CS5430 – System Security Spring 2018

Assignment 5

Due: May 7, 2018

Problem 1: Logging[1]

Your task in this problem is to design the logging system for an automated teller machine (ATM),

ATM functional requirements. An ATM comprises the following subsystems: (1) a processor, which performs computation, (2) a magnetic card reader, (3) a keypad, (4) a monitor, (5) several bill storage canisters, one per denomination, (6) a bill disburser, (7) a thermal printer for customer receipts, (8) a system clock, (9) a slot through which customers can deposit envelopes, (10) a communication link to a bank-operated network, and (11) a physical key-operated switch to enable an operator to start and stop the machine. The ATM provides service to customers only if the key-operated switch is set to the "on" position. When an operator turns the switch to that position, the operator is required to enter into the keypad the amount of cash in each canister. (The denominations of the canisters are known to the machine.) The switch can be turned to "off" only if there is no customer currently in the midst of a series of transactions. When off, no customers will be served, and the operator can remove deposit envelopes, reload the ATM with cash, and replenish the paper for receipts.

The ATM will not provide service to customers unless the communication link is active. If the link ever is inactive, the ATM cancels any in-process transactions and refuses to begin any new transactions.

To receive service, a customer inserts their ATM card into the reader. The ATM reads the magnetic stripe and returns the card to the customer. The customer then enters a PIN on the keypad. The ATM authenticates the card and PIN, which requires using the communication link to the bank. If authentication fails, the ATM re-prompts for the PIN. After three incorrect attempts, the ATM communicates to the bank that the card should be deactivated. If authentication succeeds, the bank retrieves a set of accounts that are associated with the card. The customer can then perform a series of transactions with those accounts, which continues until the customer indicates they are finished or until the ATM detects that the customer has failed to respond to an input prompt within a period of one minute. At any point a customer may abort a transaction in process by pressing a Cancel key on the keypad instead of entering the next input requested by the ATM. If a transaction succeeds, the ATM prints a receipt for the customer.

The following transactions are supported:

1. Withdraw cash from an associated account in any denominations supported by the ATM. If the ATM does not have sufficient cash available, it cancels the transaction. Furthermore, the bank must authorize the debit of funds. The ATM does not participate in that authorization decision.

2. Deposit funds into an associated account. The customer places cash or checks in an envelope and enters the amount of the deposit on the keypad. The bank will later decide whether the claimed amount is consistent with the funds available from the envelope and, if so, will credit the account. The ATM does not participate in that decision.

3. Transfer funds between associated accounts.

4. Query the bank for the funds available in an associated account.

[1] adapted from [Wang et al., The formal design model of an automatic teller machine (ATM), IJSSCI 2(1):102–131, Jan–Mar 2010], from Prof. Russell Bjork (Gordon University), and from Prof. Michael Clarkson (Cornell University)

What to do. Your design should answer the following questions:

- What log files will be kept, and where?
- What is the format of an entry in a log file?
- What are each of the possible entries that could be made into a file? Specify the information contained in the entry and the conditions under which the entry will be made. The entries should cover at least 10 different types of events.

You are permitted to propose modest hardware improvements to the ATM for the purpose of logging.

What to submit: a pdf file `logging.pdf` containing your solution

Problem 2: Tamper-proof Logs

Consider the following revisions of the tamper-proof log protocol discussed in class. In each, what new attacks would be possible against the protocol? What attacks would still be impossible?

Part A

```
1. ek = H("encrypt", ak)
2. x = m, MAC(m; ak)
3. record x in log
4. ak = H("iterate", ak)
```

Part B

```
1. ek = H("encrypt", ak)
2. x = Enc(m; ek)
3. record x in log
4. ak = H("iterate", ak)
```

Part C

```
1. ek = H("encrypt", ak)
2. x = AuthEnc(m; ek; ak)
3. record x in log
4. ak = ak  // i.e., no change to ak
```

What to submit: a pdf file `tamperproof.pdf` containing your solution

Problem 3: Auditing[2]

Write a program whose purpose is automated review of a Linux system log. Your program should take as input the file /var/log/auth.log from Ubuntu 16.04. Then, your program should process the log to determine whether an attacker has attempted to login to an account for which he is guessing the password. Finally, your program should produce a single line of output that would be suitable to append to /var/log/syslog—that is, the output should be consistent in format and information with other entries in syslog. Somewhere in that line should be either the string "OK" or "INTRUSION" to indicate your program's determination. In case of "INTRUSION", the output should further estimate the number of users whose accounts the attacker attempted to access.

Since what constitutes an attack depends in part on the expected workload of the system, assume that the system is a server for a CS department at a small college: there are about 100 users, all of which commonly login at least once per day. Be clear in your solution about any additional assumptions you are making about the workload.

Hint: Here are some questions you need to address: What constitutes an attack? What evidence is recorded in auth.log? How can you parse auth.log? How can a program recognize an attack based on the evidence? What information should you put into syslog? How should you format that syslog entry?

Implementation: Your program should run on the same Ubuntu 16.04 virtual machine as in A3. As in A3, you may use any programming or scripting language that can be installed on the virtual machine you created according to the instructions above, as long as that language can be installed by apt-get.

Execution: Your program should accept the name (including path) of the file to be processed as a command-line argument. Do not hardcode /var/log/auth.log as that filename: the course staff needs to be able to test your program on many log files—as do you.

Design document: Provide a file named design.pdf containing an explanation of the design of your automated reviewer, focusing on (i) what specific criteria you use to determine whether an attack occurred and (ii) why you are using these criteria. Discuss false positives and false negatives in light of points (i) and (ii). This design document will be worth about half the total points for this problem.

Usage: Provide a file named README.txt describing how to install and execute your program. If you use any packages installed by apt-get, then you need to document at the beginning of your source code how the grader can easily install those packages in their own virtual machine. Document the exact command the grader should type in the virtual machine to run your program. It should not take more than a couple of minutes (excepting whatever time apt-get itself consumes) for the grader to follow the installation and execution instructions; if it does, your solution might be penalized.

What to submit: a zip file `authlog.zip` containing your source code, readme, and design document.

[2] adapted from Prof. Jim Martin (Clemson University)