REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* 11/13/2012	**2. REPORT TYPE** Final	**3. DATES COVERED** *(From - To)* 07/01/2007 - 09/30/2012

4. TITLE AND SUBTITLE (MURI-07) HELIX: A SELF-REGENERATIVE ARCHITECTURE FOR THE INCORRUPTIBLE ENTERPRISE	**5a. CONTRACT NUMBER**
	5b. GRANT NUMBER FA9550-07-1-0532
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) John C. Knight	**5d. PROJECT NUMBER**
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Virginia, University of California Davis, University of California Santa Barbara, University of New Mexico	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Office of Scientific Research Suite 325, Room 3112 875 Randolph Street Arlington, VA 22203-1768	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFOSR
	11. SPONSOR/MONITOR'S REPORT NUMBER(S) AFRL-OSR-VA-TR-2012-1202

12. DISTRIBUTION / AVAILABILITY STATEMENT

Distribution A - Approved for public release

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The focus of the Helix project was to dramatically improve the status quo for architecting secure and resilient information systems by developing a coherent body of scientific and engineering knowledge to design and evaluate new algorithms and concepts for self-regenerative systems. The hallmarks of such systems is that they are automated and proactive; they can adapt and reconfigure themselves to present attackers with an ever-shifting attack surface; and they can self-heal when confronted with both known and unknown attacks.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF: DISTRIBUTION A			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** Dr. Robert Herklotz
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	20	**19b. TELEPHONE NUMBER** *(include area code)* 703-696-6565

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

HELIX: A SELF-REGENERATIVE ARCHITECTURE FOR THE INCORRUPTIBLE ENTERPRISE

MURI Topic: Enterprise Health: Self-Regenerative Incorruptible Enterprise

MURI Program Manager: Dr. Robert Herklotz, AFOSR

Contract: FA9550-07-1-0532

Dates: 2007-2012

PI: John C. Knight (knight@virginia.edu), http://helix.cs.virginia.edu/

Participating Institutions:

University of Virginia (Lead): John C. Knight (Principal Investigator), Jack W. Davidson, David Evans, Westley Weimer, Anh Nguyen-Tuong

University of California Davis: Hao Chen, Karl N. Levitt, Jeff Rowe, Zhendong Su, Felix Wu

University of California Santa Barbara: Frederic Chong

University of New Mexico: Stephanie Forrest, Jared Saia

1 Helix Summary

The focus of the Helix project was to dramatically improve the status quo for architecting secure and resilient information systems by developing a coherent body of scientific and engineering knowledge to design and evaluate new algorithms and concepts for self-regenerative systems. The hallmarks of such systems is that they are **automated** and **proactive**; they can adapt and reconfigure themselves to present attackers with an **ever-shifting attack surface**; and they can **self-heal** when confronted with both **known** and **unknown attacks**.

Over the 5-year period funded by the MURI research program, the Helix team has successfully pioneered innovations in key areas of research to lay the groundwork for self-regenerative systems. We would like to thank the MURI program for enabling research that otherwise would probably not have seen the light of day. Many exciting breakthroughs resulted from the collaboration between disciplines and researchers that were brought together under this MURI umbrella.

The table below provides a summary of highlights organized by research areas and serves as a road map for the rest of this report.

Research Areas	**Highlights**
Artificial Software Diversity	Developed several practical high-entropy diversity techniques, pioneered high-

(Section 3.1)	frequency temporal diversity techniques, proactive diversity.
Application-level Virtual Machines (Section 3.2)	Developed foundational and optimized infrastructure for armoring binaries with security transformations, without requiring access to source code.
Automated Program Repair (Section 3.3)	Developed automated repair techniques using genetic evolutionary algorithms. Fixed 55% of real-world bugs for under $8 each.
Malware Detection and System Repair (Section 3.4)	Developed novel framework for automatically removing malware from and repairing its damage to a system. Developed first practical framework for detecting and analyzing vulnerabilities of insecure component usage based on differential testing.
End-to-end Security for Web and Mobile Applications (Section 3.5)	Invented techniques for preventing XSS exploits. Developed techniques to enable cross-application information-flow tracking. Developed techniques for detecting potentially malicious clones for Android applications. Developed automated scalable privacy scanners for Android.
Scalable Distributed Algorithms (Section 3.6)	Described and proved correct the first algorithm to solve the Byzantine agreement problem in a scalable manner. Described and proved correct an efficient algorithm to solve the Secure Multiparty Computation (SMPC) problem.
Hardware/Software Co-design for Security (Section 3.7)	Developed hardware design and architecture techniques to provide a provably secure foundation for the Helix system. Developed techniques that dramatically reduce the performance and energy overhead of software diversity techniques.

To provide context for the research thrusts and to situate the importance of the various key technical accomplishments, we present a high-level conceptual overview of the Helix Self-Regenerative Architecture as described in the *original proposal* (Section 2). The Helix vision drove the research agenda and provided a common framework on which to ground our collaborative research efforts.

2 Helix Self-Regenerative Architecture Conceptual Overview

Figure 1 provides a high-level overview of the *Helix self-regenerative architecture*. An application running in Helix is treated in a holistic way, with information being shared across development, deployment, execution, and response phases in ways that are not possible with traditional architectures. Helix provides a novel restructuring of the standard program development and execution tool chain. Instead of viewing the tool chain as just a series of steps to transform an application from source code to executable form, we take a more comprehensive view in which program metadata can be deposited in an *Application Information Repository (AIR)*, and subsequently manipulated and enhanced at all phases of a program's lifecycle, to enable the development of novel and accurate self-regeneration algorithms.

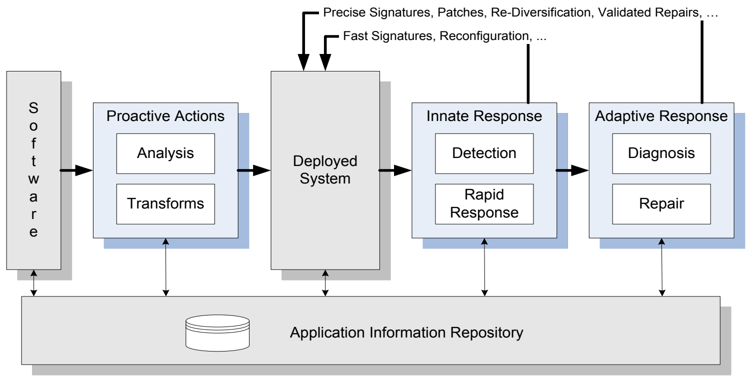
Precise Signatures, Patches, Re-Diversification, Validated Repairs, …
Fast Signatures, Reconfiguration, …
Software
Proactive Actions
Analysis
Transforms
Deployed System
Innate Response
Detection
Rapid Response
Adaptive Response
Diagnosis
Repair
Application Information Repository

Figure 1. Conceptual Overview of Helix Architecture

Starting with applications in source or binary form as input, Helix proactively analyzes and transforms applications to augment them with self-sensing and self-protection capabilities. The primary goals of this pre-deployment phase are: (1) to infer and record information about application structure and semantics that can be used for automated analysis and repair in later phases; (2) to transform applications to provide full, or at least partial, immunity to a large range of attack classes; and (3) to instrument applications with accurate sensing capabilities that will collect information from executions to enable detection and analysis of attacks.

