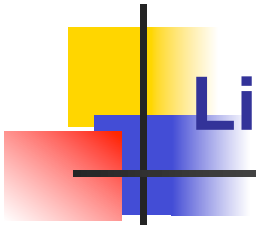
Data on Kernel Failures and Security Incidents

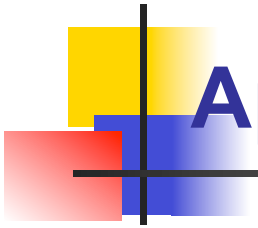
Ravishankar K. Iyer
(W. Gu, Z. Kalbarczyk, G. Lyle, A. Sharma, L. Wang…)

Center for Reliable and High-Performance Computing
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign



Linux on Pentium vs PowerPC G Series

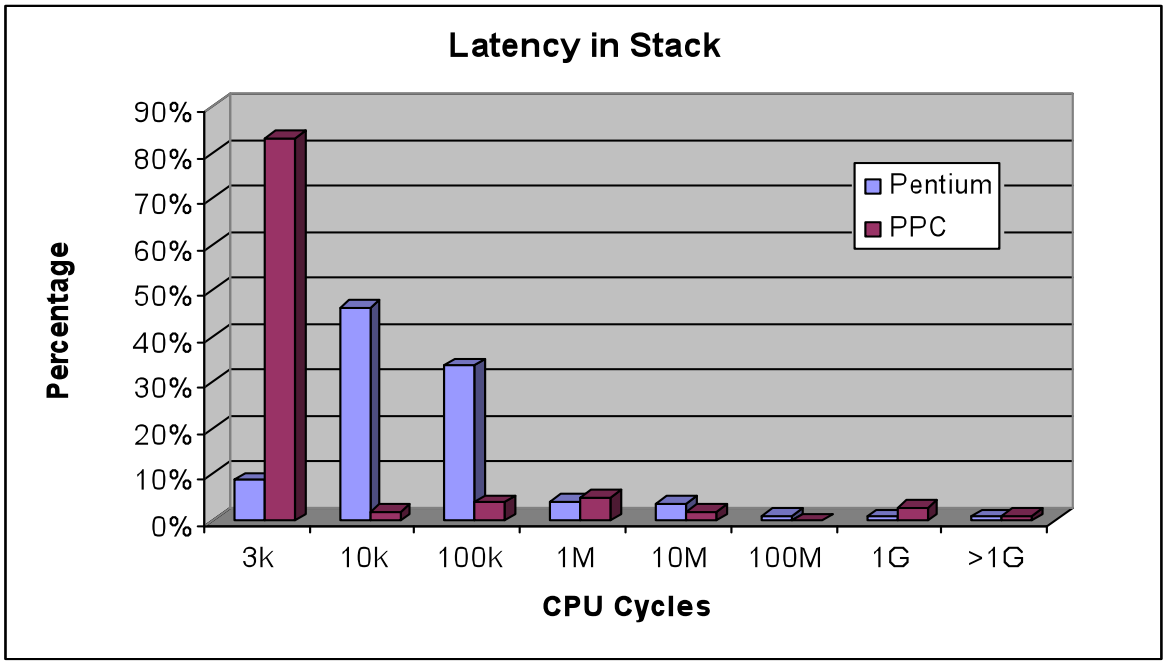
- Analyze Linux kernel responds to transient errors that impact kernel code, kernel data, kernel stack, and processor system registers
  - Target platforms: Pentium 4, PowerPC G4
- How processor hardware architecture (instruction set architecture and register set) impacts kernel behavior in the presence of errors
- Important step in:
  - establishing benchmarking procedure for analyzing and comparing different platforms
  - facilitating analysis of *costs–reliability–performance* tradeoffs in selecting a computing platform

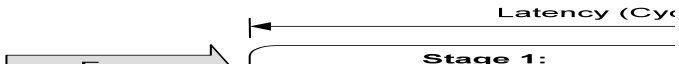


Approach & Error Model

- Automated error injection into the kernel code, kernel data, kernel stack, and processor system registers
  - Over 115,000 errors injected
- Error model
  - Transients emulated by injecting single/multiple/high-level errors into the kernel and CPU registers
  - Error origin not presumed
    - timing issue, noise source, address bus error
  - Error location
    - random within the range for stack/data/register injections or pre-selected (based on profiling) functions for code injection

