Research Statement

Avik Chaudhuri

http://www.cs.umd.edu/~avik

My research aims to bring principles and techniques from programming languages to the design and implementation of secure and correct systems. To this end, I conduct research that spans the areas of programming languages, security, and verification, while maintaining a strong relevance to systems. In the past, I have studied systems as diverse as operating systems, file systems, web-application frameworks, and mobile-phone platforms. I have developed new principles and techniques—strongly founded in type theory, logic, and abstract interpretation—to analyze such systems for security and correctness. I believe that similar foundations can (and should) guide the construction of secure and correct systems from scratch.

To me, research is often a bridge to an ambitious goal—a bridge that needs to be crossed in steps. I tend to work on projects that try to bring cutting-edge research ideas to mainstream practice. This is seldom a trivial task; various practical issues can (and often do) expose inadequacies in conventional research ideas. But these inadequacies usually provide new research opportunities, and solving such problems usually have the potential for significant impact. In the past few years, I have organized and collaborated on various such research efforts, which are discussed below.

Dissertation Work: Foundations of Access Control for Systems Security

Security in systems often relies on access control, although access control seldom guarantees security per se. For instance, access control rarely eliminates unintended information flow. Furthermore, implementations of access control in modern operating systems and file systems are inherently complex, due to practical design requirements such as distributability, adaptability, and efficiency. As such, the implementations do not always have their expected properties. In my dissertation work, I studied the formal implications of access control for security in a variety of modern systems such as Windows Vista [22], Asbestos [18], OSDFS [21], Plutus [24], and PCFS [20]. Technically, these studies contributed several techniques to find and eliminate security holes in the design of such systems. More generally, these studies helped bring to secure system design some of the same formal understanding that we already enjoy in the field of secure communication, thanks to decades of previous work.

For example, during an internship at Microsoft Research, I initiated a research program with the aim of explaining the somewhat unconventional security designs of Windows Vista and Asbestos. While these designs were based on well-studied models of access control with security levels, they deviated from those models in some radical ways, to allow communication across security levels, and to allow dynamic control of security levels. The resulting designs admitted various information-flow attacks, some of which were expected and some of which were not. I showed how query evaluation in a decidable, dynamic logic-programming language can be used to verify information-flow consequences of such designs efficiently [17]. In particular, this technique led to the discovery of an interesting formal property of Windows Vista's security design: that any possible information-flow attack requires the participation of trusted code, and can therefore be eliminated by static analysis. Based on this insight, I developed a type system to enforce the required information-flow guarantees for code running on Windows Vista [16].

This strategy turned out to be successful for other systems as well. I showed how an existing automated verifier for the applied-π calculus [6] can be used to find (and fix) serious attacks on existing cryptographic file-system protocols [7]. The same technique also exposed unexpected security implications of some common optimizations in such protocols; for instance, I discovered that a particular optimization actually strengthens the resulting security guarantees. I also showed how powerful semantic concepts such as full abstraction and refinement can guide the derivation of secure distributed implementations of access control [9, 13, 12], on the way exposing various security weaknesses in existing networked storage protocols. These results highlighted the difficulties of implementing dynamic access control correctly, and stirred new interest in the verification of storage protocols (most notably inside the Ceph project [2]). Going further, I invented language-based techniques to verify dynamic security specifications in such systems, by showing how to combine static type checks with dynamic access checks in radically new ways [8, 14]. The key idea in these techniques was to exploit the failure of dynamic access checks to provide strong security guarantees; in contrast, previous techniques tried to guarantee the success of those checks. Recently, I applied some of these techniques to implement language support for systems that rely on proof-carrying authorization [15].