At deployment time, Helix-enabled applications possess self-monitoring, self-protection and self-actuation capabilities. These capabilities are enabled by running applications under control of *Strata*, a lightweight virtual machine known as a *software dynamic translator* (SDT). Strata, developed at the University of Virginia, provides detailed, precise, and selective monitoring of running applications. Combined with new synthetic diversity techniques, this monitoring provides the ability to detect attacks and to diagnose properties about the attack and the exploited vulnerability. Strata provides the ability to rewrite application code on-demand for dynamically shifting the attack surface of applications, and effecting repair actions (including patch installation) without requiring system restarts.

Detection of an attack triggers an *innate response*—a quick response that contains the damage caused by the attack and restores service, though possibly in a degraded form. Examples include reconfiguring the application to provide higher security protection with reduced performance, providing degraded levels of services, slowing down programs, installing fast (but possibly imprecise) signatures and re-diversifying applications.

While the innate response buys time and limits damages, the *adaptive response* provides a long-term improvement to the application. It involves diagnosing the root cause of an attack and developing semantically-valid responses. Using information generated by precise sensors and made available through the AIR, Helix generates signatures and filters automatically to block further attacks, even if they are polymorphic or metamorphic. In addition to masking vulnerabilities, Helix will seek to repair them permanently. In cases where the attack is detected only after some persistent damage has occurred, Helix will use state regeneration techniques to return the system to an uncompromised state while losing as little good data as possible.

Sensor alerts, repair and recovery actions will be propagated throughout the Helix architecture using scalable, robust, and attack-resilient protocols. In this way, knowledge obtained or inferred in one part of the system will benefit the system as a whole.

3 Research Thrusts

3.1 Artificial Software Diversity

By presenting a continuously changing attack surface, Helix disrupts attackers' observer-orient-decide-act (OODA) loop, forcing attackers to operate within controlled and configurable small windows of vulnerability. Helix has developed innovative high-entropy diversity techniques that operate in both the spatial and temporal dimensions. Helix further improves on the state-of-the-art by *proactively* morphing the attack surface of programs, instead of reacting to attackers' probes. All universities collaborated on this research thrust.

3.1.1 Capabilities and accomplishments

- Developed first practical implementation of Instruction Set Randomization using dynamic binary rewriting techniques. Our implementation is both fast (<10%) and practical, operating directly on binaries without mandating the availability of source code.
- Developed first algorithm and implementation of temporal diversity operating at high-frequencies (<100msec). Our technique operates directly on binaries.
- Developed model for evaluating the effectiveness of temporal diversity.
- Developed Instruction Location Randomization (ILR), a high-entropy successor to Address Space Layout Randomization. On a 32 bit architecture, ILR provides 31 bit of entropy instead of 17 bit for ASLR.
- Developed Stack Layout Transformation, a technique to randomize the layout of the stack that operates directly on binaries.

- Developed ability to generate program variants proactively using genetic programming techniques.

3.1.2 Noteworthy events

- 2007 - Invited Lectureship, 3rd International Summer School on Advanced Architecture and Compilation for Embedded Systems, L'Aquila, Italy
- 2008 - IEEE Computer Society Taylor L. Booth Education Award
- 2008 - Invited Lectureship, Inaugural Indo US Engineering Faculty Leadership Institute, Mysore, India
- 2008 - Invited Talk, "Software Dynamic Translation: Implementation and Applications," Microsoft Research, Bangalore India
- 2010 - Invited Distinguished Lecture, "Software Protection and Assurance using Process-level Virtualization," Sandia National Laboratory, Albuquerque, NM
- 2010 -Invited Lectureship, Inaugural ACM International Summer School on Information Security and Protection, Beijing, China
- 2011 - Invited Lectureship, 2nd ACM International Summer School on Information Security and Protection, Ghent Belgium
- 2012 - Filed provisional patent application for ILR technique. Currently converting to a full patent application
- 2012 - Invited Lectureship, 3rd ACM/DAPA International Summer School on Information Security and Protection, Tucson, Arizona

3.1.3 Research Summary

Instruction Location Randomization. ILR randomizes the location of every instruction in a program, thwarting an attacker's ability to re-use program functionality (e.g., arc-injection attacks and return-oriented programming attacks). ILR operates on arbitrary executable programs, requires no compiler support, and requires no user interaction. Thus, it can be automatically applied post-deployment, allowing easy and frequent re-randomization. Our preliminary prototype, working on 32-bit x86 Linux ELF binaries, provides a high degree of entropy. Individual instructions are randomly placed within a 31-bit address space. Thus, attacks that rely on a priori knowledge of the location of code or derandomization are not feasible. We demonstrated ILR's defensive capabilities by defeating attacks against programs with vulnerabilities, including Adobe's PDF viewer, acroread, which had an in-the-wild vulnerability. The average run-time overhead of ILR was 13% with more than half the programs having effectively no overhead (15 out of 29), indicating that ILR is a realistic and cost-effective mitigation technique.

Instruction Set Randomization (ISR). We pioneered the use of binary rewriting techniques for implementing fast and practical realizations of instruction set randomization. We further improved on ISR by developing a temporal version of ISR wherein the program binary can be re-encrypted at a very fast rate (< 100msec).

Temporal Diversity. To improve the strength of an artificially diverse system in which searching the state space can be accomplished relatively rapidly, an intuitive approach is to frequently re-randomize the system attribute subject to diversity, i.e., to effect dynamic artificial diversity. Applying dynamic artificial diversity effectively changes the attack surface seen by the adversary, and we have coined the phrase Metamorphic Shield (MMS) to describe the approach. The intuition of many people is that an MMS would provide considerable protection even in a context of low entropy because of the re-randomization. The idea of using an MMS in a general way by varying a variety of system characteristics over time is tempting. Contrary to intuition, our analysis reveals that dynamic diversity provides limited benefit except in special cases. In particular, it offers benefit for attacks that seek to leak information. We present a case study of the use of dynamic diversity applied to Instruction Set Randomization that is subject to an incremental attack on the key. Further, we demonstrated the ability to dynamically modify the attack surface of program binaries at the rate of 100 msec.

