"Overview of Language-Based Security" Bibliography
Dan Grossman
13 October 2004

Papers are organized by topic, in the order of the presentation. There are more papers here than discussed in the presentation. Language-based security is a big area and a moving target; please do not interpret this bibliography as more than a rough idea of papers to look at.

Topics:

- Safe C and C-like languages
- Systems in High-Level Languages (a small sample)
- Phantom Types and Security Properties via Types
- Language-Based Alias Control (a small sample)
- Proof-Carrying Code
- Typed Assembly Language
- Compiler Verification
- Inline Reference Monitors, Software-Fault Isolation, Program Shepherding
- Language-Based Information-Flow Security
- Software Model Checking
- Metacompilation and Other Static Bug-Finding Tools
- More Focused Run-Time Tools
- Confined Types
- Stack Inspection
- Type Qualifiers
- Approaches Using Statistics (not discussed at all)

Safe C and C-like languages

Todd Austin, Scott Breach, and Gurindar Sohi. Efficient detection of all pointer and array access errors. In *ACM Conference on Programming Language Design and Implementation*, pages 290–301, Orlando, FL, June 1994.

Richard Jones and Paul Kelly. Backwards-compatible bounds checking for arrays and pointers in C programs. In *AADEBUG'97. Third International Workshop on Automatic Debugging*, volume 2(9) of *Linköping Electronic Articles in Computer and Information Science*, Linköping, Sweden, 1997.

George Necula, Scott McPeak, and Westley Weimer. CCured: Type-safe retrofitting of legacy code. In *29th ACM Symposium on Principles of Programming Languages*, pages 128–139, Portland, OR, January 2002.

Jeremy Condit, Matthew Harren, Scott McPeak, George Necula, and Westley Weimer. CCured in the real world. In *ACM Conference on Programming Language Design and Implementation*, pages 232–244, June 2003.

Trevor Jim, Greg Morrisett, Dan Grossman, Michael Hicks, James Cheney, and Yanling Wang. Cyclone: A safe dialect of C. In *USENIX Annual Technical Conference*, pages 275–288, Monterey, CA, June 2002.

Dan Grossman, Greg Morrisett, Trevor Jim, Michael Hicks, Yanling Wang, and James Cheney. Region-based memory management in Cyclone. In *ACM Conference on Programming Language Design and Implementation*, pages 282–293, Berlin, Germany, June 2002.

Dan Grossman. Type-safe multithreading in Cyclone. In *ACM International Workshop on Types in Language Design and Implementation*, pages 13–25, New Orleans, LA, January 2003.

Michael Hicks, Greg Morrisett, Dan Grossman, and Trevor Jim. Experience with safe manual memory-management in Cyclone. In *International Symposium on Memory Management*, October 2004.

Sumant Kowshik, Dinakar Dhurjati, and Vikram Adve. Ensuring code safety without runtime checks for real-time control systems. In *ACM International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, pages 288–297, Grenoble, France, October 2002.

Systems in High-Level Languages (a small sample)

Brian Bershad, Stefan Savage, Przemyslaw Pardyak, Emin Gün Sirer, Marc Fiuczynski, David Becker, Susan Eggers, and Craig Chambers. Extensibility, safety and performance in the SPIN operating system. In *15th ACM Symposium on Operating System Principles*, pages 267–284, Copper Mountain, CO, December 1995.

Emin Gün Sirer, Stefan Savage, Przemyslaw Pardyak, Greg DeFouw, Mary Ann Alapat, and Brian Bershad. Writing an operating system using Modula-3. In *Workshop on Compiler Support for System Software*, pages 134–140, Tucson, AZ, February 1996.

Wilson Hsieh, Marc Fiuczynski, Charles Garrett, Stefan Savage, David Becker, and Brian Bershad. Language support for extensible operating systems. In *Workshop on Compiler Support for System Software*, pages 127–133, Tucson, AZ, February 1996.

Thorsten von Eicken, Chi-Chao Chang, Grzegorz Czajkowski, Chris Hawblitzel, Deyu Hu, and Dan Spoonhower. J-Kernel: A capability-based operating system for Java. In *Secure Internet Programming, Security Issues for Mobile and Distributed Objects*, volume 1603 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.

Godmar Back, Wilson Hsieh, and Jay Lepreau. Processes in KaffeOS: Isolation, resource management, and sharing in Java. In *4th USENIX Symposium on Operating System Design and Implementation*, pages 333–346, San Diego, CA, October 2000.

