Hacker Highschool

SECURITY AWARENESS FOR TEENS

LESSON 3
PORTS AND PROTOCOLS

Hacker Highschool
SECURITY AWARENESS FOR TEENS

ISECOM

WARNING

Table of Contents

Contributors

Marta Barceló, ISECOM

Pete Herzog, ISECOM

Glenn Norman, ISECOM

Chuck Truett, ISECOM

Bob Monroe, ISECOM

Kim Truett, ISECOM

Gary Axten, ISECOM

Marco Ivaldi, ISECOM

Simone Onofri, ISECOM

Greg Playle, ISECOM

Tom Thomas, ISECOM

Mario Platt

Ryan Oberto, Johannesburg South Africa

Vadim Chakryan, Ukraine

Peter Houppermans

ISECOM

Introduction and Objectives

In the far depths of the past, before there was an Internet, electronic communication was pure voodoo. Every computer manufacturer had their own idea about how machines should talk over a wire. And nobody even considered the possibility that a Wang computer might communicate with a Burroughs machine.

The world changed when scientists and students experienced the joy of using a terminal to access a mainframe computer. The famous IBM PC arrived, and quickly owners wanted to access that mainframe from their personal computer. Soon modems were making dial-up connections and users were working in terminal emulators. Networking had graduated to a Black Art, and the insiders were called (really) **gurus**.

The world shifted again dramatically when the Internet, which started as a military project, was opened to the public. Networking had always been local, meaning confined to one office or at most one campus. How were all these different systems going to talk?

The answer was "wedge" a universal address system into existing networks, a system we generally call **Internet Protocol (IP)**. Think about it this way: imagine your friend overseas sends you a package. That package may travel by plane, train or automobile, but you don't really need to know the airline schedule or the location of the nearest train station. Your package will eventually arrive at your street address, which is ultimately the only thing that matters. Your **IP address** is a lot like this: packets may travel as electrons, beams of light or radio waves, but those systems don't matter to you. All that matters is your IP address, and the IP address of the system with which you're talking.

One thing that complicates this idea in the real world is that more than one person may be living at a single address. In the networking world, that's what's happening when one server provides for instance both regular HTTP and secure HTTPS, as well as FTP. Notice the P at or near the end of those acronyms? That's always a dead giveaway for **protocol,** which is just another way of saying "a type of communication."

This lesson will help you understand how protocols and their ports work in Windows, Linux, and OSX. You'll also become familiar with several utilities (some of which have already been introduced in the previous lesson) that explore your system's networking capabilities.

At the end of the lesson you should have a basic knowledge of:

- the concepts of networks and how communication takes place
- IP addresses
- ports and protocols

Basic Concepts of Networking

The starting point for networking is the local area network (**LAN**). LANs let computers in a common physical location share resources like printers and drive space, and **administrators** control that access. Sections below describe common network devices and topologies.

Devices

Going forward in your career as a hacker, you're going to see a lot of network diagrams. It's useful to recognize the most common symbols:

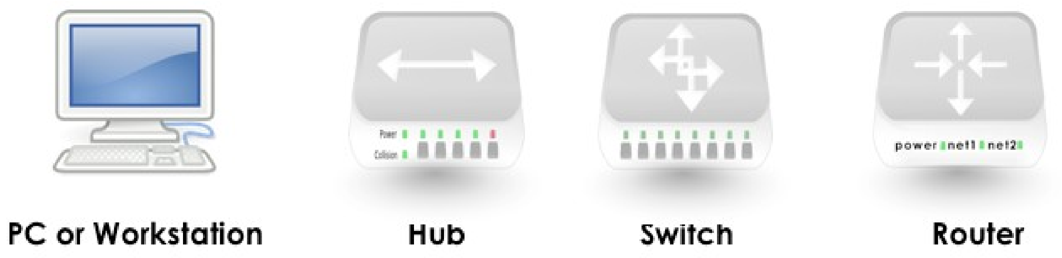

Figure 3.1: Common Network Symbols

A **hub** is like an old-fashioned telephone party line: everybody is on the same wire, and can hear everyone else's conversations. This can make a LAN noisy fast.

A **switch** is better: it filters traffic so that only the two computers talking to each other can hear the conversation. But like a hub, it's used only on a LAN.

A **router** sits between LANs; it's used to access other networks and the Internet, and it uses IP addresses. It looks at the packets being sent and decides which network those packets belong on. If the packet belongs on the "other" network, like a traffic police, it sends the packet where it belongs.

Topologies

A **topology** is just another way of saying "the way we connect it". The kind of decisions we make regarding our topology can affect positively as well as negatively in the future, depending on technologies being used, technological and physical constraints, performance and security requirements, size and nature of the organization, etc.

A LAN's physical structure can look like any of the following physical topologies:

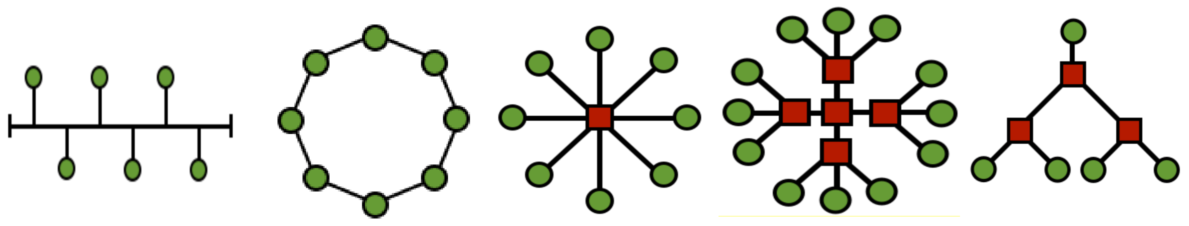

Figure 3.2: Toplogies

In a **bus** topology, all the computers are connected to a single cable, and each computer can communicate directly with any of the others. But break any part of the bus, and everyone's off the network.

