Cryptography and Network Security Final Project - Blackhat

Ryan Prashas Leith Reardon Dylan Ross

August 18, 2020

Contents

Introduction

The project that we were given is written in C, and uses various libraries in order to implement a large number of options for both symmetric and public key cypto systems in order to implement a pseudo-ssh algorithm in order to create a secure connection between a bank and an atm. Below are our attacks on the implementation.

1 Password Management

Done by Dylan Ross

In the project we were given to attack, the first vulnerability that we found became apparent immediately upon reading the documentation. The code does not make use of passwords of any kind, and instead allows the ATM to check the balance of, deposit into, and withdraw from arbitrary accounts without authentication. The could trivially be leveraged by an attacker to steal money from other people's accounts by selecting a different account during a withdrawal.

It could be said that this attack is outside of the scope of attacking the cryptographic communication between an ATM and a bank as the ATM could be handling user authentication locally, but that would likely not be feasible. In order for that scheme to work, every ATM would need to locally store the credentials of every user in the bank, which would require a significant amount of memory and as such would drastically increase the price of the ATM machines themselves. Additionally, this scheme would require updating the local storage of every ATM owned by the bank whenever a new user account is created, which would cost a lot of time and money.

2 ATM Authentication

Done by Dylan Ross

Another vulnerability that we found in our assigned project is that the bank never authenticates that the client is an actual ATM once a connection is established. As such, there is nothing to distinguish an attacker connecting to the bank from their laptop from an ATM that can ensure that various rules are being followed. For example, we assumed that the ATM would realistically check that the user actually inserted money before sending a message to the bank saying to deposit money into their account. However, an attacker being able to connect using a different device can send arbitrary deposit messages to the bank in order to add money into their account and withdraw it from an actual ATM.

This attack, when combined with the lack of passwords mentioned in section 1, could also allow an attacker to bypass any local credential checking done by the ATM in order to send withdraw messages with a different account number, which would empty the target account's balance without the money going to anyone else. However, even if passwords were implemented on the bank side to prevent such an attack, an attacker would still be able to arbitrarily deposit money into their own account as they would know their own credentials and be able to send them to the bank.

We were able to generate a working proof of concept for this attack by creating a copy of the atm.c file and modifying it slightly for ease of use and to remove parts that were unneccesary for the attack. After compiling this new file with the same flags used for atm.c (done by copying from the make file), we could send arbitrary deposit messages and received the correct responses from the bank. However, this attack could not be shown very well using the current implementation of the bank because the bank does not store account balances at any point. Instead, it initializes every account to a pre-determined balance at runtime. As such, our proof of concept cannot be shown to work between sessions on this implementation, but neither do any legitimate transactions done by the ATM itself. Due to the theory behind the attack and the proper response codes received by the proof of concept, there is no reason to believe this attack would not work on an actual bank implementation that saves user balances.