

Overview of Secure Systems Lab

R. Sekar

Secure Systems Lab

Stony Brook University

Secure Systems Lab Facts

- Average 12 graduate/students plus 1 Postdoc/visiting scholar over past few years
 - Roughly equal number of PhD and MS
 - Most students supported as RAs
- Systems and practice-oriented research
 - Built and released several open source packages over past several years
- Very visible in top systems security forums
 - ~4 papers/year in conferences with ~15% acceptance rate

Research Experience with Industry

- Funded projects with several industry partners
 - Telcordia Technologies
 - Computer Associates
 - Global Infotek
 - Dolphin Technologies
 - ImmUNET Security
 - ASPEngines
- Total funding \$1M over 5 years
- 5 years experience working in industry

Overview of Research at Security Lab

- Applying concepts from compilers, operating systems and formal methods for cyber attack:

- Detection **1997**
- Prediction **1998**
- Containment **2000**
- Prevention **2003**
- Recovery **2004**
- Response **2005**



Application Layer Anomaly Detection

- Developed a novel anomaly detection technique for programs, based on learning finite-state automata by observing program behaviors
- Formed the basis of a host-based intrusion prevention system developed by Sana Security
 - Received 3rd round venture funding based on this technique
- Formed the baseline of modern application layer anomaly detection techniques developed over past 5 years

Randomization-based defense against Buffer Overflow Attacks

- Developed address obfuscation (aka address-space randomization), the first comprehensive defense against all memory corruption attacks
 - Almost zero performance overhead
- Implemented it on Linux
- Technology has now found its way into many Linux and FreeBSD distributions
 - Incorporated to a limited degree on Windows Vista
- Recently worked with industry partner to implement this technique on Windows

Safe-execution environments

- Efficient and practical approach to try out unsafe system changes
 - Software updates
 - Configuration changes
 - Vulnerability testing
 - Safe-execution of untrusted code
- Implemented into a downloadable tool ***Alcatraz*** that works with arbitrary binaries on Linux
- Best paper award, ACSAC 2003
 - A few years later, another paper building on this idea won best paper award again!

Other Recent Projects

- Model-Carrying Code
 - A *practical* framework for code producers and consumers to collaborate to achieve security
 - One of the first approaches to address *policy development* problem
- Automated generation of attack signatures
 - Provides the benefits of patches (immunity to attacks without losing legitimate requests or application crash) while eliminating drawbacks (long delays for availability, compatibility problems)
- A secure platform for cybersecurity experiments
 - Support malware and hacking projects while fully protecting underlying computing platform

Defenses for the Next Generation of Attacks

R. Sekar

Secure Systems Lab

Stony Brook University

Example: SQL Injection

- Attacker-provided data used in SQL queries

```
$cmd = "SELECT price FROM products WHERE  
                                name='\" . $name . \"'"
```

