Hacker Highschool

SECURITY AWARENESS FOR TEENS



LESSON 4
PLAYING WITH DAEMONS

HACKING IS LEARNING
www.hackerhighschool.org

ISECOM

WARNING

The Hacker Highschool Project is a learning tool and as with any learning tool there are dangers. Some lessons, if abused, may result in physical injury. Some additional dangers may also exist where there is not enough research on possible effects of emanations from particular technologies. Students using these lessons should be supervised yet encouraged to learn, try, and do. However ISECOM cannot accept responsibility for how any information herein is abused.

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 HHS web page at http://www.hackerhighschool.org/licensing.html.

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

Table of Contents

Contributors

Pete Herzog, ISECOM
Marta Barceló, ISECOM
Chuck Truett, ISECOM
Kim Truett, ISECOM
Marco Ivaldi, ISECOM
Bob Monroe, ISECOM
Jaume Abella, ISECOM
Greg Playle, ISECOM
Guiomar Corral, Barcelona
Ashar Iqbal

Introduction

There are thousands of different human languages and dozens of dialects within some of them. You may know several languages yourself, but the odds of you being able to travel across the world and speak to everyone you meet are slim to none.

Yes, you could argue that mathematics is a universal language or music speaks to everyone, but let's be realistic. Try and order a glass of soda with just a little bit of lemon and a scoop of ice cream using those "universal languages" and see how far you get.

If you happen to be in a country where you do not speak the language, please send ISECOM a video of yourself using bagpipes or a saxophone to order a soda. We really want to see that! We may not want to hear it, but we sure do want to see it.

But every day millions of people communicate with one another using a single common language over the Internet. Humans may not all speak the same language; however, our computers and networks do.

> The model we use in contemporary networks is the **client-server model.** Physical computers (**hosts** or **servers**) offer **services** (in UNIX they're called **daemons**: **disk access and execution monitors** – now go impress someone). Think of a web server: it serves up a web page when you ask for it. No mystery.

But actually "you" don't request that page; your web browser does, meaning it's a **client** (or more formally, your computer is). And your computer can be a server, too, at the same time. That's the beauty of networking: you do this for me; I do that for you.

Multiply this model a million times, and you have the Internet. Consider this: millions of computers are offering some kind of service. What does it take to be a client? And is it possible to **subvert** all of this? (Go look up that word if you're not dead certain what it means. This is a hacker course, after all.)

Ready or not, let's dive in.

Services

You have a computer, and you know that there's useful information on it, or you may participate in the common hallucination that you have nothing of digital value. You also know that other people, millions of other people, also have computers and their computers also may have useful information, not to mention handy resources like processors and RAM and disk space and bandwidth.

Now, you can assume that these other people, and these other computers, may very likely have information of interest to someone. The only problem is how to get at all that useful information.

Computers communicate with each other easily through ports, using the protocols we discussed in Lesson 3, but that doesn't really let you read the streams of binary data that computers exchange (unless you have some serious extra time on your hands). You need a way for your computer to get data, interpret it for you, and present it in some form you can use.

The way computers transfer data is through **network services**, or simply **services**. These services allow you to view web pages, exchange email, chat, and interact with remote computers. These services are mapped to port numbers.

Your computer, the **local computer**, uses programs called **clients** to interpret the information that you receive. You might receive information from a **server** (providing a service/running a daemon), over a **Tor** network, from **Torrent seeders** or over **peer-to-peer** networks.

> Of course, your computer can provide services to other remote computers as well, meaning it's a data server or service provider. If you happen to have malware on your computer, you may be providing quite a few services that you don't know about.

Examples of clients include web browsers, email clients, chat programs, Skype, Tor clients, Torrent clients, RSS and so on. These are the applications in the **application layer** of the TCP/IP protocol stack. At the application layer, all the data that is transmitted, encapsulated, encrypted, decrypted, addressed and so forth by the lower layers is turned into something that you, the user, can read and understand.

HTTP and the Web

When we talk about "the Internet," most people are actually thinking of the **World Wide Web**. The World Wide Web, or just the **Web**, isn't the Internet, it's just a small portion of the services available. Usually it just involves viewing web pages through a browser.

> The actual Internet, by the way, consists of all the computers, routers, wires, cable and wireless systems that move all kinds of data around. Only a fraction of this is web traffic.

The web uses **HTTP** or **HyperText Transfer Protocol** and applications (clients) called **web browsers** to access documents on **web servers**. Information from the remote computer is sent to your local computer using the HTTP protocol, usually over port 80. Your web browser interprets that information and renders it on your local computer.

All browsers are not created equal. Each offers different tools and renders HTML content in slightly (or very) different ways. Security and privacy issues may be handled with various degrees of success. This means that you should know what your browser can and can't do, and what settings and plugins give you that perfect balance of security and privacy (unless you like malware, advertisements, spam and your neighborhood knowing that you like to watch kittens play in green jello).

> The **hypertext** part of the HTTP protocol refers to the non-linear way you read it. You normally read in a linear fashion: page 1 then page 2; chapter 1 then chapter 2; lesson 1 then lesson 2, and so on. Hypertext lets you look at information in a non-linear way. You can jump around from topic to topic, learning as you go, then retrace your steps and maybe go see other information before you finish the parent article. That's the difference between hypertext and plain text.

In hypertext, words and ideas connect not only with the words that directly surround them, but also with other words, images, videos, and music. Hypertext isn't restricted to the Web. Most full-featured word processors let you create locally stored pages in web, or HTML, format. You read these pages in your web browser and they act like any other web page, only they are stored on your local computer, not a remote computer.

