

TIMOTHY K. TSAI

Email: timothytsai@computer.org
Citizenship: USA

Web: http://www.tsai-family.com/tim_work.htm
(including contact information)

EDUCATION

Ph.D. Electrical Engineering, University of Illinois at Urbana-Champaign, 1996.
MS Electrical Engineering, University of Illinois at Urbana-Champaign, 1993.
BS Electrical Engineering, Brigham Young University, 1990.

SKILLS

Operating systems:
Platforms: Microsoft Windows, Linux, Solaris, other UNIX systems
Topics: Drivers, libraries, memory management, checkpointing, process monitoring/migration
Programming:
Languages: C/C++, Java, Perl, shell scripts, assembly, HTML, JavaScript, SQL
Environments: Debuggers, profilers, make tools, source code control, vim, UNIX utilities, databases
Fields of experience: Reliability, security, hardware and software architecture, software testing, telephony systems, data mining

PROFESSIONAL EXPERIENCE

Hitachi Global Storage Technologies (San Jose, CA), 2007 – Present

Research Staff Member

- Led research effort into statistical analysis of hard disk drive reliability using data mining on manufacturing and field data. This project attempts to leverage operation data from a large population of enterprise disk drives to achieve failure prediction, reliability model validation, testing validation, and other engineering insights.

Sun Microsystems, Inc. (Santa Clara, CA), 2004 - 2007

HPCS RAS team

- Responsible for modeling, design, risk reduction, and coordination with software teams for the software RAS architecture of Sun's DARPA Phase II HPCS (High Productivity Computing Systems) project, including modeling and analysis of large-scale hardware and software failure modes, automated checkpointing for supercomputing, data integrity for capacitive coupling and optical networks, and fault management and prediction systems.

Staff Engineer, SunCARE

- Developed SAS and Java tools for TDR (Time-Dependent Reliability) analysis of field failures, including integration with customer systems management databases, deployment of TDR tools to service and account teams, and research into expert systems to guide statistical reliability analysis.
- Conducted experiments to evaluate robustness of the Solaris operating system, including the use of fault injection.

Avaya Labs, Avaya, Inc. (Basking Ridge, NJ), 2000 - 2004

Member of Technical Staff, Avaya Labs

- Set up lab to perform fault injection testing of PBX and IP phones for security vulnerabilities.
- Performed research into security intrusion detection and cryptographic solutions for IP telephony.
- Continued development of Libsafe security library, including detection of format string attacks, porting to Windows NT, and improvements in robustness and performance. Oversaw integration of Libsafe into Avaya telephony products, including the Linux-based PBX and voice messaging server.
- Developed SVI security patch management system to generate automatic notifications of security vulnerabilities affecting Avaya product platforms.
- Led work on BOVScan tool to scan Linux executables and libraries for buffer overflow vulnerabilities.

Cigital, Inc., previously known as Reliable Software Technologies (Dulles, VA), 2000

Senior Research Scientist, Cigital Labs

- Developed fault injection testbed for Java component reverification project.

Lucent Technologies (Murray Hill, NJ), 1996 - 2000

Member of Technical Staff, Bell Laboratories

- Developed the Libsafe security library to detect buffer overflow security attacks for Linux systems. Libsafe was originally included in several Linux distributions. Also developed the Libverify buffer overflow tool, which extends the detection coverage for buffer overflow attacks.
- Member of the software-implemented fault tolerance (SwiFT) team, which performed pioneering work in software-based fault tolerance technology including automatic process restart and failover, checkpointing, and reliable file access. The SwiFT work has been introduced into many Lucent telephony products and has also been licensed to other companies including Tandem. Participated in porting the software to Linux and Windows NT and fault-injection based testing and studies. Also, researched techniques for software-implemented N-modular redundant systems and for monitoring of network-based services.
- Developed DTS fault injection tool for Windows NT and Linux. In contrast to much of the previous fault injection work, the DTS tool focused on testing of realistic commercial server systems with the goal of making progress towards a dependability benchmark. Experimental studies with the DTS tool revealed insights in comparing competing applications, as well as the same application on different platforms.

