THE JOURNAL OF DIGITAL FORENSICS, SECURITY AND LAW

Journal of Digital Forensics, Security and Law

Volume 14 | Number 4 Article 4

April 2020

Teaching Data Carving Using The Real World Problem of Text Message Extraction From Unstructured Mobile Device Data Dumps

Gary D. Cantrell
Southern Utah University, cantrellg@gmail.com

Joan Runs Through
runsthrough@dixie.edu

Recommended Citation

Cantrell, Gary D. and Runs Through, Joan (2020) "Teaching Data Carving Using The Real World Problem of Text Message Extraction From Unstructured Mobile Device Data Dumps," *Journal of Digital Forensics, Security and Law*: Vol. 14 : No. 4 , Article 4.
Available at: https://commons.erau.edu/jdfsl/vol14/iss4/4

TEACHING DATA CARVING USING THE REAL-WORLD PROBLEM OF TEXT MESSAGE EXTRACTION FROM UNSTRUCTURED MOBILE DEVICE DATA DUMPS

Dr. Gary Cantrell[1] and Joan Runs Through[2], ED.S., M.S.
[1]Southern Utah University
Computer Science and Information System
cantrellg@gmail.com
[2]Dixie State University
Digital Forensics Crime Lab
runsthrough@dixie.edu

ABSTRACT

Data carving is a technique used in data recovery to isolate and extract files based on file content without any file system guidance. It is an important part of data recovery and digital forensics. However, it is also useful in teaching computer science students about file structure and the binary encoding of information, especially within a digital forensics program. This work demonstrates how the authors teach data carving using a real-world problem they encounter in digital forensics evidence processing involving the extracting of text messages from unstructured small device binary extractions. The authors have used this problem for instruction in digital forensics courses and other computer science courses.

Keywords: Mobile Forensics, Small Device Forensics, Mobile Triage, Digital Triage, Data Recovery, Data Carving, Binary Files.

1. INTRODUCTION

Data carving is the search and extraction of lost files based on internal file structure or content instead of external file system metadata. This process is often necessary when the file system has been damaged, corrupted, or reformatted. It is also essential to data recovery when the extraction of digital memory leaves the technician with a binary image that does not have a file system to assist in isolating individual files within that image. Data carving is an important topic in a digital forensics course or any technology course that discusses data recovery. It can also be used in any computer science course to facilitate understanding of file structure and binary file IO. This manuscript presents a method used by the authors to teach data carving to undergraduate and law enforcement students using a real-world problem encountered in a digital forensics laboratory.

The foundation of the problem introduced to the students is that of mobile triage and text message extraction. Short Message Service files, often referred to as SMS or text messages, do not have a standardized file format or an easy to predict structure; therefore, automated tool extraction is limited. Using this real-world problem as a basis for instruction demonstrates that learning how a technique works is more important than learning about tools that perform a technique with little user direction.

The remainder of this manuscript will present the information in the same order used to instruct undergraduate students. It is presented as a real-world problem and solution as commonly encountered by the authors. Presenting this information in this manner helps undergraduates not only understand data carving but comprehend how it could be used in real-world situations.

2. SMALL DEVICE TRIAGE PROCESS

In addition to digital forensic education, the authors of this work specialize in the forensic analysis of mobile devices for law enforcement agencies, including the direct extraction of the data on the NAND memory chip often called chip off forensics. The end result of the chip off forensics process, when performed on devices programmed with proprietary or unsupported operating systems, is an unstructured physical memory dump. The process presented in the classroom is the method used by the authors in the recovery and presentation of SMS and MMS text messages from these unstructured physical memory dumps. This process involves:

- Isolating potential messages.
- Filtering out redundant and meaningless data.
- Applying client feedback to identify and verify message threads of importance.

