**Assignment 3: The Highs and Lows of Information Flow**
**15-316 Software Foundations of Security and Privacy**

Due: **11:59pm**, Friday 3/9/18
Total Points: 50

1. **Flow types (10 points).**

   Consider the following program.

```
if(a = b) {
  while(a > 0) { a := a - 1 }
} else {
  a := 0
}
```

   To complete this problem, you should use the following symbols to denote various parts of the program.

   $\alpha_1 =$ the entire program $\quad \alpha_2 = \texttt{while}(a > 0)\, a := a - 1 \quad \alpha_3 = a := a - 1 \quad \alpha_4 = a := 0$

   Identify a typing context $\Gamma$ under which this program typechecks using the system discussed in lecture, and show a derivation of $\Gamma \vdash \alpha_1$ using Asgn, Comp, If, and While. To reduce the amount of writing needed to complete the problem, you do not need to show a full derivation for expressions, but if you incorrectly type an expression without showing work you will not receive as much partial credit.

2. **Better reasoning with deduction (15 points).**

   This problem contrasts techniques for verifying non-interference of programs. The first part will use the information flow type system from Lectures 11 and 12, and the second will rely on the deductive approach discussed in Section 4.1 of Lecture 10. We did not discuss the latter extensively in lecture, so you are encouraged to consult that section before attempting Part 2.

   **Part 1 (5 points).** Write a program and give a corresponding type context under which the program satisfies non-interference, but is not well-typed under the type system discussed in lecture. For example, the program from the previous question satisfies non-interference under $\Gamma = (a : \texttt{L}, b : \texttt{H}, \texttt{pc} : \texttt{L})$, but is not well-typed in that context. Find another example, but keep in mind that you want to keep the program as short as possible to reduce the amount of work you need to do in Part 2.

   **Part 2 (10 points).** Use the *self-composition* technique discussed in Lecture 10 and the axioms of dynamic logic to conduct a sequent calculus proof that your program satisfies non-interference under the type context $\Gamma$ you provided in Part 1. That is, create a self-composed program from your answer to Part 1 and show that for any pair of initial states that agree on L-typed variables under your $\Gamma$, the final states will also agree on L-typed variables.

3. **Leveraging interference (10 points).**

   Consider the following program, under the type context $\Gamma = (a : \texttt{H}, b : \texttt{L}, c : \texttt{L})$.

```
if(a < 0) {
  if(b < a) c := 0
  else c := 1
} else {
  if(a < b) c := 0
  else c := 1
}
```

   Describe a procedure that leverages the fact that this program does not satisfy non-interference under $\Gamma$ to learn the value of the H-typed variable. You can make use of the following assumptions.

- Assume that an attacker can control the values of `L`-typed variables prior to executing the program, and observe their value afterwards. They can neither control nor observe `H` variables at any point.
- $-N \leqslant a \leqslant N$ for some $N$ whose value is unknown to the attacker.
- Finally, the attacker can run the program with different `L` inputs any number of times, and the `H` input will remain the same.

How many times does the attacker need to run the program using your procedure to learn $a$?

4. **More unfinished business (15 points).**

Complete the soundness proof of the information flow type system discussed in class for the case of conditional commands. More precisely, if $\Gamma \vdash \texttt{if}(Q)\,\alpha\,\texttt{else}\,\beta$ for type context $\Gamma$, and $\omega_1, \omega_2$ are states such that $\omega_1 \approx_{\Gamma,\mathrm{L}} \omega_2$ and $\omega_1', \omega_2'$ are states such that $\langle \omega_1, \texttt{if}(Q)\,\alpha\,\texttt{else}\,\beta\rangle \Downarrow \omega_1'$ and $\langle \omega_2, \texttt{if}(Q)\,\alpha\,\texttt{else}\,\beta\rangle \Downarrow \omega_2'$, then $\omega_1' \approx_{\Gamma,\mathrm{L}} \omega_2'$.