Global Information Assurance Certification Paper

Finding dsniff on your network
Richard Duffy
November 28, 2001

Overview

This paper covers some ways to detect dsniff and two of its utilities, arpspoof and macof, on a network. Arpspoof and macof tools were used with dsniff to determine if dsniff could be detected. The following programs were used to detect various aspects of dsniff: Arpwatch, ZoneAlarm, Antisniff and tcpdump. Our existing Fluke network test equipment was connected to the network to evaluate what indicators each could provide about dsniff and its tools.

Switched networks are no longer secure

For years hubs have been replaced with switches to improve network performance and security. The protection against packet sniffing that I assumed the switches provided can be easily circumvented by a widely distributed collection of tools known as dsniff. Two papers I found in the SANS reading room, "Introduction to dsniff" by Lora Danielle and "Penetration Testing with dsniff" by Christopher R. Russel, provide a good overview of dsniff and its uses. Since I had no knowledge of how dsniff would affect the network I decided to install and run dsniff on one of our test networks. This addressed local security issues and eliminated the possibility of taking down our production routers during the testing. Instead of using our assigned IP (Internet Protocol) range, I chose a range of private IP addresses. This allowed further testing so that the following questions could be answered: Are there any signs that would indicate that dsniff is running and your data is being sniffed? What steps should be taken to protect your network from having unauthorized users run dsniff to breach the network security?

Preparing to run dsniff

Many steps had to be taken just to get my system ready to install dsniff. My daily duties normally include configuring routers, troubleshooting network problems and installing new hardware. I had never before managed a Linux system or installed the operating system. I installed Red Hat 7.1 and disabled all the unnecessary services. I then retrieved the files needed to get dsniff and fragrouter up and running. The installation of dsniff and fragrouter was simplified by the step-by-step instructions by Lora Danielle in her paper mentioned above.

Written Permission

The GSEC course material repeatedly warned network administrators to secure written permission before scanning any network. Since lacking the proper permission can cause instant termination I had to investigate the actual policy at our University. As it turned out, the University's acceptable use policy states that network staff members in our department have permission to run packet sniffers as necessary.

Dsniff and associated programs

There are too many programs associated with dsniff to explore each in this paper, so my testing was limited to the following programs: arpspoof, dsniff, macof and fragrouter. I found fragrouter easy to use and help is readily available by typing **man fragrouter** or **fragrouter -help**. Below you will find a list of descriptions for the various dsniff programs. This text was taken from the dsniff "README" written by the author Dug Song.

Arpspoof - redirect packets from a target host (or all hosts) on the LAN intended for another local host by forging ARP replies. This is an extremely effective way of sniffing traffic on a switch. kernel IP forwarding (or a userland program which accomplishes the same, e.g. fragrouter :-) must be turned on ahead of time.

Dnsspoof - forge replies to arbitrary DNS address / pointer queries on the LAN. this is useful in bypassing hostname-based access controls, or in implementing a variety of man-in-the-middle attacks (HTTP, HTTPS, SSH, Kerberos, etc).

Dsniff - password sniffer. handles FTP, Telnet, SMTP, HTTP, POP, poppass, NNTP, IMAP, SNMP, LDAP, Rlogin, RIP, OSPF, PPTP MS-CHAP, NFS, VRRP, YP/NIS, SOCKS, X11, CVS, IRC, AIM, ICQ, Napster, PostgreSQL, Meeting Maker, Citrix ICA, Symantec pcAnywhere, NAI Sniffer, Microsoft SMB, Oracle SQL*Net, Sybase and Microsoft SQL auth info.

dsniff - automatically detects and minimally parses each application protocol, only saving the interesting bits, and uses Berkeley DB as its output file format, only logging unique authentication attempts. full TCP/IP reassembly is provided by libnids(3) (likewise for the following tools as well).

filesnarf - saves selected files sniffed from NFS traffic in the current working directory.

macof - flood the local network with random MAC addresses (causing some switches to fail open in repeating mode, facilitating sniffing). a straight C port of the original Perl Net::RawIP macof program.

mailsnarf - a fast and easy way to violate the Electronic Communications Privacy Act of 1986 (18 USC 2701-2711), be careful. Outputs selected messages sniffed from SMTP and POP traffic in Berkeley mbox format, suitable for offline browsing with your favorite mail reader (mail -f, pine, etc.).

msgsnarf - record selected messages from sniffed AOL Instant Messenger, ICQ 2000, IRC, and Yahoo! Messenger chat sessions.

