

Digital Evidence



Harley Kozushko
Sunday, November 23, 2003

The last few years have seen a rise in the number of digital attacks. This rise has prompted the need for prosecution support. Intrusion Detection Systems have provided that prosecution support through the log files they produce which reflect the actions of the attacker. In some occasions these logs have even been accepted as direct evidence. Direct evidence is usually something tangible that is presented to prove a fact. This paper will serve as a reference guide toward the recovery, collection and preservation of digital evidence. In addition, a complete review of the forensic toolkit Penguin Sleuth Linux is attached in Appendix A.

Digital evidence encompasses any and all digital data that can establish that a crime has been committed or can provide a link between a crime and its victim or a crime and its perpetrator. Intrusion Detection Systems are a great source of digital evidence. They collect information from a variety of system and network sources then analyze the information for signs of intrusion and misuse. There are two types of Intrusion Detection Systems – Host-Based and Network-Based.

In the host-based intrusion detection architecture, the system is used to analyze data that originates on computers (hosts). Thus, this architecture is used for detecting insider attacks and misuse. For example, an employee who abuses their privileges, or students changing their grades. Host-based systems examine events like what files are accessed and what applications are executed. Logs are used to gather this event data. However, the audit policy is very important because it defines which end-user actions will result in an event record being written to an event log, for example, logging all accesses of mission-critical files. Host-based intrusion detection systems reside on every system and usually report to a central command console. To detect misuse, signatures, or predefined patterns of misuse are compared with the data from the log files. When there is a correlation, either the security administrator is notified of the potential misuse, or a predefined response to the misuse is enacted.

In the network-based intrusion detection architecture, the system is used to analyze network packets. Network-based architectures are used to detect access attempts and denial of service attempts originating outside the network. This architecture consists of sensors deployed throughout a network. These sensors then report to a central command console. Similar to host-based architectures, packet content signatures are used to identify misuse. These signatures are based on the contents of packets, headers and flow of traffic. However, it is important to note that encryption prevents detection of any patterns in the contents of the packet.

However, because network-based intrusion detection can relate information like IP addresses which can be spoofed, a valid user could be wrongly attributed to perpetrating the misuse of the system. This makes network-based intrusion detection data invalid, which also makes most, but not all network digital evidence invalid. However, host-based intrusion detection data can be valid digital evidence.

With a review of sources of digital evidence like intrusion detection systems follows an explanation of the procedures and issues surrounding the recovery, collection, and preservation of digital evidence.

Because data forensics is relatively new, laws dictating the validity of evidence are sketchy and not widely known. However, validity is essential to fully prosecute the misuser. So evidence has to come from the security administrator who must ensure the validity of that evidence. This means the security administrator has to know the rules that govern the admissibility of evidence in the United States.

All evidence must pass the test of admissibility and weight. Admissibility is a set of legal rules applied by a judge in order to allow the use of evidence in a court of law. These rules are extensive. Weight is a measure of the validity and importance of the evidence. Weight is essentially whether the judge or jury believes the evidence. There are few guidelines except what is convincing and well presented. Evidence must be authentic, accurate, and complete for it to pass any standard of weight.

Computer forensic evidence is just like any other evidence in the sense that it must be: authentic, accurate, complete, convincing to juries, and in conformity with common law and legislative rules (admissible). To ensure authenticity, the must be proven to come from where it purports. The evidence is accurate and reliable if the substance of the story the material tells is believed and is consistent, and there are no reasons for doubt. Evidence is complete if the story that the material purports to tell is complete. That is, there are no other stories that the material also tells that might have a bearing on the legal dispute or hearing.

Digital evidence can be classified, compared, and individualized in several ways. One of those ways is by the contents of the evidence. For example, investigators use the contents of an e-mail message to classify it and to determine which computer it came from. Also, swap files and slack space contain a random assortment of fragments of digital data that can often be classified and individualized. Another way to classify digital evidence is by function. This is when investigators examine how a program functions to classify it and sometimes individualize it. For example, a program that appears to do something amusing or useful but actually does something else is classified as a Trojan horse program. In addition, digital evidence can be classified by characteristics like file names, message digests, and date stamps.

