Junfeng Yang

Computer Science Department
Columbia University
New York, New York

Phone: (650) 387-8388
junfeng@cs.columbia.edu
http://www.cs.columbia.edu/~junfeng

Research Interests

Operating systems, software error detection, security, and storage systems.

Education

Ph.D. in Computer Science, expected 2008	STANFORD UNIVERSITY

Dissertation: Automatically Finding Serious Storage System Errors
Advisor: Dawson Engler

M.S. in Computer Science, 2002	STANFORD UNIVERSITY

B.S. in Computer Science, 2000	TSINGHUA UNIVERSITY, CHINA

Entrance exam waived based on outstanding high-school record

Special academic program, 1995-1997	TSINGHUA UNIVERSITY, CHINA

Only 60 out of about 2000 freshmen selected

Honors and Awards

2004	Best Paper Award of the Sixth USENIX Symposium on Operating Systems Design and Implementation (OSDI)
2003	Stanford School of Engineering Fellowship
2002	Siebel Scholarship
1995–1997	Tsinghua Outstanding Scholarship for Undergraduates

Research Experience

Research Assistant — Stanford University

2004–2006 — **Automatically finding serious storage system errors using model checking** Developed a new approach to check storage systems. Led the design, implementation, and evaluation of two checking frameworks, FiSC [1, 2] and EXPLODE [3], which formed my thesis work. Adapted ideas from model checking, a formal verification technique, to find unrecoverable data-loss errors (e.g. file system root corruptions).

I designed and built FiSC to thoroughly check file systems. Despite the wildly-different, poorly-specified guarantees provided by these file systems, I invented a set of powerful checks that require little FS-specific knowledge to apply. For example, one of them checked the atomicity guarantees provided by write-ahead-logging and found eight data-loss errors. FiSC found over 33 errors in four widely-used, well-tested Linux file systems, 11 of which can cause data loss, 21 of which were serious enough to lead to immediate kernel patches within two days. The paper from this work won Best Paper Award of the Sixth USENIX Symposium on Operating Systems Design and Implementation (OSDI'04).

I designed and built EXPLODE to generalize the above approach to other storage systems, and to make checking lightweight. Instead of porting a system to run inside a model checker, I invented an *in-situ* checking architecture that interlaces the mechanisms

needed for comprehensive checking with the checked system. This architecture drastically reduced the checking overhead by several orders of magnitude. It also reduced the invasive infrastructure needed (down to a single device driver in EXPLODE). I created a C++ checking interface in which developers simply write a few hundred lines of code to check their storage system. Along with my colleagues, I checked a wide range of seventeen storage systems, including three version control software, a database, NFS, ten file systems, a software RAID and a popular commercial virtual machine. We found errors in every system checked, 36 errors in total, most of which can cause serious data loss.

2005–2006   **Automatically generating malicious disks using symbolic execution.** With Can Sar, Paul Twohey, Cristian Cadar and my advisor Dawson Engler, co-proposed a threat model in which malicious disks crash or exploit an OS kernel through the file system mount interface. We used EXE, a testcase generator using symbolic executions, to discover nine malicious disk images [4]. I identified several drawbacks of EXE, and proposed two optimizations to make the work possible. This application greatly improved EXE and remains as its most significant benchmark.

2003   **Extensible, expressive and lightweight annotation language and system.** Led the design, implementation, and evaluation of MECA, an extensible, expressive system and language for statically finding security errors [5]. Extended C grammar and modified the GNU C Compiler to allow programmers to declare and use domain-specific annotations as inlined comments in their code. Invented several practical constructs to effectively annotate large bodies of code and suppress false positives. To reduce annotation overhead, I used aggressive compiler analysis and statistical analysis to infer missing annotations. The most thorough case study of MECA, a checker that detects invalid use of user-provided pointers in OS kernel, used only 75 manual annotations to automatically derive more than ten thousand missing ones and check millions of lines of code in the Linux kernel. It found 44 security holes with only 8 false positives.

2004   **Correlation Exploitation in Error Ranking.** With Ted Kremenek, Ken Ashcraft, and Dawson Engler, co-proposed an online ranking algorithm for error report inspection [6]. It uses feedbacks from users to dynamically rank likely errors over false positives, by exploring correlations in error reports.