sshmitm - SSH monkey-in-the-middle. proxies and sniffs SSH traffic redirected by dnsspoof(8), capturing SSH password logins, and optionally hijacking interactive sessions. only SSH protocol version 1 is (or ever will be) supported - this program is far too evil already.

tcpkill - kills specified in-progress TCP connections (useful for libnids-based applications which require a full TCP 3-whs for TCB creation).

tcpnice - slow down specified TCP connections via "active" traffic shaping. forges tiny TCP window advertisements, and optionally ICMP source quench replies.

urlsnarf - output selected URLs sniffed from HTTP traffic in CLF (Common Log Format, used by almost all web servers), suitable for offline post-processing with your favorite web log analysis tool (analog, wwwstat, etc.).

webmitm -HTTP / HTTPS monkey-in-the-middle. transparently proxies and sniffs web traffic redirected by dnsspoof(8), capturing most "secure" SSL-encrypted webmail logins and form submissions.
webspy - sends URLs sniffed from a client to your local Netscape browser for display, updated in real-time (as the target surfs, your browser surfs along with them, automagically). a fun party trick. :-)

Details on the commands used

Arpspoof
Issuing the command **arpspoof -t 10.10.30.0 10.10.30.1** from the system running the arpspoof tool resulted in all packets that were destined for the router 10.10.30.1 from all hosts on the network to be rerouted to the system running the command. By changing the target (-t) address to a single host address instead of the entire network range it will reroute the information from that single host to the system running arpspoof. See example 1 for a sample of the output of the arpspoof screen. Here, the system running arpspoof (MAC (Media Access Control) address 0:20:e0:69:a5:51) is sending out broadcast ARP (Address Resolution Protocol) replies stating it has IP address 10.10.30.1. Also captured on that same screen are the ARP replies sent by arpspoof as the program is terminated. To restore IP forwarding back to the actual router the arpspoof program broadcasts ARP replies using the router's actual MAC address (0:30:7b:81:aa:61).

Example 1 (Output from the arpspoof screen)

```
Arpspoof running -- MAC address of the spoofing box
0:20:e0:69:a5:51 ff:ff:ff:ff:ff:ff 0806 42: arp reply 10.10.30.1 is-at 0:20:e0:69:a5:51
0:20:e0:69:a5:51 ff:ff:ff:ff:ff:ff 0806 42: arp reply 10.10.30.1 is-at 0:20:e0:69:a5:51
0:20:e0:69:a5:51 ff:ff:ff:ff:ff:ff 0806 42: arp reply 10.10.30.1 is-at 0:20:e0:69:a5:51
Arpspoof stopped -- actual router MAC address
0:30:7b:81:aa:61 ff:ff:ff:ff:ff:ff 0806 42: arp reply 10.10.30.1 is-at 0:30:7b:81:aa:61
0:30:7b:81:aa:61 ff:ff:ff:ff:ff:ff 0806 42: arp reply 10.10.30.1 is-at 0:30:7b:81:aa:61
0:30:7b:81:aa:61 ff:ff:ff:ff:ff:ff 0806 42: arp reply 10.10.30.1 is-at 0:30:7b:81:aa:61
```

Fragrouter
Warning, before running arpspoof, you must start a program to forward packets from the spoofing system back to the real router. The command **fragrouter –B1** will forward packets to the original router.

Tcpdump
Running the tcpdump program will allow you to see communications of other systems on the switched network and determine if arpspoof is working. The command **tcpdump –n** will send the data to the screen. The –n option tells tcpdump not to convert IP addresses or port numbers to names. Another command, **tcpdump –n –w filename** will write the data to the specified file. All the data goes directly to the file not to the screen. The file can be read with the command **tcpdump –n –r filename**. Note that tcpdump only sees half of the conversation (the local side)

so if you want to see all of a conversation then monitor the switch port to the router.

Macof
The command **macof** will start flooding MAC and IP addresses to the network. At this point, some switches were overwhelmed with the load and were difficult to communicate with until the macof stream was stopped.

Dsniff
Dsniff can be run with the command **dsniff –c –m**. This command seemed to work best for my testing. The man page for dsniff explains all the flags but the –c tells dsniff to do half duplex on TCP (Transmission Control Protocol) reassembly. The –m flag enables automatic protocol detection. Captures of the dsniff screens for a telnet, ftp and a web login are found in example 2. Notice the port numbers after the IP addresses for the source and destination.