It's easy to create your own web page. The easiest way to do this is to use one of the common word processors like OpenOffice/LibreOffice Writer, Microsoft Word, or WordPerfect. These programs will allow you to produce simple web pages, combining text, hypertext and images. Plenty of people have made functional web pages using these (or even simple text editors like vi, found on most Unix platforms). Other text editors include Microsoft Notepad, Notepad++, SciTe, emacs and so on.

But these pages aren't flashy. Flashy means **CSS** and **scripts** and animations. You can spend lots of money on fancy web page design applications. These apps allow you to create interesting effects on your web page, but they're more complex to use. Still, they usually make the overall job easier. The lower-cost alternative is to get one of the text editors tailored for working with HTML and scripting languages, learn the syntax of HTML and scripting and code your own web pages from scratch.

Once you have the pages designed, you'll need a computer to put them onto, if you want other people to see them. **Internet Service Providers (ISPs)** provide **web hosting** on their web servers.

You can run a web server from your own home, using your own computer, but there are a whole handful of issues. Information stored on a web server is only available when that server's powered up, operating properly and has an open connection. So if you want to run a web server from your bedroom, you have to leave your computer on all the time; you have to make sure that the web server program is operating properly all the time (this includes troubleshooting hardware problems, controlling viruses, worms and other attacks, and dealing with the inevitable bugs and flaws within the program itself); and you have to keep a connection to the Internet open, which must be stable and fast. ISPs charge extra for a fast upload connection and a fixed IP address, which is why most people pay someone else to do all this.

A web hosting company stores your web pages on their computer. It's nice to let their servers be attacked, instead of yours. A good web hosting company will have multiple, redundant servers and a regular backup policy, so that your site doesn't disappear just because of hardware problems; a support staff keeps servers running despite attacks and program bugs; and a number of open connections to the Internet give some guarantee against outages. So all your have to do is design your web page, upload it to the hosting company's server, turn off your computer and go to sleep. Your web page will be available to the entire world, as long as you keep paying the bill.

It's also possible to find organizations that offer free web hosting. Some of these organizations are funded by paid advertising, which means that anyone who wants to view your web page will first have to view someone else's advertisement. But they won't have to buy anything, and you won't have to pay anything.

Exercises

4.1 A web page is just text that tells the browser where images, videos, and other things should be. You can see what it looks like by viewing the Page Source. Go to your favorite browser, point it at ISECOM.ORG and load the page. Now view the source. You'll see some tags with the word "meta" in them. For example, the first is meta-charset="utf-8". What does that mean? What is the point of it?

4.2 Find 3 more meta tags and explain what the point of them is. You may need to search around the web to figure it out so think carefully about what key words you will use in your search to make sure you get the right answers.

4.3 Save the ISECOM.ORG Page Source to your computer. Drag it to the browser. What changed? Why do you think it changed?

4.4 Open the ISECOM.ORG Page Source in a text editor and you'll see it's just words and numbers. Whatever you change or type in there will affect how the page looks when you save it and drag it back into the web browser. Delete things and you'll see it deleted. Change words and they will appear as you typed them. Now strip the page of anything else and add your name so it shows up larger and bolder than the other words. Try it. Save it. And drag it to the web browser and see if you were successful. No? Then keep trying!

*See the **Feed Your Head: Playing With HTTP** at the end of this lesson for an opportunity to dig deeper.*

Email – SMTP, POP and IMAP

The second most visible aspect of the Internet is probably email. On your computer you use an email client, which connects to a mail server. When you set up your email account, you get a unique name in the form of **user@domain** and you have to create a password.

There are two parts to email: **SMTP (Simple Mail Transfer Protocol)**, which *sends* mail, and the mail server, either **POP (Post Office Protocol)** or **IMAP (Internet Message Access Protocol)**, which *retrieves* your email.

The SMTP protocol (we'll remind you again) is used to *send* email. SMTP defines the **fields** in an email message, including the FROM, TO, SUBJECT, CC and BODY fields. Plain old SMTP does not require a password and sends everything in clear text; everybody gets to read your mail. This may not have been bad when the protocol was designed and the Internet was a small world inhabited by like-minded people. But it left a loophole that allowed any user to send **spam** and do other nasty things like **email spoofing**, which basically means lying about (spoofing) the sending address. Almost all contemporary mail servers use Secure SMTP, which means you must prove your identity before you can send an email.

In later lessons we'll show you how spoofing works and how to look for it in email headers. This bit of knowledge can turn you from the sheep to the wolf awfully quickly.

POP3 (Post Office Protocol version 3) is a "store and dump" protocol. The email server receives your email and stores it for you, until you connect and download (dump) your email. Then your outgoing mail is sent using SMTP. This is a good way to approach email if you have a dial-up connection, since it takes less time to send and receive email, and you can read your email offline.

IMAP, on the other hand, by default stores your mail on the server. Many corporate email solutions use a form of IMAP, depending on their software vendors. In IMAP, you can create folders in your mailbox and move messages between these folders. When you connect to the IMAP server, your mailboxes and the server synchronize the folders, contents, incoming email and deleted email. This has quite an advantage: you can get to all your mail from any computer or device you use: laptop, kiosk, phone or tablet. Plus you can download and store email in personal data files on your own computer.

However, there are also two drawbacks: first, obviously, is that you need to exchange more information, so you need a faster connection and more time. The second is that space is limited. Your mail server will assign you a mailbox size that you can't exceed. If you run out of space, you won't be able to receive messages unless you delete emails (or pay for more space). Ultimately this means that corporate IMAP email requires data management. You have to move mail to local storage and clean out your sent mail, spam, and trash on a regular basis to conserve space. Mail with attachments will destroy you. In this age of free Internet email accounts with huge free data storage, all this upkeep may seem stupid. Until you get sued. Or someone compromises the mail server and gets ALL your email.