Godmar Back, Patrick Tullmann, Leigh Stoller, Wilson Hsieh, and Jay Lepreau. Techniques for the design of Java operating systems. In *USENIX Annual Technical Conference*, pages 197–210, San Diego, CA, June 2000.

Grzegorz Czajkowski and Thorsten von Eicken. JRes: A resource accounting interface for Java. In *ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 21–35, Vancouver, Canada, October 1998.

Chris Hawblitzel and Thorsten von Eicken. Luna: A flexible Java protection system. In *5th USENIX Symposium on Operating System Design and Implementation*, pages 391–403, Boston, MA, December 2002.

Matthew Flatt, Robert Bruce Findler, Shriram Krishnamurthi, and Matthias Felleisen. Programming languages as operating systems (or revenge of the son of the Lisp machine). In *4th ACM International Conference on Functional Programming*, pages 138–147, Paris, France, September 1999.

Phantom Types and Security Properties via Types

Matthew Fluet and Riccardo Pucella. Phantom types and subtyping. In *2nd IFIP International Conference on Theoretical Computer Science*, pages 448–460, Montreal, Canada, August 2002.

James Cheney and Ralf Hinze. First-class phantom types. Technical Report CUCIS TR2003-1901, Department of Computer Science, Cornell University, 2003.

David Walker. A type system for expressive security policies. In *27th ACM Symposium on Principles of Programming Languages*, pages 254–267, January 2000.

Language-Based Alias Control (a small sample)

Jonathan Aldrich, Valentin Kostadinov, and Craig Chambers. Alias annotations for program understanding. In *ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications*, pages 311–330, Seattle, WA, November 2002.

Robert DeLine and Manuel Fähndrich. Enforcing high-level protocols in low-level software. In *ACM Conference on Programming Language Design and Implementation*, pages 59–69, Snowbird, UT, June 2001.

Dave Clarke and Tobias Wrigstad. External uniqueness is unique enough. In *European Conference on Object-Oriented Programming*, pages 176–200, Darmstadt, Germany, July 2003.

Alex Aiken, Jeffrey Foster, John Kodumal, and Tachio Terauchi. Checking and inferring local non-aliasing. In *ACM Conference on Programming Language Design and Implementation*, pages 129–140, San Diego, CA, June 2003.

Philip Wadler. Linear types can change the world! In M. Broy and C. Jones, editors, *Programming Concepts and Methods*, Sea of Galilee, Israel, April 1990. North Holland. IFIP TC 2 Working Conference.

Overview of Language-Based Security

Fred B. Schneider, Greg Morrisett, and Robert Harper. A language-based approach to security. In *Informatics: 10 Years Back, 10 Years Ahead*, volume 2000 of *Lecture Notes in Computer Science*, pages 86–101. Springer-Verlag, 2001.

Dexter Kozen. Language-based security. In *Mathematical Foundations of Computer Science*, volume 1672 of *Lecture Notes in Computer Science*, pages 284–298. Springer-Verlag, September 1999.

Proof-Carrying Code

George Necula. Proof-carrying code. In *24th ACM Symposium on Principles of Programming Languages*, pages 106–119, Paris, France, January 1997.

George Necula and Peter Lee. The design and implementation of a certifying compiler. In *ACM Conference on Programming Language Design and Implementation*, pages 333–344, Montreal, Canada, June 1998.

Christopher Colby, Peter Lee, George Necula, and Fred Blau. A certifying compiler for Java. In *ACM Conference on Programming Language Design and Implementation*, pages 95–107, Vancouver, Canada, June 2000.

George Necula and Peter Lee. Efficient representation and validation of proofs. In *13th IEEE Symposium on Logic in Computer Science*, pages 93–104, Indianapolis, IN, June 1998.

George Necula and Shree Rahul. Oracle-based checking of untrusted software. In *28th ACM Symposium on Principles of Programming Languages*, pages 142–154, London, England, January 2001.

Andrew Appel. Foundational proof-carrying code. In *16th IEEE Symposium on Logic in Computer Science*, pages 247–258, Boston, MA, June 2001.

Nadeem Hamid, Zhong Shao, Valery Trifonov, Stefan Monnier, and Zhaozhong Ni. A syntactic approach to foundational proof-carrying code. In *17th IEEE Symposium on Logic in Computer Science*, pages 89–100, Copenhagen, Denmark, July 2002.

Typed Assembly Language

Greg Morrisett, David Walker, Karl Crary, and Neal Glew. From System F to typed assembly language. *ACM Transactions on Programming Languages and Systems*, 21(3):528–569, May 1999.

