

Chapter -3

Existing Sniffer Detection Methods

In this chapter we have described in detail the existing methods of sniffer detection; as in the first chapter we have given details about sniffers.

3.1 Introduction :

Ethernet networks are shared communication channels; the network interface of a computer on this type of network can see all the packets transmitted on the segment it present on. Every packet has a header telling the receiver of the packet. Under usual operating events, only the machine with that proper address is supposed to allow the packet. However, the Network Interface Card (NIC) can be set to a particular mode called **promiscuous mode** to receive all packets on the wire. A sniffer is a particular program or piece of code that keeps the NIC in the promiscuous mode. As explained in the introduction, sniffers are one of the first legal tools that allowed system administrators to analyze their network and do troubleshoot the network problems. But, crackers have uses sniffers maliciously to spy on networks and take various kinds of data. Crackers install sniffers to get user names, passwords, credit card numbers, personal information, and other relevant information that could be destructive to a person, corporation or even nation. When they obtain this information, crackers will use the passwords to attack further sites and they can even turn a profit from selling credit card numbers. Sniffer software can be downloaded from the Internet easily. The most popular sniffers are: Wireshark, Tcpdump [1], Sniffit [2], Ethereal [3], under Unix-like platform and Analyzer [4], Windump [5]. At SecurityFocus.com, there more worth of sniffer tools. Here we should note that the belief that one can be protected from sniffers by using switches is no longer factual [6].

3.1.1 Promiscuous mode

As it was stated in above, on a shared type network all hosts receive all the traffic sent to the network, and it is the host's job and responsibility to sort out the messages of its own. The Network Interface Cards itself decides if a frame is addressed to it, and only if it finds this to be the case will the frame be forwarded to the operating system. When the NIC is set into another operating mode, called 'promiscuous' mode by its driver then there is

chance of sniffing. A NIC which is set to promiscuous mode will forward all the received Ethernet frames to the operating system. This way every frame will be processed by the operating system, not only the ones which are meant to be sent to that host, so promiscuous mode is essential for sniffing. Without promiscuous mode, the NIC itself rejects the frames with other destination addresses, so the user level application would not have a chance to listen to network traffic between other hosts. The network code of the kernel also performs sorting, and rejects frames with wrong addresses. So we have a so-called hardware filtering done by the NIC itself and a higher level filtering, or so called software filtering by the kernel. According to Mumatz & Yasir [7], software filtering is varied by kernel to kernel so different operating systems and even different kernel versions may act differently regarding this matter. This is an important issue regarding the working principles of sniffer a detector. Sniffing can be defined as a case when a user space application receives and processes frames sent to other host(s) on the network. Every data that is sent or received by a host in plain text can be read. Some examples: By reading the HTTP request messages we can track what web pages a host visits. Telnet, POP3 and FTP passwords and logins are in plain text, so if a host logs on to any of these services we can get the login instantly. Normally, emails are not encrypted, so every time a host downloads emails we can simply read them. These are just a few examples, but it is clearly shown that sniffing has a huge impact on privacy and secrecy on the network. Following are common methods used for sniffer detection.

3.2 Sniffer detection methods

As we cannot guess what a host does with the packets it receives (reads and analyses them or drops them), we think about a host as a sniffing host, if we found its NIC in promiscuous mode, then we found that the host alters the network in some way to route packets to itself. These two situations normally do not happen in a network, so if we find this, we have a good reason to suspect the presence of the act of malicious sniffing. Therefore detecting sniffers can be come up to from two directions. One is to find a NIC in promiscuous

mode; another is to find out whether a host has altered the network. Already we have classified sniffers as active or passive sniffers in first chapter. Passive sniffers work on shared mediums and do not work on switched environment. In this work the terms 'sniffer' and a 'host with its NIC in promiscuous mode' are equal.

3.2.1 Ping method

Ping is an ICMP [10] (Internet Control Message Protocol) echo request/echo reply packet couple. As the networking program of a host operating system receive an echo request, it reply to it with an echo reply packet. When we ping to the host with a ping application command, we normally give the destination IP address. The assembly of the ICMP packet and the Ethernet frame will be handled by the operating system. Usually the destination address of the Ethernet frame that encapsulates the ICMP packet determined by the ARP(*Address Resolution Protocol*) protocol. This ensures that the frame, the echo request travels in will have its MAC address and IP address consistent, and ultimately the Ethernet frame will travel to the host that we want to ping. Promiscuous hosts are not concerned with MAC(media access control) addresses. The NIC will forward each Ethernet frame to the operating system. The NIC will make sure neither the destination address field of the frame, nor the content. This way all echo request ICMP packets will be forwarded to the operating system, apart from the Ethernet destination field of that request. The ping method, as sniffer detection method can be summarize as this: Creates a ping packet with the suspected host's IP address, and puts it into an Ethernet frame, that has some non-existent destination addresses, and then waits for reply. In order to perform that, we fake an Ethernet frame with mismatching MAC address and IP address. We do not rely on ARP; we forge our own Ethernet frame in our way. If the NIC is in promiscuous mode, it will not filter out this frame and it will pass the payload to the operating system. From the operating system's perspective this is a normal ICMP echo request packet with the IP address, so as every well-bred operating system, it will send back an echo reply message. If we receive an echo reply for an echo request which was created with the wrong MAC address on purpose,

we can be sure, that the replying host's NIC is in promiscuous mode, and as likely sniffs the network [8].

3.2.2 ARP method

The ARP method is very alike to the ping method. The main difference is that we use ARP packets, not ICMP packets. We modify Ethernet destination field the similar way and with the similar purpose. ARP request messages are normally sent out to broadcast addresses, since the reason is to locate the corresponding MAC address for an IP address. Generally it happens if we send out a fake ARP request message that has some random non-existent MAC destination, every normal host's NIC will drop this packet like they do that with every packet meant to another address. However, the NIC in promiscuous mode will accept this packet and it will forward it to the operating system. From the operating system's point of view it is an ARP request packet, and again as every well-bred operating system it will send back an ARP reply message. If we notice this behavior, we know that the host's NIC is in promiscuous mode. During practice, the operating system has a software level filter built into their networking program, and will not answer blindly each packet from the NIC. This execution varies from OS to OS. As the experiments of Abdelallah Elhadj et. al. [6] and Mumatz & Yasir [4] and Sanai [9] shows, it became very important what MAC address we put into the Ethernet frame.

- **Detecting Promiscuous Node Using ARP Packets**

In the local network, the sniffing has been a large threat. Malicious users can easily steal secret documents and anyone's privacy through sniffing the network. Sniffing causes interruption into privacy, and it can be done simply by downloading free sniffer software from the Internet and installing them into their personal computer. However, so far there is no good way to detect which PC's are sniffing the network. This part will discuss the use of Address Resolution Protocol (ARP) packets to efficiently detect malicious users when they are sniffing the local area networks.

- **The regular sniffing**

The local network is typically composed of the Ethernet. On an Ethernet using IP protocol, information is sent in plain text, except an encryption program is used. When someone sends information on the network, he expects someone on the other side of the network to receive that information. Unluckily, the mechanism of Ethernet allows unauthorized people an opportunity to steal and look at the data. We know that an Ethernet based network mechanism by sending packets to all nodes on the network, and it expects that only the intended node(s) will receive the packets. At the same time, the other nodes just drop the messages. Whether to receive or drop the messages is decided by the Network Interface Card (NIC). The NIC does not receive all the packets on the network although it is connected to the Ethernet; instead it filters out the desired packets, which this specific computer should receive. For the rest of this document, we will call the filter of the NIC the Hardware Filter. Sniffing is completed by setting the NIC of its own PC to an explicit mode, such that the NIC will receive all data incoming to it, no matter whether it is the planned destination. This NIC mode is called as the Promiscuous Mode.