Software Mutational Robustness and Proactive Diversity. Mutational robustness is a key concept in evolutionary biology, which we believe is applicable to software. Given a population of variant programs, created from an original program by applying random mutations, we measure which variants still pass all available test cases and call them neutral. The fraction of all variants that are neutral is defined as the program's mutational robustness. Even when the mutations are restricted to statements executed by the test cases, mutational robustness is surprisingly high, 36.75% on a corpus of programs taken from 22 production software projects, the Siemens benchmark suite, and a few specially constructed programs. The results hold for mutations at both the source code and assembly instruction levels, across various programming languages, and are not correlated with inadequate test suite coverage.

We believe that mutational robustness is an inherent property of software, existing even when a program is correct according to its specification. Rather than an overhead cost or indicator of test suite inadequacy, it is an opportunity to create useful proactive diversity because neutral mutations often cause nontrivial algorithmic changes. In an initial demonstration, we generated and selected diverse populations of neutral program variants. For a program with seven or more held-out latent bugs, we can, on average, construct and select seven neutral variants such that each bug is repaired by at least one variant.

3.1.4 Publications

Daniel Williams, Wei Hu, Jack W. Davidson, Jason D. Hiser, John C. Knight, Anh Nguyen-Tuong, "Security through Diversity: Leveraging Virtual Machine Technology," IEEE Security and Privacy, vol. 7, no. 1, pp. 26-33, Jan./Feb. 2009, doi:10.1109/MSP.2009.18

Co, M., Coleman, C. L., Davidson, J. W., Ghosh, S., Hiser, J. D., Knight, J. C., Nguyen-Tuong, A., 'A Lightweight Software Control System for Cyber Awareness and Security,' International Symposium on Resilient Control Systems, Idaho Falls, ID, August, 2009.

Knight, John. "Diversity." *Dependable and Historic Computing* (2011): 298-312.

Co, Michele, Jack W. Davidson, Jason D. Hiser, John C. Knight, Anh Nguyen-Tuong, David Cok, Denis Gopan et al. "PEASOUP: preventing exploits against software of uncertain provenance (position paper)." In *Proceedings of the 7th International Workshop on Software Engineering for Secure Systems*, pp. 43-49. ACM, 2011.

Evans, David, Anh Nguyen-Tuong, and John Knight. "Effectiveness of moving target defenses." *Moving Target Defense* (2011): 29-48.

Wang, A., Hiser, J.D., Nguyen-Tuong, A., Davidson, J. W., Knight, J. C., 'Component-Oriented Monitoring of Binaries for Security,' in 44th Hawaii International Conference on System Sciences (HICSS), January, 2011.

Goues, Claire, Anh Nguyen-Tuong, Hao Chen, Jack W. Davidson, Stephanie Forrest, Jason D. Hiser, John C. Knight, and Matthew Gundy. "Moving Target Defenses in the Helix Self-Regenerative Architecture." Moving Target Defense II (2012): 117-149.

Eric Schulte, Zachary P. Fry, Ethan Fast, Stephanie Forrest, Westley Weimer: Software Mutational Robustness: Bridging The Gap Between Mutation Testing and Evolutionary Biology. CoRR abs/1204.4224 (2012).

Hiser, Jason, Anh Nguyen-Tuong, Michele Co, Matthew Hall, and Jack W. Davidson. "ILR: Where'd My Gadgets Go?." In *Security and Privacy (SP), 2012 IEEE Symposium on*, pp. 571-585. IEEE, 2012.

Rodes, Benjamin. "Stack layout transformation: towards diversity for securing binary programs." In *Proceedings of the 2012 International Conference on Software Engineering*, pp. 1543-1546. IEEE Press, 2012.

Rodes, B., Nguyen-Tuong, A., Hiser, J. D., Knight, J., Davidson, J., Co, M., 'Defense Against Stack Based Attacks Using Speculative Stack Layout Transformation,' in Third International Conference on Runtime Verification, Istanbul, Turkey, September, 2012.

3.2 Application-level Virtual Machines

We have demonstrated and validated through experimental evaluation the effectiveness of using software dynamic translation technique for armoring binaries with various security transformation techniques, e.g., ISR, ILR, SLX. The advantages of using software dynamic translation are that security transformations can be applied directly on binaries post-deployment, security transformations may be composed to provide defense-in-depth, and finally, security configurations may be tailored on a per application basis. The work on the Strata virtual machine was carried out at the University of Virginia.

3.2.1 Capabilities and accomplishments

- The Strata virtual machine is a key enabling technology for a variety of security transformations, including the aforementioned Instruction Set Randomization (ISR), Instruction Location Randomization (ILR), Stack Layout Transformations (SLX), and Metamorphic Shield techniques (Section 3.1.2).
- The Strata virtual machine enables security transformations to be efficiently applied directly to binaries, without requiring source code. The ability to apply security transformations post-deployment is a key advantage in making the application of powerful security techniques practical.

3.2.2 Noteworthy events

- The Strata virtual machine developed at UVA plays a central role as a foundational technology for an IARPA-funded project under the umbrella of the STONESOUP program.

3.2.3 Research Summary

Key to enabling the composition of powerful security transformations is the re-engineering of the standard development toolchain. Instead of viewing the tool chain as just a series of steps to transform an application from source code to executable form, we take a more comprehensive view in which program metadata can be deposited in an *Application Information Repository (AIR)*, and subsequently manipulated and enhanced at all phases of a program's lifecycle. The enhanced Helix toolchain culminates in an execution environment wherein an application-level virtual machine, Strata, controls the execution of the binary and can apply arbitrary security transformations (See Publications, 3.2.4).

We believe that the Strata virtual machine technology is ready to transition from Technical Readiness Level 5 (TRL-5: testing of integrated technology components in representative environment) to Technical Readiness Level 6 (TRL-6: prototype implementation on full-scale realistic systems).

3.2.4 Publications

Daniel Williams, Wei Hu, Jack W. Davidson, Jason D. Hiser, John C. Knight, Anh Nguyen-Tuong, "Security through Diversity: Leveraging Virtual Machine Technology," IEEE Security and Privacy, vol. 7, no. 1, pp. 26-33, Jan./Feb. 2009, doi:10.1109/MSP.2009.18

Co, M., Coleman, C. L., Davidson, J. W., Ghosh, S., Hiser, J. D., Knight, J. C., Nguyen-Tuong, A., 'A Lightweight Software Control System for Cyber Awareness and Security,' International Symposium on Resilient Control Systems, Idaho Falls, ID, August, 2009.