Latency Distributions for Stack Injection



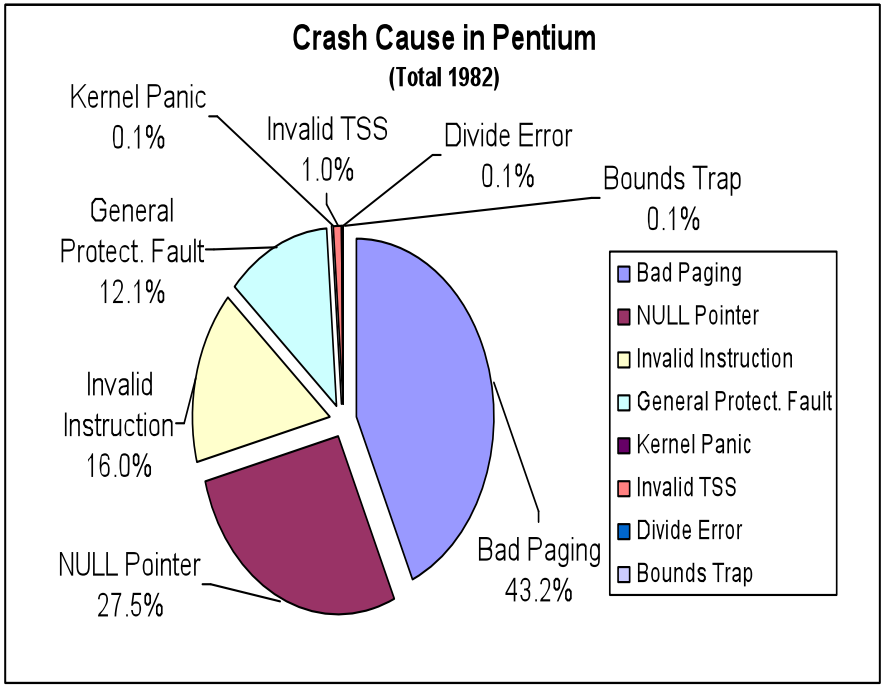


Early detection of kernel stack overflow on PPC major contributor to reduced crash latency

Distributions of Crash Causes

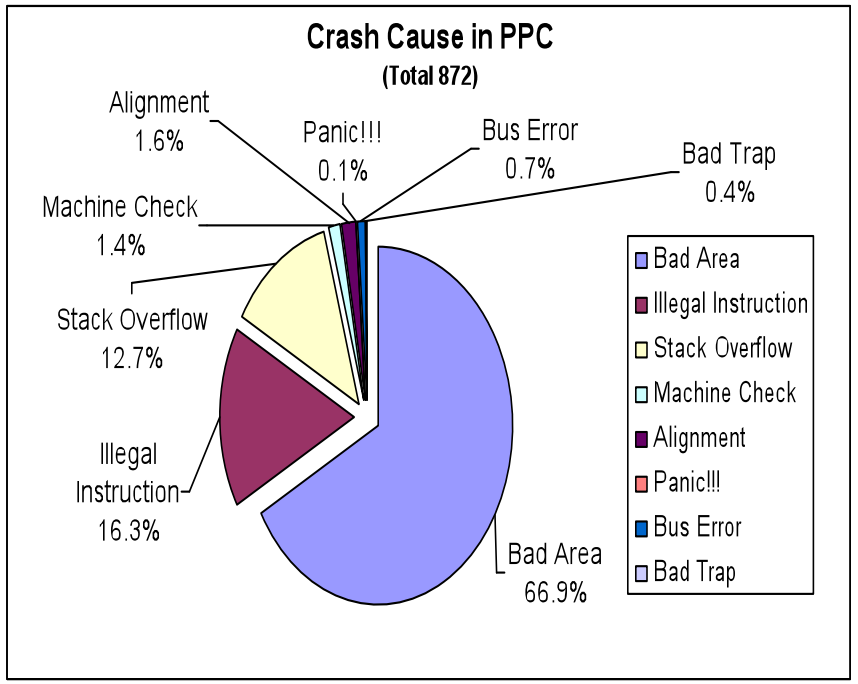
Linux kernel 2.6.2

Intel Pentium

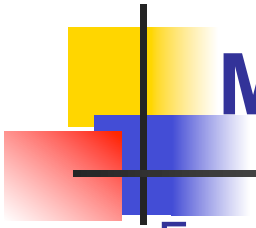


- NULL Pointer: NULL pointer de-reference;
- Bad Paging: Other bad paging except NULL pointer;
- General Protection Fault: Exceeding segment limit;
- Kernel Panic: Operating system detects an error;
- Invalid TSS: Selector, or code segment is outside table limit;
- Bounds Trap: Bounds checking error.

PPC G4

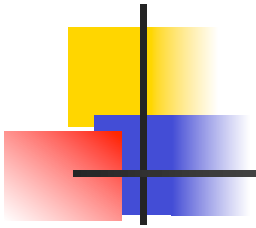


- Bad Area: Bad paging access including NULL pointer;
- Stack Overflow: Stack pointer of a process is out of range;
- Machine Check: Errors on the processor-local bus;
- Alignment: Load/Store operands are not word-aligned;
- Bus Error: Protection faults;
- Bad trap: Unknown exceptions.