Both POP and IMAP servers require a password to access your account. But both protocols send *everything* in the clear, including passwords, so anyone can potentially read them. You have to use some form of encryption to mask the login process (like SSL) and the contents of the mail. That's why many email clients have a *Use SSL* check box.

When you click the Send button in your email client, two things happen: first your client forces you to log in to the SMTP server (even though you've already logged in to the POP server, dang it!), and then it sends your outgoing mail (via the SMTP protocol).

This got annoying by the mid-1990s, when servers began to use a protocol called **POP-before-SMTP**: you first send the POP server your user name and password, then your incoming mail downloads, then the SMTP server checks you against the POP server ("Is this guy okay?" "Yeah, I authenticated him.") and sends your messages. It's a nice time-saver.

One important item to remember is, despite being password protected, email is not a way to send secure information. Most POP clients and servers require that your password be communicated – unencrypted – to your mail server. This doesn't mean than anyone who receives an email from you also receives your password; but it does mean that someone with the right knowledge and tools can sniff out your password – as well as the contents of your emails. (For ideas on making your email more secure, see **Lesson 9: Email Security**.)

Exercises

4.5 Send yourself an email from your main account, to your main account. Send the same email to your same main account from another account, for instance a free online account (come on, we know you have them). How long do the two messages take to arrive? If there is a difference, why?

4.6 Look at one of the billions of spam emails that clog your inbox. Can you determine who actually sent you a particular email? Is there any kind of hidden information in emails, for instance? If there is, how can a clever hacker see it?

4.7 Can you delay the sending of an email until a certain time or day (which can really screw up Deniability)? Can you think of cool way to use email sending delays to mess with your friends?

IRC

IRC (Internet Relay Chat) is a great place to see the unregulated nature of the Internet at its best. Or worst. On IRC, anyone with anything to say gets a chance to say it. IRC is also known as **Usenet** or **news groups**. Each news group has its own name, sub-name, sub-sub-name and so forth.

You may be familiar with chat rooms. IRC is just like a chat room, only there are no rules beyond basic **netiquette**, and quite often there are no chaperones. You may find exactly what you are looking for on an IRC channel, or you may find something you never knew existed.

> All the rules that you've heard about chat rooms are applicable to IRC channels. Don't tell anyone your real name. Don't give out your phone number, your address, or your bank account numbers. But have fun! If you are roaming around, be careful about the content available. Not everything on the Internet is malware-free, and not everyone on the Internet is nice.

IRC is not secure and everything you type is passed in clear text from IRC server to IRC server. You can set up private conversations between yourself and another IRC member but those are transmitted in the clear as well. Using a nickname will only get you a little privacy. If you're planning on conducting any malicious or unsavory actions, don't use the same nickname for every account. Using the same nickname is an excellent way of getting tracked down by the police. Or much less savory people.

Topics are called "channels." Since there are thousands of channels, we'll give you a URL that lists many of them for you to pursue until you lose your mind:

```
http://www.nic.funet.fi/~irc/channels.html
```

If you are having problems with the comments made by another member, you can either report them to the moderator (if there is one), or have that person **bumped** from that channel. If you don't want to hear what someone has to say, you can always block them or ignore their messages. Maybe that thread's not for you anyway.

Exercises

4.8 Find three IRC channels that focus on security topics. How do you join in the public conversation? What do you have to do to have a private conversation with a person?

4.9 What port does IRC use?

4.10 It is possible to exchange files through IRC. How do you do this? Would you *want* to exchange files through IRC?

4.11 What's the major difference between MIME and SMIME? When you see an "S" in an acronym, does that mean anything special to you as a *Secure* (hint, hint) minded person?

FTP

Old-school **File Transfer Protocol (FTP)** typically runs over ports 20 and 21. Guess what: it lets you transfer files between two computers. While it can be used for private file transfers, because it doesn't use encryption it's more commonly used for free, anonymous FTP servers that offer public access to collections of files, like the ISO for that cool new Linux distro.

Anonymous FTP was once the main way for computer users to exchange files over the Internet. While there are plenty of anonymous FTP servers used to distribute files illegally (which is a nice way to spread binary disease), more are legally used to distribute programs and files. You can find servers that offer anonymous FTP services through the usual methods, for instance search engines. But remember: FTP logins are sent in clear-text. Yes, even though we're talking about a user name and password. (Is that lame or what?) There is secure FTP (SFTP) but it's not universally used.

Most anonymous FTP servers allow you to access their files using the FTP protocol through a web browser. There are also some really great FTP clients that work like a file management program. Once you're logged into the FTP server, you can move files onto your computer in much the same way you move files on your own computer. FTP just takes a bit longer to download each file over your computer, mainly because the FTP server may be located on the other side of the planet.

