Digital Investigation (2004) 1, 28—43

ELSEVIER

Digital Investigation

www.elsevier.com/locate/diin

TOOL REVIEW

Network traffic as a source of evidence: tool strengths, weaknesses, and future needs

Eoghan Casey

Knowledge Solutions LLC, 61535 S Hwy 97 #9-148, Bend, OR 97702, USA

Received 11 December 2003; revised 19 December 2003; accepted 19 December 2003

KEYWORDS
Network traffic;
Network investigations;
Digital evidence;
Forensic examination;
Computer crime

Abstract Digital investigators require specialized knowledge and tools to process network traffic as a source of evidence. Existing open source tools can be used for basic tasks in simple cases but lack the functionality of commercial tools that are specifically designed to process network traffic as evidence. These commercial tools reduce the amount of time and specialized technical knowledge required to examine large quantities of network traffic but even these tools are lacking from a forensic standpoint. This paper discusses the strengths and shortcomings of existing tools in the context of the overall digital investigation process—specifically the collection, documentation, preservation, examination and analysis stages. In addition to highlighting the capabilities of different tools, this paper familiarizes digital investigators with different aspects of network traffic as a source of evidence. Based on this discussion, a set of requirements is proposed for tools used to process network traffic as evidence in the hope that existing developers will enhance the capabilities of their tools to address the weaknesses.

Introduction

With the increasing cost of computer crimes such as computer intrusions, intellectual property theft, destructive programs, software and media piracy, and extortion committed with the assistance of computers, organizations are seeking more effective ways to detect and respond to these problems. As a result, information security professionals are being asked to cultivate their networks as a source of evidence, installing systems to monitor log files and network traffic for suspicious activities. Captured network traffic is a particularly compelling form of digital evidence because it can be used to show all of the offender's actions, like a videotape of a convenience store robbery.

Many organizations underestimate the importance of processing digital evidence, not realizing that these data provide the foundation for conclusions and decisions relating to an incident. Weak evidence can lead to inaccurate conclusions and poor decisions that can cause more damage and liability than the incident itself. For instance, when employees are fired as a result of an incident but claim that their dismissal was unfair or unfounded, improperly processed evidence can make it more

E-mail address: eco@corpus-delicti.com.

1742-2876/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.diin.2003.12.002

difficult to justify the decision and defend against the unfair dismissal claims. This puts the organization in a potentially costly situation if the employees sue. Therefore, even when an incident will be handled internally by an organization and will not result in legal action, the associated digital evidence should be handled using the same principles as digital evidence that is destined for court.

As more organizations preserve traffic relating to offenses on their networks, law enforcement agencies need to become familiar with this form of digital evidence. Additionally, as attorneys present stronger defences, criminal investigators need to build stronger cases by gathering more corroborating evidence. For instance, when a defendant claims that someone else placed child pornography on his computer via a Trojan horse program, network's traffic collected prior to seizing his computer could be used to disprove this claim. Furthermore, this network traffic could be used to demonstrate that the defendant distributed child pornography to others on the Internet, resulting in more severe charges. In addition to substantiating evidence found on an offender's computer, network traffic can contain information that is not otherwise available such as IP addresses of cohorts and passwords used to access servers or Internet Relay Chat (IRC) channels.

Network traffic presents a number of challenges as a source of evidence. There is generally only one opportunity to capture data as it travels through a network and inadequate evidence collection systems can result in unrecoverable losses. Additionally, networks comminute data before transmitting them, making it necessary to piece together packets to obtain data in their original form. It can also be difficult to find and extract specific items from the large number of flows on a network. Furthermore, network traffic comprises many different protocols and media types, adding complexity to an already complicated source of digital evidence.

Fortunately, tools are available that provide reliable acquisition and powerful analysis capabilities. Many of these tools are designed with information security in mind rather than evidence processing and, therefore, have shortcomings from a forensic standpoint. The term *forensic* is used to describe a characteristic of evidence that satisfies its suitability for admission as fact and its ability to persuade based upon proof or high statistical confidence (Casey and Palmer, 2004). This characteristic applies to disciplinary hearings in an organization as well as legal proceedings in court. By processing digital evidence properly, an organization can protect themselves against liabilities such as invasion of privacy and unfair dismissal claims when dealing with internal disciplinary matters relating to policy violations.

This paper outlines the fundamental requirements of tools that are used to collect, reconstitute and dissect network traffic, demonstrating key issues using both open source and commercial tools. The first part of this paper treats the examination and analysis of network traffic that has already been captured, demonstrating how this type of evidence can be useful in an investigation. The second part of this paper deals with the more mundane but critical issues of collection, preservation and documentation. In each section, strengths and weaknesses in specific tools are presented and suggestions for standards and future research are provided. Although commercial applications that are designed with evidence handling in mind have more capabilities than open source tools, no tool currently fulfills all of the requirements.

Examination and analysis

When assessing tools that are used to process digital evidence, it is useful to clarify the difference between examination and analysis. In essence, the examination process extracts and prepares data for analysis. The examination process involves data translation, reduction, recovery, organization, and searching. For example, know files are excluded to reduce the amount of data, and encrypted data are decrypted whenever possible to recover incriminating evidence. A thorough examination results in all relevant data being organized and presented in a manner that facilitates detailed analysis. The analysis process involves critical thinking, assessment, experimentation, fusion, correlation, and validation to gain an understanding of and reach conclusions about the incident based on available evidence (Casey and Palmer, 2004). In general the aim of the analysis process is to gain insight into what happened, where, when, how, who was involved, and why.

For example, in a child pornography investigation, the product of the examination process would include all graphics or video files from network traffic, as well as Web sites accessed and all Internet communications such as IRC, Instant Messaging (IM), and e-mail. Furthermore, the examination process would involve a search for specific usernames and keywords to locate additional data that may be relevant. Once most of the data that might be relevant to the investigation have been extracted from network traffic and made readable, they can be organized in ways that help an individual analyze them to gain an

understanding of the crime. As the analysis process proceeds, a more complete picture of the crime emerges often resulting in leads or questions that require the analyst to return to the original data to locate additional evidence, test hypotheses and validate specific conclusions.

As another example, in a computer intrusion investigation the product of the examination process would include known hacker toolkits, summaries of host activities (e.g., tabulating top talkers, top pairs), potentially malicious activities (e.g., using Snort signatures, deviations from network activity baselines), as well as all Internet communications such as IRC. Additionally, the examination process would involve a search for specific usernames, IRC channel names, and keywords to locate additional data that may be relevant. These data are then analyzed to develop a better understanding of the incident, again resulting in leads or questions that require the analyst to return to the original data to locate additional evidence, test hypotheses and validate specific conclusions.

The examination process is generally more susceptible to computer automation than analysis since the later requires some degree of critical thinking. For instance, in a child pornography investigation, all images are extracted from network traffic during examination and then an individual analyzes them to determine which are relevant to the case. In an intrusion investigation, all host interactions are produced during the examination and then an individual analyzes them to determine which are relevant to the incident and to interpret their significance and meaning. This is not to say that computer automation is not useful for certain forms of analysis. On the contrary, computers can be very helpful for finding links and patterns in data that a human analyst might otherwise overlook. However, such analysis tools require more human interaction than examination tools that simply extract and present data in a way that facilitates analysis.