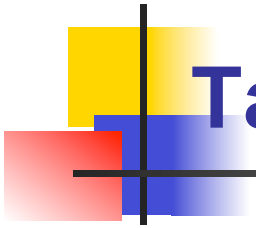
Major Findings

- Error activations similar for both processors – manifestation percentages for Pentium P4 twice as high
  - Stack errors: 56%(P4) vs. 21%(G4); Kernel data: 66% (P4) vs 21%(G4)
- Less compact fixed 32-bit data and stack access makes G4 platform less sensitive to errors
- Pentium processor being very aggressive in providing high performance delays error detection
  - Higher error propagation – crash delays of billion of CPU cycles, when a machine executes in presence of an active error
- Variable length instruction format on P4 allows a bit error to alter a single instruction into a sequence of multiple valid instructions
  - Can lead to poorer diagnosability
  - Can reduce crash latency (fail fast)



Error Behavior Comparison of Multiple Computing Systems:

A case study of Linux on Pentium, Solaris on SPARC, and AIX on POWER



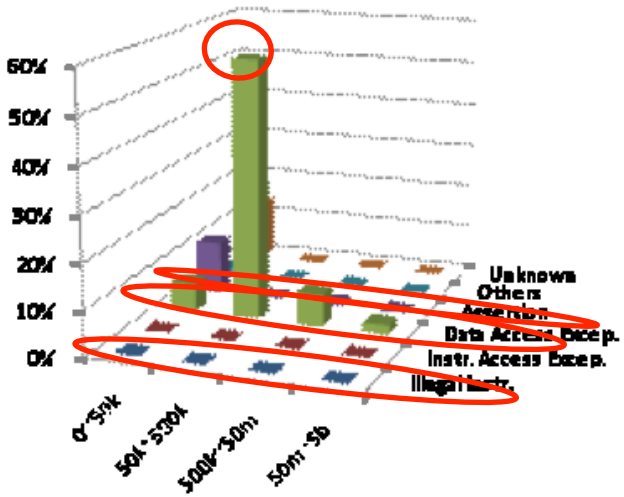
Target Machines

Processor	Hardware		System Software		
	CPU [GHz]	Memory [MB]	Distribution	Kernel	Compiler
Pentium 4	3.00	1024	SUSE 9	2.6.5-7.97	gcc-3.3.3
UltraSPARC IIIi	1.36	2048	Solaris 10	5.10	gcc-3.4.3
POWER 5	1.65	2048	AIX 5.3	TL 05	XL C V8

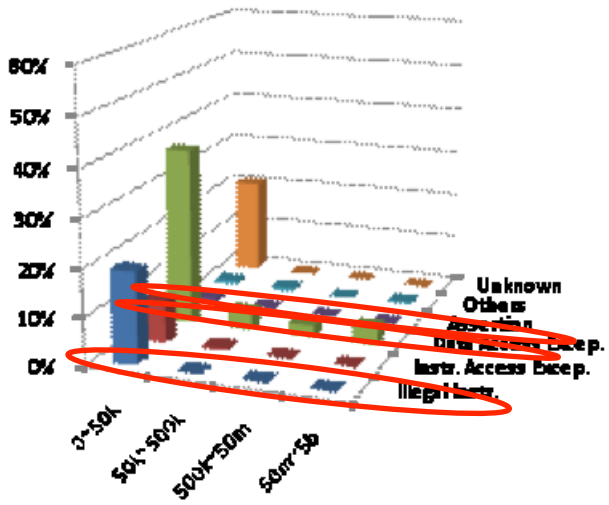
- Workload: Apache Web Server
- Injected to kernel text, stack, and system registers
- More than 27,000 injection
- Average 2~5 min per injection

