Secure and reliable software is hard to build, but the costs of failure are steep. Data breaches caused by attackers exploiting software vulnerabilities made many headlines in 2011 and 2012 and show no sign of abating. Sony, RSA Security, and PBS—among other high-profile companies—were compromised, their intellectual property stolen, and the privacy of their customers impacted, all because of software vulnerabilities. Software reliability problems have led to bungled lotteries, medical device failures, the early release of convicted felons, and enumerable other problems.

The precise details of software failures are often scarce, but it’s clear that the defects underlying many problems could have been identified earlier using static analysis. As software platforms proliferate, from mobile devices to the cloud to embedded devices such as the smart grid, it will be even more difficult to get software right.

What we need is a way for software developers to ensure that the software they develop can withstand hostile attacks. Many security processes can be embedded into software’s development to make it more secure. Some examples from Microsoft’s Secure Development Life Cycle (SDLC; www.microsoft.com/security/sdl/default.aspx) include security requirements, threat modeling, static analysis, dynamic analysis, security review, and product incident response.

Each security process has its strengths and weaknesses. One measure of a process’s value is the quantity of important security defects it can detect. Another is how early in the development process it can be performed (thus making correction less expensive). A third measure is how automated the process can be (automation reduces the cost of applying the process). Static analysis fares well by all of these measures, which has helped it to become one of the most popular security tools used during software development.
Static analysis can be thought of as a partial automation of code review. During code review, the programmer builds a mental model that describes the data and the program’s control flows as they perform their review. Using this mental model, the reviewer is looking to see that the code correctly implements the design—in a security review, it’s looking to see that the code correctly defends against known attacks. Automated static analysis puts the code review process on steroids.Millions of lines of code can be modeled and checked in a period of hours instead of months. And because the process doesn’t depend on a human reviewer’s skills, the output is much more consistent.

The Current State of the Technology

Static analysis holds huge promise for creating secure and reliable software. It’s likely that no other single technology in the past 10 years has helped find as many bugs in software—consequently, many of the largest software companies in the world use it religiously. Still, we’re experiencing the tool’s “middle school” years in terms of its maturation.

Accuracy and comprehensiveness over languages and frameworks continue to evolve, and the difficulties some organizations have experienced in adopting static analysis into their software development processes have slowed its usage. Most software development life cycles do not include static analysis, and the technology isn’t taught in computer science programs. The limited static-analysis functionality currently available in some compilers is turned off by default.

Clearly, there’s still much work to be done for all the software built to benefit from static analysis.

In This Issue

For this special theme issue of IEEE Security & Privacy, we selected articles from a wide variety of static-analysis experts from research teams, academia, government, and commercial software companies. The broad spectrum of ideas covered range from the practicality of building and making static-analysis tools usable in a major software company to ways of cataloging the problem space of vulnerabilities in software.

Cristina Cifuentes and her colleagues at Oracle Labs describe their journey from a research project to a tool supporting commercial developers in “Transitioning Parfait into a Development Tool.”

Robert A. Martin and Steven M. Christey of MITRE describe ways to catalog the problem space of software vulnerabilities in their article, “The Software Industry’s ‘Clean Water Act’ Alternative.”

In “SAVI: Static-Analysis Vulnerability Indicators,” James Walden and Maureen Doyle of Northern Kentucky University describe the ability of static analysis to predict the number of vulnerabilities that will be publicly reported over time for a given piece of software.

“Measuring the Value of Static-Analysis Tool Deployments,” authored by Paul Anderson of GrammaTech, explores the difficulty of determining the value of different static-analysis tools, building on false positive rates, false negatives rates, and the human interpretation of results that’s inherent in using any security testing process.

Paul E. Black at the US National Institute of Standards and Technology (NIST) describes the process for testing and measuring the accuracy of static-analysis tools at NIST’s Static Analysis Tool Expo in “Static Analyzers: Seat Belts for Your Code.”

Also included in this issue is a roundtable discussion with a group of static-analysis experts who share insights from their many years building, reviewing, and deploying the technology. The discussion starts with the history of static analysis and how important it is to the US Department of Defense today. It includes additional talk about how to get static analysis into the development process of a large commercial company, which is likely at the top of the list for many application security practitioners today.

During our roundtable discussion, Bill Pugh of the University of Maryland said, “In the software community, development teams are completely oblivious—still—to the need to adopt appropriate software security and privacy practices. How can we make software security tools appropriate and useful and used?” Much education must be done on the importance of using this technology. Our hope is that this issue helps educate software developers and security practitioners about the benefits and the possibilities for static analysis. We hope it can help drive adoption and allow static analysis to reach its full potential.

Brian Chess is one of the founders of Fortify Software. Contact him at brian.chess@gmail.com.


Selected CS articles and columns are also available for free at http://ComputingNow.computer.org.