Tandem Computers (Austin, TX), 1994

Intern, ServerNet Architecture Group

- Developed testbed to perform hardware fault injection verification of low-level Tandem ServerNet protocols.

Tandem Computers (Austin, TX), 1993

Intern, System Validation Group

- Performed verification and validation of Tandem Integrity operating systems.

REFERREED PUBLICATIONS

Timothy Tsai, Kalyan Vaidyanathan, and Kenny Gross, "Low-Overhead Run-Time Memory Leak Detection and Recovery," in *Proceedings of the 12th IEEE International Symposium on Pacific Rim Dependable Computing (PRDC-2006)*, December 18-20, 2006, Riverside, CA, USA, pp. 329-337.

Sachin Garg, Navjot Singh, and Timothy Tsai, "Short Paper: Schemes for Enhancing the Denial-of-Service Tolerance of SRTP," in *Proceedings of the IEEE Conference on Security and Privacy for Emerging Areas in Communications Networks (SecureComm)*, September 5-9, 2005, Athens, Greece, pp. 409-411.

Alan Wood, Swami Nathan, Tim Tsai, Chris Vick, Larry Votta, and Anoop Vetteth, "Multi-Tier Checkpointing for Peta-Scale Systems," in *Supplemental Proceedings of the International Conference on Dependable Systems and Networks (DSN-2005)*, June 28-July 1, 2005, Yokohama, Japan, pp. 112-121.

Saurabh Bagchi, Yu-Sung Wu, Sachin Garg, Navjot Singh, and Timothy Tsai, "SCIDIVE: A Stateful and Cross Protocol Intrusion Detection Architecture for Voice-over-IP Environments," in *Proceedings of the International Conference on Dependable Systems and Networks (DSN-2004)*, June 28-July 1, 2004, Florence, Italy, pp. 433-442.

Deron Liang, P. Emerald Chung, Yennun Huang, Chandra Kintala, Woei-Jyh Lee, Timothy K. Tsai, and Chung-Yih Wang, "NT-SwiFT: Software Implemented Fault Tolerance on Windows NT," *Journal of Systems and Software*, vol. 21, Issues 1-2, April 2004, pp. 127-141.

Timothy K. Tsai and Navjot Singh, "Reliability Testing of Applications on Windows NT," in *Proceedings of the International Conference on Dependable Systems and Networks (DSN-2000)*, June 25-28, 2000, New York City, NY, USA, pp. 427-436.

Arash Baratloo, Timothy K. Tsai, and Navjot Singh, "Transparent Run-Time Detection of Stack Smashing Attacks," in *Proceedings of USENIX 2000 Technical Conference*, June 18-23, 2000, San Diego, CA, USA, pp. 251-262.

Timothy K. Tsai and Navjot Singh, "How Reliable Is Your NT Application?: Using Fault Injection to Test Critical Applications," *Windows 2000 Magazine*, web exclusive, February 2000, <http://www.windowsitpro.com/Article/ArticleID/7708/7708.html>.

Timothy K. Tsai, Mei-Chen Hsueh, Hong Zhao, Zbigniew Kalbarczyk, and Ravishankar K. Iyer, "Stress-based and Path-based Fault Injection," *IEEE Transactions on Computers*, vol. 48, no. 11, November 1999, pp. 1183-1201.

Timothy Tsai, "Fault Tolerance Via N-Modular Software Redundancy," in *Proceedings of the 28th Symposium on Fault-Tolerant Computing (FTCS-28)*, Munich, Germany, June 23-25, 1998, pp. 201-206.