Flow reconstruction

Because most networks use TCP/IP to transmit data between hosts, it will be the focus of this paper but the concepts presented here can be applied to other forms of network traffic. Each TCP connection (a.k.a. *TCP stream*) is bi-directional, comprising one *flow* for receiving data and a second *flow* for sending data. Because TCP breaks data into packets prior to transmission, tools for examining network traffic require some ability to reconstruct flows as depicted in Fig. 1.

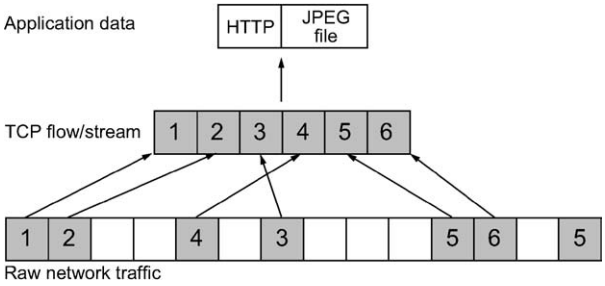

Figure 1 A conceptual representation of packets in network traffic relating to a single flow being extracted and reconstituted to obtain the data they carry.

Note that packets may need to be put into the correct order and duplicates may need to be discarded before the data are available in its original form. For further discussion of network abstraction layers and protocols see Casey (2004).

Many tools for examining network traffic have the ability to reconstruct TCP streams. The open source `tcpflow` utility can be used to break network traffic into individual flows as shown here, placing data from each flow in a separate file labeled with the source and destination IP addresses.

```
$tcpflow -v -r kazaa-2003112901.dmp
tcpflow[1248]: tcpflow version 0.21 by
  Jeremy Elson ⟨jelson@circlemud.org⟩
tcpflow[1248]: looking for handler for
  datalink type 1 for interface kazaa-
  test3.dmp
tcpflow[1248]: found max FDs to be 16
  using OPEN_MAX
tcpflow[1248]: 192.168.000.005.01657-
  216.228.047.214.03359: new flow
tcpflow[1248]: 192.168.000.005.01657-
  216.228.047.214.03359: opening new
  output file
⟨cut for brevity⟩
tcpflow[1248]: 068.226.166.163.01214-
  192.168.000.005.01668: new flow
tcpflow[1248]: 068.226.166.163.01214-
  192.168.000.005.01668: opening new
  output file
tcpflow[1248]: 192.168.000.002.49178-
  066.113.201.011.00110: closing file
tcpflow[1248]: 192.168.000.005.01261-
  192.168.000.003.00139: new flow
tcpflow[1248]: 192.168.000.005.01261-
  192.168.000.003.00139: opening new
  output file
tcpflow[1248]: 066.113.201.011.00110-
  192.168.000.002.49178: closing file
```

Additional effort is required to extract useful information from these flows. Unfortunately, the UNIX `file` utility is not useful for classifying the data in all of these flows because many contain additional information as shown here.

```
$ cat 012.218.244.049.02048-192.168.
  000.005.01666
HTTP/1.1 206 Partial Content
Content-Range: bytes 4471656-
  6005739/6005740
Content-Length: 1534084
Accept-Ranges: bytes
Date: Sat, 29 Nov 2003 16:50:41 GMT
Server: KazaaClient Jul 6 2003 17:
  54:13
Connection: close
Last-Modified: Sat, 11 Oct 2003 05:
  27:22 GMT
X-Kazaa-Username: jstrsclwn
X-Kazaa-Network: KaZaA
X-Kazaa-IP: 12.218.244.49:2048
X-Kazaa-SupernodeIP: 12.212.26.
  202:2748
X-KazaaTag: 4=Hey Ya (Real Song)
X-KazaaTag: 6=Outkast
X-KazaaTag: 8=Speakerboxxx/Love Below
X-KazaaTag: 14=hip-hop
X-KazaaTag: 1=2003
X-KazaaTag: 10=en
X-KazaaTag: 26=4:08
X-KazaaTag: 12=Outkast
X-KazaaTag: 5=365
X-KazaaTag: 21=135
X-KazaaTag: 3==Q7CdYz2sanwkyi/
  AMQM8alVCsOI=
Content-Type: audio/mpeg
⟨binary MPEG data cut for brevity⟩
```

In this instance, KaZaA downloaded an MPEG audio file in three segments from three different sources. Therefore, to obtain the complete file, the three segments have to be combined in the correct order.

```
$ grep -a Content-Range 'grep mpeg *
  | awk '{print $3}''
012.218.244.049.02048-192.168.000.
  005.01666:Content-Range: bytes
  4471656-6005739/6005740
066.031.090.197.01214-192.168.000.
  005.01664:Content-Range: bytes
  0-4242993/6005740
068.226.166.163.01214-192.168.
  000.005.01668:Content-Range: bytes
  3921372-4242993/6005740
```

Another open source tool called Ethereal has a "Follow TCP Stream" feature that reconstructs both directions of a TCP stream and displays the data that were transferred. For instance, Fig. 2 shows this Ethereal feature used to recover part of the MPEG file that was transferred using KaZaA in the previous example.

In addition to reconstructing TCP streams, Ethereal has the ability to interpret (a.k.a. decode) some application layer protocols that are encapsulated within the streams.

Protocol decoding

By decoding protocols, more information can be obtained and more filtering and searching functions can be performed to locate important items. For instance, by decoding File Transfer Protocol (FTP) traffic as shown in Fig. 3 it is possible to create a filter that focuses on FTP commands, making it easier to see user activities and find important items like passwords.

Importantly, Ethereal makes assumptions about the expected behavior of protocols that prevent it from automatically classifying traffic that does not meet these basic assumptions. In Fig. 3a, Ethereal does not automatically recognize and classify FTP traffic because a port other than the default port (21) was used. Therefore, it was necessary for an individual to inspect packets manually, identify the FTP traffic, and instruct Ethereal to decode it correctly.

Manually inspecting packets is time consuming, and correctly identifying protocols requires specialized knowledge. To reduce the amount of time and specialized knowledge required to examine network traffic, some commercial applications provide more powerful decoding features. For instance, NetIntercept automatically identifies and decodes a wide variety of protocols based on the associated protocol standards rather than relying on the default behavior of protocol implementations. The usefulness of this protocol decoding feature is most apparent when examining traffic that contains compound objects such as a Word document within a Zip file within an MIME encoded e-mail attachment as shown in Fig. 4. Without the capability to decode each layer, keyword searches would not find relevant text in these Word documents.

NetIntercept's protocol decoding capabilities are also useful for finding traffic that violates expected behavior such as an FTP server running at a non-standard port. This protocol anomaly detection feature is conceptually similar to the file signature mismatch detection provided by most