- **Basic Concepts of Promiscuous Node Detection**

Instead of sending out unlawful packets, network sniffing is completed by receiving all packets. Since it does not disturb the network traffic at all, it is not easy to detect such behavior. However, the state of the NIC in promiscuous mode is clearly different from that in normal mode. A packet that is supposed to be filtered by the hardware filter is now moved to the system kernel. As a result, whether to respond to the packet depends totally on the internal software. Our way to detect promiscuous node can be demonstrated by an example from the real world. As the sniffing node receives all the packets, including those that are not targeting to it, it may make mistakes such as responding to a packet, which originally is supposed to be filtered by the NIC. Therefore, our promiscuous node detection is performed by checking the responses of ARP packets, when ARP request packets are sent to all nodes on the network [9].

3.2.3 ARP poisoning

A sniffer might send out fake ARP reply packets to switches which will cause traffic to be aimed at it this is called APR poisoning, and as such it is a dynamic sniffing technique. Khcherif [11] described this method. However, ARP poisoning can be used to locate sniffers, and this is a method for sniffer detection that works in a switched environment. The working principle is to send out fake ARP reply packets with the intention of ARP poisoning. The ARP reply packet contains a fake IP address and MAC address. We send out this packet in an Ethernet frame that has a few fake MAC destination address in order to be received only by hosts in promiscuous mode. This way the sniffer will think that the fake IP address communicates to our host. If we send out fake ARP reply packets not to the broadcast address but to some random non-existing address, only the sniffing host will accept that ARP reply packet and so, only the sniffing host ARP cache will be poisoned. Then establish a TCP connection and sniff the network to see if the host transmits based on the faked entry.

3.2.4 Latency method

This method depends on very simple assumption that as a sniffer processes all the network traffic, it causes a notable workload to that machine. According to Abdelallah Elhadj et. al. [9] When an NIC is in promiscuous mode, all Ethernet traffic will generate hardware interrupts which will cause the Ethernet driver code to execute. Furthermore, with a sniffer running, captured packets must be passed to the user code running the sniffer. Crossing the kernel boundary is widely known to be somewhat expensive. Therefore, under heavy traffic, a sniffer will heavily degrade performance on promiscuous host. This method works by pinging the suspected machine and measure the average ping times. Then we overflow the network with messages that would normally not pass by the suspected NIC, and directly ping it again. If the NIC is in promiscuous mode, the traffic will cause a little workload to the sniffer, since the NIC will not drop any packets. If the ping after flooding is higher than before, we can assume that the enlarged ping time is due to the increased workload of the suspected machine, and from that we can bring to a close that its NIC is in promiscuous mode.

Of course this method can cause false positives, since higher ping after flood is not necessarily in a cause and effect relationship with a NIC in promiscuous mode.

3.2.5 Decoy method

This method works somewhat differently, since it is not based on protocol level performance or characteristics. The Decoy method contains creating a trap for the sniffer. We generate traffic on the network which is very tempting for the sniffer. This can be FTP, TELNET or POP3 connections as they are unencrypted or information sent in non- encrypted mail. The reason is of course to observe if there is some activity which can be originated from the fact that someone sniffed this information [9]. This is a very environment- specific method, and may not be really considered as an IT based method.

3.2.6 DNS decoy method

The DNS decoy method is a particular case of decoy methods. Graham [12] describes the DNS decoy method: Many sniffing programs do automatic reverse-DNS lookups on the IP addresses they see. Therefore, a promiscuous mode can be detected by watching for the DNS traffic that it generates. This method can notice dual-homed machines and can work remotely. We need to monitor incoming inverse-DNS lookups on the DNS server in your organization. Simply do a ping sweep all through the company against machines that are known not to be present. anyone doing reverse DNS lookups on those addresses are attempting to lookup the IP addresses seen in ARP packets, which only sniffing programs do. This same technique works locally. Configure the detector in promiscuous mode itself, then send out IP datagrams to bad addresses and watch for the DNS lookups. One interesting issue with this technique is that hacker-based sniffing programs tend to resolve IP addresses as soon as they are found, whereas commercial programs tend to delay resolution until the point where the sniffer user views the protocol decodes.

3.2.7 Source-route method

Graham [12] describes this method also: Another method involves configuring the source-route information within the IP header. This can be used to detect sniffers on other, nearby segments.

1. Create a ping packet, but put a loose-source route to force it by another machine on the similar segment. This machine should have routing disabled, so that it will not in fact forward it to the target.
2. If you get a response, then it is likely the target sniffed the packet off the wire.
3. In the response, double check the TTL field to find out if it' came back due to sniffing.(rather than being routed correctly)

In loose source-routing, an option is added to the IP header. Routers will ignore the destination IP address and instead forward to the next IP address in the source-route option. This means when you send the packet, you can say “please send packet to ‘A’, but route it through ‘B’ first”. In this scenario, both “A” and “B” are on the segment. A does not route, and therefore will drop the packet when received. Therefore, “B” will only respond if he has sniffed the packet from the wire. On the off chance that A does indeed route (in this case B will respond), then the TTL field can be used to verify that B responded from routing through A, or answering directly.

3.3 Use of MAC addresses at destination node

With both ping and ARP methods, the destination address of the Ethernet frame is critical. As mentioned before, in promiscuous mode the NIC accepts all frames regardless of the destination address. However this is not sufficient, we need to find reply from the networking program of the target host. This depends on the MAC destination address, as well as the OS of the target host. According to Sanai [9] the following MAC addresses are important to try and as explained below:

FF-FF-FF-FF-FF-FF broadcast address:

All nodes should receive this kind of packet and respond because it is a broadcast address. A usual ARP request packet uses this address.

FF-FF-FF-FF-FF-FE fake broadcast address:

This address is a fake broadcast address missing the last 1 bit. This is to check whether the software filter examines all bits of the address and whether it will respond.

FF-FF-00-00-00-00 fake broadcast 16 bits:

This address is a fake broadcast address in which only the first 16 bits are the same as the broadcast address. This may be classified as a broadcast address and replied when the filter function only checks the first word of the broadcast address.

FF-00-00-00-00-00 fake broadcast 8 bits:

This address is a fake broadcast address in which only the first 8 bits are the same as the broadcast address. This may be classified as a broadcast address and replied when it only checks the first byte of the broadcast address.

01-00-00-00-00-00 group bit address.

This is an address with only the group bit set. This is to check whether this address is considered as a multicast address as Linux does.

01-00-5E-00-00-00 multicast address 0

Multicast address 0 is usually not used. We use this as an example of a multicast address not registered in the multicast list of the NIC. The hardware filter should reject this packet. However, this packet may be misclassified to be a multicast address when the software filter does not completely check all bits. The system kernel thus may reply to such packet when the NIC is set to promiscuous mode.