There are five properties that evidence must have in order to be useful. The first property is that digital evidence has to be admissible. That is, evidence must be able to be used in court. Failure to comply with this rule is equivalent to not collecting the evidence in the first place, except the cost is higher. Another property is authenticity. Evidence must be tied to the incident in order to prove something. Moreover, the evidence must be shown to relate to the incident in a relevant way. Completeness is also another property that dictates the admissibility of digital evidence. It's not enough to collect evidence that just shows one perspective of the incident. Not only should evidence be collected that can prove an attacker's actions, but also evidence that could

prove their innocence. For instance, if it can be shown that the attacker was logged in at the time of the incident, it also needs to be shown who else was logged in or not so as to possibly prove the innocence of another person. This is called exculpatory evidence, and is an important part of proving a case. Another property is reliability. More specifically, the evidence collection and analysis procedures must not cast doubt on the evidence's authenticity and veracity. One more rule is that of believability. The evidence that is presented should be clearly understandable and believable by a jury. After all, there's no point in presenting a binary dump of process memory if the jury has no idea what it all means.

Using the preceding five rules, some basic do's can be derived. It is crucial that the handling and therefore the corruption of original data be minimized. Once a master copy of the original data is created, it should not be touched nor the original. Secondary copies should be handled instead. In addition, any changes made to the originals will affect the outcomes of any analysis later done to copies. For example, no programs that modify the access times of files should be run. Also, any changes should be accounted for, and detailed logs should be kept of the actions taken. However, when evidence alteration is unavoidable it is absolutely essential that the nature, extent, and reasons for the changes need be documented. Furthermore, as accurate an image of the system should be captured as possible. Capturing an accurate image of the system is related to minimizing the handling or corruption of the original data. Differences between the original system and the master copy count as a change to the data. These differences must be accounted for. Also, the person who collected the evidence should be prepared to testify. Without the collector of the evidence being there to validate the documents created during the evidence-collection process, the evidence becomes hearsay, which is inadmissible. In addition, it is very important to ensure the actions taken on the data are repeatable. If the actions can't be replicated and the same results can't be reached, those original actions won't be believed. This rules out any trial and error actions. Finally, it is vital to work fast. The faster the work takes place, the less likely the data is going to change. Volatile evidence may vanish entirely if it is not collected in time. Automation of certain tasks makes collection proceed even faster.

Using those same five rules, some don'ts can also be derived. The system should not be shutdown before collecting evidence. There is the possibility of loss of volatile evidence and the attacker may have trojaned the startup and shutdown scripts, Plug and Play may alter the system configuration and temporary files systems may be wiped out. Also, no programs should be run on the affected system. There is the possibility of inadvertently triggering something that could change or destroy evidence. Any programs used should be on read-only media and should be statically linked.

There is a general four step procedure to follow when collecting evidence. First the evidence has to be identified. Evidence must be distinguished from junk data. For this purpose what the data is should be known, where it is located, and how it is stored should also be known. Next, the evidence should be preserved. The evidence must be preserved as close as possible to its original state. Any changes made during this phase must be documented and justified. Also, all procedures used in the examination should

be audible, that is, a suitably qualified independent expert appointed by the other side of a case should be able to track all the investigations carried out by the prosecution's experts. Subsequently, all evidence must be analyzed. That is, the stored evidence must be analyzed to extract the relevant information and recreate the chain of events. Finally, the evidence must be presented. Communicating the meaning of the evidence is vitally important; otherwise you can't do anything with it. The manner of presentation is important, and it must be understandable by a layman to be effective.

Through every step of the procedure, it is crucial to record and document everything that is done and everything that is used. This ensures that the procedure is repeatable. However, these records and documents are extensive. Who initially reported the suspected incident along with the time, date, and circumstances surrounding the suspected incident should be recorded. Also, details of the initial assessment leading to the formal investigation should be recorded. In addition, it is important to record the names of all persons conducting the investigation. The case number of the incident and the reasons for the investigation should also be recorded. Furthermore, a list of all computer systems included in the investigation, along with complete system specifications should be recorded. Network diagrams and applications running on the computer systems previously listed should be recorded. Also, a copy of the policy or policies that relate to accessing and using the systems previously listed should be recorded. A list of administrators responsible for the routine maintenance of the system as well as a detailed list of steps used in collecting and analyzing evidence should be recorded. Finally, an access control list of who had access to the collected evidence at what date and time should be recorded.