Wang, A., Hiser, J.D., Nguyen-Tuong, A., Davidson, J. W., Knight, J. C., 'Component-Oriented Monitoring of Binaries for Security,' in 44th Hawaii International Conference on System Sciences (HICSS), January, 2011.

Hiser, Jason, Anh Nguyen-Tuong, Michele Co, Matthew Hall, and Jack W. Davidson. "ILR: Where'd My Gadgets Go?." In Security and Privacy (SP), 2012 IEEE Symposium on, pp. 571-585. IEEE, 2012.

Rodes, Benjamin. "Stack layout transformation: towards diversity for securing binary programs." In Proceedings of the 2012 International Conference on Software Engineering, pp. 1543-1546. IEEE Press, 2012.

Rodes, B., Nguyen-Tuong, A., Hiser, J. D., Knight, J., Davidson, J., Co, M., 'Defense Against Stack Based Attacks Using Speculative Stack Layout Transformation,' in Third International Conference on Runtime Verification, Istanbul, Turkey, September, 2012.

3.3 Automated Program Repair

Automatic repair of programs has been a longstanding goal in software engineering, yet debugging remains a largely manual process. The collaboration between Wes Weimer at the University of Virginia, and Stephanie Forrest at the University of New Mexico has yielded novel automated methods for locating and repairing bugs in software using evolutionary programming techniques. The approach is generic and was demonstrated to work for a wide range of software vulnerabilities (both security-critical and non-security-critical) and covers a wide range of attack classes and programming languages. Towards the end of the project, the evolutionary programming approaches were extended to optimize programs for graphical applications, outperforming the best-known algorithms for producing shaders.

3.3.1 Capabilities and accomplishments

- Demonstrated self-healing in a closed loop.
- Automatically repaired real-world bugs for less than $8/bug on average.
- Demonstrated the ability to automatically repair security vulnerabilities, covering a wide range of programming languages and attack classes.
- Extended the work genetic programming to automatically generate shaders in graphical applications. The shaders produced were best in class and outperformed the best known algorithms to date.

3.3.2 Noteworthy events

- 2009 - Best Paper Award, International Conference on Software Engineering
- 2009 - IFIP TC2 Manfred Paul Award "for excellence in software: theory and

practice"

- 2009 - Best Paper Award, Genetic and Evolutionary Computing Conference
- 2009 - Gold Award, Human-Competitive Results Produced By Genetic and
- Evolutionary Computation
- 2009 - Best Paper Award, Search Based Software Testing
- 2010 - Invited Keynote, Systems Programming Languages and Applications: Software for Humanity Conference
- 2010 - Forrest appointed Co-Chair of SFI Science Board
- 2011 - Organized Seminar, "Self-Repairing Programs", Dagstuhl
- 2012 - Bronze Award, Human-Competitive Results Produced By Genetic and Evolutionary Computation
- 2012 - Invited Talk, World Economic Forum 2012 - Featured Paper Award, IEEE Transactions on Software Engineering
- 2012 - Association for Computing Machinery/AAAI Allen Newell Award. For innovations in computing technology that have made significant contributions that enable computer science to solve real-world challenges

3.3.3 Research Summary

The Helix self-healling approach works on off-the-shelf legacy applications and does not require formal specifications, program annotations or special coding practices. Once a program fault is discovered, an extended form of genetic programming is used to evolve program variants until one is found that both retains required functionality and also avoids the defect in question. Standard test cases are used to exercise the fault and to encode program requirements. After a successful repair has been discovered, it is minimized using using structural differencing algorithms and delta debugging. We recently conducted a systematic evaluation 105 defects from 8 open-source programs totaling 5.1 million lines of code and involving 10,193 test cases. Our method (called GenProg) automatically repaired 55 of those 105 defects. Public cloud computing prices allow our 105 runs to be reproduced for $403; a successful repair completes in 96 minutes and costs $7.32, on average.

3.3.4 Publications

J. Karlin, J. Rexford, and S. Forrest "Autonomous security for autonomous systems" Computer Networks 52:29082923 (2008).

P. Holme, J. Karlin, and S. Forrest "An integrated model of traffic, geography and economy in the Internet." ACM SIGCOMM Computer Communication Review 38:3, pp. 7-15 (2008).

M. Moses, S. Forrest, A.~L. Davis, M. Lodder, and J.~H. Brown ``Scaling theory for information networks.'' Journal of the Royal Society Interface 5:29, pp. 1391-1510 (2008).

J. R. Crandall, R. Ensafi, S. Forrest, J. Ladau, and B. Shebaro. The ecology of modern malware. Proceedings of the New Security Paradigms Workshop (NSPW) (2008).

F. Esponda, S. Forrest, and P. Helman "Negative representations of information." International Journal of Information Security 8:5 pp. 331 (2009). doi:10.1007/s10207-009-0078-1.

S. Forrest, T. Nguyen, W. Weimer, and C. Le Goues ``A Genetic Programming Approach to Automated Software Repair.'' Genetic and Evolutionary Computation Conference (2009) Best Paper Award, Genetic Programming Track.

C. Le Goues, T. Nguyen, S. Forrest, W. Weimer ``GenProg: Automatic Bug Correction in Real Programs.'' ACM Transactions on Software Engineering 38:1 pp. (2012).

W. Weimer, S. Forrest, C. Le Goues, T. Nguyen ``Automatic program repair with evolutionary computation.'' Communications of the ACM Research Highlight 53:5 pp. 109-116 (2010).

E. Fast, C. Le Goues, W. Weimer, and S. Forrest ``Designing better fitness functions for automated program repair.'' Genetic and Evolutionary Computation Conference (2010).

C. Le Goues, S. Forrest, W. Weimer ``The case for software evolution.'' 2010 FSE/SDP Workshop on the Future of Software Engineering Research (submitted June, 2010).

E. Schulte, S. Forrest, and W. Weimer ``Automated Program Repair through the Evolution of Assembly Code'' 25nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2010).

S. Hofmeyr, T. Moore, S. Forrest, B. Edwards and G. Stelle "Modeling Internet-scale policies for cleaning up malware." Tenth Workshop on Economics of Information Security (WEIS) (2011).

S. Banerjee, D. Levin, M. Moses, F. Koster, and S. Forrest "The Value of Inflammatory Signals in Adaptive Immune Responses." International Conference on Artificial Immune Systems (ICARIS). 2011.

