id {ade1bb12-4374-4304-980b-7ea67c5f4672}, Created on: 2019-06-05 16:14:49.2358140 UTC) mounted OK!
        VSS 4    (Id {81b8cac2-70c1-457a-860d-445670fe0165}, Created on: 2019-06-06 18:03:49.6276570 UTC) mounted OK!
        VSS 7    (Id {a4693e01-4b9b-488e-a2af-f4ef15d6932f}, Created on: 2019-06-12 03:33:54.6762530 UTC) mounted OK!

Mounting complete. Navigate VSCs via symbolic links in 'C:\ShadowCopies_C'

To remove VSC access, delete individual VSC directories or the main mountpoint directory
```

.\VSCMount.exe -dl C --mp C:\ShadowCopies --ud

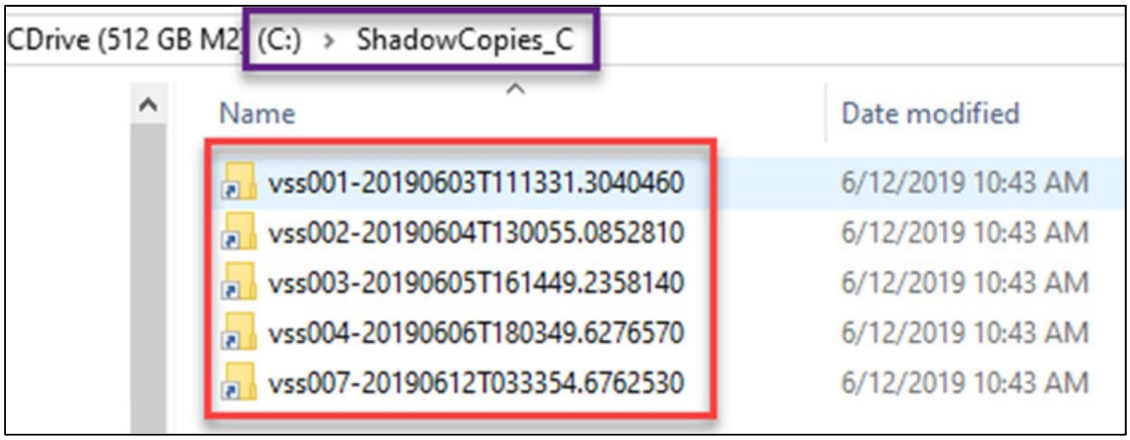
VSCMount is a command line tool that enables you to find and mount all available VSCs for a given volume in one operation. It works from both cmd.exe and PowerShell.

At a minimum, the --dl and --mp switches are required. This tells the program where to look for VCSs and where to create symbolic links for the VSCs that are located. The --ud switch adds a timestamp to each directory which can help find data if you are targeting a specific timeframe.

In the example above we are mounting all available VSCs from drive letter C:\ under the directory named C:\ShadowCopies_C. As we will see next, we could just as easily mount all the VSCs from an image file as well and substitute the drive letter for the image in place of the C:\ drive above.

Because VSCMount handles all available VSCs automatically, you are less likely to make syntax errors and have all the available VSCs mounted at once in a single place. By using the --ud switch, the creation time for the VSC is appended to the link so you can quickly pivot into the relevant timeframe for your investigation.

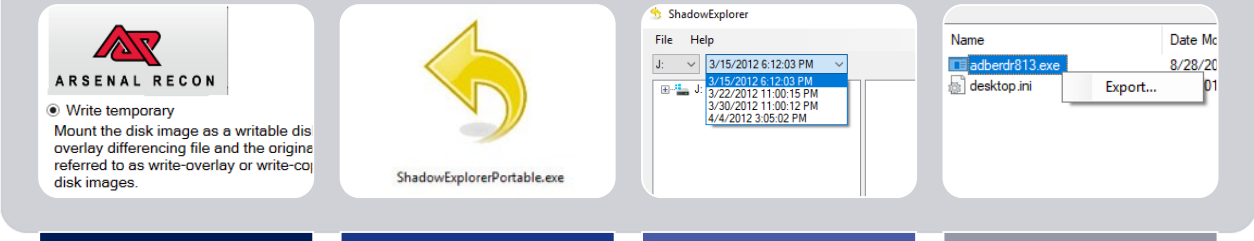
VSC: Automation with VSCMount (2)



Each link in the directory points to the numbered VSC. Each link also contains the VSC creation timestamp.

Under the mount point directory, a symbolic link for each VSC is created.

ShadowExplorer: Examine/Export Files from VSC



Step 1: Mount disk image in Arsenal Image Mounter in "**`Write Temporary`**" Mode

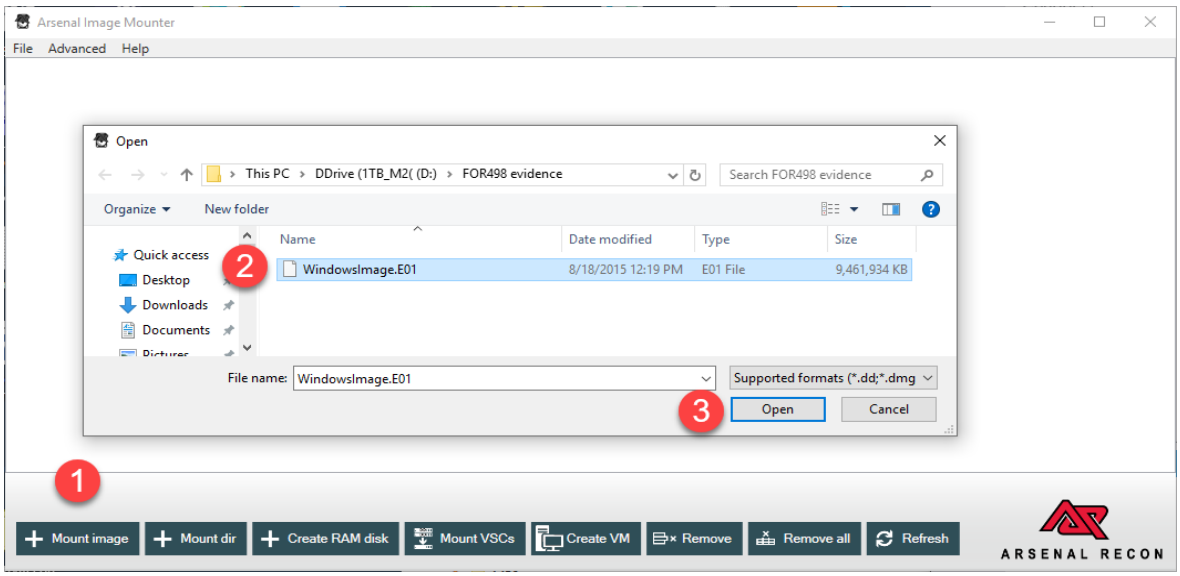
Step 2: Launch ShadowExplorer as Administrator

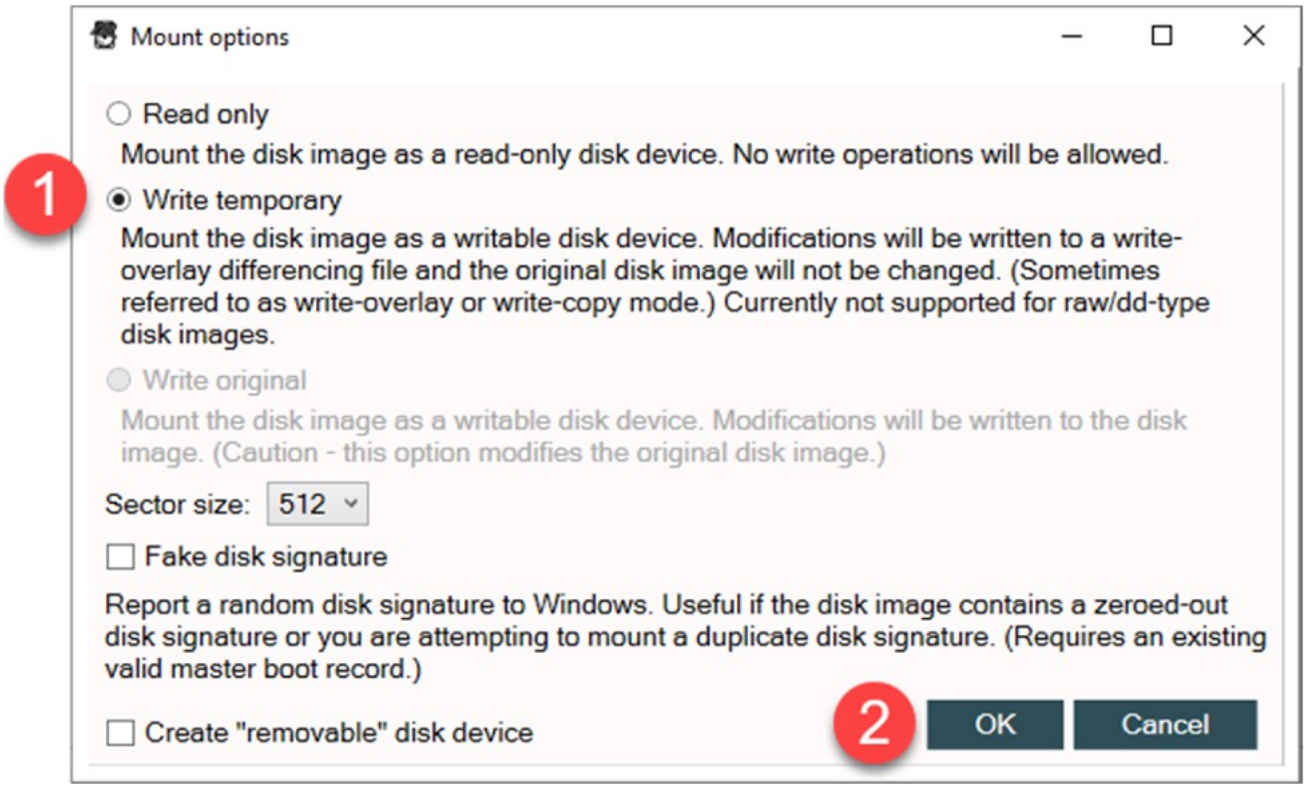
Step 3: Browse Snapshots

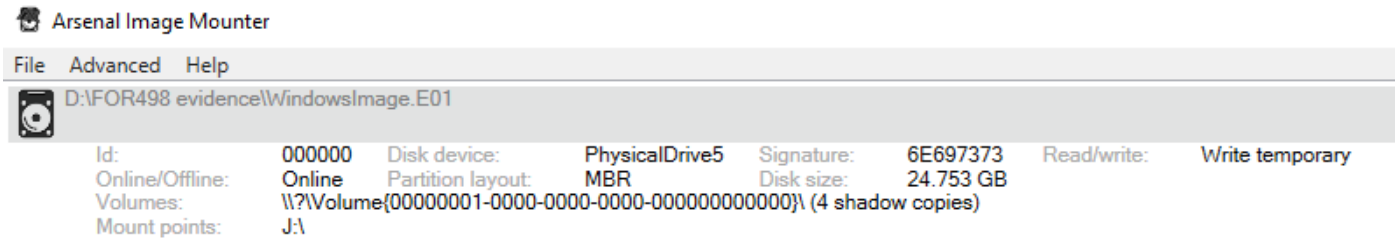
Step: 4 Extract Files using **`Right Click -> Export`**

Examining Volume Shadow data is made easy with ShadowExplorer. ShadowExplorer allows for a user to browse through a familiar Explorer folder interface in the VSCs of their choosing. This allows the investigator to quickly parse and extract files of interest from the volume shadow. In my experience, ShadowExplorer is the perfect tool to attempt to extract targeted information from a previous time.

Step 1: Mount disk image in Arsenal Image Mounter in "**`Write Temporary`**" Mode. NOTE: Arsenal Image Mounter is needed, as FTK Imager's mount capability does not expose the VSCs to the underlying operating system.

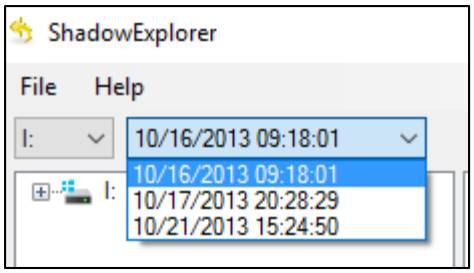


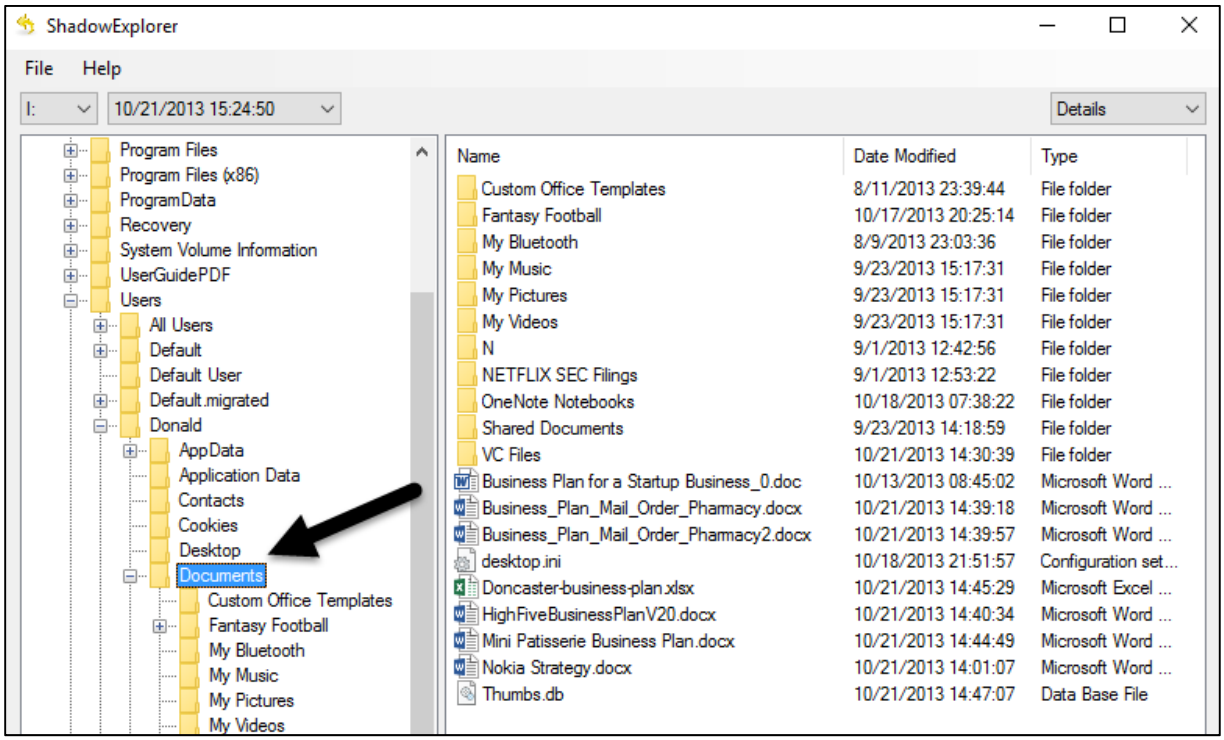




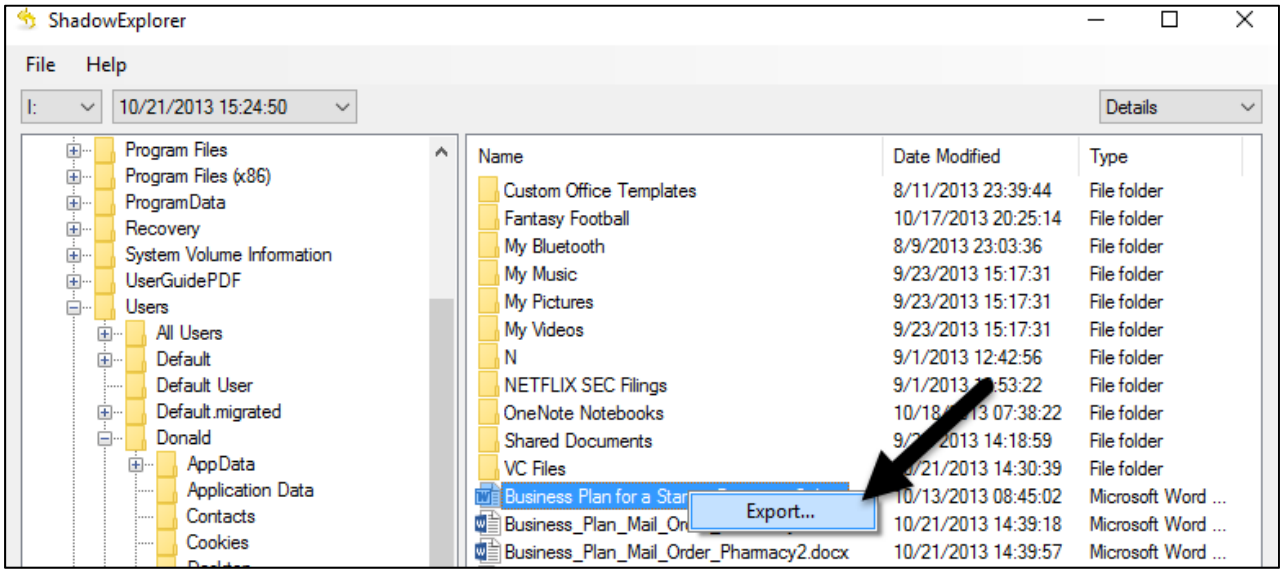
Step 2: Launch ShadowExplorer as Administrator. If you do not launch ShadowExplorer as Administrator, then it is likely not going to be able to parse all the files/folders available to the analyst.

Step 3: Browse Snapshots. Browsing snapshots is easy to accomplish in ShadowExplorer. If you can find files in Windows Explorer, then this interface will make you feel right at home.





Step 4: Extract Files using **`Right Click -> Export`**



Summary

- Understanding file systems goes a long way in understanding what is available and how
- The NTFS is the most robust and commonly used file system in use today
- ADS allows us to determine how a file may have gotten onto the system
- VSC lets us go back in time to see how a system looked at a previous moment
- ShadowExplorer is a quick and easy way to navigate through previous iterations of a file system

This page intentionally left blank.

Exercise 4.1

Volume Shadow Copy Acquisition

Synopsis: In this exercise, you will identify Volume Shadow Copies within a forensic image. You will then manually mount one of the VSCs, as well as use a tool to mount several them simultaneously. Finally, you will extract data out of one of the VSCs.

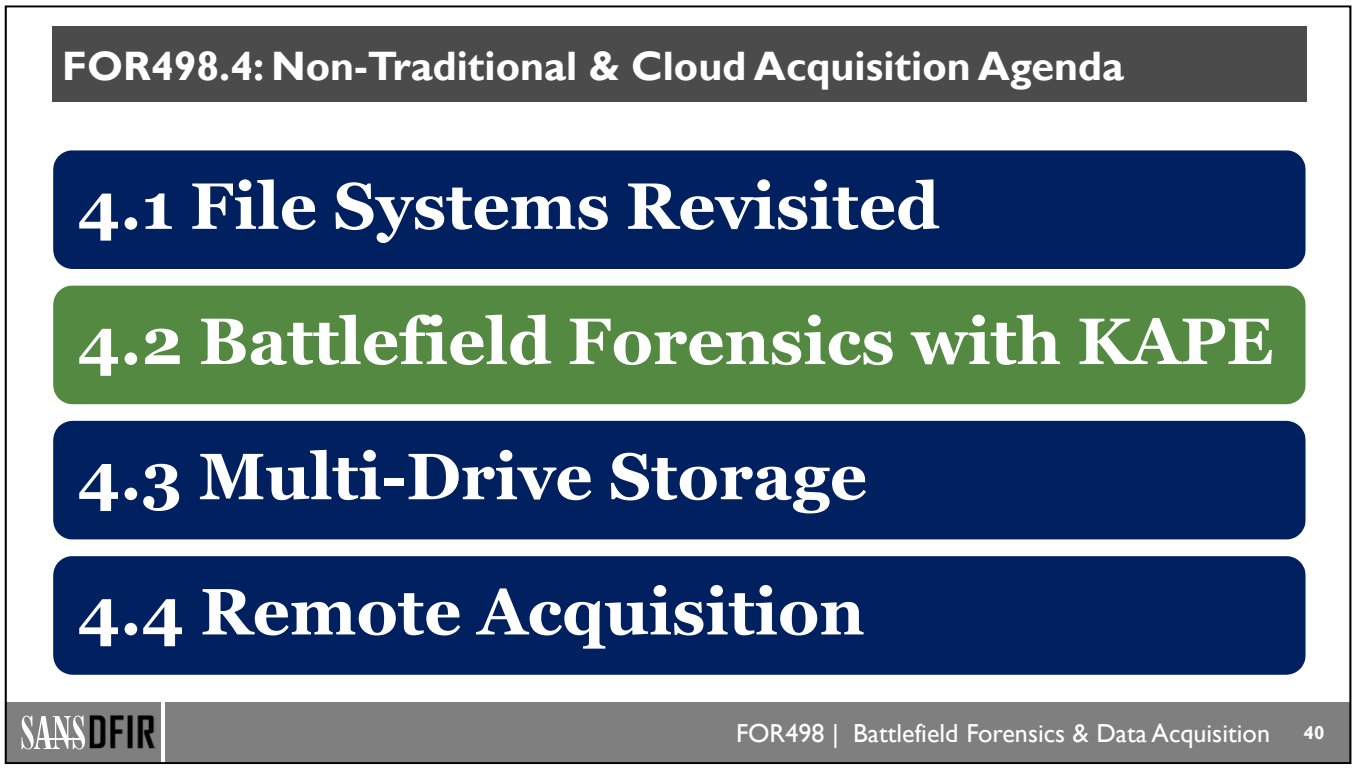
Average Time: 30 Minutes

This page intentionally left blank.

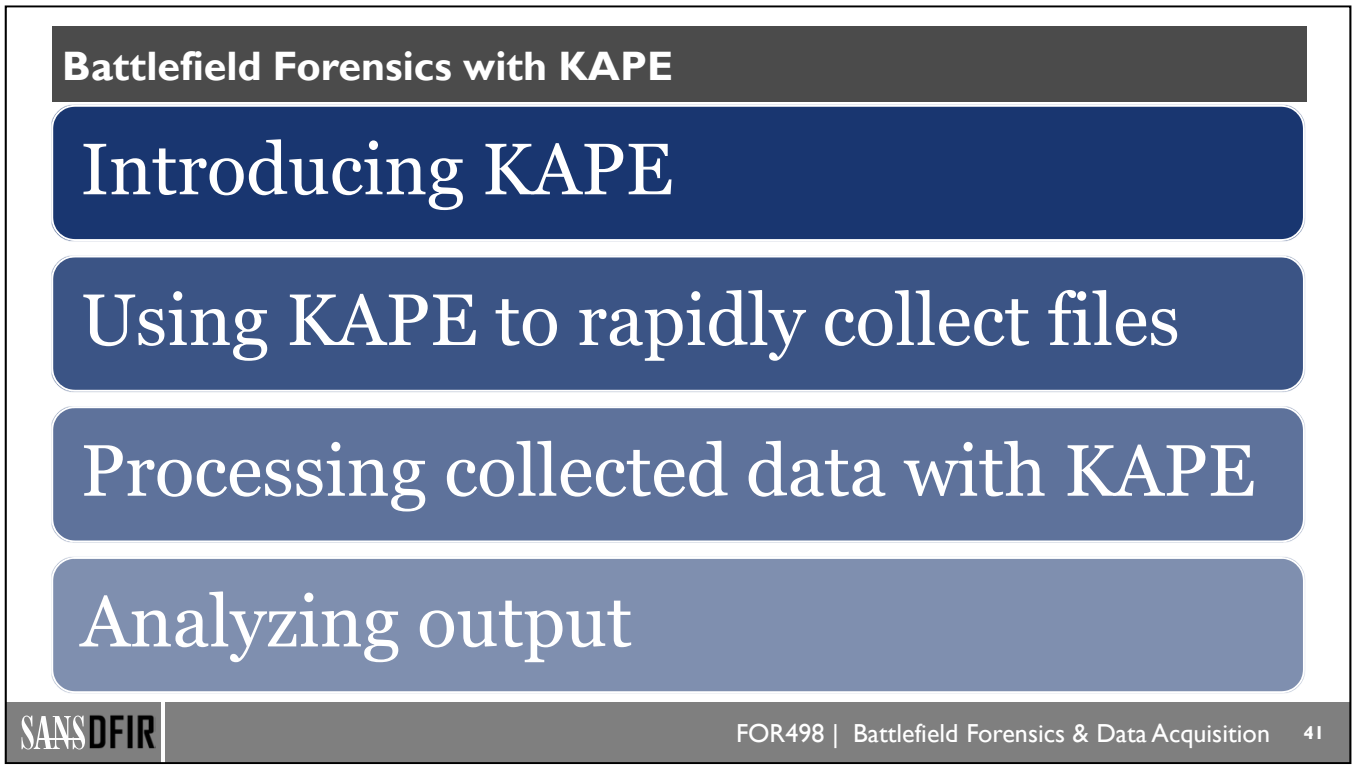
Exercise 4.1 Takeaway

- Volume Shadow Copies are a fantastic resource of historical forensic data.
- There are many ways to access VSCs, ranging from command line to graphical user interfaces.
- Due to how much data is available in VSCs, the best approach is a targeted triage collection against VSCs instead of imaging a VSC in its entirety.

This page intentionally left blank.



This page intentionally left blank.



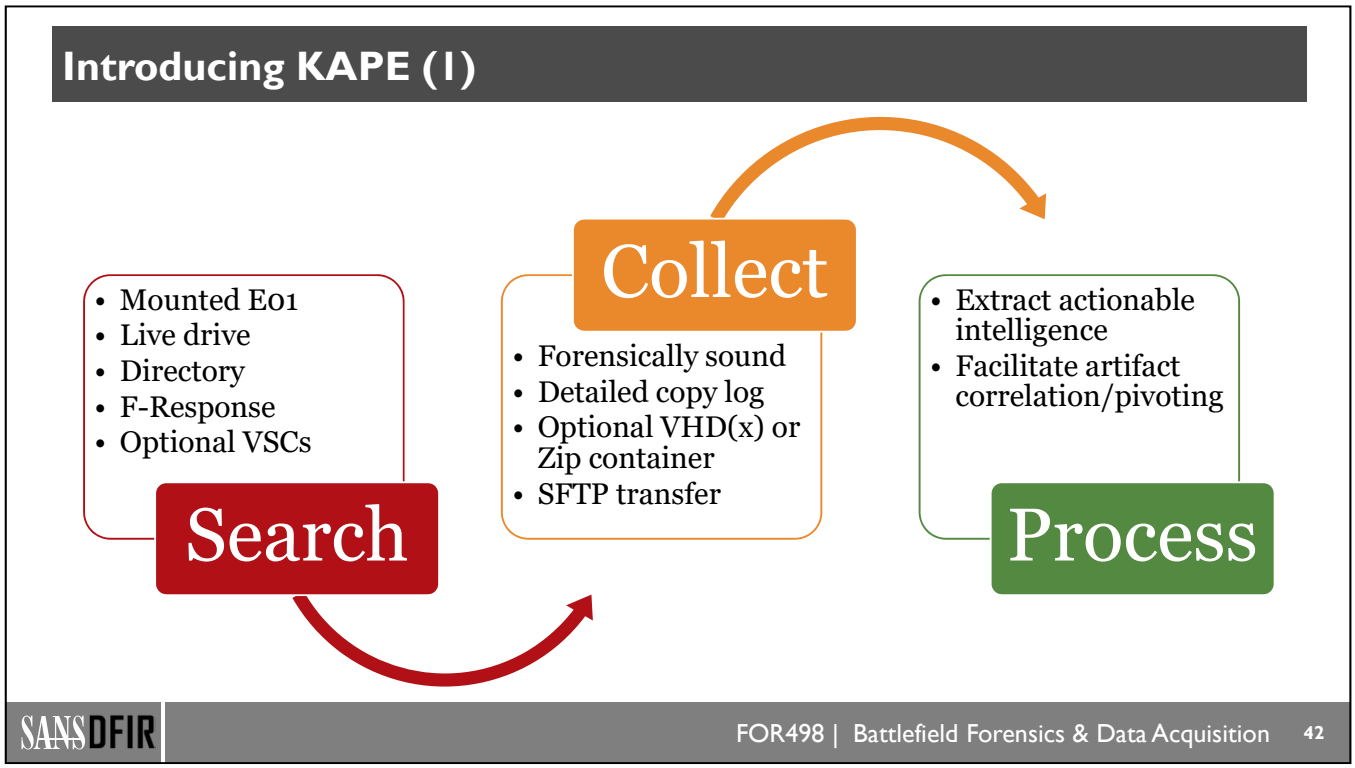
This module will explain what KAPE can do, how it works, and most importantly, why you want to use it.

KAPE, or Kroll Artifact Parser and Extractor, is a piece of software written by Eric Zimmerman. KAPE was written after years of experience spent in the trenches working cases against a wide range of data sources, including phones, computers, and loose media. All of the lessons learned from processing thousands of machines in both a live response and dead box fashion went into the design and implementation found in KAPE. KAPE strives to be a tool that can be tailored to YOUR needs without requiring constant tweaking from a programmer. With KAPE, you get to decide what is collected, when it is collected, and optionally, if any of the collected data is further processed using the programs of your choosing.

Many options exist when it comes to triaging computers and phones, and while many perform certain functions very well, like copying files from a target, they often have weaknesses such as not preserving metadata, the inability to copy locked files and alternate data streams, and rigid specifications when it comes to the kinds of files to collect. KAPE does not suffer from any of these issues and is blazing fast too!

In and of itself, KAPE knows how to do very little. In fact, without some help, it cannot do anything at all! Believe it or not, KAPE was designed this way intentionally. Think of KAPE as a powerful engine that can move your case from point A to point B very rapidly, but only once you provide the “gas” in terms of simple to create and modify configuration files that instruct KAPE on what to collect. By decoupling a list of files to copy from the program itself, KAPE becomes a tool that brings significant benefit to any kind of case that involves digital storage. Furthermore, it allows you, the investigator, the person who knows your case and needs better than anyone, to tailor KAPE to meet your very specific needs.

In the next sections, we will learn how to effectively wield KAPE in any investigation you find yourself a part of, to provide you actionable intelligence in 90 minutes or less (often, much less!). Let’s begin!



Why use KAPE?

KAPE is primarily a triage program that will target a device, find the most forensically relevant artifacts, and parse them within a few minutes. Because of its speed, KAPE also allows investigators to find and prioritize the more critical systems in their case. Additionally, KAPE can be used to collect the most critical artifacts at the start of the traditional imaging process. While the imaging completes, the data generated by KAPE can be reviewed for leads, building timelines, etc.

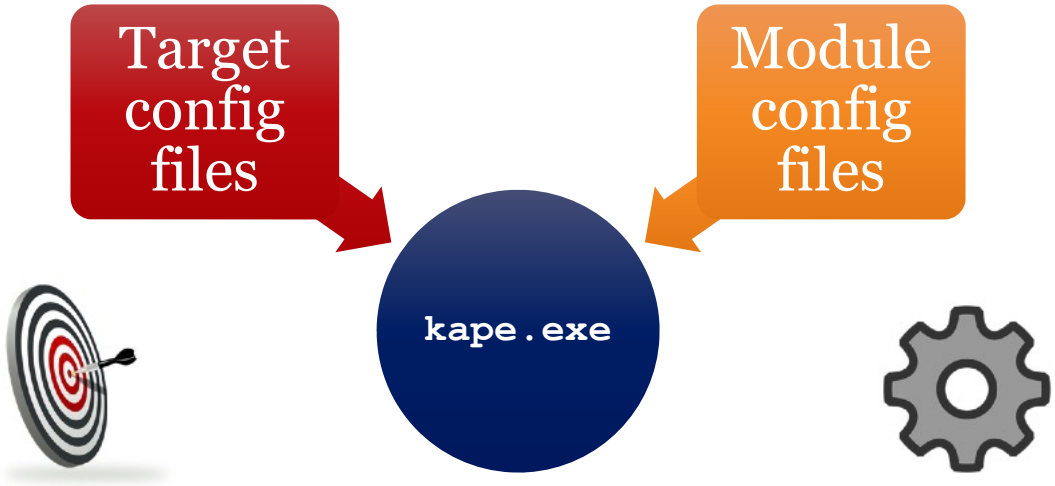
What is it?

KAPE is a multi-function program that serves two primary functions: 1) collect files and 2) process collected files with one or more programs.

In and of itself, KAPE does not know how to do anything related to the two functions above. What KAPE does know how to do however, is read configuration files on the fly, and based on the contents of these files, collect and (optionally) process files. This makes KAPE very extensible, in that the program's author does not need to be involved to add functionality.

At a high level, KAPE works by adding file masks to a queue. This queue is then used to find and copy out files from a source location. For files that are locked by the operating system, a second pass takes place that knows how to bypass the locking. At the end of the process, KAPE will make a copy and preserve metadata about all available files from a source drive into a given directory. The second (optional) stage of processing is to run one or more programs against the collected data. This too works by either targeting specific file names or directories. Various programs are run against the files, and the output from the programs is then saved in directories named after a category, such as EvidenceOfExecution, BrowserHistory, AccountUsage, and so on. By grouping things by category, examiners of all levels have a means to discover relevant information regardless of the individual artifact that a piece of information came from. In other words, it is no longer necessary for an examiner to know how to process prefetch, shimcache, amcache, userassist, and so on as it relates to evidence of execution artifacts. Depending on the experience level of the examiner, they may not even know about half of those artifacts. By thinking categorically and grouping output in the same way, a wider range of artifacts can be brought to bear for any given requirement.

Introducing KAPE (2)



KAPE is powered by targets and modules. Both targets and modules are defined using YAML (YAML Ain't Markup Language). The syntax for creating targets and modules is both flexible and easy to use. KAPE comes with dozens of preconfigured targets and modules that serve as examples to create your own targets and modules. The included configurations can also be used as is and have been designed and tested with thoroughness and accuracy in mind. By following along with the included files, you can quickly extend KAPE to meet any additional requirements you have.

Targets

Target configurations contain a list of file masks that allow for finding various types of items on a storage device. Targets allow for collecting files by full name, extension, or even an entire directory structure (with optional recursive support).

Modules

Module configurations contain information about a program to execute, including command line arguments, export format, etc. Modules are responsible for running commands either against a live system or against files collected by one or more target configurations.

Introducing KAPE: Target Configurations

- A collection of targets
  - Single files
  - Entire directory structures
  - Files based on extension
- Target configs should be
  - Specific
  - Granular
  - Focused

```
Description: File system metadata
Author: Eric Zimmerman
Version: 1
Id: cf748dbc-1c5a-4ae9-8f68-4aba6505e181
RecreateDirectories: true
Targets:
    -
        Name: $MFT
        Category: FileSystem
        Path: C:\$MFT
        IsDirectory: false
        Recursive: false
        AlwaysAddToQueue: true
        Comment: ""
    -
        Name: $LogFile
        Category: FileSystem
        Path: C:\$LogFile
        IsDirectory: false
        Recursive: false
        AlwaysAddToQueue: true
        Comment: ""
```

Targets are essentially collections of file and directory specifications. KAPE knows how to read these specifications and expand them to files and directories that exist on a target system. Once KAPE has processed all targets and has built a list of files, the list is processed, and each file is copied from the source to the destination directory.

For files that are locked by the operating system and therefore not able to be copied by regular means, the file is added to a secondary queue. This secondary queue contains all the files that were locked or in use. After the primary queue is processed, the secondary queue is processed and a different technique, using raw disk reads, is used to bypass the locks. This results in getting a copy of the file as it exists on the hard drive, even for files that are currently in use.

Regardless of how the file is copied (either regularly or via raw access), the original timestamps from all directories and the files themselves are reapplied to the destination files. The metadata is also collected into log files, both in text form and CSV.

Targets should include specific files, or groups of files, that are related. For example, a target that collects jumplists can collect both automatic and custom jumplists. A target that collects lnk files can collect any files that match *.lnk. While you can put both of these file types together, it is better to keep targets specific and granular to a single artifact and then combine the targets into a new target that contains both the granular targets.

Exceptions to this rule would be things like file system data where the $MFT, $LogFile, and $J are often needed and used for most analysis involving the NTFS file system.

By keeping targets granular, you gain the flexibility to create other targets that reference these granular targets on an ‘as needed’ basis.

The notion of a target configuration being focused means that a path to a given artifact is preferred, as this results in much faster processing. For example, while it is possible to start at the root of a drive (C:\ as an example) and find all *.lnk files, it is better to add the directories where lnk files are generally found, including each user's Recent folder and the Desktop, as this allows for KAPE to traverse a data source much faster than walking the entire directory structure.

Target files can contain wildcards which allow for collecting a given set of files from all users on a drive. This makes it easy to collect all ntuser.dat hives, or PST files for all users, regardless of how many there are.

Introducing KAPE: Module Configurations

- A collection of processors
  - Allow for different export formats
- Module configs must only call a single executable
- Modules belong to a category
  - Categories group related artifact types together

```
Description: 'JLECmd: process jumplist files'
Category: FileFolderOpening
Author: Eric Zimmerman
Version: 1
Id: 81fe4336-eb10-4733-a770-cb57ec9bd108
ExportFormat: csv
Processors:
    -
        Executable: JLECmd.exe
        CommandLine: -d %sourceDirectory% --csv %destinationDirectory% -q
        ExportFormat: csv
    -
        Executable: JLECmd.exe
        CommandLine: -d %sourceDirectory% --html %destinationDirectory% -q
        ExportFormat: html
    -
        Executable: JLECmd.exe
        CommandLine: -d %sourceDirectory% --json %destinationDirectory% -q
        ExportFormat: json
```

Like targets, modules are defined using simple YAML [1] properties and are used to run programs. These programs can target anything, including files collected via the target capabilities as well as any other kinds of programs you may want to run on a system from a live response perspective. For example, if you collected jumplists, a tool like JLECmd could be used to dump the contents of the jump lists to CSV.

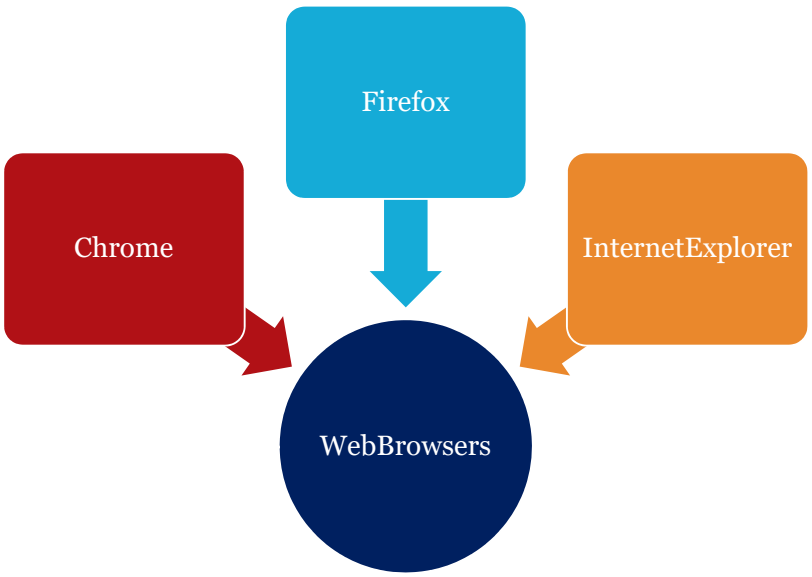
If you also wanted to collect the output of "**`netstat.exe`**" or "**`ipconfig /dnscache`**", you could do so. Each of these options would be contained in its own module and then grouped together based on commonality between the modules, such as "Network Live Response", for example.

Each module must only reference a single executable. Each defined processor for the same executable allows for the module to export data in different formats like CSV, HTML, or JSON. This makes KAPE able to generate data that can be passed on to a big data system such as SOF-ELK® or Splunk, loaded into Timeline Explorer for an analyst to review, or HTML for reporting findings to less technical users.

The other aspect of a module that is very important to understand is that each module is tied to a category. While the category is something you as an end user controls, it is recommended to stick with common categories such as evidence of execution, file and folder opening, browser history, etc. The SANS red poster is an excellent place to start for such things. [2]

[1] The Official YAML Web Site | `https://for498.com/u39nk`
[2] SANS - Information Security Resources | `https://for498.com/v1rb2`

Introducing KAPE: Compound Targets and Modules