With knowing the rules and guidelines surrounding the identification and collection of evidence, some collection steps can be introduced. First the evidence has to be found. This is a very dynamic step that changes on a case by case basis. However, it is helpful to use a checklist to double-check that everything that is being looked for is there. Next, the relevant data needs to be found out of the collected data. Once the data has been found, the part that is relevant to the case needs to be figured out. In general, err should be on the side of over-collection. After that an order of volatility needs to be created. Now that what to gather is known, a best order needs to be worked out in which to gather it. This ensures the minimized loss of uncorrupted evidence. Next, external avenues of change need to be removed. It is essential that alterations to the original data are avoided. Preventing anyone from tampering with the evidence helps you to create as exact an image as possible. Then it is time to collect the evidence. The evidence needs to be collected using the appropriate tools for the job. During the collection it is beneficial to re-evaluate the evidence that has already been collected. It may be the case that something important may be missed. Finally, everything should be documented. The collection procedures may be questioned later, so it is important that everything is documented.

With the evidence collected, before going any further, it is important to compare digital evidence with physical evidence. Digital evidence can be duplicated exactly and a copy can be examined as if it were the original. Also, examining a copy will avoid the risk of damaging the original. With the right tools, it is very easy to determine if digital evidence has been modified or tampered with by comparing it with the original. Digital evidence is relatively difficult to destroy. Even if it is “deleted”, digital evidence can be recovered. When criminal attempt to destroy digital evidence, copies can remain in places they were not aware of.

With the background on the guidelines and procedures of how to collect evidence, a more detailed description of collection can follow. The focus of digital evidence is on the contents of the computer as opposed to the hardware. With that in mind, there are two kinds of copies: copy everything, or just copy the information needed. When there is plenty of time and uncertainty about what is being sought, but a computer is suspected to contain key evidence, it makes sense to copy the entire contents. When collecting the entire contents of a computer, the general concept is the same in most situations. All related evidence should be taken out of RAM, and then the computer should be shutdown. Next, the hardware configuration of the system should be documented as well as the time and date of the CMOS. The computer should be booted using another operating system that bypasses the existing one and does not change data on the hard drive(s). Finally, a copy of the digital evidence from the hard drive(s) should be made. When collecting the entire contents of a computer, a bit stream copy of the digital evidence is usually desirable. In short, a bit stream copy copies what is in slack space and unallocated space, whereas a regular copy does not. Slack space is defined to be the unused space in a disk cluster. The DOS and Windows file systems use fixed-size clusters. Even if the actual data being stored requires less storage than the cluster size, an entire cluster is reserved for the file. This unused space is called the slack space.

Taking data collection one step further, an agenda for duplication and preservation can be described. There are many tools to make bit stream backups of hard disks and floppy disks. Among them are tools like Encase, dd, ByteBack, and SafeBack. It is always to record the tool that used to make the copy. When making the bit stream image, note and document how the image was created. It is desirable to also note the date, time and examiner.

However, the empirical law of digital collection and preservation states that if only one copy of the digital evidence is made, that evidence will be damaged or completely lost. Therefore, at least two copies of the evidence are taken. One of these is sealed in the presence of the computer owner and then placed in secure storage. This is the master copy and will only be opened for examination under instruction from the Court in the event of a challenge to the evidence presented after forensic analysis on the second copy.

Digging even further into collecting and preserving digital evidence, some programs for the tasks are found. Collecting evidence out of RAM in Unix machines could be a problematic task. Using the dd command an image of RAM could be created from the /dev/mem device. However, there has to be some way of knowing which programs are related, and to what areas of memory are these programs attributed to. This is where the lsof program comes into play. There is also the 'lsof' program that can provide a list of files and sockets that a particular program is running. As it pertains to the bit stream backup, investigators can use the 'dd' command. However, whenever digital evidence is copied onto a floppy disk, compact disk, tape, or any other form of storage media, an indelible felt-tipped pen should be used to label it with the following information: the current date and time and the date and time on the computer (any discrepancy should be documented), the initials of the person who made the copy, the name of the operating system, the program(s) and/or command(s) used to copy the files (copies of software used needs to be retained), and the information believed to be contained in the files.

Since the evidence has been collected, it is important to ensure the integrity of the evidence. That is, contamination needs to be controlled. Originals should never be used in forensic examination – verified duplicates should be used instead. Once data has been collected, it must be analyzed to extract the evidence to present and rebuild exactly what happened. Keep in mind that everything that is done must be fully documented, the work will be questioned and it must be shown that the results are consistently obtainable from the procedures performed. To reconstruct the events that led to the system being corrupted, a timeline must be created. Log files used time stamps to indicate when an entry was added, and these must be synchronized to make sense. Also keep in mind that the data on the backups has to also be free from contamination. So, when analyzing backups it is best to have a dedicated host for the job. This examination host should be secure, clean, and isolated from any network. Again, everything that is done needs to be documented to ensure that what is done is repeatable and capable of always giving the same results.