T. Nguyen, D. Kapur, W. Weimer, and S. Forrest "Using Dynamic Analysis to Discover Polynomial and Array Invariants." International Conference on Software Engineering (ICSE 2012). ACM SIGSOFT Distinguished Paper Award.

C. Le Goues, M. Dewey-Vogt, S. Forrest, and W. Weimer "A systematic study of automated program repair: Fixing 55 out of 105 bugs for $8.00 each." International Conference on Software Engineering (ICSE 2012).

C. Le Goues, W. Weimer, and S. Forrest "Representations and Operators for Improving Evolutionary Software Repair." Genetic and Evolutionary Computation Conference (GECCO 2012). Nominated for best paper.

C. Le Goues, A. Nguyen-Tuong, H Chen, J. W. Davidson, S. Forrest, J. D. Hiser, J. C. Knight and M. Van Gundy. "Moving Target Defenses in the Helix Self-Regenerative Architecture." In S. Jajodia, A. K. Ghosh, V. S. Subrahmanian, V. Swarup, C. Wang, and X. S. Wang Eds. Moving Target Defense II: Application of Game Theory and Adversarial Modeling. pp. 115–146 (2012).

B. Edwards, T. Moore, G. Stelle, S. Hofmeyr and S. Forrest. "Beyond the Blacklist: Modeling Malware Spread and the Effect of Interventions." New Security Paradigms Workshop (to appear).

3.4 Malware Detection and System Repair

In cases when an attack is detected only after some persistent damage has occurred, this research thrust seeks to use state regeneration techniques to return a Helix system to an uncompromised state while losing as little good data as possible. This research thrust was a collaborative effort between UC Santa Barbara and UC Davis.

3.4.1 Capabilities and accomplishments

- Detected malware overlooked by up to 97% of commercial anti-malware tools
- A new, effective program behavior representation for matching and detection of polymorphic malware
- First practical framework for detecting and analyzing vulnerabilities of insecure component usage based on differential testing
- Dynamic and static binary analysis to detect unsafe dynamic component loadings

3.4.2 Noteworthy events

- Dr. Su chaired ISSTA 2012, the leading venue in software testing and analysis. Su co-chaired the highly regarded International Static Analysis Symposium (SAS), 2009
- Dr. Su was named a highly prestigious Chancellor Fellow at UC Davis (2010-2015)

3.4.3 Research Summary

Malware Detection. We have introduced a precise and succinct program behavior representation that characterizes high-level object-accessing patterns as regular expressions. We show that software variants derived from the same code exhibit similar behavior representations and develop effective clustering and matching algorithms. Our evaluation results on a large malware collection demonstrate that the new model is both precise and succinct for effective and scalable matching and detection of polymorphic malware. Related to this effort, we have also developed a highly effective binary clone detection technique.

Back-to-the-future. Back-to-the future is a novel framework for automatically removing malware from and repairing its damage to a system. The primary goal of our framework is to preserve system integrity. Our framework monitors and logs untrusted programs' operations. Using the logs, it can completely remove malware programs and their effects on the system. Our framework does not require signatures or other prior knowledge of malware behavior. We implemented this framework on Windows and evaluated it with seven spyware, trojan horses, and email worms. Comparing our tool with two popular commercial anti-malware tools, we found that our tool detected all the malware's modifications to the system detected by the commercial tools, but the commercial tools overlooked up to 97% of the modifications detected by our tool. The runtime and space overhead of our prototype tool is acceptable. Our experience suggests that this framework offers an effective new defense against malware.

Differential testing for analysis. We have formulated and explored a number of novel, yet practical problems concerning dynamic component loading. In a sequel of papers, we have developed dynamic and static techniques to detect unsafe dynamic loadings and detected serious vulnerabilities in popular software. We have also developed the first practical framework for detecting and analyzing vulnerabilities of insecure component usage. We introduce and formulate the problem. Our core analysis approach is based on differential testing. Our results on popular Windows applications have revealed new security vulnerabilities, which were acknowledged by affected software vendors.

3.4.4 Publications

Back to the Future: A Framework for Automatic Malware Removal and System Repair. Francis Hsu, Hao Chen, Thomas Ristenpart, Jason Li, and Zhendong Su. 2006 Annual Computer Security Applications Conference, Miami Beach, FL, December, 2006.

Feature Omission Vulnerabilities: Thwarting Signature Generation for Polymorphic Worms. Matthew Van Gundy, Hao Chen, Zhendong Su, and Giovanni Vigna. 23rdAnnual Computer Security Applications Conference, Miami Beach, FL, December 10-14, 2007.

Bezoar: Automated Virtual Machine-based Full-System Recovery from Control-Flow Hijacking Attacks. Daniella A. S. de Oliveria, Jedidiah R. Crandall, Gary Wassermann, Shaozhi Ye, S. Felix Wu, Zhendong Su, and Frederic T. Chong. In Proceedings of NOMS 2008, Salvador, Bahia, Brazil, April 7-11, 2008.

Detecting Code Clones in Binary Executables. Andreas Saebjoernsen, Jeremiah Willcok, Thomas Panas, Daniel Quinlan, and Zhendong Su. In Proceedings of ISSTA 2009, Chicago, IL, July 19-23, 2009.

Automatic Detection of Unsafe Dynamic Component Loadings. Taeho Kwon and Zhendong Su. IEEE Transactions on Software Engineering (TSE), 38(2), pages 293-313, 2012 (invited paper). Extended version of our ISSTA'10 paper.

Modeling High-Level Behavior Patterns for Precise Similarity Analysis of Software. Taeho Kwon and Zhendong Su. In Proceedings of ICDM 2011, Vancouver, Canada, December 11-14, 2011.

Detecting and Analyzing Insecure Component Usage. Taeho Kwon and Zhendong Su. In Proceedings of ACM SIGSOFT FSE 2012, Research Triangle Park, NC, November 2012.

Static Detection of Unsafe Component Loadings. Taeho Kwon and Zhendong Su. In Proceedings of CC 2012, Tallinn, Estonia, March 24-April 1, 2012.

3.5 End-to-end Security for Web and Mobile Applications

The Helix vision applies to all levels of the hardware/software stack. In this research thrust, we investigate security techniques for coping with malware, and for securing web and mobile applications. Hao Chen, Zhendong Su and their collaborators at the University of California at Davis primarily led this research thrust.

