Hacker Highschool
SECURITY AWARENESS FOR TEENS

LESSON 7
ATTACK ANALYSIS

Hacker Highschool
SECURITY AWARENESS FOR TEENS

ISECOM
INSTITUTE FOR SECURITY AND OPEN METHODOLOGIES

"License for Use" Information

The following lessons and workbooks are open and publicly available under the following terms and conditions of ISECOM:

All works in the Hacker Highschool project are provided for non-commercial use with elementary school students, junior high school students, and high school students whether in a public institution, private institution, or a part of home-schooling. These materials may not be reproduced for sale in any form. The provision of any class, course, training, or camp with these materials for which a fee is charged is expressly forbidden without a license including college classes, university classes, trade-school classes, summer or computer camps, and similar. To purchase a license, visit the LICENSE section of the Hacker Highschool web page at www.hackerhighschool.org/license.

The HHS Project is a learning tool and as with any learning tool, the instruction is the influence of the instructor and not the tool. ISECOM cannot accept responsibility for how any information herein is applied or abused.

The HHS Project is an open community effort and if you find value in this project, we do ask you support us through the purchase of a license, a donation, or sponsorship.

Table of Contents

Contributors

Pete Herzog, ISECOM
Chuck Truett, ISECOM
Marta Barceló, ISECOM
Kim Truett, ISECOM

ISECOM
INSTITUTE FOR SECURITY AND OPEN METHODOLOGIES

7.0 Introduction

There are a lot of programs on your computer that will want to open up network connections. Some of these programs have valid reasons for connecting (your web browser won't work nearly as well without access to a network connection as it will with one), others have been written by people with motives ranging from questionable to criminal. If you want to protect your computer, you'll have to learn how to detect network access, and identify the source and intent. Not every attempt at network access is an attack, but if you don't know how to identify friend from foe, you might as well just leave your door open.

7.1 Netstat and Host Application Firewalls

To be able to identify an attack, you have to know what applications and processes normally run on your computer. Just looking at a graphical interface, whether in Windows or Linux, won't let you see what's going on underneath the surface. *Netstat* and a *firewall* can be used to help you identify which programs should be allowed to connect with the network.

7.1.1 Netstat

(netstat is also discussed in section 5.2.3) The *netstat* command will display the status of the network. Netstat can give you information about what ports are open and the IP addresses that are accessing them, what protocols those ports are using, the state of the port, and information about the process or program using the port.

At a command prompt enter:

`netstat -aon` (for Windows) or

`netstat -apn` (for Linux)

and netstat will produce a display similar to this:

```
  Active Connections
    Proto  Local Address          Foreign Address        State           PID
    TCP    0.0.0.0:1134           0.0.0.0:0              LISTENING       3400
    TCP    0.0.0.0:1243           0.0.0.0:0              LISTENING       3400
    TCP    0.0.0.0:1252           0.0.0.0:0              LISTENING       2740
    TCP    257.35.7.128:1243      64.257.167.99:80       ESTABLISHED     3400
    TCP    257.35.7.128:1258      63.147.257.37:6667     ESTABLISHED     3838
    TCP    127.0.0.1:1542         0.0.0.0:0              LISTENING       1516
    TCP    127.0.0.1:1133         127.0.0.1:1134         ESTABLISHED     3400
    TCP    127.0.0.1:1134         127.0.0.1:1133         ESTABLISHED     3400
    TCP    127.0.0.1:1251         127.0.0.1:1252         ESTABLISHED     2740
    TCP    127.0.0.1:1252         127.0.0.1:1251         ESTABLISHED     2740
```

Now, you need to match the numbers in the PID column with names of the processes that are running. In Windows, you should bring up the *Windows Task Manager*, by pressing

CTL+ALT+DEL. (If it doesn't show a PID column, click on *View*, then *Select Columns*, then select *PID*.) In Linux, go to a command prompt and enter `ps auxf` to display the processor status.