After the evidence has been collected, it needs to be preserved. Message digests are used for just this purpose. A message digest algorithm can be thought of as a black box, that accepts a digital object as input and produces a number as output. A message digest always produces the same number for a given input. Likewise, a good message digest algorithm will produce a different number for different inputs. Therefore, an exact copy will have the same message digest as the original but if a file is changed even slightly it will have a different message digest from the original. The MD5 algorithm can be used for calculating message digests. The algorithm uses the data in a digital object to calculate a combination of 32 numbers and letters. It is highly unlikely that two files will have the same message digest unless the files are duplicates. Message digests provide a method of near individualization and therefore, are sometimes referred to as digital fingerprints and are useful for determining if a piece of digital evidence has been tampered with. In essence, message digests speak for the integrity of the file.

As digital evidence pertains to host-based intrusion detection systems, the primary source of digital evidence is in log files. Under U.S. Code Title 28 Section 1732, log files are admissible as evidence if they are collected in the regular course of business. Rule 803(6) of the Federal Rules of Evidence states that logs, which might be considered hearsay, are admissible as long as they are collected in the course of regularly conducted business activity. This means that it is much safer to log everything all the time and deal with the storage issues, than to turn on logging only when an incident is suspected. Another factor in the admissibility of log files is the ability to prove that they have not been subjected to tampering. Whenever possible, digital signatures should be used to verify log authenticity. Other protective measures include storing logs on a dedicated logging server and/or encrypting log files.

Computer log files are created routinely and contain information about acts and events made at specific times by, or from information transmitted by, a person with knowledge. Some computer-generated information has been seen as so reliable that it has been accepted as direct evidence. Direct evidence is usually something tangible that is presented to prove a fact. It is important to keep these log files secure and to back them up periodically. Because logs are automatically time stamped, a single copy should suffice, although they should be digitally signed and encrypted when important, to protect them from contamination.

There are many different log files that are potential sources for digital evidence. Among them are acct, which contains every command typed by every user on the computer. Loginlog records failed logins. Syslog contains the main system log file that contains a wide range of messages from many applications. Sulog records every attempt to log in as the administrator of the computer. Utmp contains a record of all users currently logged into a computer. The 'who' command accesses this file. Wtmp contains a record of all of the past and current logins and records system startups and shutdowns. The 'last' command accesses this file. Finally xferlog contains a record of all files that were transferred from a computer using the file transfer protocol.

However, it is a lot more difficult to find digital evidence on the internet. For example, e-mail and internet mail can be used misused. First, a description of message transfer units is in order. They are computers which are the equivalent of post offices for electronic mail. The message transfer unit adds the current time and name of the message transfer unit along with some technical information to the header. This is called a received header. Therefore, to track an email message back to the sender, the route the e-mail traveled can be retraced by reading through the e-mail's received headers.

At the transport and network layer there are some services on the internet that can be helpful in learning more about the sender of an e-mail message. These are services like telnet and finger. The most direct method of finding contact information for a given host is to search the Whois database. (<http://whois.arin.net>) A program called traceroute provides a list of routers that information passes through to reach a specific host. This program is very useful for determining which computers were involved in the transport of information on the internet. This is important because intermediate routers sometimes

have relevant digital evidence in log files. And recall that the Whois database contains contact information for the people who are responsible for each router.

At this point the router to which first sent the e-mail has been identified. This may be the local ISP, but not the actual computer. To find out this information, the IP address has to be retrieved. This could be a problem because IP addresses could be spoofed. To combat IP spoofing two protocols are used, mainly bootstrap protocol (BOOTP) and dynamic host configuration protocol (DHCP). BOOTP and DHCP are quite similar. Both require hosts to identify themselves using a MAC address before obtaining IP addresses. When a computer boots up, it sends its MAC address to the BOOTP or DHCP server which recognizes the MAC address and sends back an IP address. The server can be configured to assign a specific IP address to a specific MAC address thus giving the effect of static IP addresses. The criminal could reconfigure his computer with someone else's IP address to hide his identity. Investigators can still identify the computer using the MAC address. The criminal's computer must use the address resolution protocol to device data from a router which requires the actual MAC address of the computer. The router would have an entry in its ARP cache showing a particular computer using someone else's IP address. Therefore, there is a good chance that the criminal's computer and the criminal himself will be located and caught.

In summary, data forensics is best applied to digital evidence that resides on host computers rather than on the network. Collection and preservation of this evidence is the key to its use in a court of law. A good forensic toolkit can provide the tools for this collection and preservation as well as analysis. And of course, the entire process needs to be documented extensively.