In the **ring** configuration, each computer is connected to the following one, and the last one to the first, and each computer can only communicate directly with the two adjacent computers.

Bus topologies are rarely used nowadays. Ring technologies are often used at the inter-state level, usually with two counter rotating rings, sending traffic in opposite directions, for reliability and fault tolerance.

In the **star** topology, none of the computers are directly connected with others. Instead they are connected through a hub or switch that relays information from computer to computer.

If several hubs or switches are connected to each other, you get an **extended star** topology.

In a star or extended star topology, all the central points are **peers,** that is, they're essentially equals. This is the most common LAN topology today.

However, if you connect two star or extended star networks together using a central point that controls or limits traffic between the two networks, then you have a **hierarchical** network topology. This is the topology usually deployed in bigger enterprises.

Game On: Leaving the Back Door Open

In the sun-baked heat of the summer, Jace was happy to help the local air-conditioned police department set up their small network. They paid her with cookies, time away from the heat, conversation and the opportunity to install backdoors. Crawling under steel work desks that hadn't been moved in decades, Jace had found the cruddiest hidden spot to hide a wifi access point. She'd just plugged it in and sprinkled trash on top of it, and was dragging a roll of Ethernet cable to the wall ports she'd installed earlier.

A heavy hand slapped the desk above her. Jace kicked metal and yelled "Ow! My head!" then added, "you sure you don't want me to set up your server?"

The cop cleared his throat and tried to put on a Dork Professor voice. "Well I would, but I'm not sure how the flux ray resistor would hold up to the micro-channel cross feeds. Especially when the full moon falls on the last Tuesday of the month."

Jace flapped her feet in mock teen irritation. "Apparently you have no problem achieving quantum levels of baloney. And when do I get my cookies, Officer Kickam?"

"Jace please, please call me Hank. You make me feel like an old man when you call me Officer Kickam." He tried to sound hurt but she knew social engineering when she heard it: he was really trying to distract her from the cookies.

"Hank, hate to break the news to you, but you *are* an old man."

"Ouch, that hurt. I am not old, I am distinguished," he countered, considering his highly polished black police shoes as Jace's tattered sneaks disappeared under the heavy desk. Then cinnamon brown eyes and a face covered in spider webs emerged. Jace still had a reel of cable under one arm. Hank helped her up and brushed the bug webs off her face and shoulders.

"Help, police brutality," Jace teased.

"Hostile criminal," Hank returned. "So educate me on your diabolical plan here," the hairy muscled lawman asked in what almost sounded to Jace like a pleading tone.

That felt good, so she asked, "Are you sure you want to know about this networking stuff?" He nodded eagerly. Jace thought: *bobble head*.

"Okay, what I did was design a network topography, like a map that shows where all the equipment, computers, hubs, jacks, switches, routers and firewalls will go. You can't start a project like this without a map," she said, glancing up at the cop. "It's all about making sure every node can talk to every other node, with no single point of failure. So, like, a bus architecture stinks, because if one node in the bus goes down, everyone else does too." Hank nodded so Jace continued.

"Think like networking is this cop shop, uh, police station, and someone just brought in a suspect. Every cop deserves their fair turn to beat up on the guy without hogging someone else's time. If the victim, I mean suspect, is moved to another cell, all the cops who still need to beat on the dude have to know where he went."

"Oh Jace, you're gonna start looking like you need a good beating too if you keep talking like that about us peace officers." Hank pulled up his gun belt and sucked in his slight belly.

Jace snorted a laugh. "So the suspect is a data packet and you police thugs are the network devices. And every device, a switch, a router, firewall, another server or whatever, needs to know that the data packet gets dealt with. You know, pummeled with police batons. I think you called it giving someone a wood shampoo."

Hank rolled his eyes and groped for the baton that he didn't have on him.

Giggling, Jace brought up the reel of cable like a shield. "Hey, I've got a spool of wire and I'm not afraid to use it. Put down the cup of coffee and nobody gets hurt." Off balance and laughing, Jace flopped onto Hank, who didn't budge. *Wow, this guy is a total rock*, she realized. The hand he laid on her shoulder reminded her of … something.

She stood a little too quickly, reddening. "So there's two kinds of devices. Smart devices and dumb ones. Just like cops." Four approaching uniforms appeared at exactly the wrong time to hear the "dumb ones, just like cops." Lamely Jace continued, "Smart devices remember everything they do. They keep logs of their activities."

"And the dumb ones? Like cops?" asked the Chief of Police.

Game Over

The TCP/IP (DoD) Model

TCP/IP was developed by the **DoD (Department of Defense)** of the United States and **DARPA (Defense Advanced Research Project Agency)** in the 1970s. TCP/IP was designed to be an open standard that anyone could use to connect computers together and exchange information between them. Ultimately, it became the basis for the Internet.

Generally, the simplest form of the TCP/IP model is called the **DoD Model**, and that's where we'll start.