- Timothy K. Tsai, Shambhu Upadhyaya, Hong Zhao, Mei-Chen Hsueh, and Ravishankar K. Iyer, "Path-based Fault Injection," in *Proceedings of the Third ISSAT International Conference on Reliability and Quality in Design*, Anaheim, CA, USA, March 12-14, 1997, pp. 121-125.
- Mei-Chen Hsueh, Timothy K. Tsai, and Ravishankar K. Iyer, "Fault Injection Techniques and Tools," *IEEE Computer*, vol. 30, no. 4, April 1997, pp. 75-82.
- Timothy K. Tsai and Ravishankar K. Iyer, "An Approach Towards Benchmarking of Fault-Tolerant Commercial Systems," in *Proceedings of the 26th Symposium on Fault-Tolerant Computing (FTCS-26)*, Sendai, Japan, June 25-27, 1996, pp. 314-325.
- Timothy K. Tsai and Ravishankar K. Iyer, "Measuring Fault Tolerance with the FTAPE Fault Injection Tool," *Lecture Notes in Computer Science*, vol. 977, 1995, pp. 26-40.
- Timothy K. Tsai and Ravishankar K. Iyer, "FTAPE: A Fault Injection Tool to Measure Fault Tolerance," in *Proceedings of AIAA Computing in Aerospace 10*, San Antonio, TX, USA, March 28-30, 1995, pp. 339-346.

INVITED PRESENTATIONS AND OTHER PAPERS

- Timothy Tsai, "A Discussion of Performance Optimizations for Compiler-Based Buffer Overflow Instrumentation," in *Supplemental Proceedings of the International Conference on Dependable Systems and Networks (DSN-2006)*, June 25-28, 2006, Philadelphia, PA, USA, pp. 230-231.
- Sachin Garg, Navjot Singh, and Timothy Tsai, "SRTP+: An Efficient Scheme for RTP Packet Authentication," technical report, Avaya Labs, No. ALR-2004-001, January 9, 2004.
- Timothy K. Tsai and Navjot Singh, "Libsafe: Transparent System-wide Protection Against Buffer Overflow Attacks," in *Supplemental Proceedings of the International Conference on Dependable Systems and Networks (DSN-2002)*, June 23-26, 2002, Washington, D.C., USA, p. 541.
- Timothy K. Tsai, "Buffer Overflow Vulnerabilities and Solutions," Panel on Security and Information Assurance, *IEEE International Conference on Communications (ICC-2002)*, April 29, 2002, New York, NY, USA.
- Timothy Tsai and Navjot Singh, "Libsafe 2.0: Detection of Format String Vulnerability Exploits," white paper, Avaya Labs Research, February 6, 2001.
- Timothy K. Tsai, "Transparent Run-Time Defense Against Stack Smashing Attacks," *Illinois Corporate Affiliates Program*, September 13, 2000, Urbana, IL, USA.
- Arash Baratloo, Timothy Tsai, and Navjot Singh, "Libsafe: Protecting Critical Elements of Stacks," white paper, Lucent Bell Labs, December 25, 1999.
- Timothy K. Tsai, Gwan S. Choi, and Ravishankar K. Iyer, "Verification of Error Models Through Low-Level Simulation," in *Proceedings of the 3rd International Workshop on Integrating Error Models with Fault Injection*, April 25-26, 1994, Annapolis, MD, USA, pp. 15-16.

PATENTS

- Sachin Garg, Navjot Singh, Timothy Tsai, Yu-Sung Wu, and Saurabh Bagchi, "Stateful and Cross-Protocol Intrusion Detection for Voice over IP," US patent number 7,451,486, granted November 11, 2008.
- Sachin Garg, Navjot Singh, and Timothy Tsai, "Method for Real-Time Transport (RTP) Protocol Packet Authentication," US patent number 7,372,856, granted May 13, 2008.
- Navjot Singh and Timothy Tsai, "Security Vulnerability Investigator," pending US patent application number 20050005152, filed July 1, 2003.
- Navjot Singh and Timothy Tsai, "Fault Tolerance Software System with Periodic External Self-Test Failure Detection," US patent number 7,096,388, granted August 22, 2006.
- Navjot Singh and Timothy K. Tsai, "Method and Apparatus for Providing Extensible Object-Oriented Fault Injection," US patent number 6,484,276, granted November 19, 2002.
- Reinhard Klemm, Navjot Singh, and Timothy K. Tsai, "Distributed Indirect Software Instrumentation" US patent number 6,216,237, granted April 10, 2001.
- Timothy K. Tsai, "Fault Tolerance via N-Modular Software Redundancy Using Indirect Instrumentation," US patent number 6,161,196, granted December 12, 2000.