Ongoing Work: Specification and Verification of Ruby Programs

Ruby [19] is a dynamically typed object-oriented scripting language, which is at the heart of Rails [25], a popular web-application framework. In ongoing work, I am developing techniques to analyze the correctness of Ruby scripts, which include web applications written in Rails. These techniques have the potential for significant impact due to the recent proliferation of web services developed in Rails, and the drastic consequences of possible web attacks on such services. Recently I showed how to apply static type inference to catch various run-time errors in Rails applications [4]. While such applications rely heavily on dynamic code generation and reflection—features that usually cause significant difficulties for typechecking—the key insight is that it is possible to translate such applications to Ruby scripts that do not rely on these features and are in fact amenable to typechecking. This translation also provides a reusable abstract semantics for Rails applications that is independent of the implementation of the framework itself. I am currently exploring a combination of static typechecking and symbolic analysis to verify finer correctness properties of Ruby scripts, including security properties that eliminate whole classes of attacks in Rails applications (such as cross-site scripting, code injection, cross-site request forgery, and session hijacking). The underlying idea is to encode such properties using a combination of types and executable object invariants, and prove that the types and invariants hold by symbolic-execution based inference.

Going further, an interesting feature of Ruby (and by extension, Rails) is its integrated support for testing during development. As future work, I plan to exploit such tests in proofs of correctness, by combining dynamic analysis with static analysis in non-trivial ways. For instance, previous work has shown how correctness properties can be verified via iterative over- and under-approximation, where testing can be used to strengthen invariants, and invariants can be used to direct testing [5].

Ongoing Work: Language-Based Security on Android

Android [1] is Google's new open-source platform for mobile phones, for which powerful applications can be developed in Java using APIs exposed by an SDK. By design, such applications can interact with other applications on the platform through well-defined interfaces. While such interactions can be controlled to some extent, there is currently no way to enforce end-to-end security guarantees on the platform, such as secrecy or integrity of sensitive data that is shared across applications. Recently I proposed support for certified installation of secure Android applications, following ideas based on proof-carrying code. To illustrate

this approach, I formalized an operational semantics and a security type system for an abstract idealization of the Android SDK [11], and implemented a prototype code verifier based on this system that provides the required guarantees. This preliminary work has already generated interest from other researchers in the community. Indeed, Android is widely regarded as a radical movement in the mobile phone industry, and in many ways this setting provides an ideal opportunity to bring language-based security to the mainstream. Furthermore, security certification of applications is expected to be very attractive financially, due to the huge market for bringing powerful services to mobile phones. Currently I am working towards an implementation of such a certifier using the Java static analysis framework WALA [3] as a front-end.

In the future, I plan to continue working on these and other research programs that are interesting both from scientific as well as engineering perspectives. In particular, I would like to aggressively push principles from programming languages to the design and analysis of new systems. The Singularity project [23] at Microsoft Research represents a similar effort. Conversely, I would like to explore new ideas and techniques in the design and implementation of programming languages. My recent work in the area of concurrent functional programming may be seen as indicative of this effort [10].

Following Dijkstra's "golden rule" for successful research, I always try to work as closely as possible to the boundary of my abilities. As such, I continuously evaluate my research, view it from new perspectives, and challenge myself to improve it where possible. I also maintain active interest in related research areas, from which I derive a rich supply of ideas and techniques to tackle new problems. By working at the edges between theory and practice, I hope to make unique and lasting contributions to both programming languages and systems of the future.

References

[1] Android project. `http://source.android.com/`.

[2] Ceph project. `http://www.ssrc.ucsc.edu/proj/petascale.html`.

[3] T. J. Watson Libraries for Analysis. `http://wala.sourceforge.net/wiki/index.php`.

[4] Jong-hoon (David) An, Avik Chaudhuri, and Jeffrey Foster. Static typing for Ruby on Rails. In *IEEE/ACM Conference on Automated Software Engineering (ASE'09)*. To appear, 2009.

[5] Nels E. Beckman, Aditya V. Nori, Sriram K. Rajamani, and Robert J. Simmons. Proofs from tests. In *ACM International Symposium on Software Testing and Analysis (ISSTA'08)*, pages 3–14, 2008.

[6] Bruno Blanchet. Automatic verification of correspondences for security protocols. *Journal of Computer Security*, 2009. To appear.

[7] Bruno Blanchet and Avik Chaudhuri. Automated formal analysis of a protocol for secure file sharing on untrusted storage. In *IEEE Symposium on Security and Privacy (S&P'08)*, pages 417–431, 2008.

