Problem

Just a Matter of Time?

In Jif, the edit/compile/repair cycle is very slow, because the compiler does complex constraint solving to ensure security guarantees. What’s worse is that Jif places greater limitations on programs which causes even more errors. For example, every potential NullPointerException exception must be caught or declared as thrown in order to prevent information leaks through control channels. Just cleaning up the basic errors in a program takes many minutes of a programmer’s time.

Where is that error?

Jif provides very complex error messages that can be hard to locate, let alone interpret. Because information flow violations can be caused by syntactically distant programming constructs, error messages can be very obscure.

How do I do that again?

Security-typed languages use complex programming constructs to ensure safety. This can make programming both tedious and confusing.

Solution: Developer Tools

IBM’s Eclipse framework has been used for building various Integrated Development Environments, notably for Java and C. We are developing a new IDE for secure programming in Jif.

Visualization Tools

1. One of the most challenging parts of Jif programming is visualizing information flows. Color syntax and integrated graphing tools will aid in this.

2. The complexity of Jif annotations can become confusing. The ability to hide and expose annotations simplifies the task of the programmer.

Automation Tools

1. The Jif programmer faces both tedious and challenging tasks. Automation tools assist in both. Assisted annotation of native Java methods is an example of reducing tedious tasks.

2. Refactoring tools will provide automated inference of security labels, derived from an intermediate-level security policy flow graph.

Cryptographic declassifiers

Policy compiler

Distributed Policy

1. The Jif programmer faces both tedious and challenging tasks. Automation tools assist in both. Assisted annotation of native Java methods is an example of reducing tedious tasks.

2. Refactoring tools will provide automated inference of security labels, derived from an intermediate-level security policy flow graph.

Work funded by Motorola and NSF Cybertrust