```
Targets:
    -
        Name: Internet Explorer
        Category: Communications
        Path: InternetExplorer.tkape
        IsDirectory: false
        Recursive: false
        Comment: ""
    -
        Name: Chrome
        Category: Communications
        Path: Chrome.tkape
        IsDirectory: false
        Recursive: false
        Comment: ""
    -
        Name: Firefox
        Category: Communications
        Path: Firefox.tkape
        IsDirectory: false
        Recursive: false
        Comment: ""
```

SANS DFIR | FOR498 | Battlefield Forensics & Data Acquisition 47

Recall that target files should be granular and specific. Also recall that modules should contain a single executable.

This keeps things simple and allows for specific collections and processing, but when multiple targets or modules need to be run, it would become tedious to have to run each target and module individually.

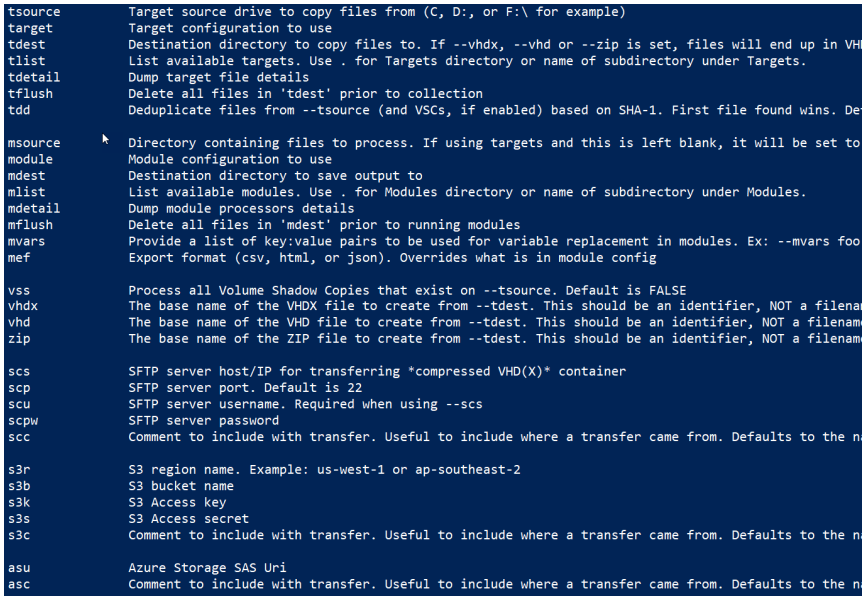
Enter compound targets and modules!

The idea here is to create a new target or module and then simply reference other targets or modules within them, including the extension. When KAPE sees this, it processes each target or module and unpacks it. There is no limit to the number of levels that can be present in compound files.

This capability is efficient because of the unique IDs that are present in each target and module. As KAPE runs targets and modules, it keeps track of each ID that is found. If a target or module is found in more than one place, KAPE makes sure that each target or module is run only once, resulting in faster processing times while avoiding redundant copying of the same thing.

You can see this behavior in action if you run the **!All** target. The **!All** target simply runs all available targets against a given source, including compound targets and the targets that are referenced in the compound target. In the case of **!All**, KAPE would run the **WebBrowsers** target as well as each individual web browser target found within it. By tracking the IDs in each, KAPE ensures that the collection for each browser's artifacts only happens once.

Introducing KAPE: Command Line Switches



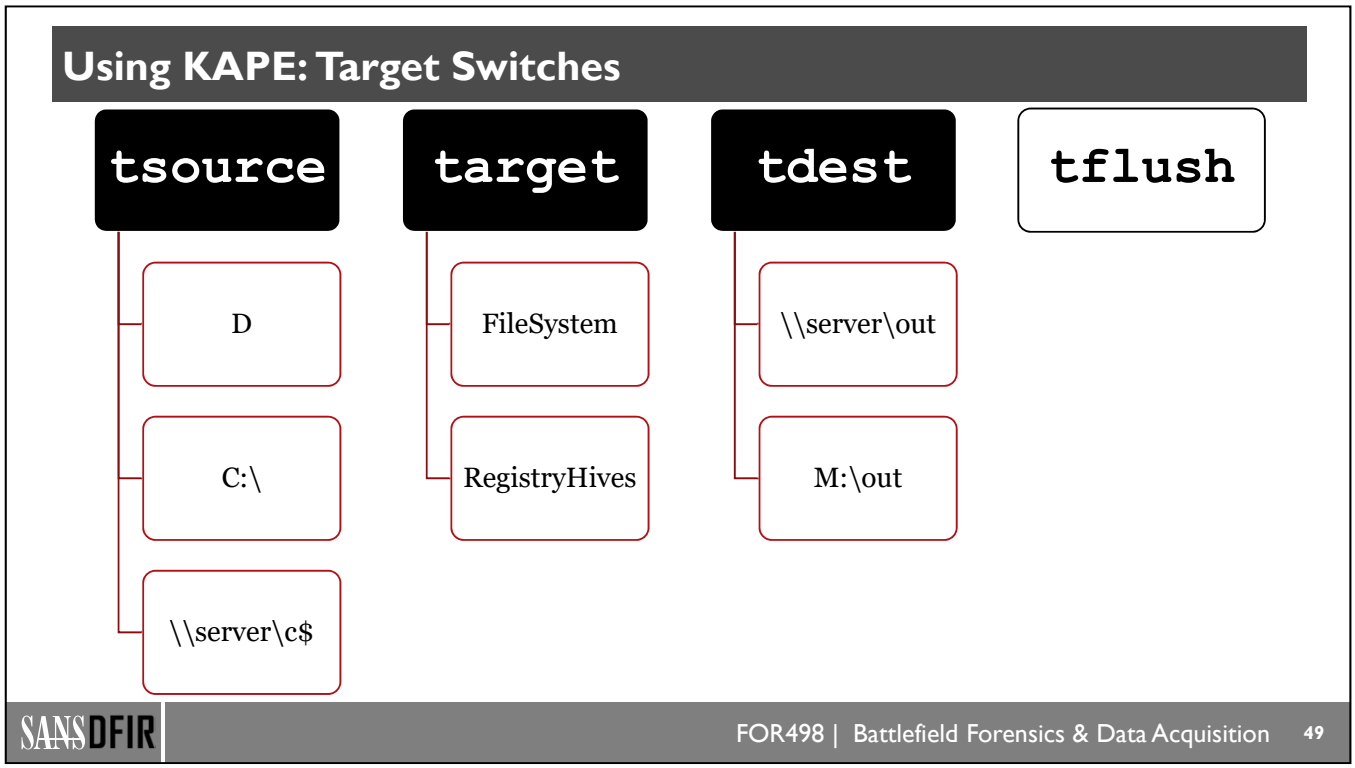
```
tsource         Target source drive to copy files from (C, D:, or F:\ for example)
target          Target configuration to use
tdest           Destination directory to copy files to. If --vhdx, --vhd or --zip is set, files will end up in VH
tlist           List available targets. Use . for Targets directory or name of subdirectory under Targets.
tdetail         Dump target file details
tflush          Delete all files in 'tdest' prior to collection
tdd             Deduplicate files from --tsource (and VSCs, if enabled) based on SHA-1. First file found wins. De

msource         Directory containing files to process. If using targets and this is left blank, it will be set to
module          Module configuration to use
mdest           Destination directory to save output to
mlist           List available modules. Use . for Modules directory or name of subdirectory under Modules.
mdetail         Dump module processors details
mflush          Delete all files in 'mdest' prior to running modules
mvars           Provide a list of key:value pairs to be used for variable replacement in modules. Ex: --mvars foo
mef             Export format (csv, html, or json). Overrides what is in module config

vss             Process all Volume Shadow Copies that exist on --tsource. Default is FALSE
vhdx            The base name of the VHDX file to create from --tdest. This should be an identifier, NOT a filena
vhd             The base name of the VHD file to create from --tdest. This should be an identifier, NOT a filenam
zip             The base name of the ZIP file to create from --tdest. This should be an identifier, NOT a filenam

scs             SFTP server host/IP for transferring *compressed VHD(X)* container
scp             SFTP server port. Default is 22
scu             SFTP server username. Required when using --scs
scpw            SFTP server password
scc             Comment to include with transfer. Useful to include where a transfer came from. Defaults to the n

s3r             S3 region name. Example: us-west-1 or ap-southeast-2
s3b             S3 bucket name
s3k             S3 Access key
s3s             S3 Access secret
s3c             Comment to include with transfer. Useful to include where a transfer came from. Defaults to the n

asu             Azure Storage SAS Uri
asc             Comment to include with transfer. Useful to include where a transfer came from. Defaults to the n
```


As KAPE is a command line tool, there are a quite a few switches to become familiar with in order to become proficient at using KAPE effectively. Do not be overly concerned with how many options there are. For most operations, only three to six are needed, depending on what you want to do. While the image above contains many of the more commonly used options, there are even more switches available in KAPE. For a full list, see the full documentation at https://ericzimmerman.github.io/KapeDocs/

Based on testing and experience, the defaults in KAPE are what most people will use most of the time. You can see the default values reflected in the descriptions of the command line switches.

In some cases, things that are off by default will need to be enabled. This is done by simply including the switch on the command line. For example, to include volume shadow copies (VSC), add **`--vss`** to the command line, which enables VSC processing. However, to disable something that is enabled by default, pass in a value of false along with the switch (**`--zv false`** for example).

Again, the defaults are what most people should use except for wanting to process volume shadow copies, but the options are there to allow for you to adjust how KAPE works in a wide variety of use case scenarios.



Required target switches are shown in the top row, using black boxes with white text. Optional switches are in the top row, using white boxes with black text. These are the switches that are used for running targets against a source. To see all available targets, use **`--tlist`** and/or **`--tdest`**

Target switches

`tsource`
The drive letter to search. This should be formatted as C, C:, or C:\

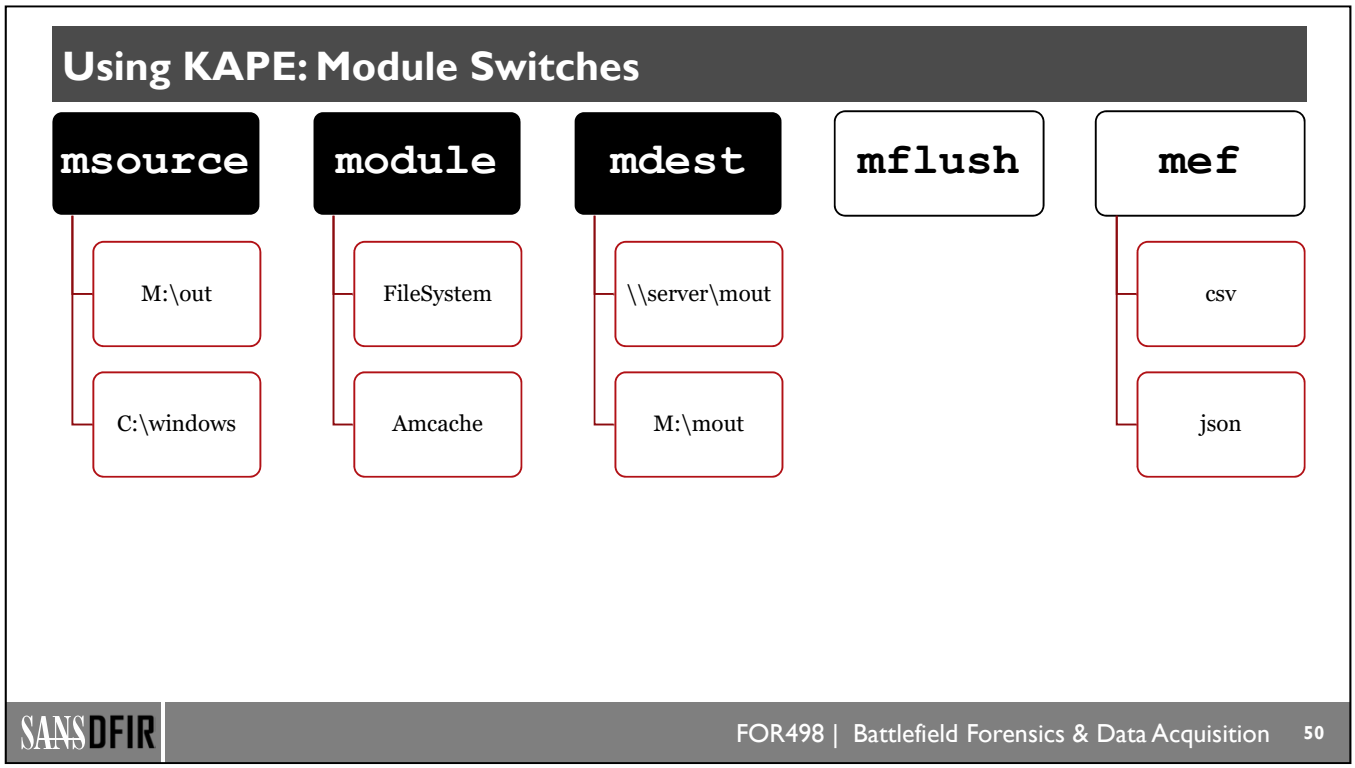
`target`
The target configuration to run, without the extension. Get a list of all available targets with the **`-tlist`** switch.

`tdest`
The directory where files should be copied to. This directory will be created if it does not exist. This can be a regular directory or a UNC path. It should of course be writable.

`tlist`
Displays available targets including the name, description, author, etc.

`tdetail`
Like **`--tlist`**, but also includes details on each target including file paths, category, etc. This is used along with **`--tlist`** and not by itself.

`tflush`
When true, deletes the directory specified by **`--tdest`** (if it exists) before searching.



Required module switches are shown in the top row, using black boxes with white text. Optional switches are in the top row, using white boxes with black text. These are the switches that are used for running modules against a source. To see all available modules, use **`--mlist`** and/or **`--mdest`**

Module switches

`msource`
The directory containing files to process. This is usually **`--tdest`** when run in conjunction with target switches.

`module`
The module configuration to run, without the extension. Get a list of all available modules with the **`--mlist`** switch.

`mdest`
The directory where output from processors should be created. This directory will be created if it does not exist. This can be a regular directory or a UNC path. This serves as the root directory, and additional subdirectories will be created based on the categories found in module configurations.

`mlist`
Displays available modules including the name, description, author, etc.

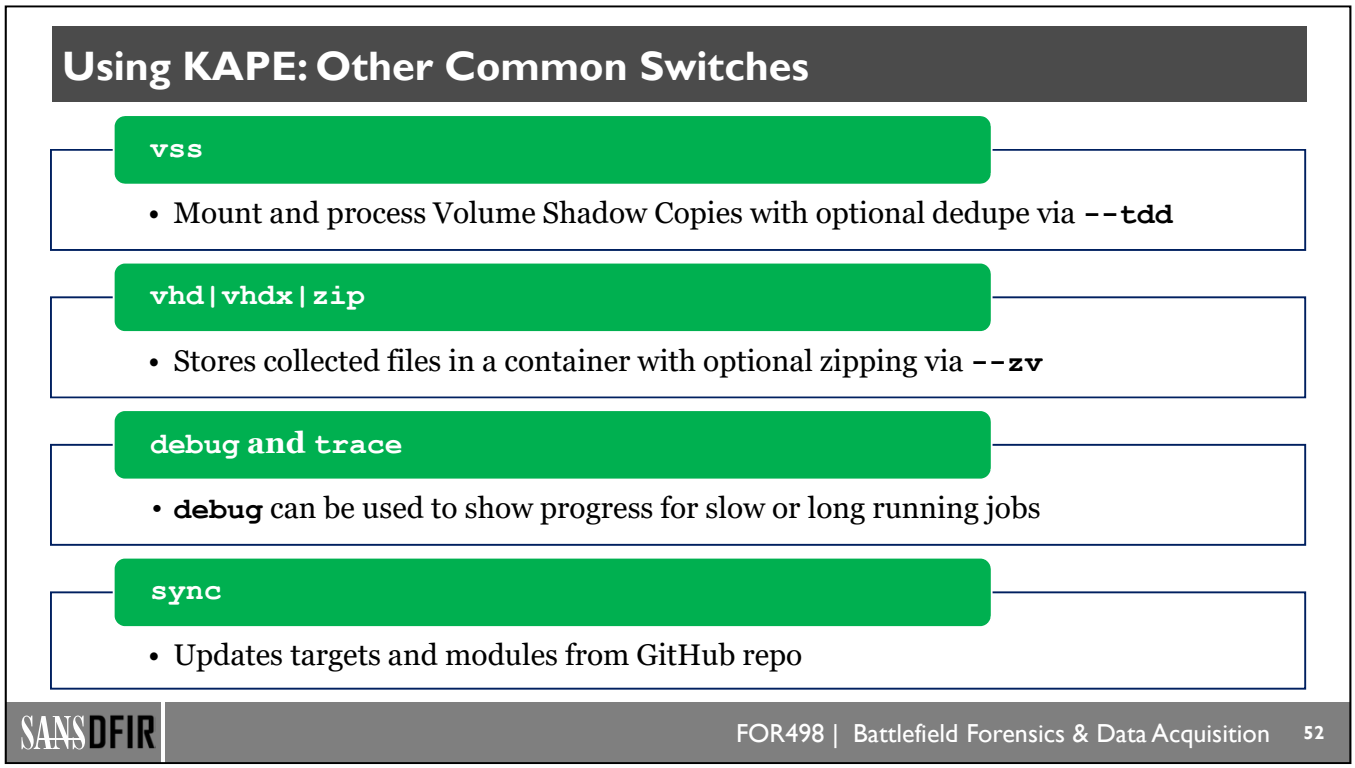
`mdetail`
Like **`--mlist`**, but also includes details on each module including command lines, category, etc. This is used along with **`--mlist`** and not by itself.

`mflush`

When true, deletes the directory specified by **`--mdest`** (if it exists) before processing files.

`mef`

The export format to use (csv, txt, json, etc.) when running processors. This overrides the default in the module configuration. If the format does not exist, the first available processor (i.e. the default) will be chosen.



Other switches

vss
Find, mount, and search all available Volume Shadow Copies on **--tsource**. Default is FALSE.

tdd
Deduplicate files from **--tsource** and VSCs based on their SHA-1 value. The first file found during target processing wins. This is generally the "active" file vs one from a VSC. Set to false to copy all files from **--tsource** and VSCs.

Vhdx (vhd and zip work the same way)
Creates a VHDX virtual hard drive from the contents of **--tdest**. This switch is used as a part of the file name for the VHDX file. For example, if "**--target EvidenceOfExecution --vhdx EricWorkstation**" was run, the corresponding VHDX file would be named something like:

2018-09-03T144144_EvidenceOfExecution_EricWorkstation.vhdx

zv
If true, zip the VHDX file, resulting in a (usually) MUCH smaller file. Default is TRUE.

zm
If true, zip the contents of **--mdest**. Default is FALSE.

debug
When true, enables debug messages. This will result in more details as to what KAPE is doing in the background as it runs.

trace
When true, enables trace messages. This will result in an even larger amount of information to be generated about what KAPE is doing as it runs.

Using KAPE: Collecting Files to USB

USB drive assigned drive letter L:

```
kape.exe --tsource c --tdest L:\collect --target EvidenceOfExecution --tflush

kape.exe --tsource c --tdest ..\collect --target EvidenceOfExecution --tflush
```


In this example, a live system serves as the source of the data. Because we must introduce KAPE to the computer, a USB drive is connected to the system. Once the drive has been assigned a drive letter (in our example above, it is drive letter **`L:`**), KAPE is used to collect evidence of execution artifacts and place them into the **`"L:\collect"`** directory. Note in the second command, a relative path is used which is useful because you may not always know the drive letter assigned to external storage!

```
"kape.exe --tsource c --tdest L:\collect --target EvidenceOfExecution --
tflush"
```

First, the **`"L:\collect"`** directory is deleted (should it exist) to ensure we only have files from this machine once we run KAPE. The **EvidenceOfExecution** target is used, which collects these files:

CONTINUED ON NEXT PAGE

Target: EvidenceOfExecution

Description: Evidence of execution related files

Author: Eric Zimmerman

Version: 1.0

Total targets: 4

Target details:

Name: Prefetch

Category: Prefetch

Path: C:\Windows\prefetch

Name: RecentFileCache

Category: ApplicationCompatability

Path: C:\Windows\AppCompat\Programs\RecentFileCache.bcf

Name: Amcache

Category: ApplicationCompatability

Path: C:\Windows\AppCompat\Programs\Amcache.hve

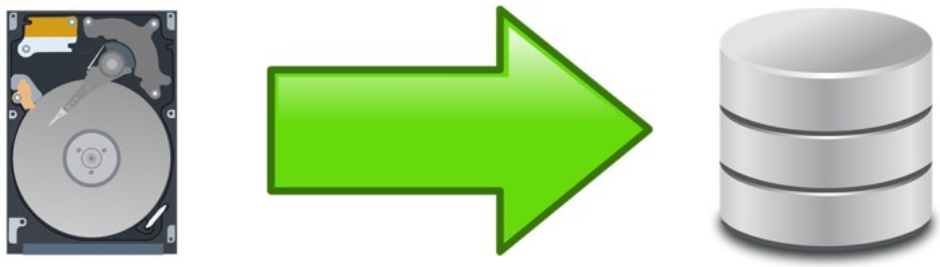
Name: Amcache transaction files

Category: ApplicationCompatability

Path: C:\Windows\AppCompat\Programs\Amcache.hve.LOG*

KAPE searches the source drive **`C:\`**, and locates any files matching the above specifications. Once all files are located, they are copied to **`"L:\collect"`**, along with various metadata (timestamps, hashes, source and destination paths, etc.)

Using KAPE: Collecting Files to VHDX



Write blocked hard drive assigned drive letter F:

```
kape.exe --tsource f --tdest C:\collect --target RegistryHives --vhdx Hives --zv
false
```


In this example, a write blocked hard drive serves as the source of the data. Once you have connected the hard drive to your write blocker of choice and the target has been assigned a drive letter (in our example above, its drive letter **`F:`**), KAPE is used to collect Registry hives and place them into a VHDX container with this command:

```
"kape.exe --tsource f --tdest C:\collect --target RegistryHives --vhdx
Hives --zv false"
```

This results in a file being created in **C:\collect** with a name like

2020-01-11T143322_Hives.vhdx

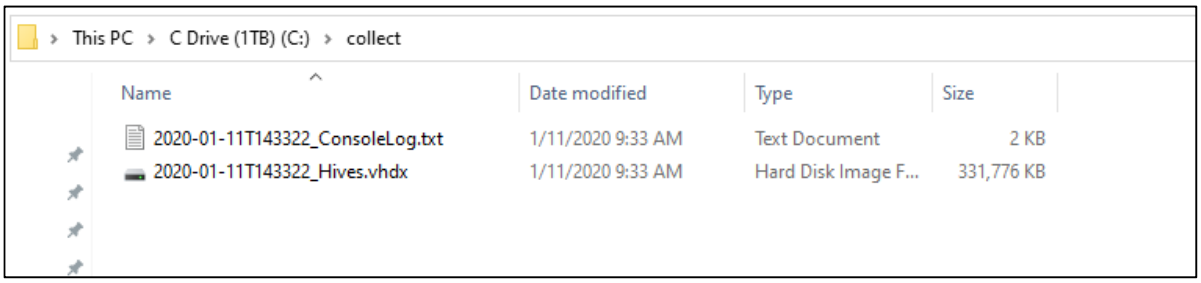
which will contain all the Registry hives found on volume "F". KAPE determines how big to make the VHDX file on the fly.

Because we want to mount the container next, the **`--zv`** option is used, which prevents KAPE from zipping the VHDX container, and thereby saving us a step down the road.

Note that the source of the data could have been any other drive letter or directory as well. While this is illustrative of a common technique used to extract data from a hard drive, the same commands could be used against any source, such as a live system, an F-Response mounted device, and so on.

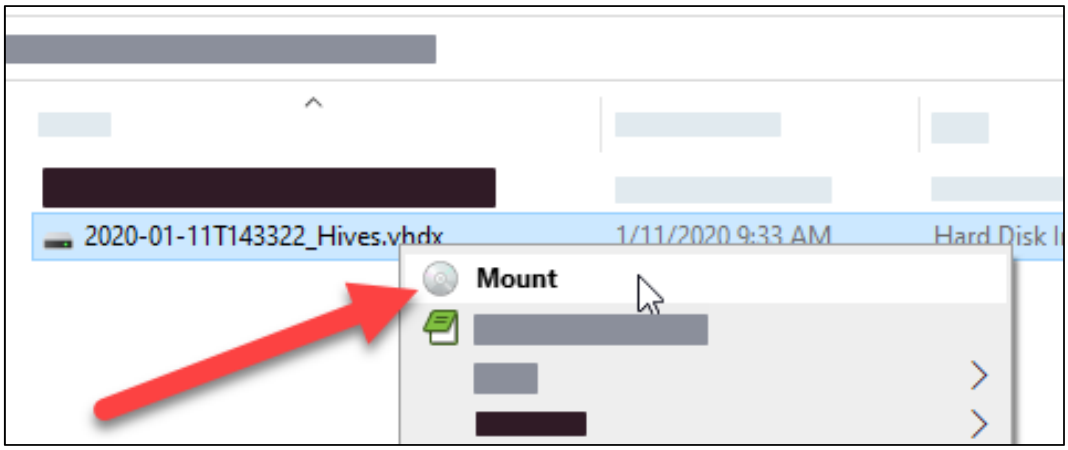
In the next step we will see several options on how we can go about accessing the data KAPE has placed in the VHDX container.

Using KAPE: Accessing the VHDX (1)



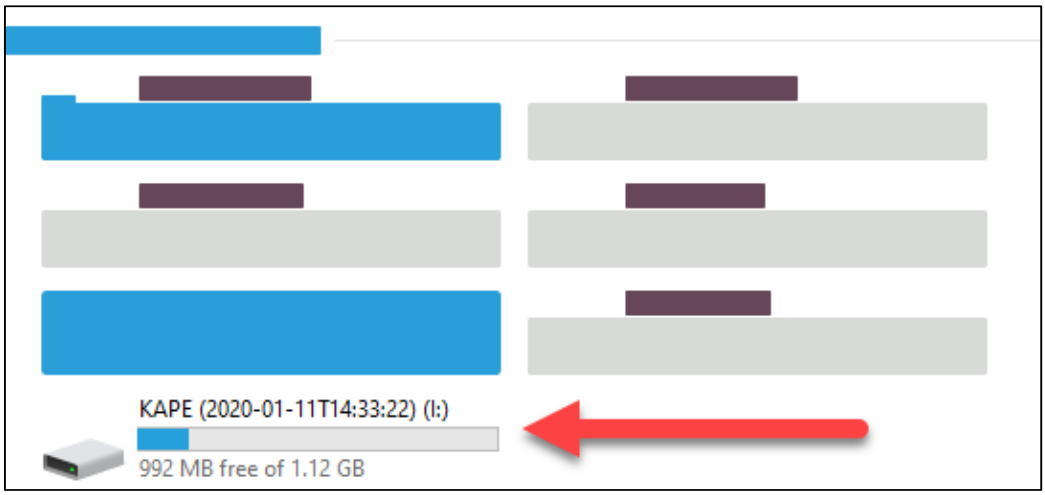
To get access to the VHDX file, it needs to be mounted. The simplest way to access it is to simply double click on the VHDX file using File Explorer.

Using KAPE: Accessing the VHDX (2)



This will mount the VHDX as a new drive letter which will then contain all the files copied as well as the various log files. You can also right click on the VHDX and choose **Mount** from the context menu.

Using KAPE: Accessing the VHDX (3)



The label for the mounted drive will contain a timestamp based on when the collection occurred (UTC time).

NOTE: The first time the VHDX is mounted, it MUST be done in non-read-only fashion. Once this is done, it can then subsequently be mounted as read-only using PowerShell, which we will see an example of below.

To unmount an image, right click on the new drive letter and choose **Eject** from the context menu.

Once a VHDX file has been initially mounted in a non-read only fashion, PowerShell can be used to mount the VHDX file as read-only using the **Mount-DiskImage** [1] command:

```
"Mount-DiskImage -ImagePath C:\collect\2020-01-11T143322_Hives.vhdx -access ReadOnly"
```

The Mount-DiskImage help is shown below.

NAME
Mount-DiskImage

More options and instructions are shown on the next page.

SYNTAX
Mount-DiskImage
[-ImagePath] <String[]>
[-Access {Unknown | ReadWrite | ReadOnly}]
[-CimSession <CimSession[]>]
[-NoDriveLetter]
[-PassThru]
[-StorageType {Unknown | ISO | VHD | VHDX | VHDSet}]
[-ThrottleLimit <Int32>]
[-Confirm]
[-WhatIf]
[<CommonParameters>]

DESCRIPTION
The Mount-DiskImage cmdlet mounts a previously created disk image (virtual hard disk or ISO), making it appear as a normal disk. This cmdlet requires the full path of the VHD or ISO file. If the file is already mounted, then the cmdlet will display the following error.

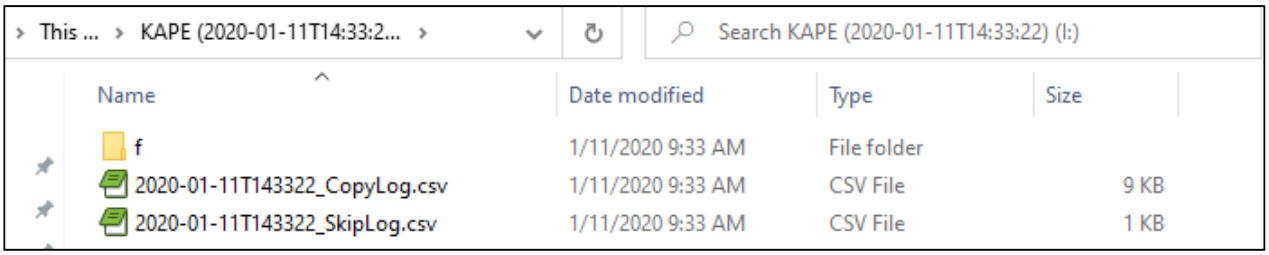
-- "The process cannot access the file because it is being used by another process."

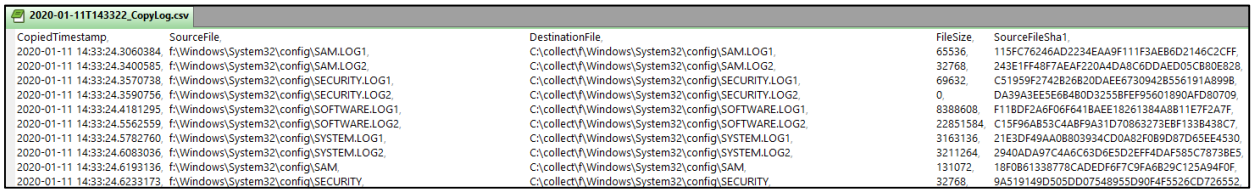
To mount a VHD file, Administrator privilege is required. Administrator privileges are not needed to mount an ISO file on Windows 8. On Windows Server 2012, only an administrator is allowed to mount or eject an ISO file.

To create and mount a VHD on a computer running Hyper-V, use the New-VHD and Mount-VHD cmdlets in the Hyper-V module (which is included in Windows 8 and Windows Server 2012 but not enabled by default). Alternatively, open Disk Management and then choose Create VHD from the Action menu.

[1] Mount-DiskImage | `https://for498.com/yfc2a`

Using KAPE: Accessing the VHDX (4)





2020-01-11T143322_CopyLog.csv

CopiedTimestamp	SourceFile	DestinationFile	FileSize	SourceFileSha1
2020-01-11 14:33:24.3060384	f:\Windows\System32\config\SAM.LOG1	C:\collect\f\Windows\System32\config\SAM.LOG1	65536	115FC76246AD2234EAA9F111F3AEB6D2146C2CFF
2020-01-11 14:33:24.3400585	f:\Windows\System32\config\SAM.LOG2	C:\collect\f\Windows\System32\config\SAM.LOG2	32768	243E1FF48F7AEAF220A4DA8C6DDAED05CB80E828
2020-01-11 14:33:24.3570738	f:\Windows\System32\config\SECURITY.LOG1	C:\collect\f\Windows\System32\config\SECURITY.LOG1	69632	C51959F2742B26B20DAEE6730942B556191A899B
2020-01-11 14:33:24.3590756	f:\Windows\System32\config\SECURITY.LOG2	C:\collect\f\Windows\System32\config\SECURITY.LOG2	0	DA39A3EE5E6B4B0D3255BFEF95601890AFD80709
2020-01-11 14:33:24.4181295	f:\Windows\System32\config\SOFTWARE.LOG1	C:\collect\f\Windows\System32\config\SOFTWARE.LOG1	8388608	F11BDF2A6F06F641BAEE18261384A8B11E7F2A7F
2020-01-11 14:33:24.5562559	f:\Windows\System32\config\SOFTWARE.LOG2	C:\collect\f\Windows\System32\config\SOFTWARE.LOG2	22851584	C15F96AB53C4ABF9A31D70863273EBF133B438C7
2020-01-11 14:33:24.5782760	f:\Windows\System32\config\SYSTEM.LOG1	C:\collect\f\Windows\System32\config\SYSTEM.LOG1	3163136	21E3DF49AA0B803934CD0A82F0B9D87D65EE4530
2020-01-11 14:33:24.6083036	f:\Windows\System32\config\SYSTEM.LOG2	C:\collect\f\Windows\System32\config\SYSTEM.LOG2	3211264	2940ADA97C4A6C63D6E5D2EFF4DAF585C7873BE5
2020-01-11 14:33:24.6193136	f:\Windows\System32\config\SAM	C:\collect\f\Windows\System32\config\SAM	131072	18F0B61338778CADEDF6F7C9FA6B29C125A94F0F
2020-01-11 14:33:24.6233173	f:\Windows\System32\config\SECURITY	C:\collect\f\Windows\System32\config\SECURITY	32768	9A519149D505DD07548955D90F4F5526CD726552

VHDX contains all files copied plus **CopyLog & SkipLog** CSV files

You are now able to see the contents of the VHDX in the same manner as you would use to navigate any other collection.

To unmount the VHDX, right click on the drive letter and choose “Eject” from the context menu.

Using KAPE: Collecting and Processing Files

E01 mounted using Arsenal Image Mounter* and assigned drive letter T:

```
kape.exe --tsource t --tdest C:\collect --target ExecutionAndLnkJump --vss
--msource c:\collect --module ExecutionAndLnkJump --mdest c:\process --mflush
```

* Arsenal Image Mounter emulates a physical disk, which gives us VSC access

In this example, we have a similar situation to our last, except rather than having a write blocked hard drive, we start with an E01 image. In this scenario, using Arsenal Image Mounter (AIM) is the best choice because as is mentioned above, AIM emulates a physical disk, which means Windows will also expose any Volume Shadow Copies (VSC) the E01 may contain.

This example includes the **`--vss`** switch, which will not only find the files of interest from the T: drive, but also finds, mounts, and deduplicates (based on SHA-1) any other files of interest from all the VSCs that are present. This all happens automatically by simply using the **`--vss`** switch!

Under the hood, KAPE mounts each available VSC to a symbolic link, then searches each one after the primary source drive is searched. That way, any files found on the primary drive are collected first, and any deduplication happens against what comes out of the VSCs.

The other set of options that is new in this example are the module related switches. Notice how the source of the module (**`msource`**) is the destination for the target (**`tdest`**). This chaining allows you to both collect AND process in the same operation.

It is generally a best practice to save the module results to a different folder than where the raw artifacts were found. In the above example, all processed output would be saved to **c:\process**. We will look at how the processing results are organized soon.

Using KAPE: Processing Files Only

Existing directory with file system related artifacts