3.5.1 Capabilities and accomplishments

- First application of ISR-inspired diversity technique to thwart cross-site scripting attack
- Discovered 57,299 potential privacy leaks in 7,414 Android applications
- Detected at least 141 Android applications that have been the victims of cloning
- A set of static and symbolic analysis techniques to detect injection and access control vulnerabilities in Web applications
- A purely client-based technique to detect XSS worms

3.5.2 Noteworth events

- ISSTA 2010 paper received an ACM SIGSOFT Distinguished Paper Award. The extended version was invited and accepted to IEEE TSE (2012) for the best papers at ISSTA 2010. Revealed serious vulnerabilities in popular software, such as Microsoft Office: http://osvdb.org/affiliations/1420-university-of-california-davis.
- SIGSOFT FSE 2012 work detected serious vulnerabilities in IETab and other browsers (altogether affecting millions of users).
- Student Fangqi Sun and Liang Xu's work at USENIX Security 2011 was invited to compete in the CSAW CyberSecurity Competition and was named a finalist.
- Su received the highly competitive Microsoft Software Engineering Innovation Foundation Award, 2012.

3.5.3 Research Summary

Noncespaces. Cross-site scripting (XSS) vulnerabilities are among the most common and serious web application vulnerabilities. Eliminating XSS is challenging because it is difficult for web applications to sanitize all user inputs appropriately. We present Noncespaces, a technique that enables web clients to distinguish between trusted and untrusted content to prevent exploitation of XSS vulnerabilities. Using Noncespaces, a web application randomizes the XML namespace prefixes of tags in each document before delivering it to the client. As long as the attacker is unable to predict the randomized prefixes, the client can distinguish between trusted content created by the web application and untrusted content provided by an attacker. To implement Noncespaces with minimal changes to web applications, we leverage a popular web application architecture to automatically apply Noncespaces to static content processed through a popular PHP template engine. We show that with simple policies Noncespaces thwarts popular XSS attack vectors.

In addition the UC Davis team has developed both static and dynamic detection techniques for injection vulnerabilities, such as SQL injection and cross-cite scripting. We have also presented the first pure client-side detection of XSS worms. In the most recent effort, we have developed the first static analysis to detect access control vulnerabilities in web applications. The core of the analysis is a novel approach to capture, in a general manner, application-specific access control policies. The prototype is practical and has detected known and new access control vulnerabilities in open-source applications.

DBTaint. Information flow tracking has been an effective approach for identifying malicious input and detecting software vulnerabilities. However, most current schemes can only track data within a single application. This single-application approach means that the program must consider data from other programs as either all tainted or all untainted, inevitably causing false positives or false negatives. These schemes are insufficient for most Web services because these services include multiple applications, such as a Web application and a database application. Although system-wide information flow tracking is available, these approaches are expensive and overkill for tracking data between Web applications and databases because they fail to take advantage of database semantics. We have designed DBTaint, which provides information flow tracking in databases to enable cross-application information flow tracking. In DBTaint, we extend database datatypes to maintain and propagate taint bits on each value. We integrate Web application and database taint tracking engines by modifying the database interface, providing cross-application information flow tracking transparently to the Web application. We present two prototype implementations for Perl and Java Web services, and evaluate their effectiveness on two real-world Web applications, an enterprise-grade application written in Perl and a robust forum application written in Java. By taking advantage of the semantics of database operations, DBTaint has low overhead: our unoptimized prototype incurs less than 15% overhead in our benchmarks.

AndroidLeaks: As mobile devices become more widespread and powerful, they store more sensitive data, which includes not only users' personal information but also the data collected via sensors throughout the day. When mobile applications have access to this growing amount of sensitive information, they may leak it carelessly or maliciously.

Google's Android operating system provides a permissions-based security model that restricts an application's access to the user's private data. Each application statically declares the sensitive data and functionality that it requires in a manifest, which is presented to the user upon installation. However, it is not clear to the user how sensitive data is used once the application is installed. To combat this problem, we present AndroidLeaks, a static analysis framework for automatically finding potential leaks of sensitive information in Android applications on a massive scale. AndroidLeaks drastically reduces the number of applications and the number of traces that a security auditor has to verify manually.

We evaluate the efficacy of AndroidLeaks on 24,350 Android applications from several Android markets. AndroidLeaks found 57,299 potential privacy leaks in 7,414 Android applications, out of which we have manually verified that 2,342 applications leak private data including phone information, GPS location, WiFi data, and audio recorded with the microphone. AndroidLeaks examined these applications in 30 hours, which indicates that it is capable of scaling to the increasingly large set of available applications.

DNADroid: Detection of plagiarized Android applications. DNADroid detects Android application cloning by robustly computing the similarity between two applications. DNADroid achieves this by comparing program dependency graphs between methods in candidate applications. Using DNADroid, we found at least 141 applications that have been the victims of cloning, some as many as seven times. We present several case studies that give insight into why applications are cloned, including localization and redirecting ad revenue. We describe a case of malware being added to an application and show how DNADroid was able to detect two variants of the same malware.

OMash: Enabling Secure Web Mashups via Object Abstractions. The current security model used by web browsers, the Same Origin Policy (SOP), does not support secure cross-domain communication desired by web mashup developers. The developers have to choose between no trust, where no communication is allowed, and full trust, where third-party content runs with the full privilege of the integrator. Furthermore, the SOP has its own set of security vulnerabilities and pitfalls, including Cross-Site Request Forgery, DNS rebinding and dynamic pharming. To overcome the unfortunate tradeoff between security and functionality forced upon today's mashup developers, we propose OMash, a simple abstraction that treats web pages as objects and allows objects to communicate only via their declared public interfaces. Since OMash does not rely on the SOP for controlling DOM access or cross-domain data exchange, it does not suffer from the SOP's vulnerabilities. We show that OMash satisfies the trust relationships desired by mashup authors and may be configured to be backward compatible with the SOP. We implemented a prototype of OMash using Mozilla Firefox 2.0 and demonstrated several proof-of-concept applications.

3.5.4 Publications

Attack of the Clones: Detecting Cloned Applications on Android Markets. Jonathan Crussell, Clint Gibler, and Hao Chen. 17th European Symposium on Research in Computer Security (ESORICS), Pisa, Italy, September 10-12, 2012.

AndroidLeaks: Automatically Detecting Potential Privacy Leaks In Android Applications on a Large Scale. Clint Gibler, Jon Crussell, Jeremy Erickson, and Hao Chen. 5th International Conference on Trust & Trustworthy Computing (TRUST), Vienna, Austria, June 13-15, 2012.

I-ARM-Droid: A Rewriting Framework for In-App Reference Monitors for Android Applications. Benjamin Davis, Ben Sanders, Armen Khodaverdian, and Hao Chen. IEEE Mobile Security Technologies (MoST), San Francisco, CA, May 24, 2012.