Layers

The simple DoD model defines four totally independent layers into which it divides the process of communication between two devices. The layers that pass information are:

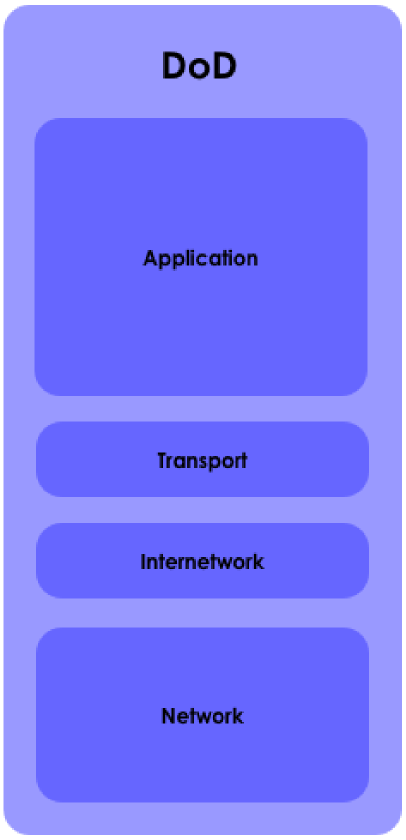

Figure 3.3: The DoD Model

Application

The application layer is exactly what you probably think it is: the layer where applications like Firefox, Opera, email clients, social networking sites, instant messaging and chat applications work. Actually quite a few applications access the Internet: some office applications, for instance, connect to online collections of clip art. The application layer

creates the payload that all the other layers carry. A good analogy is a postal system. The application creates the package that it wraps with instructions on how the package should be used. Then it hands off the package to the mail room: the Transport layer.

Transport

The transport layer sets up network connections, which are called **sessions.** In the world of the Internet, the primary protocol at the Transport layer is **TCP, the Transmission Control Protocol**. TCP adds another "wrapping" to the outside of the package, with instructions about which package it is (say, 1 of 3), how to make sure the package got there, and whether the package is intact.

Let's say you're going to email a letter to your mother. The letter may be tiny or huge, but it's too big to send over the Internet in one piece. Instead, TCP breaks up that letter into **segments,** small chunks that are consecutively numbered, with a little bit of error-checking code at the end. If a packet gets corrupted in transit, TCP requests a retransmission. At the receiving end, TCP puts the pieces back together in the correct order and your mother gets the letter in her email.

But don't forget that TCP isn't the only game in town: **UDP** also functions at this layer, and in particular it does NOT create sessions. It just shoots a stream of **datagrams**, which are similar to segments, but UDP never checks if you've received it.

Whether TCP or UDP, all traffic is assigned to specific **port numbers** at this layer.

Internetwork

This layer adds information about the source and destination addresses, and where the **packet** begins and ends. It's like a delivery company that gets packages to the correct address. It doesn't care if all the packets make it, or if they are intact; that's the Transport layer's job. The major protocol at this level is, appropriately, **IP (Internet Protocol)**. And this is the layer that uses IP addresses to get packets to the right place by the best route.

Network Access

This layer is the low-level physical network that you use to connect to the Internet. If you're dialing up, we're sorry, and you're using a simple **PPP** connection. If you have **DSL** you may be using **ATM** or **Metro Ethernet**. And if you have cable Internet you're using a **DOCSIS** physical network. It doesn't matter what kind you use, because TCP/IP makes everything work together. The network access layer consists of the Ethernet cable and **network interface card (NIC)**, or the wireless card and access point. It handles the lowest level ones and zeroes (bits) as they go from one point to another.

Feed Your Head: See "The OSI Model"

See "The OSI Model" at the end of this lesson for an alternative take on network modeling.

Protocols

So now you're connected to the Internet. That seems simple enough, but consider the usual situation you're in: you are conducting innocent, important research on the Internet, while your dear brother or sister is wasting time streaming a movie. Why don't these two streams of traffic get mixed up? How does the network tell them apart?

The answer is **protocols,** which are like languages that different kinds of traffic speaks. Web traffic uses one protocol, file transfers another one, and email a different one still. Like all things digital, protocols don't really use names down at the network level; they use IP addresses and **port numbers**.

Application layer protocols

FTP or *File Transfer Protocol* is used for the transmission of files between two devices. It uses one port to deliver data, and another port to send control signals ("I got the file! Thanks!"). The most commonly used ports are 20 and 21 (TCP).

HTTP or *Hyper-Text Transfer Protocol* is used for web pages. This traffic usually uses TCP port 80. **HTTPS** is a secure variant that encrypts network traffic, usually on TCP port 443.

SMTP or *Simple Mail Transfer Protocol* is the protocol that sends email. Its TCP port is 25.

DNS or *Domain Name Service* is how a domain like ISECOM.org gets mapped to an IP address like 216.92.116.13. It uses port is 53 (UDP).

Transport layer protocols

TCP and **UDP** are the two main protocols used by the transport layer to transfer data.

TCP or Transmission Control Protocol establishes a logical connection (a **session**) between two hosts on a network. It sets up this connection using the three-way handshake.

1. When my computer wants to connect with yours, it sends a **SYN** packet, which is basically saying, "Let's synchronize clocks so we can exchange traffic with timestamps."
2. Your computer (if it's going to accept the connection) responds with a **SYN/ACK** acknowledgment packet.
3. My computer seals the deal with an **ACK** packet, and we're connected.

But this happens only with TCP. Instead, **UDP or User Datagram Protocol** is a transport protocol that doesn't even care if you have a connection. It's like a fire hose: if you catch the stream you catch it, and if you don't, you don't. This makes UDP very fast, so it's useful for things like streaming voice and video, where missing a single frame doesn't matter much or online gaming, where missing a single frame doesn't matter much (depending on which side of the bullet you are).

