

Assignment 3

Due: November 29, 11:59PM EST

Homework 3 is about using JIF to control information flow in Java programs.

If your group is changing, make sure you e-mail Michelle with the new information as soon as possible.

Submission will be via the Blackboard dropbox and must be received by **11:59 PM EST** on **November 29, 2010**. You may resubmit as many times as you like before the deadline; only the last submission will be graded. (NOTE: Make sure you use “send” and not “upload” with the blackboard dropbox, or your homework will not be submitted properly.) Please read and follow the instructions on the following pages carefully to ensure you receive maximum credit for your submission.

Even though we’re only handing out the instructions for part 1 now, the homework will all be turned in together, and we expect one writeup (`gXX_hw3.pdf`) that covers all pieces. Complete details of what to submit and in what format will be included in the further instructions.

1 Resources

- Jif Reference manual (<http://www.cs.cornell.edu/jif/doc/jif-3.3.0/manual.html>)
- Jif warmup exercises (to be posted)
- Jif FAQ (to be posted)

2 The assignment

This assignment is in three parts. All parts are due at the same time.

2.1 Part 1 (20%)

This part only uses Java and not JIF.

In part one you are given code for three Java classes: **ExamRoom**, **Question**, and **Exam**, and for the Java interface **IStudent**. This code is found in `/home/user/part1/`. These classes and interface can be described as follows:

- class **ExamRoom** – This file is the starting point of the application. Function `main()` lives here.
- class **Question** – A data structure representing a question. This class has three fields: the text of the question, the array of options, and the answer.
- class **Exam** – This class has the method `runExam()` that runs the exam, and the method `questionPool()` that returns an array of questions.
- interface **IStudent** – This is an interface for a Student class. It has three methods that class Student has to implement.
 - `getAnswer()` – This method accepts the text of the question and an array of answers.
 - `passResult()` – This method accepts an integer which is the total number of points that student has collected.

- *tellResult()* – In this method, a student prints out the total number that he gets from Examiner via *passResult()*.

The task in this part is to write a malicious class **Student** (implementing **IStudent**) that leaks Alice's answers to Bob, allowing students to cheat during the exam. Alice may follow some simple strategy when choosing the right answer. An example of such a strategy might be just a random guess. In order to compile against the provided class **ExamRoom**, make sure your constructor in the **Student** class accepts a string argument that is a name of the student.

You need to demonstrate the attack by providing the output (as below) that shows that Bob always chooses the same answer as Alice. You must also submit your code. There should be no direct communication between Bob and Alice, or use of static class fields — the attack must exploit a weakness in the Exam class.

```
> java ExamRoom
Starting exam
Alice got question: What cafe in Goteborg offers Kope Luwak coffee?
Options: (0) Blue Mountain Cafe (1) Mauritz Kaffe
::Alice replies 1
Bob got question: What cafe in Goteborg offers Kope Luwak coffee?
Options: (0) Hello, Bob. I think the answer is 1. -Alice (1) Mauritz Kaffe
::Bob replies 1
Alice got question: What's the price of a Kope Luwak espresso?
Options: (0) 100SEK (1) 60SEK
::Alice replies 0
Bob got question: What's the price of a Kope Luwak espresso?
Options: (0) Hello, Bob. I think the answer is 0. -Alice (1) 60SEK
::Bob replies 0
exam finished
Student Alice got 1 points.
Student Bob got 1 points.
done.
```

Explain how Bob obtains the answer from Alice. You must hand in your `Student.java` code from this part of the lab, and your writeup should explain how the attack works.

This section will be graded by script – we will copy your version of `Student.java` into a directory with my versions of `Exam`, `ExamRoom`, `Question` and `IStudent`. Then we will run `java ExamRoom` and match your output with the expected output. Make sure your text matches the example exactly (with the possible exception of which answers Alice chooses) if you want full credit. One quarter of the points for this section will be for a brief but clearly writeup that explains how and why this attack works.

3 Submission and grading

Complete submission instructions will be released with the complete assignment. Late submissions will incur a penalty of 20% per day.