... Use cmd as an SQL query

- **Attacker-provided name:**

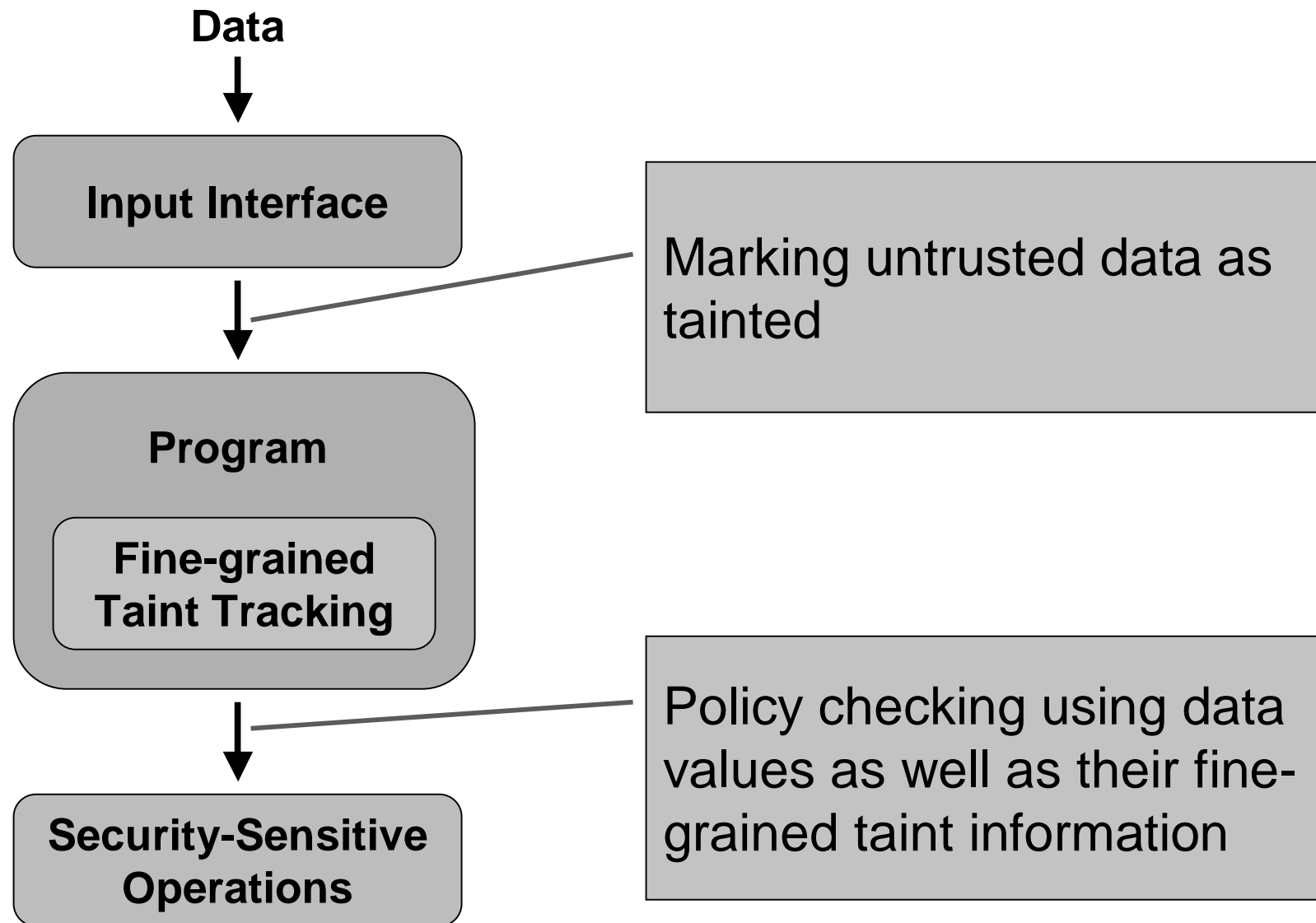
- xyz'; UPDATE products SET price=0 WHERE
name='OneCaratDiamondRing