PROFESSIONAL ACTIVITIES

- International Conference on Dependable Systems and Networks (DSN/PDS)
- | | |
|----------------------------|------------------------|
| Program Committee Co-Chair | 2003 |
| Program Committee Member | 2002, 2004, 2005, 2009 |
| Publicity Chair | 2006 |

International Symposium on Software Reliability Engineering (ISSRE)
 Program Committee Co-Chair 2002
 Program Committee Member 2001, 2003, 2004, 2007
 Workshops, Panels, and Tutorials Co-Chair 2007
 Pacific Rim Dependable Computing conference (PRDC)
 Program Committee Member 2007, 2008, 2009
 International Computer Software and Applications Conference (ICCCN)
 Program Committee Member 2001, 2002, 2003
 International Computer Software and Applications Conference (COMPSAC)
 Program Committee Member 2004
 International Conference on Dependability (DEPEND)
 Program Committee Member 2009
 Testing: Academic and Industrial Conference (TAIC PART)
 Program Committee Member 2009
 Workshop on Applied Software Reliability (WASR)
 Program Committee Member 2006
 Workshop on Compiler and Architectural Techniques for Application Reliability and Security (CATARS)
 Program Committee Member 2008
 Reviewer for IEEE Transactions on Computers, IEEE Transactions on Dependable and Secure Computing, and
 Journal of Pattern Recognition Research

PROFESSIONAL MEMBERSHIPS

Member of IEEE and IEEE Computer Society

REFERENCES

Available upon request.

DETAILED DESCRIPTIONS OF PROFESSIONAL EXPERIENCE

Research activity at Hitachi Global Storage Technologies:

Statistical analysis of hard disk drive reliability using data mining

Although many of the individual failure modes for hard disk drives (HDD) are fairly well known, the overall phenomenon of HDD failure is less well understood. The lack of understanding includes lack of practical time-to-failure distributions, an understanding of the relative contributions of individual failure modes, and an ability to predict impending failures. The project attempted to investigate HDD reliability by using statistical methods on a large population of deployed enterprise HDD. A large population is necessary to extract statistically significant results despite relatively low failure rates. Much of the work focused on the logistical requirements for data collection, clean, and archiving, as well as the requisite data understanding and transformations for utilizing data mining algorithms. The main data mining algorithms utilized included classification trees and sequence detection. However, as with all data mining, the key was quality data and understanding the raw and transformed data needed.

Research activity at Sun Microsystems:

Reliability, availability, and serviceability (RAS) for large-scale computing systems

This work is a part of Phase II of the DARPA High Productivity Computing Systems research project awarded to Sun Microsystems to develop a next generation supercomputer capable of petaflop performance. The scale necessary for such a system necessarily involves such a large number of processors, memories, and other components that component failures are inevitable. Hence, failure detection, isolation, containment, and recovery are absolutely essential for achieving high performance and productivity. My work involved FMEA (failure mode effects analysis) of both hardware and software, design of the system fault tolerance architecture, and analysis and modeling of the RAS of components. Some of the interesting problems included automated checkpointing techniques that both leveraged and encompassed the large scale of resources, design of the software fault tolerance architecture, and modeling of failure rates and scenarios.

Automated support for Time-Dependent Reliability (TDR) field failure analysis

This work is a part of Phase II of the DARPA High Productivity Computing Systems research project awarded to Sun Microsystems to develop a next generation supercomputer capable of petaflop performance.