[8] Avik Chaudhuri. Dynamic access control in a concurrent object calculus. In *International Conference on Concurrency Theory (CONCUR'06)*, pages 263–278, 2006.

[9] Avik Chaudhuri. On secure distributed implementations of dynamic access control. In *LICS/CSF Joint Workshop on Computer Security (FCS-ARSPA-WITS'08)*, pages 93–107, 2008.

[10] Avik Chaudhuri. A Concurrent ML library in Concurrent Haskell. In *ACM International Conference on Functional Programming (ICFP'09)*. To appear, 2009.

[11] Avik Chaudhuri. Language-based security on Android. In *ACM Workshop on Programming Languages and Analysis for Security (PLAS'09)*, pages 1–7, 2009.

[12] Avik Chaudhuri and Martín Abadi. Formal security analysis of basic network-attached storage. In *ACM Workshop on Formal Methods in Security Engineering (FMSE'05)*, pages 43–52, 2005.

[13] Avik Chaudhuri and Martín Abadi. Formal analysis of dynamic, distributed file-system access controls. In *IFIP International Conference on Formal Methods for Networked and Distributed Systems (FORTE'06)*, pages 99–114, 2006.

[14] Avik Chaudhuri and Martín Abadi. Secrecy by typing and file-access control. In *IEEE Computer Security Foundations Workshop (CSFW'06)*, pages 112–123, 2006.

[15] Avik Chaudhuri and Deepak Garg. PCAL: Language support for proof-carrying authorization systems. In *European Symposium on Research in Computer Security (ESORICS'09)*. To appear, 2009.

[16] Avik Chaudhuri, Prasad Naldurg, and Sriram Rajamani. A type system for data-flow integrity on Windows Vista. *ACM SIGPLAN Notices*, 43(12):9–20, 2008. *Best paper of PLAS'08.*

[17] Avik Chaudhuri, Prasad Naldurg, Sriram Rajamani, Ganesan Ramalingam, and L. Velaga. EON: Modeling and analyzing dynamic access control systems with logic programs. In *ACM Conference on Computer and Communications Security (CCS'08)*, pages 381–390, 2008.

[18] Petros Efstathopoulos, Maxwell Krohn, Steve VanDeBogart, Cliff Frey, David Ziegler, Eddie Kohler, David Mazières, Frans Kaashoek, and Robert Morris. Labels and event processes in the Asbestos operating system. In *ACM Symposium on Operating System Principles (SOSP'05)*, pages 17–30, 2005.

[19] David Flanagan and Yukihiro Matsumoto. *The Ruby Programming Language*. O'Reilly, 2008.

[20] Deepak Garg and Frank Pfenning. A proof-carrying file system. Technical Report CMU-CS-09-123, Carnegie Mellon University, 2009.

[21] Shai Halevi, Paul A. Karger, and Dalit Naor. Enforcing confinement in distributed storage and a cryptographic model for access control. Cryptology Eprint Archive Report 2005/169, 2005.

[22] Michael Howard and David LeBlanc. *Writing Secure Code for Windows Vista*. Microsoft Press, 2007.

[23] Galen Hunt, James R. Larus, Martín Abadi, Mark Aiken, Paul Barham, Manuel Fahndrich, Chris Hawblitzel, Orion Hodson, Steven Levi, Nick Murphy, Bjarne Steensgaard, David Tarditi, Ted Wobber, and Brian D. Zill. An overview of the Singularity project. Technical Report MSR-TR-2005-135, Microsoft Research, 2005.

[24] Mahesh Kallahalla, Erik Riedel, Ram Swaminathan, Qian Wang, and Kevin Fu. Plutus: Scalable secure file sharing on untrusted storage. In *USENIX Conference on File and Storage Technologies (FAST'03)*, pages 29–42, 2003.

[25] Sam Ruby, Dave Thomas, and David Heinemeier Hansson. *Agile Web Development with Rails, Third Edition*. The Pragmatic Bookshelf, 2009.