

CS 6V81-002: Quiz 8 Solutions

February 20, 2008

1. CCured enforces which of the following security policies? (circle all that apply)

- (a) **control-flow integrity**
- (b) **memory safety**
- (c) **type safety**
- (d) **buffer-overflow protection**

CCured enforces all of the above security policies. Control-flow integrity is enforced because programs cannot jump through pointers. Memory safety is enforced by the pointer analysis. Section 5.1 of the paper is devoted to proving that CCured enforces type safety. CCured's bounds-checks on sequence and dynamic pointers provides buffer-overflow protection.

2. Which of the following are true of pointers in CCured? (circle all that apply)

- (a) **At runtime, the size of a dynamic pointer is larger than that of a safe pointer.**
- (b) At runtime, the size of a dynamic pointer is larger than that of a sequence pointer.
- (c) **It is possible for a dynamic pointer to point into the middle of an array of values, some of which are integers and some of which are safe pointers.**
- (d) **It is possible for a sequence pointer to point into the middle of an array of values, all of which are dynamic pointers.**

Dynamic pointers and sequence pointers are both represented as the pair $\langle h, n \rangle$, making them equal in size. Dynamic pointers can have pointer arithmetic and type-varying targets, making (c) a correct answer. Sequence pointers can point into an array of dynamic pointers as long as all of the pointers in the array have the same type.

3. Which of the following are true of type inference in CCured? (circle all that apply)

- (a) **It is linear in the size of the program being analyzed.**
- (b) It tries to infer DYNQ types for as many pointers as possible, resorting to SAFE and SEQ types only when it cannot statically prove that a DYNQ type is safe.
- (c) If the C program is unsafe (i.e., there exist security-violating runs), the type inference algorithm always reports a compile-time error.
- (d) If the C program is safe (i.e., there are no security-violating runs), the algorithm always converts it to a CCured program with equivalent behavior.

CCured's type inference algorithm is linear in the size of the code. It tries to infer DYNQ types only as a last resort, making (b) an incorrect answer. Some unsafe C operations are only detected at runtime (e.g., dereferencing an out-of-bounds pointer), making (c) incorrect. The behavior of some safe C programs is changed (e.g., sizeof will return a different result for pointer variables that CCured has changed to dynamic), making (d) incorrect.

4. In CCured a “home” is... (circle one)

- (a) any valid memory address
- (b) a memory address to which at most one pointer points
- (c) the high-order bits of a pointer to a valid memory address
- (d) a unique integer identifying a range of valid memory addresses**

Dynamic and sequence pointers are paired with a unique integer called a “home,” which identifies the range of valid memory addresses to which the pointer may point. If one considers the entire $\langle h, n \rangle$ pair to be the pointer, then the home constitutes the high-order bits of the pointer, so I accepted (c) as a correct answer as well.

5. The CCured paper says, “We treat unions as syntactic sugar for casts.” Describe one disadvantage to this approach as compared with Cyclone’s tagged union approach.

Even if union fields are used consistently by the original program, CCured conservatively assumes that there is casting between their types. It will therefore infer dynamic types for pointer fields that a tagged union structure could have declared to be safe or sequence. This can have a domino effect as other pointers that might alias with the union fields are inferred to be dynamic as well.