In the case of our example results listed above, we find that PID 3400 belongs to our web browser and PID 2740 belongs to our email client, both of which we have knowingly executed, and both of which have valid reasons for establishing connections to the Internet. However, PID 3838 belongs to a program named 6r1n.exe, and PID 1516 belongs to a program named buscanv.exe, neither of which we are familiar with.

However, just because you don't recognize the name of a program, that doesn't mean that it doesn't have a reason to be running on your system. The next step in this process is for us to go to an Internet search engine and try to discover what these two programs do.

In our search, we discover that buscanv.exe is required by our virus scanner and should be running. However, 6r1n.exe could be a trojan. Looking again at the display from netstat, we can see that the port associated with the 6r1n.exe program is 6667, an IRC port commonly used by trojans for remote access. At this point, we begin researching methods for removing the trojan.

7.1.2 Firewalls

Now, you could sit at your computer and run netstat over and over and over and over, keeping a constant vigil on the data moving in and out of your computer, or you could use a *firewall* program to do it for you.

A firewall monitors network traffic on your computer and uses a number of rules or *filters* to determine whether or not a program should be allowed to access the network. A firewall can filter data according to IP addresses and domain names, ports and protocols, or even transmitted data. This means that you can do things such as:

block or allow all data coming from a specific IP address

block or allow all data coming from a specific domain

close or open specific ports

block or allow specific protocols

block or allow packets which contain specific data strings.

You can also combine these filters to allow for careful control of the data that is allowed through the network. For example, you could:

allow data from *www.ibiblio.com* through ports 20 or 21 only

allow data from *www.google.com* that uses the UDP protocol

allow data from *www.yahoo.com* only through port 80 and only if the packets contain the text string "I will not waste bandwidth".

You, however, won't need to work out all the rules on your own. You can take advantage of the firewalls ability to set these filters itself. After you first install a firewall, you will be hit with a flurry of warnings and requests for access, and you will have to determine whether or not a program will be allowed to access the network. (The firewall may also give you the option to let the firewall determine what rights programs have to access the network, but then you wouldn't learn anything, would you?) This process is going to be similar to the one that we used to identify the programs listed by netstat. A program named *iexplorer.exe* is obviously Microsoft's Internet Explorer and, if you use it as your web browser, then the firewall must allow it to access the Internet. But a program named *cbox.exe* could be anything. You've got no

choice but to go to your preferred web search engine and check it out. (Of course, before you can do this, you've got to tell the firewall to allow your web browser to access the Internet.)

The firewall program should also give you the option to allow access to a program repeatedly, or just once. Some programs – like your web browser – should be allowed to access the network anytime, but for other programs – such as the ones that automatically check for program updates – you can learn a lot about how your computer works by having the firewall ask for permission every time that the program requests access.

Firewalls are available as stand-alone programs (including a number of free versions for both Windows and Linux) or they are often bundled with anti-virus software. Additionally, Windows XP comes with a built-in firewall, but, as is the case with Windows Internet Explorer, it will be targeted by people looking for exploits – flaws in other firewalls may never be found, but flaws in a Microsoft firewall will be found and they will be exploited.

Exercises:

Open up a command prompt on your computer and enter:

`netstat -aon` (for Windows) or

`netstat -apn` (for Linux)

Match the PID numbers with program names and try to determine which programs on your computer are accessing the network. (This is something that you can try at home, also.)

