Lightweight Formal Methods and Beyond

David Evans (Professor of Computer Science, University of Virginia)
SM/PhD advisee of John Guttag (1993-1999)

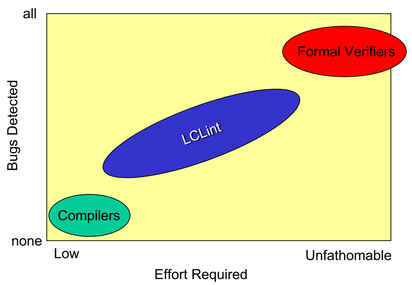
Introduction to Formal Methods

MIT Laboratory for Computer Science

In 1993, I was a fourth-year undergraduate at MIT and took a seminar on Formal Method that John co-taught with Jeannette Wing (JG PhD 1983) and learned the beauty and power of proving properties of software.

Since getting a simple traffic light example to work through the automatic theorem proving tools took me two days, I wanted to work on something easier and more scalable, and connected with a vision John had for incorporating specifications with code and lightweight static analyses.

LCLint: Annotation-Assisted Static Checking



Relaxing requirements for soundness and completeness made it possible (even in the 1990s) for formal methods to be *useful*.

This resulted in an open-source tool, that I still occasionally get bug reports for.

David Evans, John Guttag, Jim Horning and Yang Meng Tan. *LCLint: A Tool for Using Specifications to Check Code*. SIGSOFT Symposium on the Foundations of Software Engineering, December 1994.

Annotating and statically checking ownership and properties of dynamically-allocated memory could detect many programming errors (and 20+ years later, has become widespread through Rust programming language.

David Evans. *Static Detection of Dynamic Memory Errors*. In SIGPLAN Conference on Programming Language Design and Implementation (PLDI), May 1996.

Allocation

`only` Refers to unshared storage; confers obligation to release this storage or transfer the obligation.

`keep` Like only, except that the caller may safely use the reference after the call. (Function parameters only.)

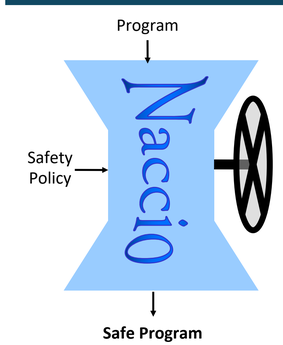
`temp` Temporary storage. Function may not deallocate or add new external references to storage. (Function parameters only.)

`owned` Refers to storage that may be shared by `dependent` references. This reference is responsible for releasing the storage.

`dependent` Refers to storage that may be shared by an `owned` reference. This reference may not release the storage.

`shared` Refers to arbitrarily shared storage; may not be deallocated. (For use with garbage collectors.)

PhD Dissertation: Policy-Directed Code Safety




```
policy LimitWrite
    NoOverwrite, LimitBytesWritten (1000000)

property LimitBytesWritten (n: int)
   requires TrackBytesWritten;
   check RFileSystem.write (file: RFile, nbytes: int)
     if (bytes_written > n)
       violation ("Writing more than " + n + " bytes.");
```

A rich language for describing safety policies and efficient implementation by rewriting programs. With John's help, this got me my dream job at UVA, but no success getting industry adoption and little progress towards better safety policies.

David Evans and Andrew Twyman. *Policy-Directed Code Safety*. In *IEEE Symposium on Security and Privacy* (Oakland), May 1999.

Formal Methods at UVA (2000-2015)

Splint
Annotation-Assisted Lightweight Static Checking

Secure Programming Lint
SPecifications Lint
First Aid for Programmers

With my first student at UVA, LCLint evolved into a tool focused on detecting security vulnerabilities.

```
char *strcpy (char *s1, const char *s2)
 /*@requires maxSet(s1) >= maxRead(s2)@*/
 /*@ensures maxRead(s1) == maxRead(s2)
     /\ result == s1@*/;
```

David Larochelle and David Evans. *Statically Detecting Likely Buffer Overflow Vulnerabilities*. In USENIX Security Symposium, 2001.

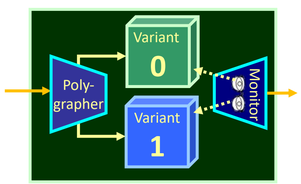
Automatically Inferring Specifications

Writing specifications is hard – we need tools to automatically infer likely specifications from program executions (and they need to be resilient to incorrect programs to infer intended specifications).

Jinlin Yang, David Evans, Deepali Bhardwaj, Thirumalesh Bhat, Manuvir Das. *Perracotta: Mining Temporal API Rules from Imperfect Traces*. In International Conference on Software Engineering (ICSE) 2006.

