Cryptography and Network Security 1 (CSCI4971) - Final Project [FINAL DRAFT]

Prof. Bulent Yener
Rensselaer Polytechnic Institute
110 8th Street
Troy, New York U.S.A. 12180
yener@cs.rpi.edu

September 28, 2011

1 Introduction

This project involves designing and implementing a protocol for communicating between ATMs and banks. Once you have finished securing the design it will be given to another team for testing. You will meanwhile be given the opportunity to attack another team's protocol.

2 Protocol Design

You will design and implement three programs: an ATM, a proxy, and a bank, which communicate over TCP sockets.

- The ATM should take one command-line argument, the port number to connect to the proxy with.

  The contents of user XXX's ATM card should be represented by a file XXX.card.

  The ATM should provide a command shell supporting the following commands:

  - login [username] - prompts for PIN and, if correct, establishes a session
  - balance - prints the current user's balance
  - withdraw [amount] - verifies sufficient funds, then prints "xxx withdrawn" and debits the balance accordingly
  - logout - terminates the current session

  - transfer [amount] [username] - verifies destination user exists, then transfers the requested amount

- The proxy should take two command-line arguments: the port number to listen on, and the port number to connect to the bank with.

  For now, the proxy will simply take messages from the ATM and bank and forwards them to the other necessary. During the attack stage, you will be able to modify the other team's proxy to record or tamper with communications.

- The bank should take one command-line argument, the port number to listen on.

  The bank should provide a command shell supporting the following commands:

  - deposit [username] [amount] - deposits money in a user's account
  - balance [username] - prints a user's balance

2.1 General requirements / notes

- Only one bank and proxy may exist at a time, however multiple ATMs may connect to the bank simultaneously.
- Account balances etc must be maintained by the bank, not stored on the ATM.
- Three users must be created when the bank starts up: Alice (balance $100), Bob (balance $50), and Eve (balance $0).
- Your programs must be written in C or C++ and run correctly on 32-bit Linux.
- You may not use any existing security protocols (SSH, SSL, etc). (Borrowing design ideas from existing protocols is entirely acceptable.)
- Cryptographic primitives must come from a standard library (Crypto++, OpenSSL, etc). Points will be deducted for re-implementing any ciphers, hashes, etc in your code. (The focus of this project is system design, not on proper implementation of cryptographic primitives.)
- You must provide a full description of your protocol in addition to your code, in sufficient detail that a third party could create a compatible implementation.
- You must submit 32-bit Linux binaries of your code for use by the attacking team, as well as complete source code. If your program contains any hard-coded master keys etc, the relevant portions of the submitted source code should be sanitized (all zeros, etc).

3 Attack

After the systems have been implemented, you will be given the code and documentation of another (randomly selected) team. You will then be expected to inspect their protocol for vulnerabilities and develop exploits for any that you find.

3.1 General requirements / notes

- Patching or modification of the bank/ATM binaries is not allowed.
- Recovery of secret keys from the bank/ATM binaries by reverse engineering is not allowed. (These two requirements simulate tamper-resistant hardware deployed in a real-world situation - while an attacker can often obtain another unit of the same machine to destroy in his lab, it is very difficult to alter or disassemble an active installation without being caught.)
- You may freely modify the proxy to tamper with, drop, duplicate, or create new messages as you see fit.
- Any and all protocol level attacks are allowed, including but not limited to buffer overflows and integer boundary handling exploits. (These two equirements simulate a real-world environment in which an attacker has obtained access to an intermediate router between your sites.)
- Submit a full report on vulnerabilities tested (including any that are present but you could not exploit for some reason) in sufficient detail to enable a third party to recreate the attack.
- Bonus points will be awarded for especially creative or devastating attacks.

4 Due dates

To be announced.

5 Grading Policy

Your project grade will be divided into two parts, weighted equally.

The defensive part of your grade will be based on an evaluation of your protocol's security by the professor and TA, as well as by how well it survives attack by your classmates.

The offensive part of your grade will be based on how effective you are at breaking the opposing team's protocol. If you are unable to find any exploitable vulnerabilities, submit a report describing in detail every attack that you considered and why their protocol is immune to it.