Homework 6
Introduction to Information Security (266-642)
Due Date: May 8, 2003 (Thursday)

Note: You can talk to your classmates, instructor, and TA about the problems. However, unless stated otherwise, problems should be written up individually. University of Wisconsin rules for academic misconduct apply.
In the homework, "the Stallings book" refers to [Sta03] and "the Handbook" refers to [MOV97] (I have linked the Handbook to the class homepage. You can download it for free.) Unless otherwise stated each part of a question has equal weight.

Question 1 (SSL [30 points]):
Part A [10 points]: Problem 17.1 from the Stallings book.
Part B [20 points]: Problem 17.2 from the Stallings book.

Question 2 (Malicious code [30 points]): Contact Mihai Christodorescu at `mihai@cs.wisc.edu` for parts A and B. Drew Bernat at `bernat@cs.wisc.edu` should be contacted about part C.
Part A [10 points] How do the propagation methods for the Morris worm and for the Slammer worm differ?
Part B [10 points]: Look at the development of viruses and other malicious codes, and how it correlates to the general history of computing. What do you think are the factors that helped/supported the rapid evolution of malicious code?
Part C [10 points]: List three methods to defeat a stack-based buffer overflow attack.

Question 3 (Network Attacks [40 points]):
Hello wily hacker! This problem guides you through the steps an attacker may use when launching an attack across a network. Your end goal: run a specific program with root privileges on a machine located somewhere in the CS building. Please direct questions about this problem to Jonathon Giffin at `giffin@cs.wisc`.

We have scripted out the series of steps necessary to reach the goal. The experiments are designed to be run from the tux lab. You may run the attack from the nova lab if you wish, however, you will be responsible for acquiring the network tools listed below. We are only providing linux versions. You cannot run the attack from outside the CS labs as the target machine is unreachable when outside the CS network.

Your target: **vulnerability.cs.wisc.edu** running a default installation of a recent Linux distribution. When running the experiments, **target only this machine**. The CSL knows this machine will be targeted and expects attacks against it. **You will be held accountable for attacks against any other machines in the CS network or outside Internet**.

You will use the following programs in the experiments:

ping, telnet, scp These are standard UNIX utilities provided on the tux machines by the CSL. Read the man pages if you are unfamiliar with these programs.

nmap Nmap is not installed on the lab machines.
We have a copy available at ~`giffin/public/642/nmap`. Run nmap with no arguments to see command line options.

attack Attack exploits a programming fault in an X windows program.
Available at ~`giffin/public/642/attack.c`. You will need to compile the attack program yourself.

If any of your experiments produce errors such as "Destination host unreachable," then `vulnerability` may have crashed. Email `giffin@cs.wisc.edu` so that I can restart the machine. (This is a good reason not to wait until the last minute to do this homework problem.)

This problem has 13 parts. Please submit answers for all parts A through M.

Part A: Make sure the target machine is running. *What is its IP address*?

Part B: Use `nmap` to learn what ports are open on the target. **Scan only the tcp ports 1 to 80**. Use the **Polite** scan rate provided by `nmap`. Note that this scan may take up to two minutes to complete. *Write down the nmap command line that you used (ie. include all command-line arguments). List the open ports that you find.*

Part C: *What does an open port signify*?

Part D: *Why would a port be listed as "filtered"*?

Feel free to play with other command line options on `nmap`, provided you target only the machine `vulnerability` and do not scan at a rate faster than **Polite**.

Part E: Network programs often advertise their version numbers. `vulnerability` is running an ssh daemon. Use `telnet` to determine version numbers for both ssh and OpenSSH. *List those version numbers*.

Part F: *How might this version number help an attacker*?

Part G: Unfortunately, we do not have an exploit for ssh. However, the telnet daemon is interesting because usernames and passwords are sent in the clear when someone logs in. Being a wily hacker, you sniff packets off of the network and find that someone with the username **billg** logs in with the password **trustworthy**. This is your opening. Connect to `vulnerability` using the stolen login. Once you are connected, type `whoami` at the prompt to see your username. *Write down the current username*.

Try to run the program `/root/642prog`. This should fail because billg does not have the

permissions necessary to execute programs in the directory `/root`.

Part H: The program `attack` exploits a vulnerability in certain X windows programs in a particular Linux release. You need to determine if this target is vulnerable. *What Linux kernel version is running*? *What Linux distribution and version is running*? *How did you determine this information*?

Part I: Verify that a process called "X" is running on the target machine. If X is not running, the attack cannot work (let us know please). *How did you determine this information*?

Part J: Read the source code for the attack. *This attack falls into what category of attacks*? *What should be the outcome of a successful attack*? *How does the attack work*?

The attack program is a local exploit: it must be run on the target machine. Make a subdirectory for yourself anywhere that you have access to. Now, upload the attack program into the new directory you created with the UNIX `scp` command. Compile the attack if needed. Figure out how to use the attack program.

Part K: A vulnerable setuid "root" program is particularly dangerous because attackers may exploit the program to gain root privileges on the machine. Find an X windows program on `vulnerability` that is setuid root. Run the attack program against the target program that you found. If the attack is successful, you should see a funny looking command prompt. (There is at least one vulnerable program on the machine). *What program did you exploit*? *What is your username following the exploit*?

Part L: You can now run arbitrary commands on `vulnerability`. **Please do not do anything destructive**. I will be unhappy if I have to reinstall the operating system or clean garbage off the hard drive. Type `/root/642prog`. *Do you have permissions to run 642prog*?

Part M: *Write down the four hex numbers output by 642prog*. This is your verification to us that you completed the experiments.

Time to clean up. Exit from the current shell to run as billg again. Delete the attack program you uploaded and the directory you created, and disconnect from the target machine.

References

[MOV97] A.J. Menezes, P.C. Van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography*. CRC press, 1997.

[Sta03] William Stallings. *Cryptography and Network Security: Principles and Practice*. Prentice Hall, 2003.