Reliability Challenges for Commodity Operating Systems

Hank Levy
Dept. of Computer Science & Engineering
University of Washington
Two Examples of Systems Research

1. Research in operating systems design
   • Making the world safe from operating system extensions

2. Internet measurement research
   • Understanding the spyware threat
Part 1

Improving the reliability of commodity operating systems

Joint work with
Mike Swift and Brian Bershad
Outline

• Problem
• Design & Implementation of Nooks
• Evaluation
• Summary
A lot of research effort in the OS community has gone into performance, rather than reliability.

The result: operating system crashes are still a huge problem today
- 5% of Windows systems crash every day

Device drivers are the biggest cause of crashes
- Drivers cause 85% of Windows XP crashes
- Drivers in Linux are 7 times buggier than the kernel
What is a Device Driver?

A module that translates high-level I/O requests to device-specific requests

- 10s of thousands of device drivers exist
  - Over 35K drivers on Win/XP!
- 81 drivers running on this laptop
- Drivers run inside the OS kernel
  - A bug in a driver crashes the OS
- Small # of common interfaces
OS Today

Application

Kernel
- Virtual Memory
- File Systems
- Networking
- Scheduling
- ...

Application

Device Drivers

70% of Linux kernel code!
Driver Reality -- Linux

[Chou et al. 2001]
Why Do Drivers Fail?

• Complex and hard to write
  – Must handle asynchronous events
    • interrupts
  – Must obey kernel programming rules
    • Locking, synchronization
  – Difficult to test and debug
    • timing-related bugs
  – Non-reproducible failures

• Often written by inexperienced programmers

• Code often not available to OS vendors
Our Goal: OS With Reliability

- Kernel
- Application
- Driver

Diagram showing the interactions between kernel, driver, and applications.
What we did

We designed and built a new Linux kernel subsystem ("Nooks") that:

- Prevents the **majority** of driver-caused crashes
- Requires **no** changes to existing drivers
- Requires only **minor** changes to the OS
- Minimally impacts performance
Outline

• Problem
• Design and Implementation of Nooks
• Evaluation
• Summary
Outline

• Problem

• Design and Implementation of Nooks
  – Isolation
  – Recovery

• Evaluation

• Summary
Nooks System Architecture

- Application
- Application
- Kernel
  - Lightweight Kernel Protection Domains
  - XPC
  - Object Table
  - Shadow Drivers
  - Wrappers
- Nooks Reliability Layer
- Driver
- Driver
- Driver
Outline

• Problem

• Design and Implementation of Nooks
  – Isolation
  – Recovery

• Evaluation

• Summary
Existing Kernels

- Application
- Kernel
- Driver
- Application
Memory Isolation

Lightweight Kernel Protection Domains
Control Transfer

Application

Kernel

Driver

Application
Control Transfer

Application

Application

Kernel

Driver

eXtension Procedure Call

XPC

XPC
Transparency

Application

Kernel

Wrappers

Driver

XPC

XPC

Application
Data Access

Application

Object Table

Kernel

Driver

Copy-in/Copy-out

Application

Timer

Buffer

…

…
Isolation (recap)

- Isolation
  - Lightweight Kernel Protection Domains
  - eXtension Procedure Call (XPC)
  - Wrappers
  - Object Table
  - Copy-in/Copy-out of Kernel objects
Outline

• Problem
• Design and Implementation of Nooks
  – Isolation
  – Recovery
• Evaluation
• Summary
Shadow Drivers

• Shadow Driver Goals:
  – Restore driver state after a failure so it can process requests as if it had never failed
  – Conceal the failure from OS and applications

➤ One shadow driver handles recovery for an entire class of drivers
Shadow Driver Overview
Preparing for Recovery

Kernel -> Tap -> Device Driver

config(…)

Tap -> Shadow Driver

config(…)

config(…)

config (…)

Recovering a Failed Driver

Kernel

Tap

Device Driver

Shadow Driver

register(...)

config

...
Recovering a Failed Driver

• Summary:
  – Garbage collect failed driver
  – Reset driver
  – Reinitialize driver
  – Replay logged requests
Spoofing a Failed Driver

- Shadow driver acts as failed driver during recovery
Spoofing a Failed Driver
Spoofing a Failed Driver

Shadow acts as driver
  – Applications and OS unaware that driver failed
  – No device control

General Strategies:
1. Answer request from log
2. Act busy
3. Block caller
4. Queue request
5. Drop request
Outline

- Problem
- Design and Implementation of Nooks
  - Evaluation
- Summary
Implementation Complexity

• Changes to existing code
  – Kernel: 924 out of 1.1 million lines
  – Device drivers: 0 out of 50,000 lines

• New code
  – Isolation: 23,000 lines
  – Recovery: 3,300 lines
    • Each shadow driver is only a few hundred lines of code
<table>
<thead>
<tr>
<th>Class</th>
<th>Drivers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sound</td>
<td>Soundblaster Audigy, Soundblaster 16, Soundblaster Live!, Intel 810 Audio, Ensoniq 1371, Crystal Sound 4232</td>
</tr>
<tr>
<td>Network</td>
<td>Intel Pro/1000 Gigabit Ethernet, AMD PCnet32, Intel Pro/100 10/100, 3Com 3c59x 10/100, SMC Etherpower 100</td>
</tr>
<tr>
<td>IDE Storage</td>
<td>ide-disk, ide-cd</td>
</tr>
</tbody>
</table>
Reliability Test Methodology

