New programming language to plug information leaks in software

November 23 2011

The current method for preventing users and unauthorised individuals from obtaining information to which they should not have access in data programs is often to have code reviewers check the code manually, looking for potential weaknesses. Niklas Broberg of the University of Gothenburg has developed a new programming language which automatically identifies potential information leaks while the program is being written.

The most common causes of security issues in today's software are not inadequate network security, poor security protocols or weak encryption mechanisms. In most cases, they are the result of imperfectly written software that contains the potential for information leaks. Users are able to exploit leaks and loopholes that are unintentionally introduced during programming, to obtain more information than they should have access to.

Unauthorised users may also be able to manipulate sensitive information in the system, such as that contained in a database. Currently, the most common method of preventing leaks, loopholes and manipulation is to rely on so-called code reviewers, who "proof-read" the code manually in order to identify errors and deficiencies once the programmers are finished with the code.

Paragon identifies potential information leaks while the program is being written

As a solution to these problems, Niklas Broberg has developed the programming language Paragon. The methodology is presented in his thesis "Practical, Flexible Programming with Information Flow Control" which was written in August 2011.

"The main strength of Paragon is its ability to automatically identify potential information leaks while the program is being developed," says Niklas Broberg. "Paragon is an extension of the commonly-used programming language Java and has been designed to be easy to use. A programmer will easily be able to add my specifications to his or her Java program, thus benefiting from the strong security guarantees that the language provides."

Two-stage security process

Niklas Broberg's method has two stages. The first stage specifies how information in the software may be used, who should be allowed access to it and under what conditions. Stage two of the security process takes place during compilation, where the program's use of information is analysed in depth. If the analysis identifies a risk for sensitive information leaking or being manipulated, the compiler reports an error, enabling the programmer to resolve the issue immediately. The analysis is proven to provide better guarantees than all previous attempts in this field.

"Achieving information security in a system requires a chain of different measures, with the system only being as secure as its weakest link," says Niklas Broberg. "We can have completely effective methods for guaranteeing the authentication of users or encryption of data, but which can be circumvented in practice due to information leaks. Security loopholes in software are currently the most common source of vulnerabilities in our computer systems and it is high time we take these

problems seriously."

Provided by University of Gothenburg