This method is applicable when no file system support or SMS/MMS format is available. In other words, this activity demonstrates how data carving works without the blind aid of automated tools. This simple model demonstrates data carving at a base level and can be adapted to any type of electronically stored information stored in an unsupported file system or file format. In the digital forensics arena, this process would be classified as a triage process due to the limitations inherent in data carving. The resulting SMS messages have no file system assigned time stamp, no information is available about their storage structure, and this technique does not reverse engineer the file system in any way. Its primary purpose is for quick extraction and analysis without file system support. Digital triage techniques are those techniques applied to live or dead systems either on-scene for quick intelligence or in-house for evidence evaluation (Cantrell et al. 2012). Digital triage concentrates on finding the most useful information in the least amount of time. Mobile device triage is often discussed as a need during search and seizure (Richard III et al. 2005; Walls et al. 2011). However, the techniques discussed in this work would not be useful for on-scene analysis. Therefore, the authors describe this method as a digital triage technique for in-house evaluation. Since it is a digital triage technique, there are several things that should be clearly explained to students. First, the resulting information should be used cautiously, if at all, in full court proceedings. It does not always produce timestamps and ownership information. Secondly, there is no reverse engineering of the file formats or file system attempted. The information is gathered by isolating potential messages without considering any file or file system format. Thirdly, an additional evaluation would likely be necessary if the extracted information is needed for use in court proceedings to confirm the validity of the extracted information. Finally, this method, although not practical in the field, is intended to gather as much information as possible in the least amount of time for in-house evaluation. For class purposes, pre-owned phones can be purchased from "the wild." For example, two websites for purchasing used phones are www.ebay.com or www.shopgoodwill.com. Phone image extractions were made with software/hardware forensic tools already in house for evidence examination or instructional purposes. Images were scanned for adult content to prevent exposure of such material in the classroom. However, a classroom disclosure is highly recommended as initial scans can fail, and it is nearly impossible to prevent adult language exposure from phones collected this way.

3. TEXT MESSAGE EXTRACTION, FILTERING AND PRESENTATION

Since this technique is utilizing data carving, it is assumed that we are starting with a binary extraction without a file system. There will be no individual files to be examined. Thus, the first step in this process is to isolate possible text messages from within the physical image. The goal is to create a file per text message. This will allow for easier filtering and report generation later. The standard data carving method is to identify where a file was once stored by searching for the first few bytes of known standard file formats as a header and then to extract bytes until a footer is found, a predetermined length is reached, or the data no longer appears valid. Individual text messages, by their nature, are predictably short when compared to other communications. Instead of setting a predefined footer, setting a 200 - 400 byte limit is generally sufficient to extract the entire message. Unfortunately, most text messages do not have a standardized header. Often, the found text messages are simply proprietary database fragments in unallocated space. Therefore, it is necessary to manually determine a binary marker to use for carving. This can be accomplished using any raw file viewer that allows for searching or the Linux command line tools such as xxd and grep. These tools allow a technician or student to view the file as the file is stored on the disk independent of the file type. These tools display the file as hexadecimal values and typically provide a window for showing the interpretation of those values as text characters. Text message isolation is performed by carving out each individual message based on a discovered binary marker. This can result in a significant number of files, including partial and duplicate messages. However, with the use of some simple tools the resulting files can be easy to organize and filter for useful information. For more advanced computer science students, this carving application can be generated by the student as an assignment involving binary file IO. For more introductory level courses or more vocational law enforcement training where students have not yet mastered a programing language, any existing data carver that will allow user defined headers and footers can be used. The open source file carver Scalpel is a useful example (Richard III et al. 2005). Its included configuration file contains many standard file formats from which users can use for file carving. It also allows the user to add carvers of their own design by editing the configuration file. Newer versions also allow for the use of regular expressions instead of set values for headers and footers. Regular expressions can be especially useful with text messages as they allow for the use of a phone number as a marker for data carving either as a header or footer depending on the cell phone make/model. Step one in this model is to identify a binary marker common to all text messages located within the binary dump in question. This marker will likely be unique to the make/model of the particular device from which the binary dump was extracted. The subject binary dump, or raw image, can be loaded up in any hex editor that will handle large files. One such example would be the Windows tool FTK Imager by AccessData. This tool is provided free and is useful for this purpose. In the Linux environment, the basic commands xxd and grep suffice for image searching. For classroom instruction, the authors use both a graphical tool and Linux command line depending on preference and level of technical expertise of the class.