Whether it is evidence on the host computer or on the network, a collection of tools, called a digital forensics toolkit, can help in the automation of routine tasks and much more. This toolkit enables the collection and preservation of digital evidence. The rest of this paper is going to focus on the tools that make up the Penguin Sleuth toolkit (<http://www.linuxforensics.org>), complete with a review of each tool.

The Penguin Sleuth version of Linux was created to be a bootable version of Linux that can fit on a CD. After all evidence has been taken out of RAM, and the computer has been shutdown, this toolkit is used. The machine is then booted up using the CD, which is on only readable memory. Once it has been fully booted up, the tools which are included in the distribution can be used to analyze the suspected partition. It is from here where the bit stream copy of the partition can be made. The process of bypassing the host's operating system only to examine it in a read-only mode is called 'dead' analysis. The Penguin Sleuth version of Linux is used for this 'dead' analysis.

Appendix A – A Review of the Tools Available in the Penguin Sleuth Version of Linux

The sleuth kit – <http://www.sleuthkit.org> – is a new and improved Coroner's Toolkit. It is an open source forensic toolkit for analyzing Microsoft and UNIX file systems. After an image has been created from a partition using the dd command, the sleuth kit can go to work. It enables investigators to identify and recover evidence from images acquired during incident response or from live systems. There are four layers of functions which analyze the image file that are in the kit. First, the file system layer interacts with the file system the disk image was created in. Examples of files systems include, Extended 2 file system, file allocation table, and new technologies files system. One way to categorize and remember functions better is that any tool that begins with the letter 'f' is related to the file system. The content layer of a file system contains the actual file content, or data. Data is stored in large chunks, with names such as blocks, fragments and clusters. All tools in this layer begin with the letter 'd'. The metadata layer describes a file or directory. This layer contains descriptive data such as dates and size as well as the address of the data units. This layer describes the data in terms that the computer can process efficiently. The structure that the data is stored in has names such as inode and directory entry. All tools in this layer begin with the letter 'i'. Finally, the human interface layer allows one to interact with files in a manner that is more convenient than directly through the metadata layer. In some operating systems there are separate structures for the metadata and human interface layers while others combine them.

The Autopsy Forensic Browser – <http://www.sleuthkit.org/autopsy>, <http://autopsy.sourceforge.net/> - is a graphical interface to utilities found in the sleuth kit. It allows the allocated and deleted files, directories, data units, and metadata of file system images to be analyzed in a read-only environment. Images can be searched for strings and regular expressions to recover deleted material. It also allows one to create a detailed time line of the Modified, Access, and Changed times of files. Hash databases are used to identify if a file is known to be good or bad. Files can also be organized based on their file type – instead of just viewing them by directory listings. Autopsy will not modify the original images and the integrity of the images can be verified in Autopsy using MD5 values.

Foremost - foremost.sourceforge.net – is a Linux program to recover files based on their headers and footers. Foremost can work on image files, such as those generated by 'dd', or directly on a drive. The headers and footers are specified by a configuration file so you can pick and choose which headers you want to look for. A full description of foremost and the configuration file are included in its man page.

Glimpse - www.webglimpse.net – is a very powerful indexing and query system that allows a quick search through files. It can be used by individuals for their personal file systems as well as by large organizations for large data collections. Glimpseindex builds an index of all text files in the tree rooted at DIR. With it, glimpse can search through all files much the same way as agrep, except you don't have to specify file names and the search is fast.

Dcfldd is just another form of the 'dd' command.

Wipe - wipe.sourceforge.net – is a tool that securely erases files from magnetic media. Recovery of supposedly erased data from magnetic media is easier than what many people would like to believe. A technique called Magnetic Force Microscopy allows any moderately funded opponent to recover the last two or three layers of data written to disk. Wipe repeatedly overwrites special patterns to the files to be destroyed. In normal mode, 34 patterns are used. These patterns were recommended in an article from Peter Gutmann entitled "Secure Deletion of Data from Magnetic and Solid-State Memory". A quick mode allows you to use only 4 passes with random patterns, which is of course much less secure.

Etherape - etherape.sourceforge.net/ - is a graphical network traffic browser. Etherape displays network activity graphically. It uses GNOME libraries as its user interface, and libpcap, packet capture and filtering library. Featuring link layer, IP and TCP modes, it displays network activity graphically. Color coded protocols are also displayed. It supports Ethernet, FDDI, Token Ring, ISDN, PPP, and SLIP protocols. It can filter traffic to be shown, and can read traffic from a file as well as live from the network.