Static Detection of Access Control Vulnerabilities in Web Applications. Fangqi Sun, Liang Xu, and Zhendong Su. In Proceedings of USENIX Security 2011, San Francisco, CA, August 8-12, 2011.

DBTaint: Cross-Application Information Flow Tracking via Databases. Benjamin Davis and Hao Chen. USENIX Conference on Web Application, Boston, MA, June 23-24, 2010.

Automatic Detection of Unsafe Component Loadings. Taeho Kwon and Zhendong Su. In Proceedings of ISSTA 2010, Trento, Italy, July 12-16, 2010.

Client-Side Detection of XSS Worms by Monitoring Payload Propagation. Fangqi Sun, Liang Xu, and Zhendong Su. ESORICS 2009, Saint Malo, France, September 21-23, 2009.

Noncespaces: Using Randomization to Enforce Information Flow Tracking and Thwart Cross-Site Scripting Attacks. Matthew Van Gundy and Hao Chen. 16th Annual Network and Distributed System Security Symposium (NDSS), San Diego, CA, February 8-11, 2009.

OMash: Enabling Secure Web Mashups via Object Abstractions. Steven Crites, Francis Hsu, and Hao Chen. 15th ACM Conference on Computer and Communications Security (CCS), Alexandria, VA, October 27-31, 2008.

Dynamic Test Input Generation for Web Applications. Gary Wassermann, Dachuan Yu, Ajay Chander, Dinakar Dhurjati, Hiroshi Inamura, and Zhendong Su. In Proceedings of ISSTA 2008, Seattle, WA, July 20-24, 2008.

Static Detection of Cross-Site Scripting Vulnerabilities. Gary Wassermann and Zhendong Su. In Proceedings of ICSE 2008, Leipzig, Germany, May 10-18, 2008.

3.6 Scalable Distributed Algorithms

Sensor alerts, repair and recovery actions need to be propagated throughout the Helix architecture using scalable, robust, and attack-resilient protocols. In this way, knowledge obtained or inferred in one part of the system will benefit the system as a whole. This work was led by Dr. Saia and his group at the University of New Mexico.

3.6.1 Capabilities and accomplishments

- Developed and proved correct the first algorithm to solve the Byzantine agreement problem in a scalable manner
- Developed and proved correct an efficient algorithm to solve the Secure Multiparty Computation (SMPC) problem

3.6.2 Noteworthy events

- Best paper award ($2,000) at the ACM Conference on Principles of Distributed Computing (PODC), 2010 for our paper "Breaking the O(n^2) Bit Barrier: Scalable Byzantine agreement with an Adaptive Adversary''
- Paper "Breaking the O(n^2) Bit Barrier" was an invited submission to the Journal of the ACM and was discussed in the the SIGACT Distributed Computing Column, 2010
- Invited talks on MURI research at the University of Maryland, University of Iowa, Sandia Labs and the Santa Fe Institute Summer School
- Invited submission to "Transactions on Algorithms" of best papers of Symposium on Distributed Algorithms (SODA) 2008 for the paper "Fast Asynchronous Byzantine Agreement and Leader Election with Full Information".
- Press Coverage: ``Professor Fights a Mathematical Battle to Keep the Virtual World Running Smoothly", ACM Technical News, 2/26/07 and UNM Today, 2/27/07.``Professor goes to war," Front page lead article in University of New Mexico Daily Lobo, 3/2/07

3.6.3 Research Summary

We described and proved correct the first algorithm to solve the Byzantine agreement problem in a scalable manner. The Byzantine agreement problem allows a network to compute reliably even when up to a 1/3 fraction of the nodes in the network are controlled by an adversary. Our algorithm is scalable in the sense that in each node in the network is required to send a number of bits that is square root of the total number of nodes in the network.

We described and proved correct an efficient algorithm to solve the Secure Multiparty Computation (SMPC) problem. In the SMPC problem, there are n players, each with a private input. The goal is to securely compute an n-ary function, f, over all inputs, without revealing anything more about the inputs than can be learned from the output of the function. The problem is complicated by the fact that up to a 1/3 fraction of the nodes in the network are controlled by an adversary. Our algorithm has bandwidth cost that improves by a factor of n^2 over previous state of the art.

We designed, implemented and simulated algorithms to solve Byzantine agreement, in the case where a random beacon is available. A random beacon is a stream of random bits, known to all the processors. In networks containing 1 million processors, our algorithm requires each processor to send at most 32,000 bits. In contrast, previous Byzantine agreement algorithms would require each processor to send 4.3 billion bits in such a situation.

We considered a game theoretic setting where players must cooperate to ensure that a message is transmitted over a channel, even when an adversary is able to block the channel. We describe an algorithm that, in the situation where the adversary spends B computational resources trying to block the message, for some unknown value B, the remaining players need spend only proportional to B^.62 computational resources in order to ensure transmittal of the message.

We considered a security game between an algorithm and an adversary over a dynamic network. In each round, the adversary removes nodes and the algorithm adds edges. We design an algorithm for such a game that ensures 1) the network always stays connected; 2) distances between pairs of nodes do not increase by much; 3) the degree of any node does not decrease by much; and 4) the distributed algorithm for self-healing is efficient in time and computational resources.

3.6.4 Publications

"Scalable Byzantine agreement with a Random Beacon" by Olumuyiwa Oluwasanmi and Jared Saia, International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS) , 2012.

"Breaking the O(nm) Bit Barrier: Secure Multiparty Computation with a Static Adversary'' by Varsha Dani, Valerie King, Mahnush Mohavedi and Jared Saia, Principles of Distributed Computing (PODC), 2012.

"Breaking the O(n^2) Bit Barrier: Scalable Byzantine agreement with an Adaptive Adversary'' by Valerie King and Jared Saia, Principles of Distributed Computing (PODC) and Journal of the ACM(JACM), 2011. Best Paper Winner.

“Conflict on a Communication Channel” by Valerie King, Jared Saia and Maxwell Young Published, Principles of Distributed Computing (PODC), 2011.

"Load balanced Scalable Byzantine Agreement through Quorum Building, with Full Information'' by Valerie King, Steve Lonargan, Jared Saia and Amitabh Trehan, International Conference on Distributed Computing and Networking (ICDCN), 2010.

"Attack-Resistant Frequency Counting'' by Bo Wu, Valerie King and Jared Saia, IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2010.

"An Empirical Study of a Scalable Byzantine Agreement Algorithm'' by Olumuyiwa Oluwasanmi, Valerie King and Jared Saia Heterogeneity in Computing Workshop (HWC), 2010.