Load driver

Inject bugs

Test

Nothing

Failure

Recovery

Reboot
Isolation Works

![Graph showing the comparison between No Nooks and Nooks for pcnet32 and Driver. The graph indicates that pcnet32 with No Nooks has a value of 119.]
Isolation Works

![Graph showing isolation works with pcnet32 driver]
Isolation Works

![Bar chart showing the comparison between No Nooks and Nooks for pcnet32 and e1000 drivers. The chart indicates that pcnet32 has 119 with No Nooks and 0 with Nooks, while e1000 has 52 with No Nooks.]
Isolation Works

![Bar Chart]

- **pcnet32**:
  - No Nooks: 119
  - Nooks: 0

- **e1000**:
  - No Nooks: 52
  - Nooks: 0

Legend:
- **No Nooks**
- **Nooks**
Isolation Works

![Bar chart showing the performance of different drivers with and without Nooks.

- pcnet32: 119 No Nooks, 0 Nooks
- e1000: 52 No Nooks, 0 Nooks
- ide-disk: 152 No Nooks, 0 Nooks]
Isolation Works
Recovery Works

Sound | Net | Storage

Driver Failures
Application Failures

Number of failures

Mp3 Player | Audio Recorder | Remote Copy | Sniffer | Compiler | Database

0 20 40 60 80 100
Relative Performance

- Mp3 Player
- Audio Recorder
- Network Send
- Network Receive
- Compiler
- Database

Relative Performance (%)
Evaluation: Bottom Line

• Isolation works
  – We can avoid crashes in the majority of driver failures

• Recovery works
  – We can keep applications running in the majority of driver failures

• The cost is acceptable
  – In many cases, the performance cost is acceptable
Summary of Part I

• We took a very targeted and practical approach to improving OS reliability
• We defined a set of new components and techniques to create a new OS reliability layer
• We used these components to build isolation and recovery services
• Our experiments demonstrate that:
  – Nooks prevents 99% of the crashes caused by our tests
  – Nooks keeps applications running in 98% of tested driver failures
  – There is high leverage in this approach
Part II

A Crawler-Based Study of Spyware on the Web

Joint work with Alex Moshchuk, Tanya Bragin, and Steve Gribble
What is spyware?

• Broad class of malicious and unwanted software
• Steal control of a PC for the benefit of a 3\textsuperscript{rd} party

• Characteristics:
  – Installs without user knowledge or consent
  – Hijacks computer’s resources or functions
  – Collects valuable information and relays to a 3\textsuperscript{rd} party
  – Resists detection and uninstallation
How do people get spyware?

• Spyware piggybacked on popular software
  – Kazaa, eDonkey

• Drive-by downloads
  – Web page installs spyware through browser
  – With or without user consent

• Trojan downloaders
  – Spyware downloads/installs more spyware
Why measure spyware?

• Understand the problem before defending against it
• Many unanswered questions
  – What’s the spyware density on the web?
  – Where do people get spyware?
  – How many spyware variants are out there?
  – What kinds of threats does spyware pose?
• New ideas and tools for:
  – Detection
  – Prevention
Approach

• Large-scale study of spyware:
  – Crawl “interesting” portions of the Web
  – Download content
  – Determine if it is malicious

• Two strategies:
  – Executable study
    • Find executables with known spyware
  – Drive-by download study
    • Find Web pages with drive-by downloads
Outline

• Introduction
• Executable file study
• Drive-by download study
• Summary
• Conclusions
Analyzing executables

- Web crawler collects a pool of executables
- Analyze each in a virtual machine:
  - Clone a clean WinXP VM
  - Automatically install executable
  - Run analysis to see what changed
    - Currently, an anti-spyware tool (Ad-Aware)
- Average analysis time – 90 sec. per executable
Executable study results

- Crawled 32 million pages in 9,000 domains
- Downloaded 26,000 executables
- Found spyware in 12.3% of them
  - Most installed just one spyware program
    - Only 6% installed three or more spyware variants
  - Few spyware variants encountered in practice
    - 142 unique spyware threats
Main targets

• Visit a site and download a program
• What’s the chance that you got spyware?
Popularity

- A small # of sites have large # of spyware executables:

- A small # of spyware variants are responsible for the majority of infections:
Types of spyware

• Quantify the kinds of threats posed by spyware
• Consider five spyware functions
  – What’s the chance an infected executable contains each function?