2001   **An empirical study of OS errors.** With Andy Chou, Dawson Engler and others, collaboratively studied error patterns in kernel code using roughly 1000 unique, automatically-detected errors on 11 Linux kernel releases over 7 years [7]. One interesting pattern was that Linux device drivers are up to ten times buggier than other kernel code.

Summer 2005   **Summer Intern**   Microsoft Research, Silicon Valley
Mentor: Lidong Zhou, John MacCormick

                Load balancing in distributed systems. Surveyed theory results on load balancing (the classic balls-into-bins problem). Based on the study, I co-designed and evaluated a distributed load balancing scheme that used multiple hash functions to achieve both good balance and high availability, in the presence of incremental system expansions, server failures, and load changes. This scheme has been extended and deployed in a distributed data store at Microsoft Research.

Teaching and Supervisory Experience

2003–now   With my advisor, I co-supervised three research projects: EXPLODE, FiSC, and MECA.

2006   Teaching Assistant, Stanford course CS240, Advanced Topics in Operating Systems

2005–2006   Informally advised a Bachelor Honors Thesis

2005   Teaching Assistant, Stanford course CS240, Advanced Topics in Operating Systems

Refereed Publications

[1] Junfeng Yang, Paul Twohey, Dawson Engler, and Madanlal Musuvathi. Using model checking to find serious file system errors. In *Proceedings of the Sixth Symposium on Operating Systems Design and Implementation (OSDI)*, San Francisco, CA, December 2004.

[2] Junfeng Yang, Paul Twohey, Dawson Engler, and Madanlal Musuvathi. Using model checking to find serious file system errors. *ACM Transactions on Computer Systems (TOCS)*, forwarded from OSDI'04, to appear.

[3] Junfeng Yang, Can Sar, and Dawson Engler. EXPLODE: a lightweight, general system for finding serious storage system errors. In *Proceedings of the Seventh Symposium on Operating Systems Design and Implementation (OSDI)*, Seattle, CA, November 2006.

[4] Junfeng Yang, Can Sar, Paul Twohey, Cristian Cadar, and Dawson Engler. Automatically generating malicious disks using symbolic execution. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2006.

[5] Junfeng Yang, Ted Kremenek, Yichen Xie, and Dawson Engler. MECA: an extensible, expressive system and language for statically checking security properties. In *10th ACM Conference on Computer and Communications Security (ACM CCS)*, Washington, DC, October 2003.

[6] Ted Kremenek, Ken Ashcraft, Junfeng Yang, and Dawson Engler. Correlation exploitation in error ranking. In *Proceedings of the 12th ACM SIGSOFT twelfth international symposium on Foundations of software engineering (FSE)*, Newport Beach, CA, November 2004.

[7] Andy Chou, Junfeng Yang, Benjamin Chelf, Seth Hallem, and Dawson Engler. An empirical study of operating systems errors. In *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles (SOSP)*, Banff, Canada, October 2001.

Conference Talks

"EXPLODE: a Lightweight, General System for finding Serious Storage System Errors", *Seventh USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, Seattle, Washington, November 2006.

"Automatically Generating Malicious Disks using Symbolic Execution", *IEEE Symposium on Security and Privacy (Oakland Security)*, Oakland, California, May 2006.

"Using Model Checking to Find Serious File System Errors", *Sixth USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, San Francisco, California, December, 2004.

"MECA: an Extensible, Expressive System and Language for Statically Checking Security Properties", *Tenth ACM conference on Computer and communication security (ACM CCS)*,Washington, DC, October 2003.

Invited Talks

"EXPLODE: a Lightweight, General System for finding Serious Storage System Errors", Microsoft Research, Mountain View, CA, October 2006

"Automatically Finding Serious Storage System Errors", Coverity, San Francisco, CA, October 2006

"Graceful Degradation", Microsoft Research, Mountain View, CA, September 2005

"Automatically Finding Serious File System Errors", Stanford Computer Forum Security Workshop, Stanford, CA, March 2005