Text Injection Result

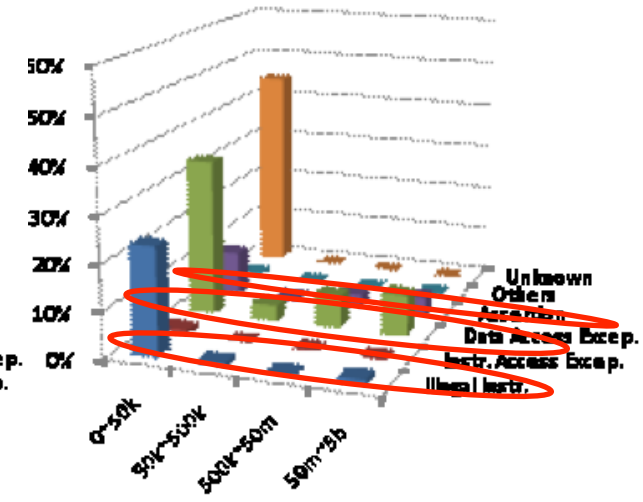
Linux System



Solaris System



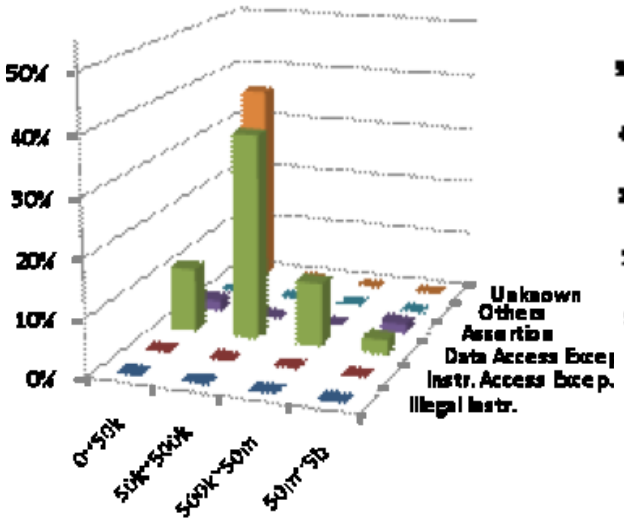
AIX System



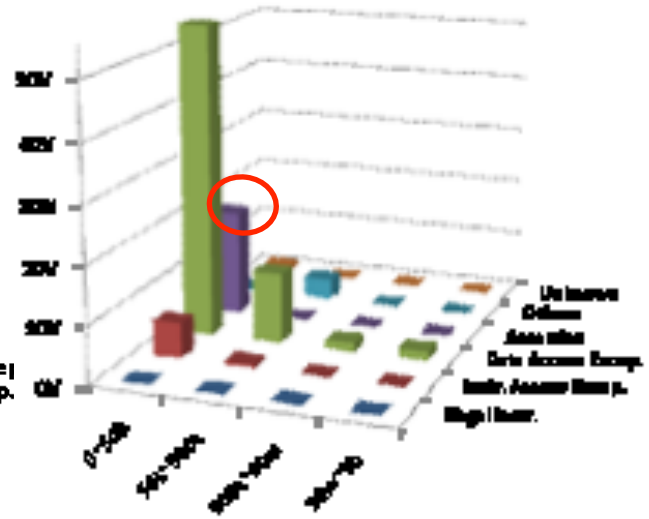
- Most crashes manifest as data access exception
- Linux has no illegal instruction
  - High data access exception
  - Longer error detection latency
- Assertions catch less than 10% crash (Solaris has 1% ~ higher hang?)

Stack Injection Result

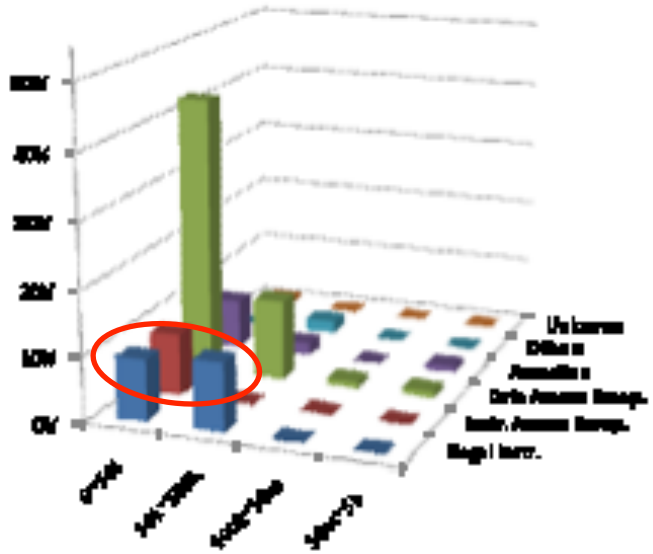
Linux System



Solaris System

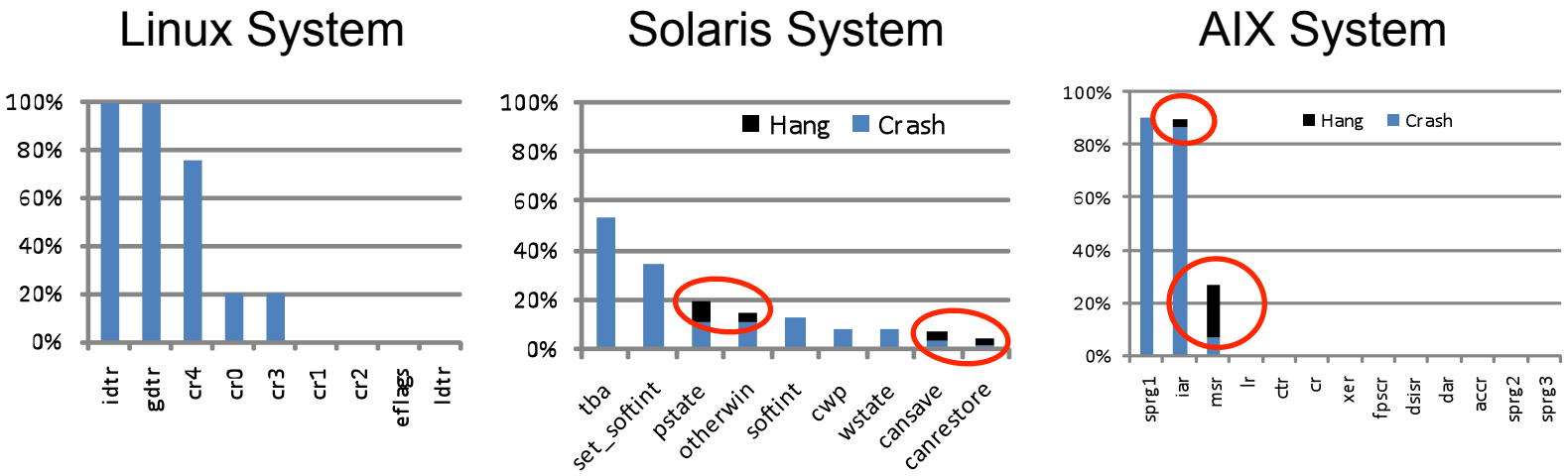


AIX System

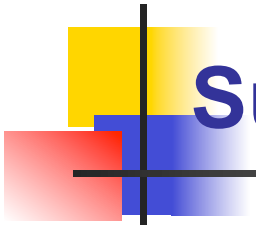


- Solaris: 18% detected by assertions
- AIX: 20% crashes manifest as illegal instruction