Fenris - razor.bindview.com/tools/fenris/ - is a tool that analyzes program execution. Fenris is a multipurpose tracer, GUI debugger, stateful analyzer and partial decompiler intended to simplify bug tracking, security audits, code, algorithm, protocol analysis and computer forensics - providing a structural program trace, interactive debugging capabilities, general information about internal constructions, execution path, memory operations, I/O, conditional expressions and much more. Because it does not require sources or any particular compilation method, this multi-component project can be very helpful for black-box tests and evaluations - but it will also be a great tool for open-source project audits, as an unmatched real-time reconnaissance tool - especially when sources are too complex or too badly written to be analyzed by hand in a reliable way and reasonable time. Fenris does not rely on GNU libbfd for any critical tasks, and because of that, it is possible and feasible to trace and analyze binaries modified to fool debuggers, crypted, or otherwise tweaked.

Honeyd - <http://www.citi.umich.edu/u/provos/honeyd/> - is a command line honeypot program. Honeyd creates virtual hosts for IP addresses matching the specific net. It can simulate any TCP and UDP service. It replies to ICMP echo requests. Currently, all UPD ports are closed by default and honeyd will reply with an ICMP unreachable port message if the configuration personality permits that. This enables a single host to claim addresses on a LAN for network simulation. The net argument may contain multiple addresses and network ranges. In order for honeyd to receive network traffic for IP addresses that it should simulate, it is necessary to either explicitly route traffic to it, use proxy arp or run arpd for unassigned IP addresses on a shared network.

Snort - <http://www.snort.org> - is an open source network intrusion detection system, capable of performing real-time traffic analysis and packet logging on IP net-works. It can perform protocol analysis, content searching/matching and can be used to detect

a variety of attacks and probes, such as buffer overflows, stealth port scans, CGI attacks, SMB probes, OS fingerprinting attempts, and much more. Snort uses a flexible rules language to describe traffic that it should collect or pass, as well as a detection engine that utilizes a modular plug-in architecture. Snort also has a modular real-time alerting capability, incorporating alerting and logging plug-ins for syslog, a ASCII text files, UNIX sockets, WinPopup messages to Windows clients using Samba's smbclient, database (Mysql/PostgreSQL/Oracle/ODBC) or XML. Snort has three primary uses. It can be used as a straight packet sniffer like tcpdump(1), a packet logger (useful for network traffic debugging, etc), or as a full blown network intrusion detection system. Snort logs packets in tcpdump(1) binary format, to a database or in Snort's decoded ASCII format to a hierarchy of logging directories that are named based on the IP address of the "foreign" host.

Dsniff - www.monkey.org/~dugsong/dsniff/ - is a command line network auditing and penetration testing tool. Dsniff is a collection of tools for network auditing and penetration testing. dsniff, filesnarf, mailsnarf, msgsnarf, urlsnarf, and webspay passively monitor a network for interesting data (passwords, e-mail, files, etc.). arpspoof, dnsspoof, and macof facilitate the interception of network traffic normally unavailable to an attacker (e.g, due to layer-2 switching). sshmitm and webmitm implement active monkey-in-the-middle attacks against redirected SSH and HTTPS sessions by exploiting weak bindings in ad-hoc PKI.

John the Ripper - www.openwall.com/john/ - is a password cracking tool. John can use a dictionary or some search patterns as well as a password file to check for passwords. John supports different cracking modes and understands many cipher text formats, like several DES variants, MD5 and blowfish. It can also be used to extract AFS and Windows NT passwords. To use John, you just need to supply it a password file and the desired options. If no mode is selected, john will try 'single' first, then 'wordlist' and finally 'incremental'. 'Single' means that John will use the rules from [List.Rules:Single]. 'Wordlist' means that John will use a user-defined word list. Finally, 'incremental' means that John will use the specified ~/john.ini definition.

Nikto - www.cirt.net/code/nikto.shtml - is a web server scanner. It scans for things like cookies, http ports, and possible CGI directories. It seems like nikto also has a mechanism of intrusion detection system evasion. There is little documentation on nikto, maybe the web page will be more helpful.

Nbtscan - www.unixwiz.net/tools/nbtscan.html - is a command line tool that scans for open NetBIOS namerservers. Put another way, nbtscan is a program for scanning IP networks for NetBIOS name information. It sends NetBIOS status query to each address in supplied range and lists received in human readable form. For each response host it lists the IP address, NetBIOS computer name, logged-in user name and MAC address.