If the text of the SMS/MMS messages are not in standard ASCII, then an additional effort is required, and the process changes from triage to analysis, and this model is no longer applicable. For example, some phones use the PDU format or 7-bit GSM format for storing text messages. This involves a 7-bit shifting alphabet that is not interpreted by most hex editors, and thus it will not be searchable (Henry-Labordere 2004). However, most of the phones examined by the authors store text messages in plain ASCII. Identifying text messages within the unstructured data dump can be done with simple word searches. Knowing the phone details is useful for constructing word search lists. However, common words and acronyms are often sufficient. Some examples include: lol, lmao, dude, hi, love you, etc. It is also often useful to search for common expletive words. The longer and more unique the word, the less chance for false positives, but this lowers the chance of finding any hits at all. At this step the user is trying to locate a sample set of text messages to analyze as opposed to all text messages on the raw image.

Once several text message examples have been identified, it is useful to print a hard copy of some of the results including both hexadecimal and ASCII and to examine this hard copy with highlighters in hand in an attempt to identify markers for carving. The authors have had consistent success with this process as an in-class activity. Once the marker has been identified in the hard copy, students are led through an activity using the Linux commands xxd, grep, and wc to get an estimate of how often the marker appears.

xxd "image name" | grep "keyword" | wc

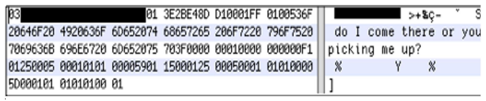
If keyword searches fail, it is necessary to escalate and determine if no ASCII or Unicode messages exist. A phone should not be dismissed from this method until it has been tested in this way. A program that filters binary images for text is useful in this determination. For example, the command line program Strings effectively pulls ASCII/Unicode from a binary dump. Depending on instructor preference, the analysis can be done from a Linux prompt or open source versions of Strings are available for Windows. The command filters through a binary image and extracts all consecutive text characters of a predefined length. The default length is typically 4 characters, but the user should confirm with the program documentation. For example, the following command will create an output file containing all characters that occur in groups of 4 or more: *strings "image file" > out.txt* Commands can vary depending on the version of Strings being used. With the phones tested four consecutive characters typically results in a very large output file. The consecutive character length can be adjusted by the user during execution to reduce the output. For example: *string -n 10 "image file" > out.txt*

will return all characters that occur in groups of 10 or more and store them in out.txt. Setting this value too low will result in more false positives, and setting this value too high might cause it to miss some information of interest. Utilizing the flag -o can also be helpful as this causes Strings to output the raw image offset along with the text string allowing the user to more easily identify that memory location.

Using Strings is a very useful method for extracting information and deciding if there is information worth parsing, but the output can be formidable to work through. Using Strings for text message extraction or searching is a last-ditch effort. In this process, it is used to confirm the absence of messages. Absence, in this case, could be due to encryption, non-ASCII/non-UNICODE message format, or simply lack of text messages present. This can also be performed prior to searching for messages as a check to determine if the image contains useful information.

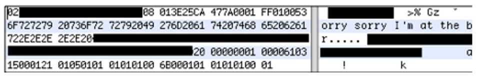
Assuming that the keyword search does not fail, the student is able to identify markers for carving by examining the areas above and below the extracted sample as the machine language/file format coding used by the file system to store these files leave residual and identical hex patterns prior to each SMS message. These markers can then be used to code a solution or used within Scalpel or any other customizable carver to extract each message into a separate file.

Image 1 and Image 2 are screenshots of individual files carved in this manner from a chip off extraction performed on a ZTE x501 Cricket phone. (Personal information has been redacted in these images for legal purposes.) This is a phone model commonly used by the authors in class. This technique is especially useful on feature phones marketed by pay-as-you-go providers such as the ZTE x501. Phones of this type are known as disposable or burner phones. They often do not have working data ports, and direct binary extraction from the NAND memory chip (chip off forensics) is the only option to retrieve a binary image, which often results in an image that has to be carved for data.



```
03[redacted]01 3E2BE48D D10001FF 0100536F   [redacted] >+%ç-  ˇ  S
20646F20 4920636F 6D652074 68657265 206F7220 796F7520    do I come there or you
7069636B 696E6720 6D652075 703F0000 00010000 000000F1   picking me up?
01250005 00010101 00005901 15000125 00050001 01010000    %       Y    %
5D000101 01010100 01                                    ]
```