Example 2 (dsniff capture)

```
----------------
11/19/01 14:36:50 tcp 10.10.30.13.2247 -> 10.10.31.64.23 (telnet)
rduffy
y!n_g%I!
exit
----------------
11/19/01 14:37:17 tcp 10.10.30.13.2248 -> 10.10.31.64.21 (ftp)
USER rduffy
PASS y!n_g%I!
----------------
11/19/01 14:37:51 tcp 10.10.30.13.2271 -> 10.10.32.101.80 (http)
GET / HTTP/1.0
Host: 10.10.32.101
Authorization: Basic c2VjdXJpdHk6cjEwZ3JBbmRF [rduffy:y!n_g%I!]
```

Some ways to detect dsniff

To minimize the costs existing test tools were used to detect dsniff. Our network consists of about 75 subnets segmented out of our class B address space. It is impractical to install permanent equipment on every subnet to detect the presence of dsniff. My first goal was to identify the indicators of dsniff running on the network using our standard test equipment while in the field. The second goal was to determine what information we could get out of our switches or routers that would help indicate dsniff is on the network. The third goal was to find other inexpensive tools to detect dsniff. We commonly use the following network test equipment: Fluke OptiView, One-Touch Series II, and the 683 Enterprise LANMeter. The other test tools I used for dsniff detection were Arpwatch, ZoneAlarm 2.6 Pro and AntiSniff version 1 by L0pht Heavy Industries.

Desktop Detection with ZoneAlarm

One inexpensive and good indicator that dsniff is on the network is to install and properly configure ZoneAlarm on the pc's within the network. Although any personal firewall can cause problems accessing certain programs and network resources, it provides excellent protection for the pc at a very low cost. If users remember to disable ZoneAlarm when troubleshooting application and network problems it should reduce the impact on help desk trouble calls. Considering how many problems ZoneAlarm will stop, it is well worth the small inconvenience. ZoneAlarm 2.6 Pro installs with local network protection set to medium and the Internet protection set to high. I found these settings to work well in most cases. After the installation you can define the systems trusted by your system by host IP address, IP range, IP subnet or domain name. Properly configuring trusted hosts or IP ranges reduced the false alarms considerably.

ZoneAlarm did not flag arpspoof running on the network, but it did prevent it from spoofing the default gateway for the specific system running ZoneAlarm. The reason for this is the spoofing system tried to communicate with the system running ZoneAlarm. ZoneAlarm blocked these packets since the arpspoof system was not listed as a trusted host in the ZoneAlarm configuration table. Zone Alarm is very helpful in detecting the macof program. Macof is used to flood a switch's forwarding table with MAC addresses until the switch cannot determine which MAC address is on each port. This often results in the switch sending packets to all ports just like a hub. This provides for easy sniffing of what should be "secure" data on a switch. In addition to sending out MAC addresses macof also sends out random IP addresses. ZoneAlarm logs the many attempts for random systems trying to talk to each other, none of which belong on the subnet. This is a huge clue that something is very wrong! For an example of the random IP addresses that ZoneAlarm logged see the short sample of the ZoneAlarm error log in example 3.

Example 3 (ZoneAlarm log. Note: none of these addresses belong on my local subnet, 10.10.30.0)

```
FWROUTE,2001/11/09,11:29:07 -7:00 GMT,70.35.0.91:2574,78.239.172.48:13766,TCP (flags:S)
FWROUTE,2001/11/09,11:29:07 -7:00 GMT,248.198.235.26:7780,37.177.83.50:29599,TCP (flags:S)
FWROUTE,2001/11/09,11:29:07 -7:00 GMT,104.4.35.97:15198,69.134.238.14:26626,TCP (flags:S)
FWROUTE,2001/11/09,11:29:07 -7:00 GMT,163.204.5.28:28599,183.214.3.124:11393,TCP (flags:S)
FWROUTE,2001/11/09,11:29:07 -7:00 GMT,49.106.45.94:24203,162.145.188.28:13002,TCP (flags:S)
FWROUTE,2001/11/09,11:29:07 -7:00 GMT,61.70.16.78:26786,41.10.170.93:32061,TCP (flags:S)
FWROUTE,2001/11/09,11:29:07 -7:00 GMT,104.84.224.75:31877,98.33.206.91:11554,TCP (flags:S)
FWROUTE,2001/11/09,11:29:07 -7:00 GMT,39.53.134.16:50324,189.96.47.35:44337,TCP (flags:S)
```

