Analyzing Java Bytecode for Secure Information Flow – and Safely Transporting It

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

Justification Statement

Access control mechanisms in current language runtimes lack support for expressing and enforcing fine-grained information-flow policies. Such policies are needed to enforce the correct handling of sensitive data. As a solution, we present a static analysis on Java bytecode that certifies its information-flow properties. Moreover, to make checking information-flow viable in a JIT-compiled environment, we also present a scheme for annotating Java bytecode with the results of our analysis in way that is easily and quickly verifiable at the target site.

Most established methods for secure information flow work at the source level (e.g. JFlow, Denning's early work). This is simply unrealistic in real-world scenarios where deployment sites rarely have access to source. Moreover, the code consumer has no way to *verify* the information-flow properties of the code he's running, even if a source-level analysis was done at the code producer's end.

Our approach addresses both these problems: firstly, our analysis works on Java bytecode. This means we do not need access to source, and can analyze the large existing body of deployed Java bytecode. It also obviates the need for the code consumer to trust the code producer's claims of information-flow security. Secondly, we are able to annotate bytecode with an easily and quickly (in linear time) *verifiable proof* of the information-flow properties that it satisfies. This makes our analysis viable in a dynamically compiled context where performing static analyses on-line is prohibitively expensive.

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 "Practical, Dynamic Information-flow for Virtual Machines". 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 (this paper), while the other 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.