Qingkai Shi

ADDRESS

Department of Computer and Science
Purdue University
305 N. University Street
West Lafayette, IN 47907
Email: shi553@purdue.edu
URL: https://qingkaishi.github.io/
ORCID: 0000-0002-8297-8998

EDUCATION & EMPLOYMENT

Purdue University 2021 - Now
Postdoctoral Research Associate

Hong Kong University of Science and Technology & Sourcebrella Inc. 2015 - 2021
Ph.D. in Computer Science (2020) & Co-founder of Sourcebrella Inc. (2021)
Thesis: Precise and Scalable Static Bug Finding for Industrial-Sized Code

RESEARCH & AWARDS

My research focuses on the use of programming language and compiler techniques, especially static analysis, for rigorously ensuring software security, including (1) static analysis for reverse engineering, (2) static analysis for vulnerability scanning, (3) static analysis for fuzz testing, and (4) static analysis for theorem proving. My research has been extensively published in premium venues and won me an ACM SIGSOFT Distinguished Paper Award and a Hong Kong PhD Fellowship.

My research on vulnerability scanning, known as the *Pinpoint Static Analyzer*, has received 200M CNY ($\approx$ 30M USD) investment. The startup that commercializes my research, Sourcebrella Inc., has been acquired by Ant Group, where *Pinpoint* is deployed in daily operations for improving the quality of Alipay, a popular digital payment app with over a billion monthly active users. To date, we have discovered hundreds of vulnerabilities and CVEs in mature software.

For further information, see https://qingkaishi.github.io/.

REFEREED PUBLICATIONS

My research has been published extensively at premium venues of programming language (PLDI, OOPSLA), software engineering (ICSE, FSE), and cybersecurity (S&P).

[1] Chengpeng Wang, Peisen Yao, Wensheng Tang, **Qingkai Shi**, and Charles Zhang. Complexity Guided Container Replacement Synthesis. In *Proceedings of the ACM on Programming Languages (OOPSLA'22)*. ACM, 2022.

[2] Yiyuan Guo, Jinguo Zhou, Peisen Yao, **Qingkai Shi**, and Charles Zhang. Precise Divide-By-Zero Detection with Affirmative Evidence. In *Proceedings of the 44nd ACM/IEEE International Conference on Software Engineering (ICSE'22)*. ACM, 2022.

[3] Heqing Huang, Yiyuan Guo, **Qingkai Shi**, Peisen Yao, Rongxin Wu, and Charles Zhang. Beacon: Directed Grey-Box Fuzzing with Provable Path Pruning. In *Proceedings of the 43rd IEEE Symposium on Security and Privacy (S&P'22)*. IEEE, 2022.

[4] **Qingkai Shi**, Peisen Yao, Rongxin Wu, and Charles Zhang. Path-Sensitive Sparse Analysis without Path Conditions. In *Proceedings of the 42nd ACM SIGPLAN Con-*

ference on Programming Language Design and Implementation (PLDI'21). ACM, 2021.

[5] Peisen Yao, **Qingkai Shi**, Heqing Huang, and Charles Zhang. Program Analysis via Efficient Symbolic Abstraction. In *Proceedings of the ACM on Programming Languages (OOPSLA'21)*. ACM, 2021.

[6] Peisen Yao, Heqing Huang, Wensheng Tang, **Qingkai Shi**, Rongxin Wu, and Charles Zhang. Skeletal Approximation Enumeration for SMT Solver Testing. In *Proceedings of the 29th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE'21)*. ACM, 2021.

[7] Peisen Yao, Heqing Huang, Wensheng Tang, **Qingkai Shi**, Rongxin Wu, and Charles Zhang. Fuzzing SMT Solvers via Two-Dimensional Input Space Exploration. In *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA'21)*. ACM, 2021.

[8] **Qingkai Shi** and Charles Zhang. Pipelining Bottom-up Data Flow Analysis. In *Proceedings of the 42nd ACM/IEEE International Conference on Software Engineering (ICSE'20)*. ACM, 2020.

[9] **Qingkai Shi**, Rongxin Wu, Gang Fan, and Charles Zhang. Conquering the Extensional Scalability Problem for Value-Flow Analysis Frameworks. In *Proceedings of the 42nd ACM/IEEE International Conference on Software Engineering (ICSE'20)*. ACM, 2020.

