

Wrapping Windows NT Binary Executables for Failure Simulation*

Anup K. Ghosh and Matt Schmid
Reliable Software Technologies
21515 Ridgetop Circle, #250, Sterling, VA 20166
{aghosh,mschmid}@rstcorp.com
www.rstcorp.com

1 Introduction

In this short paper, we describe a tool for testing the reliability and robustness of Windows NT software applications under stressful environmental conditions, *i.e.*, under system resource failure conditions. Windows NT systems are increasingly being deployed in mission-critical applications such as for command and control in US Navy ships [1]. However, as recently as July, 1998, the Navy's Aegis missile cruiser, USS Yorktown, suffered a significant software problem in the Windows NT systems that control the "smart ship" that effectively left the ship dead in the water [2]. The ship had to be towed to the Norfolk Naval shipyard because a database overflow error (resulting from a divide by zero operation) caused the ship's propulsion system to fail.

The research approach and prototype tool described here are specifically designed to analyze commercial off-the-shelf (COTS) software for Win32 systems where source code is not released, but binary executables are available for dynamic analysis. The purpose of this research is to assess the robustness of software applications to failing system resources such as memory allocation functions and system I/O functions. The tool gives an analyst the capability to artificially simulate stressful conditions (*e.g.*, complete memory utilization) that a program may experience during its lifetime using simple toggle functions.

2 Approach

Given the constraint of working with binary executables without resorting to decompilation techniques, the approach in this research project has been to instrument interfaces between the application program under analysis and the shared libraries within the operating system that the application uses. The

approach is to "wrap" a binary executable with an instrumentation layer such that all interactions between an application and the operating system can be captured, observed, perturbed, and questioned.

Windows NT applications import hundreds of system functions from shared libraries called Dynamically Linked Libraries (DLLs). As implied by their name, these libraries are linked during runtime. System DLLs make a good candidate for studying the effect of system failures on applications because they typically contain the core functions within the operating system that applications require. As such, they can be a single point of failure in a system. If the core OS functions fail, the programs that use them may fail in turn. For this reason, selectively simulating failures of operating system resources (such as memory allocation/deallocation, file system operations, and other system I/O operations) can identify how robust, or conversely, how vulnerable, an application is to failing system resources. Studying these failure modes is important in critical applications where system resources may be unavailable during peak periods when they are most essential.

3 Wrapping Binary Executables

The approach taken to simulate failed system resources is to wrap binary executables with an instrumentation layer that simulates system failures. Figure 1 illustrates how program executables are wrapped. The application's Import Address Table (IAT), which is used to look up the address of imported DLL functions, is modified for functions that are wrapped to point to the wrapper DLL. For instance, in Figure 1, functions S1 and S3 are wrapped by modifying the IAT of the application. When functions S1 and S3 are called by the application, the wrapper DLL is called instead. The wrapper DLL, in turn, executes, providing the ability to modify, perturb, question or simply log the request to the target DLL.

*This work is sponsored by the Air Force Research Laboratory and the Defense Advanced Research Projects Agency (DARPA) under Contract F30602-97-C-0117.

