

ISA 656, Assignment 3

GENERAL INSTRUCTIONS

This homework has two parts: one part with a set of theory questions and a programming part. Programming can be done in either C or Java. Submissions should include the **annotated** source code. Programs that do not compile will get a low grade. Make sure your programs do not crash when given bad input, but rather produce a useful warning or error message and take the appropriate action (recover or quit).

SUBMISSION INSTRUCTIONS

The compressed files (either tar or zip) will be submitted using email to astavrou@gmu.edu.

A compressed (zip or tar, and gzipped) file named {your last name}-hw3.tgz or {your last name}-hw3.zip containing a subdirectory for each problem, named p1, p2, etc. Each subdirectory should contain:

- All source code, including any test programs, for the problem, if required.
- A file answering any questions posed.
- Any additional files required.

Please make sure that you send me a **single compressed file with all the documents and code. The compressed file should not exceed 1MB.**

INTRUSION DETECTION USING SNORT (70 points)

In this problem you will learn to use Snort, a popular intrusion detection system.

Read the Snort user's manual and FAQ at

<http://www.snort.org/docs>

Command-line switches that you should pay particular attention to include the following:

-r tf Read and process tcpdump file <tf>.

-l ld Log to directory <ld>.

Snort will fail without these arguments, unless you have root privileges on your machine. Use the traffic dump file, hwk3_s08.dmp (on the class web site, see below) as the argument to the -r switch and a directory of your choosing for -l.

Detect an intrusion

Download from the course website:

http://www.cs.gmu.edu/~astavrou/courses/isa_656_S08/hwk3_s08.dmp

This file contains a tcpdump listing of all the traffic (some interesting, some not so interesting) to and from dsl.gmu.edu.

Your goal is to determine what attacks, if any, this machine made or sustained during the duration of the traffic recording.

After making sure that you have created an appropriate log directory (see above!), run Snort on an appropriately defined set of snort.conf files to detect the following:

- All TCP and ICMP traffic to and from an IP address of your choosing.
- All TCP traffic to port 80.
- All TCP traffic on port 80 to an IP address of your choosing.
- A port scan. What is the range of ports in the scan?

- A worm. Use the worm signature provided in file [hwk3_worm_s08.txt](#) from the class web-site

Submit in a document file with a discussion of what you found and a copy of your rules; do not submit their output.

Extra credit

List all attacks against dsl.gmu.edu and their frequency.

PROGRAMMING (30 points)(+10 points extra credit)

Augment your Chat utility to use hybrid encryption. Hybrid encryption is a method for combining the strength of public key cryptography with the performance of a symmetric key block cipher. TLS/SSL do use that as part of what we call a Cipher suite:

<http://java.sun.com/javase/6/docs/technotes/guides/security/jsse/JSSERefGuide.html> – SSLOverview

PART A:

Create (or augment your existing utility) a utility that generates an RSA public/private key pair. The server uses this utility to generate a key pair and sends the public key to the client using the Diffie-Hellman key agreement protocol. The client encrypts the data using the server's public key.

PART B:

Create a chat encryption utility that uses the RSA from Part A to encrypt a communications using Java Secure Sockets. The chat server should be able to accept multiple connections and broadcast the communications to all connected users using secure sockets to each of the clients.

In addition, each connected user should receive a list of the other users and be able to select which of the users will receive his message by defining it at the beginning of the message. For example, assuming users test1, test2, and test3 are connected then test1 can communicate privately with test2 only by typing:

(in test1's terminal):
[test2] Hello test2

similarly for communicating with test3:

(In test1's terminal):
[test3] Hello test3

For this last part you should maintain a synchronized database of the connected users both on the server and on each client. This database should be updated each time a user joins or leaves the chat server. Each user should be able to see the list of connected users by typing "[show users]".

In order to be able to send both messages and control commands without having to create a complicated protocol, you can use two sockets for each client: one for control commands and one for messages.