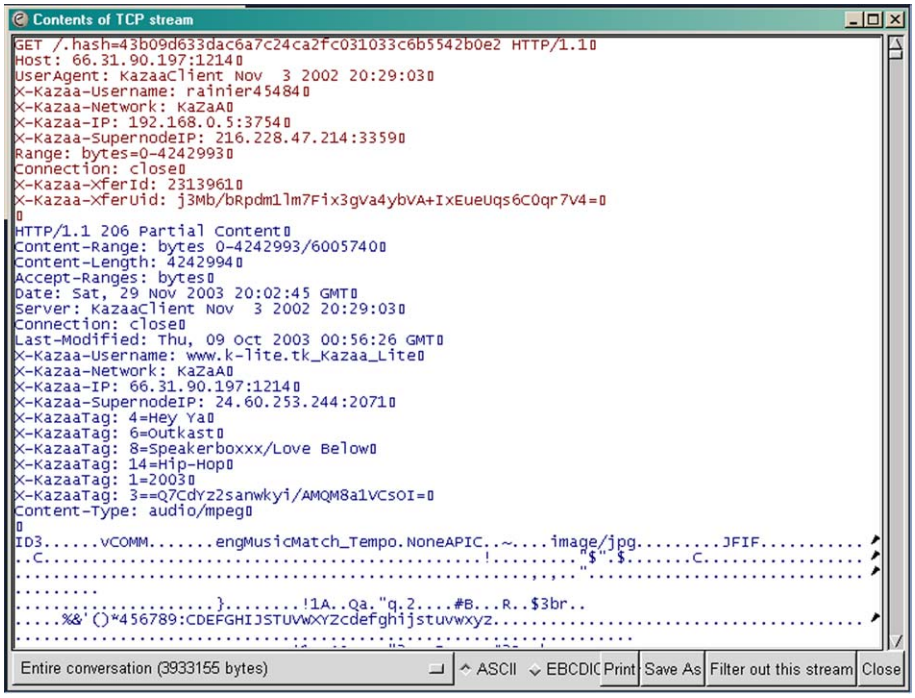

Figure 2 Ethereal used to reconstruct one TCP stream in a peer-to-peer file exchange. The HTTP GET request from one KaZaA client is shown at the top of the screen and the corresponding response from the other KaZaA client is shown on the lower portion of the screen.

media examination tools like FTK and EnCase. For instance, this feature can uncover some methods of hiding data within network traffic, helping investigators identify suspicious activities. Also this feature can be used to detect when packets were dropped based on missing SEQ numbers. NetIntercept lists all such anomalies in an Alerts section of its graphical user interface and can generate a printable report of this information.

NetDetector can also decode a wide variety of protocols and sometimes uses variable characteristics such as port numbers to decode some protocols (e.g., VoIP). This ability to automatically decode a wide range of protocols is useful when processing large amounts of network traffic, extracting large numbers of files, searching for particular types of data, or examining complex interactions between many hosts. Additionally, when protocols are decoded automatically, useful examination features can be implemented more effectively such as keyword searching in Fig. 4 and an image gallery that shows thumbnails of all graphic files recovered from network traffic.

Whenever a tool is used to abstract information from raw digital data, errors can occur in the form of Tool Implementation Errors and Abstraction Errors (Carrier, 2003). For this reason, courts often require some assurance that the computer systems involved were functioning properly (PACE, Section 69). To assess how reliable a given tool is and find Tool Implementation Errors, it is necessary to review the source code and/or test it under controlled conditions (Carrier, 2002). No formalized testing or code review has yet been performed on tools for processing network traffic by an objective group such as NIST Computer Forensic Tool Testing Program [http://www.cftt.nist.gov]. Code review can also reveal security vulnerabilities that could be used to interfere with the tool or evidence.

Abstraction Errors can occur when data are summarized to facilitate examination as described in the next section. Because of the Abstraction Error inherent in such summarizations, digital evidence examiners should verify the underlying data and translation before reaching conclusions based on the data. For instance, Tcpdump or Ethereal can be used to view network traffic at a low level and validate results obtained using more sophisticated tools. Also, NetIntercept and NetDetector have components that enable this type of low-level packet inspection, giving examiner's access to the data in a less abstracted form.

Data reduction

To locate specific items of interest in network, it is usually necessary to perform some form of data

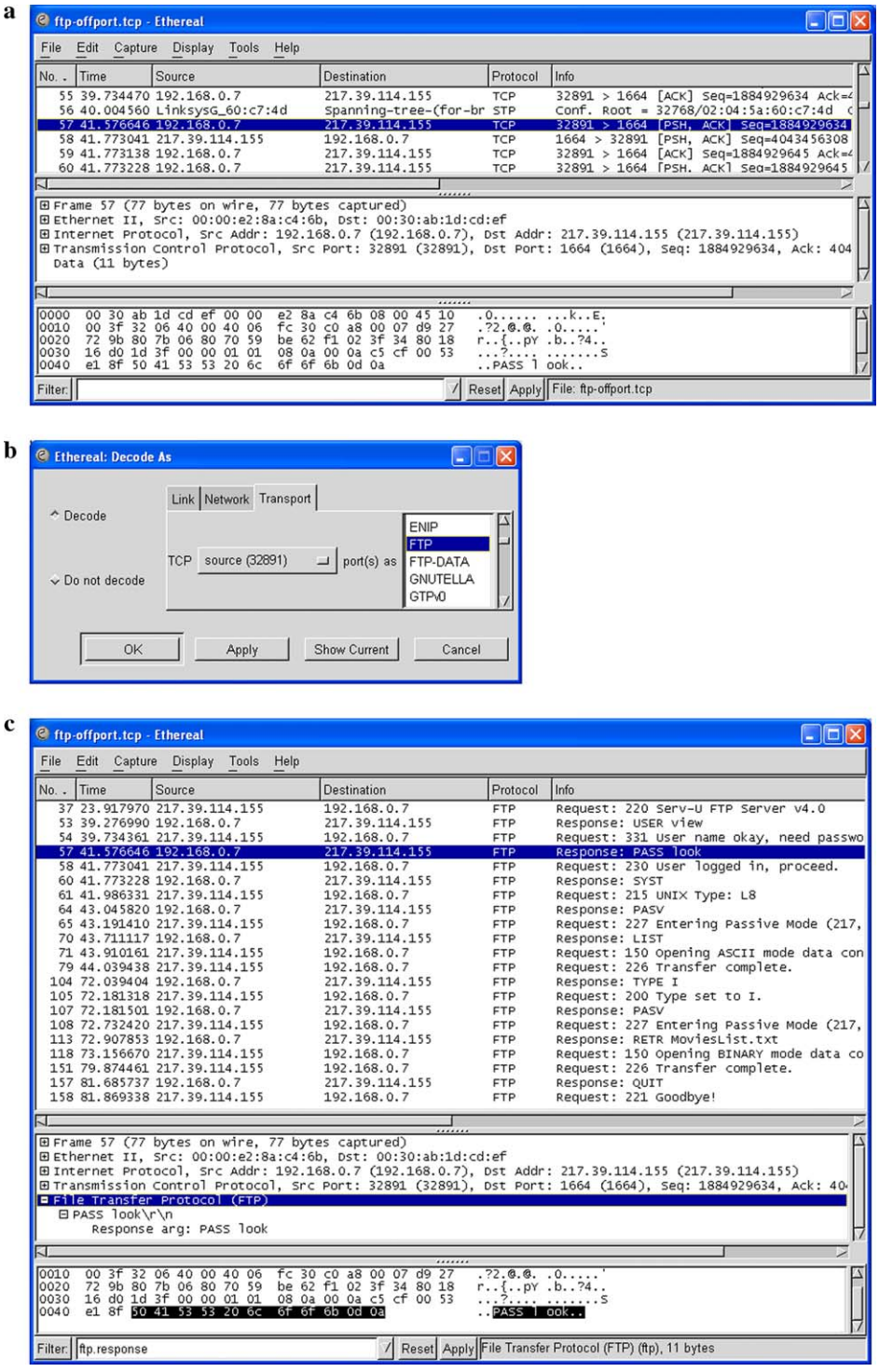

Figure 3 (a) FTP traffic viewed using Ethereal prior to decoding the protocol. (b) The "Decode As" feature in Ethereal being used to decode FTP traffic. (c) FTP traffic viewed using Ethereal after decoding the protocol and applying a filter that focuses on FTP commands and responses.

reduction. To extract only Web traffic, for instance, one might look for traffic to port 80 but this would miss relevant Web traffic if the server was using a different port, such as 8080. Additionally, this approach might fail to exclude large amounts of undesirable traffic (e.g., IM, P2P) that is using port 80 to pass through a firewall. Ethereal has a comprehensive syntax and a graphical user interface to help users construct filters that specify what data to display.