Internet layer protocols

IP or Internet Protocol serves as a universal protocol to allow any two computers to communicate through any network at any time. It's like the postal carrier who delivers mail; all it does is get packets to their destination address.

Internet Control and Management Protocol (ICMP)

ICMP is the protocol that the network devices and network administrators use to troubleshoot and maintain the network. It includes things like **ping** (Packet InterNet Groper) and similar commands that test the network and report errors. Because people have used things like ping floods to bring down hosts and networks, most systems limit ICMP to one response per second.

To summarize, ports and protocols come together like this:

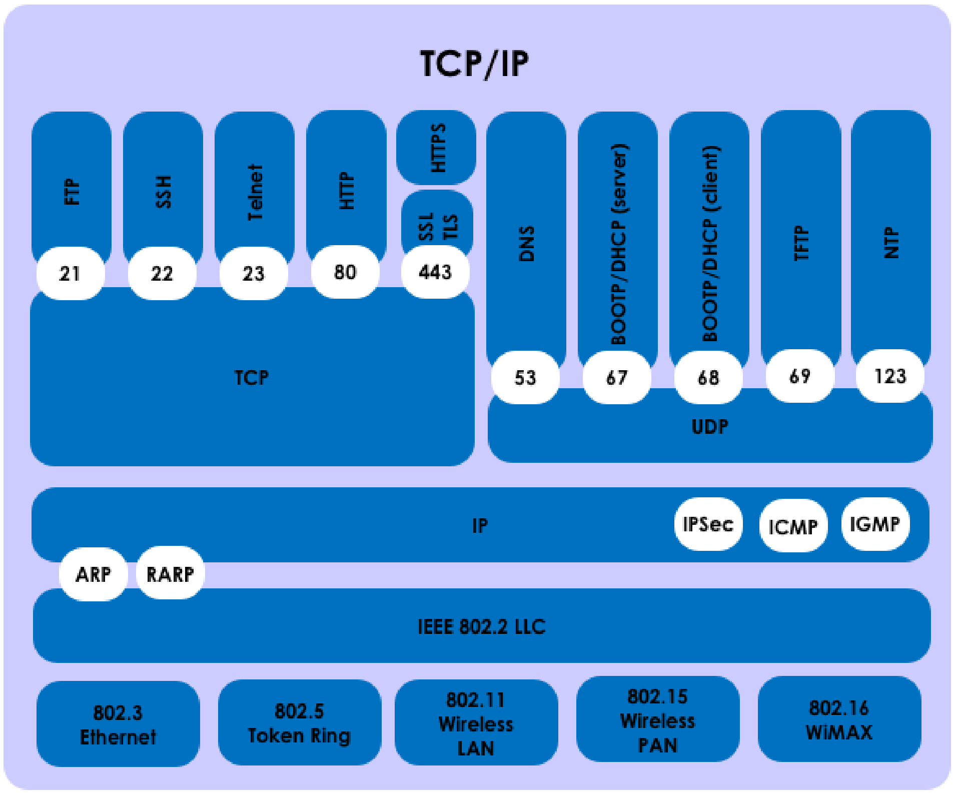

Figure 3.4: The TCP/IP Stack

IPv4 Addresses

Domain names are handy for humans, because we're good at remembering names like ISECOM.org. But networks don't actually understand them; they only understand numeric IP addresses. So when you ask for ISECOM.org, your computer does a quick lookup using **DNS (Domain Name Service)** to find the corresponding IP address.

IP addresses are like street addresses. If you want mail, you have to have one. **IPv4** addresses consist of 32 bits that are divided in four 8-bit **octets,** which are separated by dots. This means that there are 2^{32} (or 4,294,967,296) unique addresses on the Internet under IPv4. Part of the IP address identifies the network, and the remainder of the IP address identifies the individual computers on the network. Think of these parts as the country/city (network) portion of the address and the street (host) portion of the address.

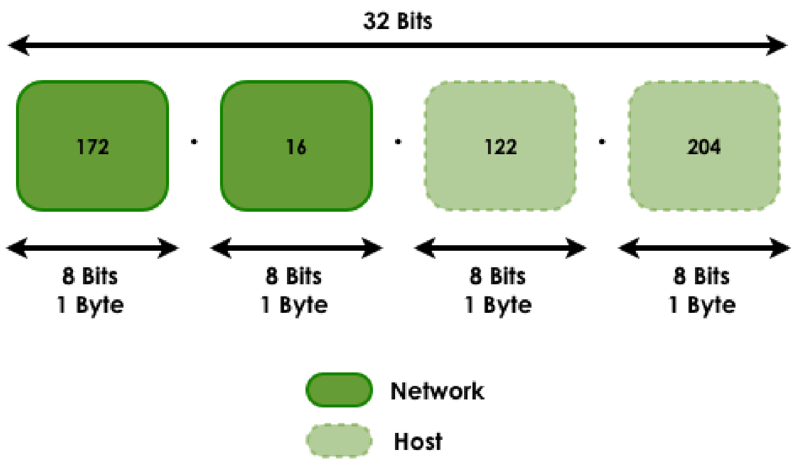

Figure 3.5: Network Number and Host ID

Returning to the postal service analogy: IP is the delivery truck that gets the packet to the correct post office. TCP is the outer wrapper with the list of how many packages are in a shipment, and which one this is (say, number 3 of 65). The host-level addresses are the particular house (computer) for which the packet is destined.

There are both public and **private (non-routable) IP addresses**. Private IP addresses are used by private networks; routers won't allow these addresses onto the Internet.