Xprobe - www.sys-security.com - is a command line remote operating system fingerprinting tool.

Ngrep - www.packetfactory.net/projects/ngrep/ - is a command line network grep function. Ngrep strives to provide most of GNU grep's common features, applying them to the network layer. Ngrep will allow specification of extended regular expressions to match against data payloads of packets. It currently recognizes TCP, UDP, and ICMP across Ethernet, PPP, SLIP, FDDI, and null interfaces, and understands bpf filter logic in the same fashion as more common packet sniffing tools, such as tcpdump, and snoop.

Nemesis - www.packetfactory.net/Projects/nemesis/ - is a command line network packet injector. The nemesis project is designed to be a command line-based, portable human IP stack for UNIX/Linux. The suite is broken down by protocol, and should allow for useful scripting of injecting packet streams from simple shell scripts. Supported protocols include ARP, DNS, ICMP, IGMP, OSPF, RIP, TCP, and UDP.

Fragroute - monkey.org/~dugsong/fragroute/ - is a command line network intrusion testing tool. Fragroute intercepts, modifies, and rewrites egress traffic destined for the specific host, implementing most of the attacks described in the Secure Networks "Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection" paper of January 1998.

Ethereal - www.ethereal.com - is a graphical network analyzer program. It allows for interactive browsing of packet data from a live network or from a previously saved capture file. Ethereal's native capture file format is libpcap format, which is also the format used by tcpdump and various other tools. In addition, Ethereal can read capture files from snoop and atmsnoop, Shomiti/Finisar Surveyor, Novell LANalyzer, Network General/Network Associates DOS-based Sniffer (compressed or uncompressed), Microsoft Network Monitor, AIX's iptrace and Cinco Networks NetXRay.

Netcat - www.atstake.com/research/tools/network_utilities/ - is a command line tool to read and write over the network using the TCP or UDP protocol. It is designed to be a reliable 'back-end' tool that can be used directly or easily driven by other programs and scripts. At the same time, it is a feature-rich network debugging and exploration tool, since it can create almost any kind of connection you would need and has several interesting built-in capabilities. In the simplest usage, 'nc host port' creates a TCP connection to the given port and the given target host. The standard input is then sending to the host, and anything coming back across the connection is sent to the standard output. This continues indefinitely, until the network side of the connection shuts down. Note that this behavior is different from most other applications which shut everything down and exit after an end of file on the standard input. Netcat can also function as a server, by listening for inbound connections on arbitrary ports and then doing the same reading and writing.

Tcpdump - www.tcpdump.org/ - is a command line tool that dumps network traffic. Tcpdump prints out headers of packets on a network interface that match the Boolean expression. It can also be run with the -w flag, which causes it to save the packet data to a file for later analysis, and/or with the -b flag, which causes it to read from a saved packet file rather than to read packets from a network interface. In all cases, only packets that match expression will be processed by tcpdump. When tcpdump finishes it will report counts of packets received by filter, the meaning of this depends on the OS on which you're running tcpdump. Tcpdump will also report the number of packets dropped by the kernel. This is the number of packets that were dropped due to lack of buffer space, by the packet capture mechanism in the OS on which tcpdump is running.

Hping2 - www.hping.org - is a command line packet assembler and analyzer. Hping2 is a network tool able to send custom TCP/IP packets and to display target replies like ping programs do with the ICMP replies. Hping2 handle fragmentation, arbitrary packets body and size and can be used in order to transfer files encapsulated under supported protocols. There are many uses for hping2. Among them are: test firewall rules, advanced port scanning, test network performance using different protocols, packet size, TOS, and fragmentation, traceroute under different protocols, firewall-like usage, remote OS fingerprinting, TCP/IP stack auditing, and much more.

Ettercap - ettercap.sourceforge.net - is a command line sniffer, interceptor and logger, for Ethernet networks. It began as a sniffer for switched LAN, but during the development process it has gained more and more features that have changed it to a powerful and flexible tool for man-in-the-middle attacks. It supports active and passive dissection of many protocols and includes many features for network and host analysis. It has five sniffing methods. The first is IPBASED, where the packets are filtered matching IP:PORT source and IP:PORT destination. Next is MACBASED, where packets are filtered matching the source and destination MAC address. Then there is ARPBASED, which uses arp poisoning to sniff in switched LAN between two hosts. Next there is SMARTARP, which uses arp poisoning to sniff in switched LAN from a victim host to all other hosts knowing the entire list of hosts. Finally there is PUBLICARP, which also uses arp poison to sniff in switched LAN from a victim host to all other hosts. With this method the ARP replies are sent in broadcast, but if ettercap has the complete host list (on start up it has scanned the LAN) SMARTARP method is automatically selected, and the arp replies are sent to all the hosts but the victim, avoiding conflicting MAC addresses as reported by win2K. There are also numerous features that come with ettercap.

