Hot-Patching a Web Server: a Case Study of ASAP Code Repair

Mathias Payer*, Thomas R. Gross

Department of Computer Science

ETH Zurich

* now at UC Berkeley

Security Dilemma

Integrity and availability threatened by vulnerabilities

Two remedies: update or sandboxing

- Security updates fix known vulnerabilities but require service restart
- Sandboxes protect from unknown exploits but stop the service when an attack is detected

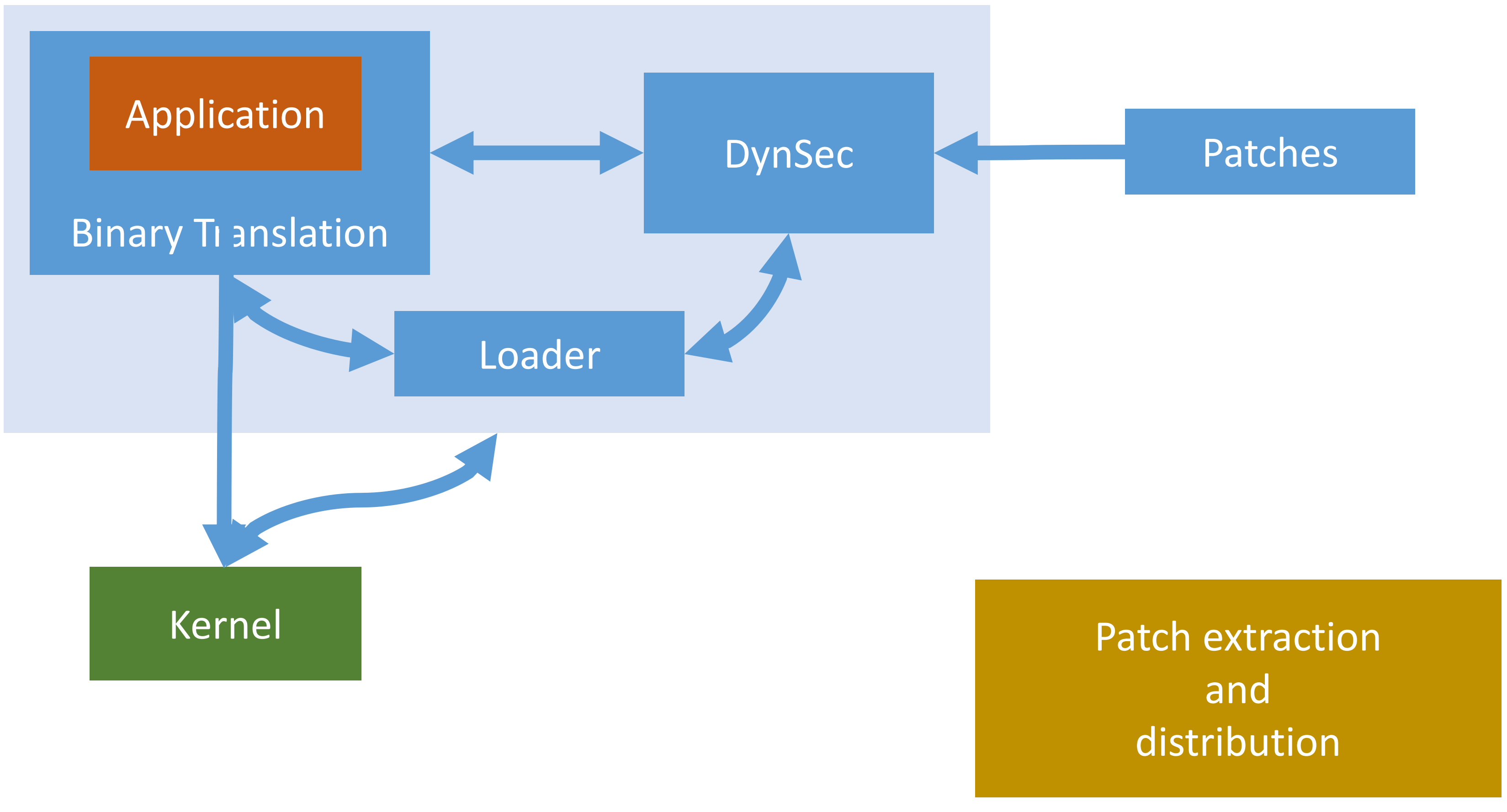
DynSec in 1 Minute

Key insight: both ***sandboxes*** and ***dynamic update mechanisms*** rely on some form of ***virtualization***