- Resulting query

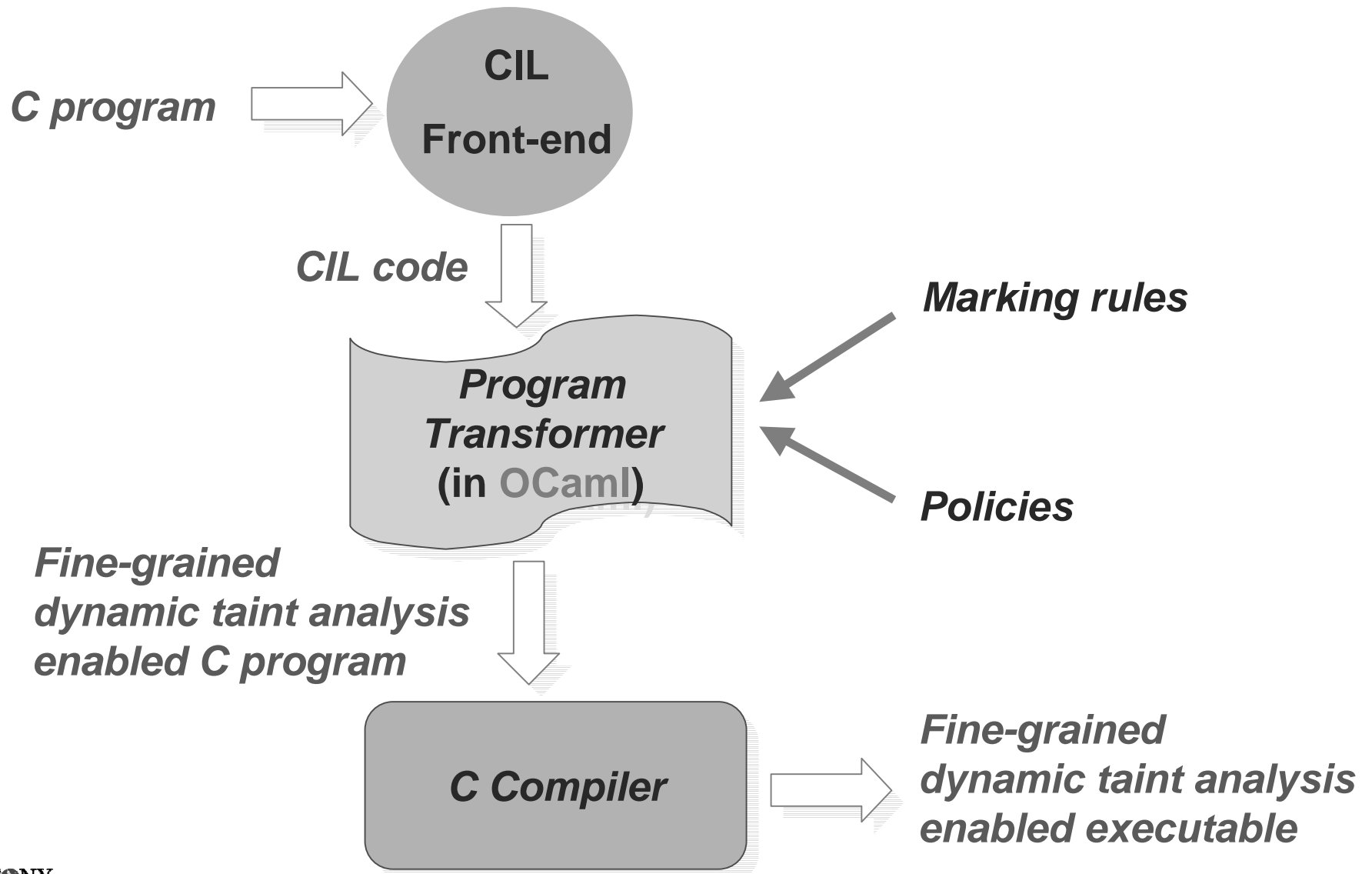
```
SELECT price FROM products WHERE name='xyz';  
UPDATE products SET price=0 WHERE  
name='OneCaratDiamondRing'
```

- Policy for detection: No tainted metacharacters (or SQL commands) in a string used as SQL query