Register Injection Result



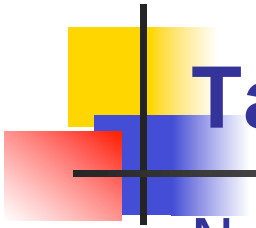
- Only a few registers are error sensitive in each system
  - Can be protected with little overhead
- Solaris and AIX have hang cases



Summary

- Linux System: highest average crash latency (50k~500k instructions)
- Solaris System: highest hang rate (3~7 times more)
- AIX System: lowest error sensitivity and the least amount of crashes in the more severe categories.

Analysis of Security Incidents



Target System

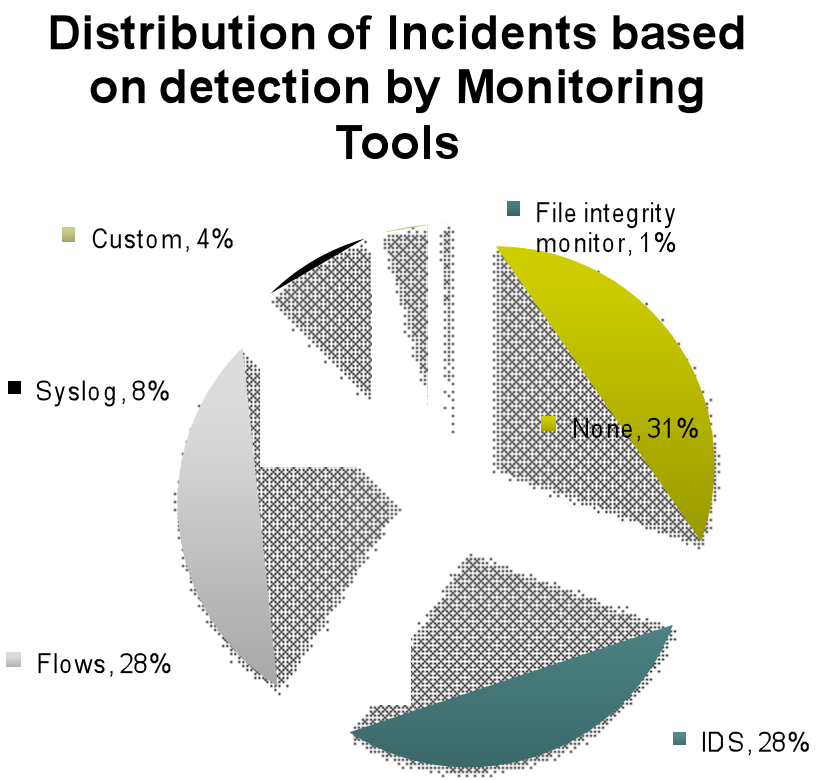
- National Center for Supercomputing Applications (NCSA)
- Number of hosts
  - 5000+ (including Clusters, workstations, laptops)
- Number of Active Users: 6000+
- Monitoring Links: 10Gb pipes
- Monitoring Tools
  - IDS (4.5GB daily logs)
  - Network Flow (2.0G)
  - File integrity check
  - Central Syslog (1.5G)
- OS Types: Linux, AIX, Solaris, OS-X, Windows Etc.

