Robustness Testing of the Microsoft Win32 API

http://ballista.org

Charles P. Shelton
cshelton@cmu.edu

Philip Koopman
koopman@cmu.edu - (412) 268-5225 - http://www.ices.cmu.edu/koopman

Kobey DeVale
Overview: Applying Ballista to Windows Systems

◆ Introduction
  • Motivation for measuring robustness of Windows Operating Systems
  • Ballista Testing Service

◆ Running Ballista on Windows
  • Test Development
  • Systems Tested

◆ Results
  • Catastrophic Failures (system crashes)
  • Comparing Windows and Linux
  • Restart and Abort Failures (task hangs and crashes)
  • Silent Failures

◆ Conclusions and Future Work
Robustness and Microsoft Windows

- **Little Quantitative data on Windows system robustness**
  - Only anecdotal evidence comparing Windows systems to POSIX systems
  - Measuring how well Windows systems handle exceptions will give us insight into its robustness
  - Specifically target Win32 API calls similar to POSIX system calls

- **Windows NT and Windows CE deployed in critical systems**
  - US Navy is moving to Windows NT as standard OS for all ship computer systems
  - Windows CE is a contender for many embedded systems
    - Emerson Electric sponsored this work
      (use Windows CE in industrial equipment?)
Ballista Robustness Testing Service

- **Ballista Server**
  - Selects tests
  - Performs pattern Analysis
  - Generates “bug reports”
  - Never sees user’s code

- **Ballista Client**
  - Links to user’s SW under test
  - Can “teach” new data types to server (definition language)
Windows Test Development

- Start with test suite of standard UNIX datatypes
- The Win32 API uses many non-standard datatypes
  - However, most of these are pointers to structures that can inherit test cases from generic pointer datatypes
  - The HANDLE datatype in Windows required the most development of new test cases
    - Win32 API uses HANDLES for everything from file pointers to process identifiers
    - Test cases were generated to specifically exercise different uses of the HANDLE datatype

- Test cases
  - 1,073 distinct test values in 43 datatypes available for testing in Win32
  - 3,430 distinct test values in 37 datatypes available for testing in POSIX (2,908 of these values in two datatypes that had no analog in Windows)
  - Limit of 5,000 test cases per function
  - Over 500,000 generated test cases for each Windows variant
  - Over 350,000 generated test cases for Linux
Systems Tested

- Desktop Windows versions on Pentium PC
  - Windows 95 revision B
  - Windows 98 with Service Pack 1 installed
  - Windows 98 Second Edition (SE) with Service Pack 1 installed
  - Windows NT 4.0 with Service Pack 5 installed
  - Windows 2000 Beta 3 Pre-release (Build 2031)
  - 143 Win32 API calls + 94 C library functions tested

- Windows CE
  - Windows CE 2.11 running on a Hewlett Packard Jornada 820 Handheld PC
  - 69 Win32 API calls + 82 C library functions tested

- POSIX System for Comparison
  - RedHat Linux 6.0 (Kernel version 2.2.5)
  - 91 POSIX kernel calls + 94 C library functions tested
Robustness Problems Found – System Crashes

Number of Functions with Catastrophic Failures

<table>
<thead>
<tr>
<th>Operating System</th>
<th>Number of Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>None</td>
</tr>
<tr>
<td>Windows 95</td>
<td>5</td>
</tr>
<tr>
<td>Windows 98</td>
<td>4</td>
</tr>
<tr>
<td>Windows 98 SE</td>
<td>4</td>
</tr>
<tr>
<td>Windows NT</td>
<td>None</td>
</tr>
<tr>
<td>Windows 2000</td>
<td>None</td>
</tr>
<tr>
<td>Windows CE 2.11</td>
<td>25</td>
</tr>
</tbody>
</table>
Data Analysis and Comparison

How do we compare robustness results of non-identical API’s?

- Win32 API is vastly different from POSIX API
- Windows CE only supports a fraction of entire Win32 API

Group functions according to services provided

- Groups of C library functions
- Groups of system calls
- Calculate percent failure rate for each function in group
- Take average of all functions in the group to determine overall group percent failure rate
- Windows CE notes
  - Functions in C File I/O and C Stream I/O groups have too many crashes to report failure rates in percent
  - Windows CE does not support functions in the C Time group: asctime(), ctime(), gmtime(), localtime(), mktime(), etc.
Failure Rates by Function Group – System Calls

The graph illustrates the group average abort + restart failure rate for different function groups across various operating systems. The function groups include:

- Memory management
- File and Directory Access
- I/O Primitives
- Process Primitives
- Process Environment

The operating systems compared are:

- Linux
- Windows 95
- Windows 98
- Windows 98 SE
- Windows NT
- Windows 2000
- Windows CE 2.11

The x-axis represents the function groups, and the y-axis shows the percentage of failure rates ranging from 0% to 100%. Each function group is represented by a bar chart, and the operating systems are indicated by different colors and chart styles.
Silent Failures

- **False negative failure detection**
  - Function called with invalid parameter values but no error reported

- **Silent failures cannot be directly measured**
  - How do you declare silent failures without annotating every test case?
  - Requires an oracle for correctness testing
  - Doesn’t scale

- **But they can be estimated**
  - We have several different implementations of the same API with identical test cases
    - Excludes Linux and Windows CE
  - Every test case with a “Pass” result with no error reported is a possible silent failure
  - Vote across identical test cases in different systems
    - Assumes the number of false Abort/Restart failures is not significant
    - Does not catch silent failure cases where all systems do not report an error
Estimated Silent Failure Rates – System Calls

- Memory management
- File and Directory Access
- I/O Primitives
- Process Primitives
- Process Environment

- Windows 95
- Windows 98
- Windows 98 SE
- Windows NT
- Windows 2000
Estimated Silent Failure Rates – C Library

Group Average Silent Failure Rate

- Windows 95
- Windows 98
- Windows 98 SE
- Windows NT
- Windows 2000
Windows Testing Conclusions

- Compare different API’s by Functional Grouping
  - Approximate an “apples-to-apples” comparison
  - Functional groupings identify relative problem areas

- Linux and Windows NT/2000 seem more robust than Windows 95/98/98 SE and Windows CE
  - Complete system crashes observed on Windows 95/98/98 SE and Windows CE; none observed on Windows NT/2000 or Linux
  - Low Abort failure rate on Win 95/98/98 SE system calls … … because of a high Silent failure rate
  - Windows CE is markedly more vulnerable to crashes

- Comparison of Windows NT/2000 and Linux inconclusive
  - Linux POSIX system calls generally better than Windows Win32 calls
  - Windows C library generally better than Linux / GNU C libraries
Future Work - Microsoft Support

- Submitted bug reports for Catastrophic failures for Windows 95/98/98 SE

- Will Windows ME (Millennium) fix the problems we found?

- Arranging to report Windows CE Catastrophic failures

- Heavy load testing