7.2 Packet Sniffers

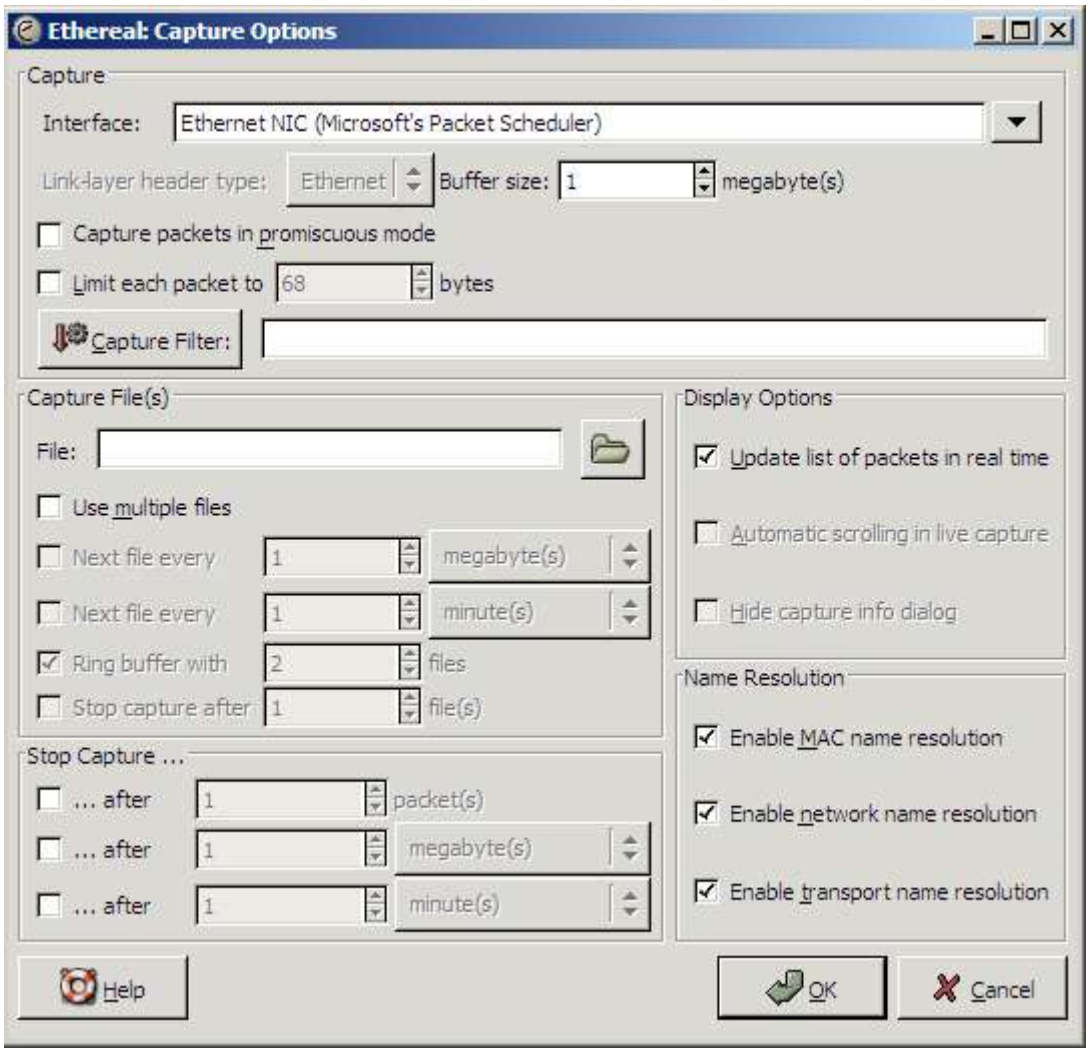
Netstat will tell you what programs are connected to the network, but it won't show you what data these programs are sending. A *packet sniffer*, however, gives you the means to record and study the actual data that the programs are sending through the network.

7.2.1 Sniffing

A packet sniffer will record the network traffic on your computer, allowing you to look at the data. *Tcpdump* (and its Windows port, *windump*) may be considered the archetypical packet sniffers, but we're going to use *Ethereal* for our examples, because its graphical interface is simpler, and it allows you to more quickly record and view a basic capture file.

