Carnegie Mellon

Ballista® Software Robustness Testing

Carnegie Mellon University
HH D-202
Pittsburgh, PA 15213
USA

ballista@ece.cmu.edu
www.ballista.org
ph: +1 412/268-5225
fax: +1 412/268-6353

Phil Koopman, Dan Siewiorek, Kobey DeVale
John DeVale, Kim Fernsler, Dave Guttendorf, Nathan Kropp, Jiantao Pan, Charles Shelton, Ying Shi

Affiliated with:

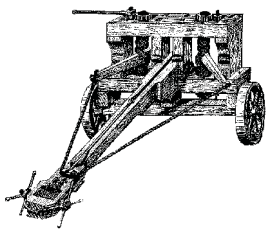
Institute for Complex Engineered Systems

Institute for Complex Engineered Systems
http://www.ices.cmu.edu

Institute for Software Research, International
http://www.isri.cs.cmu.edu

Electrical & Computer ENGINEERING

Electrical & Computer Engineering Department
http://www.ece.cmu.edu



June 2002

Ballista® automated robustness testing characterizes the exception handling effectiveness of software modules. For example, Ballista testing can find ways to make operating systems crash in response to exceptional parameters used for system calls, and can find ways to make other software packages suffer abnormal termination instead of gracefully returning error indications. Ballista is a "black box" software testing tool, and is works well on robustness testing the APIs of Commercial Off-The-Shelf (COTS) software.

MEASURING SOFTWARE ROBUSTNESS

The success of many products depends on the robustness of not only the product software, but also operating systems and third party component libraries. But, until now, there has been no way to quantitatively measure robustness. Ballista changes this by providing a simple, repeatable way to directly measure software robustness without requiring source code or behavioral specifications. As a result, product developers can use robustness metrics to compare off-the-shelf software components, and component developers can measure their effectiveness at exception handling.

The Ballista testing approach is both scalable and portable across a wide variety of application domains. No behavior specification is required for testing – the implicit specification of "doesn't crash; doesn't hang" suffices. Additionally, tests are created based on the data types of the parameter list rather than based on module functionality, exploiting the fact that in most APIs there are fewer data types than functions/calls.

Our research findings contradict some widely held opinions about robustness. For example, we have found one-line user programs that crash commercial operating systems without a need for concurrency or complex timing conditions. Also, extensive parameter checking to avoid such vulnerabilities can be provided at essentially no run-time penalty by using appropriate techniques.

BALLISTA APPROACH

Ballista testing begins by identifying the data types used by an API under test. Application-specific data types can inherit base test cases from predefined data types in the Ballista testing tool set. Then, the Ballista test harness generator is given the signature for a function to be tested in terms of those data types, and generates a customized testing harness. The test harness composes combinations of test values for each parameter, and reports robustness testing results. Ballista uses a C++ testing harness, but can be used with most interfaces that are C++ linkable. While testing large databases this way is not a particularly effective approach, Ballista has been demonstrated to be effective and scalable when testing a variety of APIs with small to moderate amounts of system state that must be set before execution of an individual test case.

ROBUSTNESS OF OPERATING SYSTEMS

Ballista testing can be performed on almost any API that employs calls with parameter lists. The POSIX (Unix) operating system API has been used as the first example for robustness testing. As the Figure below shows, many robustness failures were observed (data from a 1999 study). Several instances were found in which a single line of C code crashed an operating system. Most events found were abnormal task terminations ("Abort" failures) or incorrect acceptance of invalid inputs.

AVAILABILITY

A Ballista testing toolkit is distributed at no charge under the GNU Public License. It comes pre-populated with data types and testing information to exercise many operating system calls and C-library functions.