Computer Security

Prof. Dr.-Ing. Volker Roth
Freie Universität Berlin

June 15, 2012

Please note that your submission will not be accepted without the properly signed Academic Integrity agreement.

Academic Integrity

We each certify that this submission is our own original work and that we have due acknowledged any work of others.

_____

Date, signature, name in block letters

_____

Date, signature, name in block letters

_____

Date, signature, name in block letters

Question 1: Information flow control proofs

Consider the following program:

```
1   {n > 0, pc ≤ PC_m}
2   procedure mod(x, n : int, var y : int)
3   var i : int;
4     i := x ÷ n;
5     y := x − i · n;
6     if y < 0 then
7       y := y + n;
8     fi
9   end
10  {0 ≤ y < n, (y − x) mod n = 0, y ≤ x ⊕ n ⊕ PC_m, pc ≤ PC_m}
```

Prove the precondition implies the postcondition as follows:

- For lines 4–7, give the pre- and postconditions.
- Show that each postcondition Q_i implies the next precondition P_{i+1}.
- Show that the requirements for the `if` statement are met.

Fill in your proof and answers below:

- P_4
- Q_4
- P_5
- Q_5
- $P_6 = \{V_6, L_6\}$
- $\{V_6, e, L_6'\}$
- P_7
- Q_7
- $\{V_7', L_6'\}$
- $P_6 \Rightarrow L_6'[\underline{pc} \leftarrow \underline{pc} \oplus \underline{e}]$
- $Q_8 = \{V_7', L_6\}$

Question 2: Information flow control certification

Consider the following program with flow specifications:

```
1  procedure foo(a : int[] class {__________}, x : int, n : int)
2    var i, j, m : int class {__________};
3    i := 0;
4    j := n;
5    while j > i do
6      m := (j − i)/2;
7      if x > a[m] then
8        i := m + 1;
9      else
10       j := m
11     fi
12   done
13   a[i] := 0
14 end
```

- Which flow specifications must be given in order to make the program secure? Fill in necessary and sufficient specifications into the boxes in the listing above.
- Justify your answer.

Question 3: Setting up working environment and simple example

It's highly recommended that you use some kind of virtualization for the following task!

- Set up a development/build environment with a C compiler and debugger on a 32bit Linux.
- Write a program that implements and invokes a simple function.
- Extend the program so that it dumps the function code to the terminal in hex.
- Extend the program so that it copies the function to memory and executes it there (This involves casting a variable to a function).
- It's pretty certain that you have to turn off certain security features in order to get executable memory. State which mechanisms got in your way and how you turned them off.
- Comment your program and print it out on 1 page. Everything beyond the first page will be ignored. The same holds for answers in ridiculously small font sizes and other nifty layout tricks you may come up with.
- Also remember that your solution will be mainly judged by the comments and explanations you give. Pure code will not earn you any points.