EE487: Applications of Cyber Engineering Name:
SX-15: SSH and Man in the Middle

Discussion: We've now seen that symmetric encryption is easy to set up and use, but very vulnerable because of the need to pass a key to the receiver. Asymmetric encryption solves the latter problem, but it's quite difficult to set up properly in the first place and it is far more computationally complex. We'll talk about how to solve both problems today. Public-private key ciphers are so intensive that they really cannot be used in the same manner as symmetric keys (encrypting large blocks of data). It would simply make everything too slow. Instead, why not use both? Can you think of how? We call this "hybrid" encryption, and it's the real way that most encryption mechanisms work in practice.

Today we're going to explore how to perform authentication using "OpenSSH" (and also how NOT to perform authentication). SSH stands for "Secure Shell." It is going to look very familiar to the shell that you have been using already to connect to your own machine. Now, however, you're going to connect to another machine, but using the network and "secure shell" connection that is encrypted. In fact, it's going to be hard even to realize that you're on someone else's computer and not your own.

If you've watched older movies with "hackers," you often see them using a text-based CLI on a target machine. They were probably using "telnet" or "remote shell" which have long been known to be insecure and are now never used by reputable IT professionals. SSH is the secure replacement for telnet and remote shell, but even then SSH is on version 2 because of flaws found in version 1. SSH is so commonly used across the Internet for remote administration of computers that it is a major vector for attack. Often SSH is automated such that computers connect automatically to perform routine scripted commands, so it's perfect for a hacker. It's typical for an SSH server to get probed or attacked several times every day-usually looking for servers still running SSH version 1 so that it can exploit its flaws.

Remember when you ran the buffer overflow exploit from bash on your own computer? Guess what – with ssh you can do that on someone else's computer anywhere in the world (assuming you have a network connection to it like the Internet). Or, let's say you merely wish to covertly intercept the communications between two parties, sort of like wire-tapping a phone line. That's where a man-in-the-middle (MITM) attack is useful.

Setup: Open your personal Ubuntu Virtual Machine (VM) using the VSphere Client and be sure you are at a CLI with the "midshipman" user prompt. You will have sudo NOPASSWD privileges on your machine, so be careful!

Reading: Erickson pages 406 to page 410 (sections 0x750 and 0x751).

EE487: Applications of Cyber Engineering Name:
SX-15: SSH and Man in the Middle

1. **OpenSSH** has two components: client (`ssh`) and server (`sshd`). The client is what you use to connect to another machine. The server is what runs and waits for connections from a client. Does this sound familiar? It should. From the network perspective, the only difference is that these connections happen on TCP port 22.

    a. What port did your web server use?

    b. What does it mean to have a port that is "open" or "listening?" Does that make a computer a client or server?

    c. The ssh server is known as a **daemon**. Look up the definition of daemon and give a brief synopsis. You will find that folks pronounce it two ways: "day-mon" or "dee-mon."

    d. Do you think that hackers like daemons? Is it easy to know if one is running?

    e. Use "`ss`" to determine if you already have an ssh server running (listening). Do you? What port would be open?

2. We're going to use the ssh client program already installed on your VM. However, we're going to use a hacker version of the sshd (server) program called **mitm-ssh**. The real ssh server (non-hacker) is at IP 192.168.1.134.

    a. First, let's log into the friendly ssh server (password: GoNavy)
    ```
    ssh RemoteMidn@192.168.1.134
    ```
    You should get a warning prompt. Go ahead and say "yes" to continue connecting. What did that cause your ssh client to do?

b. Once connected, you will have a shell prompt on the server. Can you verify what machine you are logged into? Use the "**`ip`**" command appropriately to verify the IP address. Is it your IP address or the server?

c. Now log out with the command "**`exit`**" and then log in again. Did you get the warning again? Why or why not?

Be sure to log out again with "**`exit`**" when done.

d. Use "**`cat`**" to look at the file **`~/.ssh/known_hosts`** on your local VM. What do you see there?

e. Your instructor will now follow the author's example on pages 407 and 408 to get the mitm-ssh server up and running on his virtual machine. When he is done, attempt to connect to the ssh server again. What happened that was different? Should you heed this warning?

f. Ignore the warning (**`ssh -o UserKnownHostsFile=/dev/null -o StrictHostKeyChecking=no RemoteMidn@192.168.1.134`**).
Again use the "**`ip`**" command. What machine are you connected to?

g. The problem is that you are actually connected through TWO ssh tunnels, one going to the hacker computer and the other from the hacker computer to the actual server. If you watch the screen, you will see that everything you do on the server is actually being watched and recorded by your instructor. Would you have been able to know this on your own?

h. How did you instructor get your computer to connect to his IP address instead of the actual server? **Hint:** think ARP.

i. What if your instructor managed to copy the keys from the ssh server. Would you have received that warning about changed keys? Would a smart hacker do this?

j. Is it important, therefore, to keep the keys on the server very protected?

k. SSH is often used by system administrators to perform automated daily computer backups of remote servers over the Internet. How useful is that to a hacker or perhaps an enemy nation-state?

3. Let's finish by generating keys for ourselves. Type the command '**`ssh-keygen`**' and follow the prompts (hit 'Enter' to use default values).

   a. Now look in your **`~/.ssh`** directory and write what you see. Do those files make sense? You should be able to figure out what each is. Write them here and what they are for (The ssh man page can help).

   b. Now do "**`ls -lha ~/.ssh`**" and look at the file permission (POSIX permissions). What do you notice? Does it make sense? If you change those permissions, ssh actually won't even run for security reasons.

   c. Use "**`cat`**" to look at the files. You could simply copy and paste the public key to give to others. Can you look at the private key? Whoa! Is that a problem? (**Hint:** take a close look at the top line.)

   d. It's encrypted, so that means we can just let anyone have it, right?

4. When you last logged into the SSH server, you used a username and password to authenticate yourself. That's not really the best way to operate. In practice, good SSH servers disallow password logins and instead use either keys or a network login service such as Kerberos.

    a. Instead, could the server use your key that we just generated to authenticate you? Explain how that might work.

    b. Let's transfer our public key to the server. Use this command and note the results:
    `ssh-copy-id –i ~/.ssh/id_rsa.pub RemoteMidn@192.168.1.134`

    c. Now log in to the SSH server again. What's different?

    d. Navigate to the **`~/.ssh/authorized_keys`** file and use "**`cat`**" to look at it. What is the purpose of this file? **Hint:** Consult the **`ssh`** man page.

5. I promised we'd get back to hybrid ciphers. When SSH sets up a session, it doesn't just authenticate the user. The server actually encrypts a randomly generated symmetric session key using the public key of the user and sends it back to the user. The user then uses that symmetric key to initiate a symmetrically encrypted "tunnel" between the two parties. All subsequent transmissions are now encrypted with the symmetric method.

    a. Why is the asymmetric method used only to send a symmetric key (what problems that we've talked about does that solve)?

    b. How does the server know that the person trying to connect is the person that it sent the symmetric key to?

EE487: Applications of Cyber Engineering Name:
SX-15: SSH and Man in the Middle

Conclusion and Results:

Your typed lab report will consist of two paragraphs, in the first paragraph:
• Briefly describe what you did in the lab in your own words.
• Discuss something new that you learned.

In the second paragraph, answer the questions:
• How could an adversary use this knowledge or these tools for malicious purposes?
• How could you use your new understanding to protect your systems and personnel from attack?

Staple the completed report to the back of your original lab and turn it in to your instructor at the beginning of the next class.