<table>
<thead>
<tr>
<th>Function</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keylogger</td>
<td>0.05%</td>
</tr>
<tr>
<td>Dialer</td>
<td>1.2%</td>
</tr>
<tr>
<td>Trojan downloader</td>
<td>12%</td>
</tr>
<tr>
<td>Browser hijacker</td>
<td>62%</td>
</tr>
<tr>
<td>Adware</td>
<td>88%</td>
</tr>
</tbody>
</table>
Example of a Nasty Executable

- [http://aaa1screensavers.com/](http://aaa1screensavers.com/)
  - “Let all your worries melt away into this collection of clouds in the sky – 100% free!”
  - [http://aaa1screensavers.com/free/clouds.exe](http://aaa1screensavers.com/free/clouds.exe)
- Installs 11 spyware programs initially
  - Includes a trojan downloader; continually installs more spyware
    - 10 more within first 20 minutes
- 12 new items on desktop, 3 browser toolbars
- Shows an ad for every 1.5 pages you visit
- CPU usage is constantly 100%
- No uninstallers
- System stops responding in 30 mins
  - Restarting doesn’t help
- Unusable system and no screensaver!
Outline

• Introduction
• Executable file study
• Drive-by download study
• Summary
• Conclusions
Finding drive-by downloads

• Evaluate the safety of browsing the Web

• Approach: automatic virtual browsing
  – Render pages in a real browser inside a clean VM
    • Internet Explorer
    • Mozilla Firefox
  – Identify malicious pages
    • Define triggers for suspicious browsing activity
    • Run anti-spyware check only when trigger fires
Event triggers

- Real-time monitoring for non-normal behavior:
  - Process creation
  - File events
    - Example: foo.exe written outside IE folders.
  - Registry events
    - Example: new auto-start entry for foo.exe

- No false negatives (theoretically)
- 41% false positives:
  - Legitimate software installations
  - Background noise
  - Spyware missed by our anti-spyware tool
More on automatic browsing

• Caveats and tricks
  – Restore clean state before navigating to next page
  – Speed up virtual time
  – Monitor for crashes and freezes

• Deciding what to say to security prompts:
  – “yes”
    • Emulate user consent
  – “no” (or no prompt)
    • Find security exploits
Drive-by download results
(unpatched Internet Explorer, unpatched WinXP)

- Examined 50,000 pages
- 5.5% carried drive-by downloads
  - 1.4% exploited browser vulnerabilities

![Graph showing percentage of pages with drive-by downloads by category]
Types of spyware

- Is drive-by download spyware more dangerous?

<table>
<thead>
<tr>
<th>Spyware Type</th>
<th>Executables</th>
<th>Drive-by Downloads</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keylogger</td>
<td>0.05%</td>
<td>0%</td>
</tr>
<tr>
<td>Dialer</td>
<td>1.2%</td>
<td>0.2%</td>
</tr>
<tr>
<td>Trojan Downloader</td>
<td>12%</td>
<td>50%</td>
</tr>
<tr>
<td>Browser hijacker</td>
<td>62%</td>
<td>84%</td>
</tr>
<tr>
<td>Adware</td>
<td>88%</td>
<td>75%</td>
</tr>
</tbody>
</table>
Is Firefox better than IE?

• Repeat drive-by download study with Mozilla Firefox
  • Found 189 (0.4%) pages with drive-by downloads
    – All require user consent
    – All are based on Java
    • Work in other browsers

• Firefox is not 100% safe
  – However, much safer than IE

<table>
<thead>
<tr>
<th>Category</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>adult</td>
<td>0</td>
</tr>
<tr>
<td>celebrity</td>
<td>33</td>
</tr>
<tr>
<td>games</td>
<td>0</td>
</tr>
<tr>
<td>kids</td>
<td>0</td>
</tr>
<tr>
<td>music</td>
<td>1</td>
</tr>
<tr>
<td>news</td>
<td>0</td>
</tr>
<tr>
<td>pirate</td>
<td>132</td>
</tr>
<tr>
<td>random</td>
<td>0</td>
</tr>
<tr>
<td>wallpaper</td>
<td>0</td>
</tr>
<tr>
<td>blacklist</td>
<td>23</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td><strong>189</strong></td>
</tr>
</tbody>
</table>
Summary

• Lots of spyware on the Web
  – 1 in 8 programs is infected with spyware
  – 1 in 18 Web pages has a spyware drive-by download
  – 1 in 70 Web pages exploits browser vulnerabilities

• Most of it is just annoying (adware)
  – But a significant fraction poses a big risk

• Spyware companies target specific popular content
  – Most piggy-backed spyware in games & celebrity sites
  – Most drive-by downloads in pirate sites

• Few spyware variants are encountered in practice
Solution Tidbit 1: Spyproxy

- Client browser
- Proxy front end
- Squid web cache
- Spyproxy
- VM worker
- Web

URL (root page)

Safe!
Solution Tidbit 2: Tahoma “Browser OS”

- Browser instance
  - Browser
  - Web doc.

- Bank Web service
  - site
  - site

- Radio Web service
  - site
  - site

- Internet

- Browser OS

- Web application 1

- Web application 2

- Client side
- Server side
Summary

- We addressed key questions about spyware
- Measured the density of spyware in the Web
- Looked at change in spyware over time (see the paper)
- Built useful tools and infrastructure
- Designed new architectures for safe browsing and spyware prevention
Thanks!

• For more info:
  
  
  
  

• www.cs.washington.edu/homes/levy