Lightweight Modeling of the Java Virtual Machine's Security Constraints using Alloy

A CS511 Term Paper Proposal
Mark Reynolds
Feb 1, 2008

The Java programming language has been touted as secure by design. Indeed, a substantial effort has been made by Java's designers to insure that insecure and erroneous program constructions present in weakly typed languages are difficult or impossible to express in Java. Java security is implemented in the Java language itself, in the Java Virtual Machine (JVM) and also in the Java runtime. The proposed focus of the current work will be to model and analyze the security constraints imposed by the JVM, in particular those imposed by the Bytecode Verifier.

The JVM performs a four stage check on all class files that it loads [1]. These phases are (1) verification that the class file format is correct; (2) name resolution (weak linking); the Bytecode Verifier itself; and (4) a synthetic execution phase. During these phases various assertions regarding program correctness are tested. These constraints have direct security implications because, for example, they insure that the preconditions associated with standard exploits such as buffer overflows [2] can never occur. Some constraints that are tested include:

- The operand stack is always the same size and has the same types
- No local variables are accessed before they are set
- Local variable are accessed in a type-safe manner
- Methods are invoked with appropriate arguments
- Opcodes have appropriate arguments on the operand stack

A substantial body of work has been done on formalizing the data flow analysis performed by the Bytecode Verifier [3,4,5], however relatively little work has been done on using model checking to analysis the JVM's Bytecode Verifier [6]. The approach taken in the cited paper used a heavyweight model checking approach in which an abstract state machine (ASM) representation was constructed and then analyzed by a model checker. Only a small subset of the JVM security constraints were analyzed.

The proposed work will use Alloy to built a lightweight model of a portion of these JVM security constraints. Specifically, these constraints can be further decomposed into constraints imposed by the Bytecode Verifier, constraints imposed by the classloader(s), and constraints imposed by the Security Manager. The proposed work will
concentrate on the Bytecode Verifier constraints.

Alloy allows structural constraints to be described very directly, Further, Alloy also makes it possible to express and check properties that fall outside what is currently possible with model checkers, in particular those that involve the temporal evolution of complex state. Alloy should be excellent tool for the proposed work.