If you don't already have Ethereal, it can be downloaded from *www.ethereal.com*. Note to Windows users: To use Ethereal on a Windows based system, you must first download and install the *WinPcap* packet capture driver. WinPcap is available on the Ethereal download page or you can go to *www.winpcap.polito.it* to download it directly.

Shut down all other applications, then start Ethereal. In the menu click on *View* then *Autoscroll in Live Capture*. Next, click on *Capture,* then *Start* to go to the *Capture Options* screen. On the *Capture Options* screen, make sure that the box marked "Capture packets in promiscuous mode" is not checked, that the three check boxes under "Name Resolution" are checked, and that the box marked "Update list of packets in real time" is checked.



Now, click on the "OK" button.

In theory, nothing should happen now. You'll see a window for Ethereal which displays the number of packets that have been captured, and, behind this, you'll see the Ethereal screen which displays the data in those packets. You may see a small amount of traffic that is caused by the computers on the local network trying to keep track of each other (ARP, NBNS, ICMP) followed by DNS activity as Ethereal attempts to resolve names.

To see activity, you're going to generate some activity. While Ethereal is running, open your web browser. Minimize everything other than the main Ethereal screen and your web browser, and arrange the Ethereal and web browser windows so that you can see both at the same time. Now go to a web search engine, such as *www.google.com*.

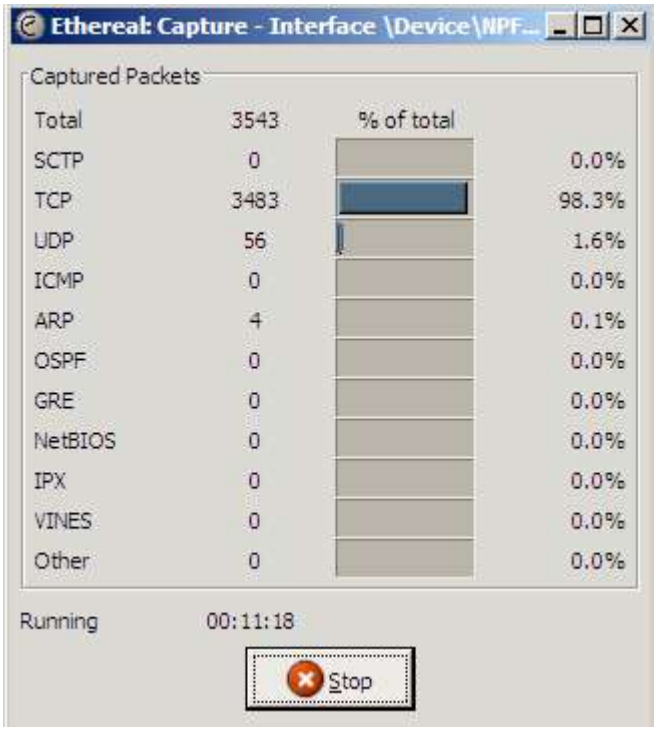
As the web page loads, your should see information about captured packets scrolling up through the Ethereal screen. Pick a search term and enter it into the search bar. Click on some of the web pages that are brought up by the search and watch what happens in Ethereal as you do.

Note: If Ethereal reports no network activity at all, you may have the wrong network interface chosen. Go to the *Interface* drop-down list in the *Capture Options* screen and choose a different network interface.

7.2.2 Decoding Network Traffic

Now that you can see the network data that's moving through your computer, you have to figure out how to decode it.

In Ethereal, the first step, before you even end the capture session, is to look at the summary capture screen that the program displays while it is performing the capture. For our web browsing session, most of the packets should have been TCP packets (although if you stopped to watch a streaming video, your UDP packet numbers will have been increased). However, if you're capturing a simple web browsing session, and you see a large number of ARP or ICMP packets, that could indicate a problem.