Figure 1. Hex editor view of ZTE x501 carved text message



```
02[redacted]08 013E25CA 477A0001 FF010053   [redacted]  >% Gz  ˇ
6F727279 20736F72 72792049 276D2061 74207468 65206261   orry sorry I'm at the b
722E2E2E 2E2E20[redacted]                               r..... [redacted]
[redacted]20 00000001 00006103                          [redacted]          a
15000121 01050101 01010100 6B000101 01010100 01           !       k
```

Figure 2. Hex editor view of ZTE x501 carved text message)

In the case of this example, no header was identified, but as is common, each text message includes a phone number, and in this case, the phone number occurs at the top of the message. This was entered into the Scalpel configuration file as:

SMS y 400 /[ˆ0-9][2-9][0-9] [0-9]-[2-9][0-9][0-9]-[0-9][0- 9][0-9][0-9][ˆ0-9]/

This line tells Scalpel to carve using a U.S. format phone number, represented by a regular expression, as a header, and to carve out 400 bytes at a time. No footer was used in this example. However, the reverse can also be performed using the phone number as a footer and a hex pattern as a header in cases where the phone number comes after the message. For this example, students recover approximately 400 individual hits. The majority (85%) of these are false hits and duplicates. Having a vast majority of recovered messages being not useful is not uncommon in real practice. Depending on the class, some simple filtering can be explored that greatly reduces the results. If students code the application themselves, this problem can be introduced as a next step. If the students have not mastered a programming language yet, a simple script can be demonstrated that

eliminates all duplicates, and all files that did not contain genuine messages. In this example, the introduced script reduces the results down to 60 individual files. Further manual examination brings the amount down to 30 unique messages. It is important to incorporate scripting and intelligent searching at this phase and teach students to work smarter, not harder. Prior to the development of physical processes such as chip off extraction and JTAG, text messages were often recorded by investigators who took pictures of the phone screen. A report compiled using this method on the same phone prior to memory extraction is shown to the class to demonstrate that not all of these messages were available through the cell phone interface, indicating some of the messages recovered were deleted messages. This adds value to the method demonstrated to the students and emphasizes that it is useful even if it is not the only option available. As previously mentioned, phones used in class are purchased used from various sources such as eBay and Goodwill. The phone models can be chosen based on the applicability of this technique. For institutes that do not have access to chip off equipment, there are software methods for obtaining images that do not require the direct data extraction from the NAND chips. The steps introduced as a triage type process and are especially useful when the primary goal is to determine if the phone is worth additional effort. It is important to emphasize this is triage, not forensic analysis. No file system interpretation is performed and certain assumptions are made about the data.
The steps introduced as a triage type process and are especially useful when the primary goal is to determine if the phone is worth additional effort. It is important to emphasize this is triage, not forensic analysis. No file system interpretation is performed and certain assumptions are made about the data.

1. Searching for keywords to identify message examples within the raw file (if no keywords are found Strings can be used to verify the absence of messages).

2. Identifying markers that can be used as headers and/or footers.

3. Carving out individual messages using these markers with an automated carving tool or student developed application.

4. Filtering the resulting files based for duplicates and non-interesting output.

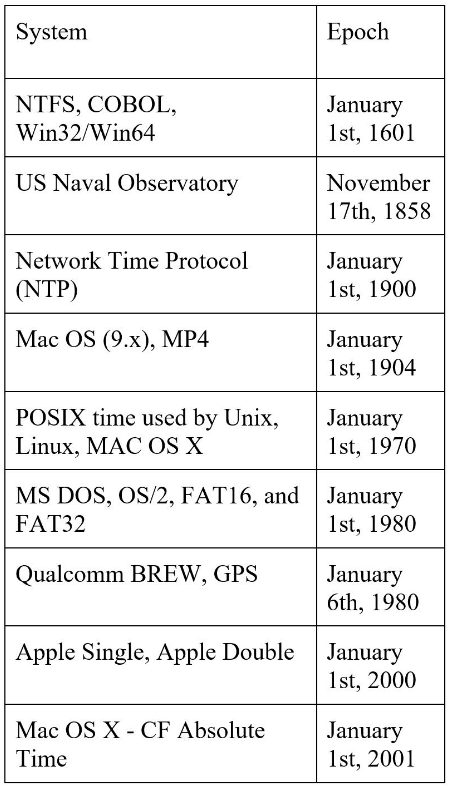
The recovered individual files can easily be combined into a basic report using Strings to extract all text content, echo to write a line of asterixis between each record, and redirect to combine the outcome into a text document. The development of this command is part of the in-class work of the students.