IP addresses within a private network should not be duplicated within that network, but computers on two different – but unconnected – private networks could have the same IP addresses. The IP addresses that are defined by IANA, the Internet Assigned Numbers Authority, as being available for private networks (see RFC 1918) are:

10.0.0.0 through 10.255.255.255	(Class A)
172.16.0.0 through 172.31.255.255	(Class B)
192.168.0.0. through 192.168.255.255	(Class C)

Classes

IP addresses are divided into classes based on what portion of the address is used to identify the network and what portion is used to identify the individual computers.

Depending on the size assigned to each part, more devices will be allowed within the network, or more networks will be allowed. The existing classes are:

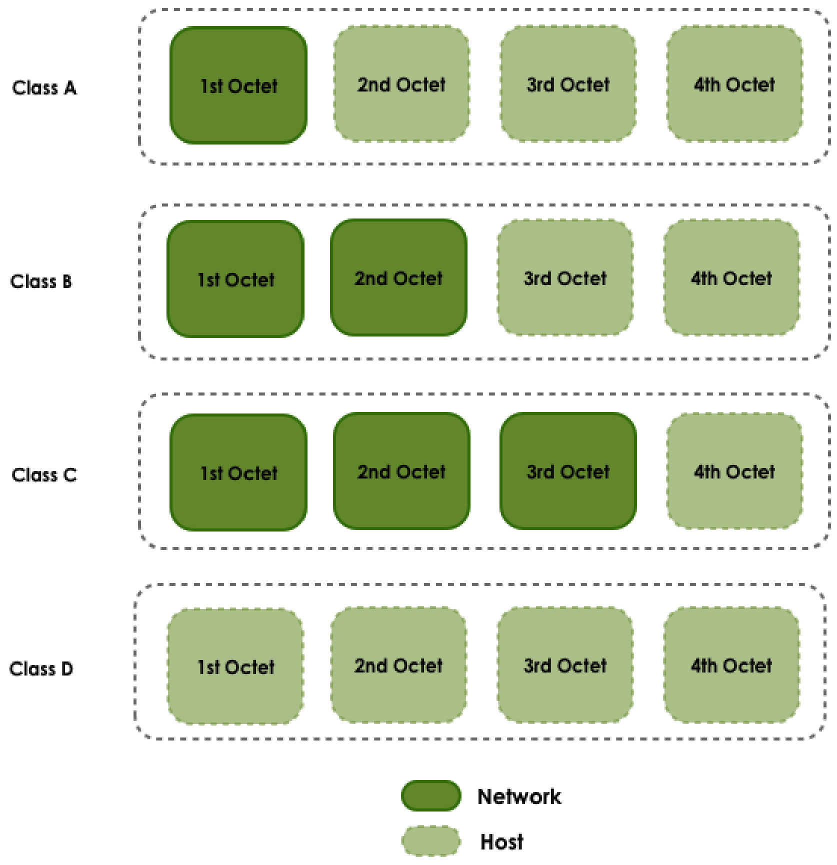

Figure 3.5: IP Class Divisions

Class A: The first bit is always zero, so this class includes the addresses between 0.0.0.0 (which, by convention, is never used) and 126.255.255.255. Note: the addresses of 127.x.x.x are reserved for the services of loopback or localhost (see below).

Class B: The first two bits of the first octet are '10', so this class includes the addresses between 128.0.0.0 and 191.255.255.255.

Class C: The first three bits of the first octet are '110', so this class includes the addresses between 192.0.0.0 and 223.255.255.255.

Class D: The first four bits of the first octet are '1110', so this class includes the addresses between 224.0.0.0 and 239.255.255.255. These addresses are reserved for group multicast implementations.

The remaining addresses are used for experimentation or for possible future allocations.

The **mask (or netmask)** is used to mark these class splits. Down in binary, a '1' bit shows the part containing the network identification and a '0' bit represents the part that identifies the individual host. The default netmasks for the first three classes are:

255.0.0.0	(Class A)
255.255.0.0	(Class B)
255.255.255.0	(Class C)

This is actually pretty slick, since networks that use default classes will mask one octet if they're Class A, two octets for Class B and three octets for Class C. Using default classes is handy – but not everyone's doing it.

What all this means is that to identify a host, you'll need both the IP address and a network mask:

IP: 172.16.1.20
Mask: 255.255.255.0

Loopback Addresses

IP addresses 127.0.0.1 through 127.255.255.254 are reserved to be used as **loopback** or localhost addresses, that is, they refer directly back to the local computer. Every computer has a localhost address of 127.0.0.1, therefore that address cannot be used to identify different devices.

There are also other addresses that cannot be used. These are the **network address** and the **broadcast address**.

Network Addresses

The network address is basically the network part of an IP address, **with zeroes where the host part would be**. This address cannot be given to a host, because it identifies the whole network, not just one host.

IP: 172.16.1.**0**
Mask: 255.255.255.0

Broadcast Addresses

The broadcast address is basically the network part of an IP address, **with ones where the host part would be**. This address can't be used to identify a specific host, because it's the address that all hosts listen to (of course that's what broadcast means: everybody listens).

IP: 172.16.1.**255**
Mask: 255.255.255.0

Ports

Both TCP and UDP use **ports** to exchange information with applications. A port is an extension of an address, like adding an apartment or room number to a street address. A letter with a street address will arrive at the correct apartment building, but without the apartment number, it won't get to the correct recipient.

Ports work in the same way. A packet can be delivered to the correct IP address, but without the associated port, there is no way to determine which application should act on the packet. A port number is also a 16 bit number, which means it can have decimal values between 0 and 65535 (2 to the power of 16).