Binary Translation (BT) provides virtualization

- Sandbox protects integrity
- Dynamic update mechanism protects availability

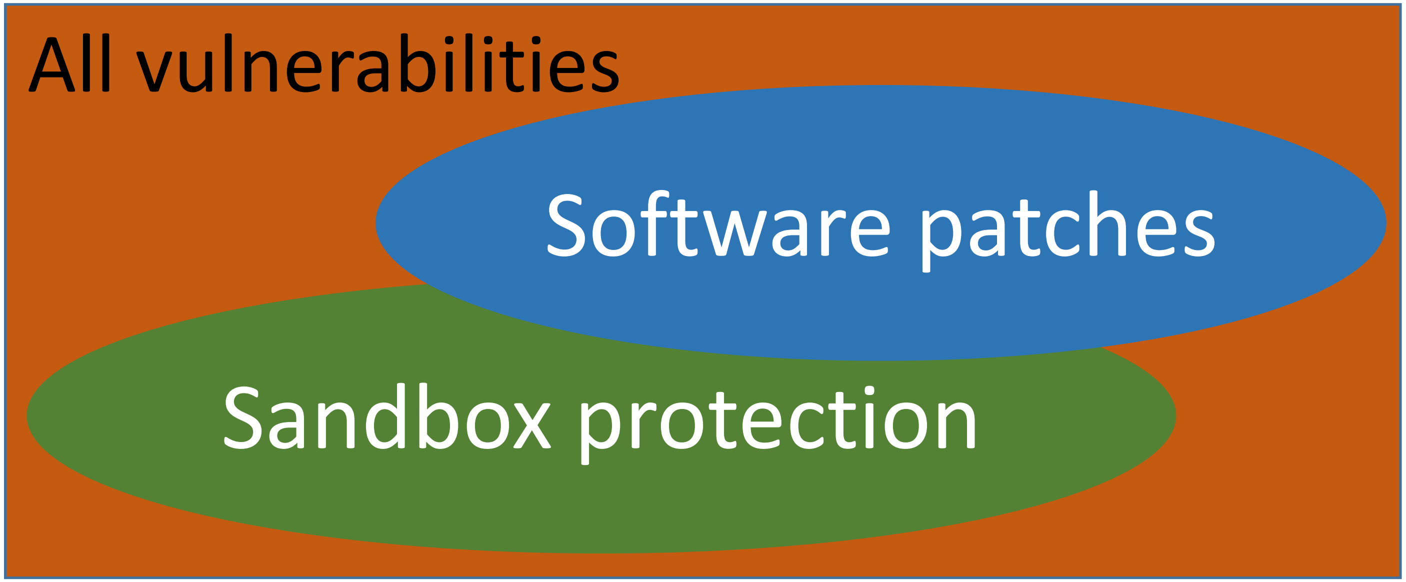
DynSec in 2 Minutes



Hot-Patching a Web Server

Analyze all security patches of Apache 2.2

- From Dec 1st 2005 to Feb. 18th 2013
- Total of 49 security bugs, most are simple
- Many different classes of bugs