In a classroom setting, the authors introduce a short script to combine all files into a single output file with each line representing the text of a single file. This results in a very basic report containing one message per line. In an effort to generate a more useful report, it is advisable to also recover the time the message was sent/received, if possible. The following section will discuss this important additional step.

4. TIME STAMP IDENTIFICATION

Section 3 discusses a method for basic data carving. Classroom instruction could stop prior to time stamp identification if demonstrating the mechanics of data carving is the only goal. Instruction could also logically move from the recovery of ASCII to the extraction of non-text-based files and the more advanced techniques used by automated tools. However, if this process is being introduced as part of a digital forensics course or the instructor wants to complete the real-world scenario, the next logical step would be to extract additional information to chronologically order and corroborate the messages with the case. This means finding and interpreting the message timestamp if present. This section will discuss that process. Timestamp information, although important adds a more complicated level of analysis. Timestamps are typically stored as the number of time intervals from a specific date commonly referred to as an epoch. For example, Unix time, also called POSIX time, is the number of seconds since January 1st, 1970. Windows time is stored as the number of 100 nanosecond intervals since January 1st, 1601. Mobile devices, which often utilize their own proprietary file systems, use the epoch or time interval of their choosing. For example, the Brew file system (an open source pseudo operating system) utilizes a time epoch of January 6, 1980. Other systems, such as many Samsung devices, use a FAT file system derivative and fall back on a January 1, 1980 time epoch (Henry-Labordere 2004). This can complicate hex value interpretations. Table 1 lists some sample time epochs for known systems to demonstrate this obstacle. The authors often use the same method for locating timestamps as they do for identifying binary markers. Print out hard copies of the found samples and use highlighters to examine the printouts. The key is observing changing patterns. Logically the timestamp should occur in at least close to the same binary offset within each message and should change only slightly between text messages.

The timestamp information can be stored in little-endian-order or big-endian order. This means the hexadecimal values observed in the editor will be read left to right or right to left, depending on how the file system stores the timestamp. This is a common complication in file analysis when a file map is not available. This ordering is most commonly at the byte level, which is represented by two hexadecimal values per byte in hexadecimal editors (although it can be in reverse nibble which would reverse the entire hexadecimal sequence). For example, the date October 31st, 1996, 11:00:00 AM would be stored as 846759600 seconds in POSIX time. In a hex editor, it would look like 0x327886B0 in big endian and 0xB0867832 in little endian. Note the pattern is reversed in sequences of two. Once the ordering is understood, the conversion can be done within a variety of hex editors. Many open source tools and Web pages will also perform these conversion tasks easily. It is also an interesting problem to provide as an assignment for more advanced students.

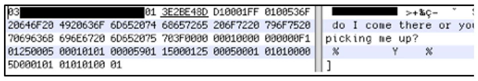


System	Epoch
NTFS, COBOL, Win32/Win64	January 1st, 1601
US Naval Observatory	November 17th, 1858
Network Time Protocol (NTP)	January 1st, 1900
Mac OS (9.x), MP4	January 1st, 1904
POSIX time used by Unix, Linux, MAC OS X	January 1st, 1970
MS DOS, OS/2, FAT16, and FAT32	January 1st, 1980
Qualcomm BREW, GPS	January 6th, 1980
Apple Single, Apple Double	January 1st, 2000
Mac OS X - CF Absolute Time	January 1st, 2001

