Matt Pletcher
Ayman Hammad
12/4/2017

ATM/Bank Design and Implementation

I. Overall Protocol:

A. Our security protocol focused on creating public/private key pairs for the bank and ATM and allowing them to use the keys to communicate messages securely.

a. The init program creates the <path>.bank and <path>.atm files which are critical to this protocol. The <path>.bank files holds the public key (K_{pub})of the Atm along with the private key (K_{priv}) of the Bank. Likewise, the <path>.atm file holds the K_{pub} of the Bank and the K_{priv} of the Atm. This allows the bank to encrypt messages sent to the Atm with the Atm K_{pub} so that only the Atm can decrypt the messages since only it knows the K_{priv}, and same with messages sent from the Atm to the Bank. In the end, we ran into a number of issues with the encryption and decryption and were unable to implement it effectively for all messages being sent and received.

b. The <user>.card files created by the bank with create-user have their contents encrypted with the Atm's K_{pub}. Attackers have access to these cards, so we encrypt them with random padding so the contents of the cards are not obvious nor can they be recovered easily without the Atm's K_{priv}. The .card files only contain a user's pin and no other information, serving as bait for attackers to try to figure out what is actually stored in them but will not give any useful information if discovered.

c. All command inputs on either system are checked for validity via regex checking exactly on the characters allowed for each field, so attempting a buffer overflow attack would be very difficult as the necessary characters for the attack wouldn't be accepted.

d. The protocol was also going to include signatures using the public/private keys to sign and verify messages from either system. We managed to implement the functions to sign and verify signatures, but couldn't put it into actual use because of how similar a signature and encrypted plaintext look in a string, we thought it would be too much time spent to try to test differentiate between the two and scrapped the idea.

e. For Bank-Atm protocol we wanted to encrypt messages before sending it to the bank and allowing the bank to decrypt but we were unable to do so. We also wanted to sign any messages to prevent attackers from impersonating the ATM and send messages. From the current protocol the ATM sends a request to the bank and the bank receives the command, based on the command it returns if the command was successful or not. Based off the response from the bank, the ATM outputs the appropriate response.

II. Threat Model:

A. Buffer overflow attacks
   a. In the project description, the code is required to run without stack guard meaning it will be vulnerable to buffer overflow attacks. To protect against buffer overflow, we use fgets and sscanf to limit the amount of input taken in by the user, and all string functions are done with the functions requiring a maximum length to ensure only the number of characters needed are used. Although this is not 100% protection, it is good protection against most buffer overflow attacks, especially since attackers can't directly inspect memory in either system.

B. Attempting log in without the <user>.card file
   a. Attackers can create a user and know their pin at the Bank, but if they want to modify the <user>.card file to attempt an attack, the Atm will not allow entry since the card will not decrypt to a valid pin, or even the correct pin if tampered with and the user will not be authenticated at the Atm.

C. Remote Deposit attack
   a. Attackers cannot remotely deposit money into a user's account by either using the Atm's withdraw or sending their own withdraw packet to the bank. Negative amounts are checked in both and the command is ignored if the amount present in negative.

D. Attempting to use a fake ATM card
   a. An attacker can try and to get encrypted bank cards and try to pass it into the ATM to get useful information, but since we use RSA public/private key encryption with padding, the ATM cards does not return any useful information to assist with a Chosen Cipher Attack. So, attackers cannot create their own fake card and try to pass it in.

E. Man in the Middle attack
    a. If we were able to finish the encrypt and decrypt we would be able to protect against this attack. This attack is similar to active communication channel attack however it refers to an attacker being able to intercept messages and altering them before they reach their destination. This attack is prevented as well because the messages are encrypted so the attacker would have to decrypt the message to be able to alter it or else the message will not be useful for the attacker.

III. Vulnerabilities:

A. Modifying messages
    a. An attacker can modify messages sent from the Atm to the bank and withdraw X dollars, as long as the user exists in the bank and has at least X dollars in the bank.
    b. An attacker can view messages sent between the Atm and Bank and freely edit them, if we had finished encryption/decryption this would not be a vulnerability.

IV. Ignored Attacks:

A. Stack Exploits
    a. The attacks are not allowed to access the bank system or the memory of the Atm, so we do not need to worry about any stack exploits.