```
kape.exe --msource D:\temp --mdest D:\filesystem --module MFTECmd --mflush
```


Finally, let's look at an example of how to process files that exist in a given location. KAPE does not require you to use both target and module operations at the same time (but one or the other at a minimum must be used). This allows for flexibility in designing pipelines for processing.

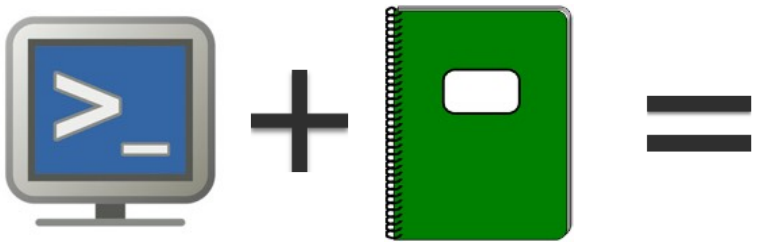
In this example, we have a similar situation to our last, but our files of interest have been extracted via other means to the **D:\temp** directory (icat, X-Ways, Encase, etc.). For the purposes of this example, let's say that there is at least a **$MFT** file in **D:\temp**.

Because we want to parse the contents of the **$MFT** file, we use the **MFTECmd** module. This module will call **MFTECmd.exe** and save the CSV output to the **D:\filesystem** directory. This CSV can then be loaded into Timeline Explorer and analyzed.

Depending on what else is present in **D:\temp**, other modules could be run to process other files of interest.

While it is possible (in some cases) to run KAPE module options against a running system, this would not work for every kind of file, since files like **$MFT** would not be available in a live environment. Locked files would also present a problem in this situation (Registry hives, event logs, etc.). This is precisely why KAPE can both collect and process things; because it solves this problem related to locked files and triage/live response.

Using KAPE: Collection Related Log Files



- Console log contains everything displayed on the screen during KAPE execution
- **`CopyLog`** contain detailed metadata about files copied
- **`SkipLog`** contain detailed metadata about files skipped

While KAPE is executing, a lot of things are happening, often at high speed. Because of this, KAPE logs everything that is displayed during execution, to a text file which starts being written in the same directory as the KAPE executable. By default, this console log will be copied to the destination directories used by KAPE as a record of everything that was done during a run. There are two options that affect the amount of data displayed during execution, **`--debug`** and **`--trace`**. These options can be used individually or together and both will result in more detail being displayed to the console, and subsequently, to the console log file.

Debug can also be used to see progress for slow links, as when it is enabled, each file copied will be logged individually whereas without debug, each copy operation is silent. Trace on the other hand, will result in much more detailed information, including things like hashing, adding data to CSV records, deleting files after zipping, deduplication information, and so on. Trace is useful for troubleshooting and to get a better understanding of the kinds of things KAPE is doing while running.

Both **`--debug`** and **`--trace`** can be used during validation of both target and module files you create. This allows you to observe exactly what files are found and added to the queue, as well as what processors were located based on file extensions, and so on.

In addition to the console log, when using the target options, all files copied and/or skipped by KAPE have their metadata tracked and recorded into CSV files.

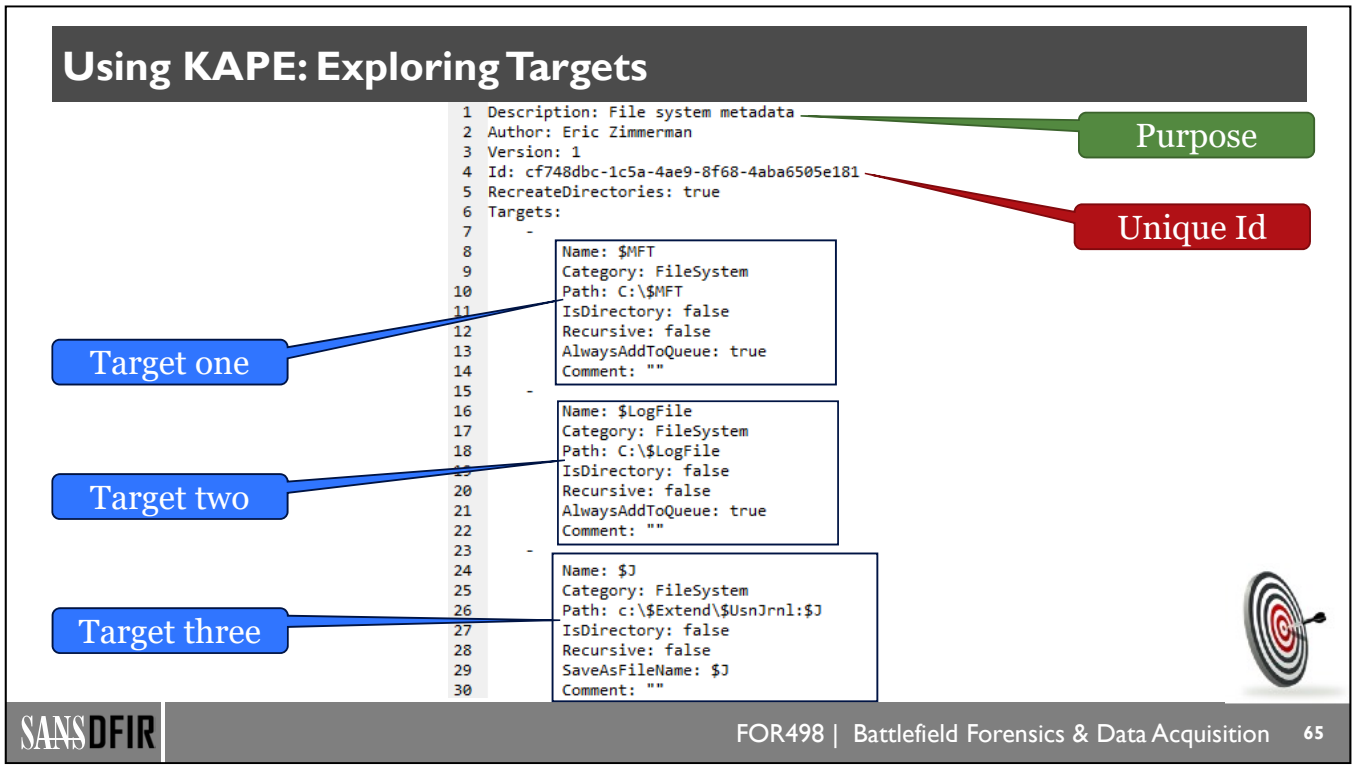
An example of the first few lines of the CopyLog CSV might look like this:

```
CopiedTimestamp,SourceFile,DestinationFile,FileSize,SourceFileSha1,Deferred
Copy,CreatedOnUtc,ModifiedOnUtc,LastAccessedOnUtc,CopyDuration
2020-01-11
14:33:24.3060384,f:\Windows\System32\config\SAM.LOG1,C:\collect\f\Windows\S
ystem32\config\SAM.LOG1,65536,115FC76246AD2234EAA9F111F3AEB6D2146C2CFF,Fals
e,2017-09-29 08:45:11.8519987,2017-09-29 08:45:11.8519987,2017-09-29
08:45:11.8519987,00:00:00.0060327
2020-01-11
14:33:24.3400585,f:\Windows\System32\config\SAM.LOG2,C:\collect\f\Windows\S
ystem32\config\SAM.LOG2,32768,243E1FF48F7AEAF220A4DA8C6DDAED05CB80E828,Fals
e,2017-09-29 08:45:11.8519987,2017-09-29 08:45:11.8519987,2017-09-29
08:45:11.8519987,00:00:00.0020022
2020-01-11
14:33:24.3570738,f:\Windows\System32\config\SECURITY.LOG1,C:\collect\f\Wind
ows\System32\config\SECURITY.LOG1,69632,C51959F2742B26B20DAEE6730942B556191
A899B,False,2017-09-29 08:45:11.8363743,2017-09-29 08:45:11.8363743,2017-
09-29 08:45:11.8363743,00:00:00.0020019
```

Whereas the SkipLog CSV might look like this:

```
SourceFile,SourceFileSha1,Reason
f:\Windows\System32\config\RegBack\SAM,DA39A3EE5E6B4B0D3255BFEF95601890AFD8
0709,Deduped
f:\Windows\System32\config\RegBack\SECURITY,DA39A3EE5E6B4B0D3255BFEF9560189
0AFD80709,Deduped
f:\Windows\System32\config\RegBack\SOFTWARE,DA39A3EE5E6B4B0D3255BFEF9560189
0AFD80709,Deduped
f:\Windows\System32\config\RegBack\SYSTEM,DA39A3EE5E6B4B0D3255BFEF95601890A
FD80709,Deduped
f:\Users\Default\NTUSER.DAT.LOG2,DA39A3EE5E6B4B0D3255BFEF95601890AFD80709,D
eduped
```

Both of these files can be imported into tools like Timeline Explorer or other analytical platforms. Of course, of no files are skipped, SkipLog.csv will not be created.



KAPE ships with many targets, including those defined by the SANS Forensics Analysis poster (i.e. the red poster), to include browser history, file system information, lnk files and jump lists, Outlook, event logs, and more. You are not bound to the targets that ship with KAPE however, and as we will see soon, it is easy to create your own targets to meet any investigative need you may have.

While the manual has a full break down of everything related to targets and the properties found within target configuration files, there are several key fields we will discuss here.

In the header, the description serves to inform the user as to what kind of things a given target configuration will collect. In the example above, the target will gather artifacts related to the file system. Also in the header is the Id field, which serves as a unique identifier for this target configuration. KAPE uses the Id field to ensure a target is only ever executed once. Therefore it is important that every unique target configuration contains a unique Id.

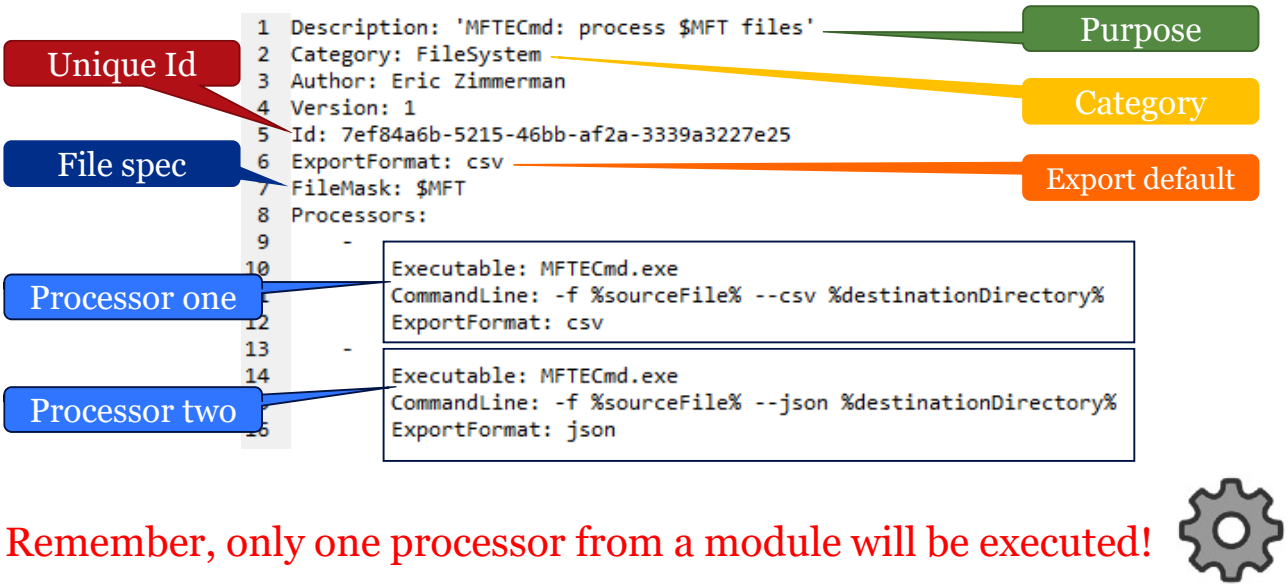
The next section in the configuration is the Targets collection. In the above example, three targets are defined. Each is very similar in its definition, with the primary differences being the Name and Path properties. In the case of $J, notice how there is an additional property, SaveAsFileName, that can be used to override what KAPE will use when copying a file from a source. In this case, **"$J"** is an alternate data stream (ADS), so the save as name is overridden to just $J to make processing later a lot simpler, in that we do not have to deal with a file containing an alternate data stream. In this example, we essentially copy the ADS to its own file vs. copying the parent file along with the ADS.

Notice as well that the first two target blocks also have the AlwaysAddToQueue option set to true. This is used to ensure KAPE will track a file in its queue of files to be copied regardless of whether Windows says the file exists or not. Because these files are special files related to the operating system, Windows does not allow access to them, even though they do in fact exist. This option ensures the file is checked from a forensics point of view vs. normal Windows file copy operations. For just about every other target configuration, this option will be set to false.

The other options in a target definition are straightforward. Each of these examples is a stand-alone file, but if Path pointed to a directory (say **`C:\windows\system32\config`**), **IsDirectory** would be set to true. When dealing with a Path that is a directory, the **IsRecursive** switch then comes into play.

KAPE includes many examples to serve as templates for you to leverage should you choose to create your own target configurations. We will also create a custom target in a lab as well to get a feel for how it works, including how to validate the syntax and test the target to ensure it behaves as you intend.

Using KAPE: Exploring Modules



Remember, only one processor from a module will be executed!

Like targets, KAPE ships with many modules and you are not bound to the modules that ship with KAPE. There are some nuances to modules to keep in mind when creating your own.

While the manual has a full break down of everything related to modules and the properties found within module configuration files, there are several key fields we will discuss as outlined above.

There is some commonality between targets and modules in that the description and Id serve essentially the same purpose as we saw with targets. Modules however have a few properties in the header that targets do not: ExportFormat and an optional FileMask.

The ExportFormat property provides the default processor that will be used for a given module. It only really comes into play when a module knows how to export data in several formats. In the above example, notice how there are export options for both CSV and JSON format. Because ExportFormat is set to 'csv', the first processor would be chosen.

FileMask is an *optional* variable that when set, tells KAPE that this module is interested in files with the name specified. This is useful to target a specific file, like a SYSTEM Registry hive, or, in this case, the $MFT. If a processor knows how to locate files it cares about based on file extension or other criteria, this becomes less important. If a processor is only expecting a certain kind of file, KAPE will call the processor for each file it finds that matches the provided FileMask.

The next section in the configuration is the Processors collection. In the above example, two processors are defined. Each is very similar in its definition, with the primary differences being the CommandLine and ExportFormat properties.

The ExportFormat property simply serves to identify what kind of file a given processor will generate. Essentially it provides valid input to the **--mef** switch, should you wish to override the default export format as indicated by the topmost ExportFormat property.

The CommandLine property deserves more attention in that there are some variables that serve as placeholders that KAPE will replace with actual values. Before covering those, notice that the executable itself is just the name of the file, without any directory information. KAPE expects the binaries to be in the **"<Kape>\Modules\bin"** directory. See the manual for more details on search order.

%sourceDirectory%: Replaced with the full path to the directory where all files can be found. This would be replaced with whatever was passed to the **--mdest** switch.

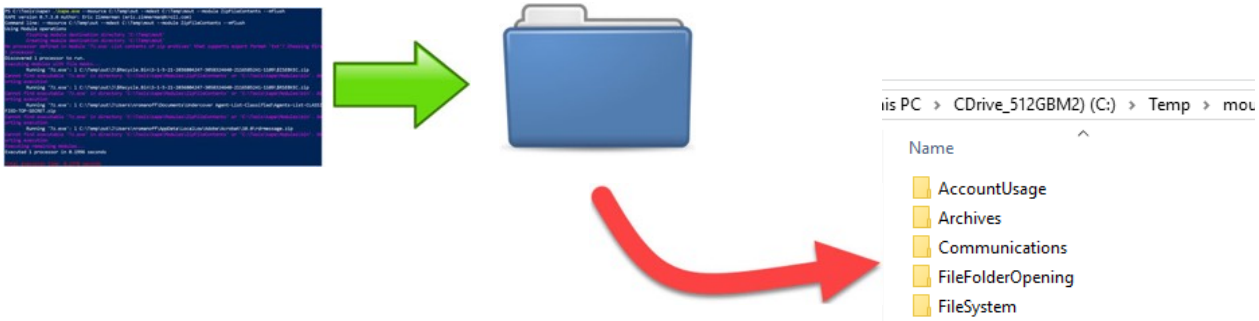
%sourceFile%: For modules that use FileMask, replaced with the full path to the file to process. This would be a full path to a file, originating at some point under the value of **--mdest**

%destinationDirectory%: The full path to the root directory where a file will be saved.

%destinationDirectory% is set to the path provided to the **--mdest** switch with the Category appended to it. For example, in the module above, if **--mdest** was set to D:\dataOut, **%destinationDirectory%** would be set to **"D:\dataOut\FileSystem"**. This allows KAPE to group output from different modules that process different files, but are related to a certain kind of artifact, in the same directory, making it easier for you to review related artifacts.

KAPE includes many examples to serve as templates for you to leverage, should you choose to create your own module configurations. We will also create a custom module in an exercise to get a feel for how it works.

Using KAPE: Analyzing Module Output (1)



- KAPE groups output by category
- Focus on the categories you need
- Pivot into other categories

When the module options are used, KAPE runs one or more module configurations against the files located in **`--msource`**. As matching files are found, the information is passed into the processors which then generate output.

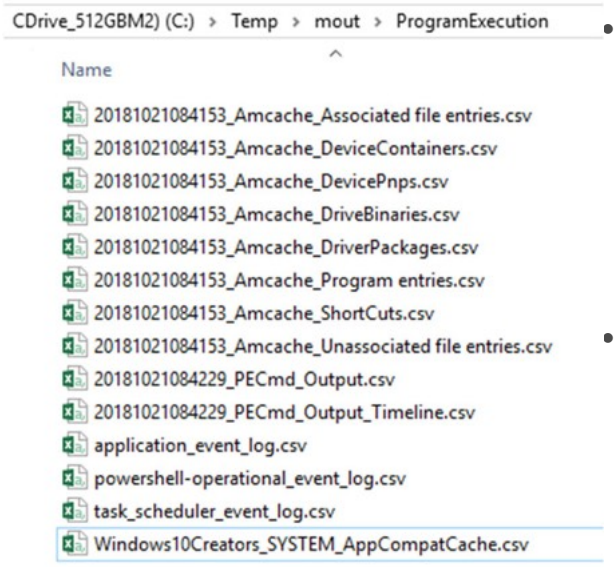
In the above image, notice how there is a single txt file (the Console log) and multiple directories. As we saw when we discussed module configurations, these directories are named after the Categories as specified in the module. The Console log file is a record of exactly what was displayed on the console during KAPE execution.

As a general best practice, focus initially on the category that answers the questions you most need answered. While that sounds simple in practice, even experienced examiners can get mired down in the vast number of artifacts available from a computer. For example, if your case involves determining what Word documents were opened, the best place to start would be the **`"FileFolderOpening"`** directory, as that would contain the output from things like lnk files, jump lists, and so on, all of which can directly help answer such a question.

Contrast this with a need to know when a certain file was created (FileSystem), or when a user account successfully logged on against a network share (AccountUsage), and you can see how organizing things into these high-level categories lets you drill down into key areas in an efficient manner.

Another major benefit from organizing things this way is the natural way it allows for pivoting into other related artifacts. For example, consider something like ShellBags that provides a rich view into directories opened by someone. One thing missing in ShellBags data however is the serial number of the device where the directory is located. Luckily, the serial number is in other artifacts such as lnk files and jump lists. By using the common denominator of the full file path, additional details can be gleaned (full path + volume serial number) which can then lead to things such as removable storage, phones, etc. Since both lnk and jumplist details would end up in the **`"FileFolderOpening"`** directory, such comparison and correlation becomes much easier!

Using KAPE: Analyzing Module Output (2)



- ProgramExecution directory contains the output from several modules
  - AmcacheParser
  - PECmd
  - AppCompatCacheParser
- CSVs can be reviewed in Timeline Explorer or Excel, loaded into SOF-ELK®, etc.

In most cases, programs used in modules know how to generate at least CSV output. As such, this will be the most common type of data you review. The amount of files in each category's directory is tied to not only how many modules ran, but the kinds of output files generated by each processor in each module.

For example, a program like AppCompatCacheParser will generate one CSV whereas a tool like AmcacheParser can generate anywhere from two to ten different output files. When multiple modules are run that are all related to ProgramExecution, you may end up with a dozen or more output files.

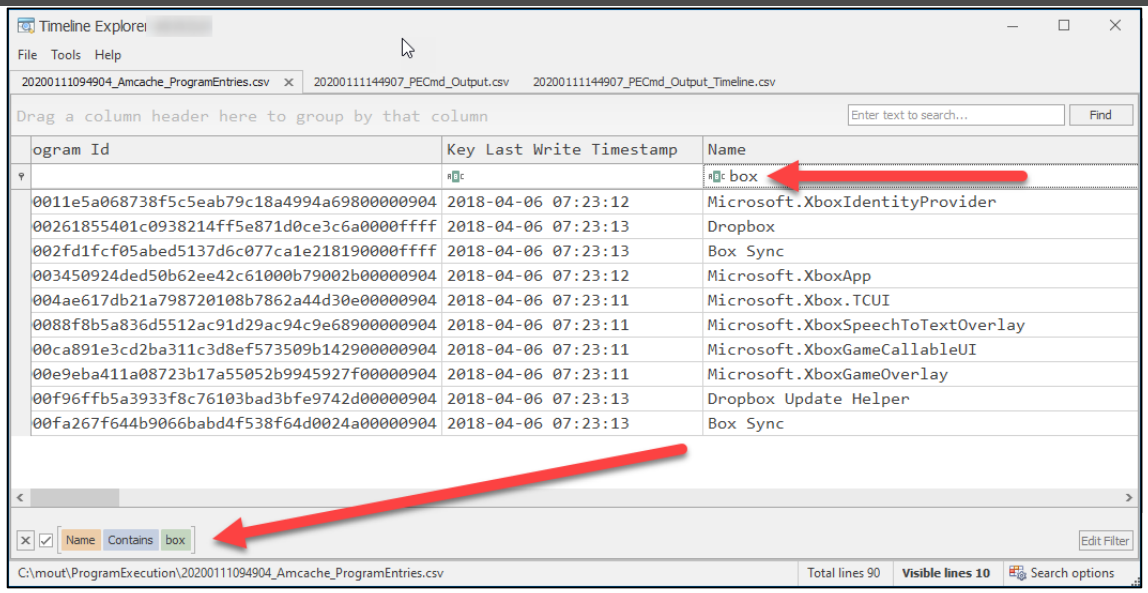
In some cases, the output files in the various directories may not all be of the same type. In cases like this, our analytical process will have to change slightly, in that we may want to use one tool for CSV output, another tool for JSON output, and yet another to look at XML.

In some cases it may make sense to ingest the output files into a tool like SOF-ELK® [1] or Splunk in order to be able to search across many different machines and file types.

Tools such as Timeline Explorer (TLE) support many formats directly, in that TLE knows which columns are present, their data types, and so on. Even in cases where the format is not directly supported, CSV and Excel files can be dropped into TLE and it will load and display them as is. For Excel files, only the first workbook is shown.

[1] Configuration files for the SOF-ELK VM, used in SANS FOR572 | `https://for498.com/colj6`

Using KAPE: Analyzing Module Output (3)



This is an example of what a few of the ProgramExecution files look like in Timeline Explorer (TLE). Once the files are loaded into TLE, the contents can be filtered, searched, grouped, and so on in order to home in on key data.

Timeline Explorer has support for the following programs built right in:

- AmcacheParser
- AnalyzeMFT
- AppCompatCacheParser
- Autorunsc
- Density Scout
- JLECmd
- LECmd
- KAPE
- LECmd
- FLS based timelines (mactime)
- PECmd
- Pescan
- SBECmd
- Shimcachemem (volatility plugin)
- ShimcacheParser
- SigCheck
- Log2Timeline based timelines (l2tcsv format)

Summary

- Familiarize yourself with the KAPE-abilities of this fantastic tool
- Learn to edit and create your own target and module files
- KAPE can extract and save data, while maintaining the underlying metadata
- Full audit trail capability allows for complete visibility of the entire process

This page intentionally left blank.

Exercise 4.2

Using KAPE for Battlefield Forensics

Synopsis: In this exercise, you will be using KAPE to perform a myriad of tasks, including familiarizing yourself with target and module files, editing and creating your own, extracting and processing data, and performing quick win Battlefield Forensics.

Average Time: 60 Minutes

This page intentionally left blank.

Exercise 4.2 Takeaway

- Storage devices are growing faster than our ability to image them efficiently.
- Battlefield forensics is the answer to this problem, as it allows for quickly collecting key data to move your case forward.
- KAPE is a powerful and flexible tool that allows for rapid collection and processing of key artifacts.
- By leveraging KAPE in your process you can focus on the important details vs. waiting hours for a full disk image to complete.
- Even in a very limited triage collection, a wide range of questions can be answered which can help drive your case forward efficiently.

This page intentionally left blank.

FOR498.4: Non-Traditional & Cloud Acquisition Agenda

4.1 File Systems Revisited

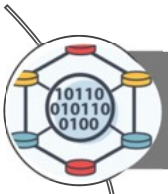
4.2 Battlefield Forensics with KAPE

4.3 Multi-Drive Storage

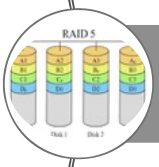
4.4 Remote Acquisition

This page intentionally left blank.

Multi-Drive Storage



Big Data Storage



RAID Scenarios

Imaging Large Storage Arrays

This page intentionally left blank.

Big Data

Data must exist somewhere, even if not immediately obvious.

The challenge is in finding the container, and more importantly, understanding how it is configured.

Multi-disk storage containers are ubiquitous.

SANS DFIR | FOR498 | Battlefield Forensics & Data Acquisition 77

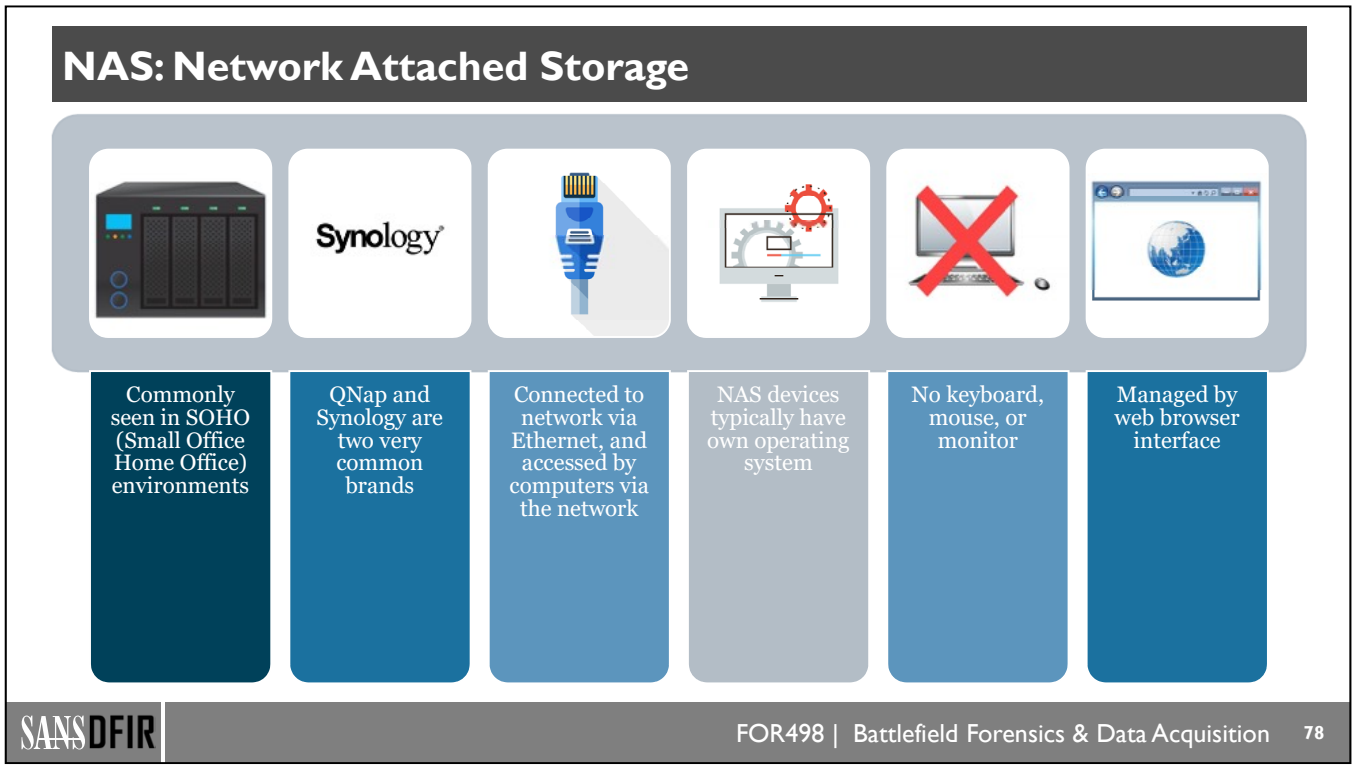
A significant challenge today is the volume of data that potentially exists within our scope of investigation. More importantly, this data must exist somewhere. That "somewhere" is on some type of storage media. Whether solid state or spinning, there must be a storage container. In the enterprise, due to storage size needs, this will mean devices that contain multiple hard drives. These hard drives are then configured in such a way as to create storage areas that are, in most cases, irrespective of the physical storage size of each individual hard drive.

A recent case in our lab (we are a very small company) involved the collection of a server room from a small oil and gas company. Almost 300 hard drives were seized, with most of them coming from servers and storage arrays. In this case, bailiffs with no technical knowledge entered the facility, turned off the power, and dismantled the entire server room down to its individual components and removed them for auction. They were subsequently turned over to us for collection and examination. Because everything had been taken apart, and none of the staff from the company were accessible any longer, we were left with the task of trying to figure out how all devices went back together. These involved normal SATA hard drives, some solid-state hard drives, many SAS drives, and most of the hard drives were Fibre Channel drives.

Consider a company like backup provider BackBlaze, [1] who have 100,000 hard drives in their facilities, and store approximately 800 PETAbytes . If you are counting, that is 800,000 Terabytes. Try to think about how you would approach an investigation in a facility this size.

Just a few short years ago, even one terabyte seemed ridiculous. Brian Hayes wrote an article on exactly this in 2002, called Terabyte Territory. [2] In 2002, Maxtor released a 120 GB HDD [3] that was the latest and greatest in storage technology and weighed in at about 750.00 USD.

[1] BackBlaze | `https://www.backblaze.com/`
[2] Terabyte Territory by Brian Hayes | `https://for498.com/ckzlb`
[3] 120 GB HDD Released | `https://for498.com/h60qi`



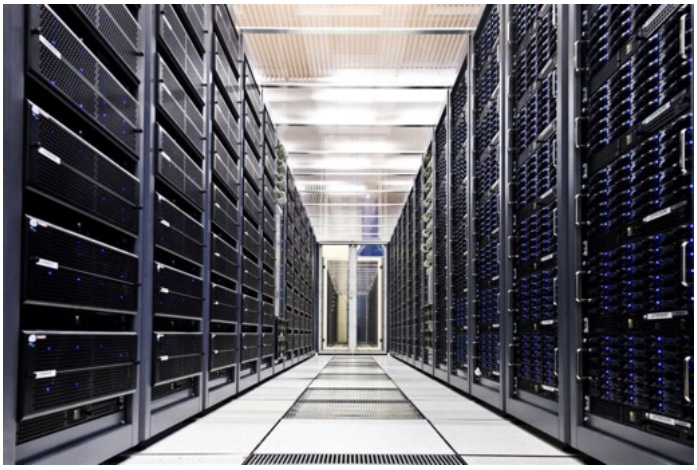
Physical storage "containers" can take on many different configurations. For home/small business use, QNap and Synology are two commonly seen brands, and they can contain multiple hard drives in various configurations. This is commonly referred to as a NAS (Network Attached Storage). This means that the device is connected to the network and is accessible by any computer configured to do so. The NAS will typically have its own embedded operating system, usually based on some flavor of Unix. The NAS does not have a keyboard, mouse, or monitor, and is managed (usually) via a web browser interface.

Depending on the make and model of the NAS, the only connection methodology will be either through dismantling the device, or through the network. Some devices have other methods of access as well, however in many cases this is usually to place data on the NAS, and not to extract data in any meaningful way.

When it comes to creating a forensic image of large storage spaces, more and more examiners are turning to the building of a NAS that is big enough to hold the amount of data they will be acquiring from something like a server. This serves a couple of purposes. The first being that the storage area from a server may exceed the size of any single available drive. For example, if the acquisition size is 36 TB, it will not fit on any single hard drive. The second purpose of a NAS in big data acquisition is that it is network addressable. As we will see later, many server environments lack interfaces faster than USB 2.0 other than the network. Therefore placing a network addressable device on the wire provides options for the examiner.

Enterprise Storage

- Can be massive
- Even simple ones can be 15 disk or larger storage enclosures, with many enclosures in an array



On the enterprise side of things, these storage containers can be massive, holding many multiples of hard drives. It is not uncommon in a smaller enterprise to see storage enclosures with 15 or 24 hard drives, and then having multiples of these enclosures. As a comparison, Google stated in 2016 that they had approximately 2.5 million servers. [1]

[1] Google Data Centers | `https://for498.com/onq9i`

RAID: Redundant Array of Independent Disks

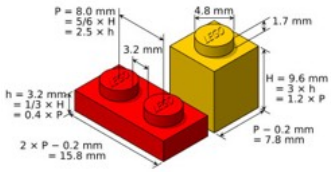
- RAID (Redundant Array of Independent Disks) allows for the use of multiple hard disks in various configurations to address different needs for different users

- Many different RAID configurations
- RAID 0, 1, 5, 6, and 10 are most commonly used

RAID (Redundant Array of Independent Disks) [1] allows for the use of multiple hard disks in various configurations to address different needs for different users. Although there are quite a number of different RAID configurations, we will only be looking at the most commonly used 5.

RAID | `https://for498.com/nxtcj`

Block Size



RAID allocates data into chunks called blocks

Common block sizes are 64 KB, 128 KB, and 256 KB

If a file spans more than one block, it will usually span more than one disk in the RAID

Block size affects disk read/write efficiency

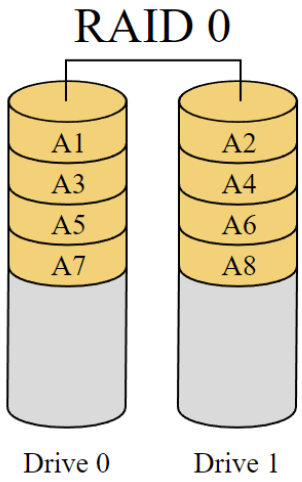
Efficiency

An important component of RAID is something called “block size”. When data is placed onto any RAID, it is done in blocks. Common block sizes are 64 KB, 128 KB, and 256 KB. Block size will affect data write efficiency, so there is no “one size fits all”.

The best way to think about block size is to understand the size of striping on RAID. In other words if a RAID has 64 KB stripe size, and it spans 4 drives with a 5th as parity, the optimal block size would probably be 256 KB. It would be fair to say that the larger the array in volume size and number of disks, the larger your block size.

RAID 0

- Commonly contains two disks but may contain more
- Total storage size of all disks is equal to user available storage
- Allows for higher speed I/O, so better for efficiency
- If one disk fails, all data is lost*



** When file sizes are larger than the block size. Any files that are smaller than block size may be carved from the remaining operational disks.*

RAID 0 is a configuration commonly consisting of 2 hard disks, and although it is widely believed that it must only be 2 disks, this is incorrect. RAID 0 can contain multiple disks. In a RAID 0 configuration, the total storage space of all of the disks is the amount of storage space available to the user.

As an example, 2 hard disks of 2 TB each will create a total storage area of 4 TB. In another example, 4 hard disks of 4 TB each can create a total storage area of 16 TB.

This configuration provides for much higher efficiency, because the array is using multiple channels for read and write operations. This is available because the data is being "striped" or "spanned" across the different disks. The downside to this is that if one disk fails, the data from all disks is unrecoverable* except via RAID recovery software or a data recovery lab.

**Suggesting file sizes that are larger than the block size of the RAID. Any files that are smaller than the block size of the RAID may be able to be carved from the remaining operational disks.*

The key usage for RAID 0 is to gain I/O (input/output) speed and economic storage amounts, but the disadvantage is the complete lack of redundancy.

PROS