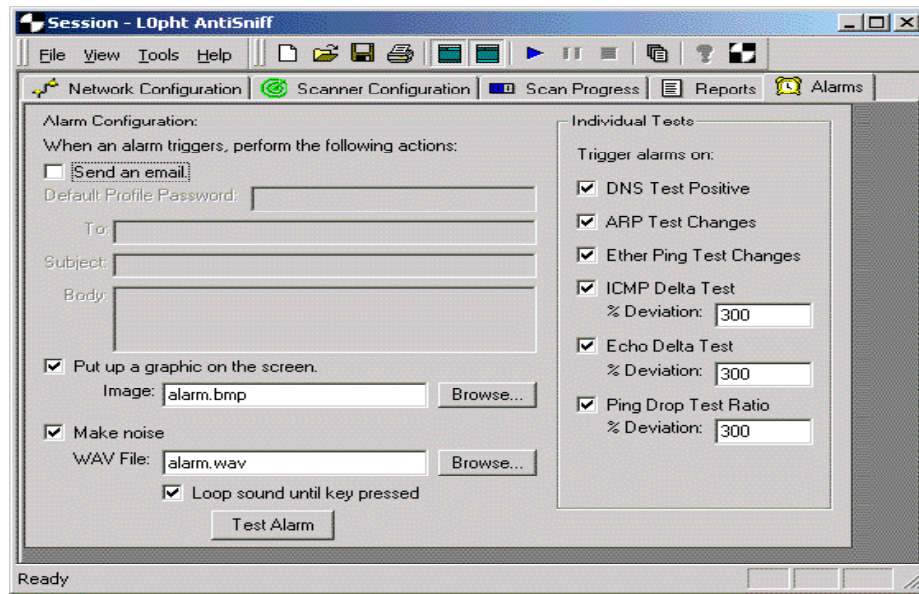
01-00-5E-00-00-01 multicast address 1

Multicast address 1 is an address that all hosts in the local network should receive. In other words, the hardware filter will pass this kind of packets by default. But it is possible that the NIC does not support multicast mode and does not respond. So this is to check whether the host supports multicast addresses. A sniffer is generally passive, it just gathers data. Hence it develops very difficult to detect sniffers, especially when running on a Shared Ethernet. But it is slightly easier when the sniffer is functioning on a Switched Ethernet network segment. When installed on a computer, a sniffer does produce some small amount of traffic. Following is an overview of the sniffer detection programs.

3.4 Review of existing Programs used to Detect Sniffer

3.4.1 Anti-Sniff:

The L0pht Heavy Industries originates the new program Anti Sniff. AntiSniff is network card promiscuous mode detector. It works by sending a series of carefully crafted packets in a certain order to a target machine, sniffing the results, and performing timing tests against the target. By measuring timing results and monitoring the target's responses on the network, it can be determined if the target is in promiscuous mode, i.e. sniffing the network. One weapon in a network intruder's collection is sniffing by placing a network card in promiscuous mode for the purpose of gathering account names and passwords. The intent of the sniffing intruder is to further the penetration into the invaded network, and potentially gain access to other networks. Detecting a network card in promiscuous mode is a good way to determine if your computer network has been compromised, but until recently you had to have direct access to the computer system with the network card. Even then it is possible that the intruder might have replaced some of the tools an administrator might use with Trojan versions that mask easy detection of the sniffing. Some advantages as accurate detection of promiscuous mode Ethernet cards [13]. Alerts sent via email -also includes visual and audio alerts. AntiSniff sessions can be stored for later use/analysis.



Screenshot 3.1 Antisniff

3.4.2 ARPWatch:

Arpwatch is a computer software tool for monitoring Address Resolution Protocol traffic on a computer network. It generates a log of observed pairing of IP addresses with MAC addresses along with a timestamp when the pairing appeared on the network. It also has the option of sending an email to an administrator when a pairing changes or is added. Network administrators monitor ARP activity to detect ARP spoofing [14].

ARP Watch

IP-address:

MAC-address:

Display period: day (86400 seconds), IP = all, MAC = all

Last hour only: [IP-based MAC-based Top stats](#)

Last day only (default): [IP-based MAC-based Top stats](#)

All: [IP-based MAC-based Top stats](#)

IP	What	IP	Old MAC	Time	Old time	Delta
2f:34:b4:0b:40:94						
	new station	192.168.186.128		Sun Sep 21 09:04:20 2008		
	new station	192.168.0.140		Sun Sep 21 10:15:00 2008		
2fe9:6c:6a:69:3c						
	flip flop	192.168.184.29	2fca:58:bb:e9d	Sun Sep 21 20:54:03 2008	Sun Sep 21 20:35:21 2008	1122 seconds
2fca:58:bb:e9d						
	flip flop	192.168.184.29	2fe9:6c:6a:69:3c	Sun Sep 21 20:20:23 2008	Sun Sep 21 18:48:27 2008	5516 seconds
2f:86:8a:52:61:3f						
	flip flop	192.168.20.231	2f86:8a:cc:bd:b9	Sun Sep 21 14:36:10 2008	Sun Sep 21 03:02:47 2008	41603 seconds
2f8a:10:f4:dd:d0						
	changed ethernet address (paradocs.homelink.ru)	192.168.184.20	2f54:9f:3:58:67	Sun Sep 21 23:50:52 2008	Sun Sep 21 23:09:57 2008	2455 seconds
2f86:8a:cc:bd:b9						
	flip flop	192.168.20.231	2f86:8a:52:61:3f	Sun Sep 21 18:20:36 2008	Sun Sep 21 14:37:18 2008	13398 seconds

<https://support.homelink.ru/cgi-bin/arpwatch.cgi?period=day;macbased=1> - Mon Sep 22 03:00:04 2008

Screenshot 3.2 Arpwatch

3.4.3 Snort:

Snort is a free and open source network intrusion prevention system and network intrusion detection system created by Martin Roesch in 1998. Snort is now developed by Sourcefire, of which Roesch is the founder and CTO. In 2009, Snort entered InfoWorld's Open Source Hall of Fame as one of the open source software of all time. Snort's open source network-based intrusion detection system (NIDS) has the ability to perform real-time traffic analysis and packet logging on Internet Protocol networks. Snort performs protocol analysis, content searching, and content matching. These basic services have many purposes including application-aware triggered quality of service, to de-prioritize bulk traffic when latency-sensitive applications are in use. The program can also be used to detect probes or attacks, including, but not limited to, operating system fingerprinting attempts, common gateway interface, buffer overflows, server message block probes, and stealth port scans. Snort can be configured in three main modes: sniffer, packet logger, and network intrusion detection [15]. In sniffer mode, the program will read network packets and display them on the console. In packet logger mode, the program will log packets to the disk. In intrusion detection mode, the program will monitor network traffic and analyze it against a rule set defined by the user. The program will then perform a specific action based on what has been identified.