Promiscuous mode detection using AntiSniff

Antisniff is a program by L0pht Heavy Industries that runs on Windows 95/98 or NT systems. It will also run on Linux, but it is my understanding that the GUI (Graphical User Interface) is not available. For this test I installed it on an old Windows 98 system. Antisniff is designed to detect NICs (Network Interface Cards) running in promiscuous mode. Promiscuous mode allows a network card to listen and receive network traffic not destined for that system. The Antisniff package also has the capability of sending emails as alerts are triggered. This program takes little effort to run and provides helpful information that someone may be sniffing traffic on the network. A license for 95/98 or NT costs about $350.00. You can download AntiSniff free for a

fifteen-day trial before purchasing the product. In order for a system to use the dsniff tools the NIC must be in promiscuous mode. Knowing a system is in promiscuous mode can give you the information to track down the offending system. By using the MAC address of the offending system, search the switch's forwarding tables until you find at which port the MAC address is located. At this point you could disable the switch port or manually track the system from the port to the end location. Keep in mind not every system that is in promiscuous mode is sniffing data from the network but it is always a good practice to know why the system is in promiscuous mode.

Running Arpwatch

Arpwatch was part of the Red Hat 7.1 installation. It comes in handy to watch the network for arpspoofing. It will detect a given IP address and MAC address pair changing to another MAC address. This happens during arpspoofing of the default gateway or a singe host. The command **arpwatch –dN** seemed to work well. (See the man pages for more descriptions of all command options). Example 4 shows the arpwatch screen after a single MAC address change of a given IP address. This is extremely helpful since the router MAC address should not change without changing router hardware. Arpwatch will also detect new systems on the network, which is beneficial to know if you did not add any systems. Another thing arpwatch finds is duplicate IP addresses. When it finds a duplicate, it logs the old MAC address and the new MAC address.

Example 4 (Arpwatch showing changing MAC address changing for a given the same IP)

```
From: arpwatch (Arpwatch)To: root Subject: flip flop
          hostname: <unknown>
        ip address: 10.10.30.1
   ethernet address: 0:20:e0:69:a5:51
    ethernet vendor: PreMax PE-200 (NE2000-clone card, sold by InfoExpress)
old ethernet address: 0:30:7b:81:aa:61
 old ethernet vendor: <unknown>
         timestamp: Tuesday, November 13, 2001 11:22:41 -0500
  previous timestamp: Tuesday, November 13, 2001 11:22:40 -0500
            delta: 1 second

From: arpwatch (Arpwatch) To: root Subject: flip flop
          hostname: <unknown>
        ip address: 10.10.30.1
   ethernet address: 0:30:7b:81:aa:61
    ethernet vendor: <unknown>
old ethernet address: 0:20:e0:69:a5:51
 old ethernet vendor: PreMax PE-200 (NE2000-clone card, sold by InfoExpress)
         timestamp: Tuesday, November 13, 2001 11:22:48 -0500
  previous timestamp: Tuesday, November 13, 2001 11:22:47 -0500
            delta: 1 second
```

Field Detection using the Fluke 683 Enterprise LANMeter

The Fluke 683 Enterprise LANMeter will assist in detecting macof. This meter has an IP address conversation matrix that will show the IP addresses of the top conversations. Normally this table will include local IP addresses paired with the IP address of the destination machine. If macof is running the table will be full of conversations between pairs of IP's not belonging on the local net. Another indicator of macof is the meter limit of 512 conversations will almost immediately be reached. A warning will be displayed indicating the limit has been reached. This would be normal on very few networks within our campus. Knowing the normal baseline for each subnet is a big advantage when testing during network problems. Although LANMeter reliably detects duplicate IP addresses, while arpspoof is running, it did not detect duplicate IP or notice the MAC address was changing from the actual router to the system running arpspoof.

Field Detection using the Fluke OneTouch Series II Network Assistant

The OneTouch has limited capability to detect macof. If macof is running the meter will indicate the station list is greater than 500. If you look at the list you will find the majority of the IP addresses don't belong on that subnet. The OneTouch did not notice the MAC address change from the actual router to the MAC of my system running arpspoof and spoofing the router address. OneTouch saw the actual router but not my system running fragrouter as a second router. This is because the system running fragrouter doesn't respond to SNMP and doesn't send OSPF packets.