Greg Morrisett, Karl Crary, Neal Glew, and David Walker. Stack-based typed assembly language. *Journal of Functional Programming*, 12(1):43–88, January 2002.

Neal Glew and Greg Morrisett. Type safe linking and modular assembly language. In *26th ACM Symposium on Principles of Programming Languages*, pages 250–261, San Antonio, TX, January 1999.

Greg Morrisett, Karl Crary, Neal Glew, Dan Grossman, Richard Samuels, Frederick Smith, David Walker, Stephanie Weirich, and Steve Zdancewic. TALx86: A realistic typed assembly language. In *2nd ACM Workshop on Compiler Support for System Software*, pages 25–35, Atlanta, GA, May 1999. Published as INRIA Technical Report 0288, March, 1999.

Dan Grossman and Greg Morrisett. Scalable certification for typed assembly language. In *Workshop on Types in Compilation*, volume 2071 of *Lecture Notes in Computer Science*, pages 117–145, Montreal, Canada, September 2000. Springer-Verlag.

Hongwei Xi and Robert Harper. A dependently typed assembly language. In *6th ACM International Conference on Functional Programming*, pages 169–180, Florence, Italy, September 2001.

Karl Crary. Toward a foundational typed assembly language. In *30th ACM Symposium on Principles of Programming Languages*, pages 198–212, New Orleans, LA, January 2003.

Compiler Verification (thanks to Sorin Lerner)

Sorin Lerner, Todd Millstein, and Craig Chambers. Automatically proving the correctness of compiler optimizations. In *ACM Conference on Programming Language Design and Implementation*, pages 220–231, San Diego, CA, June 2003.

Sorin Lerner, Todd Millstein, Erika Rice, and Craig Chambers. Automated soundness proofs via local rules. In *32nd ACM Symposium on Principles of Programming Languages*, Long Beach, CA, January 2005.

A. Pnueli, M. Siegel, and E. Singerman. Translation validation. In *Tools and Algorithms for Construction and Analysis of Systems*, volume 1384 of *Lecture Notes in Computer Science*, pages 151–166. Springer-Verlag, 1998.

George C. Necula. Translation validation for an optimizing compiler. In *ACM Conference on Programming Language Design and Implementation*, pages 83–95, Vancouver, Canada, June 2000.

Martin Rinard and Darko Marinov. Credible compilation. In *FLoC Workshop Run-Time Result Verification*, July 1999.

J. Guttman, J. Ramsdell, and M. Wand. VLISP: a verified implementation of Scheme. *Lisp and Symbolic Compucation*, 8(1–2):33–110, 1995.

D. P. Oliva, J. Ramsdell, and M. Wand. The VLISP verified PreScheme compiler. *Lisp and Symbolic Computation*, 8(1–2):111–182, 1995.

Inline Reference Monitors, Software-Fault Isolation, Program Shepherding

Fred B. Schneider. Enforceable security policies. *ACM Transactions on Information and System Security*, 3(1):30–50, February 2000.

Ulfar Erlingsson and Fred B. Schneider. IRM enforcement of Java stack inspectio. In *IEEE Symposium on Security and Privacy*, pages 246–255, Oakland, CA, May 2000.

Ulfar Erlingsson and Fred B. Schneider. SASI enforcement of security policies: A retrospective. In *New Security Paradigms Workshop*, pages 87–95, Ontario, Canada, September 1999.

Ulfar Erlingsson. *The Inlined Reference Monitor Approach to Security Policy Enforcement*. PhD thesis, Cornell University, 2003.

Kevin W. Hamlen, Greg Morrisett, and Fred B. Schneider. Computability classes for enforcement mechanisms. Technical Report CUCIS TR2003-1908, Department of Computer Science, Cornell University, 2003. A version will appear in TOPLAS. DO ME

Lujo Bauer, Jarred Ligatti, and David Walker. More enforceable security policies. In *Workshop on Computer Security Foundations*, Copenhagen, Denmark, July 2002.

Robert Wahbe, Steven Lucco, Thomas Anderson, and Susan Graham. Efficient software-based fault isolation. *ACM SIGOPS Operating Systems Review*, 7(5):203–216, December 1993.

Christopher Small and Margo Seltzer. MiSFIT: constructing safe extensible systems. *IEEE Concurrency*, 6(3):33–41, July–September 1998.