[10] Heqing Huang, Peisen Yao, Rongxin Wu, **Qingkai Shi**, and Charles Zhang. Pangolin: Incremental Hybrid Fuzzing with Polyhedral Path Abstraction. In *Proceedings of the 41st IEEE Symposium on Security and Privacy (S&P'20)*. IEEE, 2020.

[11] Peisen Yao, **Qingkai Shi**, Heqing Huang, and Charles Zhang. Fast Bit-Vector Satisfiability. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA'20)*. ACM, 2020.

[12] Gang Fan, Chengpeng Wang, Rongxin Wu, Xiao Xiao, **Qingkai Shi**, and Charles Zhang. Escaping Dependency Hell: Finding Build Dependency Errors with the Unified Dependency Graph. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA'20)*. ACM, 2020.

[13] Chunrong Fang, Zixi Liu, Yangyang Shi, Jeff Huang, and **Qingkai Shi**. Functional Code Clone Detection with Syntax and Semantics Fusion Learning. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA'20)*. ACM, 2020.

[14] Yang Feng, **Qingkai Shi**, Xinyu Gao, Jun Wan, Chunrong Fang, and Zhenyu Chen. DeepGini: Prioritizing Massive Tests to Enhance the Robustness of Deep Neural Networks. In *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA'20)*. ACM, 2020.

[15] Gang Fan, Rongxin Wu, **Qingkai Shi**, Xiao Xiao, Jinguo Zhou, and Charles Zhang. SMOKE: Scalable Path-Sensitive Memory Leak Detection for Millions of Lines of Code. In *Proceedings of the 41st ACM/IEEE International Conference on Software Engineering (ICSE'19)*. IEEE, 2019. **ACM SIGSOFT Distinguished Paper Award**

[16] **Qingkai Shi**, Xiao Xiao, Rongxin Wu, Jinguo Zhou, Gang Fan, and Charles Zhang. Pinpoint: Fast and Precise Sparse Value Flow Analysis for Million Lines of Code. In *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'18)*. ACM, 2018.

[17] **Qingkai Shi**, Zhenyu Chen, Chunrong Fang, Yang Feng, and Baowen Xu. Measuring the Diversity of a Test Set with Distance Entropy. In *IEEE Transactions on Reliability (TR'16)*, Vol. 65, No. 1. IEEE, 2016.

[18] **Qingkai Shi**, Jeff Huang, Zhenyu Chen, and Baowen Xu. Verifying Synchronization for Atomicity Violation Fixing. In *IEEE Transactions on Software Engineering (TSE'16)*, Vol. 42, No. 3. IEEE, 2016.

PATENTS

Defect detection method, device. system, and computer readable medium.

- US Patent No. 20190108003
- China Patent No. 201811013103, 201811013000, 201811015751, 2018110146864

INVITED TALKS

Fast and Precise Static Bug Detection for Industrial-Sized Code. Texas A&M University, College Station, United States, July 2021.

Taming Path-Sensitivity for Static Code Analysis on Industrial Scale. Nanjing University, Nanjing, China, April 2021.

The Next Generation Software Security Analysis. Nanjing University, Nanjing, China, May 2020.

Marrying Taint Analysis with Big Data Analytics. Southern University of Science and Technology, Shenzhen, China, November 2018.

PRESENTATIONS

Path-Sensitive Sparse Analysis without Path Conditions. In *the 42nd ACM SIGPLAN conference on Programming Language Design and Implementation (PLDI'21)*, Virtual, Canada, June 2021.

Pipelining Bottom-up Data Flow Analysis. In *the 42nd ACM/IEEE International Conference on Software Engineering (ICSE'20)*, Seoul, South Korea, May 2020.

Conquering the Extensional Scalability Problem for Value-Flow Analysis Frameworks. In *the 42nd ACM/IEEE International Conference on Software Engineering (ICSE'20)*, Seoul, South Korea, May 2020.

Fast and Precise Sparse Value Flow Analysis for Million Lines of Code. In *the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'18)*, Philadelphia, United States, June 2018.

PROFESSIONAL SERVICES

Reviewer, *IEEE Transactions on Dependable and Secure Computing*

Reviewer, *IEEE Transactions on Reliability*

Member of Artifact Evaluation Committee, *the 49th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'22)*

TEACHING EXPERIENCE

Teaching Assistant, COMP3111/3111H: Software Engineering (Fall 2018)

Teaching Assistant, COMP4111: Software Engineering Practices (Spring 2018)

Teaching Assistant, COMP4111: Software Engineering Practices (Fall 2016)