Practical, Dynamic Information-flow for Virtual Machines

Vivek Haldar, Deepak Chandra, Michael Franz
University of California, Irvine.
{vhaldar, dchandra, franz}@uci.edu

Justification Statement

Current access control mechanisms for Java lack support for mandatory access controls, which are needed when strict information separation is needed, or when sensitive data is handled. They cannot enforce policies that explicitly track data through the virtual machine. Static approaches to controlling information flow do not handle dynamic policies very well, and force a very early binding of code and policy. While operating systems have supported mandatory access controls for a long time, virtual machines currently do not have any support for it.

As a solution, in this paper, we have presented the design and implementation of mandatory access controls in a Java virtual machine. We chose an object to be the basic unit of protection. This is the natural level of abstraction at which a programmer thinks while writing Java code, as well the core concept around which a JVM is built, and seems to be the natural abstraction for reasoning about information flow in a JVM.

The central new paradigm in this paper is the exploration of *adding mandatory access controls to a modern language runtime*. We not only try to explain why this is needed and the tangible benefits of having such a technique, but also how to meaningfully integrate MAC into a language runtime, what the design and implementation issues involved are, and how to specify MAC policies at the level of a language runtime. To the best of our knowledge, this is the first attempt to integrate these two very different domains.

Attendance Statement

All three authors would like to attend the workshop.

<u>Special Note to Reviewers</u>

We have submitted two papers to NSPW – this one and another one titled "Analyzing Java Bytecode for Secure Information Flow – and Safely Transporting It". Both these papers try to tackle the same problem – secure information flow in a Java Virtual Machine – from two very different perspectives. One of our approaches uses static analysis on Java bytecode (the other paper), while this paper describes an approach that is more dynamic in nature, and relies on runtime enforcement in the JVM.

Since these two papers tackle the same problem, there is a large overlap between the "Existing Approaches" and "Related Work" sections of these two papers. This is simply necessary, as we're trying to explain the background and motivate a solution for exactly the same problem in both papers. Otherwise, these are two completely different papers, and explore two very different ways of attacking the same problem.