Vladimir Kiriansky, Derek Bruening, and Saman Amarasinghe. Secure execution via program shepherding. In *11th USENIX Security Symposium*, pages 191–206, San Francisco, CA, August 2002.

Language-Based Information-Flow Security

Andrei Sabelfeld and Andrew C. Myers. Language-based information-flow security. *IEEE Journal on Selected Areas in Communications, special issue on Formal Methods for Security*, 21(1):5–19, January 2003.

This is a 15-page survey paper with 147 references available at `www.cs.chalmers.se/~andrei/jsac.pdf` and `www.cs.cornell.edu/andru/papers/jsac/sm-jsac03.pdf`; there is little point in repeating additional citations here.

Software Model Checking

Patrice Godefroid. Model checking for programming languages using VeriSoft. In *24th ACM Symposium on Principles of Programming Languages*, pages 174–186, Paris, France, January 1997.

Thomas Ball and Sriram Rajamani. The SLAM project: Debugging system software via static analysis. In *29th ACM Symposium on Principles of Programming Languages*, pages 1–3, Portland, OR, January 2002.

Thomas Ball and Sriram Rajamani. Automatically validating temporal safety properties of interfaces. In *8th International SPIN Workshop*, volume 2057 of *Lecture Notes in Computer Science*, pages 103–122, Toronto, Canada, May 2001. Springer-Verlag.

Thomas Henzinger, Ranjit Jhala, Rupak Majumdar, and Grégoire Sutre. Lazy abstraction. In *29th ACM Symposium on Principles of Programming Languages*, pages 58–70, Portland, OR, January 2002.

Thomas Henzinger, Ranjit Jhala, Rupak Majumdar, George Necula, Grégoire Sutre, and Westley Weimer. Temporal-safety proofs for systems code. In *14th International Conference on Computer Aided Verification*, volume 2404 of *Lecture Notes in Computer Science*, pages 526–538, Copenhagen, Denmark, July 2002. Springer-Verlag.

James Corbett, Matthew Dwyer, John Hatcliff, Corina Pasareanu, Robby, Shawn Laubach, and Hongjun Zheng. Bandera: Extracting finite-state models from Java source code. In *22nd International Conference on Software Engineering*, June 2000.

John Hatcliff and Matthew Dwyer. Using the Bandera tool set to model-check properties of concurrent Java software. In *CONCUR 2001*, June 2001.

Gerard Holzmann. Logic verification of ANSI-C code with SPIN. In *7th International SPIN Workshop*, volume 1885 of *Lecture Notes in Computer Science*, pages 131–147, Stanford, CA, August 2000. Springer-Verlag.

Gerard Holzmann. Static source code checking for user-defined properties. In *World Conference on Integrated Design and Process Technology*, Pasadena, CA, June 2002. Society for Design and Process Science.

Metacompilation and Other Static Bug-Finding Tools

Stephen Johnson. Lint, a C program checker. Computer Science Technical Report 65, Bell Laboratories, December 1977.

David Evans. Static detection of dynamic memory errors. In *ACM Conference on Programming Language Design and Implementation*, pages 44–53, Philadelphia, PA, May 1996.

David Evans, John Guttag, Jim Horning, and Yang Meng Tan. LCLint: A tool for using specifications to check code. In *2nd ACM Symposium on the Foundations of Software Engineering*, pages 87–96, New Orleans, LA, December 1994.

David Evans and David Larochelle. Improving security using extensible lightweight static analysis. *IEEE Software*, 19(1):42–51, January 2002.

Splint manual, version 3.0.6, 2002. `http://www.splint.org/manual/`.

William Bush, Jonathan Pincus, and David Sielaff. A static analyzer for finding dynamic programming errors. *Software Practice and Experience*, 30(7):775–802, June 2000.

Dawson Engler, Benjamin Chelf, Andy Chou, and Seth Hallem. Checking system rules using system-specific, programmer-written compiler extensions. In *4th USENIX Symposium on Operating System Design and Implementation*, pages 1–16, San Diego, CA, October 2000.

Benjamin Chelf, Dawson Engler, and Seth Hallem. How to write system-specific, static checkers in Metal. In *ACM Workshop on Program Analysis for Software Tools and Engineering*, pages 51–60, Charleston, SC, November 2002.

Seth Hallem, Benjamin Chelf, Yichen Xie, and Dawson Engler. A system and language for building system-specific static analyses. In *ACM Conference on Programming Language Design and Implementation*, pages 69–82, Berlin, Germany, June 2002.