Figure 3. Timestamp epoch samples with associated systems

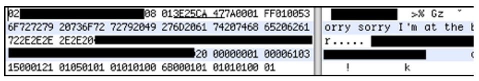
The pattern to look for within a hex editor is a number as described that does not change much from message to message. The length will vary depending on the time unit used, but seconds stored in 4 bytes is very common. Time intervals between text messages as conversations will not span long periods of time. The previous example was based in the year 1996. 1996 in POSIX time would start with 820454400 seconds and end with 852076799 seconds. These are hex strings 0x30E72400 and 0x32C9A8FF respectively. Within the entire year, the highest order hex values go from 0x30 to 0x32. The month of October would be 844128000 seconds through 846806399 seconds. These hex strings would be 0x32505F00 and 0x32793D7F respectively. In this case, the highest order byte is the same for the entire month 0x32. The closer the text messages are in time the more higher order values will be the same. Searching for and identifying timestamps is a great in class exercise and demonstrates concepts within file structure and time stamp storage.

As part of the real-world scenario, it should be explained that unless documentation can be found and cited that describes how timestamps are stored on the system under examination, then there should always be a disclaimer included with the reported results. This disclaimer should explain that the timestamps displayed are only an estimate and need to be confirmed, or the method used to confirm their accuracy should be presented. As this is used as a triage type method, this is an acceptable compromise as long as the requesting agent understands the limitations of timestamp provided. Composing such a disclaimer can and should be included as part of the instruction. For example, Image 3 and Image 4 are the same as Image 1 and Image 2, respectively, with the perceived timestamps underlined. Image 3's timestamp is 0x3E2BE48D, which translates to a date of Mon Jan 20 04:59:09 2003 in POSIX time, and image 4's timestamp is 0x3E25CA47 which translates to date of Wed Jan 15 13:53:27 2003 in POSIX time. (It should also be noted that the timestamps do not occur at exactly the same offset causing difficulty with automated extraction.)



```
03[redacted]01 3E2BE48D D10001FF 0100536F  | [redacted] >+%ç-
20646F20 4920636F 6D652074 68657265 206F7220 796F7520 |  do I come there or you
70696368 696E6720 6D652075 703F0000 00010000 000000F1 | picking me up?
01250005 00010101 00005901 15000125 00050001 01010000 |  %        Y    %
5D000101 01010100 01                                  | ]
```

Figure 4. Hex editor view of ZTE x501 carved text message with timestamp underlined



```
02[redacted]08 013E25CA 477A0001 FF010053 | [redacted]  >% Gz
6F727279 20736F72 72792049 276D2061 74207468 65206261 | orry sorry I'm at the b
722E2E2E 2E2E20[redacted]                             | r..... [redacted]
[redacted]20 00000001 00006103                        | [redacted]
15000121 01050101 01010100 68000101 01010100 01       |    !        k
```

Figure 5. Hex editor view of ZTE x501 carved text message with timestamp underlined

Once found, these dates can be corroborated with other messages. In the absence of file system corroborative information, the following three questions should be asked by the examiner/student:

1. Could the timestamps be a product of random chance, or do the timestamps occur at the same or similar offsets?
2. Do ordering the messages chronologically by this timestamp appear to produce a logical conversation?
3. Do the timestamps found provide reasonable dates based on the make/model of the phone and the case details?

In this example question 3 fails. The year 2003 is not reasonable based on the case information and phone type. The make and model of the device examined (ZTE X501) was not approved for use by the FCC until February 24, 2012, and therefore was not in use in the year 2003. However, questions 1 and 2 indicate that the time interval is reasonably correct; therefore, the epoch is the likely problem. As stated earlier, the Brew file system, commonly used on many CDMA feature phones, uses seconds as time intervals, but the epoch is 10 years and 5 days after the Unix epoch, which was the epoch originally used in this example. Advancing the timestamp in this example by the Brew offset moves the dates to a reasonable range in the year 2013.

Even if not exact, any timestamps recovered can be useful. They can be used for the chronological ordering of messages, and if the phone is still in working order, comparing the timestamps of a few raw text messages to their counterparts as viewed on the live device can reveal the correct adjustment necessary to make the recovered timestamps accurate. Once the epoch offset is obtained, time stamps can be more accurately reported taking care that the adjustments consider leap years and other caveats. This activity emphasizes that timestamps should be used with great care when presenting the results to the requesting agent. When presented with data obtained through this triage model, the requesting agent must be made to understand that the provided timestamps must be confirmed prior to use in court. The importance of exact timestamp values is dependent on the case under evaluation, but it is vital these timestamps are not referred to as the correct time unless it has been validated, and even then, they should be reported as computed timestamps, not actual timestamps.

