Over the last few years, a number of information-secure systems have been written using security-typed languages: JPMail (email client), SIF (web application server), FlowWall (firewall).

Each of these applications was developed from scratch; however, there are a lot of existing applications that we would like to certify as information-flow secure.

There are two main challenges with this approach:

- Code is not likely to be information-flow secure "out of the box"
- Information-flow errors can be quite complex, involving conditionals, loops, exceptions, and span across methods.

A common way to verify information-flow security in code is to check the solvability of a constraint system. To better track errors in program code, we expand the constraint solver with a blame dependency graph, which provides errors with a complete and minimal explanation. If an error is detected, we use the explainer to determine what caused that failure.

Each of the errors highlighted by our system is a legitimate information-security security leak, and has a number of possible fixes based on system policy. Based on heuristics, we search for resolution sites that are compatible with system policy.

Groupings of similar errors can be used to solve more than one error at once; the dependency graph allows us to determine if two errors have similar causes, which may suggest a single fix. We can also introduce trusted declassifiers -- declassification statements that can be controlled at the policy level -- in order to make sure the application is not performing any unknown declassification.

Information-flow security is a desirable security property: information-flow secure systems will not leak secret information to public channels when run. Security-typed languages allow programmers to write programs that will be certified as being information-flow secure. However, these languages require programmers to start from scratch and provide a large number of manual annotations. This is not a scalable way to build real systems. We propose a different approach: retrofit programs for security by finding and resolving their information-flow errors using a semi-automated procedure.

Papers:
