

Reliability Challenges for Commodity Operating Systems

Hank Levy

Dept. of Computer Science & Engineering
University of Washington

Part 1

**Improving the reliability of
commodity operating systems**

Joint work with
Mike Swift and Brian Bershad

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

Outline

- Problem
- Design & Implementation of Nooks
- Evaluation
- Summary

The High Level Picture

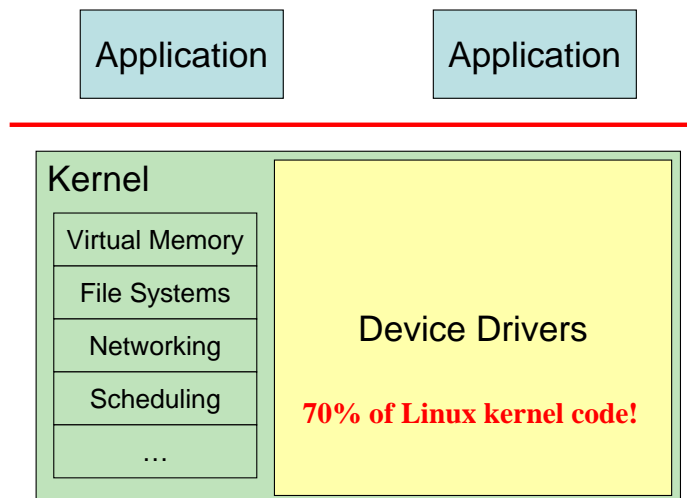
- A lot of research effort in the OS community has gone into *performance*, rather than reliability.
- The result: operating system crashes are still a huge problem today
 - 5% of Windows systems crash every day
- **Device drivers** are the biggest cause of crashes
 - Drivers cause **85%** of Windows XP crashes
 - Drivers in Linux are **7** times buggier than the kernel

What is a Device Driver?

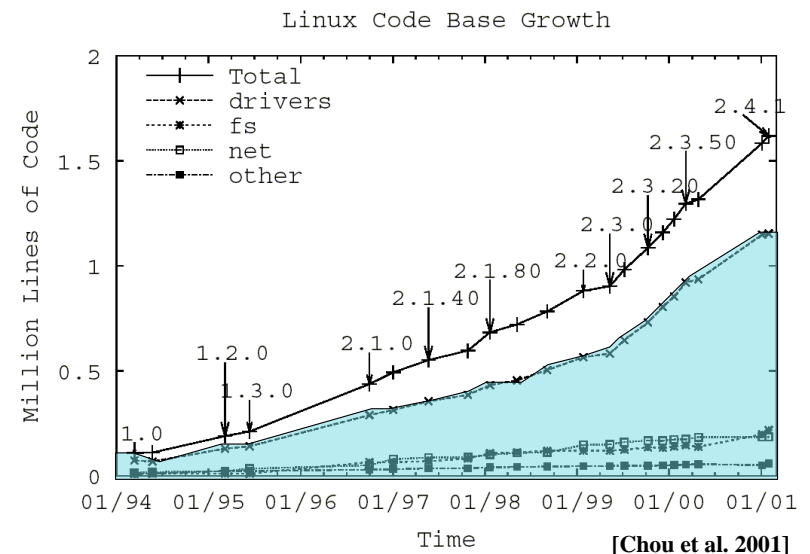
A module that translates high-level I/O requests to device-specific requests

- 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



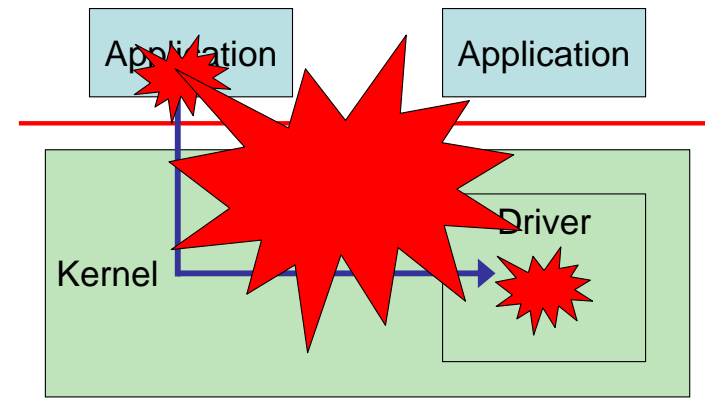
Driver Reality -- Linux



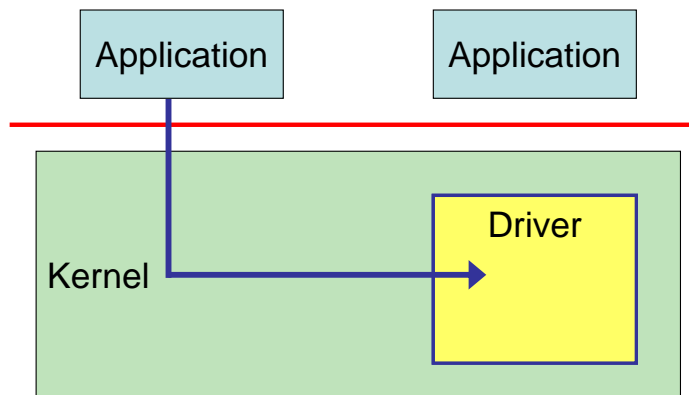
Why Do Drivers Fail?

- 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

OS Today



Our Goal: OS With Reliability



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

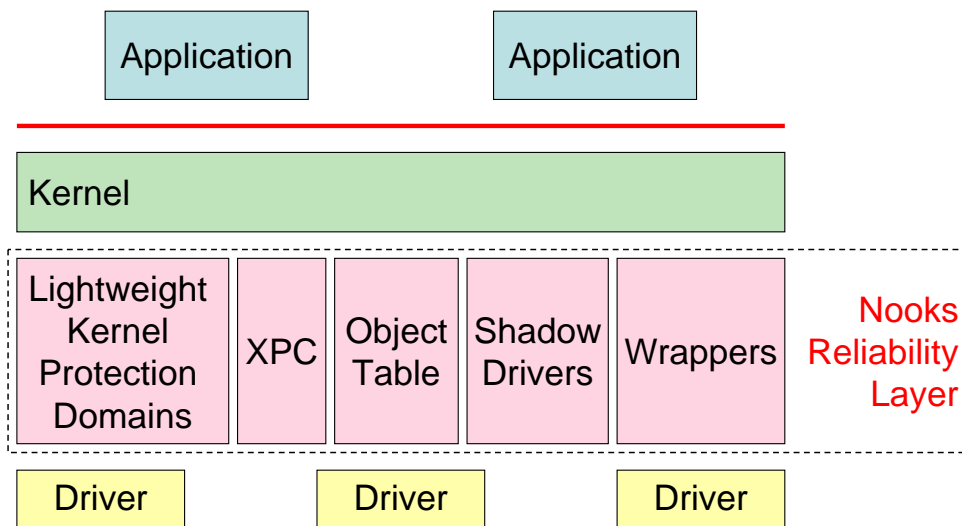
Outline

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

Outline

- Problem
- **Design and Implementation of Nooks**
 - Isolation
 - Recovery
- Evaluation
- Summary

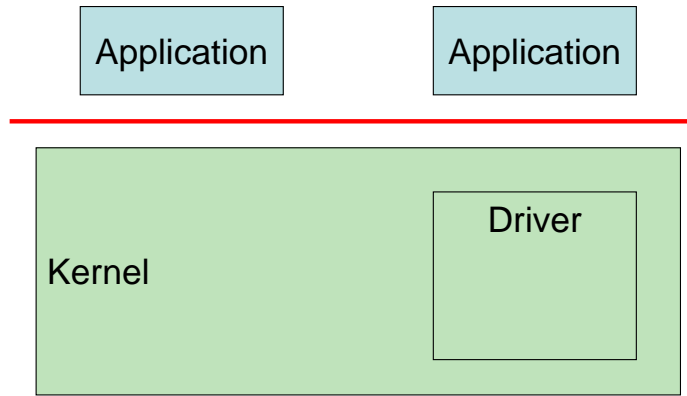
Nooks System Architecture



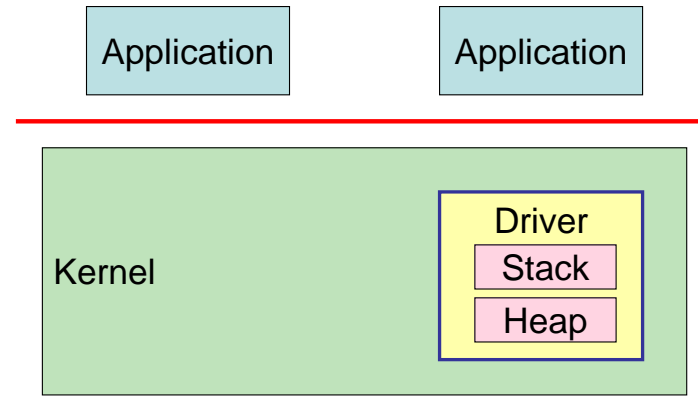
Outline

- Problem
- **Design and Implementation of Nooks**
 - **Isolation**
 - Recovery
- Evaluation
- Summary

Existing Kernels

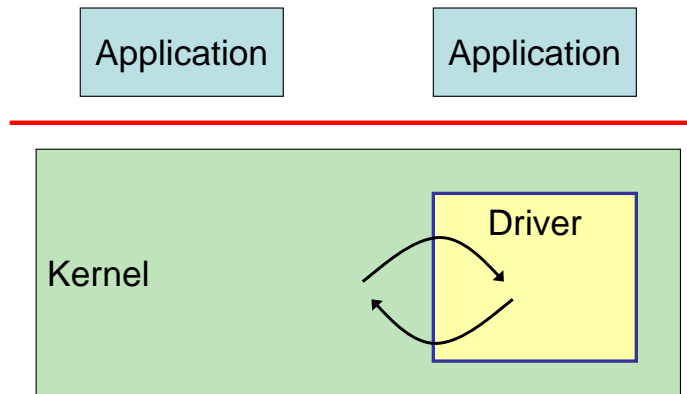


Memory Isolation

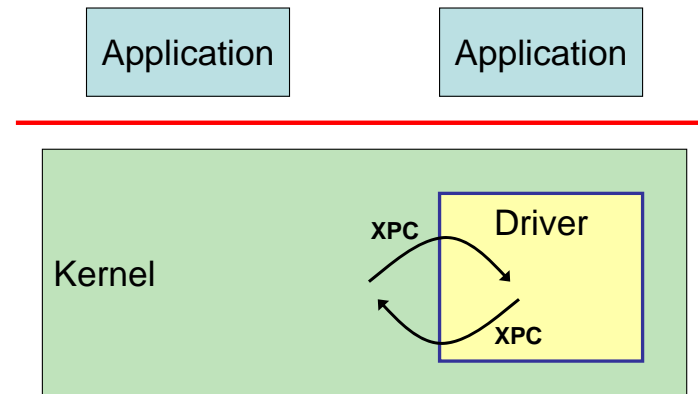


Lightweight Kernel Protection Domains

Control Transfer

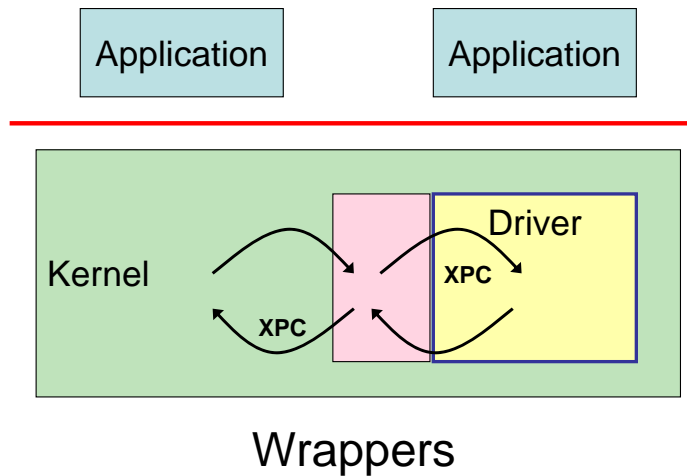


Control Transfer

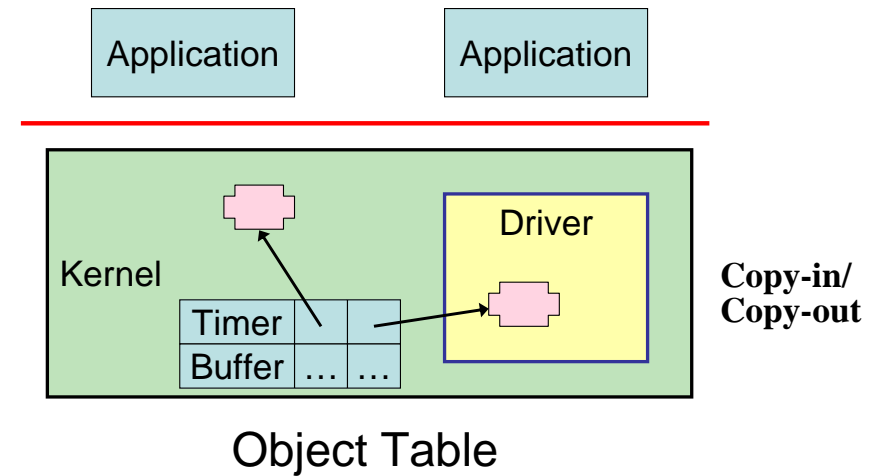


eXtension Procedure Call

Transparency



Data Access



Isolation (recap)

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

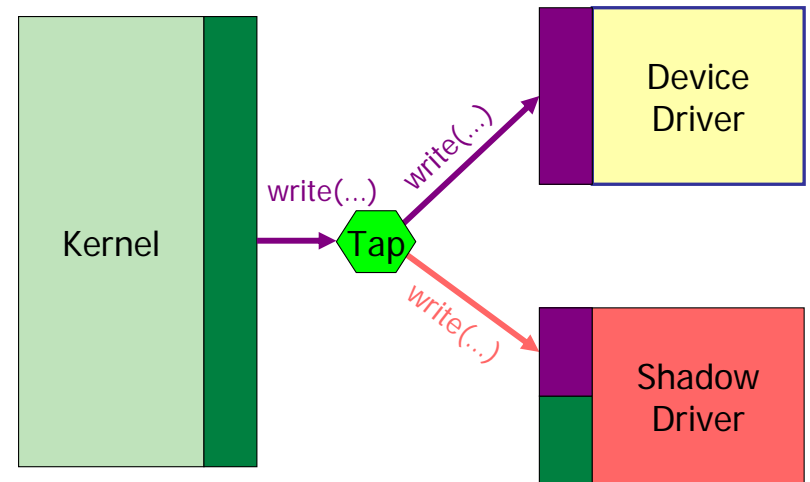
Outline

- Problem
- Design and Implementation of Nooks
 - Isolation
 - Recovery
- Evaluation
- Summary

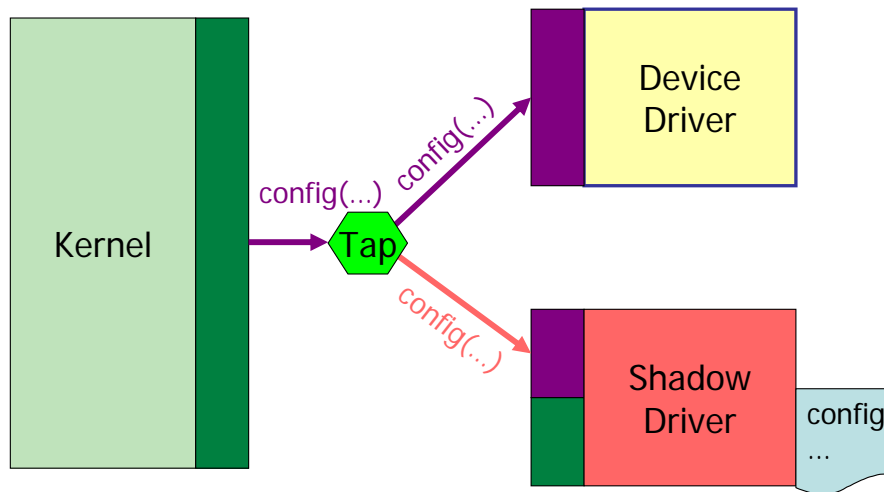
Shadow Drivers

- 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

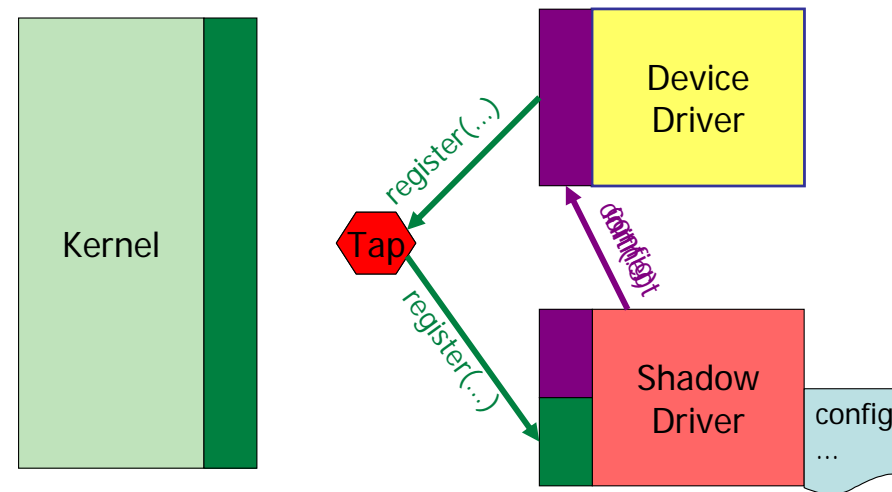
Shadow Driver Overview



Preparing for Recovery



Recovering a Failed Driver



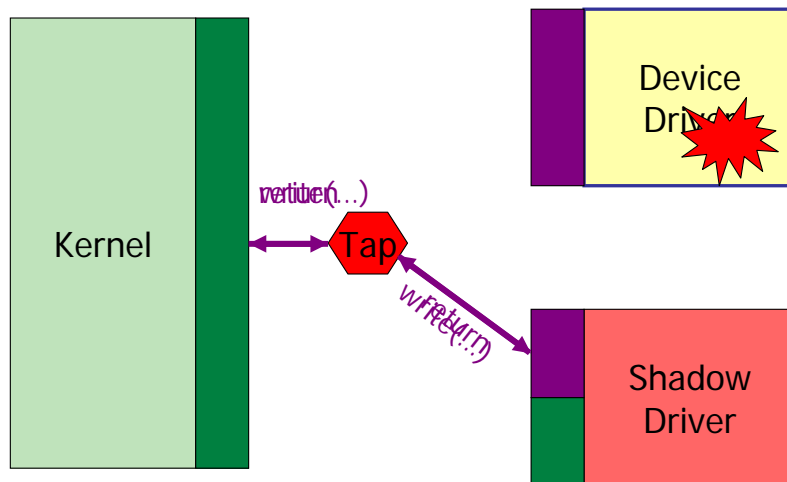
Recovering a Failed Driver

- 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

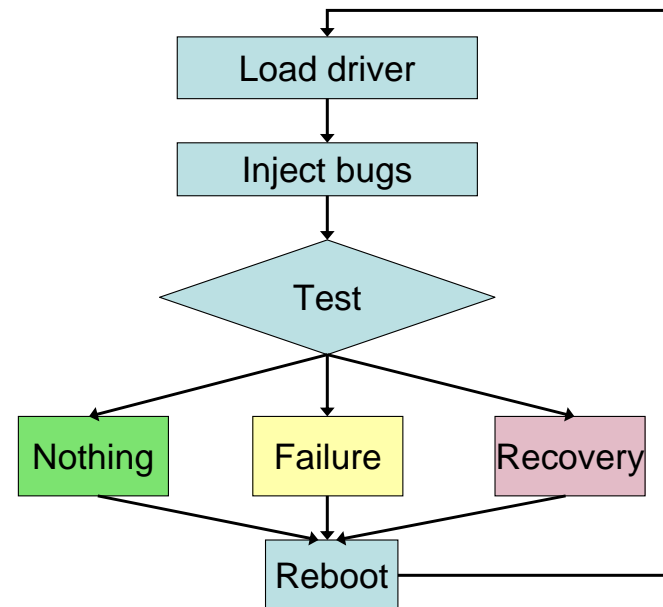
Implementation Complexity

- 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

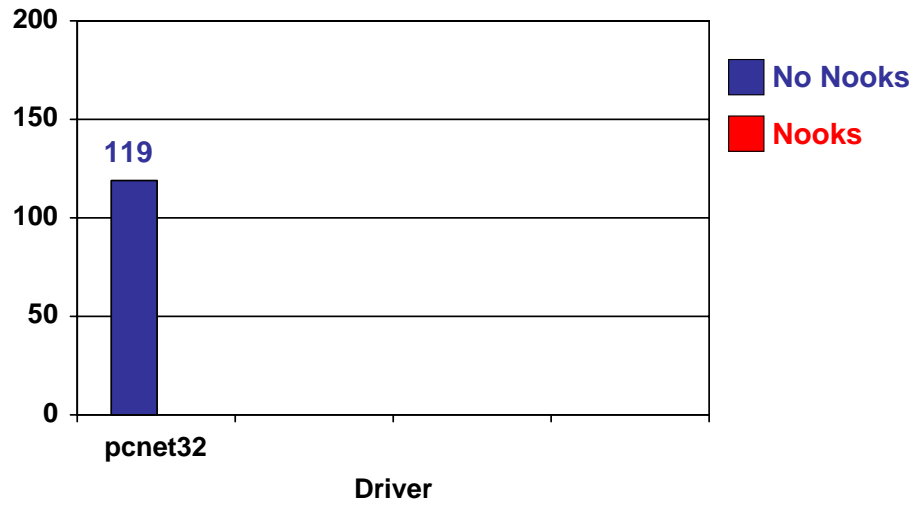
Drivers Tested

Class	Drivers
 Sound	Soundblaster Audigy, Soundblaster 16, Soundblaster Live!, Intel 810 Audio, Ensoniq 1371, Crystal Sound 4232
 Network	Intel Pro/1000 Gigabit Ethernet, AMD PCnet32, Intel Pro/100 10/100, 3Com 3c59x 10/100, SMC Etherpower 100
 IDE Storage	ide-disk, ide-cd

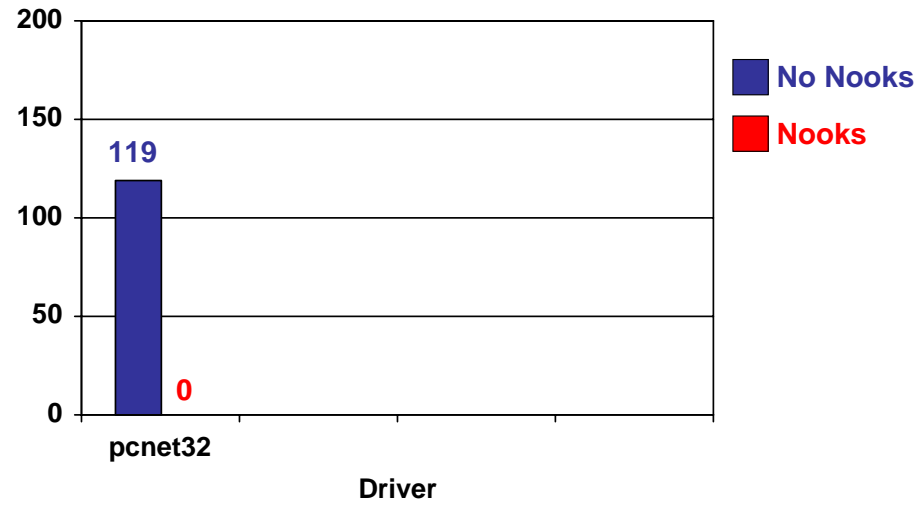
Reliability Test Methodology



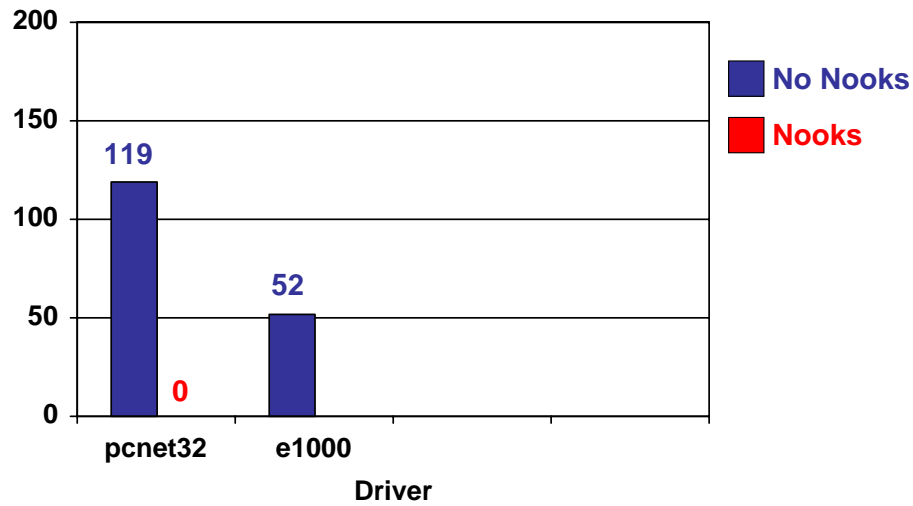
Isolation Works



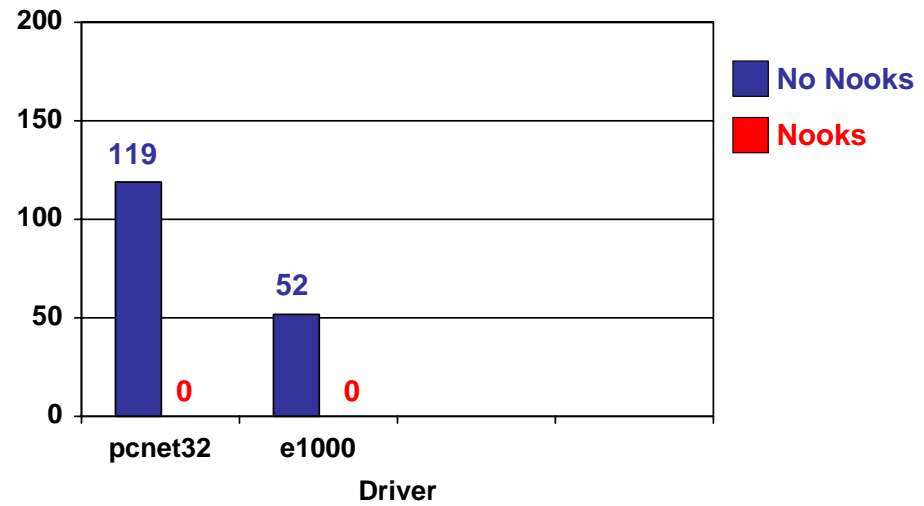
Isolation Works



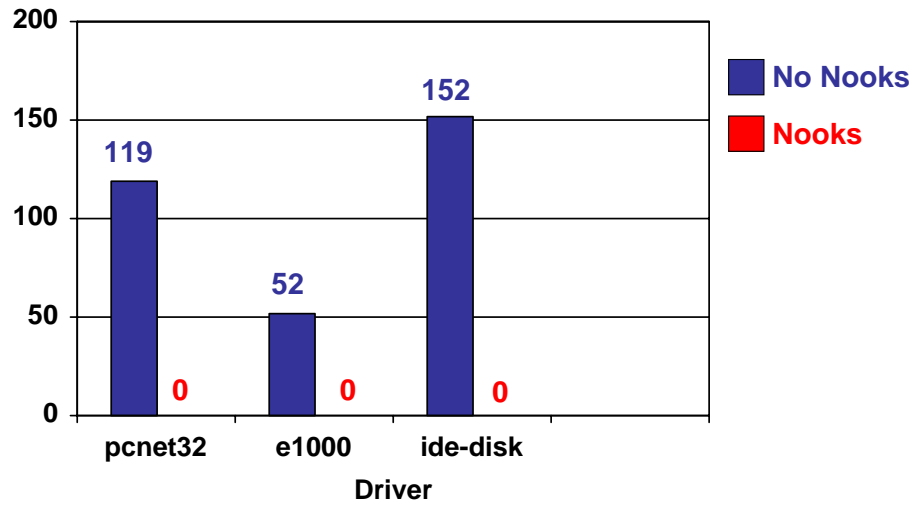
Isolation Works



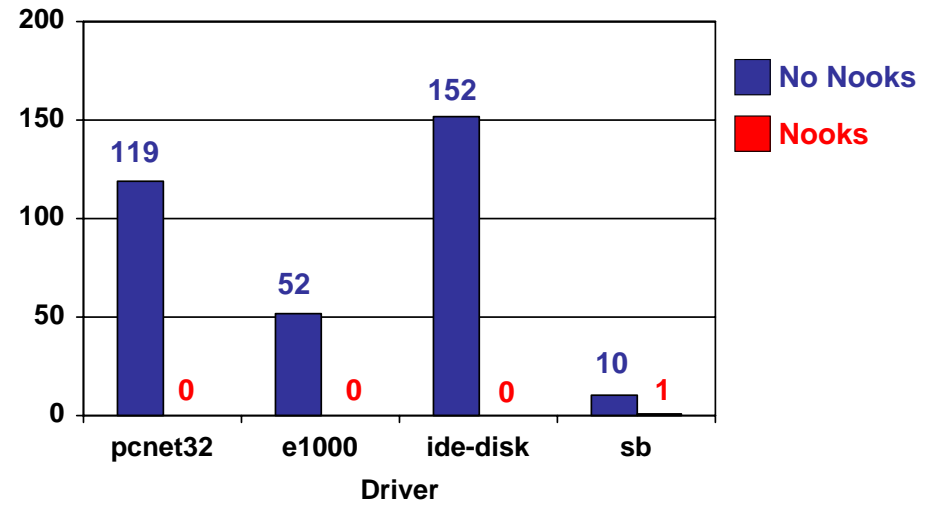
Isolation Works



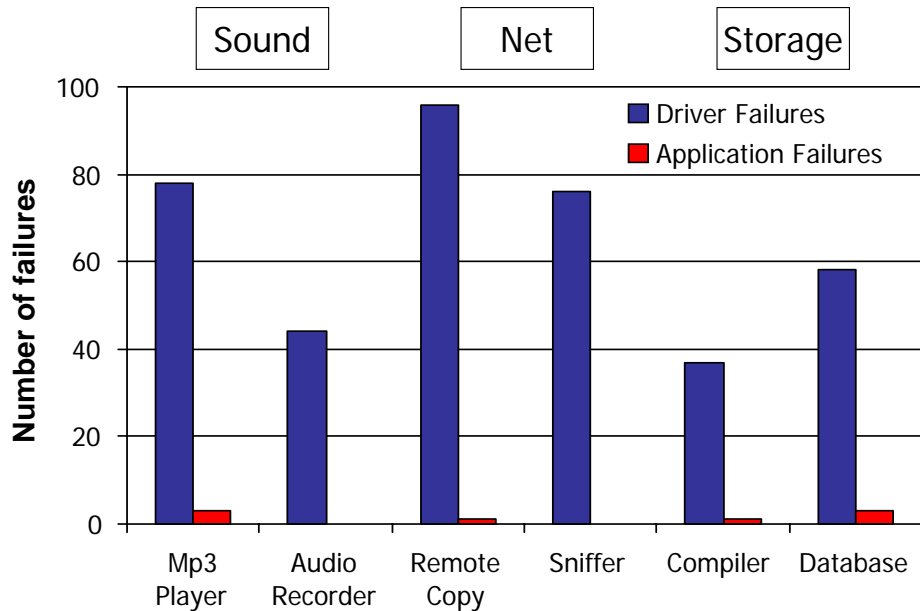
Isolation Works



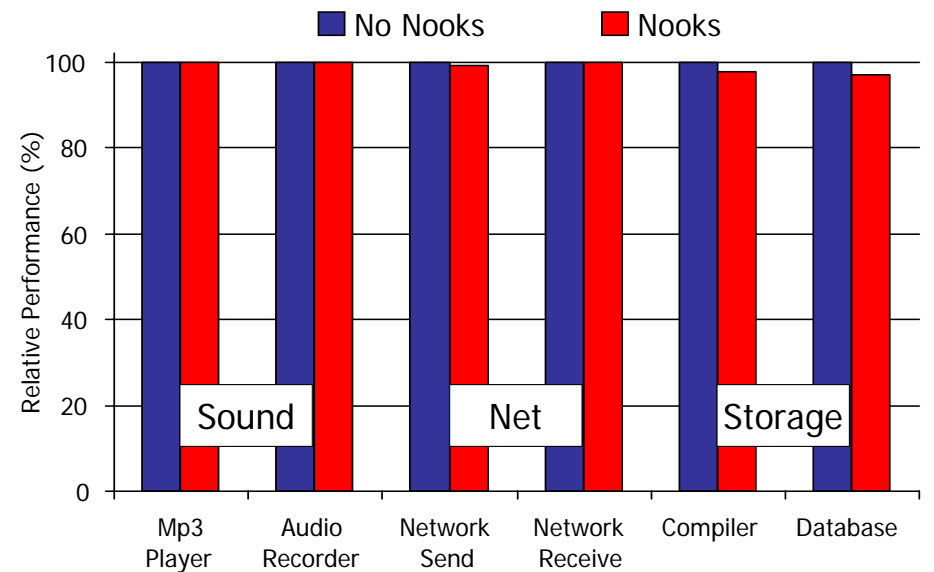
Isolation Works



Recovery Works



Relative Performance

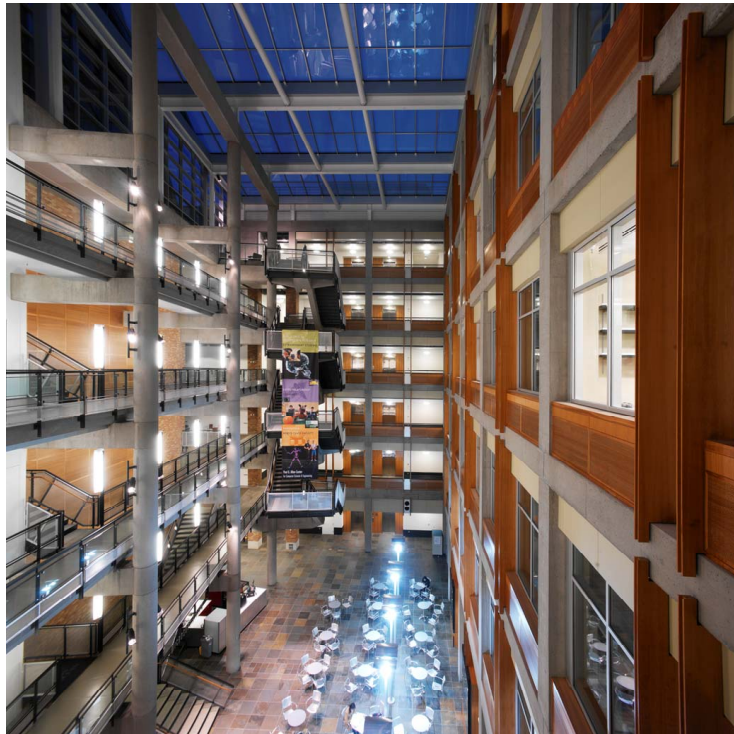


Evaluation: Bottom Line

- 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

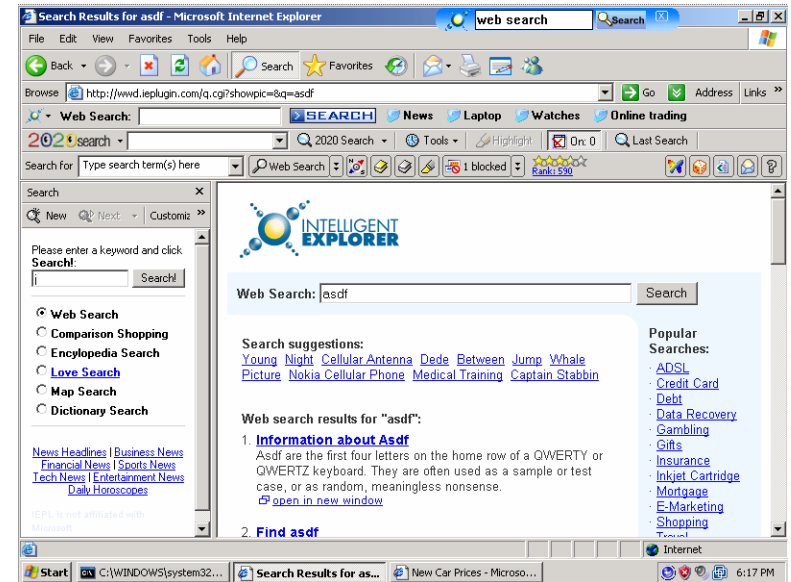
A Crawler-Based Study of Spyware on the Web

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 3rd party
- Characteristics:
 - Installs without user knowledge or consent
 - Hijacks computer's resources or functions
 - Collects valuable information and relays to a 3rd party
 - Resists detection and uninstallation

You know it when you see it



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

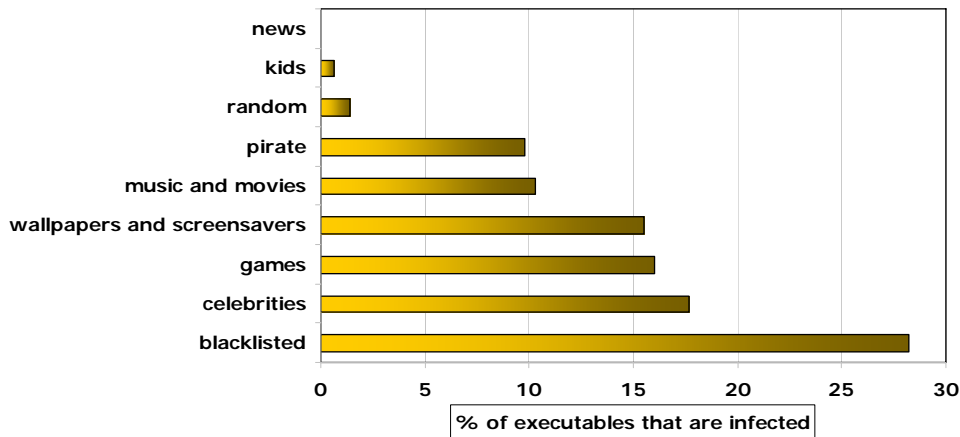
- Web crawler collects a pool of executables
- Analyze each in a virtual machine:
 - Clone a clean WinXP VM
 - Automatically install executable
 - Run analysis to see what changed
 - Currently, an anti-spyware tool (Ad-Aware)
- Average analysis time – 90 sec. per executable

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?



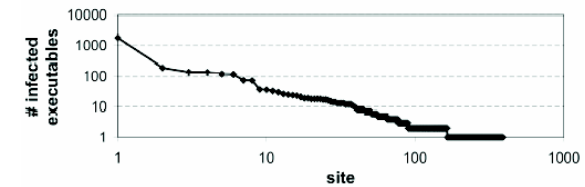
Types of spyware

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

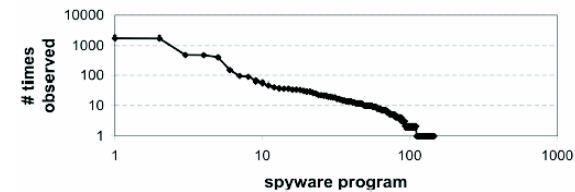
Keylogger	0.05%
Dialer	1.2%
Trojan downloader	12%
Browser hijacker	62%
Adware	88%

Popularity

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

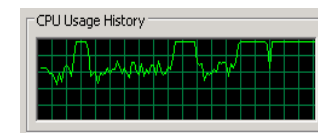


- A small # of spyware variants are responsible for the majority of infections:



Example of a Nasty Executable

- <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>
- 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

Event triggers

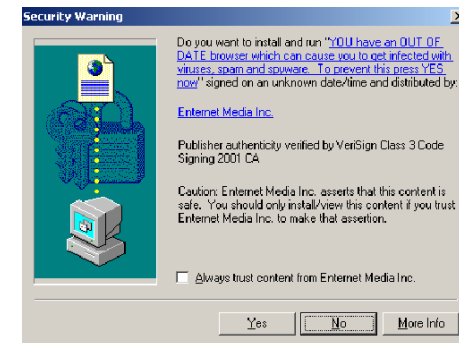
- Real-time monitoring for non-normal behavior:
 - Process creation
 - File events
 - Example: foo.exe written outside IE folders.
 - Registry events
 - Example: new auto-start entry for foo.exe
- No false negatives (theoretically)
- 41% false positives:
 - Legitimate software installations
 - Background noise
 - Spyware missed by our anti-spyware tool

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

More on automatic browsing

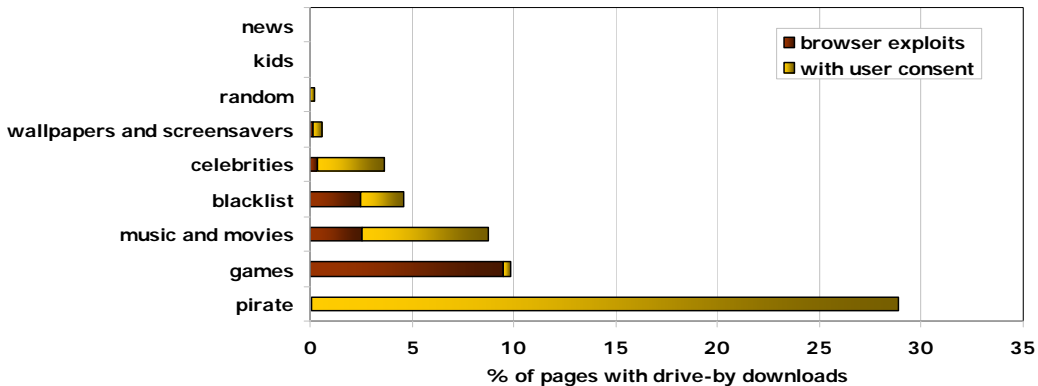
- Caveats and tricks
 - Restore clean state before navigating to next page
 - Speed up virtual time
 - Monitor for crashes and freezes
- Deciding what to say to security prompts:
 - “yes”
 - Emulate user consent
 - “no” (or no prompt)
 - Find security exploits



Drive-by download results

(unpatched Internet Explorer, unpatched WinXP)

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



Types of spyware

- Is drive-by download spyware more dangerous?

	Executables	Drive-by Downloads
Keylogger	0.05%	0%
Dialer	1.2%	0.2%
Trojan Downloader	12%	50%
Browser hijacker	62%	84%
Adware	88%	75%



Is Firefox better than IE?

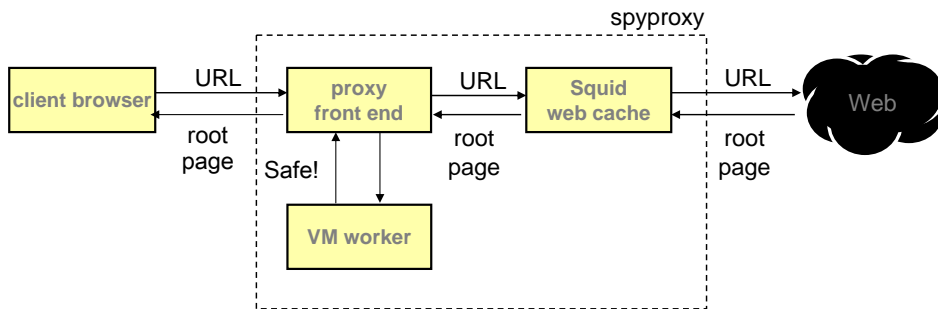
- Repeat drive-by download study with Mozilla Firefox
- Found 189 (0.4%) pages with drive-by downloads
 - All require user consent
 - All are based on Java
 - Work in other browsers
- Firefox is not 100% safe
 - However, much safer than IE

adult	0
celebrity	33
games	0
kids	0
music	1
news	0
pirate	132
random	0
wallpaper	0
blacklist	23
Total:	189

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

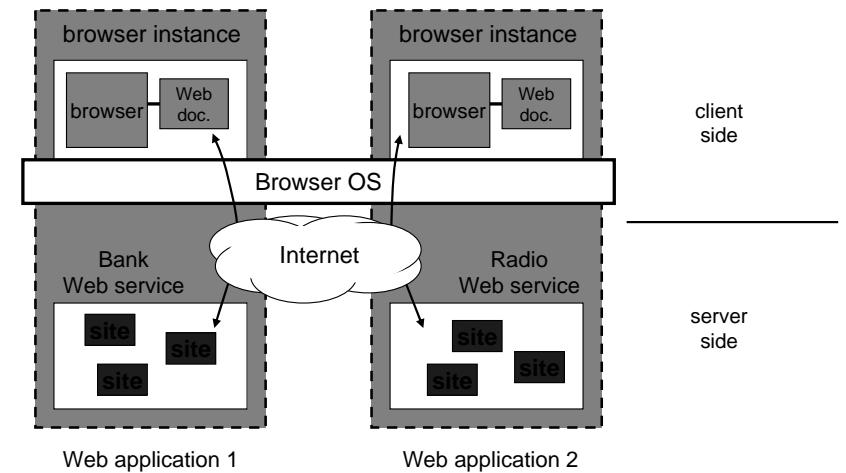
Solution Tidbit 1: Spyproxy



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

Solution Tidbit 2: Tahoma “Browser OS”



Thanks!

- For more info:
 - *Improving the Reliability of Commodity Operating Systems*, Proc. of ACM Symp. On Operating Systems Principles, 2003.
 - *Recovering Device Drivers*, ACM/USENIX Conf. on Operating Systems Design and Impl., 2004.
 - *A Crawler-based Study of Spyware in the Web*, Network and Distributed Systems Security Symp., 2006
 - *A Safety-Oriented Platform for Web Applications*, IEEE Symp. On Security and Privacy, 2006.
- www.cs.washington.edu/homes/levy