Approach: Taint-Enhanced Policy Enforcement



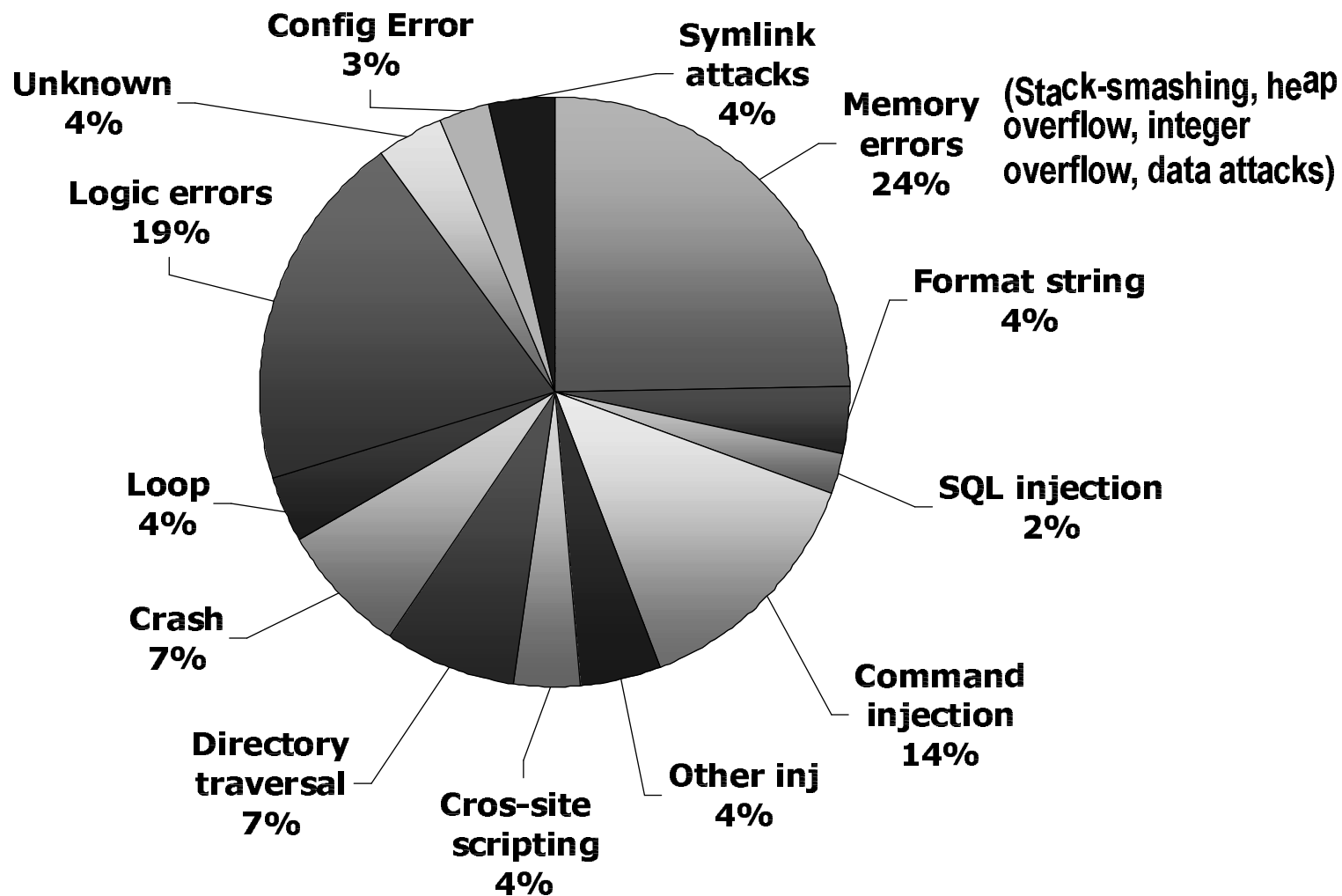
Implementation



Attacks used in Evaluation

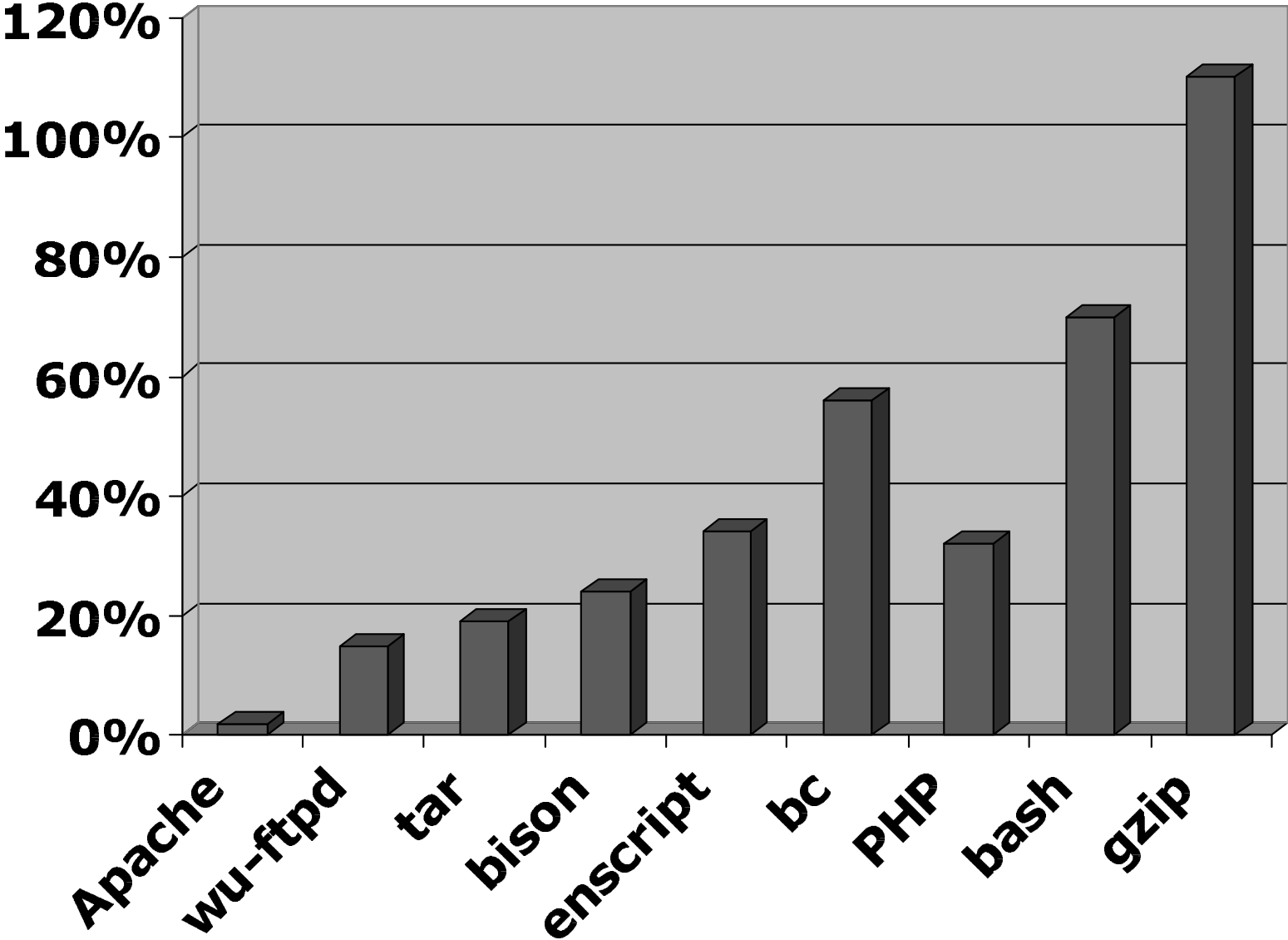
CVE#	Program	Language	Attack type
CAN-2003-0201	samba	C	Stack smashing
CVE-2000-0573	wu-ftpd	C	Format string
CAN-2005-1365	pico server	C	Directory traversal
CAN-2003-0486	phpBB 2.0.5	PHP	SQL injection
CAN-2005-0258	phpBB 2.0.5	PHP	Directory traversal
CAN-2002-1341	SquirrelMail 1.2.10	PHP	Cross site scripting
CAN-2003-0990	SquirrelMail 1.4.0	PHP	Command injection
CAN-2005-1921	PHP XML-RPC	PHP, C	Command injection
CVE-1999-0045	nph-test-cgi	BASH	Shell meta-character

CVE Vulnerabilities, 2003 and 2004



Non-specific attacks (logic, config errors and unknown attacks) are the only ones that aren't handled

Performance Overheads



Related Work

- Information flow

- Coarse granularity: track at the level of entire programs or at the level of individual variables
 - Too coarse to reliably detect any of these attacks!
- Implicit flows produce excessive false positives

Ours is the first approach to detect all these attacks in a unified framework

- Attack detection on web apps

- Static taint analysis [Huang et al 04, Livshits et al 05]
 - Cannot distinguish benign dependencies from vulnerabilities!

Dynamic tracking [Tuong et al 05, Pietraszek 05]

- Parallel development to our work

Proposed Work

- Optimization to further improve performance
- Development of a policy language and runtime infrastructure for policy enforcement
- Experimental evaluation with large web applications

Summary

- Unified approach to stop a variety of attacks exploiting software implementation errors
- Reasonable performance
- Potential application to strengthen access control policies (MAC or DAC)
- Decoupling security policies from software implementation details can provide increased assurance
- For more information, see our USENIX Security '06 paper (available from <http://seclab.cs.stonybrook.edu/pubs.htm>)