Openssh - www.openssh.com - is a secure remote connection utility. It is a free version of the SSH protocol suite of network connectivity tools that increasing numbers of people on the Internet are coming to rely on. Many users of telnet, rlogin, ftp, might note realize that their password is transmitted across the Internet unencrypted. Openssh encrypts all traffic to effectively eliminate eavesdropping, connection hijacking, and other network level attacks. Additionally, openssh provides a myriad of secure tunneling capabilities, as well as a variety of authentication methods.

Kismet - www.kismetwireless.net - is a graphical wireless network sniffing tool. Kismet is an 802.11b wireless network sniffer. It is capable of sniffing using almost any wireless card supported by Linux, including cards based on the Prism/2 chipset with the wlan-ng driver, cards based on the Lucent chipset, and a few other cards. Kismet supports logging to the wtapfile packet format and saves detected network information as plaintext, XSV, and XML. Kismet is capable of using any GPS supported by gpsd and logs and plots network data.

Airsnort - airsnort.shmoo.com - is a graphical wireless network intrusion tool. Airsnort is a wireless LAN tool which recovers encryption keys. Airsnort operates by passively monitoring transmissions, computing the encryption key, when enough packets have been gathered. 802.11b, using the Wired Equivalent Protocol (WEP), has many security flaws. Airsnort attempts to secure these vulnerabilities.

GPG - www.gnupg.org/ - is an encryption and signing utility. Basically the GNU version of PGP.

Openssl - www.openssl.org/ - is a cryptography toolkit implementing the Secure Sockets Layer and Transport Layer Security network protocols and related cryptography standards required by them. The openssl program is a command line tool for using the various cryptographic functions of openssl's crypto library from the shell. It can be used for: Creation of RSA, DH, and DSA key parameters, calculation of message digests, encryption and decryption with Cyphers, and SSL/TLS client and server tests.

Lsof - is a command line utility that lists all open files that a specific program is using. An open file maybe a regular file, a directory, a block special file, or a character special file, an executing text reference, a library, a stream or a network file. A specific file or all files in a file system may be selected by path.

Hunt - lin.fsid.cvut.cz/~kra/index.html - is a command line TCP/IP exploit scanner. Hunt is a program for intruding into a connection, watching it and resetting it. Note that hunt is operating on Ethernet and is best used for connections which can be watched through it. However, it is possible to do something even for hosts on another segment or host that is on switched ports.

Stunnel - stunnel.mirt.net - is an ssl connection package. The stunnel program is designed to work as SSL encryption wrapper between remote clients and local or remote servers. The concept is that having non-SSL aware daemons running on your system you can easily set them up to communicate with clients over secure SSL channels. Stunnel can be used to add SSL functionality to commonly used inetd daemons like POP-2, POP-3, and IMAP servers, to standalone daemons like NNTP, SMTP, and HTTP, and dial tunneling PPP over network sockets without changes to the source code.

Arpwatch - is a command line Ethernet monitor. Arpwatch keeps track for Ethernet/IP address pairings. It syslogs activity and reports certain changes via email. Arpwatch uses pcap to listen to arp packets on a local Ethernet interface.

Dig - is a command line tool for querying domain name servers. Dig (domain name groper) is a flexible tool for interrogating DNS name servers. It performs DNS lookups and displays the answers that are returned from the name server(s) that were queried. Most DNS administrators use dig to troubleshoot DNS problems because of its flexibility, ease of use and clarity of output.

Chkrootkit - www.chkrootkit.org - looks for signs of a root kit. It examines certain elements of the target system and determines whether they have been tampered with.

References:

Casey, Eoghan. Computer Evidence and Computer Crime: Forensic Science, Computers, and the Internet. Cambridge: Cambridge University Press, 2000.

Kurose James F. and Keith W. Ross. Computer Networking. United States of America: Pearson Education, 2002.

R. Rivest, "The MD5 Message Digest Algorithm,"RFC 1321, Apr. 1992.
<http://www.rfc-editor.org/rfc/rfc1321.txt>.

Vacca, John R. Computer Forensics Computer Crime Scene Investigation. Massachusetts: Charles River Media, 2002.

Also, every email address that was listed in the Appendix was used for a reference.