Large storage size for less money

Much faster than normal storage

Easy to set up

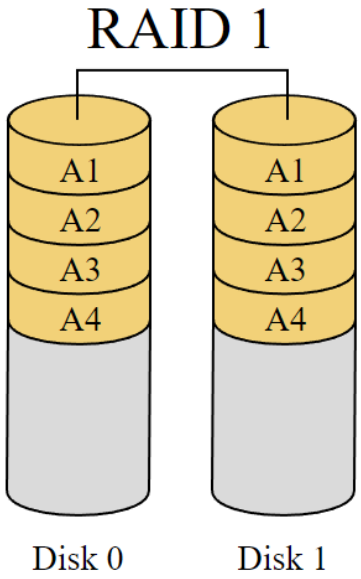
CONS

No redundancy of data

If one drive fails, all data is lost

RAID 1

- Contains two disks
  - Disks must be same size
- Total usable space is ½ of total of both disk sizes
- Allows for redundancy
- Data is written to both drives simultaneously



RAID 1 is a configuration that consists of 2 hard disks. These disks must be of equal size, or at least the reported sizes must be equal. The total usable space is equal to the size of one of the disks.

As an example, 2 hard disks of 2 TB each will create a storage area of 2 TB. In another example, 2 hard disks of 4 TB each can create a total storage area of 4 TB.

This configuration provides for complete redundancy, because data is written simultaneously to both disks in the array. This way, if either disk fails, the data is still completely recoverable from the other disk.

The key usage for RAID 1 is to gain critical data redundancy, as the data is being written (or mirrored) to two separate hard disks at once. The disadvantages are that I/O speed is reduced compared to RAID 0, and the cost for storage is exactly double the cost for the same data space without redundancy.

PROS

Data redundancy

CONS

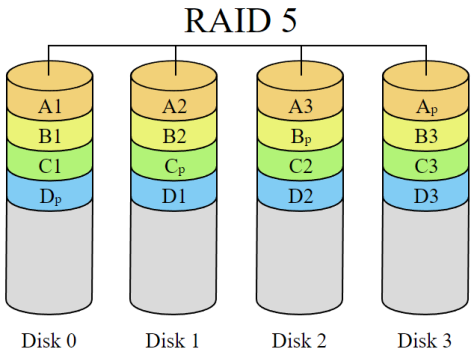
Storage price exactly double

No faster than normal disk configuration

In the case of a drive failure, user must shut down array to recover data

RAID 5

- Consists of three or more disks
  - Disks must be same size
- Total usable storage space equal to total size of all disks in array, minus size of one disk
- Provides redundancy and mission critical up time
- Can remove and replace a failed disk without turning off the array



RAID 5 is a configuration that consists of 3 or more hard disks. These disks must be the same size to avoid wasting storage space. The maximum amount of disks is unlimited (in theory) but will be limited by the hardware in use. The total usable space will be equal to the total of all disks in the array minus the size of one of the disks.

As an example, 3 hard disks of 2 TB each can create a storage area of 4 TB. In another example, 8 hard disks of 4 TB each can create a storage area of 28 TB.

This configuration provides for not only redundancy, but also for protection against disk failure. More importantly than that, RAID 5 will continue to operate seamlessly, even during a drive failure. The data is striped across all drives, but is also duplicated, creating something called parity. This parity will allow for the rebuilding of a data drive that fails, without any data loss.

The key usage for RAID 5 is to gain redundancy of data and allow for the loss of a single drive in the array, without data loss or loss of functionality. The disadvantage to RAID 5 is that data sizes can get too large for proper rebuilding of data in a disk loss situation.

PROS

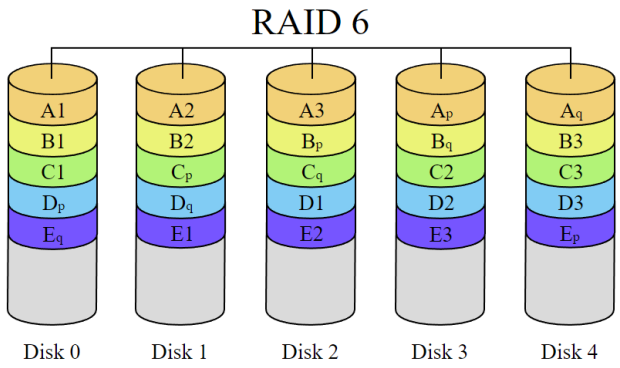
Data redundancy
Hot swappable disks for zero down time
One disk can fail without any data loss
Very large storage sizes

CONS

Harder to configure
Higher storage costs
Very large storage sizes

RAID 6

- Consists of four or more disks
  - Disks must be same size
- Total usable storage space equal to total size of all disks in array, minus size of two disks
- Provides redundancy and mission critical up time
- Can remove and replace two concurrently failed disks without turning off the array



RAID 6 is a configuration that consists of 4 or more hard disks. These disks must be the same size, to avoid wasting storage space. The maximum amount of disks is unlimited (in theory), but will be limited by the hardware in use. The total usable space will be equal to the total of all disks in the array minus the size of two of the disks. RAID 6 conceptually is identical to RAID 5, except there are 2 parity disks instead of one.

As an example, 4 hard disks of 2 TB each can create a storage area of 4 TB. In another example, 8 hard disks of 4 TB each can create a storage area of 24 TB.

This configuration provides for not only redundancy, but also for protection against disk failure. More importantly than that, RAID 6 will continue to operate seamlessly, even during up to two drive failures. The data is striped across all drives, but is also duplicated, creating something called parity. RAID 6 uses 2 different types of parity, and thus will allow for the failure of any 2 drives in the array, without any data loss.

The key usage for RAID 6 is to gain redundancy of data and allow for the loss of two disks in the array, without data loss or functionality. It provides more protection than RAID 5. The disadvantage to RAID 6 is that data sizes can get too large for proper rebuilding of data in a disk loss situation.

PROS

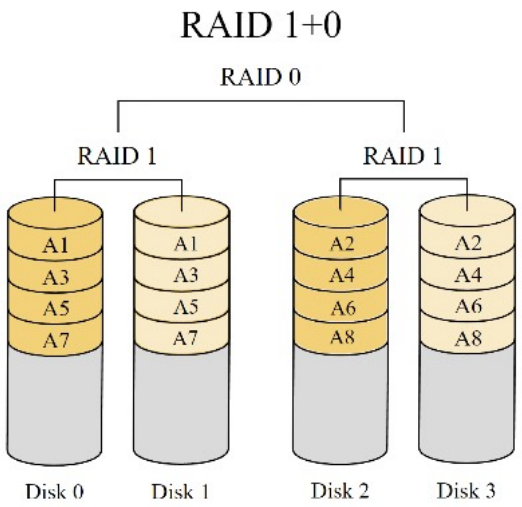
Data redundancy
Hot swappable disks for zero down time
Two disks can fail without any data loss
Very large storage sizes

CONS

Harder to configure
Higher storage costs
Very large storage sizes

RAID 10

- Consists of four or more disks
  - Disks must be same size
- Total usable storage space equal to ½ total size of all disks in array
- Provides redundancy of RAID 1 with I/O speed of RAID 0



RAID 10 (1+0), also known as nested RAID, is a configuration that consists of 4 or more hard disks. These disks should be the same size. The total usable space will be equal to the total of all disks in the array divided by 2. RAID 10 is essentially 2 RAID 1 configurations that are managed together as a RAID 0 configuration. This is NOT the same as a RAID 01 (2 X RAID 0 managed in RAID 1), but is often confused.

As an example, 4 hard disks of 2 TB each can create a storage area of 4 TB. In another example, 8 hard disks of 4 TB each can create a storage area of 16 TB.

The key usage for RAID 10 is to provide for the I/O speed of RAID 0 with the redundancy of RAID 1, and allow for much larger data sets.

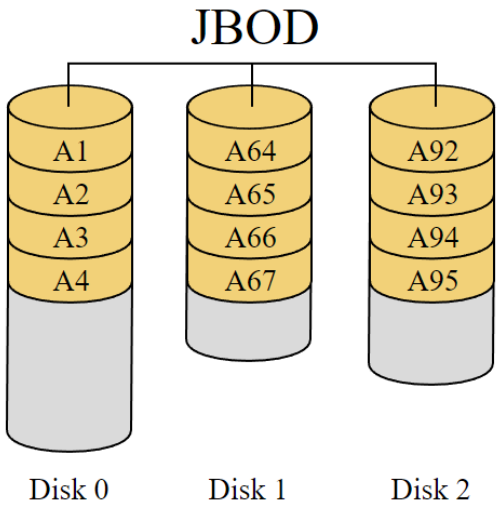
Details on other RAID configurations can be seen at `https://for498.com/170ik`

PROS
Data redundancy
Fast I/O speed
Allows for much larger data sets
Allows for fast rebuilding of data

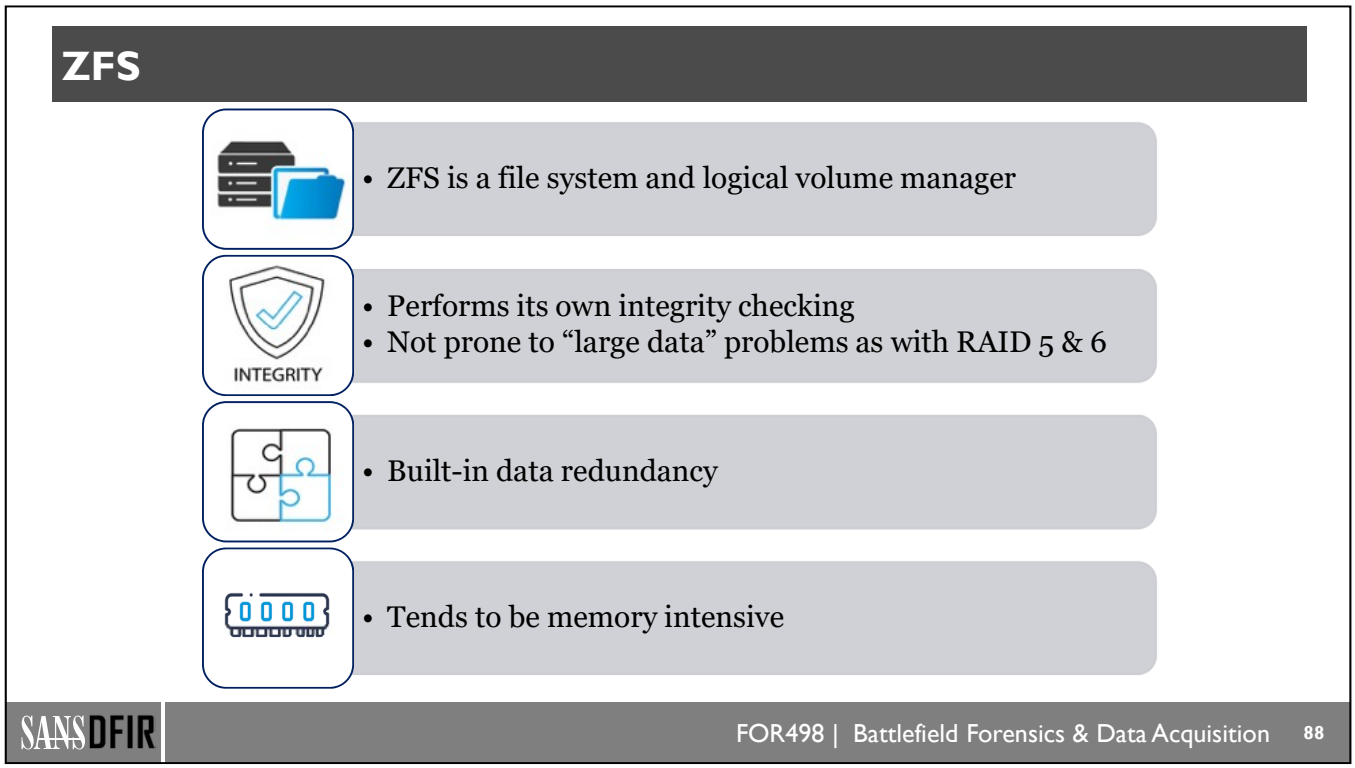
CONS
Harder to configure
Higher storage costs

JBOD: Just a Bunch of Disks

- Can be any number and size of disks
- Normal I/O speed
- Usually used with non-standard file systems
- Disks can be added as needed
- Data not striped, so data loss not as critical
- No redundancy
- Not always hot-swappable



JBOD (Just a Bunch Of Disks) is a configuration that is as the name suggests. An array will have any number of hard disks of any size, and a file system will utilize these drives as storage space. If using JBOD and the NTFS file system, there is little in the way of redundancy and/or data backup, as there would be with a RAID 1. However, JBOD configurations will more typically be used with file systems such as ZFS.

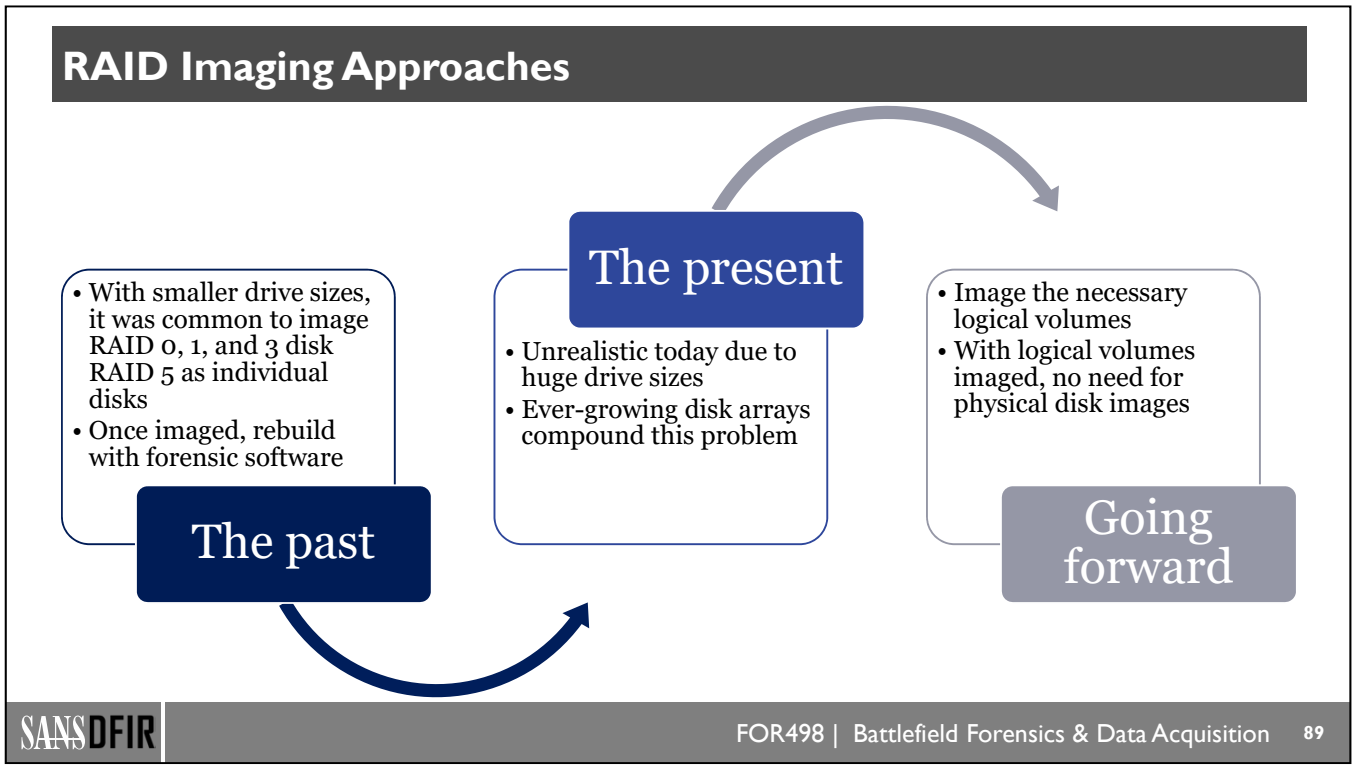


ZFS [1] is an operating system, as well as a logical volume manager that allows for the pooling of data across large arrays. Unlike RAID, it performs its own integrity checking (using checksums) and data redundancy to ensure that data is protected. Because of its unique checksum storage capabilities, it is not subjected to the problems that are now being presented in RAID 5 and 6 configurations.

Although ZFS uses a "backronym" of Zettabyte File System, it was never intended to be an acronym for anything. It is simply a File System named Z, whereas the term Zettabyte has to do with data size.

As a point of reference, 1 Zettabyte is 1000 Exabytes; 1 Exabyte is 1000 Petabytes; and 1 Petabyte is 1000 Terabytes.

[1] Z File System (ZFS) | `https://for498.com/cm-ge`



In earlier times of RAID, when 2 disk RAID 0, RAID 1, and 3 disk RAID 5 were starting to be used, it was not uncommon to shut off the server and image each disk using a physical acquisition methodology. Most forensic suites of tools had the capability of rebuilding the RAID from the created images.

Today with multi disk arrays, not to mention ever growing disk sizes, this is becoming less of a viable solution. As a result, the first responder should be approaching the problem more efficiently. This is done through live acquisition of the necessary disk volumes.

In many cases, the disk volumes are drive letters, and as such, can be imaged in their entirety. If the RAID 5 is 3 disks of 2 TB each, with a single volume, then the first responder should see a 4 TB volume. During setup of the acquisition process, it is this 4 TB volume that will be collected. As a result, you will also have collected all addressable unallocated file space. More importantly, you will not have to go through the steps of trying to rebuild an array by "knitting together" the individual disk images to achieve the same result.

This will also allow the first responder to choose only the volumes (or specific data) that they need for the investigation, rather than collecting potentially many TB of unnecessary data.

RAID (Possible) Gotchas

RAID controllers contain own BIOS

- Accessed in the similar manner as a normal computer BIOS
- Key combo usually shown on screen during boot

Tread carefully

- Very fragile as pertains to "well meaning" repair attempts
- When in doubt, ask a professional!

Beware of rebuild

- Never use the "rebuild" function to attempt to recover data
- Can often lead to irreversible damage to data

Although RAID addresses many issues such as redundancy, recovery, and storage size, it is not without its potential problems, and we will look at a few issues that can affect not only the administrator, but also the analyst and the first responder. Once we understand some of the issues, we can better understand how to identify problems that are occurring in the process, and potential troubleshooting to obtain a successful outcome.

It must be understood that when physical disks in a RAID set fail, it will usually take the advanced skills of a data recovery lab to recover the data. ANY "best attempts" by well-meaning people will, in most cases, render the data completely unrecoverable by anyone including a data recovery lab, so it is best to leave this to the professionals. Losing access to data on a RAID system is bad enough. Being responsible for never being able to recover it again is just making matters worse.

The first (and most important) rule in attempting to recover data from any failed RAID array is to NOT rebuild the array, or attempt to have the RAID software "fix" things. There will never be a good outcome from these attempts.

RAID Issues and Remedies by Level

RAID level *	# of disk failures	Result	Possible remedy
0	1	All data inaccessible	Professional recovery lab, all disks required
1	1	Data inaccessible by normal means, but intact	Some enclosures rebuild second disk automatically when replaced
5	1 | 2	Data accessible | data inaccessible	Professional recovery lab
6	2 | 3	Data accessible | data inaccessible	Professional recovery lab

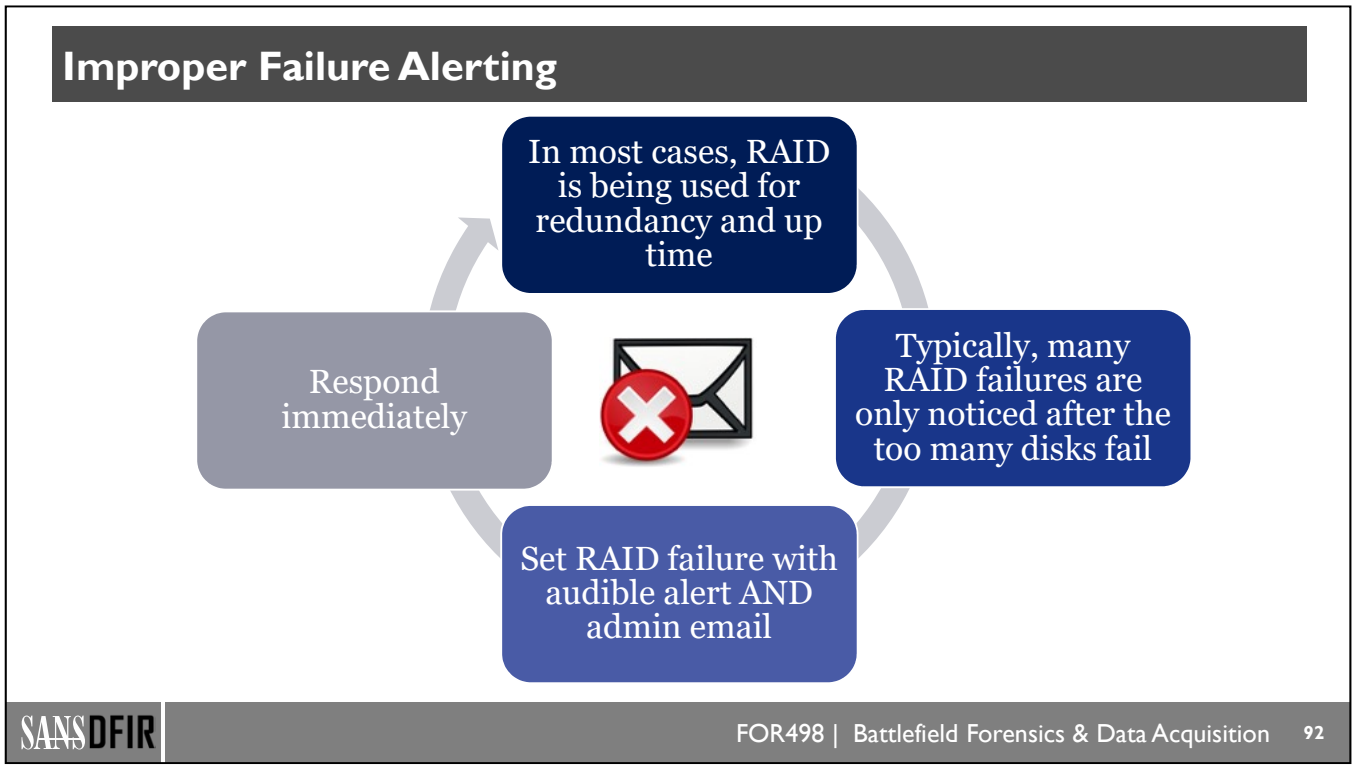
* All RAID levels assume minimum number of disks present

If one disk in a RAID 0 fails, it will render (reasonably) all data inaccessible. The only recovery option available would be a professional data recovery lab, and all disks will be necessary for recovery.

In a RAID 1 configuration, if one disk fails, the other disk should be accessible, and a data recovery lab should be unnecessary, however the RAID will cease to function, and data will be inaccessible by normal means. The remaining good disk merely needs to be slaved to another computer to access the data. In some cases, RAID 1 enclosures will rebuild a second disk automatically, once the bad disk is replaced.

In a RAID 5 configuration, if any one hard disk fails, the data should be recoverable from the rest of the disks in the array, with no advanced skills needed. In fact, the purpose of a RAID 5 environment is to allow a user to remove the failed disk and replace it with a new disk without ever having to shut the array off. Then the rest of the array repopulates the new disk with data. Note that without some warning system, an end user will not know that a disk has failed.

In a RAID 6 configuration, if any two hard disks fail, the data should be recoverable from the rest of the disks in the array, with no advanced skills needed. In fact, the purpose of a RAID 6 environments is to allow a user to remove the failed disks and replace them with new disks without ever having to shut the array off. Then the rest of the array repopulates the new disks with data. Note that without some warning system, an end user will not know that a disk has failed.



The most common issue with RAID failure is improper or inadequate alerting and/or response. The idea behind RAID (other than RAID 0) is to allow for the replacement of a failed disk without losing any data. But how do you know when a disk has failed? In the case of RAID 5, it is all too common to find out a disk has failed only when a second disk has failed, and the data is now inaccessible!

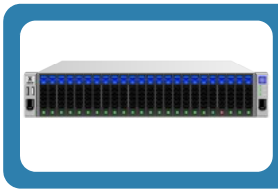
Any RAID configuration will allow the user to set certain warnings. For example, when one disk fails, an audible alarm sounds. This only works if there are actually people around to hear it! A second action could/should be to email someone a message at the moment of disk failure. They can then respond appropriately. Never respond by simply stopping the alarm and putting the replacement on a "to do" list!

Disks will typically fail because of issues that were caused at the time of disk creation in the factory, or because of operational conditions. All the drives in the array were probably bought at the same time and are from the same factory batch, or they are all being used in the same operational conditions. It stands to reason that if one disk fails, there is almost certainly more disk failure in the immediate future. The takeaway here is to never wait to replace a disk.

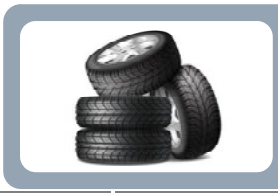
NOTE: RAID is NOT a replacement for a proper backup regimen!

Planning for Failure

Follow the "one is none, and two is one" principle



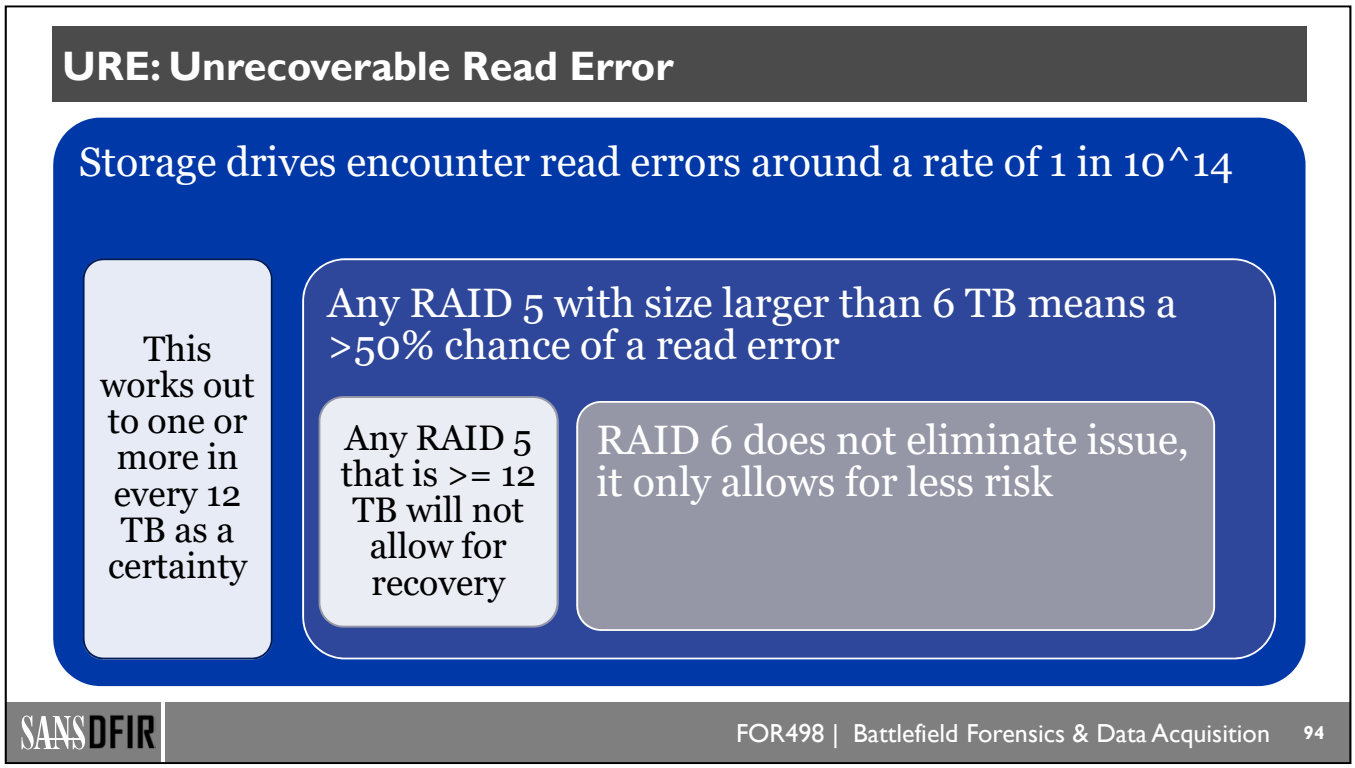
Buy enclosures with more bays than necessary



Populate two bays with "hot spares" for automatic fail over

- Alternative is to have extra disks identical to disks in the array
- Keep these in packaging in the room with the array

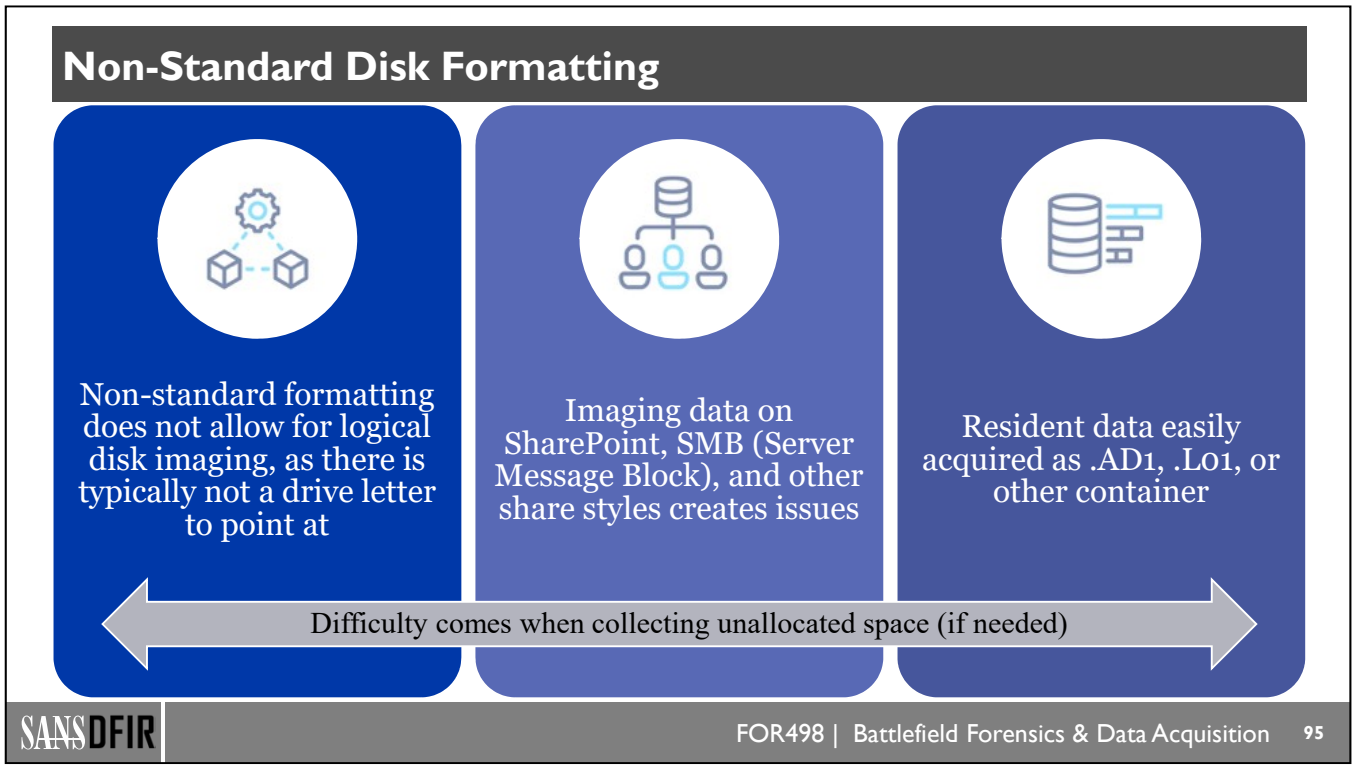
It is recommended that when planning for RAID, any enclosure will have more disk slots than needed for all the disks that are planned. Then extra disks are inserted as "hot spares". This way if a disk fails, the RAID automatically starts to repopulate the hot spares immediately. At the very least, if there are no spare slots, always purchase an extra two or three disks and keep them nearby. Then you can "hot swap" immediately, without waiting to buy new disks of the correct brand and size. This literally means having extra hard drives sitting on top of the array, just in case.



As storage sizes in RAID sets continue to get larger due to disk size, a new issue is raised. The issue of URE (Unrecoverable Read Error). It is commonly accepted that the failure rate on today's hard disk areas is somewhere in the area of 10^14. In other words, it is a certainty that in the normal course of operation, an array will encounter a URE approximately every 12 TB. With arrays of any size larger than 6 TB, the threat of a RAID 5 encountering this URE during the rebuild of a failed disk is better than 50%, increasing more with larger sizes.

Of course with RAID 6, we attempt to mitigate this, but this is only prolonging the inevitable. Therefore RAID 5 and 6 are falling out of favor in larger arrays. It is also important to understand that as arrays get larger, the time it takes for the array to rebuild a failed drive becomes unacceptably long, which also creates its own potential for data loss.

Debate rages on between two camps over URE. One camp agrees with the statistics put forth on this page. The other camp suggests that this is an incorrect calculation, because it is suggesting that the one bad sector causing URE will happen every time. The argument is usually put forth thus. If you are playing Russian roulette with a revolver, you have a one in six chance of getting the chamber holding the bullet. However each new spin resets that chance. While this is true when you are only using one bullet, when it comes to your data, that math does not work. If a single drive fails, parity must rebuild that drive and reintegrate it back into the array. This does not happen in a vacuum. Suggesting we have a six-disc array and only one drive dies and we only must rebuild the data on the one drive, the Russian roulette argument is true. However in an array, we must re-calculate the data across the array. The risk is simply too high, and a new solution must be found.



A commonly seen concern now is where companies are using non-standard formatting for storage arrays. These non-standard formats include the previously mentioned ZFS. Another commonly seen one is NFS (Network File System). [1] A third is MPFS (Multi Path File System).

These file systems do not use RAID, but rather they use the file system to much more efficiently deal with data, thereby mostly eliminating the URE issues being created with large disk RAID sets.

An additional level of complexity for the first responder is the collection of things like SharePoint, SMB (Server Message Block), and other "share" styles that do not contain drive letters. In these cases, data is usually collected in a logical fashion, such as using the .AD1,.L01, or other container format. This method will certainly allow the first responder to gather any resident data, however there is no access to any unallocated space, in the case of deleted data.

If the collection of unallocated space is necessary from these shares, the first responder must know the specific size of the user's share. Forensic collection will then start from the byte offset point of the start of the share and collect to the byte offset of the end of the share, based on the known share size. This technique is beyond the scope of this course.

[1] Network File System (NFS) | `https://for498.com/fl-je`

[2] Multi-Path File System (MPFS) | `https://for498.com/2cqb-`

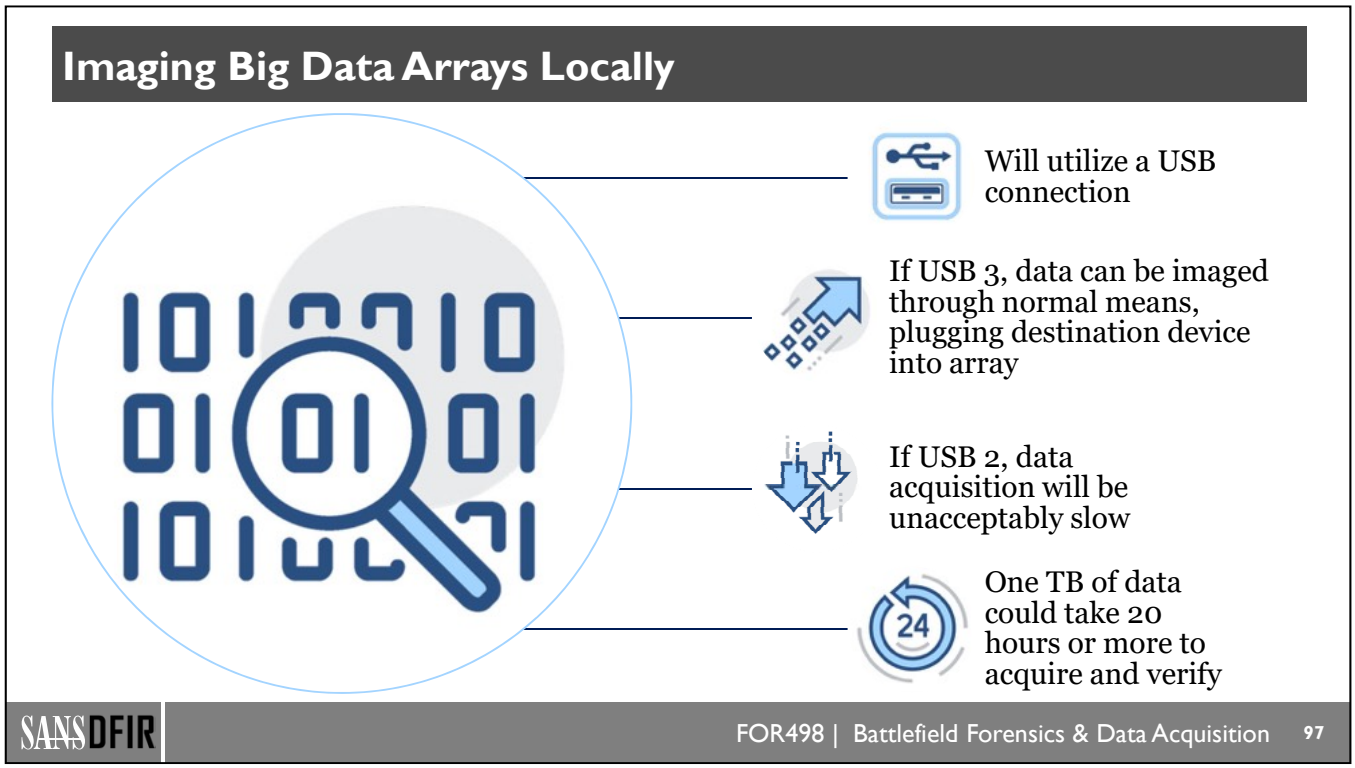
Imaging Big Data Arrays

Two methods of accessing data

Via Ethernet

Directly from controller or array

Accessing RAID volumes or non-standard formatted arrays is not as simple as collecting an image on a local machine. There are typically 2 different approaches. One approach is from across the network via Ethernet, and the second is directly from the server itself. Each is not without its potential challenges.



The difficulty for acquisition is not only in how to access the data, but with the mechanism that will be used. Many devices only utilize network access, so data will have to be extracted via ethernet. Many other rack mounted solutions will have USB access, but in many cases, this USB access is limited to USB 2.0, due to age and configuration of devices. In this case, an acquisition of even a small amount of data, say 1 TB, will take 8 hours in reasonably optimal conditions just to move. Add the overhead of acquisition hashing, potential compression, and verification, and that 1 TB of data can take well over 20 hours to acquire and verify. Clearly a different way has to be found.

If acquiring at the server itself, most devices will have USB connections. Unfortunately, many servers use USB 2.0 rather than the much faster 3.0 or 3.1. If faced with this dilemma, it may be prudent to consider network, or "across the wire" acquisition.

If acquiring "across the wire", or via Ethernet, there are a number of considerations. How far geographically are you from the data you are acquiring? What is the speed of the network you are acquiring across? How much data are you acquiring? How many people are "on the wire" at the time? If you are physically near the server, it can be faster to acquire through a Gigabit Ethernet than using a USB 2.0 connection.

Imaging Big Data Arrays Remotely

- Limit concurrent bandwidth activity
- F-Response
- Use 1 Gigabit or faster network
- Ensure you are as geographically close to the data as possible

This page intentionally left blank.

Summary

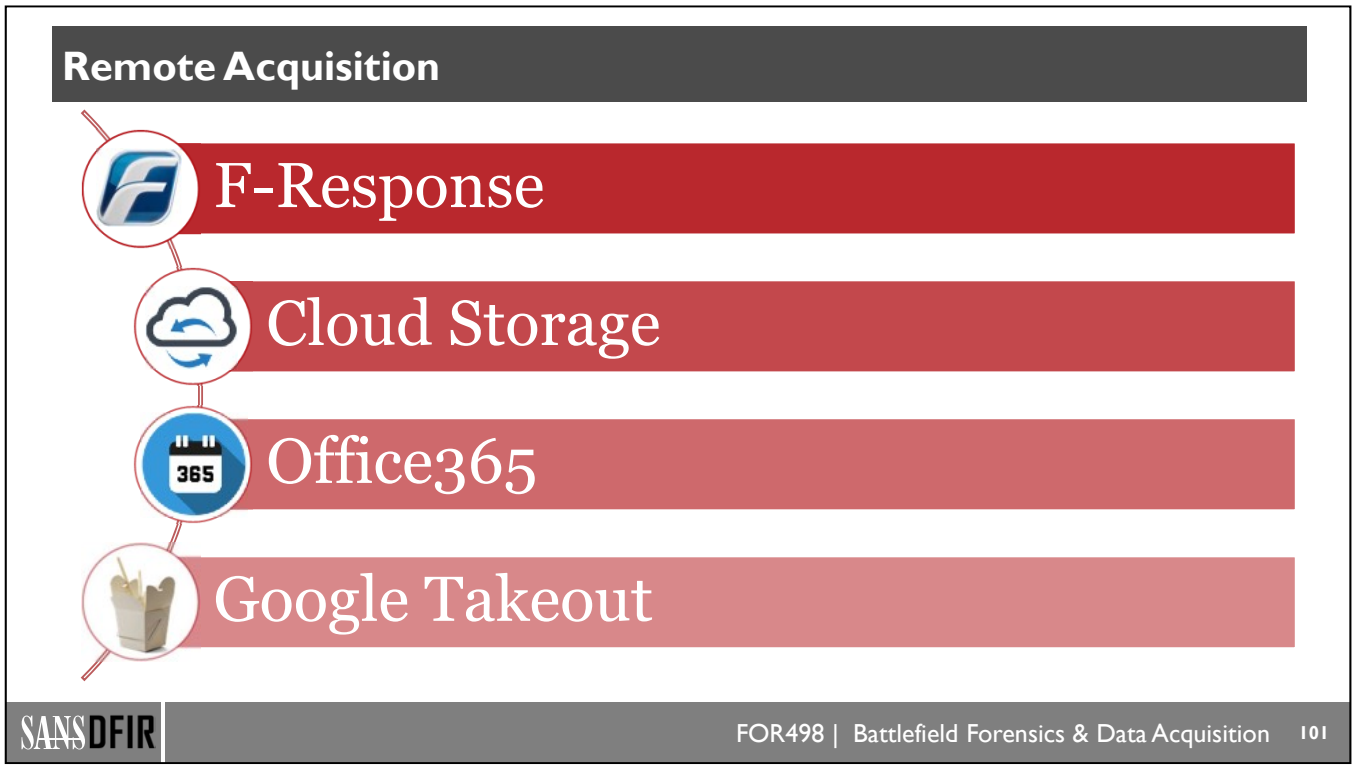
- RAID allows for large data storage relatively economically
- RAID allows for redundancy and zero down time
- RAID is NOT a replacement for an effective backup regimen
- There are alternatives to RAID that more effectively address any issues with URE
- Big data can present many acquisition problems simply based on file system type and physical access limitations

This page intentionally left blank.

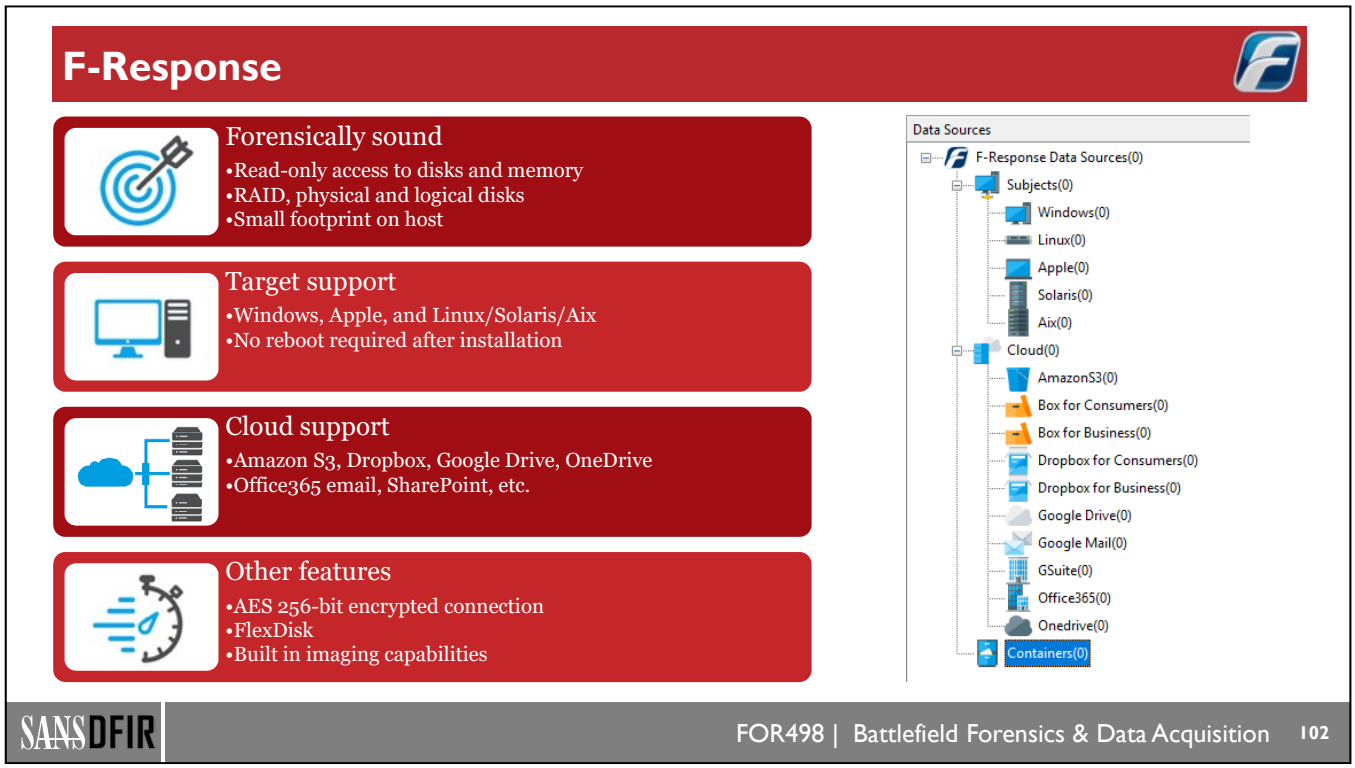
FOR498.4: Non-Traditional & Cloud Acquisition Agenda

- 4.1 File Systems Revisited
- 4.2 Battlefield Forensics with KAPE
- 4.3 Multi-Drive Storage
- 4.4 Remote Acquisition

This page intentionally left blank.



This page intentionally left blank.



"F-Response is a forensic, e-discovery, and incident response connection and collection application. F-Response was designed to provide direct, read-only access, to remote physical machines (disks, raid, volumes, and memory) as well as remote cloud storage providers. In addition, F-Response provides a clean and simple optional imaging capability for collecting F-Response presented data from multiple sources." [1]