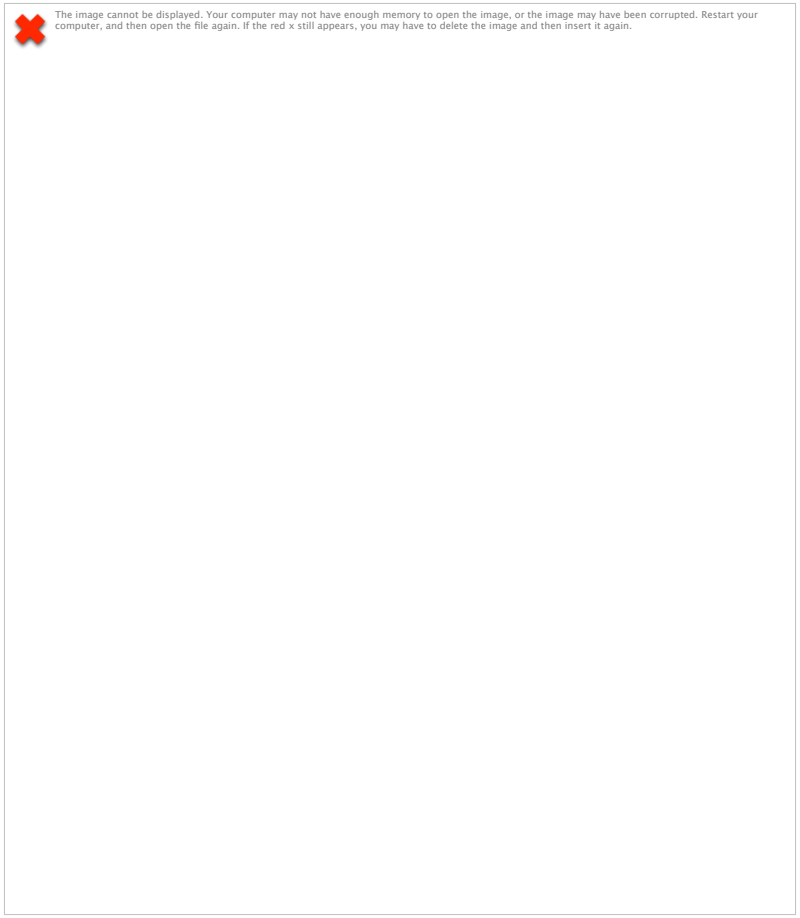
What the Data Shows

- Efficiency of security monitoring based on real incident Data
  - What types of incidents ALL monitors detected and what they missed - reasoning on why they missed it.
- Classification of security incidents – a process to identify incidents based on generated alerts
  - How directly an alert is points to the problem (dubious alerts)
  - Difficulties in distinguishing incident types – determines the type of response required for each incident
  - Types of incidents where one monitoring tool works better to another
- Distribution of incidents based on alerts, monitoring tools
  - Given correct alert, how quickly is it generated to identify the incident
  - Correlation between severity of incidents and types of alerts
- Accuracy of monitors
  - False positive investigation
- State machine models and usefulness

Alert and Incidents

- Number of Incidents: 150 over 5 years (2005 – 2009)
- Monitoring Tools: IDS (4.5GB daily logs), Network Flow (2.0G), File integrity check, Central Syslog (1.5G)





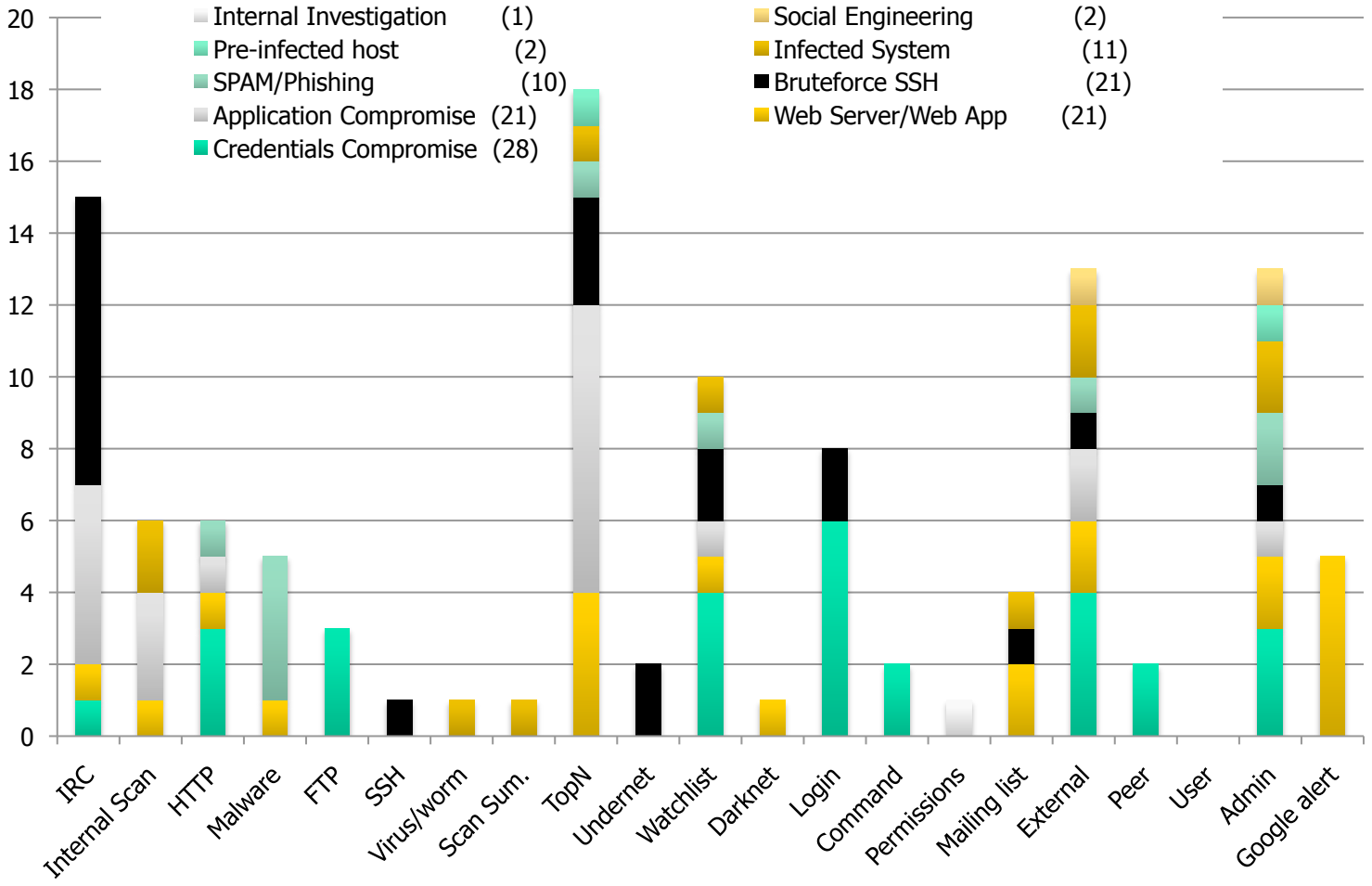
Observations on Attacks

- Attacks primarily targeted towards authentication mechanisms password guessing (bruteforce ssh),
  - password stealing (credentials compromise),
  - exploiting vulnerability (VNC null session),
  - Open X-server key stroke logging
- Use of local root escalation exploits and installing sophisticated root kits and trojaned versions of ssh and sshd for stealing user credentials.
- Often attackers take same or similar steps in order to achieve their goal, irrespective of the vulnerability compromised.