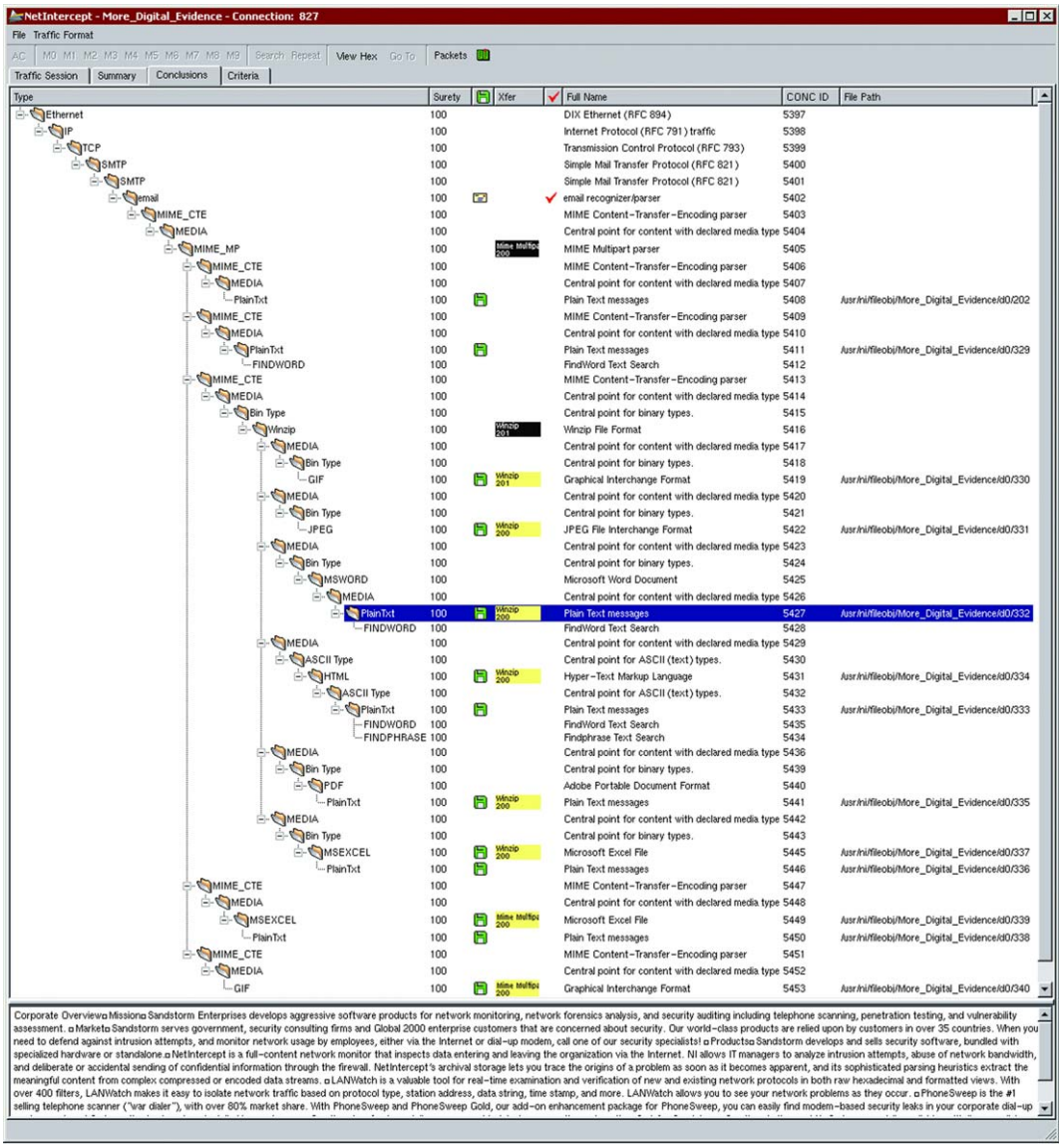

Figure 4 NetIntercept being used to decode e-mail traffic and display a Word document within a Zip file within an MIME encoded e-mail attachment, enabling keyword searches of the Word documents (figure from Casey, 2004).

As noted in the previous section, applications that automatically decode protocols can support more advanced data processing and filtering features. For example, after decoding protocols, NetIntercept extracts noteworthy elements such as usernames, passwords, file names and credit card numbers, and stores them in a database to facilitate examination and analysis. NetIntercept presents this data in a simple but powerful graphical user interface shown in Fig. 5 to help perform data reduction. By selecting items of interest in the columns, a digital evidence examiner can focus on certain types of network traffic, specific IP addresses, and even specific file names. Additional columns can be added by the examiner as needed.

Also, NetIntercept and NetDetector both have the ability to summarize and tabulate data extracted from network traffic to help examiners find interesting items. For example, a list of the top talkers or top pairs on a network can reveal hosts that are being misused. NetDetector uses Snort intrusion detection signatures to identify potentially malicious activity (Fig. 6).

Notably, tools for processing network traffic do not currently have a feature to identify known files using MD5 values. This technique is used in media analysis to exclude files that are not relevant to an investigation such as those in the NIST NSRL and Hashkeeper hash sets. By realizing that network flows are simply streams of bytes similar to

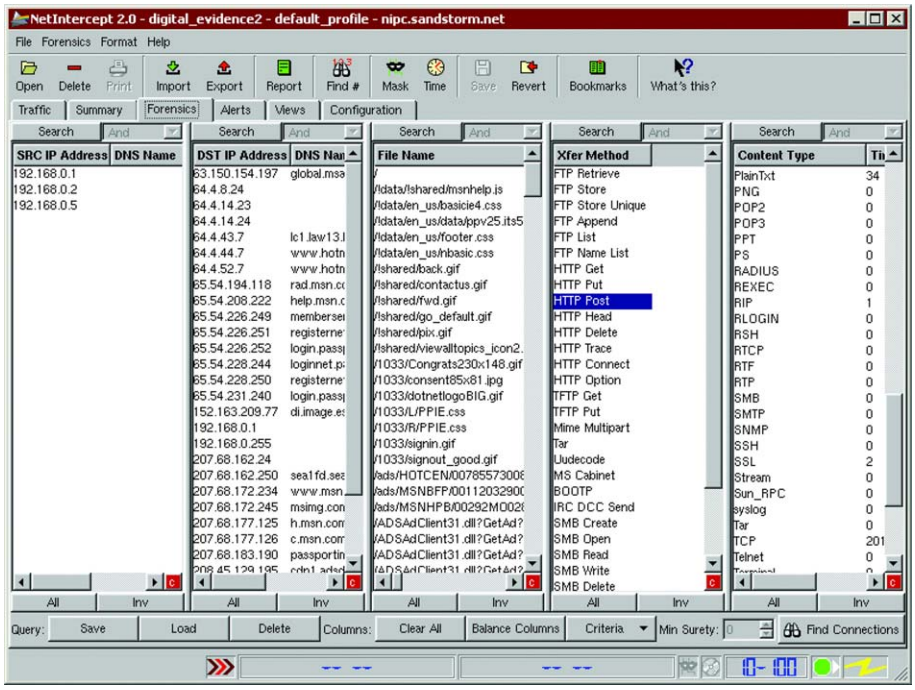

Figure 5 The Forensics tab in NetIntercept facilitates data reduction (figure from Casey, 2004).

data on a hard drive, the same technique can be used to identify known files in network traffic. This technique can also be used to spot suspicious files such as known hacker tools and exploits, or intellectual property that should not be transferred on the network.