Snort IDS Console - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address https://localhost:8080/

Snort IDS Console Unfilter Refresh every 30 secs View alerts since 6 AM or on

Alert Information		Sensors			Top Sources		Top Targets			Top Target Ports					
	#	%	Sensor	Sigs	Alerts	IP Address	Sigs	Alerts	IP Address	Sigs	Alerts	TCP	#	UDP	#
Signatures:	62			19	482		6	186		6	186	80	513	1434	1,259
TCP Alerts [View]	1,126	42%		13	177		5	5		5	5	139	180	53	242
UDP Alerts [View]	1,523	57%		11	240		3	21		3	24	443	122	177	9
ICMP Alerts [View]	0	0%		11	131		2	108		2	352	1433	23	111	6
Total Alerts [View]	2,649	100%		9	298		2	92		2	92	3389	19	69	2

Alert Overview by Signature

Earliest Alert: 2004-12-29 06:01:03

Latest Alert: 2004-12-29 15:57:12

Signatures					
Prio	Signature	# Sensors	# Alerts	# Srcs	# Dests
1	WED-MISC: cross site scripting attempt [sid 1497]	2	353	2	2
1	P2P Fastrack kazaa/morpheus traffic [sid 1699]	2	145	3	49
1	MS-SQL/SMB raiserror possible buffer overflow [sid 1380]	2	117	1	1
1	WEB-MISC: NetObserver authentication bypass attempt [sid 2441]	1	110	1	1
1	MS-SQL/SMB xp_cmdshell program execution [sid 681]	2	33	1	1
1	WEB-MISC: PCT Client Hello overflow attempt [sid 2515]	2	25	1	8
1	MS-SQL xp_cmdshell - program execution [sid 687]	1	17	2	1
1	MS-SQL/SMB xp_reg* registry access [sid 689]	2	12	1	1
1	MS-SQL/SMB sp_password password change [sid 677]	2	10	1	1
1	MS-SQL/SMB sp_delete_alert log file deletion [sid 678]	2	10	1	1
1	MS-SQL sp_start_job - program execution [sid 673]	2	6	1	1
1	MS-SQL sa login failed [sid 688]	1	5	1	1

Done

Internet

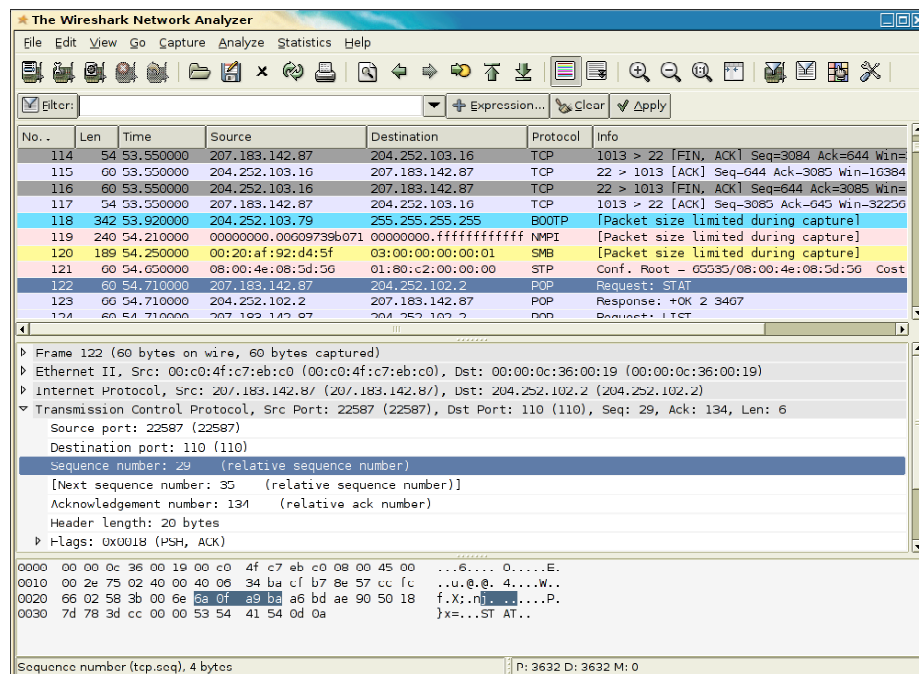
Screenshot 3.3 Snort

3.5 Review of some commonly used sniffing tools

Following are some commonly used network sniffing tools by intruders; which are easily available on the Internet.

3.5.1 Wireshark:

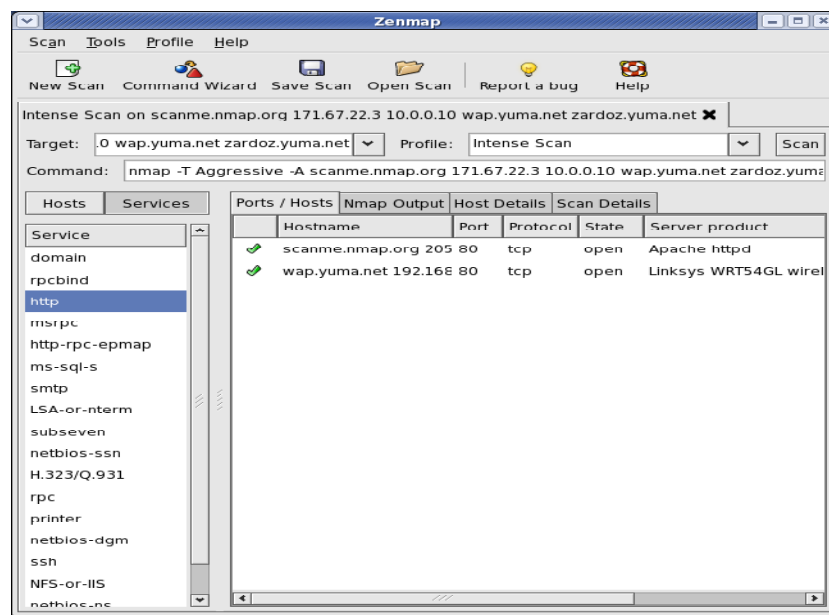
Wireshark is most powerful network protocol analyzers on the market. Infact, Wireshark is considered the de facto standard in the industry. This analyzer features: Live capture and offline analysis; standard three-pane packet browser; multi-platform, captured network data can be browsed by GUI. Other features: powerful display filters; rich VoIP analysis; read/write many different capture file formats; capture files can be compressed with on the fly; live data can be read from Ethernet, IEEE 802.11, PPP/HDLC, ATM, Bluetooth, USB, Token Ring, Frame Relay, FDDI, and others; decryption support; coloring rules; output can be exported to XML, PostScript, CSV, or plain text [16].



Screenshot 3.4 Wireshark

3.5.2 Zenmap:

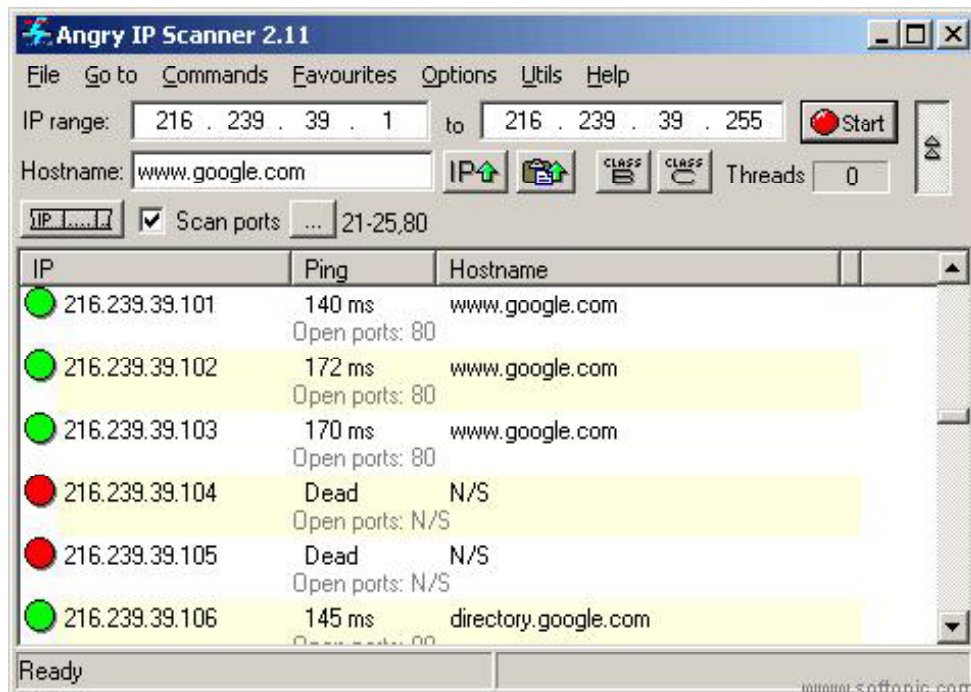
Zenmap is the authorized graphical user interface (GUI) for the Nmap Security Scanner. Zenmap is accessible for Windows, Linux, Mac, and BSD. Zenmap may be used to read live captures or save captures for later viewing. With Zenmap you can enable the features of Nmap to help you with: network inventory, managing service upgrade schedules, and monitoring host or service uptime [17]. Features comprise: Host discovery; port scanning; version detection; OS detection; scriptable interface; web scanning; full IPv6 support; Nping support; fast scanning; and much more .Zenmap is the official Nmap Security Scanner GUI. It is a multi-platform (Linux, Windows, Mac OS X, BSD, etc.) free and open source application which aims to make Nmap easy for beginners to use while providing advanced features for experienced Nmap users.



