

## CS 6241 : Homework II

Total points : 100

Due : Beginning of class, Monday, February 16<sup>th</sup> 2009

### Important Policies:

1. Homeworks are non-collaborative, please direct questions regarding clarifications to the TA
  2. Although some homework problems may have solutions available in books, internet sources etc. you are supposed to solve them on your own without looking up such solutions. This rule will be STRICTLY enforced and any act of such lookup will be considered an act of plagiarism and will result in the strictest penalty as per Georgia Tech honor code.
  3. When necessary make suitable reasonable assumptions and clearly state them. The solutions should be neatly written/typed using standard pseudo code notations.
1. **Problem I (50 points) [Taint Propagation]** Taint propagation is an important problem encountered in security. A given value  $t_1$  is defined to be tainted by another value  $t_2$  if there exists a dataflow which can lead to generation of value  $t_1$  from  $t_2$  (either directly or transitively). Input values of a program (that are read by a program) are the starting points of attacks and an attacker (through the knowledge of the program), can introduce a malicious input value and taint or corrupt important values generated in the program. It is proposed to solve this problem generating two sets :  $ITaint(t_1)$  is the set of values  $t_1$  can taint where  $t_1$  is the input value to a program,  $TaintedBy(t_2)$  on the other hand gives a set of input values  $t_1$  that can taint a given program value  $t_2$ .
    - (a) Compute  $ITaint(t_1)$  and  $TaintedBy(t_2)$  assuming reaching definition analysis is already performed. (Use the results of reaching definition analysis).
    - (b) It is desired that  $ITaint(t_1)$  and  $TaintedBy(t_2)$  be computed in a standalone manner, viz. without performing reaching definition analysis. Devise a dataflow framework for the same assuming we are computing these on a whole program basis.
    - (c) It is also desired to compute these sets for a specific input value  $v_1$  or for a specific value  $v_2$  to check its vulnerability. Show how will you perform demand-driven analysis for this purpose modifying (b).
    - (d) How does the data flow formulation change in the presence of pointers?
  2. **Problem II [Bounds Checking] [50 points]** Array bounds checking is important to detect the overflow or underflow of array indices. A redundant bounds check exists at a given point if it is already covered by a check that must have executed earlier. ANSI C mandates array bounds

**check be implemented in every compiler that is ANSI C conformant and your boss has come to you to devise as efficient a solution as possible.**

- (a) Show a range of possibilities that could exist to define and detect redundant array bounds checks.**
- (b) Pick one of the possibilities from above and devise a dataflow framework that will efficiently remove redundant array bounds check.**
- (c) Through code motion one could also move or hoist a given check at a given point to cover other checks – show how you will carry out this optimization.