More Focused Run-Time Tools

Reed Hastings and Bob Joyce. Purify: Fast detection of memory leaks and access errors. In *Winter USENIX Conference*, pages 125–138, San Francisco, CA, January 1992.

Bruce Perens. Electric fence, 1999.
`http://www.gnu.org/directory/All_Packages_in_Directory/ElectricFence.html`.

Crispin Cowan, Calton Pu, Dave Maier, Heather Hinton, Jonathan Walpole, Peat Bakke, Steve Beattie, Aaron Grier, Perry Wagle, and Qian Zhang. StackGuard: Automatic adaptive detection and prevention of buffer-overflow attacks. In *7th USENIX Security Symposium*, pages 63–78, San Antonio, TX, January 1998.

Alexey Loginov, Suan Hsi Yong, Susan Horwitz, and Thomas Reps. Debugging via run-time type checking. In *4th International Conference on Fundamental Approaches to Software Engineering*, volume 2029 of *Lecture Notes in Computer Science*, pages 217–232, Genoa, Italy, April 2001. Springer-Verlag.

Confined Types

Boris Bokowski and Jan Vitek. Confined types for Java. *Software Practice and Experience*, 31(6), 2001.

Christian Grothoff, Jens Palsberg, and Jan Vitek. Encapsulating objects with confined types. In *ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications*, Tampa Bay, FL, October 2001.

Tian Zhao, Jens Palsberg, and Jan Vitek. Lightweight confinment for Featherweight Java. In *ACM Conference on Object-Oriented Programming, Systems, Languages, and Applications*, Anaheim, CA, October 2003.

Stack Inspection

Dan S. Wallach, Dirk Balfanz, Drew Dean, and Edward W. Felten. Extensible security architectures for Java. In *16th ACM Symposium on Operating System Principles*, pages 116–128, Saint-Malo, France, October 1997.

Dan S. Wallach and Edward W. Felten. Understanding Java stack inspection. In *IEEE Symposium on Security and Privacy*, pages 52–63, Oakland, CA, May 1998.

Dan S. Wallach, Edward W. Felten, and Andrew W. Appel. The security architecture formerly known as stack inspection: A security mechanism for language-based systems. *ACM Transactions on Software Engineering and Methodology*, 9(4), October 2000.

Cedric Fournet and Andrew D. Gordon. Stack inspection: Theory and variants. *ACM Transactions on Programming Languages and Systems*, 25(3):360–399, 2003.

Type Qualifiers

Jeffrey Foster, Manuel Fähndrich, and Alexander Aiken. A theory of type qualifiers. In *ACM Conference on Programming Language Design and Implementation*, pages 192–203, Atlanta, GA, May 1999.

Jeffrey Foster, Tachio Terauchi, and Alex Aiken. Flow-sensitive type qualifiers. In *ACM Conference on Programming Language Design and Implementation*, pages 1–12, Berlin, Germany, June 2002.

Umesh Shankar, Kunal Talwar, Jeffrey Foster, and David Wagner. Detecting format string vulnerabilities with type qualifiers. In *10th USENIX Security Symposium*, page ???, Washington, D.C., August 2001.

Xiaolan Zhang, Antony Edwards, and Trent Jaeger. Using CQUAL for static analysis of authorization hook placement. In *11th USENIX Security Symposium*, pages 33–48, San Francisco, CA, August 2002.

Alex Aiken, Jeffrey Foster, John Kodumal, and Tachio Terauchi. Checking and inferring local non-aliasing. In *ACM Conference on Programming Language Design and Implementation*, pages 129–140, San Diego, CA, June 2003.

Approaches Using Statistics (not really discussed)

William Bush, Jonathan Pincus, and David Sielaff. A static analyzer for finding dynamic programming errors. *Software Practice and Experience*, 30(7):775–802, June 2000.

Glenn Ammons, Rastislav Bodik, and James Larus. Mining specifications. In *29th ACM Symposium on Principles of Programming Languages*, pages 4–16, Portland, OR, January 2002.

Dawson Engler, David Chen, Seth Hallem, Andy Chou, and Benjamin Chelf. Bugs as deviant behavior: A general approach to inferring errors in systems code. In *18th ACM Symposium on Operating System Principles*, pages 57–72, Banff, Canada, October 2001.

Ben Liblit, Alex Aiken, Alice X. Zheng, and Michael I. Jordan. Bug isolation via remote program sampling. In *ACM Conference on Programming Language Design and Implementation*, pages 141–154, San Diego, CA, June 2003.