Field Detection using the Fluke OptiView Integrated Network Analyzer

The OptiView does detect the IP address of the router changing from one MAC address to another while arpspoof is running. It flags the IP that has the problem showing the conflicting MAC address for the same IP. This meter gathered information from the network devices under test and seemed to detect more problems if it is on the network for a while. Since the OptiView can be remotely controlled it is easy to connect to a remote network and randomly check for problems. If macof is running on the network the OptiView can show a list of MAC addresses that it sees. It was noticeable that there were more than 16,000 local MAC addresses found. Since our campus has a total of approximately 16,000 unique MAC addresses there should not be 16,000 MAC addresses on a single subnet. It is difficult to determine which MAC addresses belong on the subnet so running the TCP capture is very helpful. The TCP screen showed thousands of IP addresses that did not belong on the local subnet. This meter has a layer 2 traceroute that is just great. If you have the MAC address of the system you want to find, it can be entered and traced by the meter. It will trace from the meter through all the switches showing port numbers and switch IP address all the way to the system you want to find. This tool can save hours locating systems. If you have ever had to check every switch on a large subnet to find a single MAC address you will appreciate this tool. Note the meter does have to be attached to the same local subnet as the MAC address that you're searching for.

Indicators of dsniff on the switch

Normally the traffic activity lights on most switches are not lit solid. This may be an indication of things such as a broadcast storm or loop in the network. While macof is running the traffic activity lights are lit solid on all connected ports. The other thing to look at is the forwarding table on the switch. The 3COM 3300 switch had 12,000 entries in the forwarding table when macof was run for a while. There were eight devices connected to the switch that showed 12,000 MAC addresses. On a HP 4000 switch I was never able to find the maximum number of MAC addresses allowed in the forwarding table. The forwarding table had many pages of addresses on the port that was connected to the system running macof.

Indicators of dsniff on the router

During the testing my original thought was that there would be entries for each of the bogus MAC addresses and IP addresses in the router's ARP table while macof was running. This was an incorrect assumption on my part since there were no incorrect addresses in the ARP table. I believe that since the router never forwarded any packets off the subnet for the incorrect IP and MAC addresses, it does not enter them in the ARP table. The next thing I checked was the Cisco router's **debug ARP** command. I found there were some definite indicators listed by debug. The **debug ARP** command is a useful tool but can be very CPU intensive and should be used with caution. With macof running, **debug ARP** output showed pages of "failed to create incomplete entry for IP address xxx.xxx.xxx." These IP's should not be on the subnet so that is also a good indicator that dsniff may be running on the network. A sample of the debug ARP while running macof is shown in example 5. The **debug ARP** command is also capable of detecting arpspoof, as it shows packets with the router's IP address and another systems MAC address. See example 6 for a sample of the router's debug ARP output.

Example 5 (Output from debug ARP on the router while macof was running)

```
24w6d: IP ARP: failed to create incomplete entry for IP address: 127.221.127.112 interface EOBC0^M
24w6d: IP ARP: failed to create incomplete entry for IP address: 127.158.83.62 interface EOBC0^M
24w6d: IP ARP: failed to create incomplete entry for IP address: 127.243.95.117 interface EOBC0^M
24w6d: IP ARP: failed to create incomplete entry for IP address: 127.105.248.91 interface EOBC0^M
24w6d: IP ARP: failed to create incomplete entry for IP address: 127.38.42.110 interface EOBC0^M
24w6d: IP ARP: failed to create incomplete entry for IP address: 127.53.51.53 interface EOBC0^M
24w6d: IP ARP: failed to create incomplete entry for IP address: 127.19.150.93 interface EOBC0^M
24w6d: IP ARP: failed to create incomplete entry for IP address: 127.64.78.115 interface EOBC0^M
24w6d: IP ARP: failed to create incomplete entry for IP address: 127.47.231.115 interface EOBC0^M
24w6d: IP ARP: failed to create incomplete entry for IP address: 127.222.144.67 interface EOBC0^M
24w6d: IP ARP: failed to create incomplete entry for IP address: 127.166.10.25 interface EOBC0^M
24w6d: IP ARP: failed to create incomplete entry for IP address: 127.200.22.117 interface EOBC0^M
24w6d: IP ARP: failed to create incomplete entry for IP address: 127.135.188.124 interface EOBC0^M
```

