Security Engineering, Assignment 4

Due date: April 12, 2015

Encrypted files with message authentication code (MAC) (Total points 50)

Part 1.

The objective of this assignment is to familiarize you with using OpenSSL library routines. You would need to start with the set-up you used in assignment 2. To this you need to add some new programs like 'fget_decrypt', 'fput_encrypt'. The 'fput_encrypt' creates a file, using text entered through standard input while storing in encrypted in the file name specified as a program argument. The file contents need to be encrypted using symmetric encryption such as AES. Upon executing the command 'fput_encrypt', it should prompt the user for a passphrase that is to be used to generate the salt, IV and key to encrypt the file. The file should then be encrypted using the generated salt, IV and key.

Similarly, the 'fput_decrypt' program should prompt the users for the passphrase from the user to decrypt the file and should print it out for the user. Again, these programs should be setuid programs which could access the users' files. These programs should also switch to the effect users' ID using seteuid() (or related) systems call(s). The file header should additionally have a HMAC field which should store the **encrypted** HMAC of the file. This should be recomputed every time the file is modified. When the file is being printed out using 'fput_decrypt' the system validate the HMAC to check the integrity of the file (to make sure it has not been inadvertently or maliciously modified).

Part 2.

The second part of the assignment involves encrypting the file contents and generating the MAC using RSA trapdoor function. The users would need to generate a random number. To do so you would need to add a program called 'gen_rand'. It should take the length of the random number as an argument and also the file to store the output. The random file should be encrypted by the user using standard symmetric key encryption (e.g. AES) and stored in the home directory. Whenever required, this file should be used to perform the RSA trapdoor function. To perform the RSA trapdoor function, you need to implement a separate program called 'fput_encrypt_rsa'. This should take the following input:

1. The random number generated above using 'gen_rand' and the output filename. The program should first decrypt the random file and obtain the random number.
2. RSA public key as input. In the absence of the public-private keypair, the program should gracefully exit. The user should be able to generate the keypair using OpenSSL function. These files should be generate such that only the owner has read and write permissions, while the others have none.
3. Destination for the encrypted output– either to be displayed on standard output (terminal) or to be written to a file.

Similarly you are also required to implement the corresponding 'fget_decrypt_rsa' program, which should be able to reverse the trapdoor function, decrypt the file and print it out to the terminal (or standard output) as the case may be.

The encryption should be performed using symmetric encryption scheme, while the MAC should be computed through the public private key-pair.

Like the previous assignments, you must correctly handle all errors and corner cases.

To demonstrate the correct functionality of the encryption schemes, the TAs should be allowed to modify the files. The decryption programs should be able to validate the MACs and gracefully terminate if the validation fails.

You would not be evaluated for the functionalities which were already demonstrated for grading assignment 2.

What you are supposed to submit:

1. C source code for the aforementioned shell server and client programs.
2. Makefile through which one could compile these programs.
3. A write up of what your system does, what all assumptions you made, the inputs that you used to test your program and all the errors that you handled.

Grading scheme:

1. Successful compilation using Makefile – 10 points
2. Use of OpenSSL library routines to perform the symmetric encryption, calculation of hashes and MACs, generation of random numbers and RSA public-private key-pairs. – 20 points
3. Successfully defending against atleast 3 attacks/bugs/errors (e.g. Detecting unwanted modification to files, determining if the file is being correctly decrypted (in case of symmetric cryptographic schemes, handling absence of files e.g. absence of random number file (wherever necessary) etc.) – 15 points (List the bugs/errors/attacks that you defend against)
4. Description of the systems, commands to execute and test the program and the assumptions that you made. – 5 points