Open-Source Tools by Yulei Sui

SVF (`https://github.com/SVF-tools/SVF`) has served as a foundational framework for developing other program analyses and has also been used for online learning of software analysis and verification (`https://github.com/SVF-tools/SVF-Teaching`. On its GitHub website, SVF has 900+ stars and 300+ forked repositories with different users and contributors from both academia and industry, including Cambridge University, UIUC, University of Wisconsin-Madison, Google, IBM, Veracode, Pathscale and ALLVM Research.

SVF has been used and cited by researchers from leading program analysis and security groups, e.g. Chopped Symbolic Execution (from Imperial College London@ICSE'18 and @FSE'19), PinPoint (from HKUST@PLDI'18), Type-based CFI (from ACSAC'18@MIT and Northeastern University), Kernel Fuzzing (from Purdue@IEEE S&P'18), Directed Fuzzer (from NTU@CCS'18), K-Miner (from TU Darmstadt@NDSS'18), Permission Check Analysis (from Virginia Tech & Zhejiang University @USENIX Security'19), probabilistic analysis (from University of Pennsylvania @PLDI'19), and Hybrid Testing (from Northeastern University @S&P'20), and system call specialization (from Northeastern University @USENIX Security'20), and hot patch generation for kernels (from NTU @USENIX Security'20), and fuzzing for kernel file system (from Georgia Institute of Technology @S&P'20).

The following lists some comments (extracted from emails and GitHub websites; only their organisations are shown with names removed for anonymity) made on Dr Sui's open-source tools by people from both academia and industry:

- *"Looking for llvm pointer analysis I stumbled over SVF and was quite impressed by it's capabilities - thanks a lot for making it open source!"*

  — A researcher working in the Cloud and Storage Security Group at IBM Research.

- *"Your tests have been most helpful! ... I'm trying to convince Arthur that SVF can solve some (all?) of the design requirement laid out by Chandler (Google)."*

  — CTO at Pathscale, HPC Compiler Company.

- *"First, I'd like to thank you for your work on SVF and for making it public! The academic LLVM community is in dire need of such a framework and I am very interested in seeing your framework become the foundation for many analyses moving forward. "*

  — Researcher at UIUC and ALLVM.

- *"I came across SVF recently and am really interested in using it. ... it's a really nice set of tools! Again thanks! "*

  —Sr. Security Researcher at Veracode, a leading company working in software application security.

- *" I remember watching your presentation on SVF and thought: that's cool, was so excited to be able to reach you. ... Would be great to be able to chat about how maybe you and our team might be able to work on interesting things together"*

– Software Engineer at Google.

- *‘I hope to use SVF for the DARPA TRACE grant application.‘*

  — A researcher from University of Wisconsin-Madison

- *“I’m currently working with your awesome SVF analysis.”*

  — A researcher from TU Darmstadt.

- *“I’ve run into your paper: Region-based Selective Flow-Sensitive Pointer Analysis. It’s a great work!”*

  — A researcher from Masaryk University.

- *“Thank you again and I think that the SVF will be promising for my work.”*

  — A researcher from Waterloo University.

- *“ We very much appreciate your work and would like to use the static pointer analysis you developed.”*

  — A researcher from Uppsala University.

- *“ I’m very interested in the SVF analysis that can be performed with your software.”*

  — A researcher from Ulm University.

- *“We heard a lot about SABER: Static Memory Leak Detection Using Full-Sparse Value-Flow Analysis. We would like to test our improvement ideas for SABER ”*

  — A researcher from Iowa State University.

- *“I recently tested SVF (commit 5355fc2). Great piece of work from my point of view! . . . Thank you for providing the code!”*

  — A researcher from at TU Ilmenau.

- *“ As far as I can tell, the tool fixes all issues I have with musl libc. Thank you.”*

  — A researcher from Binghamton University.

- *“Thanks for your great project. ”*

  — A researcher from Hong Kong University of Science and Technology.

- *“ Thanks for the quick answer:) I am looking at SVF implementation, in order to test one of our ideas. ”*

  — A researcher from Tel-Aviv University

- *“ Many thanks for this excellent codebase! I’m new to SVF but I have updated it for LLVM 5.0.0 for a project of mine. ”*

  — A researcher from Cambridge University, UK

- *“Awesome project! ”*

  — A researcher from Columbia University

- *“Thank you for sharing and maintaining such a great project! ”’*

  — A researcher from Northwestern University

For teaching and education, SVF not only have been used inside UTS but also worldwide. For example, SVF has been recognised as a representative pointer analysis approach in Prof. Phillip B. Gibbons’s ‘Optimizing Compilers for Modern Architectures‘ class at Carnegie Mellon University (`https://www.cs.cmu.edu/afs/cs/academic/class/15745-s19/www/lectures/L13-Pointer-Analysis-pre-class.pdf`). SVF has also been used as a representative static analysis approach in Prof. Qirun Zhang’s ‘Topics in Program Analysis‘ class at Georgia Institute of Technology (`https://helloqirun.github.io/course/cs8803/index.html`). SVF has also been referred as a standard alias analysis framework in Prof. Simone Campanoni’s ‘Advanced Topics in Compilers‘ class at Northwestern University USA `https://users.cs.northwestern.edu/~simonec/files/Teaching/ATC/slides/NOELLE_dependences.pdf`