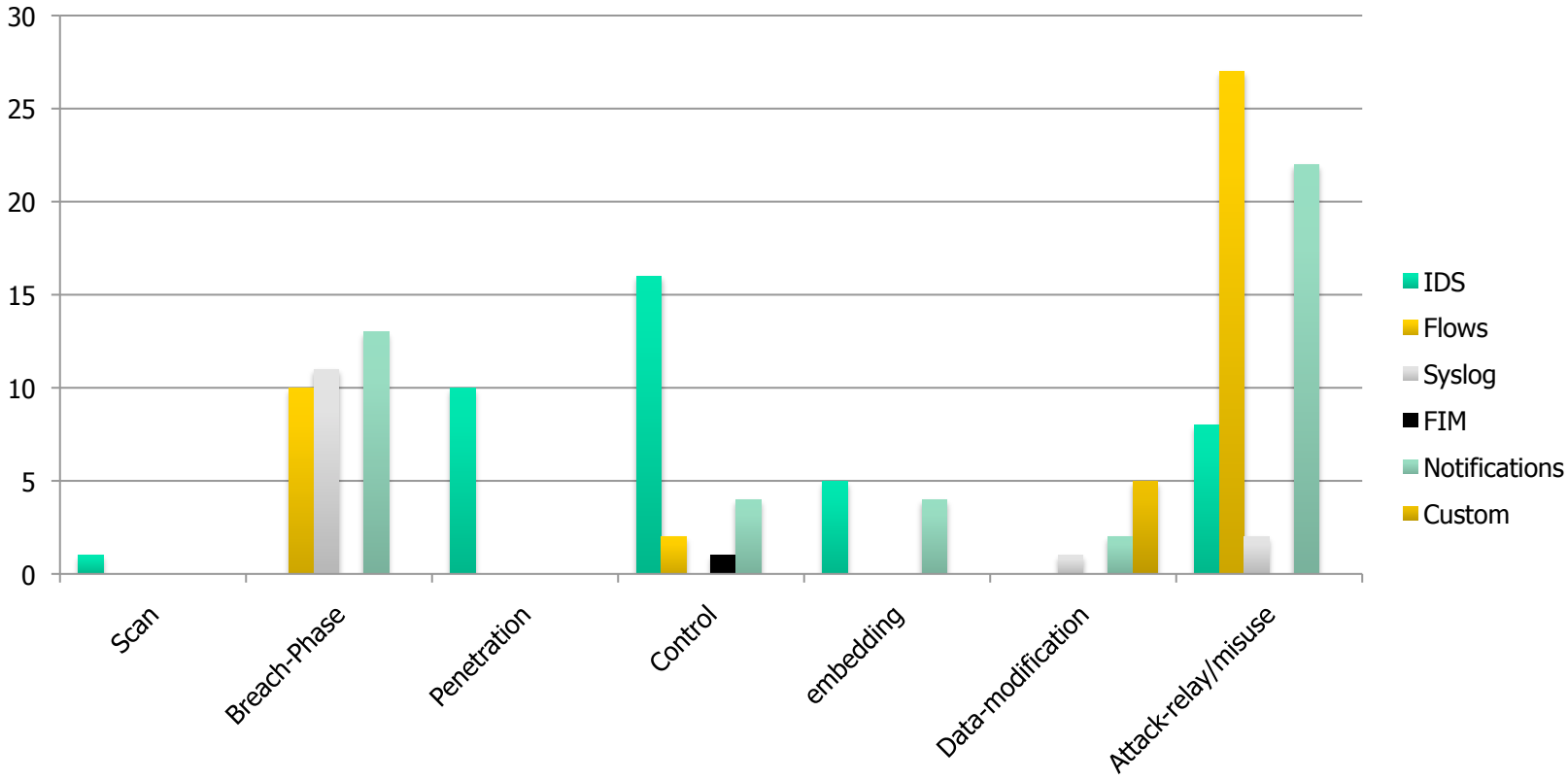
Observations on Attack Detection

- Variety of incidents caught by generic alerts focused on the action of an attacker or behavior of the compromised host.
  - Such alerts (e.g. TopN) caught incidents irrespective of the vulnerability exploited.
- Signature matching and IDS built-in malware detectors are found to be of limited use in detecting malware such as Virus and bot installers.
- Exploit specific alerts requires expertise to develop, deploy, constant maintenance
  - Often can be subverted by the attackers.
- Behavior based detectors a key reason in missed incidents and false positives
- More then 50% of incidents were detected in the last stage of an attack lifecycle, i.e., attack-relay/misuse.
  - Often too late to prevent system/application damage

