

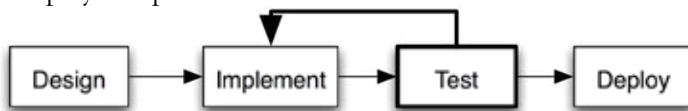
DEADBOLT

Automated testing for PCS software robustness and security



Overview

Vulnerabilities in process control systems (PCS) and SCADA software threaten availability and integrity of control and data acquisition services provided by these systems. The DEADBOLT tool suite provides PCS and SCADA software developers with state-of-the-art technology that facilitates automatic discovery of buffer overflow and resource exhaustion vulnerabilities. Buffer overflow errors can be exploited by a malicious attacker to make unauthorized changes to system configuration, learn confidential information or force process control software to fail. Resource exhaustion conditions can result in software failing to support legitimate operations (denial of service) and can be triggered by causes ranging from simple equipment malfunction to a worm infection or a malicious attack. Since availability and integrity of control system operations are of critical importance, finding and addressing these errors in PCS and SCADA software before it is deployed is paramount.



“Software bugs, or errors, are so prevalent and so detrimental that they cost the US economy an estimated \$59.5 billion annually. Approximately 60% of the costs are borne by software users and 40% by software developers and vendors. Although all errors cannot be removed, more than a third of these costs (an estimated \$22.2 billion) could be eliminated by an improved testing infrastructure that enables earlier and more effective identification and removal of software defects.”

Source: “The Economic Impacts of Inadequate Infrastructure for Software Testing”, NIST 2002.

The DEADBOLT tool suite focuses on helping software developers discover these errors during implementation and testing phases of the software development life-cycle. DEADBOLT supplements functional testing by

- Employing runtime monitoring to identify memory corruption and management errors
- Automatically generating test cases that trigger buffer overflows, memory corruption errors or resource exhaustion
- Providing detailed information about discovered errors, including exact location of faulty code and input that triggers erroneous behavior

DEADBOLT’s error reports greatly facilitate debugging and enable software developers to create more secure and robust implementations of PCS and SCADA software.

What is a Buffer Overflow?

Memory in C and C++ programs is managed by creating and destroying memory objects. This can be done explicitly by the programmer or automatically by the system. A buffer overflow is any access to an object that ventures beyond the bounds of its allocated memory. Writing beyond the bounds of an object frequently enables an attacker to modify program logic by changing operating parameters or subvert the program altogether and execute arbitrary code.

A Stack-Smashing Attack

“Stack-smashing” is a common attack that exploits an overflow in a stack buffer. Upon entry into a function the program saves a “return address,” specifying code that should be executed after the function is completed. This important value is stored on the stack, next to function arguments and buffers used by the function for processing inputs. An attacker can craft an input that is too large for the buffer, and, if appropriate checks are not made, the function will continue to write beyond the bounds of the buffer, overwriting the saved return address. A buffer overflow thus allows an attacker to insert attack code into the buffer and change the return address. This attack code is then executed when the current function is completed. In effect, the attacker takes over the process and can execute arbitrary code with the privilege level of the process that has been exploited.



Key Features & Benefits

- Improved testing enables identifying and fixing bugs early
 - Reduces cost of development & maintenance
 - Reduces time-to-market
 - Enables more robust PCS operation in face of malicious attacks
- Focuses on critical problems in PCS environment and applications
 - Embedded systems with limited resources
 - PCS software in C and C++
- DEADBOLT keeps it simple
 - Requires no changes to application source code
 - Integrates with well-established development environment and tools

Functional Description

DEADBOLT is a suite of developer tools that complement functional and regression testing during the implementation and testing phases of the software development life-cycle. DEADBOLT tools require source code for the application under test and a collection of sample valid inputs. Final version of the tool suite will be incorporated into an integrated development environment (IDE), such as Eclipse, which provides developers with easy navigation and editing capabilities, debuggers, performance profiling tools, etc. This integration will ensure that DEADBOLT tools are easy to use and leverage an environment already familiar to developers.