5. PROCESS AUTOMATION

The lengthier part of this method is the examination of the individual files created during the carving process. The example presented produced 500 individual files that were eventually reduced to 30 actual messages. This is not necessarily typical. The number of individual messages can range from a few hundred to thousands. The quantity of messages found on a device is dependent on the size of the memory on the phone, the operating system installed on the device, the tendency of the user to write and receive SMS, and the markers used for carving the data. Extracting corrupted files and duplicates is a common problem with any type of data carving and should be discussed with students. Recovering the data is only part of the solution. Practitioners, whether doing data recovery or digital forensics, need to consider the client's needs. Presenting the client with an overwhelming number of files to sort through is not good practice, and providing a simple report with just text messages and no timestamps may not be sufficient.

While processes such as finding the markers, carving out the data, and extracting the timestamp can be performed manually, time constraints hold that it is better if the filtering, timestamp extraction, and reporting are automated. The authors typically demonstrate this automation with the use of Perl scripts. Perl was chosen for its strong regular expression ability that can facilitate the filtering of messages (Christiansen et al. 2012). However, any scripting or programming language with which the instructor is familiar should suffice. Published examples exist of similar uses for scripting during digital forensics (Garfinkel 2009; Cantrell et al. 2013). The authors provide sample scripts and teach this process during academic, collegiate, and law enforcement training. Including how to create these automation scripts here is beyond the scope of this work.

How much scripting to include in the classroom environment is dependent on the technical level of the class. Scripts can be introduced as complete with the instructions on how to run the script, or scripts can be provided with the instruction that they need to be adapted or edited to fit the subject binary dump, or scripts can be created from scratch by the students.

During script execution, each file is accessed; a check is run to see if it has been encountered previously; some simple tests are run to see if it is a valid message. The output is then filtered for ASCII characters, and divided into at least phone number and message, but can also include incoming/outgoing, timestamp, and to/from columns. This output is then sent to a comma separated value file that can be further manually edited for readability with a spreadsheet program.

Optionally timestamps are extracted and converted to human readable dates. This is a simple matter if they occur in predictable offsets, but as shown in the previous example, this is not always the case. As also discussed, the higher order values should fall within a predictable range once identified. For example, as already described, if one was looking for timestamps within the year 1996, the script could be written to scan for the hex values 0x30, 0x31, and 0x32. This can alert the program that the following, or the previous, depending on ordering, 3 bytes can be combined with this byte to form a plausible date. In the case of reverse nibble, the search would simply be reversed to 0X03, 0x13, and 0x23.

The narrower the user can make the search, the less chance of a false positive. However, being such small files, the probability of this random occurrence is reduced, leading to few or no false positives. Of the phones evaluated by the authors at present, few have produced false positives due to random chance when searching in this manner.

Automating the process is introduced as adaptive scripting, not as complete software development. In the experience of the authors, it is more useful to have multiple adjustable scripts than to try and create a single robust application. The variety of phones an examiner will see in the lab, combined with constantly changing technology, makes it difficult for one application to suffice.

6. AGENT FEEDBACK

This process has been described numerous times in this manuscript as a triage method. As such, it is presented to the client as an informative report only. It is vital that students include disclaimers and present the information with this made clear. Part of the process is to present the information to the client. If the client wishes to use the information as court admissible evidence, further analysis can be done. In the lab, the authors typically request that the client mark any messages of importance, and further analysis can be done if needed. For example, further analysis could be the reverse engineering of the phone by purchasing a like phone, planting messages at known times, and extracting them for analysis.

It is the experience of the authors that this is seldom the case. Even if the triage report has limits as court admissible evidence, it still serves as good intelligence. Often the client will not come back for further analysis either because they learn the phone text messages do not contain useful evidence or the information suffices as intelligence only. In either case, the triage report often suffices freeing up valuable lab time for the examination of other devices.