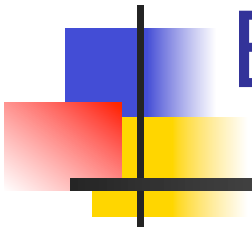
Distribution of Incident Types vs. Alert Types



Stage of Incident Detection



Incident phase v/s Monitoring tool

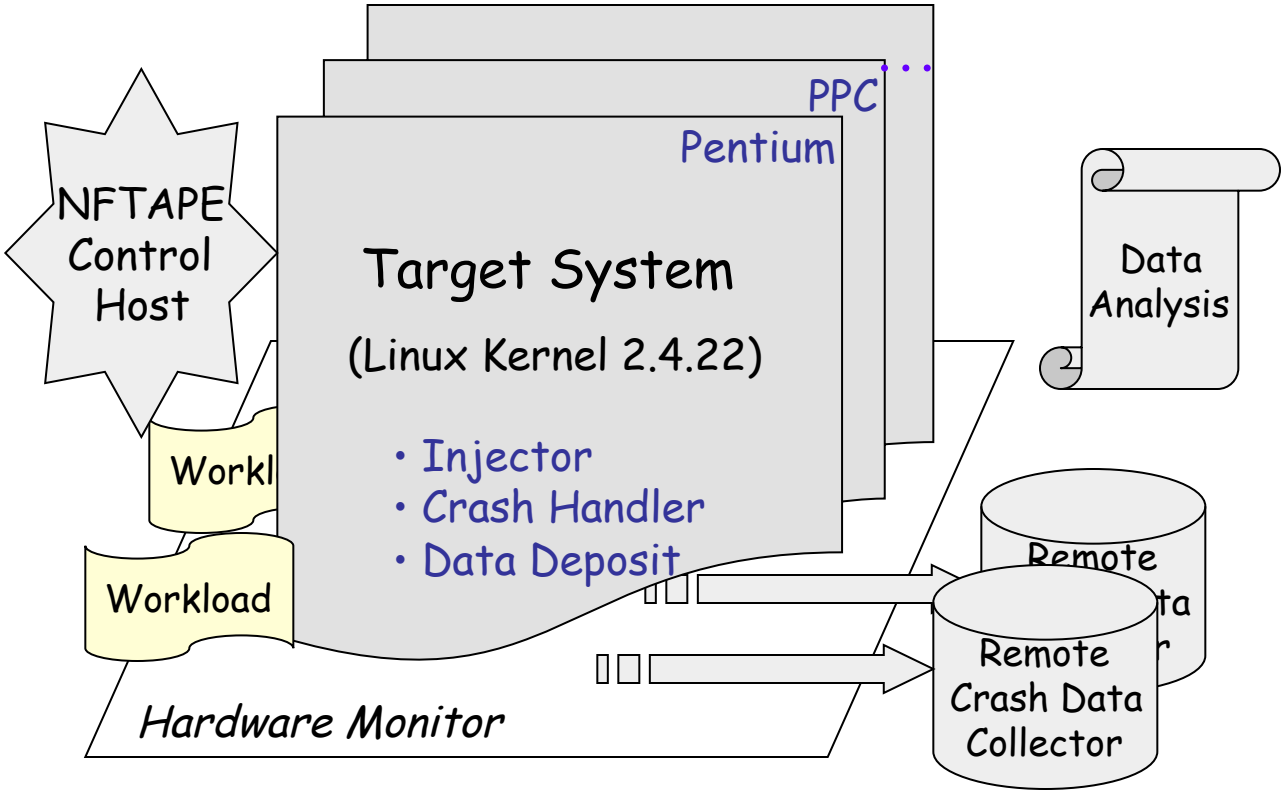


Backup

ILLINOIS

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Error Injection Environment



Processor	Hardware		System Software		
	CPU Clock [GHz]	Memory [MB]	Distribution	Linux Kernel	Compiler
Intel Pentium 4	1.5	256	RedHat 9.0	2.4.22	GCC 3.2.2
Motorola MPC 7455	1.0	256	YellowDog 3.0	2.4.22	

Bit Error causing Change in Instruction Group

	Location	Machine Code	Assembly
Original five instructions	c011e2a6:	**8b** 73 18	mov 0x18(%ebx),%esi
	c011e2a9:	85 f6	test %esi,%esi
	c011e2ab:	74 33	je c011e2e0 <do_fork+0x4a0>
	c011e2ad:	8b 43 24	mov 0x24(%ebx),%eax
	c011e2b0:	85 c0	test %eax,%eax
New five instructions	c011e2a6:	**83** 73 18 85	xorl $0xffffff85, 0x18(%ebx)
	c011e2aa:	f6 74 33 8b	div 0xffffff8b(%ebx,%esi,1),%al
	c011e2ae:	43	inc %ebx
	c011e2af:	24 85	and $0x85, %al
	c011e2b1:	c0	(bad) ← Crash here

One bit flip in Pentium code stream changes a group of instructions to another group of instructions which are much easier to crash