F-Response was designed from the ground up to for simplicity of operation. The F-Response Enterprise Management Console (FEMC for short) enables investigators to perform any size network deployment to a virtually limitless number of remote target machines. F-Response allows you the examiner to obtain completely vendor neutral, write protected access to remote physical disks, logical volumes, and in some cases physical memory from over ten different remote operating system environments.

The Enterprise edition of F-Response also includes access to the F-Response Accelerator, a secondary connectivity tool allowing for an unlimited number of remote examiners as well as optional HIPAA (Health Insurance Portability and Accountability Act) compliant and industry standard AES (Advanced Encryption Standard) 256-bit encryption for connections to almost all supported target platforms. F-Response Consultant+Covert was designed to be covert and efficient allowing the investigator to access multiple machines quickly without concern for alerting the end user. For teams needing unlimited connectivity, F-Response Enterprise does the same, but is not licensed on a per seat basis, and one license of F-Response Enterprise provides unlimited client installations, unlimited target connections, and unlimited examiner connections.

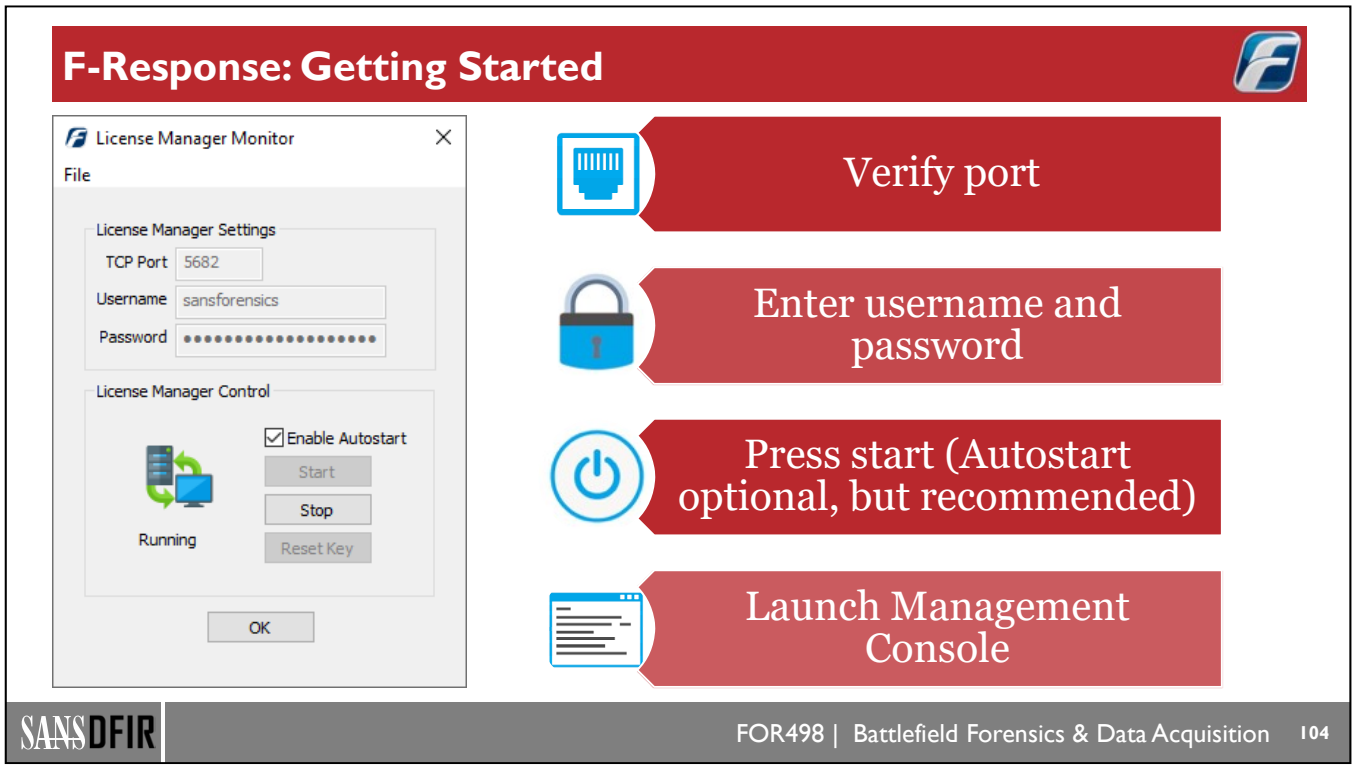
F-Response provides direct, live, read-only access to the remote target computer's disks, volumes, and in certain cases physical memory. Because all access is at the physical level, there is no file level locking. F-Response gives you access to any and all content on the remote target, including protected system content (Registry files, e-mail PSTs, database files, etc.).

F-Response supports the largest array of remote target platforms including Windows 2000, XP, 2003, Vista, 2008, 7, 8, 10, 2012, and 2016. Physical memory is supported on both 32-bit and 64-bit versions of Windows.

Finally, F-Response is vendor neutral and works with just about any tool! In many ways, it even makes existing tools that much better, because it essentially adds significant remote forensics capabilities where they may not have previously existed.

Note: If you register your provided dongle, you get additional months on your license. Look in your kit for additional details.

[1] F-Response Consultant | `https://for498.com/pg3uc`

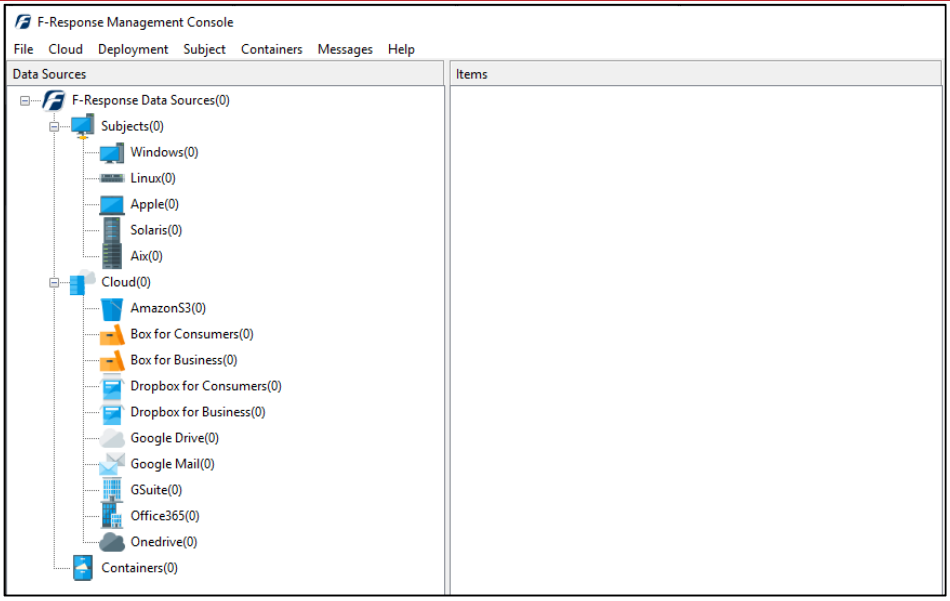


F-Response comes in two parts. The first is the management side which is where you will spend most of your time. The management console is installed via a typical MSI installer and the latest version can be downloaded from the F-Response website. Once the software is installed, several pieces need to be configured for it to work.

The first thing to do is start the License Manager Monitor. This program allows you to configure the license manager, including the default port and the username and password to be used when connecting to remote end points. Depending on the environment you are operating in, you may need to take steps to open the TCP port specified in order to allow clients to communicate with the license manager. On a local area network this is less of a concern, but when in doubt, verify the ports with the network administrators should any communication issues arise.

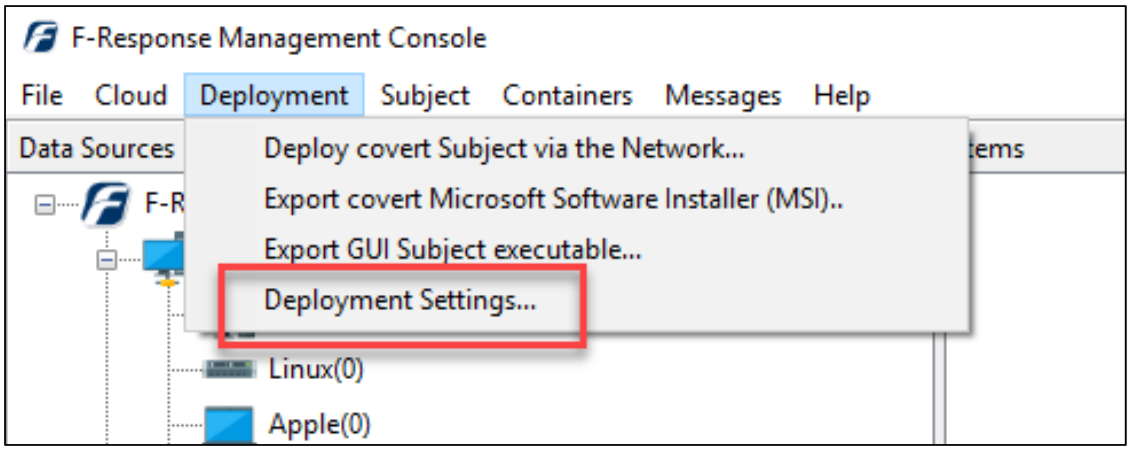
Next, a username and password must be entered. This is the username and password that will be used to configure the host-based client so it can access the license manager. The license manager determines whether the instance with the username and password is allowed to run. The same credentials are used to authenticate against a host when connecting to it.

F-Response: The Console and Configuration (1)



After starting the console, the first thing to do is finish the configuration of the F-Response agent.

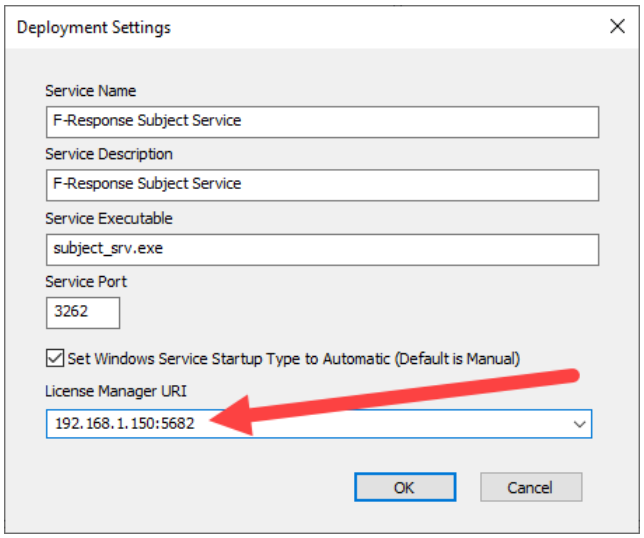
F-Response: The Console and Configuration (2)



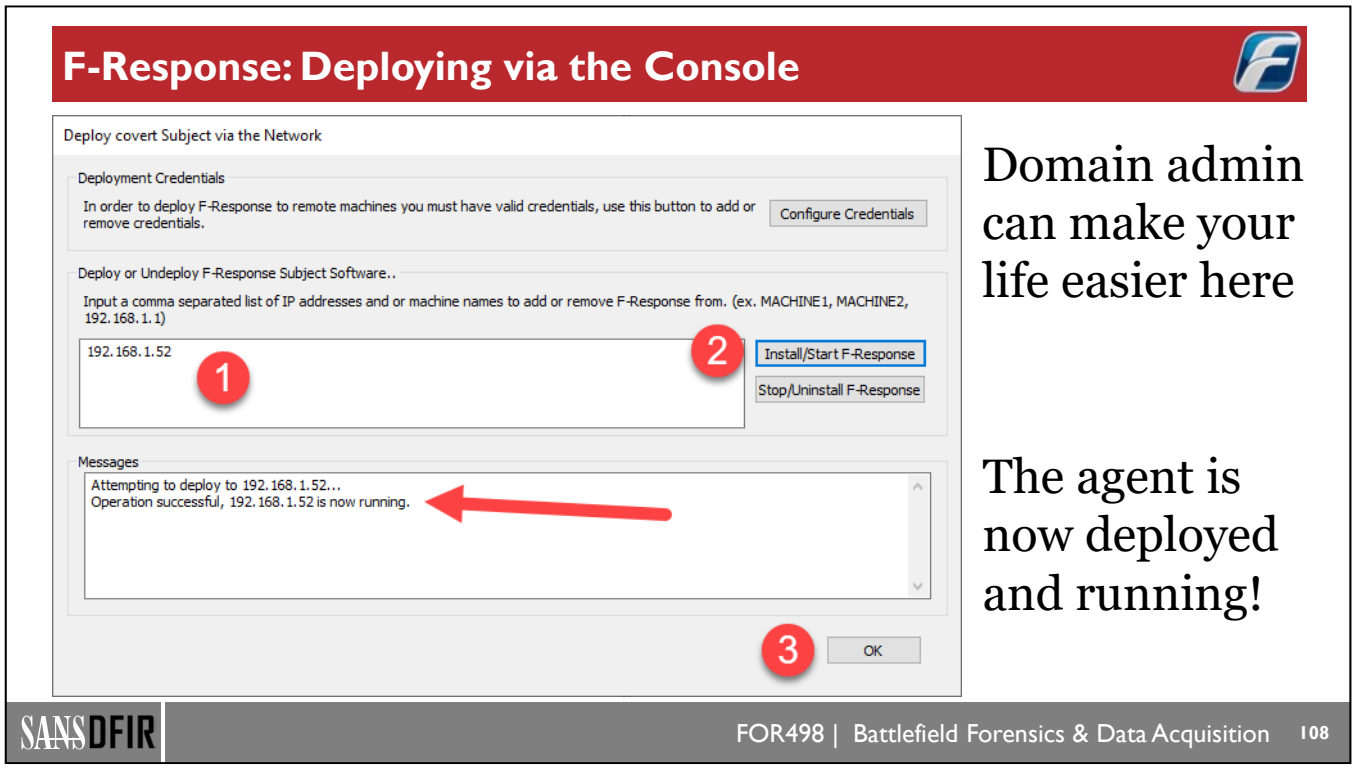
Under the **Deployment** menu, click on **Deployment Settings**.

F-Response: The Console and Configuration (3)

Select the IP and port for the licensing manager from the dropdown



The only required thing to do on this screen is select the **License Manager URI** from the dropdown. Depending on the type of investigation you are conducting, you may want to change the Service name, description, and executable name to better blend in with the system F-Response is being deployed to. This allows F-Response to hide in plain sight more so than if the default name and description are kept. Once the **Deployment Settings** are updated, click **OK** to save them.

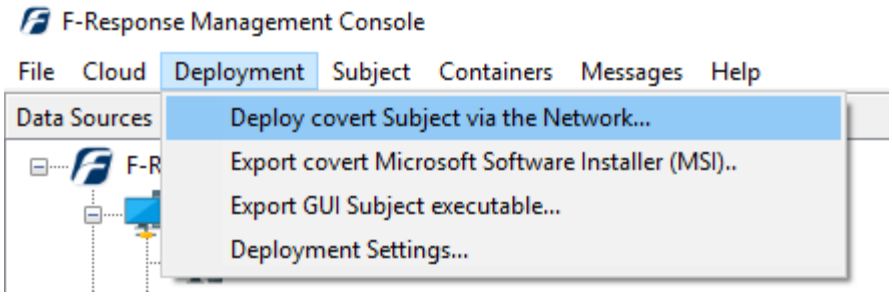


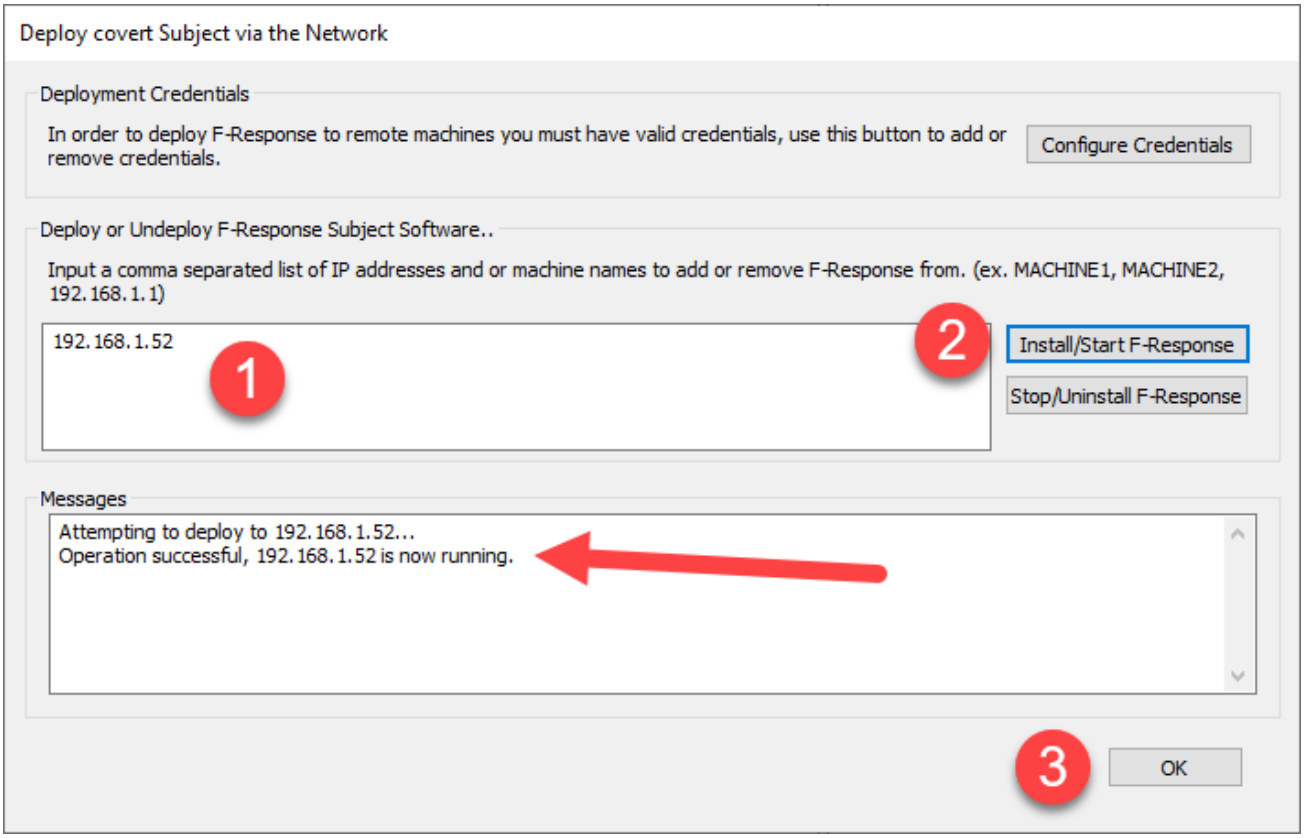
With the client configuration done, click the **Deployment** menu, then select **Deploy covert Subject via the Network**. Since we are using the Consultant+Covert option, we will only be able to covertly deploy one agent at a time, but the Consultant edition lets you manage multiple clients once the agent is installed via other means, like group policy or old school sneakernet deployment on USB drives or network shares.

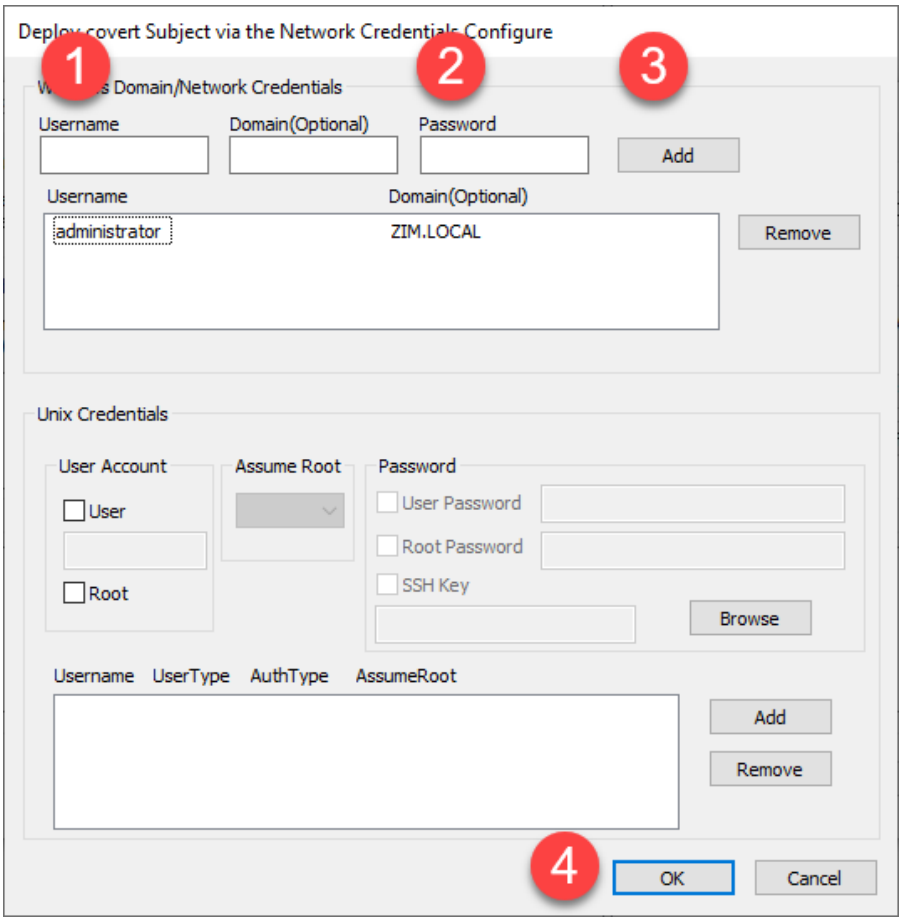
Once in the **Deploy covert Subject via the Network** menu, the first thing to do is **Configure Credentials** via the button on the top right. In the credentials dialog, enter the credentials for the target machines and click **Add** to save. If you are on a Windows domain, entering an account with domain administrator privileges can make life a lot easier here. Once all the credentials are added, click OK to save and return to the previous screen.

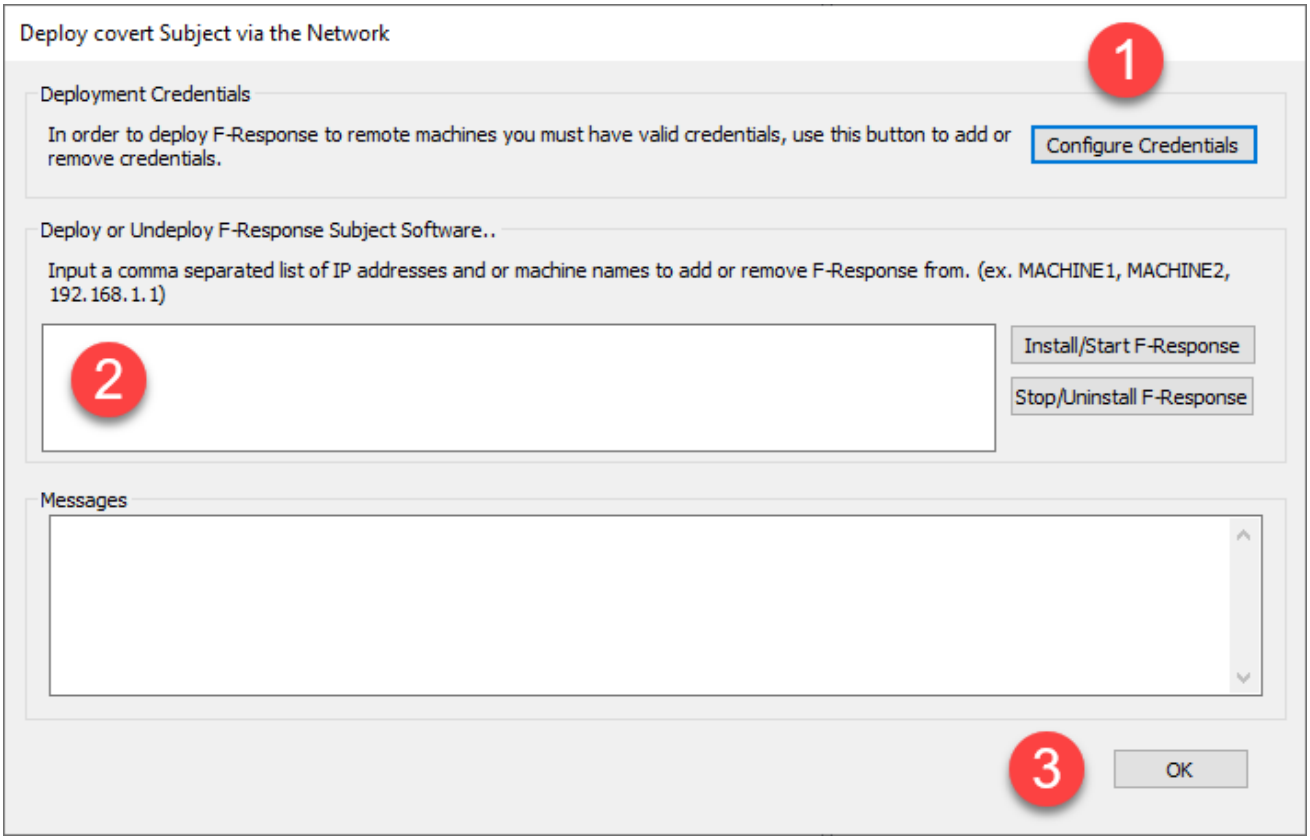
In the middle box, enter the hostname or IP address of the computer you want to push F-Response to, then click the **Install/Start F-Response** button. This will initiate the installation and starting of F-Response on the remote machine. A message will be displayed when the operation is complete. Click **OK** to continue at this point as you are now ready to interact with the deployed agent.

NOTE: Ensure you have allowed the licensing manager's port through any client firewalls on the machine where the License Manager service is running! Without the port open, the deployment will fail. Should this happen, the **Messages** box will reflect the fact that the Licensing Manager could not be reached.

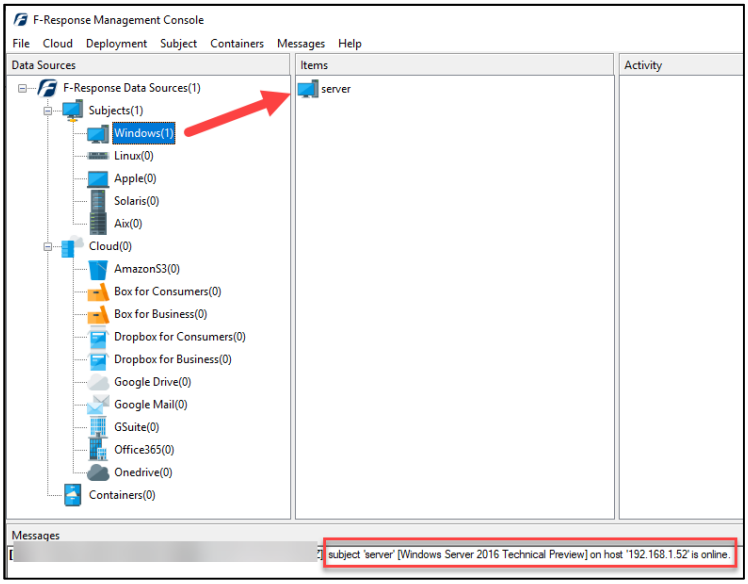






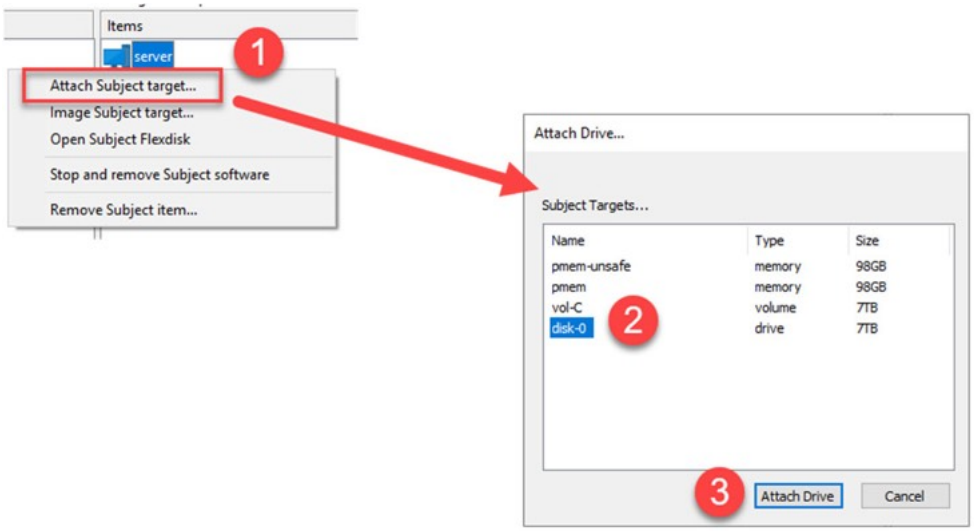


F-Response: Accessing Client Resources (1)



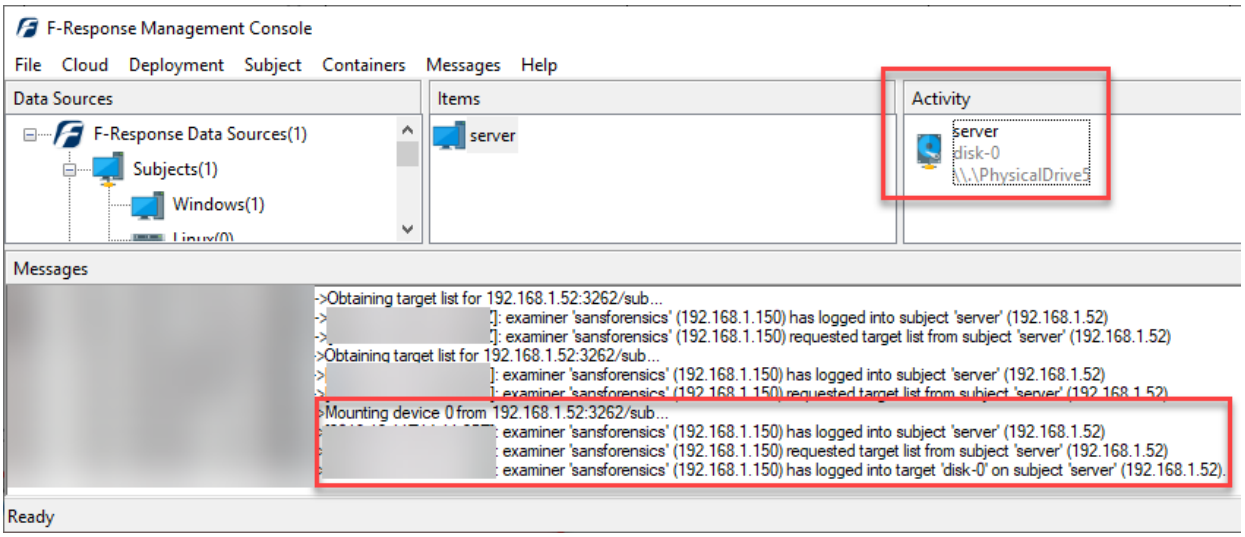
With the agent deployed and started, the remote machine will "check in" with the console and report its availability. Connected devices show up under the Items column and can be interacted with via double clicking, or the context menu via right clicking a device. The context menu contains several options including attaching a remote disk (**Attach Subject target**) as well as imaging a device directly from the Management Console.

F-Response: Accessing Client Resources (2)



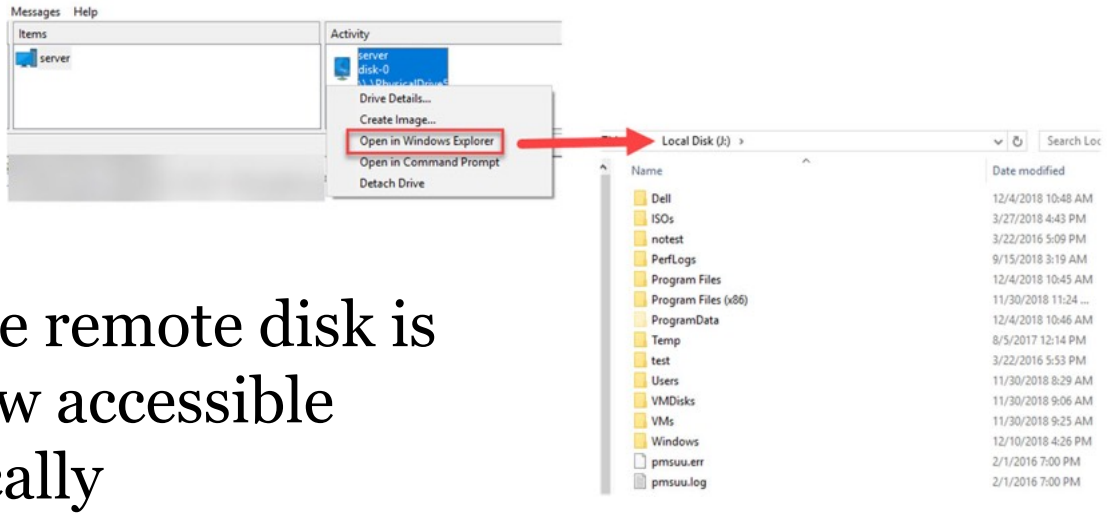
Double clicking on a device under Items is the same as a right click and selecting **Attach Subject target**. This instructs F-Response to reach out to the remote device and get a list of available drives to connect to. Drives in this case includes both physical and logical drives, as well as physical memory. To attach a device, select it from the list, then click **Attach Drive**. F-Response then mounts the remote device locally. This results in a new **"PhysicalDrive"** showing up in Windows. For remote drives that Windows can natively mount (such as an NTFS or FAT formatted volume), this also results in a new drive letter showing up under My Computer. This new drive can be interacted with like any other drive. F-Response takes care of making the remote device seem as if it was local.

F-Response: Accessing Client Resources (3)



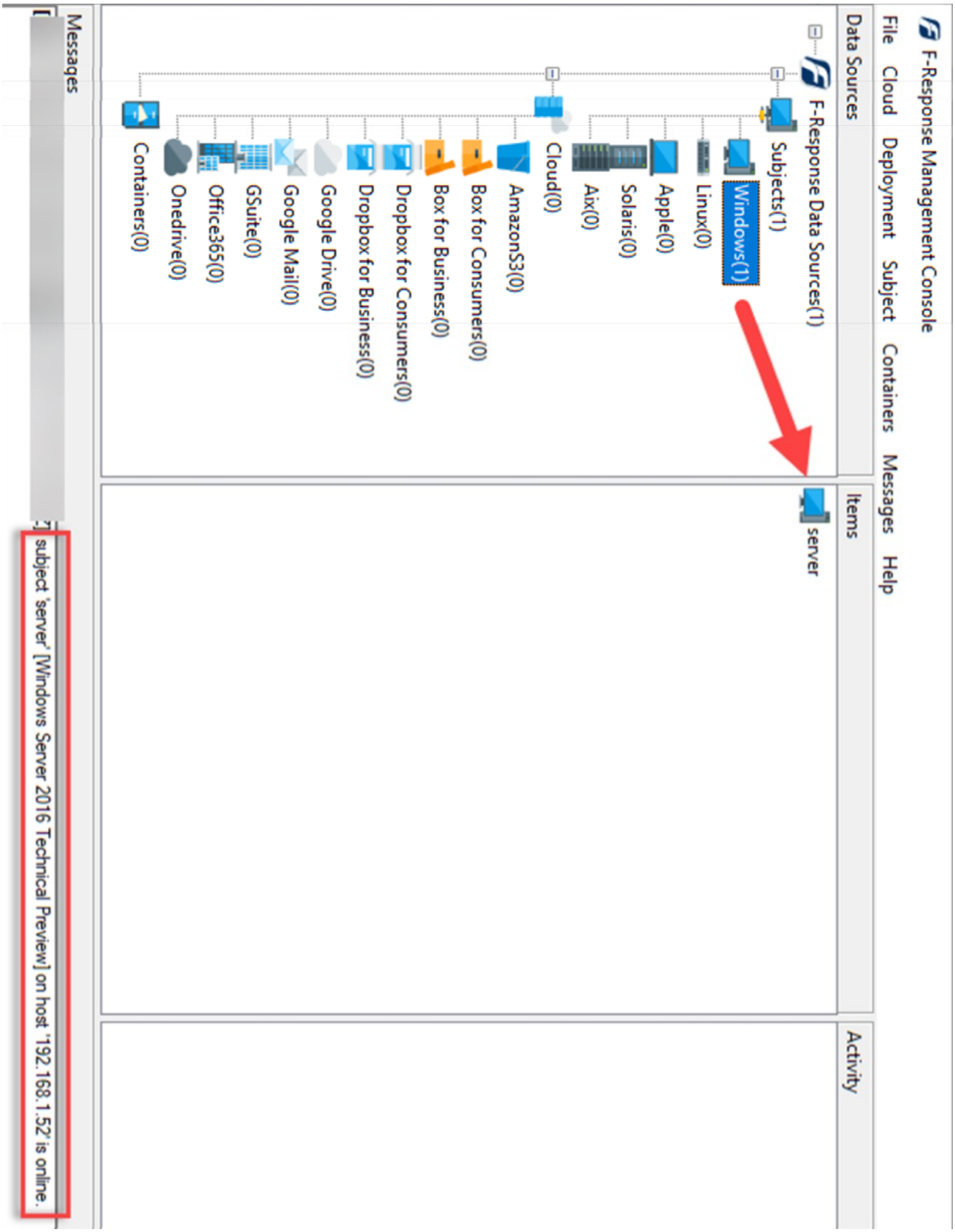
This page intentionally left blank.

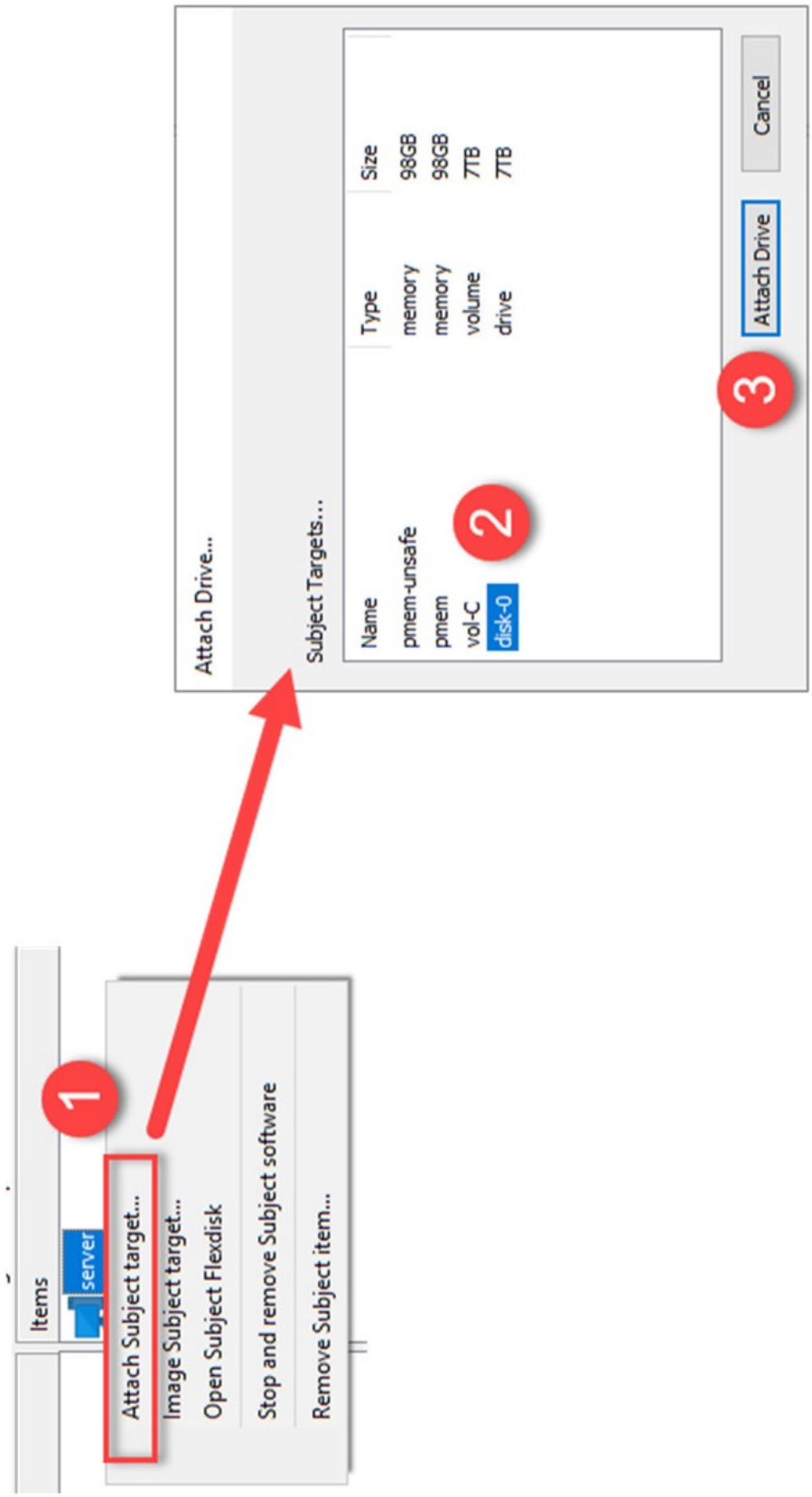
F-Response: Accessing Client Resources (4)

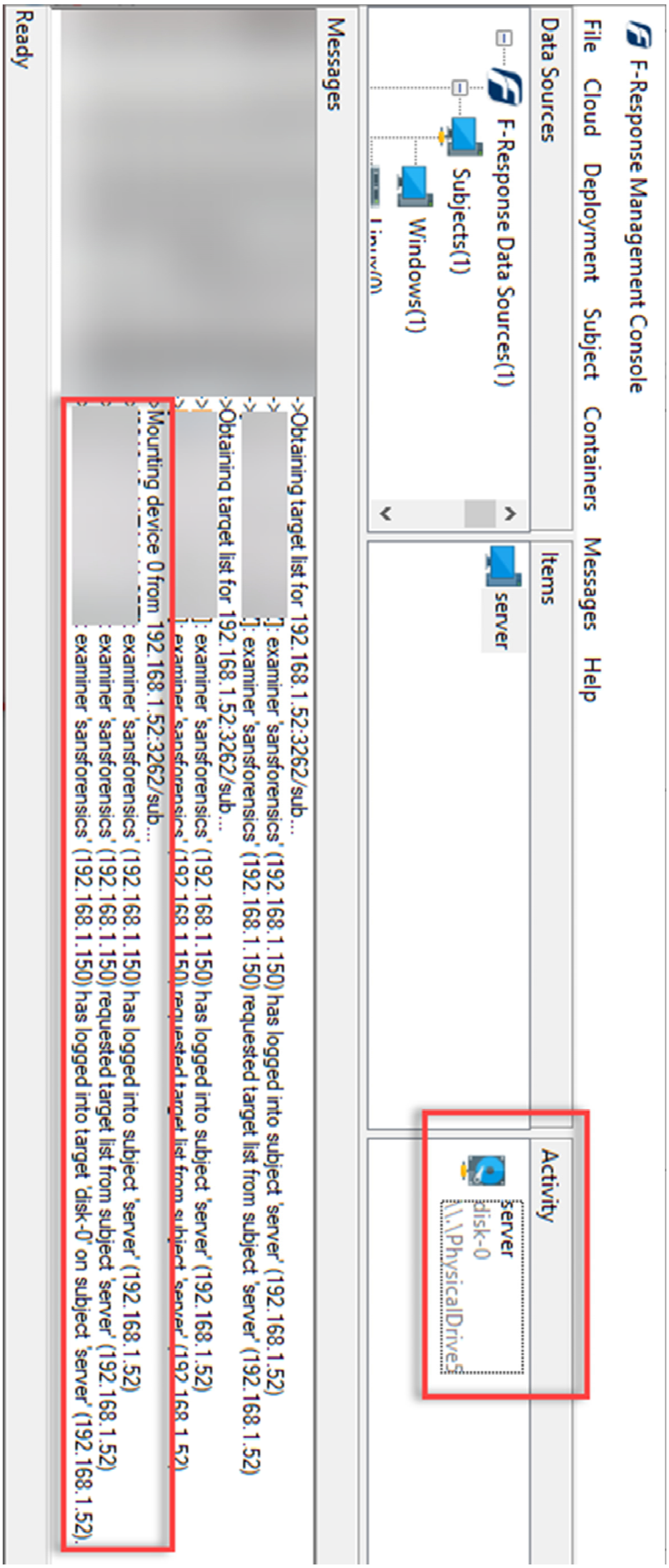


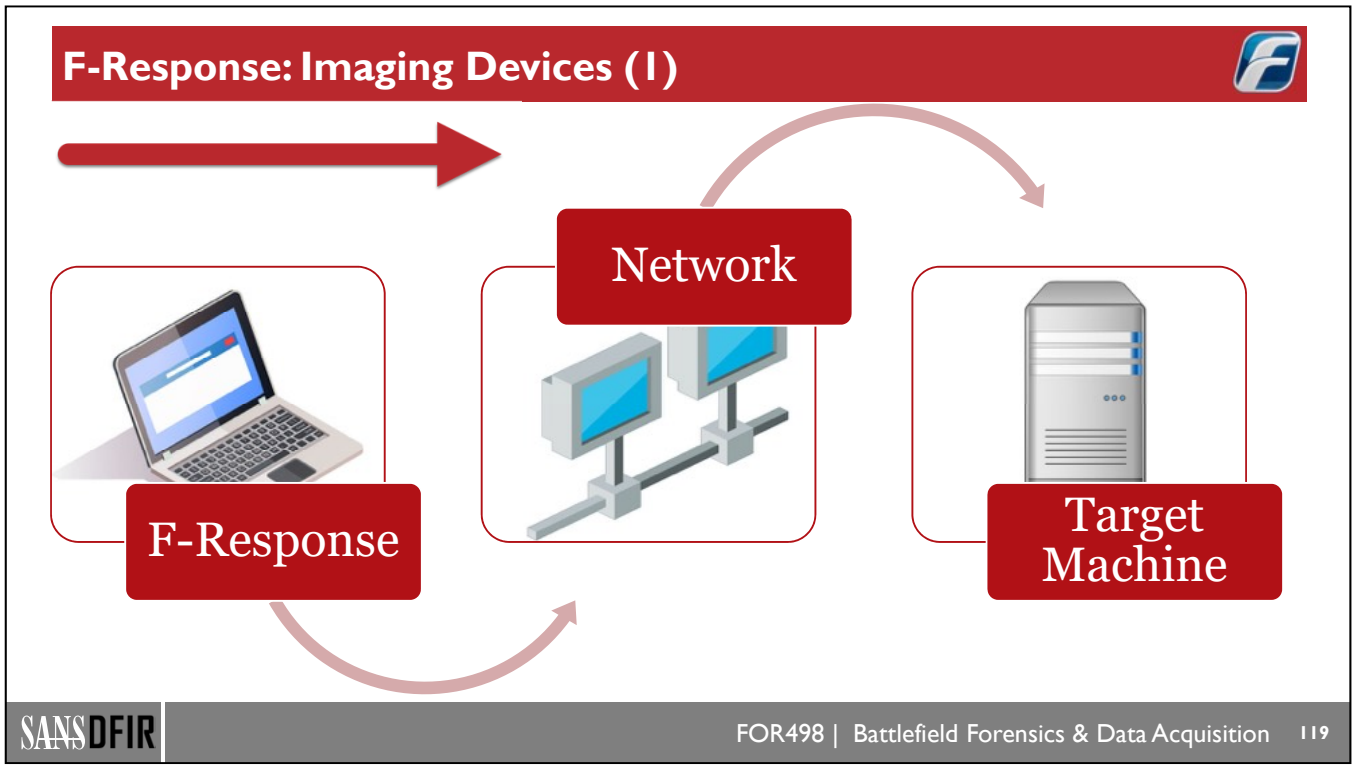
The remote disk is now accessible locally

Once a drive is mounted, it will show up under the Management Console under the **Activity** column. Right clicking on an item under the **Activity** column brings up another context menu that allows for interacting with the newly attached local device. In the example above, the **Open in Windows Explorer** option was selected, which results in being able to interact with the remote drive to browse for files, copy files, and so on.





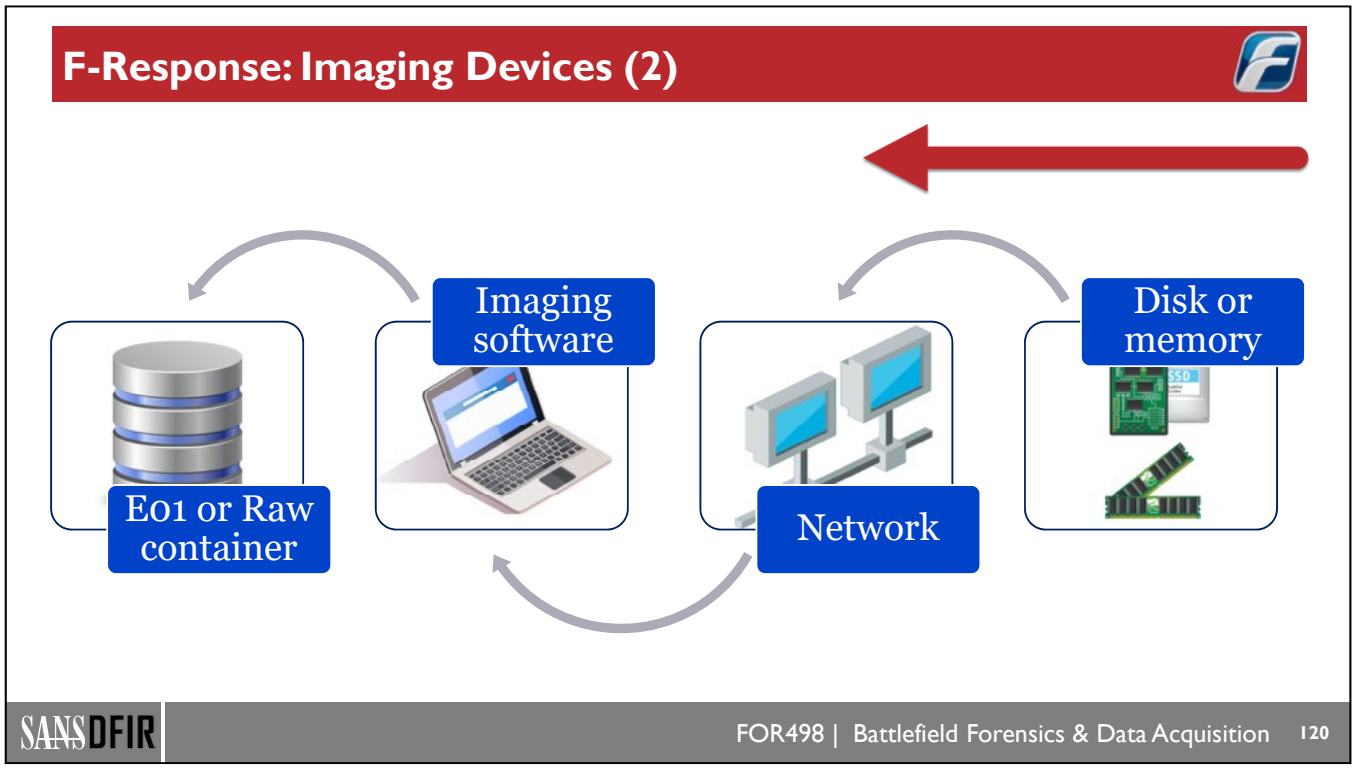




Now that we have seen the various parts of F-Response and how to access remote devices, let's look at the overall process to actually use F-Response for acquisition. Because F-Response acts as a bridge, connecting a remote machine to our local machine, there are no additional moving parts beyond F-Response that are needed.

In the above overview, we have our examiner laptop with F-Response installed. The laptop is connected to a network, either on the same local network as the target machine, or across the Internet. In either case, it is necessary for the ports on the laptop and target machine to be available. In some cases, this will involve port forwarding or allowing an application through a firewall. Once the laptop can talk to the target machine, F-Response is deployed and started. The target machine then checks in with the F-Response Management Console and can now be interacted with.

This is the "Push" phase, where we get F-Response running on the target machine.



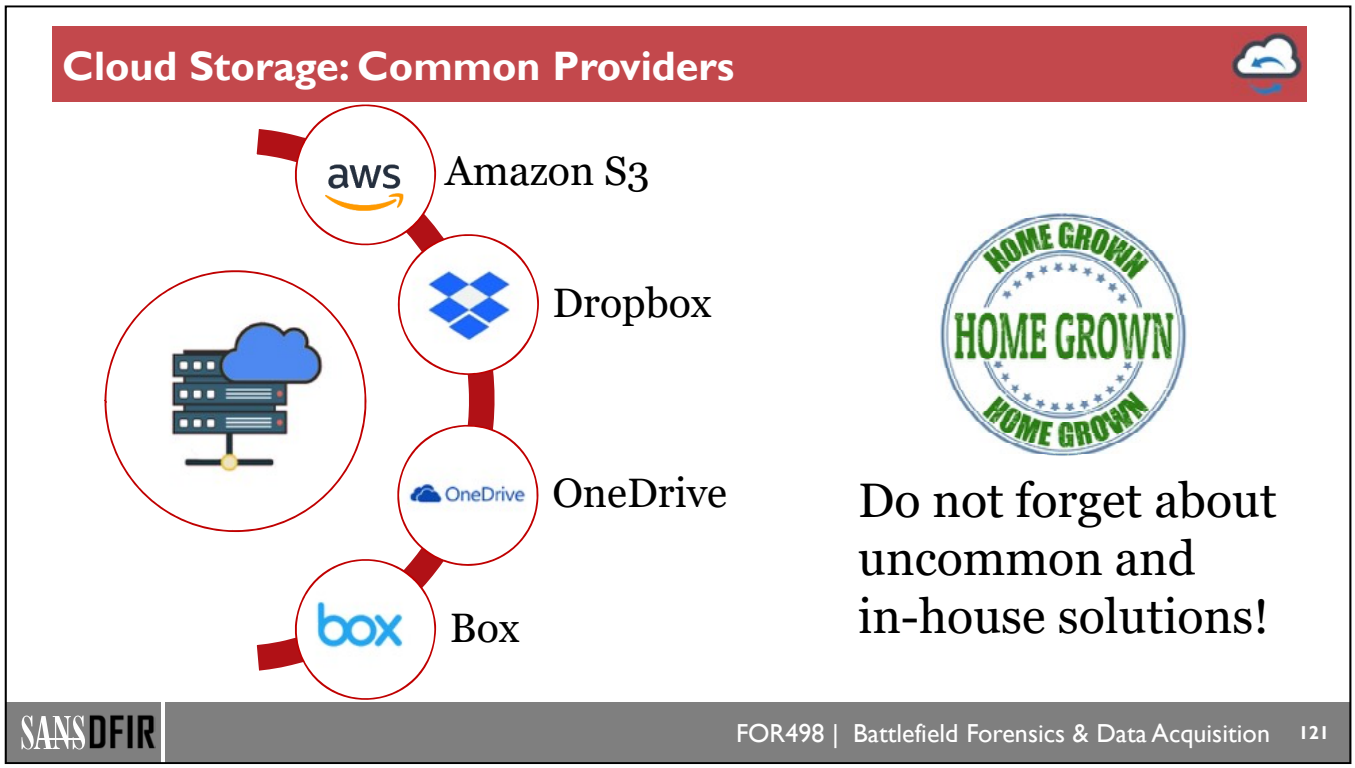
In contrast with the "Push" phase, this is the "Pull" phase, where we start getting data off of a target device, whether it's a disk or physical memory.

With the target machine connected, one or more of the available drives can be attached locally. There are two primary scenarios you will most likely run into; imaging a hard drive, and imaging memory. In both cases, the first thing that must be done is to attach the hard drive or memory to the local machine. In the example above, the laptop is where F-Response is running and where the attached device would end up as a new physical device. In the case of a hard drive, as we have seen previously, the physical device may show up as a drive letter, but in the case of physical memory, a new drive letter will NOT show up because there is not a recognizable file system in memory.

With the physical device available locally on our laptop now, we can fire up imaging software of our choice (recall that F-Response is vendor neutral) and point it at the physical device we wish to collect. When collecting a hard drive, it can be imaged to essentially any container type you want, like E01, AFF, or a raw file, but for physical memory, imaging to a raw file may make sense as memory forensics tools expect to be given a raw copy of memory as opposed to some form of compressed memory. Tools such as X-Ways Imager, FTK Imager, or even a dd variant can be used to collect both disks and memory.

Once the collection has started, it is just a matter of waiting for the collection to finish.

Finally, recall also that F-Response includes imaging built directly into the software. This is a very handy feature because it supports auto reconnect and resuming of the imaging process should the connection drop. When using F-Response to image you do not even have to attach a device locally!

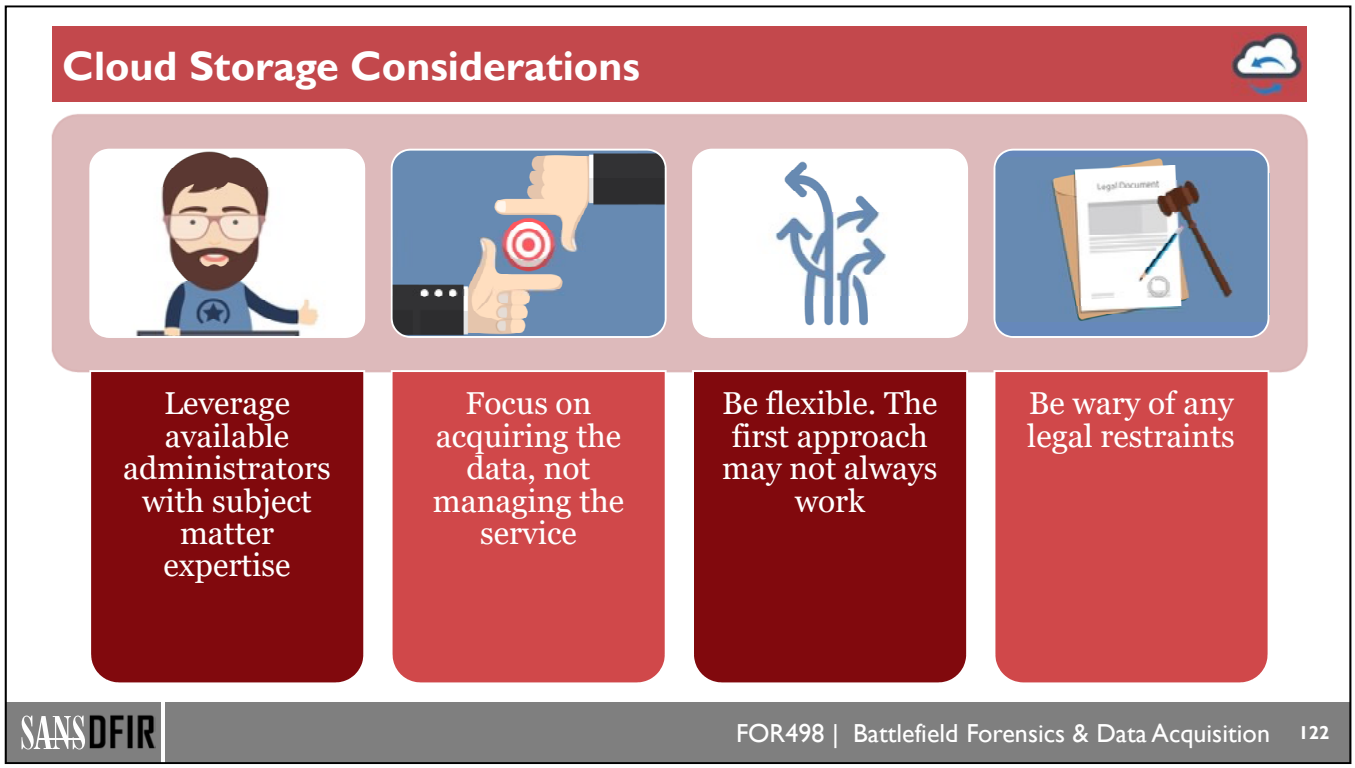


Cloud storage opens a wide range of possibilities for both consumers and businesses. Being able to access files from anywhere on fast and resilient networks negates the need to carry files around or revert to legacy applications like FTP. But with this new-found freedom comes new challenges as well, especially from an acquisition perspective. How can digital forensics practitioners locate and acquire data stored on cloud providers in as forensically sound a manner as possible? As we will see, this is a potential moving target due to how many providers exist, their capabilities for more formal collections, the ability to access the data in a read-only fashion, and so on. To further cloud this issue (see what I did there?) is the ability for end users and businesses to stand up their own cloud storage capabilities using vendor supplied software (Synology Drive for example) to open source projects.

Because of the wide range of possibilities that exist when it comes to cloud storage, forensics practitioners must do their due diligence in looking and accounting for cloud storage in their investigations. This involves everything from asking questions and looking at what software is installed and/or running on a computer, to looking at network traffic. Even once you have identified the cloud storage in use, the challenge may not end there. How do you go about accessing it? Are there forensic tools that know how to collect against your data source in a read-only fashion? If not, can you still collect the data?

The bottom line is that for most cases, acquiring the data via whatever means necessary is completely fine, so long as you document, what you did and why you did it. With that said, this does not mean that the easiest method is the correct one! If there is a way to acquire the data in a read-only fashion, it should be used, even if there is a much easier way to do it that allows for read/write access for example. Why? Because we as digital forensics practitioners, must always strive to be as minimally intrusive as possible.

Whether the data you need to collect is sitting with a common and well supported provider, or you find that your data is sitting on a custom, in-house solution, take the time to identify and document what you are dealing with and the exact steps you took to access and acquire the data.



Before looking at some techniques to acquire data from various cloud providers, let's take a moment to discuss the considerations below. While certainly not exhaustive for every possible scenario that you may encounter, if you keep these things in mind it can help make your job easier.

Leverage available administrators

This truth should be recognized in every investigation. In many situations, there will be a subject matter expert you can lean on to help with the collection, at least from the perspective of getting access to the data itself. Certain platforms can present a significant challenge when it comes to gaining access, such as Amazon AWS (Amazon Web Services)[1]. Configuring access to S3 buckets, with its access rules and permissions, could take a long time to decipher and figure out what is necessary to adjust in order to gain access to data, but for someone whose job it is to manage AWS daily, this may take just a few moments. The other benefit of using available administrators is the reduction of risk when it comes to improperly accessing data based on setting up too broad of permissions, for example. Exposing sensitive data to the world due to misconfiguring permissions when you only needed to access a handful of files is a nightmare no one wants to deal with, so find and allow subject matter experts to help.

Focus on the data

This one is somewhat related to the first in that some of the services where data lives can be vastly complex. Our job is to collect data and tell the story found in that data, not to become the subject matter expert in how to use the service. In some cases, it will become necessary to dabble in things to get familiar or, in the cases where there is not an expert available, to become proficient enough to get the job done. The real take away here is to not get distracted by what is holding the data. You may never run across the cloud storage solution again, so balance the time you spend learning about the platform against the need to collect, preserve, and analyze the data quickly.

[1] Amazon AWS | `https://for498.com/cp6nr`

Be flexible
This is solid advice for many things in digital forensics. Rarely is the path (always) straight. There are times when a given technique may not work. Another circumstance leading to the need to be flexible is being given incorrect information at the start of a project or before arriving on scene, but once you have eyes directly on the situation, things are not as described. We must try to anticipate the unexpected in our planning to minimize the impact the unexpected can take on our cases.

Legal restraints
This can be the most damaging aspect of a case. Proper care and diligence must be taken as it relates to being able to legally collect the data in the first place. If not done properly, then even if you successfully collect it, it may be inadmissible in court if it was seized without a search warrant, for example. This becomes an issue where a computer is located at a residence or business that has some connectivity to storage, but the data is not locally cached on the computer itself. In this circumstance, can you as the incident responder connect to the cloud service to acquire that data? It all depends on the laws in your jurisdiction, the way your legal paperwork is written, and so on. If you find yourself in this situation and are unsure if you are covered from a legal perspective, do not interact with the remote data until you talk to a lawyer. Depending on the nature of the case, you may find you have to get another search warrant for the cloud storage provider, and in this case, the search provider would be the one gathering the data under the account and returning it to you as a part of the search warrant requirements.

Cloud Storage

Service	Access requirements	Access method
Amazon AWS S3	Access key and password | username and password	AWS cli, F-Response, Cyberduck, Web browser
BackBlaze B2	Account Id/master application key | username and password	F-Response, Cyberduck, Web browser
DropBox	Username and password	F-Response, Cyberduck, Web browser
OneDrive	Username and password	F-Response, Cyberduck, Web browser
Box	Username and password	F-Response, Web browser
Google Cloud Storage	Project Id and password	F-Response, Cyberduck, Web browser
Azure storage	Account name and password	Cyberduck, Web browser
Google Drive	Email address and password	F-Response, Cyberduck, Web browser

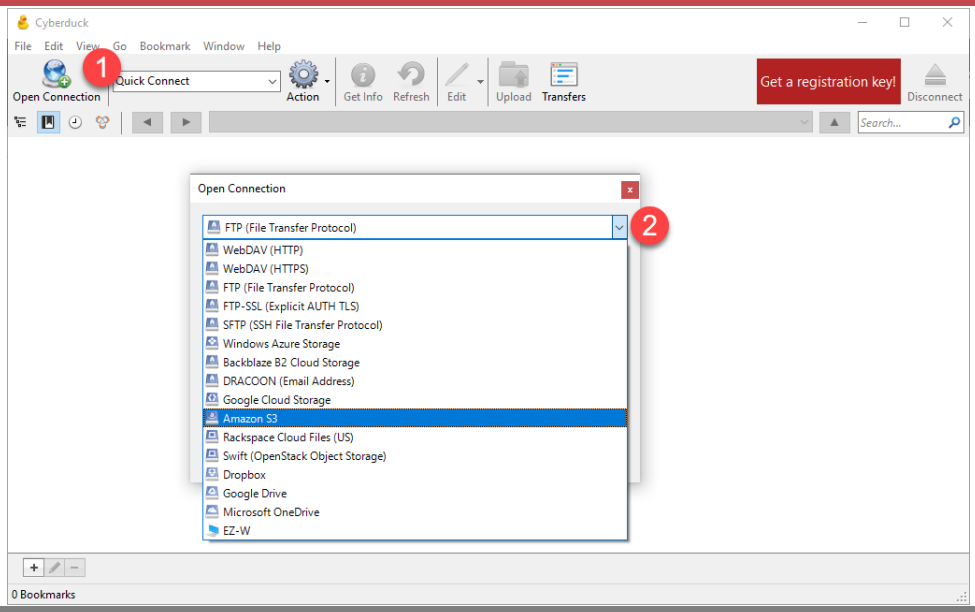
When it comes to acquiring data from a cloud provider, what kind of information do we need? We will look at some specific examples next, but the chart above shows you the common things you will need to access different kinds of data in the cloud. Many of the services provide an Application Programming Interface (API) that allows for accessing data programmatically. The means of authentication changes from provider to provider, with the most common means of access being derived from a username and password. Other services, however, require granting permissions to a user, which generates an access key and password (separate from the user's password in most cases).

Regardless of the control mechanisms in place to access the data, once these are known, we can take steps to access the data. Notice that in every case, a web browser can be used to access the service directly. Once authenticated, the services allow for browsing and downloading of the data stored therein. In the case of something like Amazon S3 (Simple cloud Storage Service)[1] and BackBlaze B2, the username and password may allow for different access than the other access means (access key and password for example). The username and password may be an administrative level password, whereas the access key and password can be set up to be much more modular and limited to a single storage bucket, for example. Beyond using a web browser, Cyberduck is a free piece of software that knows how to connect to many different cloud storage providers. Some providers also provide their own tools, such as the AWS Command Line Interface, that can interact with services in different ways. We will see Cyberduck and AWS CLI usage next.

[1] Amazon S3 | `https://for498.com/7khaq`

[2] BackBlaze B2 | `https://for498.com/fjhs9`

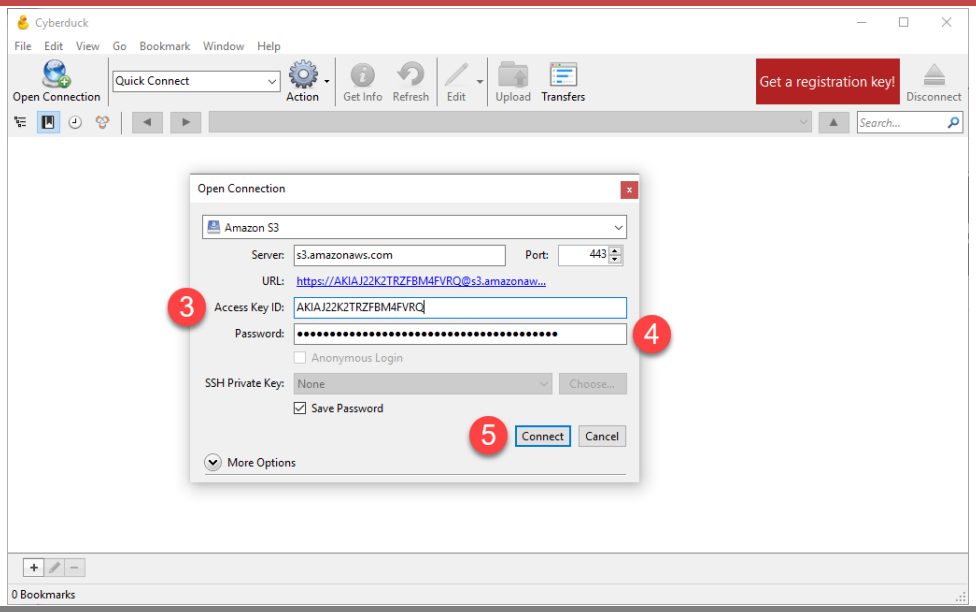
Cloud Storage: Cyberduck (1)



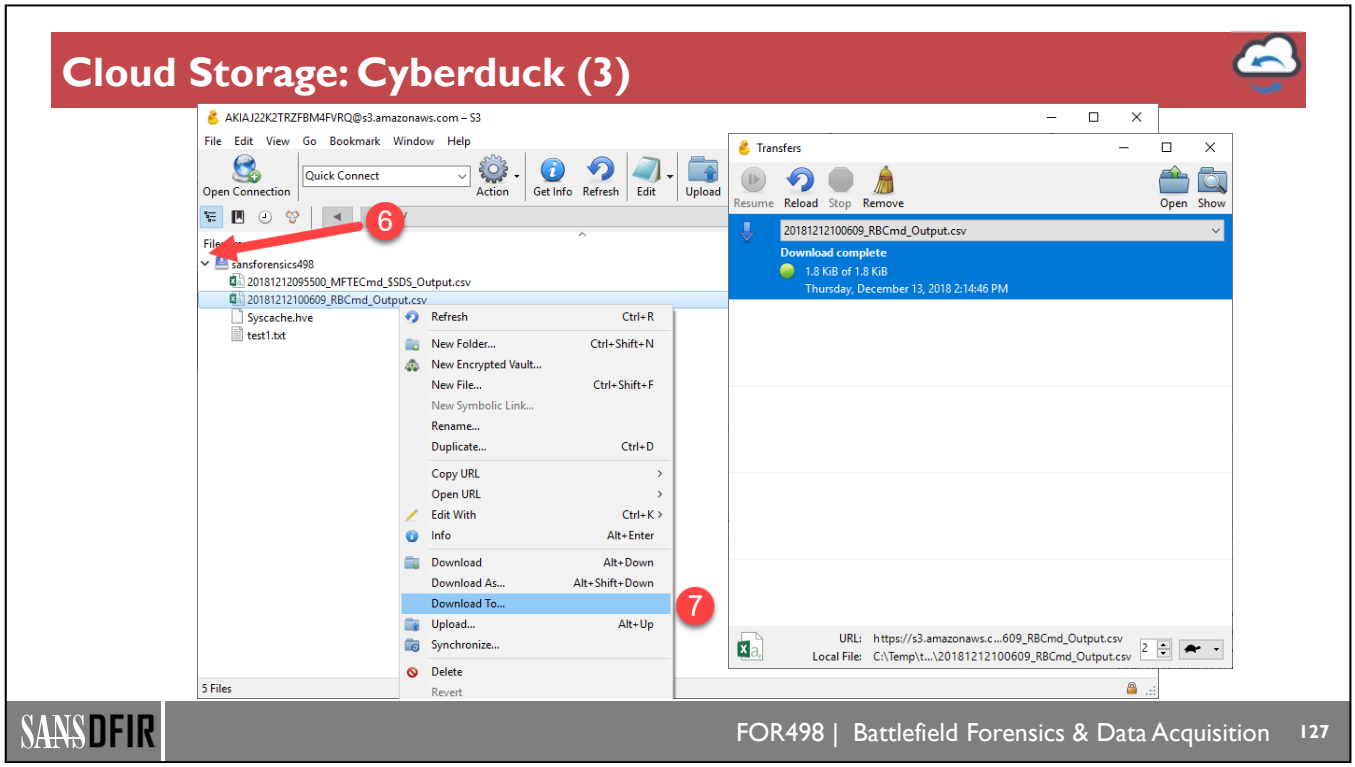
Cyberduck is a Graphical User Interface (GUI) based program that supports a wide range of cloud storage providers. Cyberduck makes it easy, once you have valid credentials, to access and interact with any of these cloud storage providers. It also includes support for things like FTP. SFTP, and WebDAV. Once Cyberduck is downloaded and installed, it can be started like any other program.

Once the program is started, click on the **Open Connection** button to bring up the **Open Connection** dialog. From here, select the type of service you want to connect to from the drop-down list. The requirements for each provider may be slightly different, and the interface will change depending on which provider is selected.

Cloud Storage: Cyberduck (2)



In the example above, we are looking at an AWS S3 connection that requires an Access Key ID and a password. Once both are entered in the proper boxes, clicking the **Connect** button verifies the information and completes the connection.

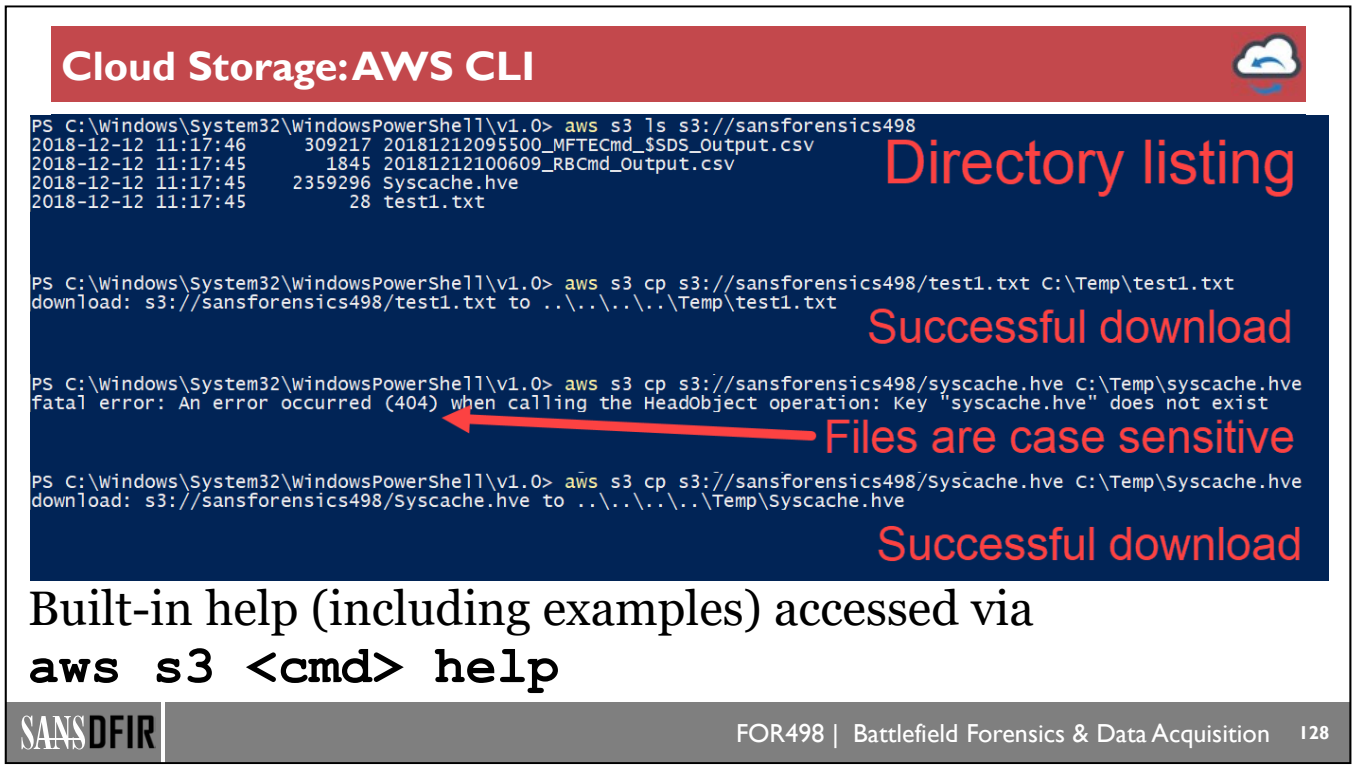


Continuing our S3 specific example, after successfully connecting, Cyberduck displays a list of buckets that are available. Clicking the drop-down to the left of the bucket name expands the list of files (and directories, if present) in the bucket.