Exercises

4.12 Windows, OSX and Linux come with a basic command line FTP client; to access it, open a command prompt or terminal window and type:

```
ftp
```

At the `ftp>` prompt, you can type `help`, to get a list of available commands.

```
ftp> help
Commands may be abbreviated. Commands are:
!               delete          literal         prompt          send
?               debug           ls              put             status
append          dir             mdelete         pwd             trace
ascii           disconnect      mdir            quit            type
bell            get             mget            quote           user
binary          glob            mkdir           recv            verbose
bye             hash            mls             remotehelp
```

```
cd              help            mput            rename
close           lcd             open            rmdir
```

The basic commands are:

Connect to the FTP server named *ftp.domain.name*:

```
ftp> open ftp.domain.name
```

List the contents of the remote working directory:

```
ftp> ls
```

or

```
ftp> dir
```

Change the remote working directory to a directory named *newdir*:

```
ftp> cd newdir
```

Download a file named *filename* from the remote computer to the local computer:

```
ftp> get filename
```

Download multiple files named *file1*, *file2*, and *file3* from the remote computer to the local computer (you can also use wildcards to download many files with the same suffix, or all the files in a directory):

```
ftp> mget file1 file2 file3
```

Uploads a file named *filename* from the local computer to the remote computer:

```
ftp> put filename
```

Disconnect from the remote FTP server:

```
ftp> close
```

Shut down your local FTP client:

```
ftp> quit
```

An FTP session, step by step

To connect to an anonymous ftp service, first open your local FTP client:

```
ftp
```

Use the open command to connect to the server. The command

```
ftp> open anon.server
```

connects your FTP client with the anonymous FTP server named *anon.server*. Substitute the name of a real server, of course.

When the remote FTP server accepts your connection, it will identify itself to your local client, then ask for a user name.

```
Connected to anon.server.
220 ProFTPD Server (Welcome . . . )
User (anon.server:(none)):
```

For most anonymous FTP servers, you should enter in the word *anonymous* (or *ftp*) as the user name. The remote FTP server will acknowledge that you are connecting as an anonymous user, and will give you instructions on what to use as a password.

```
331 Anonymous login ok, send your complete email address as your
password.
Password:
```

In most cases, the remote server does not check the validity of the email address entered as a password, so it will not stop you from accessing the service if you enter an invalid address. This is considered to be a breach of netiquette, but it's actually necessary: do not give away your real email address! After you have entered a password, the remote server will send a welcome message to your local computer.

```
230-
   Welcome to ftp.anon.server, the public ftp server of anon.server. We
   hope you find what you're looking for.
   If you have any problems or questions, please send email to
   ftpadmin@anon.server
   Thanks!
230 Anonymous access granted, restrictions apply.
```

From here, you can use the ls, dir, cd and get commands to download files from the remote server to your local computer.

Exercises

4.13 Using these examples, find and download a file from an anonymous FTP server.

4.14 Use your web browser and a search engine to find an anonymous FTP server that has a copy of *Alice in Wonderland*, then, using the command line FTP client – not your web browser – download the file.

4.15 What are the better FTP clients out there? Can they automate all the command-line stuff and provide you a nice graphical interface? Do you lose any functionality you have at the command line?

4.16 Could your computer become an FTP server?

Telnet and SSH

Telnet allows a local user to send a wide variety of commands to a remote computer. This allows the local user to instruct the remote computer to perform functions and return data to the local computer, almost as if you were sitting at a keyboard in front of the remote computer. **Secure Shell (SSH)** is intended as a secure, encrypted replacement for clear-text telnet.

Again, most Windows, OSX and Linux versions come with a basic, command line telnet client; to access it, open a command prompt or terminal window and type:

```
telnet
```

To access a telnet server, you will need to have an account and password set up for you by the administrator of the server, because the telnet program lets you do lot of things, some of which could severely compromise the remote computer.

Telnet was used back in the day to allow computer administrators to remotely control servers and to provide user support from a distance. This service is part of the old Internet and isn't used much anymore.

Telnet can also be used for a number of other tasks, such as sending and receiving email and viewing the source code for web pages (although telnet is probably the most difficult way to do these things). Many of these things are legitimate, but they can be abused for illegal or immoral reasons. You can use telnet to check your email, and view, not just the subject line, but the first few lines of an email, which will allow you to decide whether or not to delete the email without downloading the entire message.

If you are going to use SSH, be sure you're using a current version, because older versions had various vulnerabilities, and many automatic vulnerability scanners constantly search for these on the Internet.

Game On: Command Me

The dark screen twinkled across the front of Grandpa's thick glasses as the cursor blinked impatiently, expecting a command. With his gray thin hair lapped lazily over his wrinkled head, Grandpa tapped the keyboard. Jace watched the silent piano player play the keys of his computer, tap, tap, tap, tap. He smiled at Jace with his head turned to look into her young eyes. "Jace, I'm going to show you a new world out there. Buckle your seatbelt," he winked at the eight year old.

Jace's feet barely reached the floor in the computer chair with her grandfather behind the computer screen. She heard the telephone dial tone coming from a small box close by. The white box lit up with green and red lights as the tone changed to a sound like a duck being swallowed by a garbage disposal. Grandpa raised his excited eyebrows and stared with all of his might at the black screen in front of him. The duck stopped wailing and all of the lights turned green on the telephone box.

Grandpa said, "Watch this."

Usually when Grandpa said, "watch this" something would explode or black smoke would come from something. Either way, "watch this" meant Grandma was going to mad about something he did wrong. Jace loved to hear those words though, because it was exciting anticipation for some wonderful event.

The computer screen woke up from its dark slumber with a banner of ASCII text surrounding the words "Welcome to Cline's Bulletin Board System (BBS)." "We're in," Grandpa clapped and attempted to high-five eight year old Jace. His hands missed her hand by several inches and he almost smacked the little girl in her face. She laughed, and Grandpa did too.

They both looked back and forth at the keyboard and the computer screen. Grandpa rubbed his fingers together while Jace rubbed her mind, trying to figure out what was going on. Grandpa began typing commands on the silent piano, head bowed down over the keys as a vulture might eat roadkill. Head up, head down, head up, head. Oops. He sat back. Grandpa had forgotten something very important.