Example 6 (Output from debug ARP on the router while arpspoof was running)

```
24w6d: IP ARP: rcvd rep src 10.10.30.1 0020.e069.a551, dst 10.10.30.0 Vlan10
24w6d: IP ARP: rcvd rep src 10.10.30.1 0020.e069.a551, dst 10.10.30.0 Vlan10
24w6d: IP ARP: sent rep src 10.10.30.1 0030.7b81.aa61,
24w6d: IP ARP: rcvd rep src 10.10.30.1 0020.e069.a551, dst 10.10.30.0 Vlan10
```

```
24w6d: IP ARP: rcvd rep src 10.10.30.1 0020.e069.a551, dst 10.10.30.0 Vlan10
24w6d: IP ARP: sent req src 10.10.30.1 0030.7b81.aa61,
24w6d: IP ARP: rcvd rep src 10.10.30.1 0020.e069.a551, dst 10.10.30.0 Vlan10
24w6d: IP ARP: sent req src 10.10.30.1 0030.7b81.aa61,
24w6d: IP ARP: rcvd rep src 10.10.30.1 0020.e069.a551, dst 10.10.30.0 Vlan10
24w6d: IP ARP: rcvd rep src 10.10.30.1 0020.e069.a551, dst 10.10.30.0 Vlan10
```

Stopping dsniff

I did check for ways to keep arpspoof from being able to spoof the router, but I was unsuccessful in preventing this from happening. Since arpspoof sends out ARP replies associating the router's IP address with the MAC address of the spoofing system I don't think there is a way you can stop the spoofing. There are a few ways to successfully detect the presence of macof and arpspoofing. If nothing else, Arpwatch is a simple program to implement at a very low cost. Having a good baseline for your network makes it easier to troubleshoot when your network is experiencing unusual activity.

Securing your data

Encrypting data will reduce the threat of someone running dsniff on your network. One of the best ways of securing your data is to use a VPN (Virtual Private Network). Since the data is encrypted from your system until it reaches the VPN concentrator in the event your data is captured at the local gateway the encryption will provide sufficient protection. It is not impossible to capture and decrypt the data but it does make it very difficult. Using SSH, a secure telnet connection is also a good idea. There are ways that one of the dsniff tools can still capture your data if your SSH is not setup properly but your data is more secure using SSH than applications that send plain text data.

Conclusion

The two dsniff utilities arpspoof and macof cannot be stopped but there are many ways to help protect systems and monitor the network for dsniff. By knowing what to look for while conducting network diagnostics with standard test equipment dsniff can be detected, traced to the offending system and stopped. Daily monitoring of your network with arpwatch will provide valuable details in detecting dsniff. Data encryption through the use of a VPN will provide a layer of protection in case dsniff is not immediately discovered. Documentation and a thorough understanding of what is "normal" for each subnet is invaluable in troubleshooting network abnormalities.

.

References

Song, Dug. “dsniff.” V2.3. 17 Dec. 2000. URL:
http://www.monkey.org/~dugsong/dsniff/ (5 Nov. 2001).

Danielle, Lora. “Introduction to dsniff.” 1 Jun. 2001. URL:
http://www.sans.org/infosecFAQ/audit/dsniff.htm (5 Nov. 2001).

Russel, Christopher. “Penetration Testing with dsniff.” 1 Feb. 2001. URL:
http://www.sans.org/infosecFAQ/threats/dsniff.htm (5 Nov. 2001).

Dunston, Duane. “Network Monitoring with Dsniff.” 29 May 2001. URL:
http://www.linuxsecurity.com/feature_stories/feature_story-89.html (5 Nov. 2001).

McClure, Stuart. & Scambray, Joel. “Switched networks lose their security advantage due to packet-capturing tool.” 29 May 2000. URL:
http://www.infoworld.com/articles/op/xml/00/05/29/000529opswatch.xml (5 Nov. 2001).

Loeb, Larry. “On the lookout for dsniff:.” Part 1. Jan. 2001. URL:
http://www-106.ibm.com/developerworks/library/s-sniff.html (5 Nov. 2001).

Silverman, Richard. “dsniff and SSH.” 22 Dec. 2000. URL:
http://sysadmin.oreilly.com/news/silverman_1200.html (5 Nov. 2001).

Edwards, Mark. “Think You're Safe from Sniffing?” 1 Jun. 2000. URL:
http://www.secadministrator.com/Articles/Index.cfm?ArticleID=8878 (5 Nov. 2001).

Fluke Corporation URL:
http://www.fluke.com/ (26 Nov. 2001).

Zone Alarm URL:
http://www.zonealarm.com/ (26 Nov. 2001).

Antisniff URL:

http://www.securitysoftwaretech.com/antisniff/ (26 Nov. 2001).

Arpwatch URL:
http://rpmfind.net/linux/RPM/Red_Hat_Linux.html (26 Nov. 2001).