Measuring the Attack Surfaces of SAP Software Systems

Pratyusa K. Manadhata[1] Yuecel Karabulut[2] Jeannette M. Wing[1]

There is a growing demand for secure software as we are increasingly dependent on software in our day-to-day life. The software industry has traditionally focused on improving code quality for improving software security and quality. The code quality improvement effort aims toward reducing the number of design and coding errors in software. An error causes software to behave differently from the intended behavior as defined by the software's specification; a vulnerability is an error that can be exploited by an attacker.

In practice, building large and complex software devoid of errors, and hence security vulnerabilities, remains a very difficult task. Software vendors have to face the hard fact that their software will ship with both known and future vulnerabilities in them and many of the vulnerabilities will be discovered and exploited. They can, however, minimize the risk associated with the exploitation of these vulnerabilities. One way to minimize the risk is by reducing software's *attack surface*. A smaller attack surface makes the exploitation of the vulnerabilities harder and lowers the damage of exploitation, and hence mitigates the security risk. The code quality effort and the attack surface reduction approach are complementary; a complete risk mitigation strategy requires a combination of both.

Michael Howard of Microsoft first introduced the informal notion of a software system's Relative Attack Surface Quotient in 2003, later generalized by Pincus and Wing. In 2006 Manadhata and Wing of Carnegie Mellon University (CMU) formalized the notion of a software system's attack surface and introduced a systematic but abstract method to measure and reduce the attack surface.

There is anecdotal evidence from the software industry to demonstrate that attack surface reduction mitigates software's security risk. For example, in case of Microsoft, the Sasser worm, the Zotob worm, and the Nachi worm did not affect some versions of Windows due the attack surface reduction in Windows. Similarly, Firefox 2.0 is immune to attacks that exploit a buffer overflow vulnerability in the Mozilla Network Security Services due to the attack surface reduction of Firefox.

CMU and SAP collaborated to apply the attack surface measurement and reduction approach to SAP's enterprise-scale business applications and technology platforms. The short-term goal of the collaboration was to demonstrate that the attack surface measurement approach is feasible for enterprise-scale software. The long-term goal was to evaluate the possibility of integrating the measurement process with SAP's software development process so that SAP's software developers and software quality group can use the measurement process in a regular basis.

The key contributions of the collaboration are the following: an instantiation of Manadhata and Wing's abstract method to obtain a concrete measurement method for SAP software systems implemented in `Java`; an implementation of a tool as an Eclipse plugin to measure a system's attack surface from the system's source code in an automated manner; the demonstration of the utility of the measurement method and the tool by measuring and comparing the attack surfaces of three versions of an SAP software system; the demonstration of the measurement result's use as a guide in reducing the system's attack surface; the demonstration of the measurement method's utility to both SAP developers and SAP customers; finally, based on SAP developers' feedback, the identification of future avenues of research to make the measurement method and the tool more useful in practice.

[1]Carnegie Mellon University, Pittsburgh, PA.
[2]SAP Research, Palo Alto, CA.