

CS 6V81-002: Quiz 10 Solutions

February 27, 2008

1. Which of the following are true of “query methods” in Polymer? (circle all that apply)
 - (a) Polymer’s type-system forces query methods to be “effect-free”.
 - (b) Query methods are invoked by Polymer at compile-time to determine what guard instructions to inject before security-relevant method calls.
 - (c) Query methods can be non-deterministic, e.g. by consulting a random number generator.**
 - (d) A Policy object can have numerous query methods, each of which might respond differently to the same action.

The paper recommends that query methods should be effect-free, but does not enforce that guideline in any way, so (a) is an incorrect answer. Query methods are invoked at runtime, not compile-time, making (b) incorrect. Policy objects each have one query method, making (d) incorrect. (However, policy composition can involve one Policy object consulting the query methods of other Policy objects.)

2. Which of the following kinds of “suggestions” are supported by Polymer? (circle all that apply)
 - (a) A suggestion can choose whether or not to inhibit an action based on the runtime values of global variables.**
 - (b) A suggestion can change the values of global variables.**
 - (c) A suggestion can wait for at least 5 instances of an action to occur and then halt the application just before the 6th occurrence.**
 - (d) A suggestion can replace the return address of the current method with a new address, thereby returning to a different caller.

Suggestions can do just about anything to the virtual machine state, including reading and writing to/from global variables (by using reflection if necessary) and maintaining internal state to count the number of occurrences of security-relevant actions. However, no Java operation can directly modify the return address of the current method. Since Polymer code is Java code, this makes (d) impossible.

3. A “Policy Combinator” in Polymer is... (circle one)
 - (a) an instance of class PolicyCombinator, which extends class Policy
 - (b) an instance of class Policy that consults other Policy objects at runtime**
 - (c) an extension to the Java type-system that allows Policy objects of types τ_1 and τ_2 to be combined to form a new Policy object of type $\tau_1 \times \tau_2$

- (d) an extension to the Java class-loader that allows a single security-relevant method call to invoke multiple query methods, whose return values are then aggregated to generate the return value of the original call

Policy combinators in Polymer are just regular Policy objects that query other Policy objects at runtime. They do not extend the type-system or the class-loader of Java.

- 4. Polymer was used to enforce which security policies on the Pooka email client? (circle all that apply)
 - (a) **Network accesses were constrained to only certain ports.**
 - (b) Filesystem accesses were constrained to only certain directories.
 - (c) **Message-sends were filtered by packet header content.**
 - (d) Communication on the POP port was constrained to obey the POP protocol.

When rewriting Pooka, the Polymer policy restricted access only to certain network ports and did spam filtering based on header content. However, no restriction was placed on filesystem access, and there was no prohibition on sending illegal POP commands to the POP network port.

- 5. Suppose query method q , when passed action a as an argument, always returns an `InsSug` suggestion that says that action a should be inserted. What happens? (circle one)
 - (a) Each action a is first removed and then reinserted, resulting in no visible change.
 - (b) **Each action a is inserted and the existing one is retained, resulting in two action a 's in a row.**
 - (c) Action a is inserted repeatedly, resulting in an infinite loop at compile-time.
 - (d) **Action a is invoked repeatedly, resulting in an infinite loop at runtime (assuming none of the a 's terminates the application).**
 - (e) Polymer rejects the policy as inconsistent.

The true correct answer to this question is (b). If you trace through the operational semantics of Polymer, you'll find that actions inserted by an `InsSug` suggestion are not subject to mediation. However, there was a sentence in the paper that talked about how a Policy combinator could be used to mediate the actions inserted by its sub-policies. If one of these combinators behaved as described in the question, it would result in an infinite loop at runtime so I accepted answer (d) as well.