Policy-Directed Code Safety

by

David E. Evans

S.B. Massachusetts Institute of Technology (1994)
S.M. Massachusetts Institute of Technology (1994)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
at the
Massachusetts Institute of Technology

February 2000

Author……
David Evans
Department of Electrical Engineering and Computer Science
October 19, 1999

Certified by……
John V. Guttag
Professor, Computer Science
Thesis Supervisor

Accepted by……
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

Policy-Directed Code Safety

by

David E. Evans

Submitted to the Department of Electrical Engineering and Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Abstract

Executing code can be dangerous. This thesis describes a scheme for protecting the user by constraining the behavior of an executing program. We introduce Naccio, a general architecture for constraining the behavior of program executions. Naccio consists of languages for defining safety policies in a platform-independent way and a system architecture for enforcing those policies on executions by transforming programs. Prototype implementations of Naccio have been built that enforce policies on JavaVM classes and Win32 executables.

Naccio addresses two weaknesses of current code safety systems. One problem is that current systems cannot enforce policies with sufficient precision. For example, a system such as the Java sandbox cannot enforce a policy that limits the rate at which data is sent over the network without denying network use altogether since there are no safety checks associated with sending data. The problem is more fundamental than simply the choices about which safety checks to provide. The system designers were hamstrung into providing only a limited number of checks by a design that incurs the cost of a safety check regardless of whether it matters to the policy in effect. Because Naccio statically analyzes and compiles a policy, it can support safety checks associated with any resource manipulation, yet the costs of a safety check are incurred only when the check is relevant.

Another problem with current code safety systems is that policies are defined in *ad hoc* and platform-specific ways. The author of a safety policy needs to know low-level details about a particular platform and once a safety policy has been developed and tested it cannot easily be transferred to a different platform. Naccio provides a platform-independent way of defining safety policies in terms of abstract resources. Safety policies are described by writing code fragments that account for and constrain resource manipulations. Resources are described using abstract objects with operations that correspond to manipulations of the corresponding system resource. A platform interface provides an operational specification of how system calls affect resources. This enables safety policies to be described in a platform-independent way and isolates most of the complexity of the system.

This thesis motivates and describes the design of Naccio, demonstrates how a large class of safety policies can be defined, and evaluates results from our experience with the prototype implementations.

Thesis Supervisor: John V. Guttag
Title: Professor, Computer Science

Acknowledgements

John Guttag is that rare advisor who has the ability to direct you to see the big picture when you are mired details and to get you to focus when you are distracted by irrelevancies. John has been my mentor throughout my graduate career, and there is no doubt that I wouldn't be finishing this thesis this millennium without his guidance.

As my readers, John Chapin and Daniel Jackson were helpful from the early proposal stages until the final revisions. Both clarified important technical issues, gave me ideas about how to improve the presentation, and provided copious comments on drafts of this thesis.

Andrew Twyman designed and implemented Naccio/Win32. His experience building Naccio/Win32 helped clarify and develop many of the ideas in this thesis, and his insights were a significant contribution to this thesis.

During my time at MIT, I've at the good fortune to work with many interesting and creative people. The MIT Laboratory for Computer Science and the Software Devices and Systems group provided a pleasant and dynamic research environment. Much of what I learned as a grad student was through spontaneous discussions with William Adjie-Winoto, John Ankcorn, Anna Chefter, Dorothy Curtis, Stephen Garland, Angelika Leeb, Ulana Legedza, Li-wei Lehman, Victor Luchangco, Andrew Myers, Anna Pogosyants, Bodhi Priyantha, Hariharan Rahul, Michael Saginaw, Raymie Stata, Yang Meng Tan, Van Van, David Wetherall, and Charles Yang. This work has also benefited from discussions with Úlfar Erlingsson and Fred Schneider from Cornell, Raju Pandey from UC Davis, Dan Wallach from Rice University, Mike Reiter from Lucent Bell Laboratories, and David Bantz from IBM Research.

Geoff Cohen wrote the JOIE toolkit used as Naccio/JavaVM's transformation engine and made its source code available to the research community. He provided quick answers to all my questions about using and modifying JOIE.

Finally, I thank my parents for their constant encouragement and support. I couldn't ask for two better role models.

Table of Contents

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.