Monkey-in-the-Middle Attack
Programming Assignment #2
CptS 425/580—Network Security
Due: 15 April 2009, 11:59 PM
(Revised 3 April 2009)

Objective:
Implement a monkey-in-the-middle attack. Subcomponents of this task will include the creation and usage of public/private encryption keys.

Overview:
Alice and Bob communicate with each other over a link that is controlled by Eve. This will be simulated with three processes and four file directories. Process ***A*** (for Alice) will open up a socket connection to Process ***E*** (for Eve). In *non-attack* mode, ***E*** will receive encrypted messages from ***A*** and forward them on to Process ***B*** (for Bob). In *attack* mode, ***E*** will decrypt ***A***'s messages then re-encrypt them with a falsified key before forwarding on the message to ***B***. Processes ***A***, ***B***, and ***E*** will each have a directory where they will store their respective private and public keys. (Assume that these directories are readable by only their respective 'owners'). A fourth directory will contain all of the public keys to be used.

Software Design & Implementation Tasks (75 pts)

1) Key Creation and Certificate Management (15 pts)
Create RSA private and public key pairs as needed by Alice, Bob, and Eve. The public keys should be stored in X.509 certificates.

Minimum Requirements: You must demonstrate the ability to create public / private key pairs and the ability to printout existing X.509 certificates. Your write-up must include a listing of one set of keys for Alice, Bob, and Eve's 'Alice' and 'Bob' counterfeits. You may either implement this code yourself, or you may use existing tools (e.g., openssl, Java's keytool, Crypto++, etc).

2) Program(s) Alice and Bob — `alice & bob` (25 pts)
Design and implement a pair of programs that will send and receive encrypted messages over a TCP/IP socket connection. All messages sent should be wrapped with <MSG> … </MSG> tags before encryption.

Minimum Implementation Requirements: Command-line argument(s) should be used to indicate whether either Alice and Bob should generate new key pairs (-g) for their private use (and write them to the appropriate directories) or use an existing set of keys (that are stored in their respective directories). All third-party public keys should be read from the public key directory. Command-line arguments should be used to specify which host (-h *host*) and port (-p *port*) to send messages to (required by Alice) and which port (-p *port*) to listen on (required by Bob). Alice should be able to send a message to Bob either directly or via Eve. Print an error message if a mismatched key was used to decrypt a message (e.g., the decrypted message does not begin with <MSG>). Alice's message may be specified on the command-line (-m *message*) or entered by the user.

3) Program Eve — `eve` (35 pts)

Design and implement a program that will forward messages from Alice to Bob. In non-attack mode, Eve will simply forward messages without changing them. In monkey-in-the-middle mode, Eve will generate new, falsified public keys for both Alice and Bob so that she may launch a monkey-in-the-middle attack. The intercepted message's plain text should be printed to the screen (or a file).

Minimum Implementation Requirements: Command-line arguments indicate the port that Eve will intercept Alice's messages (-i *interceptPort*) and the host (-h *host*) and port (-p *port*) that Even should use to forward/send messages to Bob. The MITM argument (-m *mitmMessage*) when present indicates that Eve should generate counterfeit key pairs for Alice and Bob (and write them to the appropriate directories) as well as modify Alice's original message with the *mitmMessage*. When in MITM mode, Eve should print the decrypted plaintext message that Alice is sending Bob. When not in MITM mode, Eve forwards Alice's cipherText to Bob.

Written Deliverables (35 pts)

1) Cryptographic Checksum of your Source Code (0 pts)

Compute an md5 or SHA1 checksum of a tar file (or similar archive) of your source code and executable image. Include the name of this file, it's size, and it's checksum in your write-up. Assignments turned in without a checksum will be subject to a grading penalty.

2) Abbreviated Software Design Document (10 pts)

Briefly describe the data structures and algorithms used to implement the three required tools. Give the sources of all third-party code and cryptography tools/mechanisms that you used in this assignment. (1-2 page limit)

3) Software Test Document (15 pts)

Describe your software test plan and methodology that you used to verify that you implemented the three required software tools correctly. List any know deficiencies. (1-2 page limit) Include an appendix that lists the various private / public keys used for one run of your monkey-in-the-middle attack. (Page limit will vary based on font size, etc.)

4) Lessons Learned / Project Evaluation (10 pts)

Discuss the lessons learned from this project. Are there improvements that should be made in a future release? Other possible questions to address: Was this assignment as easy as you anticipated? Did this assignment adequately introduce you to the complexities of key management? Etc.

5) Time Logs (Optional, no points)

List the time spent during software design, implementation, testing. Also list the time spent writing up your project.

Additional Notes:

- You may implement this assignment in the programming language of your choice.
- If a set of test files is given, they will be given at least one week before the assignment is due. You will not receive full credit if your programs cannot correctly process these files. But, these files by themselves may not be sufficient to fully test your programs.
- Program demonstrations will occur in the department's computer lab. (This does not mean that your code has to execute on a departmental machine, you may ssh to another host or bring in a laptop if you choose).