He paused and spoke like a teacher. "Jace, I'm sorry but I forgot to tell you what's going on here. Right now, I am connected to another computer over our telephone line. That noisy thing over there is called a "Modem" and its job is to convert digital signals to analog and vice versa." Jace already knew way too much about telephone systems due to Grandpa's desire to play around with it every chance he could. 48 volts during normal use and 90 volts during a telephone ring notification, she knew more than any telephone technician needed to know. Plain old telephone system or POTS was a joke between Grandpa and herself. Grandma didn't seem to

get the POTS joke which made it that much funnier.

Telephone lines can be tapped by an interested party, but that could be detected by using a voltage regulator. The telephone line voltage would spike momentarily and keep slightly elevated if someone tried to tap the line. Jace thought that Grandpa loved his voltmeter more than he loved Grandma; he never left home without it. Grandpa even went so far as to name his device "Valerie." Valerie the voltmeter. That was his best friend, besides Jace.

Jace rolled her smarty-pants eyes at the Grandpa, more so at the lecture on analog modulation to digital modulation, converting sound to a digital signal. That's pretty much what a modem does. Grandpa continued his lecture to the reluctant student, "The computer I'm connected to allows me to connect to other computers and play around with whatever services they provide." Her ears picked up a word that she hadn't heard before, "services."

"Grandpa what do you mean 'services?'" the curious girl asked expecting an answer involving fast food. "Excellent question, my dear," Grandpa was expecting Jace to ask a question like that. "My computer is connected to a network of computers or I have the ability to connect to other computers across the world," he gladly replied. "This modem lets me talk to these other computers that offer access to files, information, people to talk with, and wonderful stuff like that. These computers offer services like File Transport Protocol, Usenet, IRC, Telnet, and Email."

Jace wasn't satisfied with the answer she was given and this usually led to many more questions fired at Grandpa in rapid succession. She loaded up her question ammo belt and began the carnage: "What is File Tranfer proto thing? What is MIC? Where is Telknit? Does Imail need special stamps? What color is it in the digital world? Who invented Usednet? Why do they call it Imail? Does Grandma know about your services? Why do they call it services? Where do babies come from? Where does Jello come from?"

Grandpa had to cover his ears to shelter his brain from the onslaught of questions. "Wait, wait, wait, slow down."

Game Over

DNS

When you want to call a friend on the phone, you need to know the correct phone number; when you want to connect to a remote computer, you also need to know its number. You may remember from previous lessons that, for computers on the Internet, this number is the IP address.

IP addresses are very easily managed by computers, but we humans prefer to use names, in this case **domain names**. For example, to connect to the Hacker Highschool website, type www.hackerhighschool.org into the address bar of a web browser. However, the web browser can't use this name to connect to the server that hosts the Hacker Highschool website – it needs the IP address. This means that your local computer must have some means of translating domain names into IP addresses. If there were only hundreds, or even thousands of computers on the Internet, then it might be possible for you to have a simple table (a **hosts file**) stored on your computer to use to look up these addresses. However, for better or worse, not only are there millions of computers on the Internet, but the correlations between domain names and IP addresses change constantly.

Domain Name Service (DNS) is used to dynamically translate domain names into IP addresses (and vice-versa). When you type the domain name *www.domainname.com* into your web browser, your web browser contacts the DNS server chosen by your ISP. If that DNS server has *www.domainname.com* in its database, then it returns the IP address to your computer, allowing you to connect.

If your DNS server doesn't have *www.domainname.com* in its database, then it sends a request to another DNS server, and they will keep sending requests to other DNS servers until one finds the correct IP address, or establishes that the domain name is invalid.

Exercises

4.17 Open a command line window and identify the IP address of your computer. What command have you used? What IP address do you have?

4.18 Identify the IP address of your DNS server. What command have you used? What is the IP address of the DNS server?

4.19 Ping *www.isecom.org*. Do you receive an answer? What IP address answers the ping?

4.20 Can you direct your computer to use a different DNS server? If so, change the configuration of your computer so that it uses a different DNS server. Ping *www.isecom.org* again. Do you receive the same response? Why?

DHCP

DHCP or **Dynamic Host Configuration Protocol** allows a local network server to hand out IP addresses within the network. The server is given a block of IP addresses to use. When a computer joins the network, it gets an IP address. When a computer leaves, its IP address becomes available for use by another computer.

This is useful for large networks of computers, since it's not necessary for each computer to have an individually assigned, static IP address. Instead, you use a DHCP server. When a new computer connects to the network, the first thing that it does is request an IP address from the DHCP server. Once it has been assigned an IP address, the computer then has access to all the services of the network.

Now think about that. Most wifi networks offer DHCP, meaning that *anyone* can get an IP address on that subnet. If you're running a coffee shop that's exactly what you want, but if you're running a secure office, you might consider using fixed IP addresses instead. It depends....

Connections

In the bad old days, computers connected to the Internet through a modem. Modems translate bits into sounds and back, **mod**ulating and **dem**odulating, thus the name. Modem speeds are measured in **baud** (a rating number) and **bps**, or bits per second. Higher baud rates usually mean higher bps, but you must also consider what you are planning to do. There are certain applications – such as telnetting into **Multi-User Dungeons (MUDs)** – for which a twenty year old 300 baud modem would still be acceptable (provided your typing speed wasn't so good), while high bandwidth applications such as streaming video can often strain even the most powerful cable or DSL connections.

ISPs

You don't just call up the Internet. You need to access a server that will connect your computer to the Internet. The server does all the heavy work, and is on all the time. The server is run by an **Internet Service Provider (ISP)**.