Data recovery

Digital objects such as e-mail messages, JPEG images, and Word documents must be recovered from network traffic before they can be analyzed. Some digital objects can be extracted quite easily form network traffic using `tcpflow`, such as the Zip archive transferred using FTP, shown here in bold.

```
$ file *
066.113.201.011.00025-
  192.168.000.002.49161: ASCII text,
  with CRLF line terminators
066.113.201.011.00110-
  192.168.000.002.49160: ASCII mail
  text, with CRLF line terminators
066.187.232.101.00443-
  192.168.000.006.32777: data
192.168.000.002.49161-
  066.113.201.011.00025: ASCII English
  text, with CRLF line terminators
192.168.000.006.32777-
  066.187.232.101.00443: 8086 relo-
  catable (Microsoft)
216.218.166.002.00021-
  192.168.000.002.49156: ASCII English
  text, with CRLF line terminators
216.218.166.002.01397-
  192.168.000.002.49159: Zip archive
  data, at least v2.0 to extract
$ unzip 216.218.166.002.01397-
  192.168.000.002.49159
Archive: 216.218.166.002.01397-
  192.168.000.002.49159
   inflating: secret.doc
   inflating: janel.tif
   inflating: jane2.tif
```

Note that the original name of the Zip archive must be obtained from another flow file. To recover the same Zip archive sent as an e-mail attachment, it is first necessary to decode the MIME encoded e-mail attachment.

```
$ munpack 192.168.000.002.49161-
  066.113.201.011.00025
rays-files.zip (application/zip)
[eco@case file-transfers]$ unzip rays-
  files.zip
Archive: rays-files.zip
   inflating: secret.doc
   inflating: janel.tif
   inflating: jane2.tif
```

Using these open source tools to recover digital objects can be time consuming, requires technical

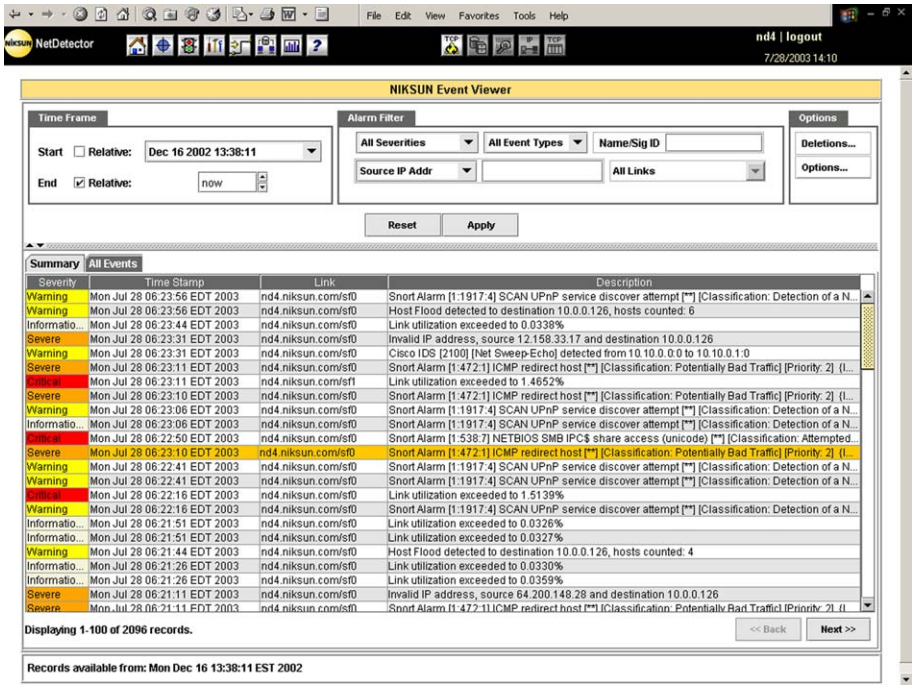

Figure 6 Snort Alerts displayed in NetDetector's Event Viewer. NetDetector's Snort feature can operate in real-time or on stored data for post-event analysis.

knowledge, and makes it more difficult to perform keyword searches of all files contained in network traffic. In this example, text in the "secret.doc" Word document can only be searched after it is extracted from the MIME encoded Zip archive. NetIntercept makes this process much easier by automatically recovering files (see the file column in Fig. 5) and including their contents in keyword searches (see Fig. 4). Additionally, NetIntercept can be configured to automatically decrypt SSL sessions if the associated encryption keys are available and can decrypt SSH sessions to specially modified servers (Corey et al., 2002).

Recovering digital objects using Ethereal is time consuming and requires some skill. For instance, to recover the reconstructed MPEG audio segment in Fig. 2, it is necessary to use the "Save As" button in the bottom right of the screen to export the data to a file. In this example, the file was downloaded in three segments making it necessary to export and combine each segment to reconstitute the whole file. Performing this process more than a few times is an inefficient use of digital evidence examiners' time.

When reconstructing HTML pages using a tool like Ethereal, it is necessary to export and view the page in a Web browser. Opening a reconstructed Web page in a standard browser can have undesirable consequences such as downloading content from the Internet or executing malicious code in the page. At the very least, this spoliation of the evidence should be avoided by performing the examination on a computer that is not connected to the Internet. This also demonstrates the importance of understanding the limitations and quirks of the tools being used to examine digital evidence. To avoid this particular type of spoliation, tools like NetIntercept and NetDetector display reconstructed Web pages in a sandbox viewer that cannot access the Internet or execute code. They also ensure that all components of a reconstructed page refer to objects within the captured evidence. A placeholder is inserted if an internal object is not available (e.g., content that was cached on the user's system and did not traverse the network).

In addition to more readily recoverable forms of data, information hidden in network traffic may be of interest in some cases. Some tools may be able to identify hidden data as an anomaly (e.g., packet size larger than claimed in header). However, not all data hiding techniques can be detected so easily and researchers are developing methods and tools to detect more forms of hidden data in network traffic, including steganography.

Keyword searching

As with storage media, keyword searches can be performed at the physical or logical level. Consider an e-mail message with the word 'blackmail' is split such that one packet contains 'black' and

the other contains 'mail'. Logical searches of network traffic can be performed using open source tools, by first reconstructing flows (e.g., using `tcpflow`) and then searching the reconstructed data for keywords (e.g., using `grep`). NetIntercept is also capable of performing keyword searches at the logical level and will find keywords that are split between packets. NetDetector has different search features including a command line tool for constructing regular expression searches. Further testing is required to determine how flexible and reliable the search features are in these tools (e.g., Unicode support, logical searches, regular expression searches).

As noted in the previous section, performing keyword searches of encoded or compressed data using open source tools requires more effort than commercial tools like NetIntercept and is less efficient when dealing with large quantities of data.