Outline

Motivation

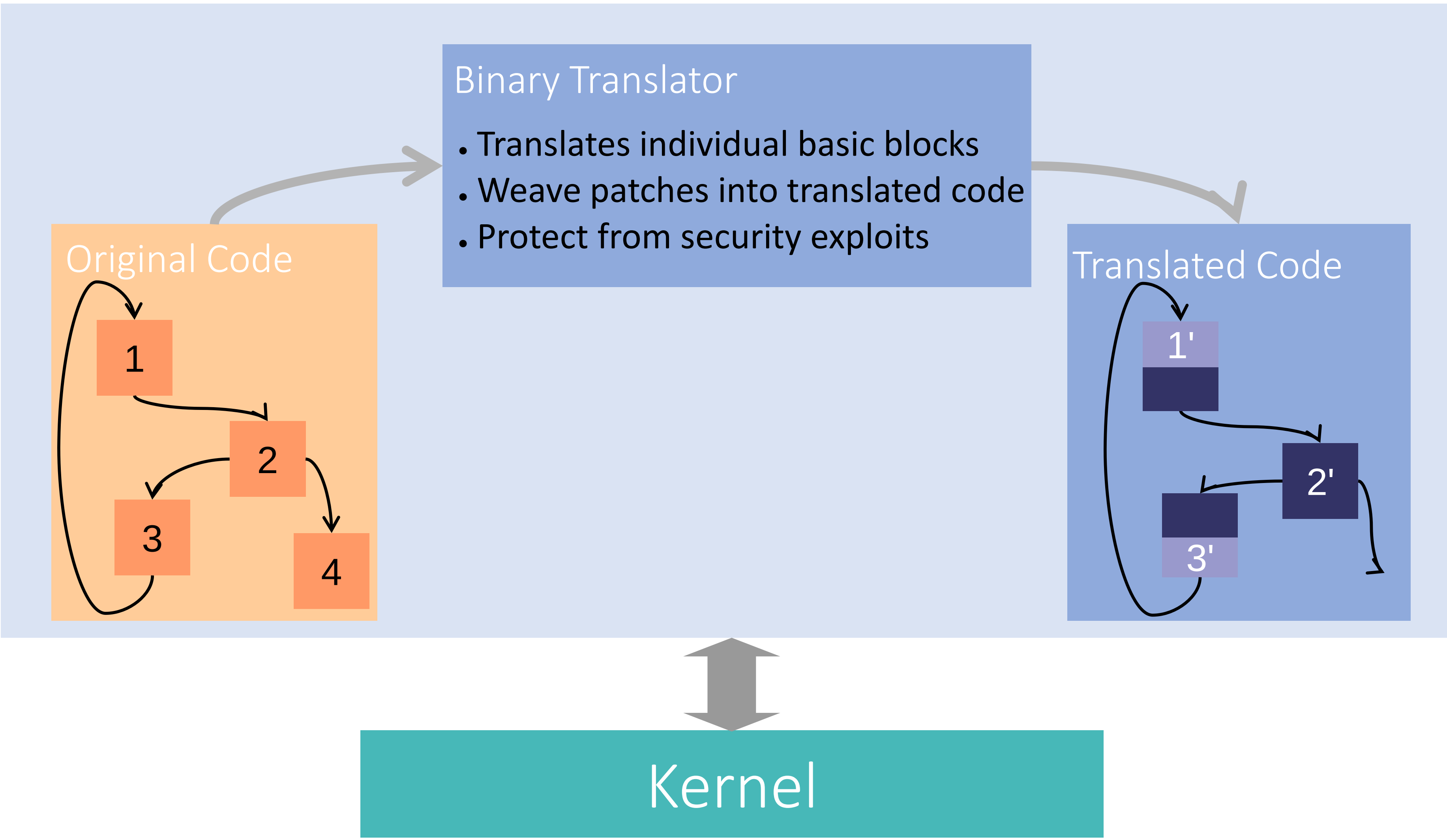
Patching architecture & distribution

Apache case-study

Evaluation

Conclusion

Code Translation



Patch Classes

Simple patch

- Only few instructions change, directly patched

Patches building on DSO:

- New import patch: additional library function used
- New function patch: additional function
- Additional call patch: calls to existing functions
- New String patch: new static string used

Other patches

- Type change, code refactoring, heavyweight changes

Patch Distribution

Most Linux distributions provide dynamic update service; piggy pack on this distribution service

- Automatically generate a dynamic patch when new package is generated
- Systems download packages and install dynamic patches to running services
- System administrators update binaries during next maintenance window

Implementation

DynSec builds on TRuE/libdetox **[IEEE S&P'12, ACM VEE'11]**

- Patching thread injected in BT layer
- Implemented in <2000 LoC
- 48 LoC changed in TRuE to add DynSec hooks
- Supports unmodified, unaware, multi-threaded x86 applications on Linux