An ISP has a point-of-presence on the Internet that is constant, and it has servers that run services you can use. But you can run these services on your own, too. For example, you can run a mail server on your local computer, but it will require you to have your computer powered up and connected to a network all the time, just waiting for those brief moments when information has to be exchanged. An ISP, however, consolidates the efforts of a large number of users, so the mail server is working all the time, instead of sitting around, doing nothing. The ISP's computers use a high speed connection to connect to a **Network Access Point (NAP)**. These NAPs then interconnect with each other through ultra-high speed connections called **backbones**. All these things together are the Internet.

Plain Old Telephone Service

Plain old telephone service (POTS) was once the most widely used method of accessing the Internet. Its primary disadvantage is its low speed, but in many cases this is made up for by its wide availability. Most national Internet service providers have a large number of local access numbers, and almost everyone still has a phone with a land line. In theory, if you had an acoustic modem and a pocket full of change, you could connect from almost any public pay phone (if you can find one). Not that you would really want to do that.

POTS is slow. The fastest telephone modems are rated at a speed of 56,600 bits per second (bps). That, however, as they explain in the small print, is a lie. Power constraints limit the actual download speed to about 53,000 bps and the effective rate is usually much lower. This doesn't compare very well with DSL or cable modems.

That said, telephone service is widely available, and POTS based ISPs are relatively cheap (and sometimes free). You wouldn't want to trade pirated movies over POTS, because it's immoral, illegal and ties up your phone line all night and maybe all weekend, but you could certainly send friendly, text based emails to Granny. And if you used telnet, you could even do it with a dusty DOS based machine that you pulled out of the basement.

DSL

A **Digital Subscriber Line (DSL)** is a method of sending large amounts of information over the wires that already exist for POTS. Its main advantage over POTS is that it is much faster than analog modems, and it provides a permanent connection. In addition, it allows you to make and receive regular telephone calls while you are connected to the Internet. Its main disadvantage is that its availability is limited by how close you are to the telephone company's switching equipment – if you live too far down the line, you're out of luck.

Cable Modems

Cable gateways don't use traditional telephone lines to connect to the Internet. Instead they use coaxial cable (or fiber-optic lines, if you're really lucky) provided by cable companies. Like DSL, cable gateways can allow you to make and receive regular telephone calls while you are connected to the Internet, and they provide a permanent connection, but cable gateways are generally faster than DSL.

Cable gateways have some basic flaws. The first is that cable gateway access is a shared resource, so your connection speed will be decreased when there are other users on the same cable with you. The second is that cable access is only available in areas where

cable companies have installed the necessary wiring. And the most serious one is that any traffic you put on the cable can be viewed by any other user on that cable! This means that if you connect your computer to the cable gateway and do not use a firewall, everyone else in the neighborhood can see your computer and all its files. Do you really want to share your bank account information like that?

Wimax

Wimax is a wireless connection method that generally competes with DSL. It is used in places where a wired infrastructure would be too expensive or difficult to setup. Signal strength can be affected by buildings, trees or other large objects. Some versions use a fixed access point, but others give you mobile access over truly large areas.

Wifi

Wifi is not a method to connect to your ISP but it is a common networking method for connecting to the Internet at home or at commercial establishments like malls or coffee shops. Most smartphones and all laptops now use Wifi so it's a favorite target for attackers. Consider yourself naked in a crowded room when you use public Wifi: cover yourself, make sure nobody's looking at you but assume everybody wants to. Of course you will read the Wireless Security lesson too, right?

Exercises

4.21 What kind of Internet connection do you have at home, if you have one? How can you tell? And most important:

4.22 Who can you see on that network? (How can you find out?)

4.23 How fast is your connection? Can you improve your speed without calling up your ISP?

4.24 What additional services does your ISP provide? We talked about services already; your ISP may support several.

4.25 What services can you provide from your own computer?

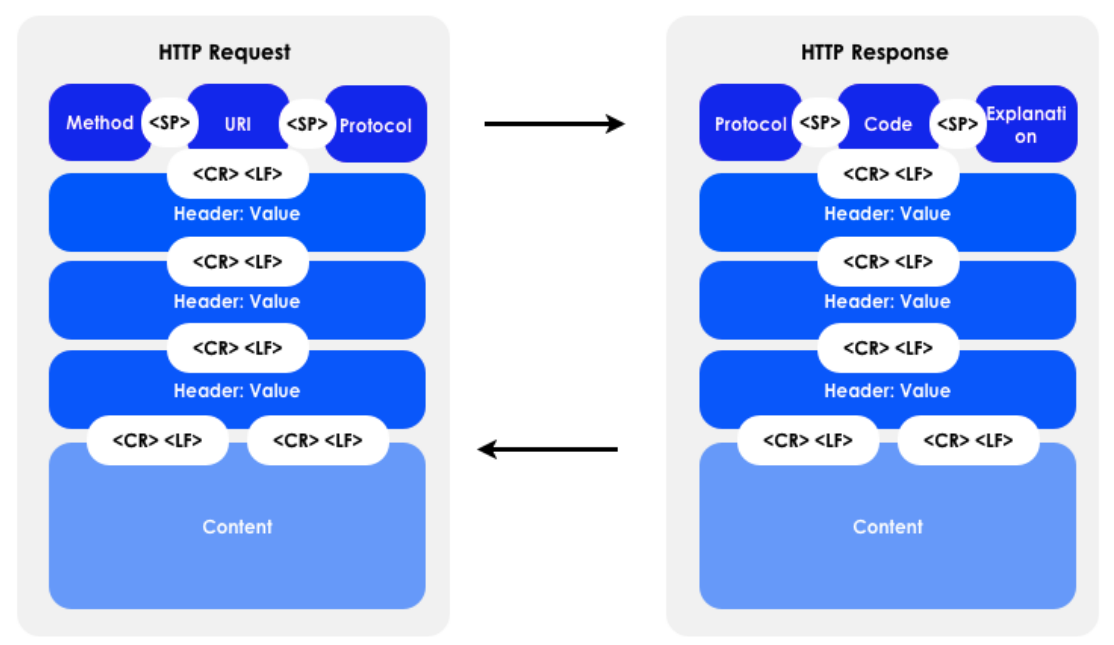
Feed Your Head: Playing With HTTP

HTTP, the acronym for Hypertext Transfer Protocol, is located on the top of TCP/IP stack as is defined in two main RFC:

- 1945 for 1.0 (based from 0.9).
- 2616 for 1.1.