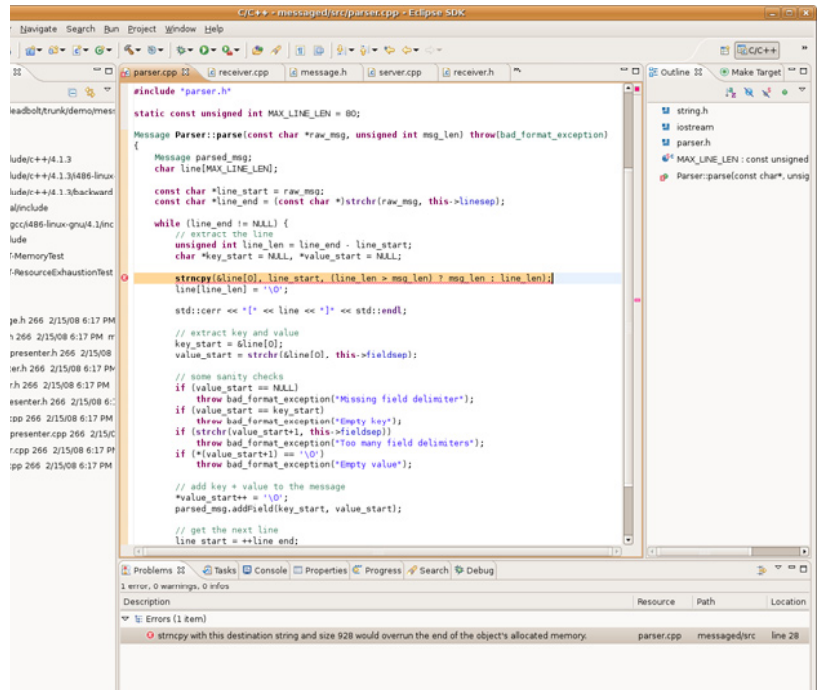
Technical Description

DEADBOLT tools employ three different technologies to discover memory corruption and resource exhaustion errors automatically: memory access monitoring, input taint-tracking and automated testcase generation. Combining these technologies in a feedback loop enables DEADBOLT to observe program execution, detect errors and generate new test cases that trigger erroneous behavior.

To detect memory corruption errors (such as buffer overflows), DEADBOLT uses source-to-source transformations to insert instrumentation that keeps track of all allocated memory and checks all memory accesses. A runtime component ensures that even very small overflows (e.g. off by one byte) are detected and promptly reported to the developer.

Taint-tracking technology is used to enable automated test case generation. Any input that may be controlled by an attacker (e.g. network packets) are labeled as “tainted”, and a runtime component monitors the way this taint propagates through the program execution. Taint-tracking enables DEADBOLT to determine which input bytes are used to make critical decisions inside the application code.

DEADBOLT’s automated test case generation is based on a white-box smart-fuzzing approach. Information from memory access monitoring and taint-tracking runtime components determines input bytes that should be changed to create a test case that is likely to trigger an error. This enables DEADBOLT to create inputs that test boundary conditions within the program and either discover errors or verify that appropriate checks within the program are implemented correctly.



The screenshot above shows the Eclipse integrated development environment with output of DEADBOLT buffer overflow detection tool pin-pointing the line of source code containing the out-of-bounds memory access as well as the error description.

Summary of Costs

	Low	Medium	High
Component	<\$1,000	\$1,000-\$10,000	>\$10,000
Engineering	Drop-in	Moderate modification	Complete System Design Lifecycle
Bandwidth/Network Burden	None-Passive Only	Moderate Traffic, Medium Overhead	Heavy Traffic, Large Overhead
Training	No training required	Moderate training	Intensive training. Additional staffing may be required.
Maintenance and Operation	< 1 hour per week	< 5 hours per week	> 5 hours per week

Technology Transfer and Readiness:

- Buffer overflow and memory corruption detection: Bench-tested.
- Resource exhaustion detection: Research prototype

This material is based upon work supported by the U.S. Department of Homeland Security under Grant Award Numbers 2006-CS-001-000001 and 2003-TK-TX-0003, under the auspices of the Institute for Information Infrastructure Protection (I3P) research program. The I3P is managed by Dartmouth College. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security, the I3P, or Dartmouth College.

Researcher Contact Information
Michael Zhivich • mzhivich@ll.mit.edu

The I3P is managed by Dartmouth College.