Screenshot 3.5 Zenmap

3.5.3 Angry IP Scanner: Angry IP [18] Scanner is open source, snappy platform scanner that is designed, from the ground up, to be incredibly fast and very easy to use. Angry IP deals the following features: Portable zero installation on certain platforms; ping checks; NetBIOS information; resolves hostnames; determines MAC address; can determine currently logged-in user; plug in system; scan

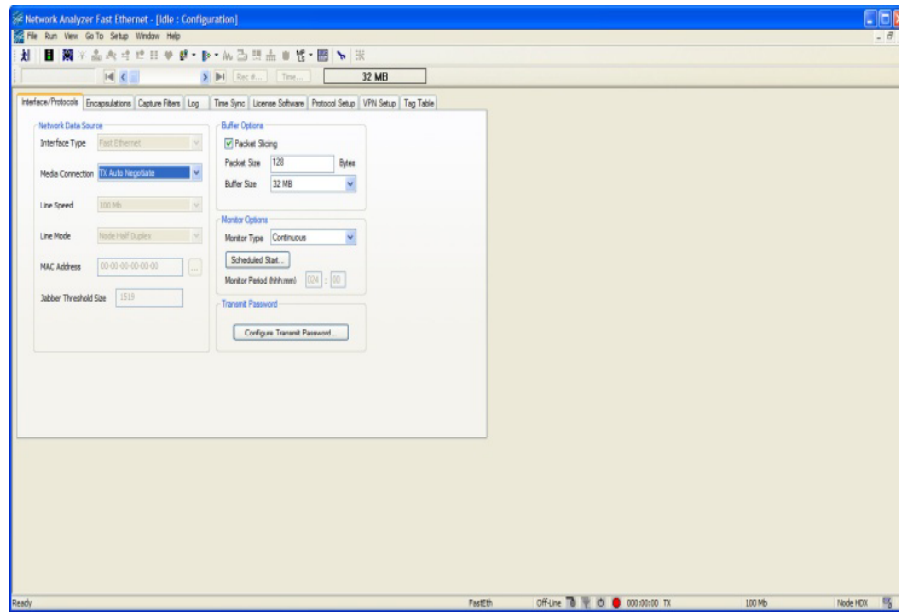
results can be saved as CSV, TXT, XML, or IP-Port list; and fast, multi-threaded scanning. Angry IP Scanner maintained by angryziber.



Screenshot 3.6 Angry IP Scanner

3.5.4 JDSU Network Analyzer Fast Ethernet:

JDSU Network Analyzer Fast Ethernet has long in features developed by Uniphase Corporation. The most popular versions of this product among our users are: 6.7, 7.1 and latest version 7.4. Although it has many features, you don't have to be a full-blown network analyst to make use of this tool. JDSU permits anyone of nearly any experience level to: Quickly decide who is on a network, who is using bandwidth, and where errors may be happening on the network [19]. We can also identify problems before they develop serious issues; use expert analysis tools to resolve network problems quickly; capture and analyze network traffic in real-time; and analyze data offline. JDSU deals multi-technology analysis, a consistent user-interface across platforms, and is scalable for distributed analysis.

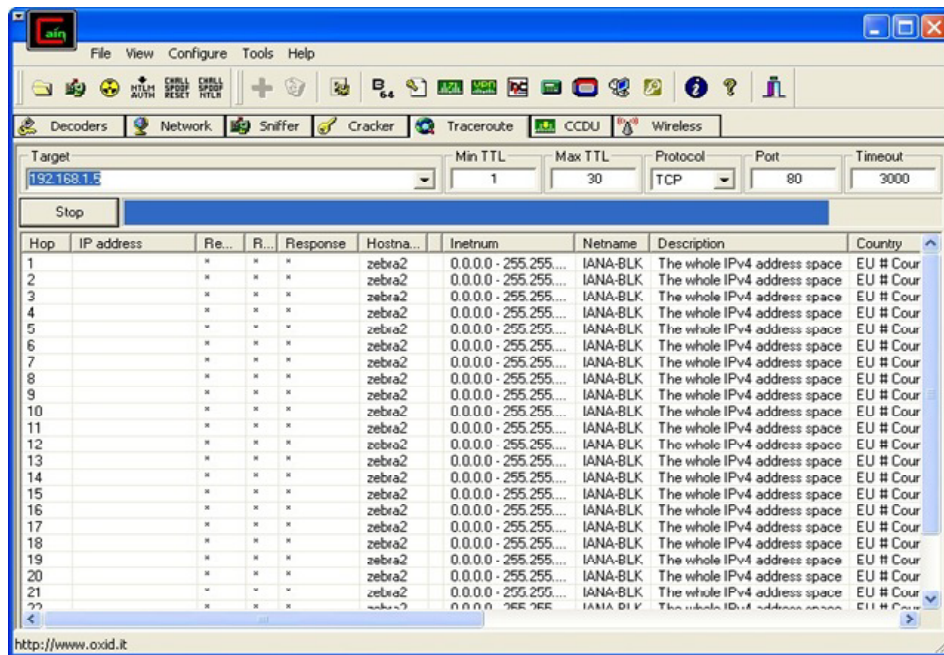


Screenshot 3.7 JDSU

3.5.5 Cain & Abel:

Cain & Abel is a password recovery tool for Microsoft Operating Systems. It permits simple recovery of different kinds of passwords by sniffing the network, cracking encrypted passwords with Dictionary, Brute-Force and Cryptanalysis attacks, recording VoIP conversations, decoding scrambled passwords, recovering wireless network keys, revealing password boxes, uncovering cached passwords and analyzing routing protocols. The program does not utilize any software vulnerabilities or bugs that could not be fixed with little attempt. It covers some security aspects/weakness present in protocol's standards, authentication methods and caching mechanisms; its main purpose is the simplified recovery of passwords and credentials from various sources, however it also ships some "non standard" utilities for Microsoft Windows users. Cain & Abel has been developed in the hope that it will be useful for network administrators, teachers, security consultants/professionals, forensic staff, security software vendors, professional penetration tester and everyone else that plans to use it for ethical reasons. The latest version (Cain & Abel v4.9.56) [20] is faster and contains a lot of new features like APR (Arp Poison Routing) which enables sniffing on switched LANs and Man-in-the-Middle attacks. The sniffer in this version can also examine encrypted protocols like SSH-1 and HTTPS,