There are some substantial upgrades and differences from 1.0 to 1.1 for Extensibility, Caching, Bandwidth optimization, Network connection management, Message transmission, Internet address conservation, Error notification, Security, integrity, and authentication, Content negotiation [3]. Differences between 1.0 and 1.1 are useful to obtain information about a web server.

Basically HTTP is a stateless protocol in which Client sends an HTTP Request to Server, which sends an HTTP Response: the Request/Response paradigm.



As you may know we can obtain a lot of information sending commands to an HTTP server. We will use some basic network tools:

- netcat: the TCP/IP tool kit
- curl: the HTTP tool kit
- proxy: like OWASP ZAP or Burpsuite free

Sniffing the Connection Between You and the HHS HTTP Server

Use a proxy to connect your browser. Go to http://www.hackerhighschool.org and intercept your request:

```
GET / HTTP/1.1
```

```
Host: www.hackerhighschool.org
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.8; rv:11.0)
Gecko/20100101 Firefox/11.0
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-us,en;q=0.5
Accept-Encoding: gzip, deflate
Proxy-Connection: keep-alive
and response:
HTTP/1.1 200 OK
Content-Length: 10376
Date: Fri, 03 Feb 2013 09:11:17 GMT
Server: Apache/2.2.22
Last-Modified: Mon, 06 Feb 2013 09:31:18 GMT
ETag: "2f42-4b8485316c580"
Accept-Ranges: bytes
Identity: The Institute for Security and Open Methodologies, The
Institute for Security and Open Methodologies
P3P: Not supported at this time, Not supported at this time
Content-Type: text/html
Connection: keep-alive

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"[]><html
xmlns="http://www.w3.org/1999/xhtml" dir="ltr" lang="en-US"
xml:lang="en"><head><meta http-equiv="Content-Type"
content="text/html; charset=UTF-8" /><title>Hacker Highschool -
Security Awareness for Teens</title>
[...]
```

Exercises

4.26 Identify parts of requests from proxy using the diagrams.

4.27 Is there interesting information in the headers?

Your First Manual Connection

Netcat can be used to connect to a web server using host port settings.

Start by typing:

```
nc www.hackerhighschool.org 80
```

Then press Enter two times.

```
GET / HTTP/1.0
```

The server will reply:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```

```
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"[]>
<html   xmlns="http://www.w3.org/1999/xhtml"   dir="ltr"   lang="en-US"
xml:lang="en"><head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>ISECOM - Institute for Security and Open Methodologies</title>
<meta name="description" content="Description" />
```

As you can see the page appears to be from isecom.org and not from hackerhighschool.org. Why?

One hypothesis could be that the same host serves up both HHS the ISECOM site. Is this possible?

To find out, check the hackerhighschool.org IP address:

```
nslookup www.hackerhighschool.org
[...]
Non-authoritative answer:
www.hackerhighschool.org      canonical name = hackerhighschool.org.
Name: hackerhighschool.org
Address: 216.92.116.13
```

And now for www.isecom.org:

```
nslookup isecom.org
[...]

Non-authoritative answer:
Name: isecom.org
Address: 216.92.116.13
```

Same IP address! Using netcat it's possible to show the host by manually adding the Host header and using HTTP 1.1:

```
GET / HTTP/1.1
Host: www.hackerhighschool.org

HTTP/1.1 200 OK
Content-Length: 10376
Date: Fri, 03 Feb 2013 09:11:17 GMT
Server: Apache/2.2.22
Last-Modified: Mon, 06 Feb 2013 09:31:18 GMT
ETag: "2f42-4b8485316c580"
Accept-Ranges: bytes
Identity: The Institute for Security and Open Methodologies, The
```

```
Institute for Security and Open Methodologies
P3P: Not supported at this time, Not supported at this time
Content-Type: text/html
Connection: keep-alive
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"[]>
<html xmlns="http://www.w3.org/1999/xhtml" dir="ltr" lang="en-US"
xml:lang="en"><head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Hacker Highschool - Security Awareness for Teens</title>
```

The Request Method

Another part of an HTTP request that can be modified is the Request Method. Typically web applications use GET and POST requests, but other request protocols may be active on a web server or application server. Common methods are:

- **OPTIONS** - used to ask what request options are supported.
  If you're running a web server, be aware that giving this information away could be a problem.
- **GET** - used to retrieve information directly via the URL, for example:
  `http://www.usairnet.com/cgi-bin/launch/code.cgi?Submit=Go&sta=KSAF&state=NM`
  See all that stuff after the question mark? That's the request data. Passing data this way is risky, because it's in plain sight, and it's easy to tinker with.
- **HEAD** - used like GET but the server does not return an actual page.
  This can be used to identify Accesses, optimizing bandwidth consumption and – in some cases – bypassing access controls. In fact some ACL implementations check only GET requests. In this case you have found a Vulnerability.
- **POST** - used to send data to web applications – like GET – but data is transmitted in the Request Body, out of sight to at least a degree.
- **PUT** - used to allocate resources on a web server or to update it.
  In many contexts this method should have been disabled or protected by an Authentication Control. In other contexts this is a delightful find.
- **DELETE** - used to free resources on a web server.
  This method should be disabled or protected by an Authentication Control. See PUT above.
- **TRACE** - used as an application layer loopback that reflects messages.
  This debug method should be disabled, in particular on production environment because is a Confidentiality Concern and introduces a Vulnerability because it can be used for Cross Site Scripting exploits.
- **CONNECT** – to use the web server as a proxy.
  This should be disabled or protected by an Authentication Control because it permits others to connect to third party services using the proxy IP.