Another way to think about this would be: every computer is a post office. Each application has its own post office box; no two applications should share the same post office box. The port number is that post office box number.

Port numbers make it possible to have multiple streams of information going to one IP address, where each one is sent to the appropriate application. The port number lets a service running on a remote computer know what type of information a local client is requesting and what protocol is used to send that information, all while maintaining simultaneous communication with a number of different clients.

For example, if a local computer attempts to connect to the website www.osstmm.org, whose IP address is 62.80.122.203, with a web server running on port 80, the local computer would connect to the remote computer using the **socket address**:

62.80.122.203:80

In order to maintain a level of standardization among the most commonly used ports, IANA has established that the ports numbered from 0 to 1024 are to be used for common, **privileged** or **well-known services.** The remaining ports – up through 65535 – are used for dynamic allocations or particular services.

The most commonly used (well-known) ports – as assigned by the **IANA** – are listed here:

Port Assignments		
Number	**Keywords**	**Description**
5	rje	Remote Job Entry
0		Reserved
1-4		Unassigned
7	echo	Echo
9	discard	Discard
11	systat	Active Users
13	daytime	Daytime
15	netstat	Who is Up or NETSTAT
17	qotd	Quote of the Day

Port Assignments

19	chargen	Character Generator
20	ftp-data	File Transfer [Default Data]
21	ftp	File Transfer [Control]
22	ssh	SSH Remote Login Protocol
23	telnet	Telnet
25	smtp	Simple Mail Transfer
37	time	Time
39	rlp	Resource Location Protocol
42	nameserver	Host Name Server
43	nicname	Who Is
53	domain	Domain Name Server
67	bootps	Bootstrap Protocol Server
68	bootpc	Bootstrap Protocol Client
69	tftp	Trivial File Transfer
70	gopher	Gopher
75		any private dial out service
77		any private RJE service
79	finger	Finger
80	www-http	World Wide Web HTTP
95	supdup	SUPDUP
101	hostname	NIC Host Name Server
102	iso-tsap	ISO-TSAP Class 0
110	pop3	Post Office Protocol - Version 3
113	auth	Authentication Service
117	uucp-path	UUCP Path Service
119	nntp	Network News Transfer Protocol
123	ntp	Network Time Protocol
137	netbios-ns	NETBIOS Name Service
138	netbios-dgm	NETBIOS Datagram Service
139	netbios-ssn	NETBIOS Session Service
140-159		Unassigned
160-223		Reserved

Encapsulation

When a piece of information – an e-mail message, for example – is sent from one computer to another, it is subject to a series of transformations. The application layer generates the data, which is then sent to the transport layer.

The transport layer takes this information, breaks in into segments and adds a header to each one, which contains ports, unique number of the segment and other session information.

Then the segment is passed to Network layer where another header is added, containing the source and destination IP addresses and more meta-information.

The next layer, which in most local networks is supplied by Ethernet, adds yet another header, and so on. This procedure is known as **encapsulation.**

Each layer after the first makes its data an encapsulation of the previous layer's data, until you arrive at the final layer, in which the actual transmission of data occurs. So encapsulation looks like this:

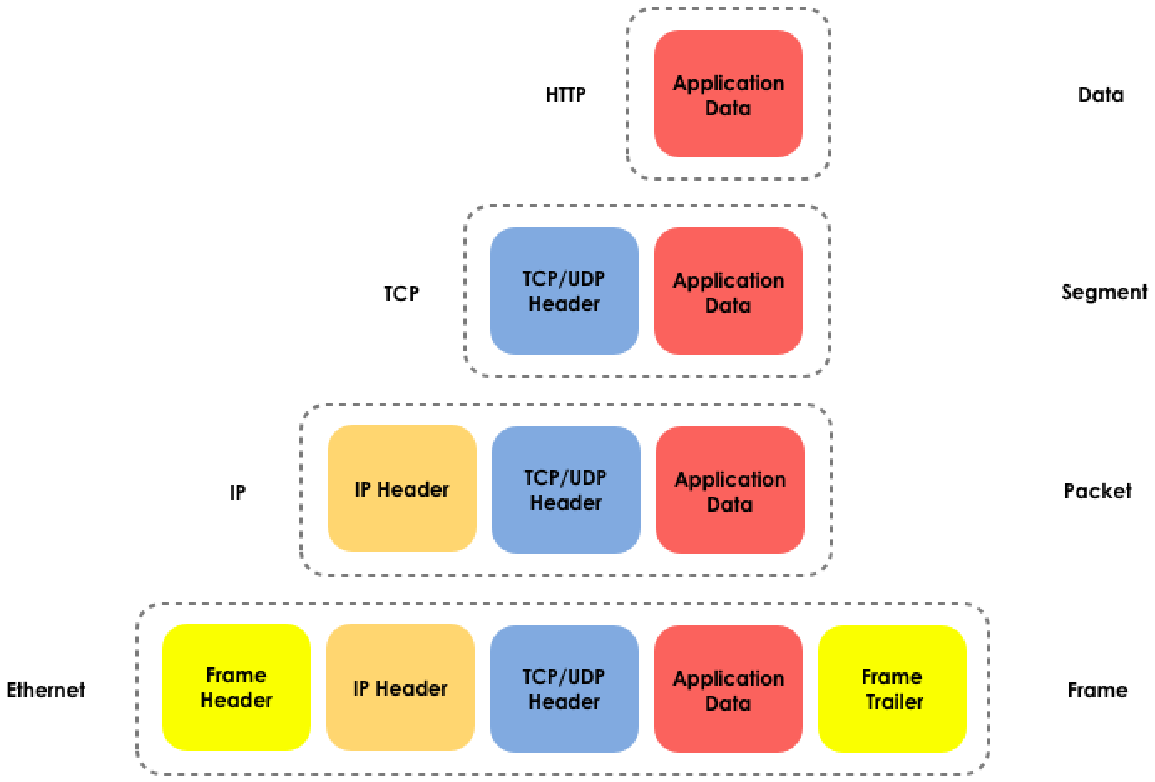

Figure 3.6: Encapsulation

When the encapsulated information arrives at its destination, it must then be de-encapsulated. As each layer passes information to the next layer up the stack, it removes the information contained in the header placed there by that lower layer.