Analysis

The process of analyzing available digital evidence to determine their meaning and significance can be very involved. It is often necessary to develop a timeline of significant events to obtain an overview of what occurred. It may also be necessary to analyze temporal data in other ways to identify patterns, gaps or other noteworthy details. For instance, a histogram showing the number of events for each unit of time can reveal periods of abnormally high activity that deserve deeper inspection. Performing relational reconstructions can reveal links between entities. For instance, diagrams showing which hosts communicated with each other (see Fig. 7) can be useful for understanding what occurred and locating additional sources of evidence on a network.

It can also be enlightening to create relational diagrams showing which users are connected to which systems. Combining temporal and relational data can further facilitate analysis. For instance, a diagram showing when certain users connected to certain systems can help an analyst gain a more complete understanding of what occurred.

There are many other techniques for correlating and analyzing data using computers to help find noteworthy patterns and details (e.g., clustering, image recognition). Correlation in the context of analyzing network traffic involves finding other events that are associated with an event of interest. For instance, when computer intruders gain access to a host, it is generally desirable to determine if they gained access to other host on the network. Simple correlation tasks can be performed using the same utilities described in the examination section. For instance, Ethereal can be used to only view traffic relating to a specific host (e.g., showing all hosts that an intruder targeted). Similarly, the NetIntercept Forensics interface (Fig. 5) can be used to focus on traffic relating to a specific host. It can also be useful to focus on traffic relating to a specific file name or transfer method. For example, selecting the HTTP POST transfer method (highlighted in Fig. 5) will show all occurrences of data being posted to a Web page using this method.

Using the NetIntercept Forensics interface, it is also straightforward to find all traffic relating to a specific user account (e.g., a suspect's account) which can help identify insecure account/password combination (e.g., Administrator with blank password). Also, when dealing with more sophisticated offenders, analysts are often looking for specific patterns of behavior such as a particular sequence of packets or commands. None of the tools mentioned in this paper currently support this type of analysis, making it necessary for analysts to search for distinctive features and then determine whether they meet the pattern of behavior they are seeking. Although it can be useful to have integrated correlation and data mining features, the absence of such features should not be viewed as a shortcoming since one tool cannot perform all tasks. When analysts feel that a certain type of computer assisted analysis might produce useful results, they can use a tool that is specially designed for that purpose.

Collection, documentation, and preservation

When collecting digital evidence it is necessary to use hardware and software well suited to the task of capturing all available data, documenting any losses, and establishing the integrity and chain of custody of the evidence. It is also important to protect digital evidence from malicious interference and unauthorized access. Therefore, systems for collecting network traffic as evidence must be highly secure and should only allow authorized encrypted remote connections, ideally via an interface that is only accessible from a private network.

It is also helpful when tools log all user actions to maintain chain of custody and to help others review a digital evidence examiner's actions. NetDetector maintains this type of log as shown in Fig. 8. The ability to review an examiner's actions helps

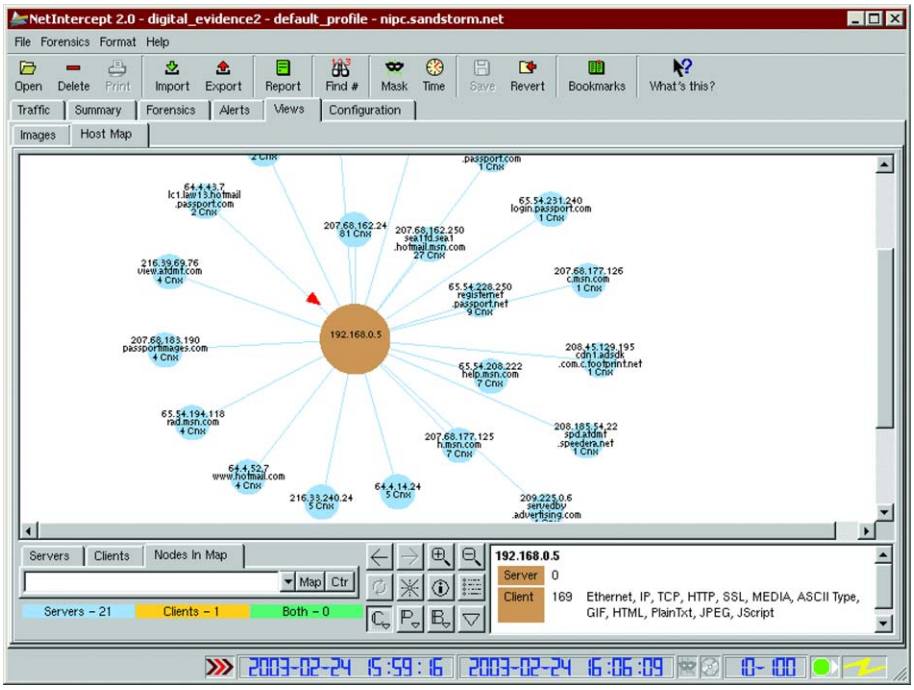

Figure 7 Host diagram from NetIntercept (figure from Casey, 2004).

others assess the work, reproduce the results, and determine if anyone viewed private data without authorization or exceeded their authorization by exceeding the bounds of an organization's privacy policy or a search warrant. When using tools that do not create this type of audit record, it is necessary to make detailed written notes. Although it is always a good practice to keep written notes when processing evidence, they rarely have the level of detail of computer generated audit logs.

Hardware and operating systems

Because there is only one opportunity to capture network traffic, it is important to use hardware and operating systems that are best suited to the task. Microsoft Windows is not particularly efficient at capturing network traffic and can become overloaded in high bandwidth environments, failing to collect important data. Some informal studies have found that OpenBSD and FreeBSD are the most reliable operating systems for collecting traffic on high bandwidth networks (Garfinkel, 2002). However, informal testing in the academic information security community indicates that Linux performs better than FreeBSD in some situations. No data have been published so a formal study is required to ascertain which systems perform better in different situations, taking into account that performance may vary depending on the average packet size, whether or not payload data are being captured, and how much packet inspection is being performed during the collection process.

Stability and security are also important considerations when developing an evidence processing system. A stable platform is required to minimize the risk of system crashes during evidence collection. A secure system is necessary to prevent malicious individuals from interfering with or gaining access to evidence via the network. FreeBSD and OpenBSD have the advantage of being quite stable and are designed to be secure by default. Although recent versions of MS Windows are quite stable, the relatively high number of vulnerabilities that have been found in recent years is a concern from a security standpoint.

Some applications for capturing network traffic as a source of evidence run on MS Windows, despite its shortcomings. For instance, DCS1000 (formerly known as Carnivore) runs on Windows making it better suited for low bandwidth network segments. Also, given the number of vulnerabilities that are being found in Windows operating systems, great care must be taken to secure these systems against intruders and malicious code on the Internet.