Also consider more protocols based on HTTP can adds more methods, as WebDAV. You can alter Request Method in order to look at server replies for interesting stuff, asking for known methods and also arbitrary words.

Requesting OPTIONS

You could start the netcat session as usual:

```
# nc www.hackerhighschool.org 80
```

But don't press Enter twice this time. Instead, type the next line:

```
OPTIONS / HTTP/1.1
```

and you'll get a response like:

```
Host: www.hackerhighschool.org
HTTP/1.0 200 OK
Date: Tue, 07 Feb 2013 08:43:38 GMT
Server: Apache/2.2.22
Allow: GET,HEAD,POST,OPTIONS
Identity: The Institute for Security and Open Methodologies, The Institute for Security and Open Methodologies
P3P: Not supported at this time, Not supported at this time
Content-Length: 0
Content-Type: text/html
```

Requesting HEAD

This time, after starting your session, enter the HEAD option.

```
# nc www.hackerhighschool.org 80
HEAD / HTTP/1.1
Host: www.hackerhighschool.org

HTTP/1.0 200 OK
Date: Tue, 07 Feb 2013 08:41:14 GMT
Server: Apache/2.2.22
Last-Modified: Fri, 13 Feb 2013 15:48:14 GMT
ETag: "3e3a-4bd916679ab80"
Accept-Ranges: bytes
Content-Length: 15930
Identity: The Institute for Security and Open Methodologies
P3P: Not supported at this time
Content-Type: text/html
Age: 45
Connection: close
```

Let Me Use You As A Proxy: the CONNECT Request

```
# nc www.hackerhighschool.org 80
```

```
CONNECT http://www.isecom.org/ HTTP/1.1
Host: www.hackerhighschool.org
```

Exercise

4.28 Use netcat (nc) to try all of the Request methods listed above on the HHS network servers or a server set up for the purpose. What kind of interesting stuff can you dig up?

Scripting HTTP requests with curl

Some Web Application Testing is based not only to Web Server response but on the (Web) Application Layer. Often you can find web application vulnerabilities by altering GET and POST parameters, altering cookies and tinkering with headers. A useful tool, for bash scripting is the **curl** command, a command-line tool for requesting a web page. But curl adds some logic over netcat.

Asking for:

```
# curl http://www.isecom.org
```

is not the same as

```
# nc www.isecom.org 80
GET / HTTP/1.1
```

To see this you can use the -v switch for verbose output:

```
# curl -v http://www.isecom.org/
* About to connect() to www.isecom.org port 80 (#0)
*   Trying 216.92.116.13...
* connected
* Connected to www.isecom.org (216.92.116.13) port 80 (#0)
> GET / HTTP/1.1
> User-Agent: curl/7.26.0
> Host: www.isecom.org
> Accept: */*
>
* HTTP 1.0, assume close after body
< HTTP/1.0 200 OK
< Date: Tue, 07 Feb 2013 09:29:23 GMT
< Server: Apache/2.2.22
< Last-Modified: Fri, 13 Feb 2013 15:48:14 GMT
< ETag: "3e3a-4bd916679ab80"
< Accept-Ranges: bytes
< Content-Length: 15930
```

```
< Identity: The Institute for Security and Open Methodologies
< P3P: Not supported at this time
< Content-Type: text/html
< Age: 247
< Connection: close
<
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"[]>
<html xmlns="http://www.w3.org/1999/xhtml" dir="ltr" lang="en-US"
xml:lang="en">
[...]
```

As you can see curl selects automatically the HTTP version 1.1, adds host header, user agent and accept. Which points to an important rule for hackers: know your tools.

Luckily curl is a nice tool can be highly customized using switches.

To see all of them use `curl –help`

Some switches for functions similar to the netcat example above are:

- **-H** to add a header line
- **-X** to select a request method (also known as Command)
- **-d** to add POST data
- **-i** to include protocol headers in the output
- **-s** to enable silent mode, useful for scripting

With curl and some bash scripting you can automate web application testing. Looking for interesting HTTP headers from a server can be automated simply with curl and grep:

```
# curl -siX HEAD http://www.isecom.org/ | grep "Server:"
Server: Apache/2.2.22
```

Exercise

4.29 Expand the script above to request more HTTP headers and potentially useful information.

References and Further Reading

http://www.ietf.org/rfc/rfc1945.txt

http://www.ietf.org/rfc/rfc2616.txt

http://www8.org/w8-papers/5c-protocols/key/key.html

http://netcat.sourceforge.net/

http://curl.haxx.se/

Conclusion

The World Wide Web is a whole lot more than the Internet: there are all kinds of services besides just HTTP. FTP, SSH, DNS, DHCP and a whole lot more all offer windows into other people's computers – and yours. Understanding how you connect to these services, whether "through the proper channels" or otherwise, is key to knowing how you or your computer can be attacked – or attack. Just remember the motto: Hack everything, but harm none.

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