Engineering

Willard Rafnsson

IT University of Copenhagen

In this assignment, we will put ourselves in the shoes of security engineers. The goal is to gain experience in expressing security requirements, and in engineering security into software. Notably, we detect, and fix, violations of security requirements.

Problem 1 : Security Requirements

Sally: "It works! See? I told you we'd finish the system in a week! I'm deploying it."
Alice: "I don't know... shouldn't we at least run some tests?"
Sally: "Lighten up, Alice, it'll be fine! It's Friiiday! Come, let's go for some cocktails!"

PayBud is an online payment system that enables an account holder to transfer money to another account, and to transfer money between the account and a credit card. PayBud has the following *functional requirements*.

1. A user can create an account.
2. A user can send credit from his/her account to another account.
3. A user can transfer money between his/her account and a credit card.

Write *security requirements* for this system. This entails the following steps.

Part 1 Perform a *threat analysis* and a *harm analysis* of PayBud.

1. What are the *assets* of PayBud? (~ 1 sentence)
2. What are the *threats* of concern? Their motivation? Capabilities? (~ 2 sentences)
3. For each asset, what are the possible *harms*? Does this affect confidentiality, integrity or availability of the asset? (see *harm triples*). (~ 2 sentences)

Part 2 Write *security goals* for PayBud based on your analysis. These have the form "the *system* should {*prevent*, *detect*} *action* on *asset*." (~ 2 sentences)

Part 3 Write *security requirements* that are needed to satisfy these security goals. These are constraints on functional requirements. (~ 2 sentences)

Note: We expect to see one asset, two harm triples, two security goals, and two security requirements.

Problem 2 : Attack, Online

PayBud is vulnerable to an Online Dictionary Attack. Exploit this vulnerability by following the instructions provided alongside this assignment. (~4 sentence)

Note: Explain every step of your attack; include each command, explain what it does, explain why you are doing it (i.e. to what end), and display the output that shows that your attack is successful.

Problem 3 : Audit

Alice: "I have reports of fraudulent transactions! I told you we shouldn't deploy!"
Sally: "Shh, my head hurts... How is this possible? I tried everything in the client."
Alice: "You overlooked something. We need to know what's going on."

Modify the source code of PayBud such that it logs security-relevant events.

A security-relevant event is an event (i.e. an action of the system) that involves a security requirement. When logging such an event, you log

- the fact that the requirement was checked,
- whether or not the requirement was met, and
- the information that lead to that decision.

Log such events to standard output. The `slf4j` logger that is already in place does this; we recommend that you use it. Your task is to determine where in the code to add logging statements, and what to put into each logging statement.

Note: Consult the lecture slides on logging for tips on what to put into a logging statement.

Part 1 Which security requirements is the above attack violating? (~1 sentence)

Part 2 Show how the logging you implemented above can reveal that this vulnerability is being exploited; present a snippet from the log from the time that the exploit was performed, and explain how the snippet reveals the exploit. (~3 sentences)

Problem 4 : Mitigation (Two-Factor Authentication)

Two-factor authentication is a first line of defense against brute-force and dictionary attacks against a login interface.

Part 1 Design a two-factor authentication protocol that uses e-mail and password. Give the protocol in protocol design syntax. Let U denote the user, C the PayBud client running in the user's browser, and S the PayBud server. Explain (in English) what each step of the protocol is. (~5 sentences)

Part 2 Modify the source code of PayBud such that it uses two-factor authentication. Explain how you modified the source code. (~5 sentences)

Note: PayBud already has the capability of sending an e-mail for forgotten passwords; you may find that bit of source code helpful. You can implement the required functionality by editing `src/.../WebServer.java`, `static/api.js`, and `static/login/{index.html,code.js}`.

Problem 5 : Analysis

Sally: "Wait, they're still getting in? How?!"
Alice: "I have an idea; I just learned about this tool. . . "

Find the vulnerability in the PayBud source code using the *spotbugs* static analysis tool.

Download and install *spotbugs*, following the instructions accompanying the assignment. Navigate to the PayBud directory (where `HOWTO.txt` is), and run the following command.

```
spotbugs -quiet -high -html -output o.html build/libs/a3.jar
```

Inspect the resulting HTML file `o.html`.

Part 1 What vulnerability did *spotbugs* find? In which files/lines? (~2 sentence)

Part 2 Demonstrate an exploit using this vulnerability. (~1 sentence)

Bonus: Show how the logging you implemented above reveals the exploit.

Part 3 Which security requirements does this exploit violate? (~1 sentence)

Part 4 What is *spotbugs*'s proposed fix? Why does that fix the vulnerability? (~2 sentences)

Part 5 Do the proposed fix. Explain (for one instance of the vulnerability) how you modified the code. (~3 sentences)

Problem 6 : Attack, Man In The Middle

Alice: "There; fixed all the *spotbugs* bugs."
Sally: ". . . Did we get all of the vulnerabilities, then?"
Alice: "I hope so. . . maybe?"

Part 1 PayBud is vulnerable to a man-in-the-middle attack. Why? (~1 sentences)

Part 2 Show the attack using protocol design syntax. Let U denote the user, C the PayBud client running in the user's browser, and S the PayBud server. Explain, in the run of the protocol, how the attacker obtains the login credentials. Explain what tool(s) you might use in this attack. (~3 sentences)

Note: For a hint, think back to the start of Assignment 1.

Bonus: Can logging reveal such an exploit? How/Why-not?