The scale necessary for such a system necessarily involves such a large number of processors, memories, and other components that component failures are inevitable. Hence, failure detection, isolation, containment, and recovery are absolutely essential for achieving high performance and productivity. My work involved FMEA (failure mode effects analysis) of both hardware and software, design of the system fault tolerance architecture, and analysis and modeling of the RAS of components. Some of the interesting problems included automated checkpointing techniques that both leveraged and encompassed the large scale of resources, design of the software fault tolerance architecture, and modeling of failure rates and scenarios.

Research activity at Avaya Laboratories, Avaya Inc.:

Computer security

This work focuses on using automated tools and middleware to detect and handle security vulnerabilities in software. Although the research prototypes have been initially implemented on Linux systems, the basic principles can be applied to systems in general. My efforts have been concentrated on several projects, including Libsafe, SVI, BOVScan, Voice over IP (VoIP) security testing, and intrusion detection systems for VoIP.

The Libsafe work at Avaya Labs builds on my previous work at Bell Labs (see the section on Computer Security at Bell Labs). Since the original version was released in 2000, many important improvements have been added. The two most significant added features are detection of format string vulnerabilities and detection of heap buffer overflows. Format string vulnerabilities are software bugs that allow unchecked strings containing “%n” as a format specifier to be passed to the printf() function or a function that eventually calls the core printf() function. Heap buffer overflows are buffer overflows that target buffers allocated on the heap. Although such overflows are more difficult to exploit than stack buffer overflows, many public security alerts have been issued for heap buffer overflows. As with stack buffer overflow vulnerabilities, format string and heap buffer overflow vulnerabilities are critical since they often result in remote root access. In addition to these feature improvements, I have also implemented various improvements to robustness and performance.

Voice over IP systems hold the promise of delivering increased feature integration and potential cost savings over a converged data and voice network. However, these advantages are tempered by the risks to the relative reliability and security of the voice network. My work on VoIP security focuses on mitigating the security risks inherent in data networks by performing security-oriented testing of VoIP phones and switches, analyzing the discovered vulnerabilities, proposing extensions to existing protocols to increase security without compromising compatibility, and developing intrusion detection systems that specifically target VoIP vulnerabilities.

Research activity at Cigital (formerly known as Reliable Software Technologies):

Java component reverification using fault injection and machine learning

This research uses fault injection and machine learning to test Java components and automatically extract rules for predicting early signs of failure. The methodology relies on Java bytecode modification and custom class loaders to transparently wrap Java classes. These Java wrappers inject faults into the Java classes and also collect data based on the fault injection. The wrappers also contain anomaly predicates that indicate obviously incorrect events, system state, or control flow. The collected data, along with the anomaly predicate data analyzed by machine learning algorithms to extract rules for matching events and system state to the triggering of anomaly predicates. Because the triggering of anomaly predicates usually signal an upcoming failure, such information can then be incorporated into intelligent wrappers around selected Java classes to permit the system to gracefully handle predicted failures before the actual failure occurs.

Research activity at Bell Laboratories, Lucent Technologies:

Reliability testing

This work focuses on the development of techniques and tools for evaluating and testing fault-tolerant systems based on fault injection techniques. The initial work included the development of new software-implemented fault injection techniques. Those experiences led to the realization of the importance of a portable, configurable, extendible fault injection testbed and the subsequent creation of the Dependability Test Suite (DTS) fault injection tool. The DTS tool is available for free download at <<http://www.research.avayalabs.com/projects/swift/ntdts>>.

The DTS work achieved several significant goals. First, we created a portable fault injection tool that yields useful results on Windows NT and Linux. The majority of fault injection research has focused on UNIX systems. Microsoft operating systems have received less attention due to the difficulty of obtaining source code and of porting existing tools to those operating systems. DTS is one of the few fault injection tools for Windows NT, and the first to be publicly available. Second, the DTS results have resulted in the improvement of fault tolerance software developed at Bell Labs. The DTS testing identified faults and scenarios that the fault tolerance software did not properly handle. We were then able to debug more easily the problems in the fault tolerance software. Third, the DTS results allowed us to compare competing fault tolerance products and applications. At Bell Labs, we were interested in determining how well our fault tolerance software performed against commercial solutions, such as Microsoft Cluster Server (MSCS). Our results showed that the Bell Labs software was able to detect and recover from failures as well as MSCS. In fact, after using the DTS results to improve the Bell Labs software, our fault tolerance software was able to recover from failures that the default MSCS software could not handle.

