Web-Based Secure Messaging System

Tom Kelliher, CS 325

100 points, due Sept. 27 and Oct. 4, 2006

1 The Task

You've just been hired by KeepingSecrets, Inc. to develop a free, Web-based crypto system for the public (any number of users). The system will be RSA-based and implemented using Perl CGI scripts. KS's CEO wants the system to provide the following functions:

1. Register a new user on the system, generating a private key/public key pair for the new user, and storing the pair.

   In addition to the key pair, the system stores username and password for each user.

2. Login and authenticate to the system as a returning user.

   A user may only send or receive messages once they've logged-in and authenticated themselves.

3. Securely send a message to another user on the system. The receiver should have clear proof of the sender's identity.

4. Securely retrieve a message sent by another user. Once a message has been retrieved, it should be deleted immediately.

2 The Details

1. You may work alone or in groups of size two. You must document all assistance you receive from anyone other than me.

2. You will find the CGI and Crypt::RSA Perl modules of great use. Documentation is available on the Web at `search.cpan.org` or on phoenix using perldoc from a shell.

3. Data may be stored in flat files or in a database (PostgreSQL). In the latter case, you will want to have a look at the DBD::PG and DBI Perl modules and have me create a database user on phoenix for you.

4. You're going to need to determine how to do file or database I/O, as well as learning to take advantage of the functional elements provided to you within a Web form, pretty much on your own. Given that this is a 300-level class, I think this is a reasonable expectation.

   Each group will make a brief presentation on the 27th. This presentation will address:

   (a) The look and feel of the UI.

   (b) How data will be stored and how it will be secured.

(c) An initial analysis of security risks from the following attack locations:

i. The application at the UI level.

ii. The Internet at the TCP/IP level.

iii. A non-root user on the host system with only shell access.

iv. A non-root user on the host system with only CGI access.

v. The root user.

Your security design should make use of principles we've discussed in class, such as a layered defense. You are to hand-in a written copy of your initial security analysis at the beginning of class.

5. Each group will make another brief presentation on the 4th. This presentation will address:

   (a) Use of the system.

   (b) Data storage within the system. (Be prepared to show actual, live file or table data.)

   (c) A final analysis of the security risks enumerated above.

6. On the 4th, each team is to turn in hard-copy of all source code, including any static HTML files, and a written analysis of security risks. Source code must be documented to the extent that a person not familiar with the code can quickly and easily become familiar with, and understand, the code.

7. Some of the things you should be pondering as you work on this project are: finding good places in the system to hide backdoors that would allow you to secretly alter data, triggering the backdoors without arousing suspicion or affecting the overall integrity of the system's data, and obscuring the backdoors from others who would be using the system or studying your source code.