The final bit of information in this great addressing scheme is the absolutely unique address of the computer's NIC: the **Media Access Controller (MAC) address**. This address is usually displayed as six two-character **hexadecimal** numbers separated by colons or hyphens (dashes). It is the physical address of the network card and supposedly can't be changed (actually, there are ways to change it, but exactly how is for you to figure out). A MAC address looks like this:

```
00-15-00-06-E6-BF
```

Exercises

3.1. Using commands you learned in Lessons 1 and 2, get your IP address, netmask, DNS hostname, and MAC address. Compare these with your partners. What seems similar and what is different? Given the IP address scheme the network is using, is this a private or a public network?

3.2. netstat

The **netstat** command tells you your network statistics: with whom you're connected, how long networking has been up, and so forth. In Linux, Windows or OSX you can open a command line interface and type:

```
netstat
```

In the CLI window, you'll see a list of established connections. If you want to see the connections displayed in numeric form, type:

```
netstat -n
```

To see the connections and the active (listening, open) ports, type:

```
netstat -an
```

To see a list of other options, type:

```
netstat -h
```

In the netstat output, look for the columns listing the local and remote IP addresses and the ports they're using:

```
Proto Recv-Q Send-Q  Local Address        Foreign Address     (state)
tcp4     0       0  192.168.2.136.1043   66.220.149.94.443   ESTABLISHED
```

The ports are the numbers after the regular IP address; they may be separated by dots or colons. Why are the ports used by the remote address different from the ports used by the local address?

Open several browser windows or tabs to various websites, then run netstat again.

If there are several tabs open, how does the browser know which information goes to which tab?

Why is it that when a web browser is used, no listening port is specified?

What protocols are used?

What happens when one protocol gets used in more than one instance?

3.3. My First Server

To perform this exercise, you must have the **netcat (nc)** program. BackTrack includes it by default, as does OSX, but you can download installers for various operating systems.

1. In a CLI window, type:

```
        nc -h
```

This displays the options that are available in netcat.

To create a simple server, In Linux or Windows type:

```
        nc -l -p 1234
```

or in OSX type:

```
        nc -l 1234
```

You just started a server listening to port 1234.

2. Open a second CLI window and type:

```
        netstat –a
```

This should verify that there is a new service listening on port 1234.

To communicate with a server, you have to have a client! In your second CLI window type:

```
        nc localhost 1234
```

This command makes a connection with the server that's listening to port 1234. Now, anything that is written in either of the two open CLI windows can be seen in the other window.

Consider the implications. How could someone abuse this capability to exploit your machine?

Netcat sends all its traffic in the clear. Is there a secure alternative?

3. Stop your server by going back to the first CLI window and typing Control-C.

4. Now create a simple text file named *test* containing the text, "Welcome to my server!"

Once you've done that, look at the following command and translate it for the instructor: what does each part do? Then, in your first CLI window, type:

```
        nc -l -p 1234 < test
```

From another CLI window, connect to the server by typing:

```
        nc localhost 1234
```

When the client connects to the server, you should see the output of the file *test*.

What protocol has been used to connect with the server?

Does netcat allow you to change this? If so, how?

Feed Your Head: The OSI Model

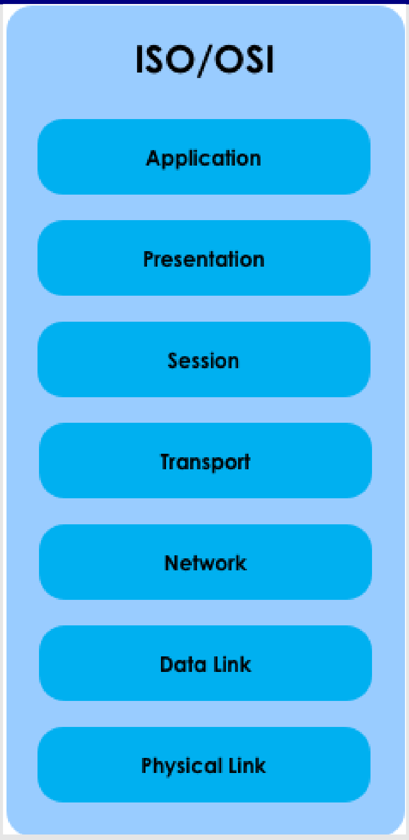

Figure 3.7: The ISO/OSI Model

The **OSI Model** was developed in the 1980s (about ten years after the TCP/IP Model) by **ISO**, the International Standards Organization. OSI stands for **Open Systems Interconnection**, and it was an attempt to standardize networking architecture that came from an organization that wasn't really involved in the development of networking.

The OSI Model is a layered model with a handful of simple rules. Similar functions are grouped together in the same layer, and (please do not forget this) every layer is serviced by the layer **beneath** it, and serves the layer **above** it.