NetIntercept and NetDetector run on versions of FreeBSD that are customized to make them more efficient at capturing network traffic. Additionally,

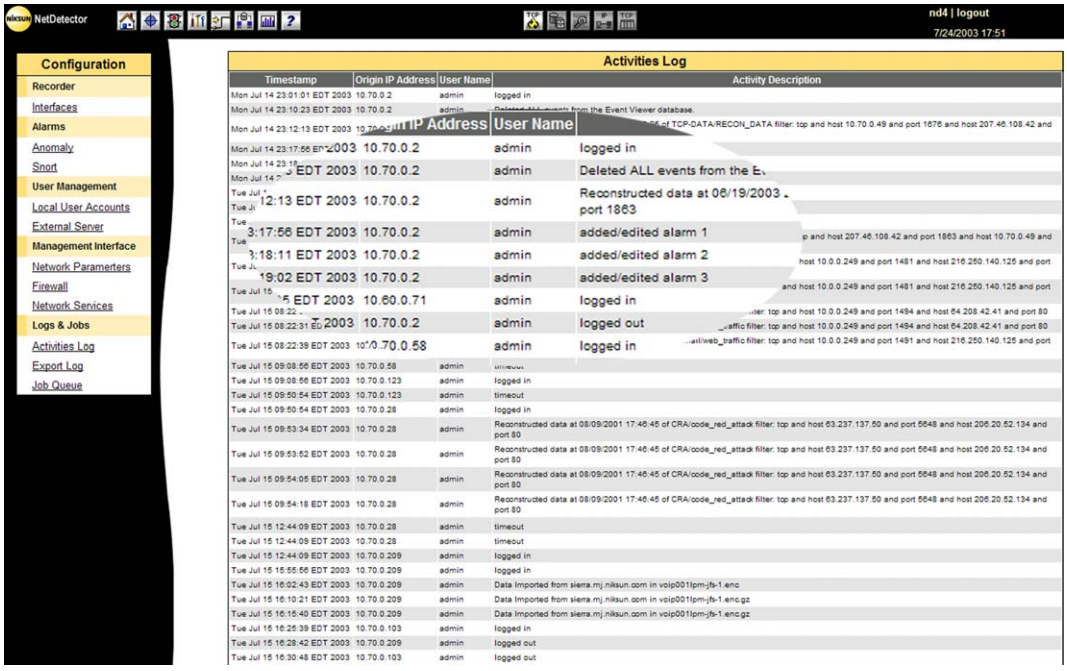

Figure 8 NetDetector Activities Log showing user activities relating to system access and data manipulation.

as a result of their performance tests, the NetIntercept designers selected Intel EtherExpress network interface cards. Taking a different approach, NetDetector systems can use a variety of network interfaces to enable direct physical access to wide-area networks (Venezia, 2003). Although this type of physical network access may be desirable in some cases, it is usually adequate to connect the evidence collection system to a switch via an Ethernet interface.

One approach to monitoring traffic on a switch is to utilize a feature called Switched Port Analyzer (SPAN). A *SPANned* port (a.k.a. mirrored port) enables eavesdropping by copying network traffic from one port on the switch to another. However, a SPANned port only copies valid Ethernet packets, does not duplicate all error information, and the copying process receives lower priority than routine data transmission which may increase dropped Ethernet frames on the monitoring port (Cisco, 2003). These shortcomings are a concern when collecting evidence because they can interfere with a complete and accurate copy of the network traffic. To mitigate these shortcomings, a hardware tap such as those made by Finisar [http://www.finisar.com] or NetOptics [http://www.netoptics.com] can be used to connect the evidence collection system to the switch port of interest. In this way, an evidence acquisition system can collect an exact copy of network traffic and any error information relating to the switch port can also be collected if desired.

Open source applications like Tcpdump and Ethereal can be compiled on a variety of platforms and are attractive because they are free. Although it can be less costly to create one's own network traffic acquisition system, this approach requires expertise. Also, by creating a system, an individual takes on the added responsibility of assuring decision makers that the system was functioning properly. Purchasing a system from a vendor takes advantage of their expertise and technical support. Furthermore, the vendor can be called on to explain the configuration and performance of the system in court, removing this burden from digital evidence examiners and allowing them to dedicate their attention to the investigation.

Standards for collection and interoperability

When collecting network traffic, the de facto standard is to store the data in a Tcpdump file with a ".dmp" extension. Most tools can import Tcpdump files, enabling digital evidence examiners to import data into their tool of choice, independent of the collection tool used. Most tools can also export in Tcpdump format, even if they store data in a proprietary database.

Notably, not all tools capture payload data by default. For instance, Tcpdump only captures 68 bytes of each packet by default whereas Ethereal uses 65 535 as a default. Other tools have

snap lengths between these two extremes. To ensure that all data are collected, it is generally advisable to configure network traffic acquisition tools with a large maximum capture length (a.k.a. snap length) such as 65 535 bytes.

Applications for collecting network traffic as a source of evidence should be read-only to prevent interference with evidence and to increase security. In other words, these tools should not send any data onto the network being monitored, including Address Resolution Protocol (ARP) replies or Domain Name System (DNS) queries. Monitoring systems can achieve this type of read-only access by using network interface cards without a transfer capability, by not loading a TCP/IP stack on the monitoring card, and by using a host-based firewall that prevents outbound connections. To enable legitimate remote access and administration, a second network interface connected to a physically separate network is generally used. Although DNS queries can be made through this second interface, it is a better practice to use DNS queries in the captured network traffic to resolve IP addresses. Ethereal, NetIntercept, and NetDetector all perform this internal DNS resolution. In this way, the original IP address to hostname mapping is preserved even if the associated entry in DNS has since changed.

Documenting losses

A failure to capture all available network traffic can result in criticisms that the data does not accurately represent what occurred, thus undermining efforts to use the data to prove what happened. The primary approach to demonstrating that a complete and accurate copy of network traffic was obtained is to routinely document any losses. It may not be feasible to prevent all losses and errors, but it is important to take steps to minimize losses and document them when they do occur.

When collecting network traffic, losses can occur in several places—interface cards can drop packets or programs used to capture network traffic can become overloaded and fail to retain all packets captured by the kernel (Casey, 2002). Although TCP is designed to retransmit dropped packets, evidence collection tools are not active participants in the communication channel and will not cause packets to be resent. Therefore, dropped packets are permanently lost.

Most evidence acquisition tools have access to the number of packets dropped by the kernel but not all record this information. Additional loss of information can be obtained from network interface cards on most UNIX systems using `netstat` or SNMP (Casey, 2002). Also, when network traffic is being monitored through a spanned port on a switch, the interface on some switches can be polled for errors when the capture is complete. Presumably these losses will be negligible unless the system is malfunctioning but documenting the errors or lack thereof will help establish this fact. Some tools document the number of packets dropped by the kernel, and NetDetector documents packets dropped by the interface card. If losses on the switch are of concern, this information can be obtained manually.

Integrity

When presented with a case, decision makers generally want assurance that the associated digital evidence has not been altered since it was obtained. The most common approach to documenting the integrity of digital evidence is to calculate its MD5 or SHA1 hash value. When processing a hard drive as a source of evidence, it is feasible and generally desirable to calculate the MD5 value prior to copying the data it contains. However, MD5 calculations of network traffic must be calculated while data are being captured or after they are saved to disk. Performing this type of calculation during collection creates additional load on the system and can result in increased losses. Therefore, it is often preferable to calculate hash values of evidence files immediately after collection is complete, documenting this value for future reference.

Another approach to maintaining the integrity of digital evidence is to save it on write once media. NetIntercept comes with a compact disk writer to facilitate this type of preservation.

Filtering

Privacy is a concern when monitoring network traffic and a tool's ability to only monitor specific network traffic may be an issue in certain cases. For instance, in some cases it is desirable to only monitor traffic from a specific MAC address or IP address. This form of filtering is not particularly difficult when the address in question is known in advance. However, when IP addresses are assigned using DHCP or RADIUS, filtering becomes more complex and fewer tools support this type of filtering. DCS1000 can monitor RADIUS traffic to detect which IP address is assigned to a given dial-up user and monitor only traffic to and from that IP

address. DCS1000 can also monitor DHCP traffic to determine which IP address is assigned to a given host (IIT Research Institute, 2000). A special version of NetDetector apparently has a similar capability. However, further testing is required to determine how these tools deal with different versions and implementations of RADIUS or how they protect against forged RADIUS or DHCP packets.