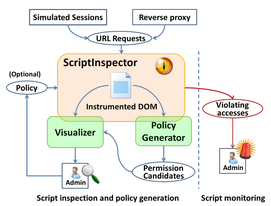
Formal Methods for Security

N-Variant Systems



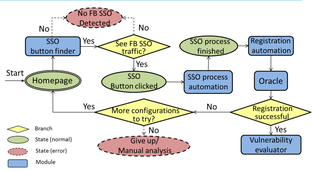
Benjamin Cox, David Evans, Adrian Filipi, Jonathan Rowanhill, Wei Hu, Jack Davidson, John Knight, Anh Nguyen-Tuong, and Jason Hiser. *N-Variant Systems: A Secretless Framework for Security through Diversity*. USENIX Security Symposium, 2006.

ScriptInspector



Yuchen Zhou and David Evans. *Understanding and Monitoring Embedded Web Scripts*. IEEE Security and Privacy ("Oakland"). 2015.

SSOScan

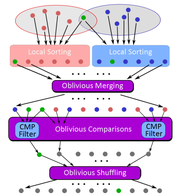


Yuchen Zhou and David Evans. *SSOScan: Automated Testing of Web Applications for Single Sign-On Vulnerabilities*. In USENIX Security Symposium, 2014.

IEEE Security and Privacy Magazine, May/June 2011. Co-edited by David Evans and Sal Stolfo

Thanks John!

Secure Computation (2009-2019)

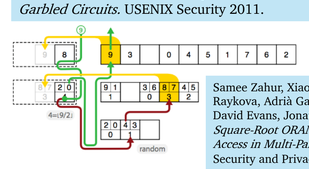


Yan Huang, David Evans, Jonathan Katz, and Lior Malka. *Faster Secure Two-Party Computation Using Garbled Circuits*. USENIX Security 2011.

Obliv-C

$$c = a \wedge b$$
$$= a \wedge (r \oplus r \oplus b)$$
$$= (a \wedge r) \oplus (a \wedge (r \oplus b))$$

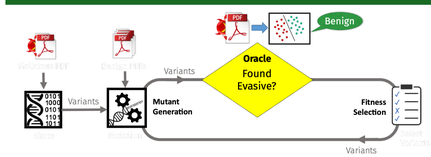
Samee Zahur, Mike Rosulek, and David Evans. *Two Halves Make a Whole: Reducing Data Transfer in Garbled Circuits using Half Gates*. EuroCrypt 2015.



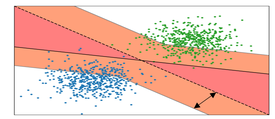
Samee Zahur, Xiao Wang, Mariana Raykova, Adrià Gascón, Jack Doerner, David Evans, Jonathan Katz. *Revisiting Square-Root ORAM Efficient Random Access in Multi-Party Computation*. IEEE Security and Privacy ("Oakland") 2016.

securecomputation.org

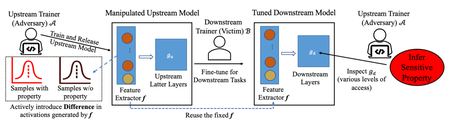
Adversarial Machine Learning (since 2015)



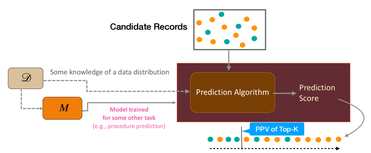
Weilin Xu, Yanjun Qi, and David Evans. *Automatically Evading Classifiers: A Case Study on PDF Malware Classifiers*. NDSS 2016.



Saeed Mahloujifar, Xiao Zhang, Mohammad Mahmoody, and David Evans. *Empirically Measuring Concentration: Fundamental Limits on Intrinsic Robustness*. NeurIPS 2019.



Yulong Tian, Fnu Suya, Anshuman Suri, Fengyuan Xu, David Evans. *Manipulating Transfer Learning for Property Inference*. CVPR 2023.



Bargav Jayaraman and David Evans. *Are Attribute Inference Attacks Just Imputation?*. In ACM CCS 2022.

Teaching

Some of John's Academic Grandchildren

Jinlin Yang (2007)
Nathanael Paul (2008)
Yan Huang (2012)
Yuchen Zhou (2015)
Samee Zahur (2016)
Weilin Xu (2019)

Bargav Jayaraman (2022)
Xiao Zhang (2022)
Fnu Suya (2023)
Josephine Lamp (2024)
Anshuman Suri (2024)