Project I
CS 6238: Secure Computer Systems
SecLogin: Stronger authentication using behavior based questions

Due Date: 2/22/2013

Maximum Number of Participants per team: Two

In this project, our goal is to implement a more secure authentication scheme (we hope so) using techniques that are inspired by the password hardening paper that was discussed in class. In particular we will explore some ideas mentioned in the active authentication DARPA program that mentions behavioral features (more information about this program is at http://www.darpa.mil/Our_Work/I2O/Programs/Active_Authentication.aspx). Although somewhat contrived, to simplify the project, we assume that the system is able to estimate certain behavior features of its user population (similar to latency and duration thresholds in the keystroke based hardening paper). It uses these features to determine if the correct user is trying to login. After the user types a password, the system asks the user a number of questions in a set of categories (e.g., how far is the user from Georgia Tech campus – can also use IP geolocation, anticipated duration of login session, how many emails will be sent etc.). A user may choose to answer a question or decline to answer it. If the value supplied as an answer is below or above a threshold value, the corresponding feature value is distinguishing. The feature value is undefined when the user declines to answer the question or the value supplied is too close to the threshold. A history file is maintained to keep track of recent answers. If an answer matches a user's typical behavior for feature *i,* a correct value is stored for either α_i or β_i (but not both). If user frequently declines to answer a question or the answers do not show a consistent behavior patten, the instruction table stores good values for both entries.

Initialization:

The initialization will require following steps:

1. Select a 160 bit prime value q and choose a random H_{pwd} value such that $H_{pwd} < q$. Also choose a value h for the size of the history file (h is the number of recent feature value vectors that are stored in the file).
2. In this scheme the number of distinguishing features will be at most equal to number of questions. If number of distinguishing features is *m*, then select a random polynomial *f* of degree *m*-1.
3. Using the polynomial *f*, generate Instruction table such that all α and β values are valid, i.e using either entry, a correct H_{pwd} can be calculated. To calculate α and β value use the password and the formula given in paper.
4. Create a fixed size History file. Select its size, pad it and encrypt it with H_{pwd}. Use suitable redundancy to ensure so you can find out when encryption is successful.

Login:

The entire authentication process will involve following steps.

1. Select appropriate value i.e., α_i or β_i from the instruction table based on the feature values $\phi(i)$ according to answers provided to different questions.
2. Extract the value selected from instruction table using password supplied.
3. Calculate the x_i, y_i coordinates based on the α_i or β_i values selected from the instruction table using the formula given in the paper.
4. Calculate value of H_{pwd} using the coordinates using polynomial interpolation.

5. Decrypt the history file using H_{pwd}. If decryption is correct then you have calculated the correct H_{pwd} and hence the login request is from the legitimate user.
6. Once you have authenticated the user, you will add the feature values for the current login in the history file and encrypt the history file again with a new H_{pwd}.
7. Select a new random polynomial *f'* and using it generate a new instruction table.
8. Compute each α and β value using password as key.

Some Important points:

- The history file should be saved to disk only after it is encrypted. Decrypted version of file should not be stored on the disk. Also the size of the file should remain constant; if the size is less than the selected size then the file must be padded before encrypting it.
- Random polynomial *f* of degree *n*-1 is defined as :
  $f(x) = hpwd + a_1x + a_2x^2 .... + a_{n-1}x^{n-1}$ where the values a_1 to a_{n-1} are chosen randomly.
- The password value can be used as it is, as key or a hash of the value can be used as the key. This choice is implementation dependent which you are free to choose.
- In calculation of α and β, G_{pwd} is a keyed hash function. You are free to choose the hash function you want to use and how you use the key supplied to it to calculate the keyed hash value.
- You can implement the project in either C/C++ or Java.

For grading, you must have a functioning implementation to receive credit. Partial credit may be awarded when some of the steps are not completely implemented. In addition to source code (it should be well commented and checked for any code vulnerabilities), you should provide information that shows that all required features are implemented.

You will be required to demo your implementation to TA. You will schedule a time for such a demo after the project is completed and submitted via t-square.