The long-term vision for the DTS work is the development of a fault tolerance benchmark that will be as easy to use and as widely accepted as many performance benchmarks. In a primitive sense, the DTS tool already has the potential to obtain quantitative comparisons of system fault tolerance. We have used results from DTS experiments to compare fault tolerance middleware, applications, and even operating systems. However, accurate comparisons require a common set of faults and workloads that is unbiased and comprehensive. The fault tolerance benchmark may express its output in terms of failure coverage (i.e., the ability of the tested system to handle correctly the failures caused by the injected set of faults). A more challenging benchmark output form is a mathematical quantity such as availability or reliability.

Software-implemented fault tolerance

Bell Labs has worked on the software-implemented fault tolerance (SwiFT) project for approximately eight years (see <http://www.bell-labs.com/projects/swift>). I joined the work in its latter stages. My contributions include development of parts of the Windows NT port for the *watchd* process-monitoring component, creation of Java interfaces to the *libft* library, and support for source code management.

I have also developed the *periodic external self-test* (PEST) method for monitoring of server processes in a client-server system. PEST monitoring is simple to use and yet detects a wide range of failures. The core idea is to create a small client program that periodically sends requests to the server program. Traditional detection methods rely on indirect detection of process failure. Such detection methods actually monitor symptoms and resources that indicate that a process has failed or that it will possibly fail. In contrast, PEST monitoring directly tests the exact specifications that determine process failure. If a server program is unable to deliver correct responses to a requesting client, then that server program has failed, and this is precisely what the PEST client program tests. Consequently, PEST is able to detect many failure scenarios that elude traditional detection methods. Such scenarios include process hangs and quality of service degradation.

Computer security

This work addresses the problem of detecting attempts to gain unauthorized access to vulnerable computer systems via buffer overflow schemes. Most current techniques rely on recompilation of source to add code to check for buffer overflow scenarios. Unfortunately, source code may not always be available, and the techniques incur a noticeable performance overhead. Our work investigates transparent methods that do not require recompilation of source code. We have developed several methods based on dynamic libraries. These methods include transparent interception of vulnerable system library functions and in-memory binary modification. In particular, the interception method for library functions incurs almost no performance overhead while being able to detect several known attacks. We have implemented this interception method as the *libsaf*e library. See <http://www.research.avayalabs.com/projects/libsafe.html> for a copy of the white paper. Libsafe has been incorporated into the major Linux distributions.

Software instrumentation

Some of my original efforts at Bell Labs dealt with the idea of creating a general-purpose software instrumentation tool. Consequently, I created the *Prism* instrumentation tool. The tool uses the *gdb* debugger to control and to observe the execution of a process. The Prism tool provided a graphical front end that allowed the user to view the source code and data structures of a program. The user could then set breakpoints at desired statements in the program and specify actions to perform at those breakpoints. An example of a useful action was to update the value displayed by a graphical object. This action allowed the user to create a graphical user interface independently of the program. The user only had to select the variables to monitor, the update breakpoints or periods, and the type of graphical object (e.g., bar charts, time graphs, etc.).

I also used the Prism tool to create a transparent, redundant fault tolerance scheme for software processes. The tool spawns multiple copies of the same process on several machines. The same breakpoints are set for all machines. When the breakpoints are triggered, a subset of the process state on each machine is sent to the Prism front-end. The front-end compares the values and identifies any mismatches. Upon detection of a mismatch, the front-end initiates recovery actions for the associated process, including terminating the faulty process and restarting a new process based on checkpoint data from the remaining processes. This scheme detects process hangs and internal data corruption without the need for recompilation of source code.