# How Useful Is Software Fault Injection for Evaluating the Security of COTS Products?

*Panel Moderator:* Jim Reynolds, Teknowledge
*Panelists:*
Matt Bishop, University of California at Davis
Anup Ghosh, Cigital
James Whittaker, Florida Institute of Technology

**Panel Abstract**

Software fault injection (SFI) is a controversial method for identifying errors and improving software. Many respected researchers believe the method holds promise, including the members on our panel, although with careful qualifications. On the other hand, COTS software manufacturers tend to view the method with skepticism for several reasons. One problem is the difficulty in verifying that injected faults are representative of real world faults. Another is that SFI may not be as efficient in identifying errors in software as more conventional testing. The three panelists explored wide-ranging alternatives to the industry view.

## Position Statements

### Matt Bishop

SFI is a testing tool that deliberately tries to cause errors in existing programs. The most common example is to supply, or inject, illegal data for input either through the standard input or through the environment. This is what attackers do when they test programs for vulnerabilities. In that context, the question of whether injected faults are representative of real world faults is not a fair question. The proper question is whether an attacker could use an injected fault to cause a security (or reliability) problem. If so, then the injected fault corresponds to a real world fault. One problem is describing the goal of the program, and its desired properties, in terms that can be used to derive inputs that can cause faults. Property-based testing tests conformance to specific properties, such as security properties. These are stated in a low-level specification language (TASpec) that is tied directly to the code. These properties can be used to generate test cases that will exercise the code to determine whether the code fails to satisfy the property. Combined with proper test coverage metrics, these faults can assure that the code satisfies the properties to an appropriate level of coverage.

### Anup Ghosh

SFI is an innovative technology that can be used to sidestep many of testing's difficult problems. Fault injection has been used successfully in the past in safety-critical applications to find failure modes that would have otherwise been extremely difficult to find using standard testing techniques. The $64,000 question is whether and to what extent fault injection is useful for assessing security of software systems. Our experience reflects that software fault injection is useful in limited contexts. For instance, we have been able to successfully inject buffer overrun attacks in stack buffer variables to determine a program's susceptibility to stack-smashing attacks. However, these results have limited value in determining whether such an attack is actually possible via standard user input (a more difficult problem to tackle). Fault injection via software wrappers can be useful in assessing the robustness of COTS operating systems by enabling simulation of failing OS resources that applications depend on. While this type of software fault injection falls more in the "dependability" bucket, it has applications to security such as determining susceptibility to denial-of-service. In the end, the value derived from SFI is highly dependent on the skill of the analyst. It is yet another tool to help determine how systems might break. Proper use of SFI can enable development of more robust systems. However, if you are looking for a tool to reveal all security-related flaws, keep looking.

### James Whittaker

There are many varieties of security-related software defects. Some can be found using fairly ordinary testing strategies; others are manifested only when the system under test is operating in a stressed environment. Stress can be induced by load, interoperability problems with

another application, or faults in the environment of the system. Indeed, the failure of one component may have cascading effects on other applications and/or the security of the network/enterprise. Software testers concerned with security must be able to force both interoperability stress and stress from faulty components in order to expose these bugs to scrutiny. Tools for such fault injection are badly needed. However, we believe that injection of faults into a software's environment are much more appropriate for security testing than injection of faults into the source code of the system under test. For one, environment faults are more realistic (they will be the ones the hackers try to exploit). Also, environment faults are much more difficult to stage in the laboratory. Thus tools to inject environment faults are what's really needed.