This layered model is a good idea, because since every layer (in theory) does its own communication, new developments in any one layer don't break any of the other ones. This feature alone may explain the Internet boom we've had since 2000, with new applications and services appearing almost each day.

Besides the two rules of this OSI model we've already discussed (similar functions are grouped, and every layer is serviced by the layer beneath it and serves the layer above it) this standard has one more strict rule. Every layer involved in communication from one computer communicates directly with the same layer on the other computer. This means that when you type www.google.com in your browser, there is a direct interaction between your computer's Layer 7 interface (your web browser) and Google.com's web servers (also a Layer 7 interface), and that the same can be said of any other layer.

So let's first define what are the OSI model layers and their respective responsibilities.

Application Layer	Responsible for direct interaction between applications and the user interface to the application, for instance the use of a web browser like IE or Firefox.
Presentation Layer	Responsible for guaranteeing that data is exchanged in a way that is comprehensible between both parties. In some services that use a form of encryption, the encryption happens at the presentation layer.
Session Layer	Responsible for dialogue control between two computers. Basically it establishes, manages and terminates all connections that happen between the computers.
Transport Layer	Provides transparent transfer of data between computers, providing reliable data transfer services to the upper layers. This means that it is responsible for assembling all data in smaller portions that can be carried reliably on a data network. If a packet is lost or not received, it is the transport layer's job to make sure that that single packet is retransmitted and then reassembled in the correct order.
Network Layer	This layer is responsible for the addressing part of the connection. Not only on ensuring that each address is *unique* on the network, but also on making sure that whatever path is available (whether a good or a bad one), it always delivers the information where it needs to go, and that our information will be sent from hop to hop until it reaches its final destination.
Data Link layer	The data link layer was designed to deal with ensuring the physical layer can recover from errors that might happen and to deal with different connecting mediums. Basically it prepares (encapsulates) data so that it can be transmitted over whatever physical means are necessary (radio waves, fiber-optic cable, copper).
Physical layer	This layer defines the physical specifications of the devices and what needs to be done in order for the information to be transmitted over the selected medium. For a WiFi connection, this is a radio signal; for a fiber connection it's the light being sent; or for a copper connection the electronic signal being sent on the wire.

These seven layers comprise everything that is needed for the reliable communication between computers.

Here's how the different models we've discussed look side-by-side:

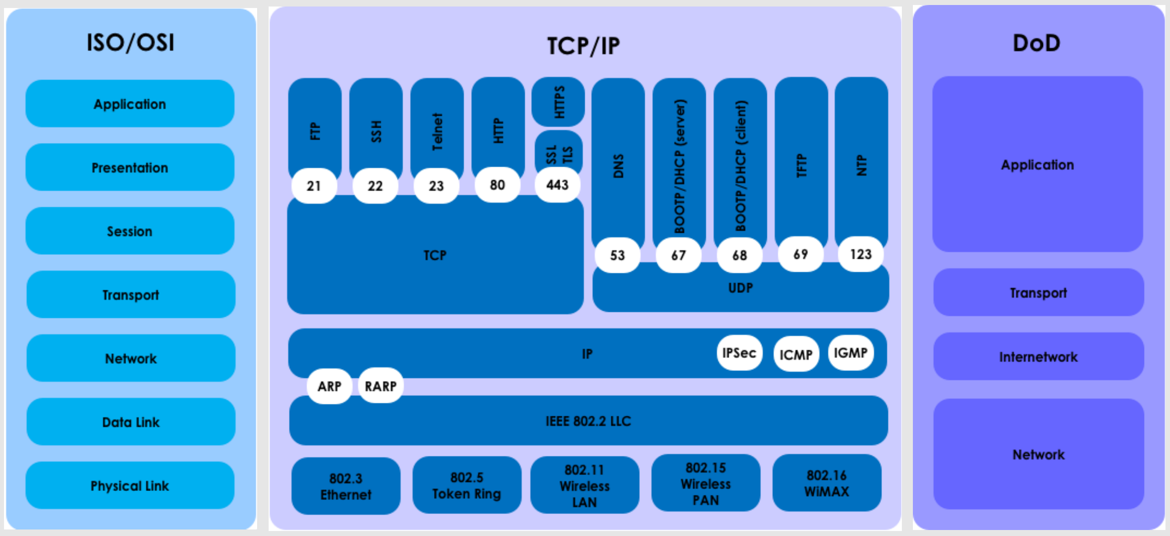

Figure 3.8: Networking Models Compared

Today's teens are in a world with major communication and productivity channels open to them and they don't have the knowledge to defend themselves against the fraud, identity theft, privacy leaks and other attacks made against them just for using the Internet. This is the reason for Hacker Highschool.

The Hacker Highschool project is the development of security and privacy awareness learning materials for junior high and high school students.

Hacker Highschool is a set of lessons and a practical means of making hackers. Beyond just providing cybersecurity awareness and critical Internet skills, we need to teach the young people of today how to be resourceful, creative, and logical, traits synonymous with hackers. The program contains free security and privacy awareness teaching materials and back-end support for teachers of accredited junior high, high schools, and home schooling. There are multiple workbooks available in multiple languages. These are lessons that challenge teens to be as resourceful as hackers, including safe Internet use, web privacy, researching on the internet, avoiding viruses and Trojans, legalities and ethics, and more.

The HHS program is developed by ISECOM, a non-profit, open-source research group focused on security awareness and professional security development and accreditation.

Hacker Highschool
SECURITY AWARENESS FOR TEENS

ISECOM