After you've ended the capture session, you're going to see output similar to this:

No.	Time	Source	Destination	Protocol	Info
1	0.000000	257.10.3.250	rodan.mozilla.org	TCP	1656 > 8080 [SYN] Seq=0 Ack=0 Win=16384 Len=0 MSS=1460
2	0.045195	257.10.3.250	rheet.mozilla.org	TCP	1657 > http [SYN] Seq=0 Ack=0 Win=16384 Len=0 MSS=1460
3	0.335194	rheet.mozilla.org	257.10.3.250	TCP	http > 1657 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460
4	0.335255	257.10.3.250	rheet.mozilla.org	TCP	1657 > http [ACK] Seq=1 Ack=1 Win=17520 Len=0
5	0.338234	257.10.3.250	rheet.mozilla.org	HTTP	GET /products/firefox/start/ HTTP/1.1
6	0.441049	rheet.mozilla.org	257.10.3.250	TCP	http > 1657 [ACK] Seq=1 Ack=580 Win=6948 Len=0
7	0.441816	rheet.mozilla.org	257.10.3.250	HTTP	HTTP/1.1 304 Not Modified
8	0.559132	257.10.3.250	rheet.mozilla.org	TCP	1657 > http [ACK] Seq=580 Ack=209 Win=17312 Len=0

```
9 2.855975    257.10.3.250          rodan.mozilla.org     TCP      1656 > 8080 [SYN] Seq=0 Ack=0 Win=16384 Len=0 MSS=1460
10 4.475529    257.10.3.250          name.server.com       DNS      Standard query PTR 250.3.10.257.in-addr.arpa
11 4.475776    257.10.3.250          name.server.com       DNS      Standard query PTR 205.111.126.207.in-addr.arpa
12 4.475854    257.10.3.250          name.server.com       DNS      Standard query PTR 202.111.126.207.in-addr.arpa
```

In this example, these twelve packets illustrate the web browser's activity as it connects with its specified start page. The most easily decoded information is in the *Source* and *Destination* columns. IP address 257.10.3.250 is the local computer; the other IP addresses have been resolved to names by Ethereal. Since the web browser used was the Mozilla Firefox browser, and since its start page was the default Mozilla Firefox page, it is not surprising to see addresses from the *mozilla.org* domain. The requests sent to *name.server.com* were probably generated by Ethereal when it sent DNS queries to resolve the IP addresses into names. (Note: these accesses by the Ethereal program were caused by the options you set in the *Display Options* and *Name Resolution* boxes. They were set to *on* in this example in order to produce a more readable output. If you toggle these options to *off*, then you won't have this extra data.)

Looking at source and destination information can help you spot unauthorized activity. For example, an unfamiliar domain name that is repeatedly accessed might indicate that you have a spyware program installed.

The next column is the *Protocol* column, which tells you what protocol the packets used. Again, to know when something is wrong here, you're going to have to know what to expect. In our web browsing session, we expect TCP and HTTP, and we understand why the DNS packets are there, but, for example, a large number of ICMP packets could mean that your machine is being pinged or traced.

The last column, *Info*, provides more detailed information about the packets. Packets 2, 3 and 4 show the TCP *three-handed handshake* of `SYN`, `SYN/ACK`, `ACK`, which indicates that a connection has been made. Packet 5 shows an `HTTP GET` command followed in packet 7 by a `304 Not Modified` response.

If you want more information about the packets, the bottom two panes in the Ethereal screen show detailed explanations. The middle pane shows the details of the packet header. The bottom pane shows a hex and ascii dump of the data in the packet.

7.2.3 Sniffing Other Computers

Some of you, having looked at the information in this section – and having looked at the data that can be recorded by Ethereal, may be wondering about the possibilities of using packet sniffing software to record activity on other people's computers. Is this possible?

Yes – and no. It's called *promiscuous mode* and it allows a packet sniffer to monitor network activity for all computers on a network. This means that you might be able to record network activity on another computer that is in your own network (depending on the way that the hardware is set up), but you can't pick any one computer at random and magically sniff their data – the two computers must be physically connected, and the hardware and software must be properly configured.