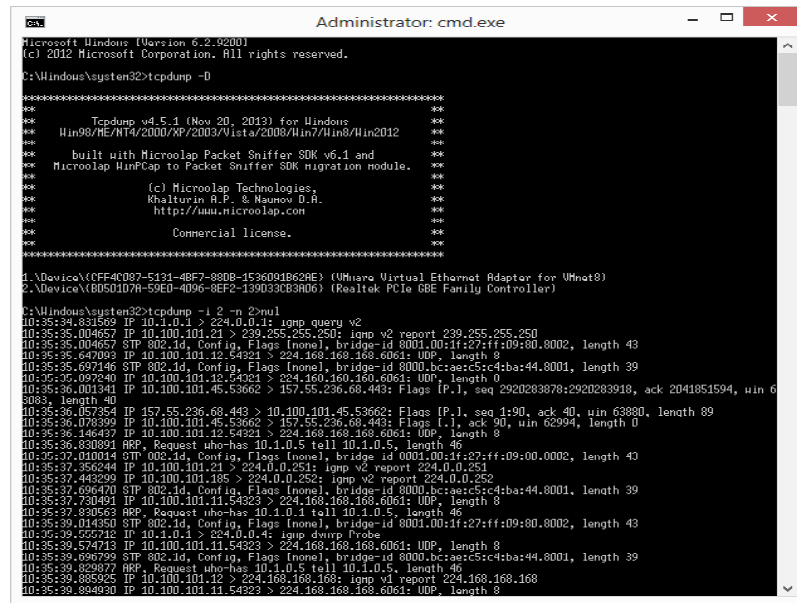
and contains filters to capture credentials from a large range of authentication mechanisms. The new version also ships routing protocols authentication monitors and routes extractors, dictionary and brute-force crackers for all common hashing algorithms and for several specific authentications, password/hash calculators, cryptanalysis attacks, password decoders and some not so common utilities related to network and system security.



Screenshot 3.8 Cain & Abel

3.5.6 Tcpcap:

Tcpcap is the network sniffer used before Wireshark came. It may not have the bells and whistles (like a GUI and logic for various application protocols) that Wireshark has, but it works well and with fewer security risks. It also requires fewer system resources. While Tcpcap doesn't receive new features often, it is actively maintained to fix bugs and portability problems. It is great for tracking down network problems or monitoring activity. There is a separate Windows port named WinDump. tcpcap is the source of the Libpcap/WinPcap packet capture library, which is used by Nmap and many other tools [21].



```
Administrator: cmd.exe
Microsoft Windows [Version 6.0.6002]
(c) 2012 Microsoft Corporation. All rights reserved.

C:\Windows\system32\tcpdump -i 2 -n 2

*****
**
** Tcpdump v4.5.1 (Nov 20, 2013) for Windows
** Win98/NT4/2000/XP/2003/Vista/2008/Win7/Win8/Win2012
**
** built with Microolap Packet Sniffer SDK v6.1 and
** Microolap WinPcap to Packet Sniffer SDK Migration module.
**
** (c) Microolap Technologies,
** Khalturin A.P. & Naumov D.A.
** http://www.microolap.com
**
** Commercial license.
**
*****

1. \Device\NPF{...} (VMware Virtual Ethernet Adaptor for VMnet8)
2. \Device\NPF{...} (Realtek PCIe GBE Family Controller)

C:\Windows\system32\tcpdump -i 2 -n 2
10:35:34.831569 IP 10.10.10.1 > 224.0.0.1: icmp query v2
10:35:35.004657 IP 10.10.10.1 > 224.0.0.1: icmp v2 report 239.255.255.250
10:35:35.004657 SIP 802.1d, Config, Flags [none], bridge-id 8000.bcaac5c4ba:44.8001, length 43
10:35:35.647093 IP 10.10.10.12.54321 > 224.168.168.168.6061: UOP, length 8
10:35:35.697146 SIP 802.1d, Config, Flags [none], bridge-id 8000.bcaac5c4ba:44.8001, length 39
10:35:35.697240 IP 10.10.10.12.54321 > 224.168.168.168.6061: UOP, length 0
10:35:36.001341 IP 10.10.10.1.45.53662 > 157.55.236.68.443: Flags [P,], seq 2920283878:2920283918, ack 2041851594, win 63083, length 40
10:35:36.057254 IP 157.55.236.68.443 > 10.10.10.1.45.53662: Flags [P,], seq 1:90, ack 40, win 63880, length 89
10:35:36.078399 IP 10.10.10.1.45.53662 > 157.55.236.68.443: Flags [I,], ack 90, win 62994, length 0
10:35:36.146437 IP 10.10.10.12.54321 > 224.168.168.168.6061: UOP, length 8
10:35:36.230091 ARP, Request who-has 10.1.0.5 tell 10.1.0.5, length 46
10:35:37.010014 SIP 802.1d, Config, Flags [none], bridge-id 8000.bcaac5c4ba:44.8001, length 43
10:35:37.356244 IP 10.10.10.1.21 > 224.0.0.251: icmp v2 report 224.0.0.251
10:35:37.443298 IP 10.10.10.1.185 > 224.0.0.252: icmp v2 report 224.0.0.252
10:35:37.696470 SIP 802.1d, Config, Flags [none], bridge-id 8000.bcaac5c4ba:44.8001, length 39
10:35:37.730491 IP 10.10.10.1.11.54323 > 224.168.168.168.6061: UOP, length 8
10:35:37.830563 ARP, Request who-has 10.1.0.1 tell 10.1.0.5, length 46
10:35:37.830563 SIP 802.1d, Config, Flags [none], bridge-id 8000.bcaac5c4ba:44.8001, length 43
10:35:39.014350 SIP 802.1d, Config, Flags [none], bridge-id 8000.bcaac5c4ba:44.8001, length 43
10:35:39.055742 IP 10.1.0.1 > 224.0.0.4: icmp dump Probe
10:35:39.574743 IP 10.10.10.1.11.54323 > 224.168.168.168.6061: UOP, length 8
10:35:39.690799 SIP 802.1d, Config, Flags [none], bridge-id 8000.bcaac5c4ba:44.8001, length 39
10:35:39.829877 ARP, Request who-has 10.1.0.5 tell 10.1.0.5, length 46
10:35:39.885925 IP 10.10.10.12 > 224.168.168.168: icmp v1 report 224.168.168.168
10:35:39.884930 IP 10.10.10.1.11.54323 > 224.168.168.168.6061: UOP, length 8
```

Screenshot 3.9 Tcpcdump

3.5.7 Kismet: Kismet is an 802.11 layer2 wireless network detector, sniffer, and intrusion detection system. Kismet will work with any wireless card which supports raw monitoring (rfmon) mode, and (with appropriate hardware) can sniff 802.11b, 802.11a, 802.11g, and 802.11n traffic. Kismet also supports plugins which allow sniffing other media such as DECT. Kismet identifies networks by passively collecting packets and detecting standard named networks, detecting hidden networks, and inferring the presence of non beaconing networks via data traffic. In Sep 25, 2013 Released the first version of Smarter Wi-Fi Manager for Android. The Latest version is Kismet 2011-03-R2 [22].