"Fast, scalable Byzantine agreement in the full information model with a Nonadaptive adversary" by Valerie King and Jared Saia, International Symposium on Distributed Computing (DISC), 2009.

"The Forgiving Graph: A Distributed Data Structure for Low Stretch under Adversarial Attack" by Tom Hayes, Jared Saia and Amitabh Trehan, Principles of Distributed Computing (PODC), 2009.

"The Forgiving Tree: A Self-Healing Distributed Data Structure" by Tom Hayes, Navin Rustagi, Jared Saia and Amitabh Trehan, Principles of Distributed Computing (PODC), 2008.

"Fast Asynchronous Byzantine Agreement and Leader Election with Full Information" by Bruce Kapron, David Kempe, Valerie King, Jared Saia and Vishal Sanwalani, Symposium on Discrete Algorithms (SODA), 2008. Invited submission to "Transactions on Algorithms" best papers of SODA 2008 .

"Reducing Communication Costs in Robust Peer-to-Peer Networks" by Jared Saia and Maxwell Young, Information Processing Letters (IPL), 2008.

3.7 Hardware/Software Co-design for Security

Research for this thrust was performed at UC Santa Barbara under the leadership of Dr. Frederic Chong. First, we have developed hardware design and architecture techniques to provide a provably secure foundation for the Helix system. Second, we have developed techniques that can dramatically reduce the performance and energy overhead of software diversity techniques.

3.7.1 Capabilities and accomplishments

- A design methodology for probably-secure hardware with guaranteed information flow properties to provide a root of trust for computer systems.
- A hardware-description language and compiler that allows straightforward design of secure hardware and synthesis to chip implementation.
- An example secure processor design and with components of system software running on the processor.
- A memory system design that significantly reduces the cost of software diversity.
- A processor design that significantly reduces the cost of software diversity.
- An information-flow tracking system that leverages commodity Itanium processors.
- A virtualization and replay system that allows recovery from control-flow attacks.

3.7.2 Noteworthy events

- Chong co-chaired the National Cyber-Leap Year computer security summit organized by NITRD for the Obama administration in 2009.
- Our work was selected as an IEEE Micro Top Pick in 2010, in which IEEE Micro selects the most influential publications from the top architecture conferences the previous year.

3.7.3 Research Summary

Although system software can be designed to increase resilience and security in a system, many vulnerabilities can be inherent in conventional hardware. Implicit information flows in common architectural mechanisms -- such as caches, branch predictors, and even the program counter -- can lead to violations in policy. Our approach is to design provably-secure systems from the most fundamental level -- from the gates up. While this work has focused on dynamic hardware enforcement techniques at the gate level, new work explores the design of a hardware description language and static analysis techniques to generate hardware according to specified information-flow policies. This static approach is somewhat more restrictive than the dynamic approach, but can dramatically reduce performance and area overheads. We also use a hybrid approach that uses symbolic analysis to verify flow policies in conventional systems at design time.

Helix techniques can be made more effective with hardware support to reduce overheads and increase performance. We have explored multicore caching techniques that can reduce data redundancy, especially in scenarios involving software diversity. We have also recently completed a study of a novel multi-threaded processor architecture that can substantially reduce instruction redundancy in scenarios involving software diversity.

Recently, we designed a hardware description language that uses type-checking to enforce information-flow properties. The compiler for this language is available here: https://github.com/vineethk/Caisson

We have also designed a full system with a simple separation kernel using our methodology. We have also developed a hardware prototype to be synthesized on our Convey FPGA infrastructure purchased under our DURIP.

3.7.4 Publications

"Bezoar: Automated Virtual Machine-based Full-System Recovery from Control-Flow Hijacking Attacks," D. A. S. de Oliveira, J. Crandall, G. Wasserman, S. Ye, Z. Su, F. Wu, and F. T. Chong. 11th IEEE/IFIP Network Operations and Management Symposium. 2008.

"From Speculation to Security: Practical and Efficient Information Flow Tracking Using Speculative Hardware," H. Chen, X. Wu, L. Yuan, B. Zhang, P. Yew and F.T. Chong. International Symposium on Computer Architecture. 2008

"Preliminary Experiments on Similar Executions with Reduced Off-Chip Accesses in Multi-core Processors,"

S. Biswas, F. T. Chong, D. Franklin and T. Sherwood. Workshop on Parallel Execution of Sequential Programs on Multi-core Architectures. 2008.

“Putting Trojans on the Horns of a Dilemma: Redundancy for Information Theft Detection,” J. Crandall, J. Brevik, G. Wasserman, D. A. S. de Oliviera, Z. Su, S. F. Wu, and F. T. Chong. Transactions on Computational Sciences: Special Issue on Security. 2009

“Execution Leases: A Hardware-Supported Mechanism for Enforcing Strong Non-Interference,” M. Tiwari, X. Li, H. Wassel, F. T. Chong, and T. Sherwood. The International Symposium On Microarchitecture. 2009

“Multi-Execution: Multicore Caching for Data-Similar Executions,” S. Biswas, D. Franklin, A. Savage, T. Sherwood, and F. T. Chong. International Symposium on Computer Architecture. 2009.

“Conflict-Avoidance in Multicore Caching for Data-Similar Executions,” Susmit Biswas, Diana Franklin, Timothy Sherwood, Frederic T. Chong. International Symposium on Pervasive Systems, Algorithms, and Networks (I-SPAN 2009)

“Function Flattening for Lease-Based, Information-Leak-Free Systems,” X. Li, M. Tiwari, T. Sherwood, and F. T. Chong. 21st IEEE International Conference on Application-specific Systems, Architectures and Processors. 2010.

“Minimal Multi-Threading: Finding and Removing Redundant Instructions in Multi-Threaded Processors,” Guoping Long, Diana Franklin, Susmit Biswas, Pablo Ortiz, Jason Oberg, Dongrui Fan, and Frederic T. Chong . International Symposium On Microarchitecture. 2010.

“Caisson: a Hardware Description Language for Secure Information Flow,” Xun Li, Mohit Tiwari, Jason Oberg, Frederic T. Chong, Tim Sherwood, and Ben Hardekopf. ACM Conference on Programming Language Design and Implementation. 2011.

“Crafting a Usable Microkernel, Processor, and I/O System with Strict and Provable Information Flow Security,” Mohit Tiwari, Jason Oberg, Xun Li, Jonathan Valamehr, Timothy Levin, Ben Hardekopf, Ryan Kastner, Frederic T. Chong, and Tim Sherwood. International Symposium on Computer Architecture. 2011.