7. DIGITAL TRIAGE METHOD SUMMARY

As a small device digital triage method, the entire process presented is as follows:

1. Text messages searched for using keywords.

2. Text message are evaluated for markers.

3. Text messages are isolated into individual files using custom carvers.

4. Individual files are filtered for duplicates and viable content.

5. Timestamp evaluation is performed.

6. Report is compiled.

7. Report is presented to the requesting agent for feedback.

8. Further evaluation is done if requested.

As a classroom exercise steps 7 and 8 will likely not occur, but as a real-world problem it should be made clear these steps are necessary. Triage does not result in admissible evidence, but the information presented can be used to determine if further analysis is warranted; it can be used in establishing a plea-bargain; it can be used for quick intelligence in emergency situations.

8. CONCLUSION AND REFLECTION

Data carving is the extraction of useful information within unstructured binary data. It is commonly used for data recovery. This manuscript presented a real-world problem used by the authors to teach how data carving works. At several points, the authors tried to present the options as to how an educator might take the problem further by requiring students to develop their own full application or simple scripts. Coding details were left out of this manuscript to prevent students from using this paper as a solution guide.

The authors have used this real-world scenario to teach data carving in both digital forensics based courses and more traditional computer science courses. They have observed that the use of a real-world problem increases student interest and participation. In fact, the authors have been able to apply student innovation to their digital forensics practice and develop more elegant and efficient solutions for actual case work.

It is the hope of the authors of this manuscript that this teaching method will be of use to other digital forensics and computer science educator.

REFERENCES

[1] Australia, School of Computer and Information Science. Mawson Lakes: University of South Australia.

[2] Mislan, R.P., Casey E., and Kessler, G.C. (2010), "The growing need for on-scene triage of mobile devices," Digital Investigation, vol. 6, no. 3-4, 2010, pp. 112 – 124.

[3] Richard III, G. and Roussev, V, (2005), "Scalpel: A frugal, high performance file carver," Digital Forensics Research Workshop, New Orleans, LA.

[4] Walls, R., Levine, B, and Learned-Miller, G. (2011), "Forensic triage for mobile with DEC0DE" USENIX Symposium (2011). Available at: works.bepress.com/erik_learned_miller/52

[5] Zimmermann, C., Spreitzenbarth, and M, Schmitt, S., (2011), Reverse Engineering of the Android File System (YAFFS2). Technical Report CS-2011-06, Friedrich-Alexander-University of Erlangen-Nuremberg.

[6] Breeuwsma, M., de Johngh, M., Klaver, C., van der Knijff, R., Roeloffs, M. (2007). Forensic Data Recovery from Flash Memory. Small Scale Digital Device Forensics Journal, 1 (1), 1-17.

[7] Cantrell, G. and Dampier, D. (2013), "Implementing the automated phases of the partially-automated digital triage process model", Journal of Digital Forensics, Security and Law, Vol 7, No 4.

[8] Cantrell, G., Dampier, D., Y. Dandass, Niu, Y., and Bogen, C. (2012), "Research Toward a Partially-automated, and Crime Specific Digital Triage Process Model," Computer and Information Science, vol. 5, no. 2, pp. 29–38.

[9] Christiansen, T. D Foy, B., Wall, L. and Orwant, J. (2012), "Programming perl: Unmatched power for text processing and scripting Fourth edition," O'Reilly Media, Sebastopol, CA.

[10] Garfinkel, S. L. (2009). Automating Disk Forensic Processing with SleuthKit, XML and Python. Systematic Approaches to Digital Forensic Engineering, 2009, (pp. 73-84).

[11] Henry-Labordere, A. (2004), "SMS and MMS interworking in mobile networks," Artech House, Norwood, MA.

[12] Gilbert, Adam M., Hunt,nd Winch, Kenneth C., (1997,

[13] Lessard, J. and Kessler, G. (2010), "Android forensics: Simplifying cell phone examinations," Small Scale Digital Device Forensics Journal. Vol. 4, No. 1.

[14] McCarthy, P. (2005). Forensic Analysis of Mobile Phones. University of South