Name	BSSID	T	C	Ch	Freq	Pkts	Size	Bcr%	Sig	Cnt	Manuf	Cty	Seen By
TRENDnet	00:14:01:5F:97:12	A	0	1	2417	1	0B	100	---	1	Trendware	---	wlan0
linksys_SES_45997	00:16:06:1B:14:FF	A	0	6	2432	1	0B	100	-78	1	Cisco-Link	---	wlan0
Autogroup Probe	00:13:E8:92:3F:CB	P	N	---	---	2	0B	---	0	1	IntelCorpo	---	wlan0
linksys	00:1A:7D:D9:BC:13	A	N	6	2437	2	0B	100	-86	1	Cisco-Link	---	wlan0
WPA41	00:1F:33:F3:C5:4A	A	N	11	2462	3	0B	---	---	1	Actiontec	---	wlan0
65103	00:1F:33:F3:C5:4A	A	N	---	---	3	0B	---	---	1	Actiontec	---	wlan0
TFS	00:09:5B:D7:9D:B2	A	N	---	---	4	0B	---	-68	1	Netgear	---	wlan0
Xu Chen	00:1B:01:F9:7D:F8	A	N	6	2437	4	0B	0%	-75	1	Actiontec	US	wlan0
TK421	00:1B:01:FE:68:77	A	0	6	2437	4	0B	---	-79	1	Actiontec	---	wlan0
meskas	00:1B:01:FE:68:77	A	0	11	2462	5	0B	100	-71	1	Actiontec	US	wlan0
Elinia-PC-Wireless	00:24:B2:0E:E6:E2	A	0	11	2462	7	0B	100	-45	1	Netgear	---	wlan0
Pickles	00:1F:33:F3:C5:4A	A	0	2	2422	8	0B	---	-75	1	Netgear	---	wlan0
BSSID: 00:1F:33:F3:C5:4A	Crypt: TKIP WPA PSK AESCCM	Manuf: Netgear	SeenBy: wlan0										
38CB	00:1B:CE:07:60:77	A	N	6	2447	19	0B	---	-82	1	HontaiPrec	---	wlan0
Danish_Penguin	00:13:10:35:59:CB	A	N	9	2462	331	2K	50%	-32	5	Cisco-Link	---	wlan0

No GPS info (GPS not connected)

45

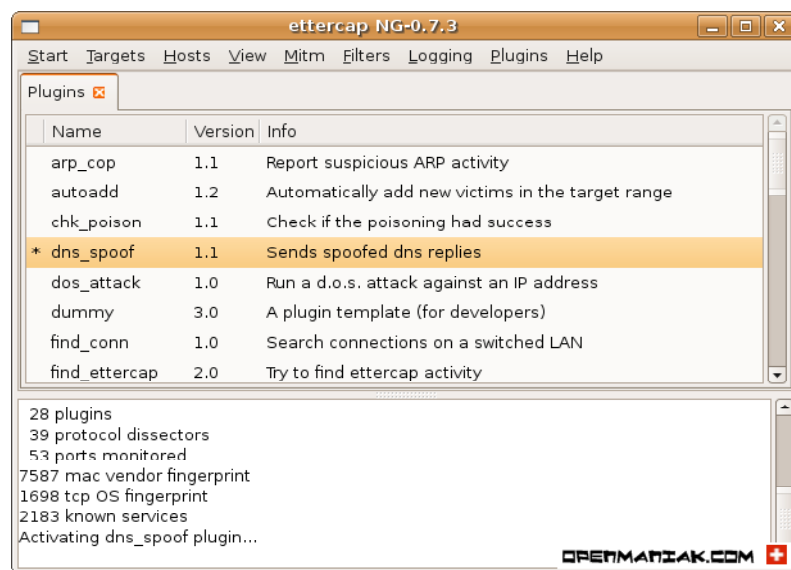
0

INFO: Detected new probe network "Danish_Penguin", BSSID 00:13:E8:92:3F:CB, encryption no, channel 0, 60.00 mb
ERROR: Could not connect to the spectools server localhost:30569
INFO: Detected new managed network "linksys_SES_45997", BSSID 00:1A:7D:D9:BC:13, encryption yes, channel 6, 54
INFO: Detected new managed network "linksys", BSSID 00:1A:7D:D9:BC:13, encryption no, channel 6, 54.00 mbit
ERROR: No update from GPS in 15 seconds or more, attempting to reconnect

Screenshot 3.10 Kismet

3.5.8 Ettercap:

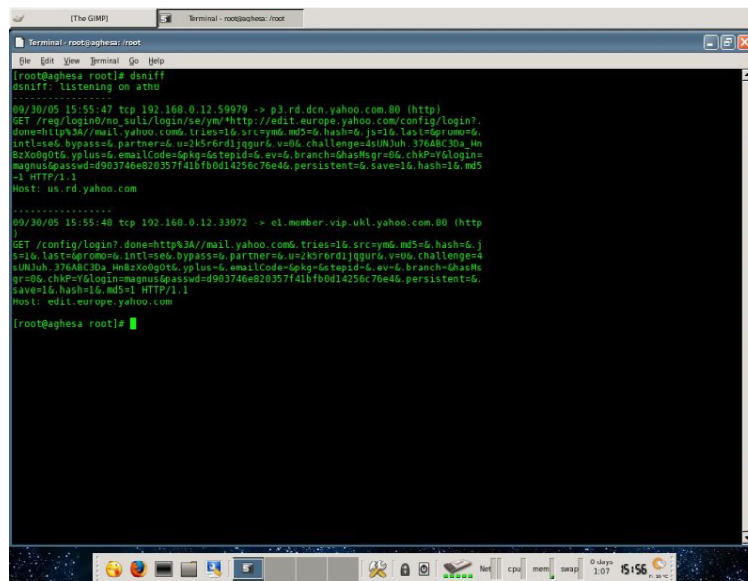
Ettercap is a complete suite for man in the middle attacks. It features sniffing of live connections, content filtering on the fly and many other interesting tricks. It supports active and passive dissection of many protocols and includes many features for network and host analysis [23].



Screenshot 3.11 Ettercap

3.5.9 dsniff:

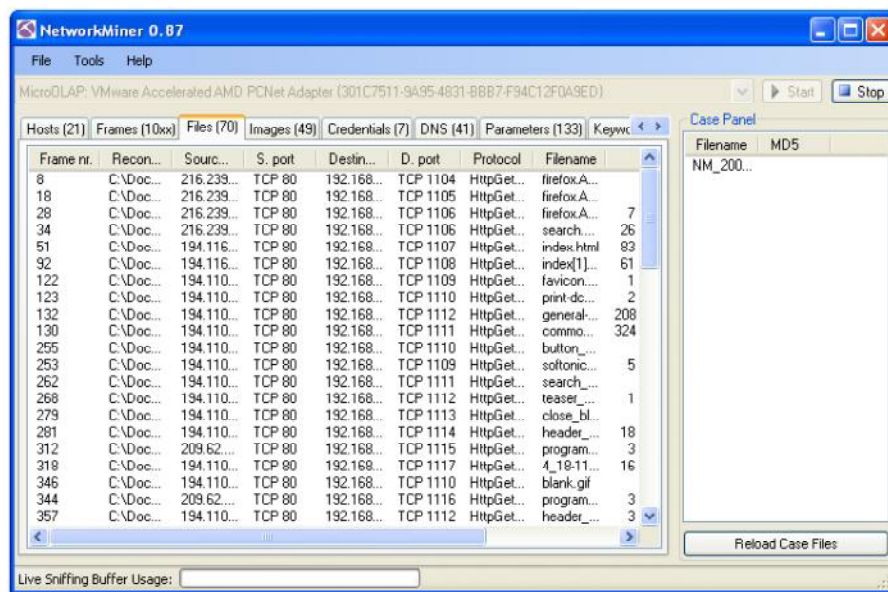
This is well-known and well-designed suite which includes many tools: dsniff, filesnarf, mailsnarf, msgsnarf, urlsnarf, and webspy passively monitor a network for interesting data (passwords, e-mail, files, etc.); arpspoof, dnsspoof, and macof facilitate the interception of network traffic normally unavailable to an attacker (e.g, due to layer-2 switching). The suite suffers from the lack of any updates in the last decade, but it is still a great toolset for handling your password sniffing needs [24].



```
[root@aghesa root]# dsniff
dsniff: listening on eth0
-----
09/30/05 15:55:47 tcp 192.168.0.12.59979 -> p3.rd.dcn.yahoo.com.80 (http)
GET /req/login0/no_suli/login/se/ym/http://edit.europe.yahoo.com/config/login?
done=http33a//mail.yahoo.com&.tries=1&.src=ym&.md5=6.hash=6.js=1&.last=6promu=6.
intinset.bypass=6.partner=6.u=2K5r6rd1jggr6.v60&.challenge=6UNJuh.376AbC30a.Mn
62x0p016.yplus=6.emailcode=6qhp&.stepid=6.ev=6.branch=6aaskgr=6.chkP=1&.login=
magnus&.passwd=6903746e820357f41bfbd14256c76e46.persistent=6.save=1&.hash=1&.md5
=1 HTTP/1.1
Host: us.rd.yahoo.com
-----
09/30/05 15:55:48 tcp 192.168.0.12.33972 -> e1.member.vip.uhl.yahoo.com.80 (http)
GET /config/login?done=http33a//mail.yahoo.com&.tries=1&.src=ym&.md5=6.hash=6.j
s=1&.last=6promu=6.intinset.bypass=6.partner=6.u=2K5r6rd1jggr6.v60&.challenge=
6UNJuh.376AbC30a.Mn62x0p016.yplus=6.emailcode=6qhp&.stepid=6.ev=6.branch=6aask
gr=6.chkP=1&.login=magnus&.passwd=6903746e820357f41bfbd14256c76e46.persistent=6.
save=1&.hash=1&.md5=1 HTTP/1.1
Host: edit.europe.yahoo.com
[root@aghesa root]#
```

Screenshot 3.12 dsniff

3.5.10 NetworkMiner: NetworkMiner is a Network Forensic Analysis Tool (NFAT) for Windows (but also works in Linux / Mac OS X / FreeBSD). NetworkMiner can be used as a passive network sniffer/packet capturing tool in order to detect operating systems, sessions, hostnames, open ports etc. without putting any traffic on the network. NetworkMiner can also parse PCAP files for off-line analysis and to regenerate/reassemble transmitted files and certificates from PCAP files. NetworkMiner collects data like forensic evidence about hosts on the network rather than to collect data regarding the traffic on the network. The main user interface view is host centric i.e. information grouped per host rather than packet centric i.e. information showed as a list of packets/frames. NetworkMiner has, since the first release in 2007, become a popular tool among incident response teams as well as law enforcement. NetworkMiner is today used by companies and organizations all over the world [25].



Screenshot 3.13 NetworkMiner

❖ **References**

- [1] <http://www.tcpdump.org/> [accessed on: 13/10/2014]
- [2] <http://reptile.rug.ac.be/~coder/sniffit/sniffit.html> [accessed on: 13/10/2014]
- [3] <http://www.ethereal.com/> [accessed on: 14/10/2014]
- [4] <http://netgroup-serv.polito.it/analyzer/> [accessed on: 16/10/2014]
- [5] <http://netgroup-serv.polito.it/windump/> [accessed on: 16/10/2014]
- [6] <http://www.lsv.ens-cachan.fr/~goubault/SECI-02/Final/actes-seci02/pdf/008-Abdelallahelhadj.pdf>, H. Abdelallah Elhadj et. al. “An Experimental Sniffer Detector: SnifferWall” [accessed on: 16/10/2014]
- [7] Mumtaz AL-Mukhtar, Yasir Ahmed Abdullah “Developing a Sniffer Detector for Windows Operating Systems” The 1st Regional Conference of Eng. Sci. NUCEJ Spatial ISSUE 11(1) pp84-90
- [8] <http://www.just.edu.jo/~tawalbeh/nyit/incs745/presentations/Sniffers.pdf>, Sumit D.,”Sniffers Basics and Detection” Information Security Management Team. [accessed on: 16/10/2014]
- [9] http://www.securityfriday.com/promiscuous_detection_01.pdf Sanai, D. “Detection of Promiscuous Nodes Using ARP Packets”. [Accessed on: 18/10/2014]
- [10] http://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus3000/sw/unicast/602_u1_1/13_nx-os.pdf [Accessed on: 18/10/2014]
- [11] <http://www.docin.com/tag/Sniffers> Khcherif, R. (n.d.), “ARP cache Poisoning For the Detection of Sniffers in an Ethernet Network.” [Accessed on: 18/10/2014]
- [12] http://www.windowsecurity.com/whitepapers/misc/Sniffing_network_wiretap_sniffer_FAQ_.html Graham, R. “Sniffing (network wiretap, sniffer)” [Accessed on: 18/10/2014]

- [13] *“[http://repo.hackerzvoice.net/depot_cehv6/CEHv6%20Module% 2010% 20Sniffers/snifferdetection.pdf](http://repo.hackerzvoice.net/depot_cehv6/CEHv6%20Module%2010%20Sniffers/snifferdetection.pdf)” Ryan Spangler, “Packet Sniffer Detection with AntiSniff” University of Wisconsin - Whitewater May 2003 [Accessed on: 18/10/2014]*
- [14] *<http://www.wilderssecurity.com/threads/quick-little-arp-watch-tutorial-linux.244758/> [Accessed on: 03/11/2014]*
- [15] *<https://www.snort.org/> [accessed on: 05/11/2014]*
- [16] *https://www.wireshark.org/docs/wsug_html_chunked/ [accessed on: 09/11/2014]*
- [17] *<http://nmap.org/book/zenmap.html> [accessed on: 10/11/2014]*
- [18] *<http://angryip.org/> [accessed on: 10/11/2014]*
- [19] *http://www.jdsu.com/ProductLiterature/dnaoverview_ds_nsd_tm_ae.pdf [accessed on: 11/11/2014]*
- [20] *<http://www.oxid.it/cain.htm> [accessed on: 15/11/2014]*
- [21] *<http://www.tcpdump.org/> [accessed on: 15/11/2014]*
- [22] *<http://www.kismetwireless.net/download.shtml> [accessed on: 15/11/2014]*
- [23] *<http://ettercap.github.io/ettercap/downloads.html> [accessed on: 15/11/2014]*
- [24] *<http://www.monkey.org/~dugsong/dsniff/> [accessed on: 15/11/2014]*
- [25] *<http://www.netresec.com/?page=NetworkMinser/> [accessed on: 15/11/2014]*