Recall from earlier that we always recommend that access to cloud providers be provisioned in such a way that read-only access is the only permission available (if possible). In other words, we do not want the ability to overwrite data, add data, or delete data. This protects us from accusations of tampering as well as mistakes that could result in the alteration of data.

Once the files and directories are displayed, selecting one or more via the mouse and right clicking brings up a context menu that allows for a wide range of options. We are most interested in the "**Download**" related options and generally want to go with the "**Download To…**" option, as this allows us to choose a local file to copy the files from the S3 bucket into. Once files are selected and the download is initiated, a **Transfers** dialog box is shown indicating the overall progress, as well as showing the remote source of the data and where the data is being copied to.

Depending on the amount of data to be copied, the transfer can take quite a while. It has also been observed that for large quantities of data or for transferring many files, using the Amazon AWS command line interface (discussed next) can provide a more stable connection. In these situations, Cyberduck can be used to get a feel for what kind of files are in each bucket, where they are located, etc., while the CLI interface is then used to initiate the mass transfer of data.



The AWS CLI is a command line tool provided by Amazon that allows for interacting with a variety of AWS services. Installers exist for Windows, Linux, and macOS, [1] which will set up everything necessary to connect to and interact with Amazon services.

In our case, we want to see how we can do the same kinds of things we did with Cyberduck, but in the AWS CLI[2]. Once the CLI is installed, start either a cmd or PowerShell session in Windows, or the terminal of choice for other operating systems. To verify things were installed property, type **`aws --version`** and press **Enter**. If you see version information returned, things are working properly.

Like Cyberduck, we need AWS S3 credentials in order to connect. To use them, we need to use the configure command. Typing **`aws configure`** prompts for various pieces of information, including the access key ID and password. Once the necessary credentials are provided, AWS has what it needs to connect. To verify that things worked, issuing the command **`aws s3 ls`** will display a list of buckets your credentials have access to. If you see a listing of bucket names, things are working as expected.

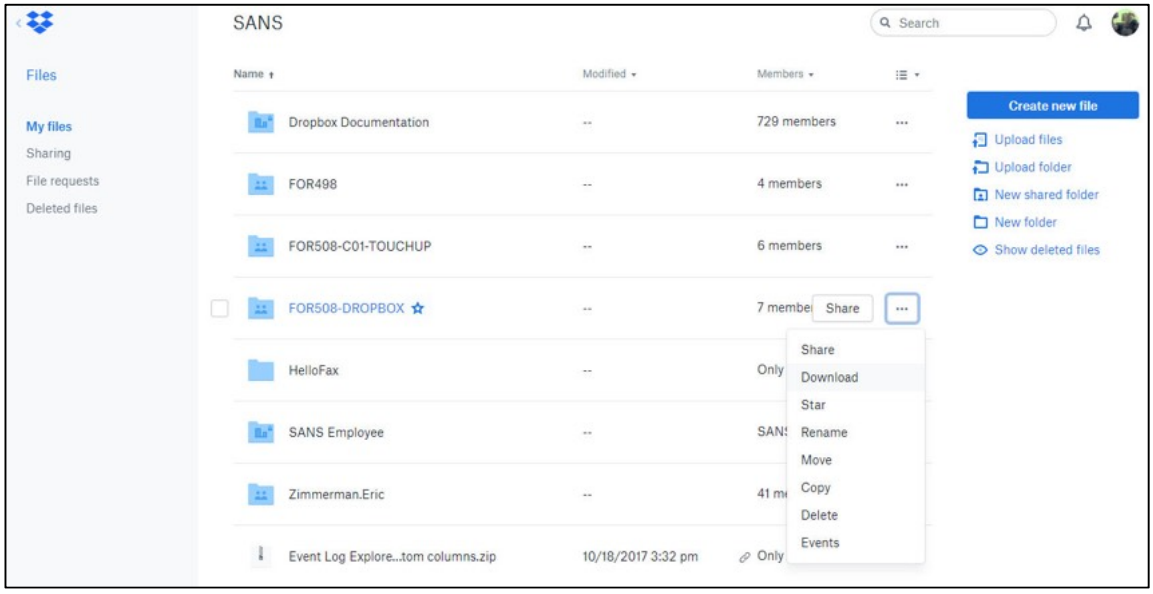
Before getting into how to transfer files, it is a good idea to know how to get more information about the different commands available, etc. The AWS application has built in help that can be accessed in the form of **`aws <cmd> help`** and **`aws <cmd> <subcommand> help`**. For example, **`aws s3 help`** would display information about how to interact with the S3 service, while **`aws s3 ls help`** would tell you all about the **`ls`** command. Example usage is also included in help screens, which makes getting up to speed very easy.

To transfer files, supply the full path to the remote file, along with the full path to the local file, as shown in the examples in the screenshots.. To copy all the files in a bucket, use the command **`aws s3 cp s3://sansforensics498 C:\Temp\ --recursive`** to download all files available under the "sansforensics498" bucket.

Many more features are available in the CLI and of course, these activities can be scripted, saving time and allowing for a standard operating methodology to be designed.

[1] Installing the AWS Command Line Interface | https://for498.com/rbg0p
[2] Using Amazon S3 with the AWS Command Line Interface | https://for498.com/1vhtj

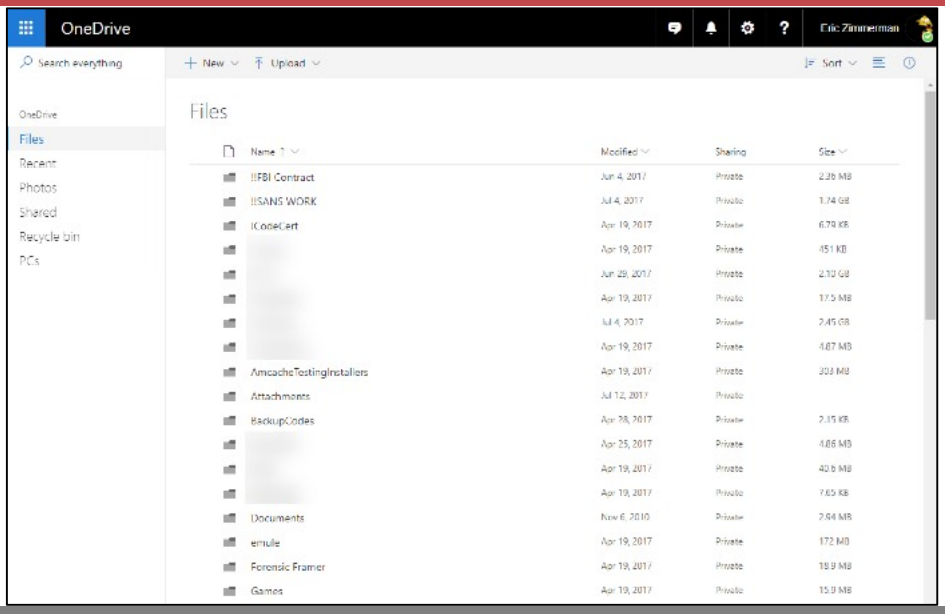
Cloud Storage: Old School (1)



Should there be no dedicated GUI or CLI tool to access, a trusty web browser interface to the data is almost always going to be available, albeit with different levels of functionality. As with everything else, valid credentials are required to access the web browser interface. Once there, the cloud storage providers typically provide a way to download files in groups (as opposed to one at a time, the horror!) by checking boxes and then initiating a single download command. This usually generates a zip file or other archive that gets downloaded by the browser. When the download finishes, unarchive the contents and you have a local copy of the data.

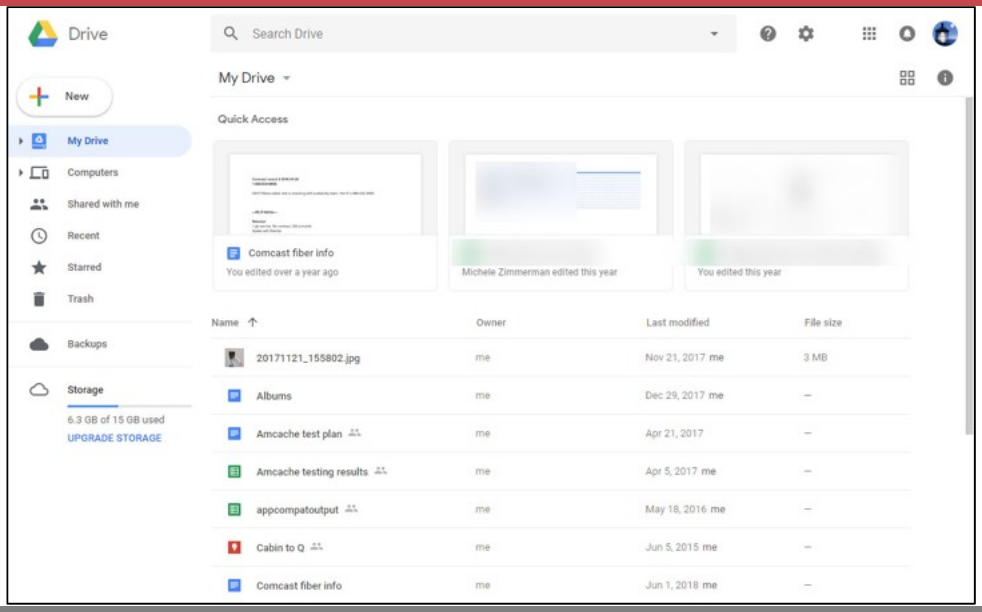
When using a web interface for these kinds of things, keeping track of your interactions is very important. For this reason, consider the use of video recording software like Camtasia, that can be utilized to start recording the screen even before the web browser is used to sign into the cloud provider. With video rolling, authentication, file selection, and downloading can all be monitored and tracked. This is useful for later documentation as well as visual proof of your actions while interacting with the cloud provider. A video is much better proof than “he said, she said” when it comes to defending yourself against allegations of tampering.

Cloud Storage: Old School (2)



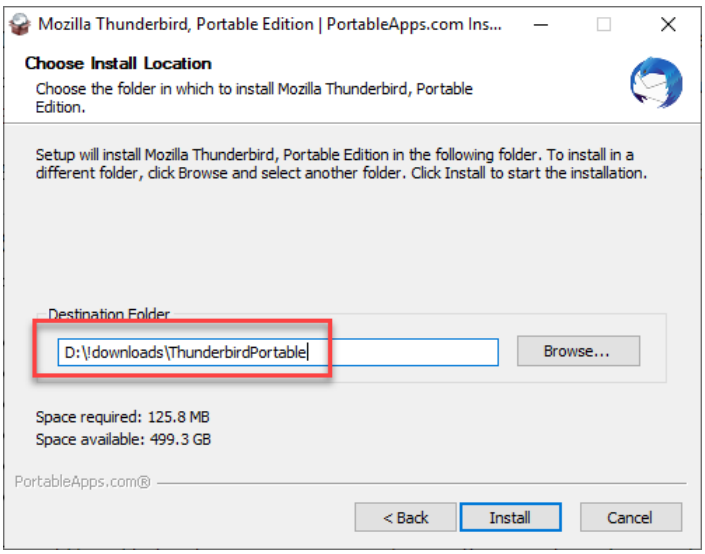
This page intentionally left blank.

Cloud Storage: Old School (3)



This page intentionally left blank.

Cloud Storage: Email Acquisition (1)



Select a directory to extract the program to

In the previous few slides we saw plenty of examples where traditional storage is made available in the cloud by various providers, but what about other means to store data that just about everyone has access to, such as email? Years worth of information, including files, is available in users email accounts. In many cases, multiple gigabyte quotas make it easy to never delete messages, so looking at email stored on a remote server (here we see the notion of the cloud again) can be very fruitful.

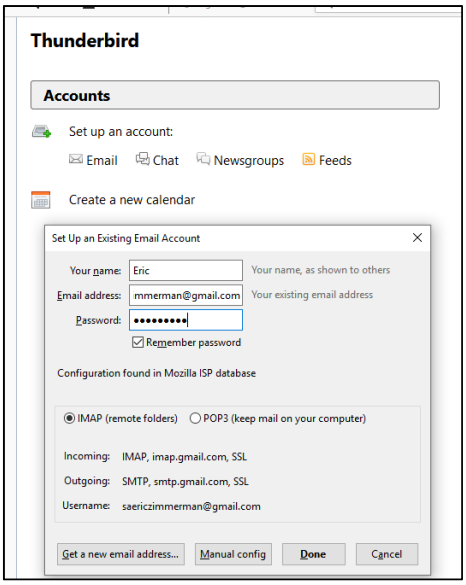
Once we locate an email account we wish to collect, we first need to determine how access to the email server is accomplished, or more specifically, what protocol is used to access the mail server. If its Post Office Protocol (POP), you may be out of luck because in most instances, when a client checks for email, the email is downloaded to the client and the email is removed from the server. Internet Message Access Protocol (IMAP) on the other hand, keeps the emails on the server and simply downloads a copy to the client machine. This means that email persists on the IMAP server, and therefore, can be collected forensically. Fortunately, IMAP is the most widely used deployment today.

Some popular tools to collect email are dedicated tools for the job, like F-Response or Aid4Mail [1], but it is also possible to use something like Thunderbird Portable [2]. It works very well and provides a means to have a self-contained copy of the data from an IMAP server (associated with one or more email addresses). Using Thunderbird to access an email account would be done no differently than when you are setting it up to manage your own email. You will need a working username and password to give to Thunderbird, which will then be used to find the appropriate email server, verify the credentials, and then connect to the mailbox. Once connected, go “offline” in Thunderbird and you will be prompted to download a copy of all the email across all the folders locally. Once this download is complete, it is important to go back into Thunderbird properties and remove the password so that it cannot go out check for new mail when you start Thunderbird later to review the email. This could get you into legal trouble.

With the email mirrored locally, you can exit Thunderbird and then copy the entire Thunderbird portable directory to a USB drive, zip it and burn it to DVD etc., to preserve the data. Working with a copy of the data, starting Thunderbird then allows you to search, sort, group, filter, and review email exactly as it would appear to the end user.

[1] Aid4Mail | https://www.aid4mail.com/
[2] Thunderbird Portable | https://for498.com/d4x9i

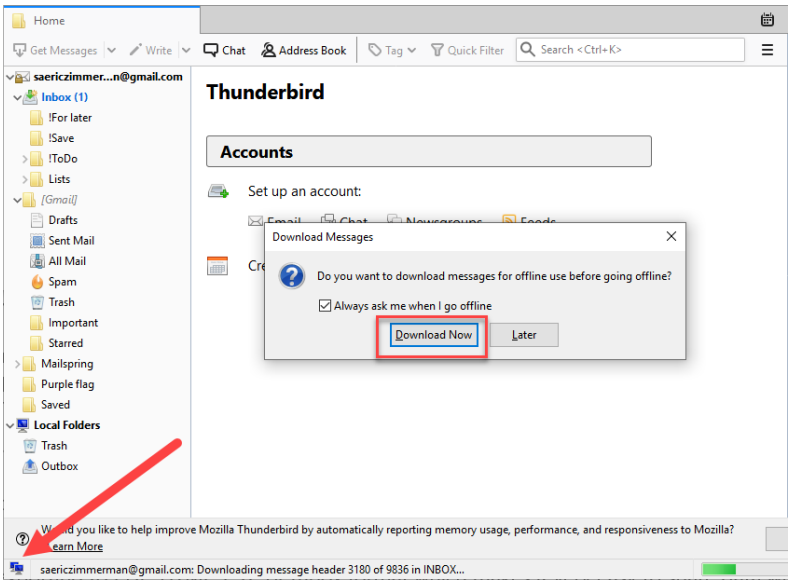
Cloud Storage: Email Acquisition (2)



Wizard walks through connecting to account

This page intentionally left blank

Cloud Storage: Email Acquisition (3)

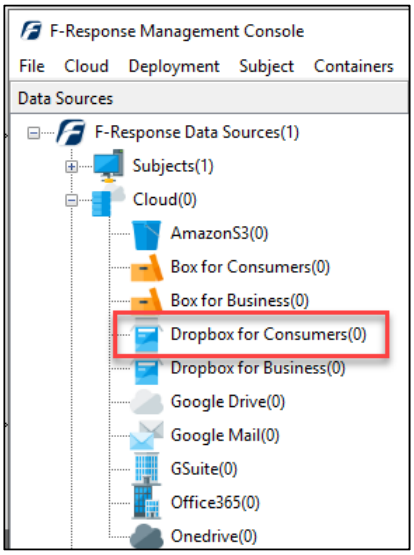


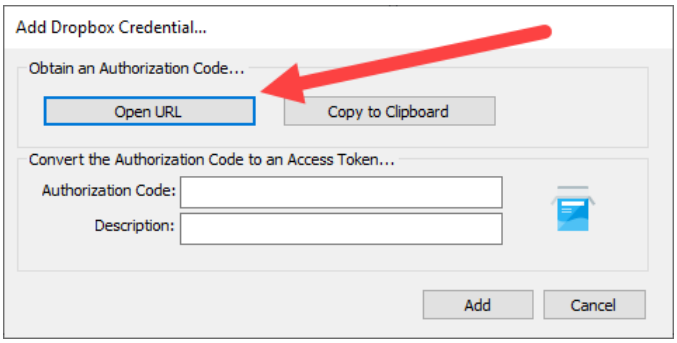
Click to go offline and sync account

NOTE: When using Thunderbird, make sure you use the PORTABLE version because the default version comingles its profile data with the account currently logged in to Windows. By using the portable version of Thunderbird, the entire program, including all its profile data, is kept in a single directory which can be moved/copied without affecting the data that Thunderbird has collected.

Other commercial tools, like Aid4Mail, can also perform acquisition and adds features like acquiring more than one mailbox at a time. It allows for exporting to PDF, HTML, PST, or mbox format which makes it very easy to share data with other analytical platforms.

Cloud Storage: F-Response Take Two! (1)



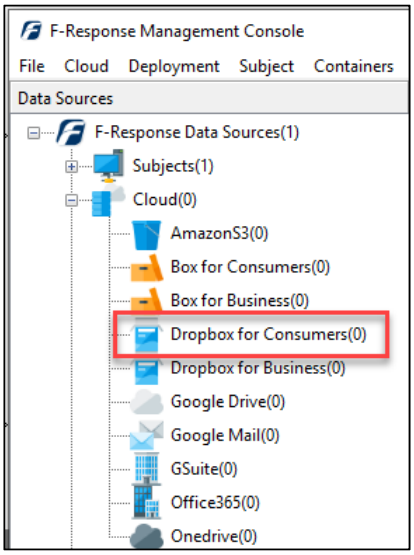


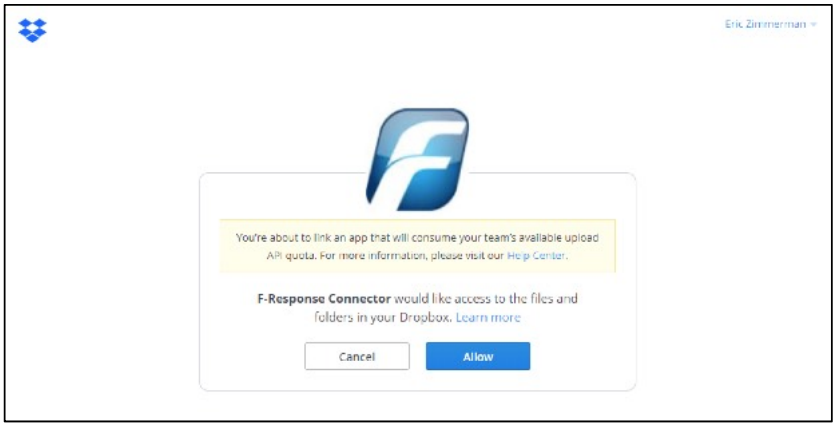
SANS DFIR FOR498 | Battlefield Forensics & Data Acquisition 137

We aren't just limited to platform specific tools when it comes to cloud collection though. Once again F-Response is useful here. Under the Cloud folder is an extensive list of available cloud providers that F-Response knows how to connect to and acquire data, all from a single interface. This can save a significant amount of time.

While connecting to each provider is slightly different, F-Response walks you through the process of connecting to and authorizing against the data source as well as initiating a collection in a straightforward manner. To start the process, double click on a cloud provider, or use the relevant option under the **Cloud | Add Cloud Credential** menu at the top.

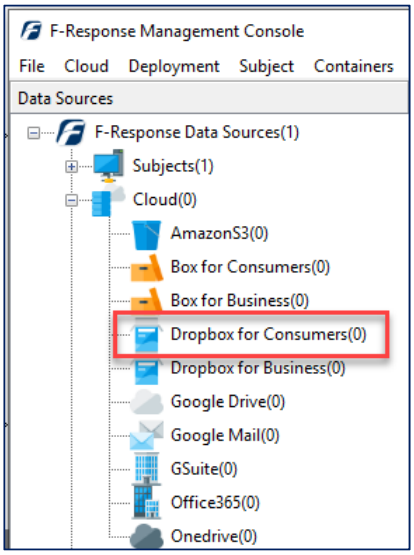
Cloud Storage: F-Response Take Two! (2)

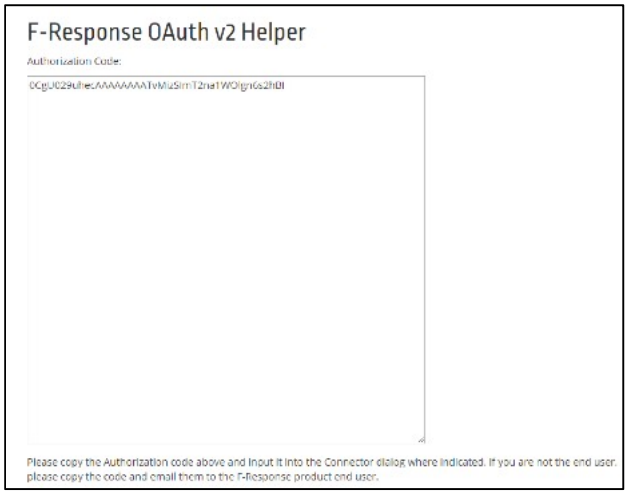




In the example above, a Dropbox collection is being configured. As with any of the supported providers, you would need to have the username and password of the target account so you can use it during the process. Notice that we are already signed into a Dropbox account when Dropbox is prompting for permission to allow the F-Response Connector to access the contents of the Dropbox account. If we were not signed in, we would first be prompted to provide the username and password on the Dropbox website. Once authenticated on the website, Dropbox would show the dialog asking for permission.

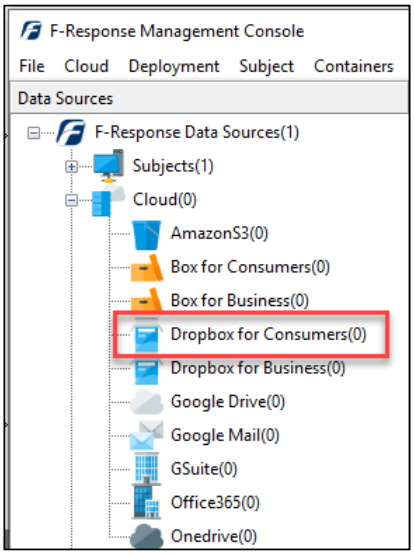
Cloud Storage: F-Response Take Two! (3)

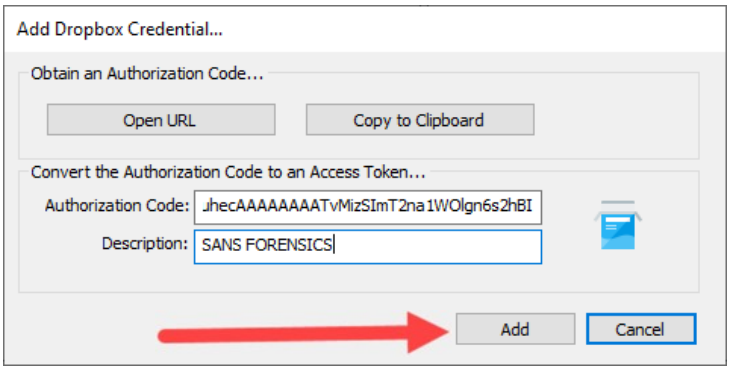




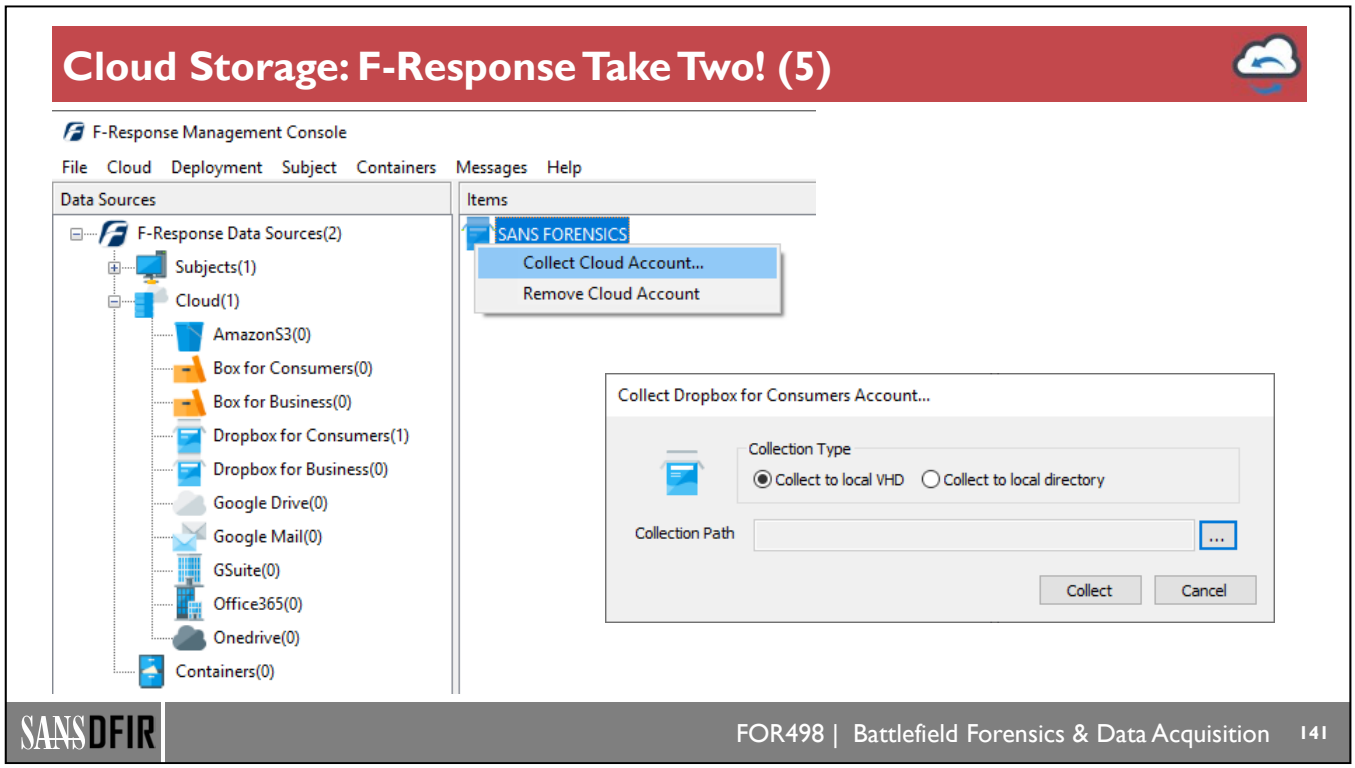
This page intentionally left blank.

Cloud Storage: F-Response Take Two! (4)





Once the authorization code is generated, it can be copied back into the F-Response credential dialog, along with a **Description** for the Dropbox connection (we used "SANS FORENSICS" here). With the necessary information in place, clicking the **Add** button finishes the connection and makes the target Dropbox account ready for collection.

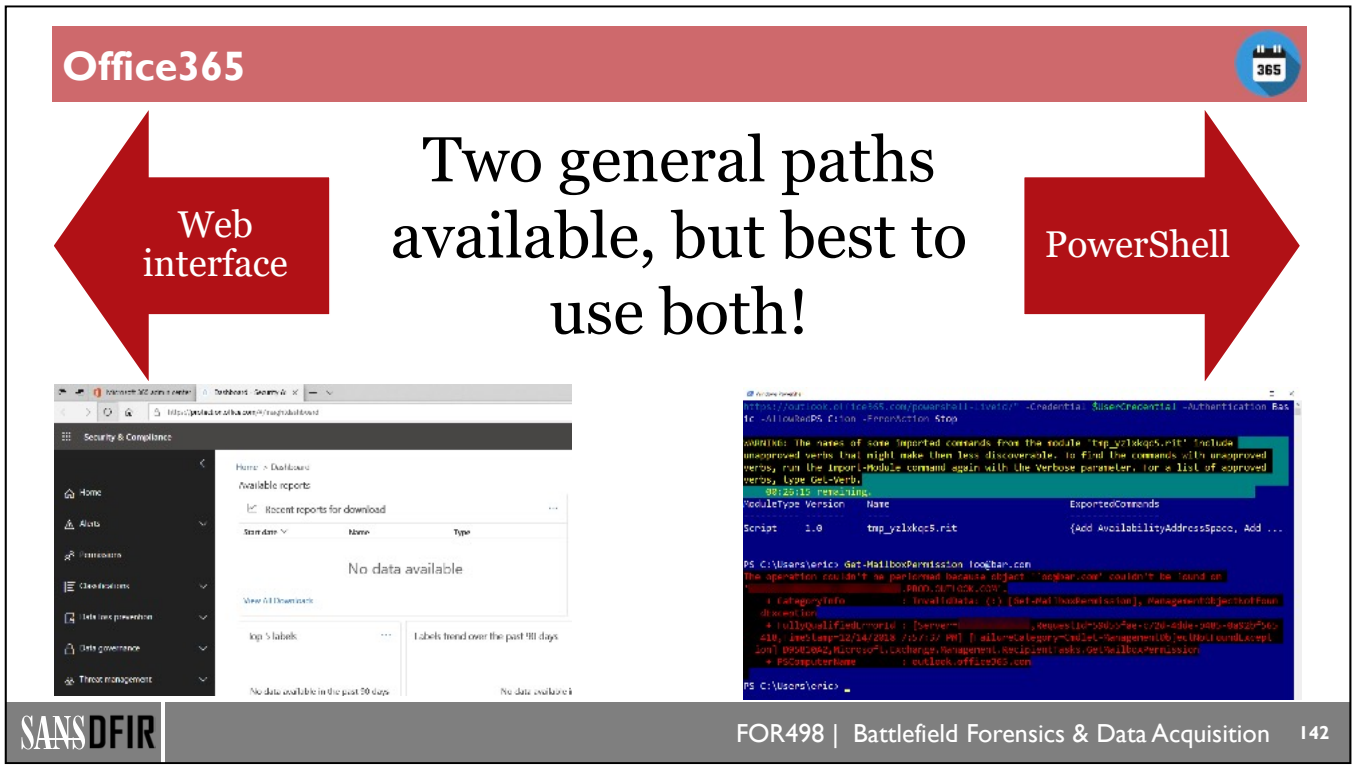


Once the Dropbox account has been added to the Management Console, interacting with it is the same as we saw earlier. Right clicking on the newly added item brings up a context menu. Selecting **Collect Cloud Account** brings up another dialog that lets you choose between writing everything to a VHD container or directly to a directory.

This example is specific to Dropbox, but the rest of the cloud providers will work in a similar fashion. Earlier we used Cyberduck to interact with an Amazon S3 bucket, but we could have just as easily (maybe even easier!) used F-Response to do the collection.

F-Response supports both consumer level and business level collections. The difference is that with the business-oriented options, you would not need everyone's username and password. Rather, you just need a single administrative account that already has permissions to access other accounts in the same business unit. This makes things much easier when doing more than a single collection.

Because F-Response is so flexible and feature rich, it can really simplify complex collection requirements easily for many of the typical scenarios you may find yourself in. By getting comfortable with F-Response and its capabilities, you can spend more time on analysis vs. trying to figure out the best way to collect evidence outside of your already existing process.



Collecting data from Office365 from an investigative perspective will typically revolve around two primary things: logs, and emails in the form of PST files. Since Office365 was released, Microsoft has made more and more features and auditing available via web sites, and increasingly, via PowerShell. These will be the two primary means of accessing the data needed for an Office365 case. So which one is better, and which one should you use? We recommend you use both, as there are some things you can only do in one and not the other, and vice versa. By leveraging the information and capabilities found in both places you will be sure you have the most complete picture possible for your investigation.

The primary log file leveraged for Office365 (O365) cases is the Unified Audit Log (UAL). This log is an aggregate of several other logs and is available via both the web site and PowerShell (we will see how to access it later). Historically, audit logs have been disabled (as hard to believe as this is) by default, but recently Microsoft has taken steps to start enabling audit logs for all tenants in O365. Even once enabled, it is possible to disable logging as well. One of the first things you should do in an investigation is check to see what logs are enabled, when they were initially enabled, and how far back the logs go.

There are some other things to be aware of when it comes to the data available in audit logs. UAL does not show message reads, folder access, logouts, search terms used, when attachments are open, or how long a session was active. Some of these missing pieces of data would be immensely helpful, and it is anticipated Microsoft will continue to improve the data available in UAL.

So how do you get at the logs that are available? Ideally, you will be provided a Global Administrator (GA) account (even if it's on a temporary basis) so that you have access to the areas of the O365 related websites to collect logs, generate PSTs, adjust permissions, and so on. A GA account is a like a domain administrator account, so protect it as such. Even with a GA account, depending on what you need to do, you may have to give the GA account additional permissions. For example, to generate PST files for one or more users, the GA account must be given the eDiscovery Manager role. [1]

Finally, remember we are primarily focused here on acquiring the data and as such, we will not be spending a ton of time doing the analysis of the data.

[1] eDiscovery Permissions | `https://for498.com/ci-ld`

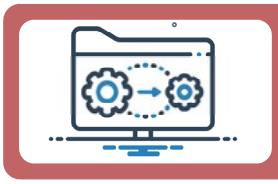
Office365:Web Resources

https://protection.office.com

- User Audit Log and content search
- Generate PST files

https://portal.office.com/adminportal

- User lookup and Licensing
- Links to dedicated admin sites (Exchange, etc.)



https://manage.windowsazure.com

- Active Directory resources
- If in use, may have additional logs

Each of the websites above offer different avenues of collection and investigation, with the first two being the most commonly used resources for O365 related cases. Here you can generate User Audit Log (UAL) files for certain date ranges via the Search and Compliance section (we will see how to do this via PowerShell next), perform lookups of users to see what licensing they have applied, what permissions exist for user mailboxes, and look at active directory related information (if used on the tenant).

When generating UAL files, it is recommended to generate one per user in order to keep things simple, and data sizes more manageable. UAL log retention is anywhere from 30-180 days and is configurable by the tenant administrators. Once the UAL files have been generated, they can be reviewed in the web browser, but be careful when it comes to the download options. Make sure you select **Download all results** or you may only get the log entries visible on the screen. Note that the log files themselves are in json format, which can present some analysis challenges due to the nature of the nested data.

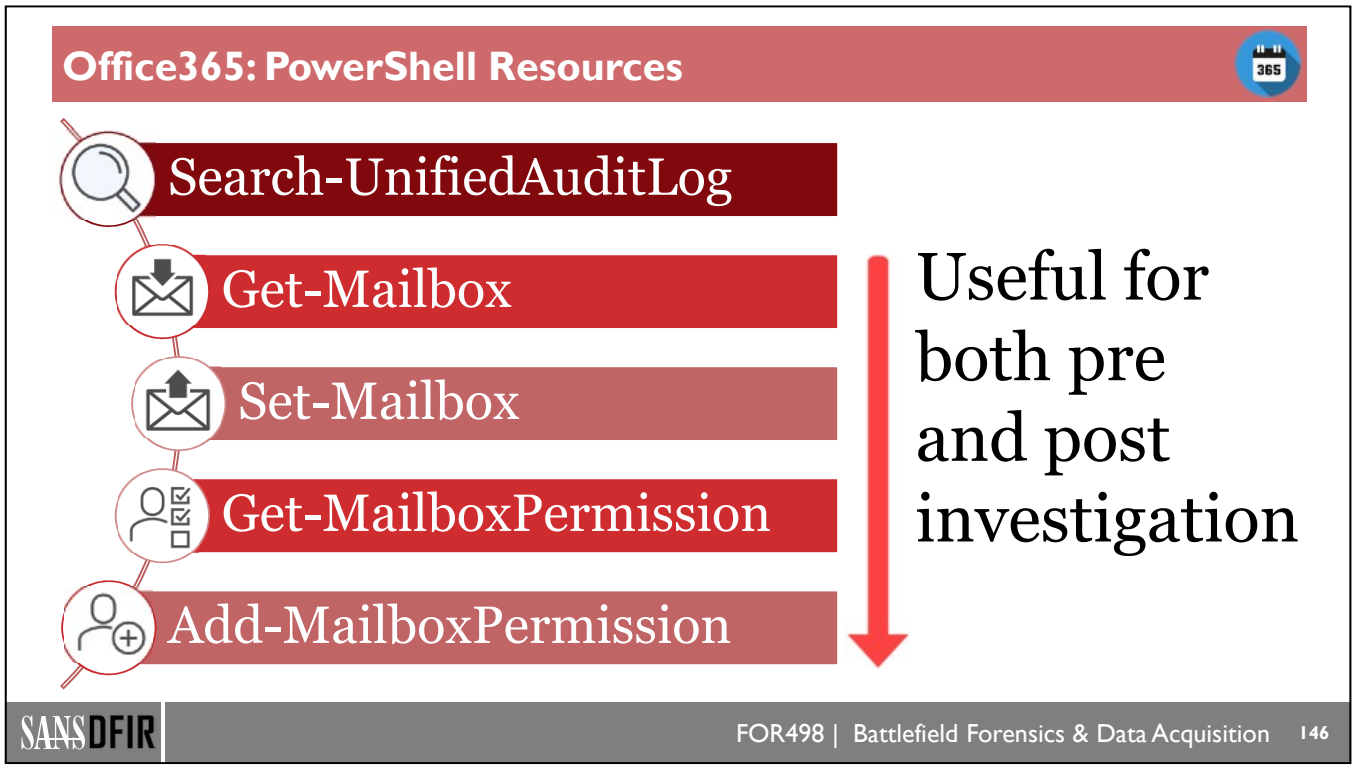
The Security and Compliance section is also where PST archives of user accounts can be generated for download. For whatever reason, this functionality seems to require Internet Explorer for it to work properly, so be sure you have access to IE should you need to generate PST files. We recommend one PST file be generated per user for the same reasons mentioned above.

Rather than document each and every step to perform these actions, it is more important to know where to find the location of how to perform these actions. This is because Microsoft has redesigned these management interfaces several times. Rather than provide details that may not be accurate as of the time you are reading this, spend a few moments on the sites to find the proper area where these actions can be performed. When the ability to perform an action via a web browser exists along with an equivalent PowerShell method (discussed next), it may make more sense to learn the PowerShell way, as that is much less likely to change over time. With that said, there are a few URLs at the bottom of this page that will walk you through the current process.

Depending on the types of queries and actions being performed, we have found that Microsoft Edge and/or Internet Explorer work best. If you are using Chrome or Firefox and find things just aren't working as you expect as it relates to dropdowns, etc., try one of the Microsoft browsers.

Finally, in cases where a tenant is using Active Directory hosted in Azure, you may have another source of logs available. These logs may include (depending on the agreement the customer has with Microsoft) reports you may have access to, such as, "users flagged for risk" and "Risky sign-ins" that can quickly generate solid leads. The downside to these logs is that, when available, they only have between 7 and 30 days of retention.

Export Office 365 mailboxes to PST using eDiscovery | `https://for498.com/jh5ct`
Exporting PSTs from O365 | `https://for498.com/aojur`



PowerShell is where Microsoft has devoted a lot of time to developing robust management features for O365. It is not perfect however, and often a blended approach with both web site resources, and PowerShell scripts will be the best way to work these types of cases. The PowerShell offerings from Microsoft compliment what is available on the O365 Admin center and allow for things like managing user accounts, SharePoint Online, Exchange admin, migrating users, and so on. [1]

From an investigative perspective, we are not so much interested in the creation of new users and management of the various components of O365, but rather, we want to focus on how we can use PowerShell to get access to log files to investigate O365 compromises.

The primary cmdlet to interact with O365 logs is Search-UnifiedAuditLog[2]. This allows for more programmatic access to the UAL than is possible with the web interface. Another possibility for programmatic access to this data is by using the Office 365 Management Activity API. [3] This is useful for both limiting the amount of data retrieved, as well as scripting the downloading of log files, splitting it out per user, and so on. The GA account we previously discussed will be required to authenticate before you will be able to use this command.

There are many options that can be used with Search-UnifiedAuditLog but the most useful are the StartDate, EndDate, Operations, and IPAddresses switches. These allow you to filter things based on a timestamp (perhaps based on when a phishing email was clicked), certain types of operations (MailboxLogon, MoveToDeletedItems, UpdateInboxRules, etc.) and interesting IP addresses. Both StartDate and EndDate are required parameters. The other operators are generally used once an initial review of the data has happened. For example, looking for logons may yield 5 IP addresses, with two of them being outside the United States. These two IP addresses can then be pivoted on to look for other compromised accounts, and so on.

Other data sources to pivot on include data from log files downloaded from the web sites previously discussed.

The Get/Set pairs are useful to interrogate mailbox settings and permissions, both to see what the current state is (when looking for malicious activity), and also to add additional permissions to a mailbox in order to search it, etc. This is where things like Add-MailboxPermission come in to play, as it may be necessary to add your GA account to an existing mailbox.

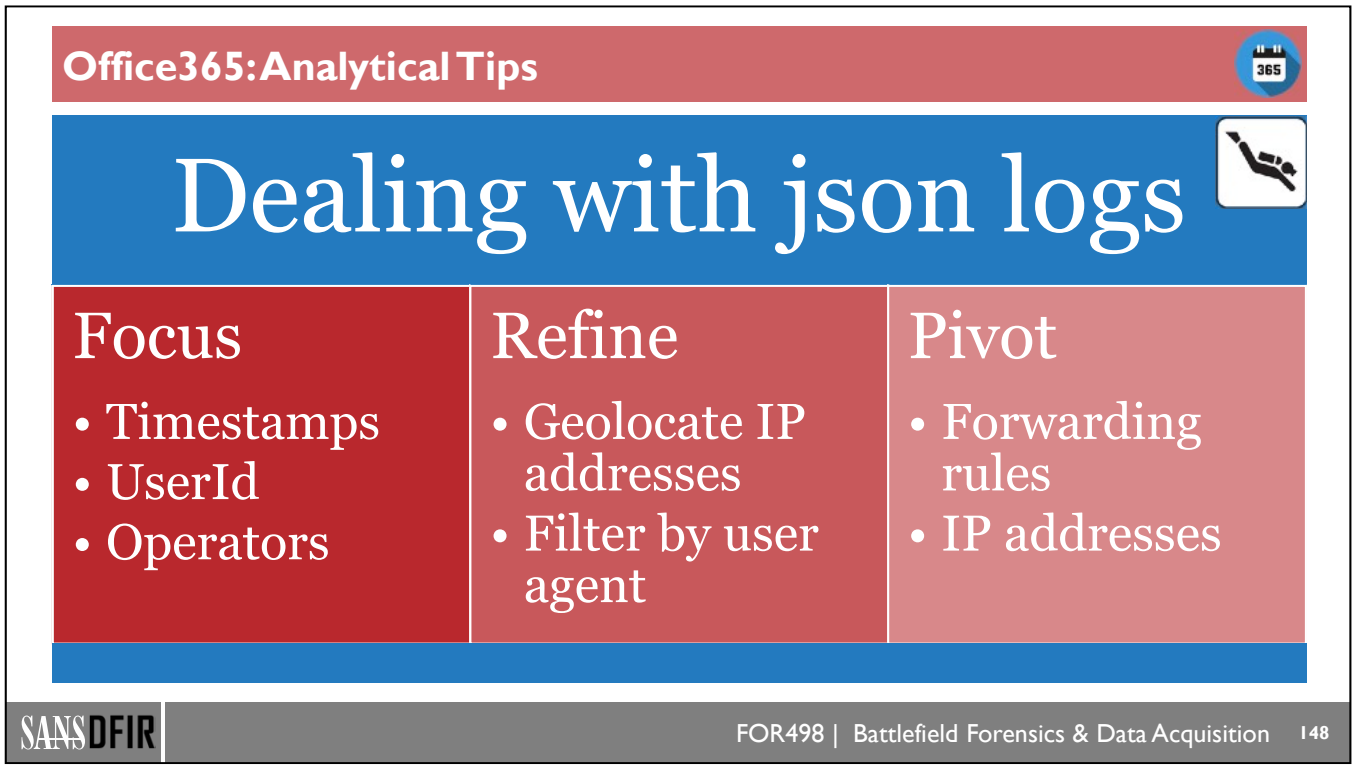
As an example of what can be done with Get-Mailbox, consider the following command that identifies users with forwarding rules in place where both delivery AND forwarding should take place:

```
Get-Mailbox | select
UserPrincipalName,ForwardingSmtpAddress,DeliverToMailboxAndForward | Export-
csv c:\temp\forwardingUsers.csv
```

Set-Mailbox can be used to enable additional logging to the UAL for user mailboxes for things that are off by default, including email moves, deletions, mailbox logins, etc.

Because of how many platforms and management tools exist for said platforms, it is useful to have a way to install all the necessary cmdlets on your system at once vs. tracking them down as you find you need them. To make this easier, Nathan Mitchell has created an installer that bundles everything into one place for PowerShell management of OneDrive, SharePoint, Outlook, and so on. [4] This single installer greatly simplifies access to the PowerShell management cmdlets you will find yourself using regularly for investigations and data collection.

[1] Manage Office 365 with Office 365 PowerShell | `https://for498.com/g5do1`
[2] Search-UnifiedAuditLog | `https://for498.com/lrt4-`
[3] Office 365 Management Activity API reference | `https://for498.com/zwgbu`
[4] PowerShell for Office 365 AppRiver | `https://for498.com/vbtpo`



The data in the Unified audit logs is, by default, in json format. While we have shown how to save this data as CSV, depending on how much data you end up with, it may make sense to work with the data in its native form vs. trying to normalize highly nested data horizontally in CSV form.

Regardless of the format of the data, there is a lot of information available. Because of this, it is helpful to focus in on the most important data, including:

- **CreationTime**: Compare events to any known intrusion windows. What activity is present after, for example, a user has clicked on a phishing email? Another great way to use timestamp data is to look for activity outside normal working hours. Why are people logging in at 3 AM EST when they do not start work until 9 AM?
- **UserId**: Which accounts show suspicious activities? You may start with one user and pivot to others based on IP addresses, rule creation, etc.
- **Operation**: Some are more important than others. Look for things like "UserLoggedIn", "PasswordLogon" and "ForeignRealm" to find forwarding rules and changes to a given mailbox. Compare these with changes the legitimate user initiated.
- **IP addresses**: Where do you normally expect legitimate users to log in from? If it is a US based company, do you see any logins from outside the United States? Look for both the "ClientIP" and "ActorIPAddress" fields.
- **User agent strings**: What does normal look like for your environment? In other words, do you expect to see Microsoft Outlook? If that is the only program legitimate users are using, look for things that aren't Outlook. Look for outliers such as phone vendors, POP or IMAP access, strange browsers, Python, etc. [1]

It is also helpful to look for UserAuthenticationMethod properties with a "ResultStatusDetail" value of "Success" as this gets you to the operations that were completed without error. It can be helpful to look at what was attempted but focusing on things that actually worked is more important.

Forwarding rules are created using a combination of operators. First, the "New-InboxRule" creates an empty rule. Next, the "Set-InboxRule" updates properties for the rule. Finally, "Set-Mailbox" commits the rule. Looking for this triplet allows you to quickly identify rogue forwarding rules (i.e. sending a copy of all email to an external Gmail account for example).

Finally, leverage the "Search-UnifiedAuditLog" cmdlet's filtering ability to reduce the amount of data you have to look at. For example, if you know a certain user has been compromised, start with that user's UAL logs. After using the techniques above to find, for example, suspicious IP addresses, pivot back into a UAL and search against just those IP addresses:

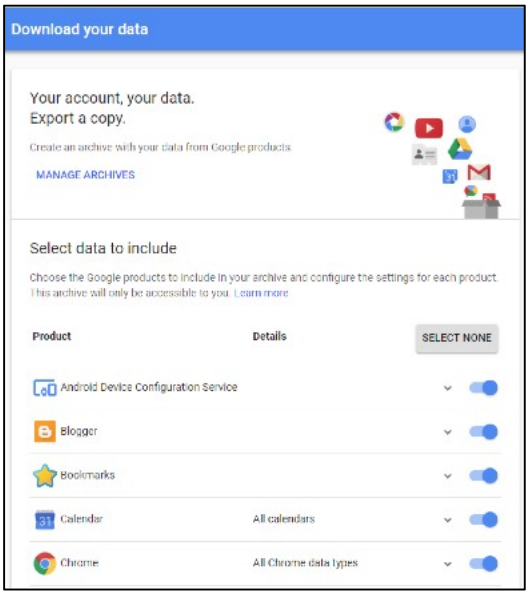
```
Search-UnifiedAuditLog –IPAddresses "1.2.3.1,1.2.3.2,1.2.3.3" –
StartDate mm/dd/yyyy –EndDate mm/dd/yyyy | Export-Csv c:\ips.csv
```

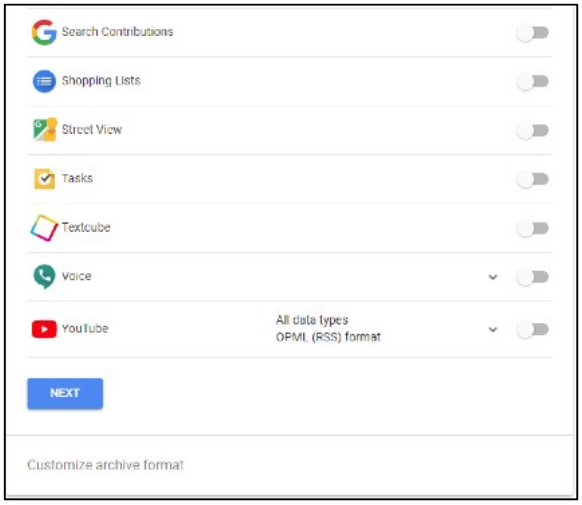
Leveraging the ability to filter on the back end makes your life a lot easier because you will have less data to deal with for the analysis phase. As your scope changes, simply repeat the search process using the new information until you have a good picture on things.

These tips will help get you started with UAL analysis, but this is by no means all you will need to do when it comes to reviewing UAL logs. There are just too many different scenarios that you may run into to have a single, linear process that addresses the needs of every investigation. Be flexible!

[1] Resource for User Agent Strings | `https://for498.com/0e3k9`

Google Takeout (1)





Google Takeout (it is also referred to as Google Takeaway in some regions) allows users of Google services to download archive files of one or more of Google's services such as Bookmarks, YouTube, Blogger, and so on. The selection of which services to archive for download is user configurable. This feature requires a user's credentials for the account (username and password) to be able to sign in to access the Takeout menu.

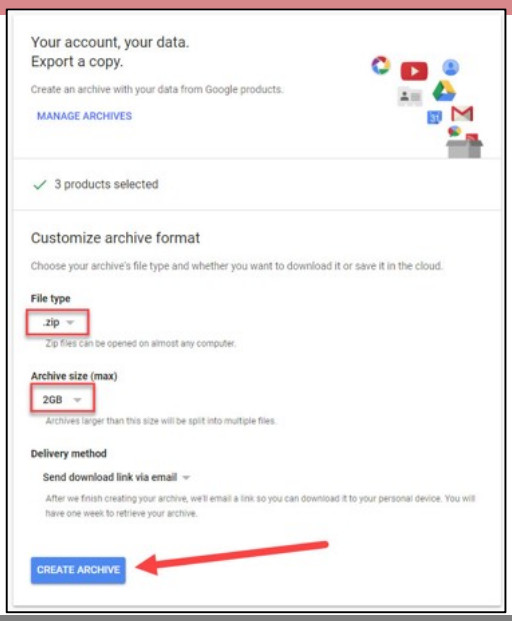
Once signed in, a wizard interface walks you through the process. Many of the services have dropdowns that allow for additional customization, such as selecting all calendars or only specific ones, and so on. When not selecting all available data, other dialogs walk you through customizing a service's sub options. In some cases however, the drop down simply states what kind of data will be provided, such as json format.

The most common things to be targeted include Gmail (with the optional ability to limit collection to certain labels), bookmarks, contacts, Chrome, Google Drive, Google Photos, and Location History. There are dozens of other options available as well, so be sure to review the available options and adjust the collection, accordingly, depending on the needs of your case. While the default includes all data from all services, if you are interested in specific answers from specific services, it will be much faster to select a few key services which will result in a smaller archive.

After selecting one or more of the products from the list, several choices are available, including the type of archive to generate (tgz or zip), as well as the maximum size of the file. By default, an email message will be sent to the account when the Takeout archive is available, but additional options exist to export the data to Google Drive, Dropbox, OneDrive, and Box.

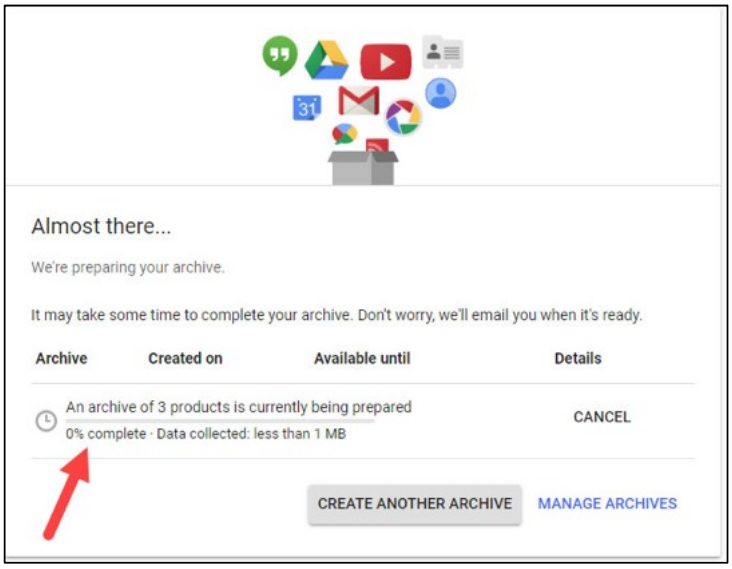
Depending on how many Google services are selected, as well as the amount of detail in each service is present, archive generation can take a long time.

Google Takeout (2)



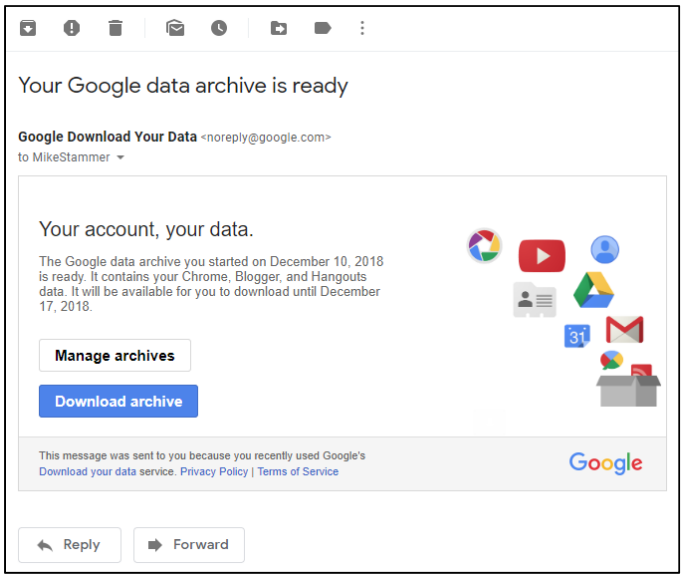
This page intentionally left blank.

Google Takeout (3)

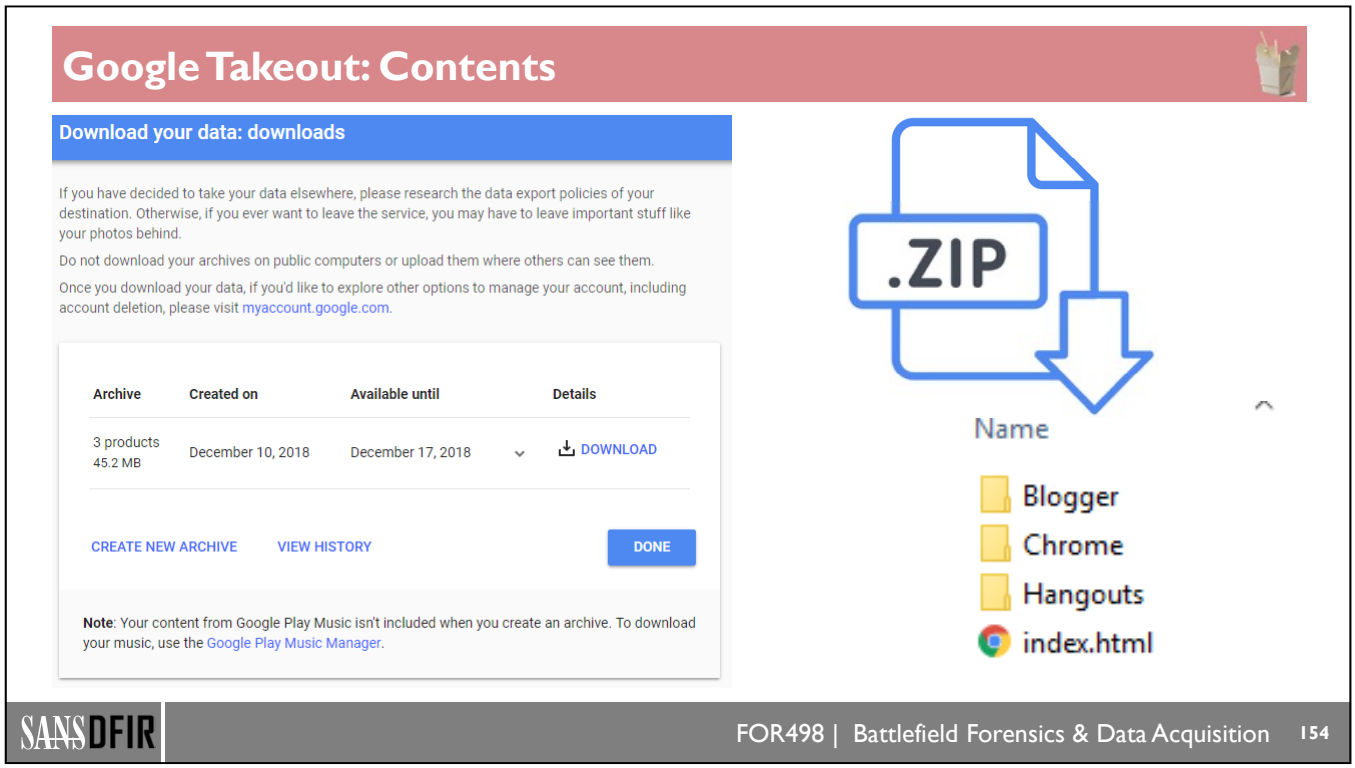


This page intentionally left blank.

Google Takeout (4)



This page intentionally left blank.

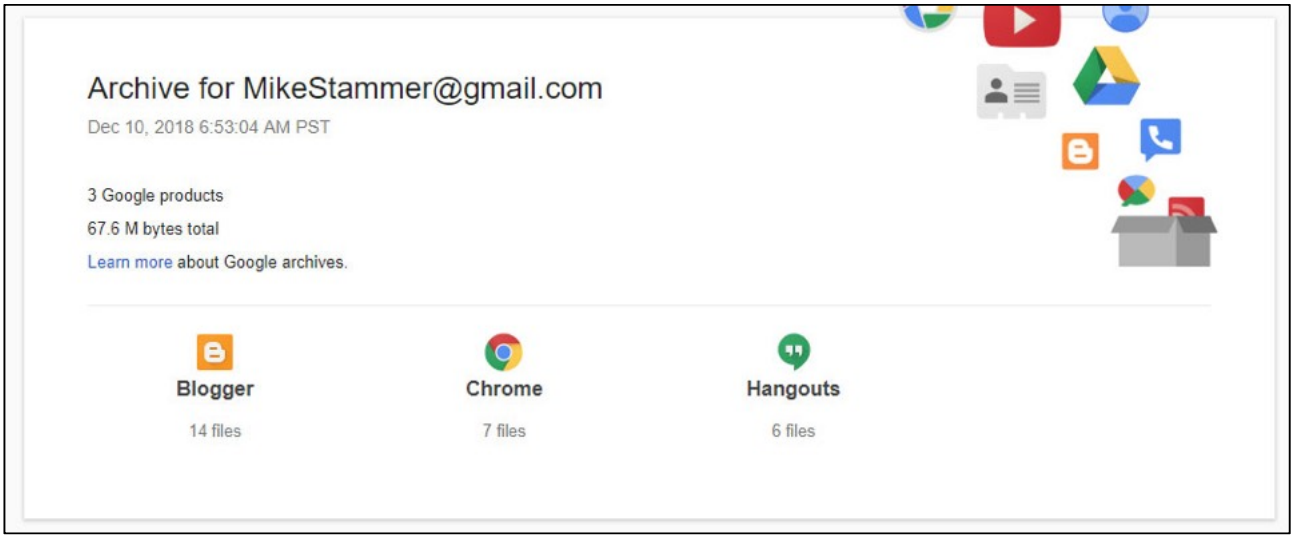


Once archive generation is complete and it is available on your local system, use your archiving tool of choice (we prefer 7-Zip) to decompress the archive. Once decompressed, each service gets a dedicated folder along with a main index that serves as an index into the data (we will see an example of this next).

Each folder contains one or more (usually more than one) files related to the service. Most will include at least a few CSV files that can be viewed in Timeline Explorer or Excel, but some of the other files, such as Chrome's Browser history, are in json format. Because of this, the timestamps related to the history is not shown using a traditional date/time format, but rather as a long series of digits which represents the time, in microseconds, from January 1, 1970. Tools such as **https://www.epochconverter.com/** can convert the numerical value to a more usable format. Since the data is in json format already, it becomes much easier to import into something like Elastic Search or SOF-ELK© which allows for mapping the numerical value to a date/time that is easier to use.

Other data formats include jpgs, mp4s, and atom, but again, what you end up with really depends on the services you include in the archive.

Google Takeout: Reviewing Data (1)



Google did a great job (compared to other companies in the tech space) in including a lot of details in a single place, such as the email address of the account, when the archive was created, the total number of products included, and the size.

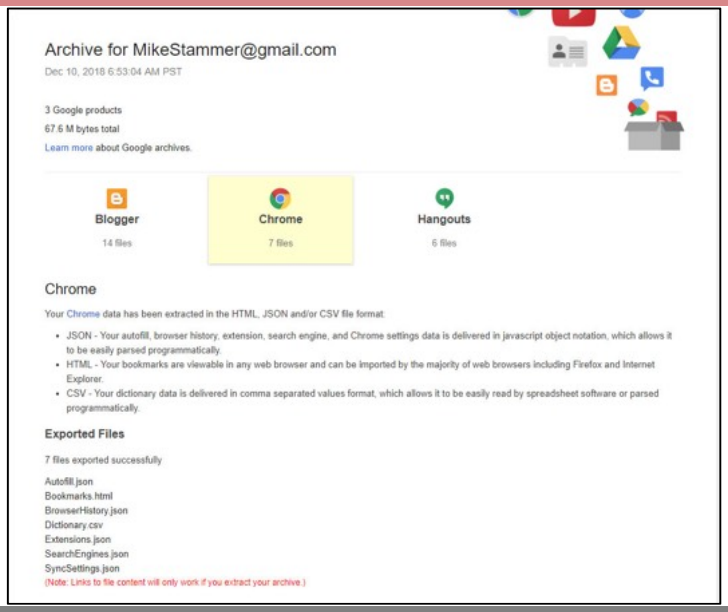
Opening the index file brings up a menu showing each of the services included in the Takeout. This makes it very easy to review the data inside the Takeout archive. It even includes a usable directory structure at the bottom of some sections to show you what files exist, all from a single interface.

In most cases, the index serves more as a way to see what exists inside the archive vs. doing any kind of serious analysis using the index. There are better tools to look at CSVs and json files. In some cases, custom tools may be needed in order to make the best use of things, especially when there is a need to cross reference or correlate data from one service to another. Since each service is essentially its own island, it is up to the analyst to build the bridges between these islands as needed.

While there are other ways to acquire some of the services from Google, Takeout makes it a one stop shop to get all the data (but not always in the best format perhaps) for a user of Google services. Earlier we saw how to collect IMAP based email using several techniques, and any of these could be used against a Gmail account as well, assuming IMAP has been enabled for the account in question. Other, more manual approaches to collecting data are also possible, such as simply browsing and making notes about a user's YouTube channel, etc.

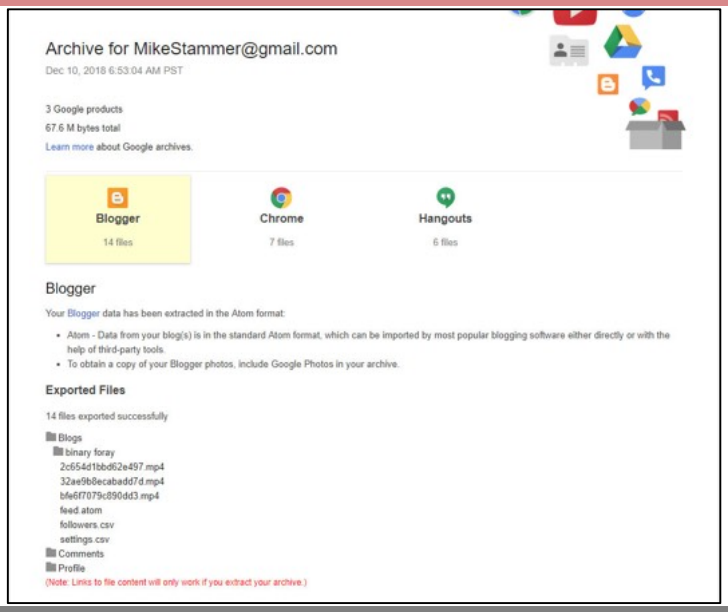
Again, remember that to use Takeout you must have a username and password for the account you want to use Takeout on. If you do not have this, the only information you will be able to access about a user is what that user has decided to make publicly available. Your only other recourse when you need more details on a Google user would be legal process, often in the form of a subpoena or search warrant. Depending on what kind of information you need about a Google user you may have to use one, or both, of these legal options to get what you need.

Google Takeout: Reviewing Data (2)



This page intentionally left blank.

Google Takeout: Reviewing Data (3)



This page intentionally left blank.

Protect Yourself!

Users often reuse passwords

If they produce their username and password for collection, could this set you up for difficulties later?

Provide a strong password you want the user to change their account password to

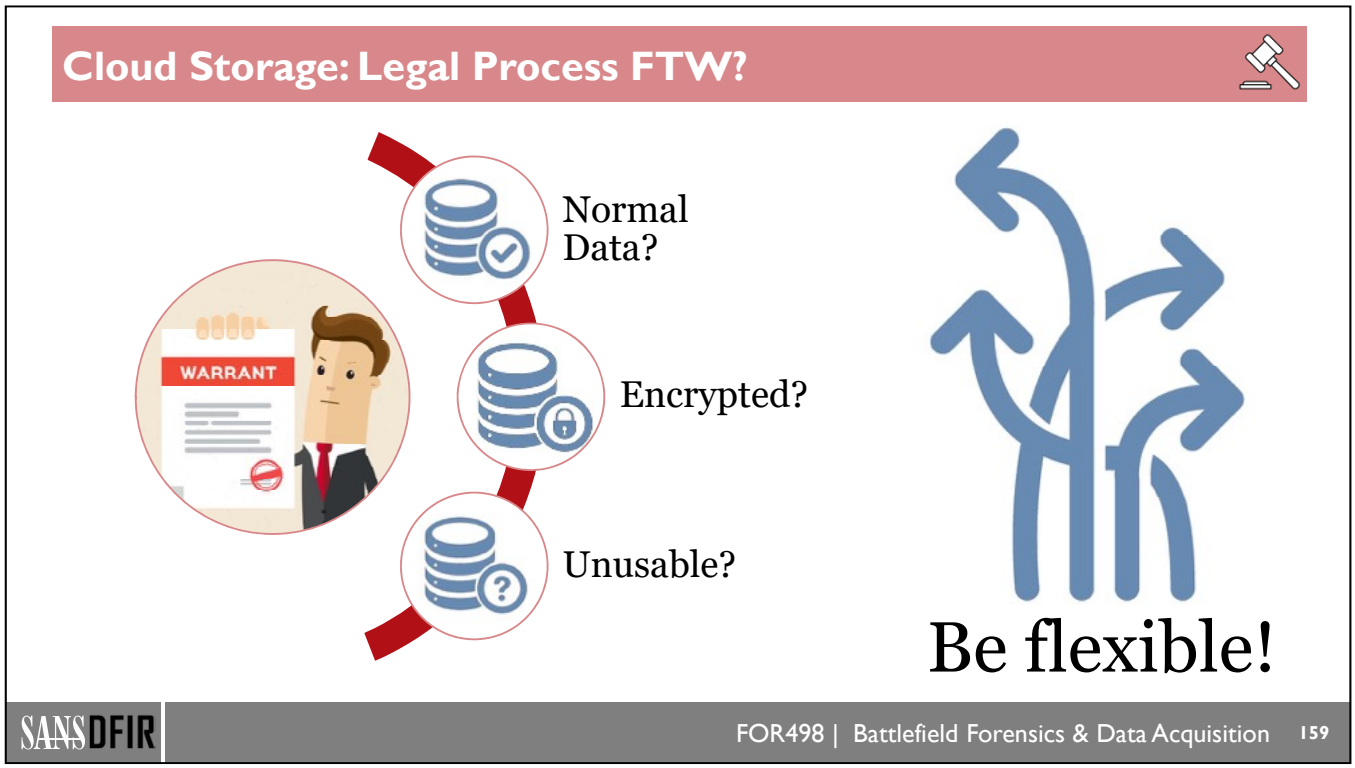
- Once the collection is complete, have them change password back

We have covered a great many options to interact with user data across a network, as well as accessing this data across the Internet. One common denominator in all of these methods of access is that you need the user's credentials.

It is quite uncommon in civil cases to obtain a subpoena to collect, for example, contents of email containers for a user who will not provide their credentials. Notwithstanding this, and in the instance that a user has provided credentials to access their data, the forensic examiner needs to protect themselves from future accusations or problems.

Although it is greatly frowned upon, the fact of the matter is that most users use the same credentials across many different platforms. This means that if a user provides their Gmail password (for example), they have probably also provided the password for their Facebook account, their Amazon account, their banking account, etc. If anything should ever go wrong down the road with any of their accounts, the accusation could be that you tampered with things, because you had their password.

As a result, the recommendation is to NOT use their password to access. We recommend that you as the examiner create a password for the exclusive purpose of accessing the necessary containers. Give this password to the user and have them change their password to the container in question, using the one you have provided. Once you are done your collection, the user can revert to the old password. This way, you never know what their password was, and you cannot be accused of using it elsewhere.



Finally, always remember that in many cases, serving the company who controls the cloud storage infrastructure you need to acquire with legal process is a means to collect the data. This would usually be the result of a search warrant when actual content is needed. But will this always be successful? In many cases, cloud providers can locate and copy everything for a given user or users as outlined in a search warrant or equivalent legal document. In ideal circumstances, you would end up with an exact copy of the data as it exists on an end user's account. But what if this isn't a possibility?

Consider the case where an end user has supplied a user-selected encryption password or key which is then used to encrypt the data before it is uploaded. In this case, even though the cloud provider successfully delivered the data they have that belongs to the account owner, you would not be able to access the data without knowing the key that was used to encrypt it as well as the means that was used to encrypt the data. The other possibility here is that the data is so spread around the globe in different countries (all with different laws) on so many different machines that the company cannot (by design?) retrieve all the data in order to facilitate providing you with a copy.

When these last two situations (or others not even imagined yet) happen, remember what was mentioned earlier. Be flexible! Is the data somewhere else that you can access, like mirrored on a hard drive you control, or backed up to another location that you can access? It may take some creativity on your part to find a way to access the data, so be flexible.

Summary

- F-Response is a great tool for reaching data not just across the network, but across the Internet
- Cloud storage involves more than we traditionally think of
- Never underestimate the power of extracting online data traditionally
- Office365 provides robust tools for the extraction of data and logs
- Google Takeout lets an examiner extract some or all of a user's repository of Google data

This page intentionally left blank.

Exercise 4.3A

Network Acquisition

Synopsis: In this exercise, you will use Cyberduck and AWS CLI to acquire an Amazon S3 bucket. You will also use Google Takeout to acquire data from one or more Google services.

Average Time: 25 Minutes

This page intentionally left blank.

Exercise 4.3A Takeaway

- Network acquisition allows for using a wide range of tools to collect data.
- Be flexible! If one tool does not work for you, pivot to another.
- Be sure to take into consideration the bandwidth available to you when performing acquisition over the network. If you are on a slow link, consider collecting only the most critical data you need first.
- For collections that you may do more than once, consider using a command line interface program which lets you automate via a script. This can save considerable time (as well as prevent mistakes!) over other access methods.

This page intentionally left blank.

Exercise 4.3B

Network Acquisition

OPTIONAL, OUT OF CLASS EXERCISE

Synopsis: In this exercise, you will use Thunderbird portable to collect IMAP data. You will also use F-Response to target and collect network-based assets.

This page intentionally left blank.