In certain cases it is desirable to implement the pen register concept when capturing network traffic—only collecting packet header information or specific items such as e-mail headers. Collecting only packet headers and ignoring payload data is straightforward process. However, extracting only address information from e-mail headers requires payload inspection and can inadvertently reveal content such as subject lines. Given the complexity, associated cost of development, and potential problems with implementing the pen register concept, it has been recommended that specialized tools be developed for these purposes rather than developing these capabilities in all tools for examining network traffic as a source of evidence (Bellovin et al., 2000).

In general, it is inadvisable to filter network traffic based on protocol during collection because there is a risk that desired protocols might be excluded by the filter. For instance, many protocols are tunneled through HTTP and could be inadvertently excluded by a filter that ignores all HTTP traffic. When it is necessary to filter based on traffic characteristics during the collection process, it is safest to only exclude what is absolutely necessary.

Conclusions and recommendations

Free open source tools like `tcpflow` and Ethereal are useful for basic examination of network traffic. NetWitness has some search and display functionality that is not available in the free tools. NetIntercept and NetDetector have the advantage of being designed with evidence processing in mind and are feature-rich, providing powerful examination and analysis capabilities. NetIntercept's user-friendly interface enables users to quickly perform complex tasks without requiring a high degree of technical knowledge. The Snort integration in NetDetector is a valuable addition for computer intrusion investigations.

DCS1000 and a specialized version of NetDetector have tap-and-trace or pen register capabilities that may be required by government investigators in some cases to reduce the risk of mistaken privacy violations. It is advisable to have specialized tools for this type of filtering during collection (e.g., RADIUS interpretation, pen register concept to extract address information from e-mail headers). Implementing such features in all tools would be costly and would not be needed in most cases.

Although many of the systems described in this paper have well-developed network traffic examination and analysis capabilities, they are still evolving as evidence processing tools. The following basic requirements are proposed for tools used to process network traffic as evidence.

- Support Tcpdump format (import and export).
- Reliable protocol identification and reconstruction (e.g., detect protocol violations, review/test for Tool Implementation Errors).
- Data reduction (e.g., various methods of locating desired items and excluding extraneous data).
- Known files (e.g., use MD5 values to exclude known files, flag known bad files such as hacker tools and exploits, and flag suspicious files such intellectual property).
- Data recovery (e.g., automated file extraction and display, image gallery, reconstitute KaZaA fragments from multiple sources).
- Recover hidden data in network traffic (e.g., detect protocol anomalies and stegonography).
- Keyword search capabilities (across fragmented messages, regular expressions, Unicode).
- Documentation (e.g., audit trail of all digital evidence examiner actions, system performance, packet losses).
- Integrity (e.g., calculate MD5 value of captured digital evidence and record for future reference, save evidence onto write once media).
- Read-only during collection (e.g., no response on network including ARP replied, offline DNS resolutions).
- Read-only during examination (e.g., do not access graphics on Web servers when reconstructing and displaying a page).
- Complete collection (e.g., capture full packet, minimize losses).
- Security (e.g., secure remote access and administration, tools should not run as Administrator or root, review tool source code for vulnerabilities).

The following table rates the tools discussed in this paper in each of these areas, providing a

negative sign (−) when features are not implemented, a positive sign (+) when features are implemented, and a double positive sign (++) when features are implemented particularly well. Notably, although the commercial applications in this table are designed with evidence handling in mind and have more capabilities than the open source tools, no tool currently fulfills all of the requirements.

Capability	Tcpflow	Ethereal	NetIntercept	NetDetector
Tcpdump import and export	+	+	+	+
Flow/stream reconstruction	+	+	+	+
Protocol decoding	−	+	++	++
Data reduction	−	+	++	++
Known file detection/exclusion	−	−	−	−
Data recovery	−	+	++	++
Hidden data detection	−	−	+	−
Keyword searching	−	−	++	++
Audit log	−	−	−	+
Integrity checking mechanism	−	−	−	−
Loss documentation	−	−	−	+
Read-only collection	System dependent	System dependent	+	+
Read-only examination	−	−	+	+
Security	System dependent	System dependent	+	+

Once tools for processing network traffic as a source of evidence have matured, there will be a need for formalized validation testing such as the testing performed by the NIST Computer Forensic Tool Testing (CFTT) project. Such testing would include evaluations of performance under various conditions such as traffic load, average packet size, whether or not payload data are being captured, and how much packet inspection is being performed during the collection process. Additionally, certain components of these tools, such as the protocol decoding modules, could be made open source to enable others to validate the implementation. Since most network protocols have open standards it is a rational step for the associated decoding methods used in these tools to be open and possibly standardized to ensure the best possible implementation.

Tool list

DCS1000 (FBI)
Ethereal (http://www.ethereal.com)
NetDetector (http://www.niksun.com)
NetIntercept (http://www.sandstorm.net)
Tcpdump (http://www.tcpdump.org)
Tcpflow (http://www.circlemud.org/~jelson/software/tcpflow/)

Acknowledgements

I thank Walker Whitehouse at Sandstorm for responding to my endless requests, his enthusiastic assistance in testing the limits of NetIntercept, and his willingness to incorporate improvements into NetIntercept based on our discussions. I also thank Parag Pruthi at Niksun for taking the time to provide me with information and answer my questions relating to NetDetector.

References

Bellovin SM, Blaze M, Farber D, Neumann P, Spafford E. Comments on the carnivore system technical review, 2000. Available from: http://www.crypto.com/papers/carnivore_report_comments.html.

Carrier B. Open source digital forensic tools: the legal argument @stake Research Report, 2002. Available from: http://www.cerias.purdue.edu/homes/carrier/forensics/docs/opensrc_legal.pdf.

Carrier B. Defining digital forensic examination and analysis tool using abstraction layers. Int J Digit Evid 2003;1(4) Syracuse, NY, Available from: http://www.ijde.org/docs/02_winter_art2.pdf.

Casey E. Error, uncertainty and loss in digital evidence. Int J Digit Evid 2002;1(2) Available from: http://www.ijde.org/archives/docs/02_summer_art1.pdf.

Casey E. Digital evidence and computer crime: forensic science, computers and the internet. 2nd ed. Academic Press; 2004.

Casey E, Palmer G. The investigative process. Digital evidence and computer crime: forensic science, computers and the internet. 2nd ed. Academic Press; 2004.

Cisco. Configuring the catalyst switched port analyzer (SPAN) Cisco Tech Notes, 2003. Available from: http://www.cisco.com/warp/public/473/41.html.

Corey V, Peterman C, Shearin S, Greenberg MS, Van Bokkelen J. Network forensics analysis. IEEE Internet Comput 2002;6(6) Available from: http://www.sandstorm.net/downloads/netintercept/ni-ieee.pdf.

Garfinkel S. Network forensics: tapping the internet. O'Reilly Network. Available from: http://www.oreillynet.com/pub/a/network/2002/04/26/nettap.html.

IIT Research Institute. Draft report: independent technical review of the carnivore system, 2000. Available from: http://www.usdoj.gov:80/jmd/publications/carniv_entry.htm.

Venezia P. NetDetector captures intrusions InfoWorld, July 2003. Available from: http://www.infoworld.com/article/03/07/11/27TCniksun_1.html?s=tc.