SY308: Security Fundamental Principles

Calendar	Policy

Project: ATM Design and Implementation

Warning: Read BOTH PART2 and PART2a even before working on part 2

In part 2, you need to secure your system. After each team submits their code, each team will be given the chance to attack another team's implementation in part 2a.

Part 2: Securing your system

You need to secure your system (bank and atm) so that the aother teams cannot succeed in attacking your system in part 2a.

Secure storage

In an actual banking system, secret information is stored in a secure storage. To simulate this physical security, the project specifies the following files:

- ssBank.bin: Only the bank is allowed to access this file. No entity other than the bank is allowed to access it; in particular, an attacker cannot access it.
- ssATM.bin: Only the atm is allowed to access this file. No entity other than the atm is allowed to access it; in particular, an attacker cannot access it.

Submission

1. Prepare the following files.
    - design.doc: The document must **describe the detail of you protocol** that the bank and the atm exchange. In particular:
        - How does your system store the keys and PINs? (Don't give the actual PIN data in the document).
        - Where and how do you use cryptographic schemes?
        - What are the messages that the bank and the ATM exchange for each transaction? Describe the format of each message.
        - The structure of the cards.
    - bank.py, atm.py, and your other python files.
    - Alice.card, Bob.card, Carol.card: the card files.
    - AlicePIN.txt, BobPIN.txt, CarolPIN.txt: text files containing PIN of each user.
    - ssBank.bin, ssATM.bin: files simulating the secure storage.
    - log-bank.txt, log-atm.txt: log files proving your system works correctly.
    - real.tar: compression file containing all of the above files.

- public.tar: compression file containing the following files. The other teams will use this version to attack your system in part 2a.
    - design.doc and your python files
    - **Alice.card, Carol.card**. Don't include Bob.card.
    - **BobPIN.txt, CarolPIN.txt**. Don't include AlicePIN.txt
    - ssBank.bin, ssATM.bin, log-bank.txt, log-atm.txt

  Make sure the tar files contain the files correctly; download the tar files yourself from the submission site, and check if the tar files are correct.

2. Submit the following files:

```
real.tar public.tar
```

3. Use the following project name:

```
project02
```

4. **Deadline: 2359 on Friday 4/19**

Part 2a: Attack!

1. Criticism on the design document

Each team: grade the other teams' design document.

- For each team's implementantion:
    - Give the right points for the document. The full credit is 100 points.
    - Explain why you give the points. For example, indicate what's good and what's to desired.

Use the following format:

```
Team name: (specify the team whose system has this bug)
Overall grade: (out of 100)
Strength of the document:
Things to be desired:

================================================================================

Team name:
Overall grade:
Strength of the document:
Things to be desired:

================================================================================

Team name:
Overall grade:
Strength of the document:
Things to be desired:
```

```
...
```

Submission:

1. Prepare design-eval.doc containing evaluation of the system of all teams (except your own).

2. Attack!

Each team will be given public.tar files submitted by the other teams: The following table summarizes what you will be given.

	Card	PIN	Balance
Alice	Yes	No	100
Bob	No	Yes	100
Carol	Yes	Yes	0

What you can change

> In your attack, you may arbitrarily modify the two things:
>
> - the router code;
> - *.card files (You can try to modify Alice or Carol's card; or you can try to forge a card for Bob).

Successful attack

A successful attack will be any attack that results in a net outflow of money to the attacker. By way of illustration, examples of successful attacks would be (these are not exhaustive):

- Withdrawing any money from Alice's or Bob's account.
- Withdrawing $1 from Carol's account without making any prior deposit at the bank. (Note that although the attacker has Carol's ATM card and PIN, the bank starts out with Carol's account initialized to $0.)
- Depositing $1 to Carol's account and then withdrawing $2.

Additionally, if you break the robustness of the system, it also counts as an attack.

- Making the system crash. (In this case, the inputs should be usual/manual/valid. Writing some script to fuzzing in a lot of input to the system will not be considered a legitimate attack.)

Not an attack

You are not allowed to look at the contents of ssBank.bin and ssATM.bin. In other words, according to the open design principle, you are allowed to inspect the algorithms but **you're not allowed to get the key by looking at the contents of secure storage**.

> **Not an attack 1:** "We looked at ssBank.bin found out a key or a PIN!"

Another attack strategy that doesn't count is to find out Alice's PIN by online brute force attack.

> **Not an attack 2:** "We wrote some script that automatically tries to log in with every possible PIN, and found out Alice's PIN!!"

As we told before, we assume that the attack can control neither the bank nor the atm. In particular, it doesn't count if your attack requires the bank to restart.

> **Not an attack 3:** "We came up with a beautiful attack, but we need to restart the bank program."

Tips

When you secure your system in Part 2, **consider the potential attacks that you make launch on other team's system as well**. That way, you can make your system more secure than others, while being able to break other systems! Obviously, you'd better not tell your team's know-how to other teams.

Report format

Use the following format attacks.txt to report each attack. Note: **in the log section, give some explanation about what you did so the instructor can regenerate the attack scenario**. If the attack cannot be regenerated, then you won't get the points.

```
Team name: (specify the team you attacked)
Attack description: (describe how the system behaves erroneously)
Log: (give the detailed execution log so the instructor can regenerate the scenario)

================================================================================

Team name: (another Attack)
Attack description:
Log:

================================================================================

Team name:
Attack description:
Log:
...
```

Submission:

1. Submit the following files:

   ```
   design-eval.doc, attacks.txt, other files used in your attack
   ```

2. Submit your files. Go to hw submission server, and Use the following project name for submission:

   ```
   project02a
   ```

3. Deadline: It's due **1159 on 5/1** (Wednesday).

Final Grade for Part 2

1. (20 pts) Design documents.
    - (5 pts) Quality of your design-eval.doc file.
    - (15 pts) The instructor will read all design-eval.doc's, and give appropriate points to your design document.

2. (80 pts) The system behavior.
    - -8: for each attack
    - If n teams identify the same flaw, then each team will gain 8/n (or 4/n) points.