7.2.4 Intrusion Detection Systems

You've probably realized that, to use a packet sniffer to detect unauthorized activity in real time, would require you to sit at your computer, watching the output of the packet sniffer and desperately hoping to see some kind of pattern. An *intrusion detection system* performs

this task for you. These programs combine the ability to record network activity with sets of rules that allow them to flag unauthorized activity and generate real-time warnings.

Exercises:

1. Open Ethereal and start a live capture. Now open your web browser and look for a plain text document to download. Download and save the text file to your hard drive, then close the web browser and end the capture session in Ethereal. Look through the packets captured by Ethereal, paying close attention to the ASCII dump in the bottom pane. What do you see?

If you have access to an email account, try checking your email while Ethereal is performing a capture. What do you see there?

2. Open Ethereal. On the *Capture Options* Screen, make sure that the box marked "Capture packets in promiscuous mode" is checked. This option may allow you to capture packets directed to or coming from other computers. Begin the capture and see what happens. Do you see any traffic that is intended for a computer other than yours?

What do you know about the hardware that connects your computer to the network? Does it connect to the other computers through a switch, a router or a hub? Go to a web search engine and try to find out which piece or pieces of hardware would make it most difficult to capture packets from other computers. What hardware would make it easiest?

3. Go to *www.snort.org*, or use a web search engine to research intrusion detection systems. How are they different from firewalls? What do they have in common with packet sniffers? What kinds of unauthorized activity can they detect? What kinds of activity might they be unable to detect?

7.3 Honeypots and Honeynets

People who like to watch monkeys go to the zoo, because there might be monkeys there. People who like to watch birds put out bird feeders, and the birds come to them. People who like to watch fish build aquariums, and bring the fish to themselves. But what do you do if you want to watch hackers?

You put out a *honeypot*.

Think about it this way – you're a bear. You may not know much (being a bear) but you do know that honey is tasty, and there is nothing better on a warm summer day than a big handful of honey. So you see a big pot full of honey sitting out in the center of a clearing, and you're thinking, 'Yum!" But once you stick your paw in the honey pot, you risk getting stuck. If nothing else, you're going to leave big, sticky paw prints everywhere, and everyone is going to know that someone has been in the honey, and there's a good chance that anyone who follows the big, sticky paw prints is going to discover that it's you. More than one bear has been trapped because of his affection for tasty honey.

A *honeypot* is a computer system, network, or virtual machine that serves no other purpose than to lure in hackers. In a honeypot, there are no authorized users – no real data is stored in the system, no real work is performed on it – so, every access, every attempt to use it, can be identified as unauthorized. Instead of sifting through logs to identify intrusions, the system administrator knows that every access is an intrusion, so a large part of the work is already done.

7.3.1 Types of Honeypots

There are two types of honeypots: *production* and *research*.

Production honeypots are used primarily as warning systems. A production honeypot identifies an intrusion and generates an alarm. They can show you that an intruder has identified the system or network as an object of interest, but not much else. For example, if you wanted to know if bears lived near your clearing, you might set out ten tiny pots of honey. If you checked them in the morning and found one or more of them empty, then you would know that bears had been in the vicinity, but you wouldn't know anything else about the bears.

Research honeypots are used to collect information about hacker's activities. A research honeypot lures in hackers, then keeps them occupied while it quietly records their actions. For example, if – instead of simply documenting their presence – you wanted to study the bears, then you might set out one big, tasty, sticky pot of honey in the middle of your clearing, but then you would surround that pot with movie cameras, still cameras, tape recorders and research assistants with clipboards and pith helmets.

The two types of honeypots differ primarily in their complexity. You can more easily set up and maintain a production honeypot because of its simplicity and the limited amount of information that you hope to collect. In a production honeypot, you just want to know that you've been hit; you don't care so much whether the hackers stay around, However, in a research honeypot, you want the hackers to stay, so that you can see what they are doing. This makes setting up and maintaining a research honeypot more difficult, because you must make the system look like a real, working system that offers files or services that the hackers find interesting. A bear who knows what a honeypot looks like, might spend a minute looking at an empty pot, but only a full pot full of tasty honey is going to keep the bear hanging around long enough for you to study it.

7.3.2 Building a Honeypot

In the most basic sense, a honeypot is nothing more than a computer system which is set up with the expectation that it will be compromised by intruders. Essentially, this means that if you connect a computer with a insecure operating system to the Internet, then let it sit there, waiting to be compromised, you have created a honeypot!

But this isn't a very useful honeypot. It's more like leaving your honey out in the clearing, then going home to the city. When you come back, the honey will be gone, but you won't know anything about who, how, when or why. You don't learn anything from your honeypot, useless you have some way of gathering information regarding it. To be useful, even the most basic honeypot most have some type of intrusion detection system.

The intrusion detection system could be as simple as a firewall. Normally a firewall is used to prevent unauthorized users from accessing a computer system, but they also log everything that passes through or is stopped. Reviewing the logs produced by the firewall can provide basic information about attempts to access the honeypot.

More complex honeypots might add hardware, such as switches, routers or hubs, to further monitor or control network access. They may also use packet sniffers to gather additional information about network traffic.

Research honeypots may also run programs that simulate normal use, making it appear that the honeypot is actually being accessed by authorized users, and teasing potential intruders with falsified emails, passwords and data. These types of programs can also be used to disguise operating systems, making it appear, for example, that a Linux based computer is running Windows.

But the thing about honey – it's sticky, and there's always a chance that your honeypot is going to turn into a bees nest. And when the bees come home, you don't want to be the one with your hand stuck in the honey. An improperly configured honeypot can easily be turned into a launching pad for additional attacks. If a hacker compromises your honeypot, then promptly launches an assault on a large corporation or uses your honeypot to distribute a flood of spam, there's a good chance that *you* will be identified as the one responsible.

Correctly configured honeypots control network traffic going into and out of the computer. A simple production honeypot might allow incoming traffic through the firewall, but stop all outgoing traffic. This is a simple, effective solution, but intruders will quickly realize that is is not a real, working computer system. A slightly more complex honeypot might allow some outgoing traffic, but not all.

Research honeypots – which want to keep the intruders interested as long as possible – sometimes use *manglers,* which audit outgoing traffic and disarm potentially dangerous data by modifying it so that it is ineffective.

Exercises:

Honeypots can be useful tools for research and for spotting intruders, but using them to capture and prosecute these intruders is another question. Different jurisdictions have different definitions and standards, and judges and juries often have varying views, so there are many questions that need to be considered. Do honeypots represent an attempt at entrapment? Is recording a hacker's activities a form of wiretapping?

And on the specific question of honeypots – can it be illegal to compromise a system that was designed to be compromised? These questions have yet to be thoroughly tested.

Discuss your opinions on the legalities of using honeypots for capturing hackers involved in criminal activities. Do you think it would be a useful tool for law enforcement agencies? Is it entrapment? Do you think it constitutes an 'attractive nuisance'? If a hacker comprises a honeypot, who do you think is ultimately responsible?

Further Reading

Netstat

http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/netstat.mspx

General Firewall Information:

http://www.howstuffworks.com/firewall.htm

http://www.interhack.net/pubs/fwfaq/

One of many free firewall programs:

http://www.agnitum.com/index.html

Firewalling for Linux:

http://www.iptables.org/

Packet Sniffing

http://www.robertgraham.com/pubs/sniffing-faq.html

Snort and IDS:

http://www.linuxsecurity.com/feature_stories/feature_story-49.html

http://www.snort.org/docs/lisapaper.txt

Honeypots:

http://www.honeypots.net/honeypots/links/