Outline

Motivation

Patching architecture & distribution

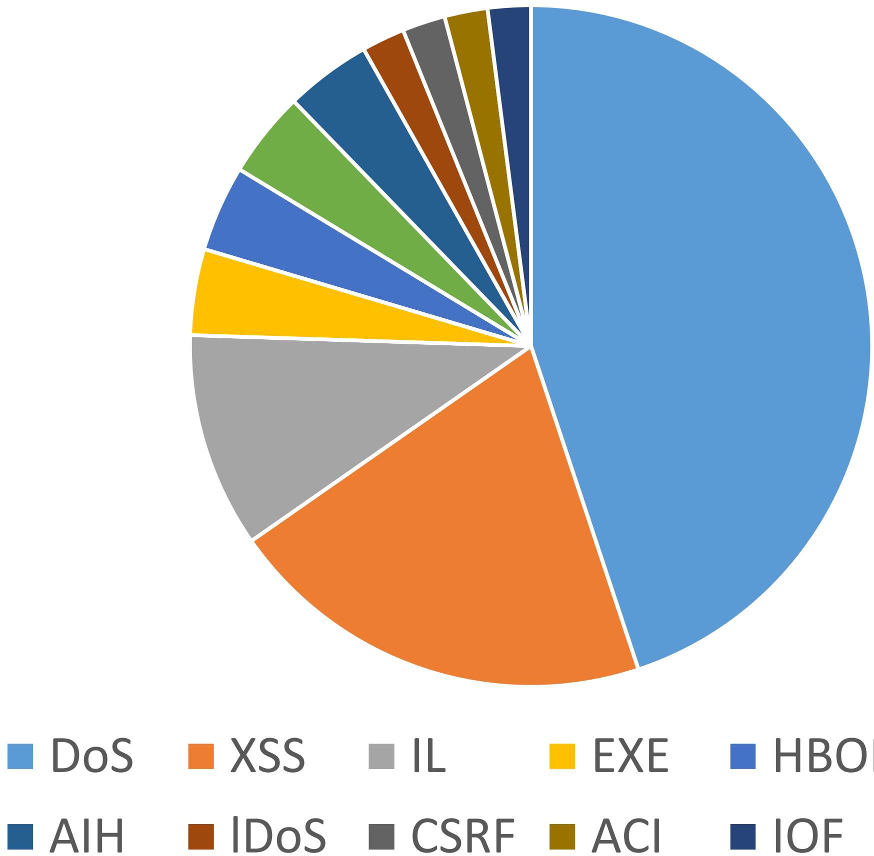
Apache case-study

- Vulnerability classes
- Distribution

Evaluation

Conclusion

Apache: Vulnerability Classes



Low	23
Moderate	19
Important	7

DynSec Coverage

Most (45/49) vulnerabilities are hot patchable

- All 7 important vulnerabilities
- 18 (out of 19) moderate vulnerabilities
- 20 (out of 23) low vulnerabilities

Patch complexity

- Important patches: 4 simple, 3 DSO patches
- Moderate patches: 6 simple, 12 DSO patches
- Low patches: 10 simple, 10 DSO patches

DynSec: Uncovered exploits

CVE-2007-3304 (lDoS, mod): signals to arbitrary PIDs

- Heavy code refactoring

CVE-2008-0005 (XSS, low): missing UTF-7 encoding

- Additional types, new functions

CVE-2012-0031 (DoS, low): scoreboard parent DoS

- Type change, new functions

CVE-2012-0883 (DoS, low): insecure variable in script

- Not applicable to DynSec (start-up script only)

DynSec: Uncovered exploits

CVE-2007-3304 (DoS, mod): signals to arbitrary PIDs

CVE

Possibility for 4 year stride without restart

CVE-2012-0031 (DoS, low): scoreboard parent DoS

- Type change, new functions

CVE-2012-0883 (DoS, low): insecure variable in script

- Not applicable to DynSec (start-up script only)

Sandbox Coverage

Protects from all code-based exploits

- Code injection
- Control-Flow redirection (ROP/partial JOP)
- System call policies

Unpatched protection for 11 (of 49) bugs

- Two important vulnerabilities (out of 7)
- 5 moderate vulnerabilities (out of 20)
- 4 low vulnerabilities (out of 21)

Outline

Motivation

Patching architecture & distribution

Apache case-study

Evaluation

- SPEC CPU 2006 performance
- Apache performance

Conclusion

Evaluation

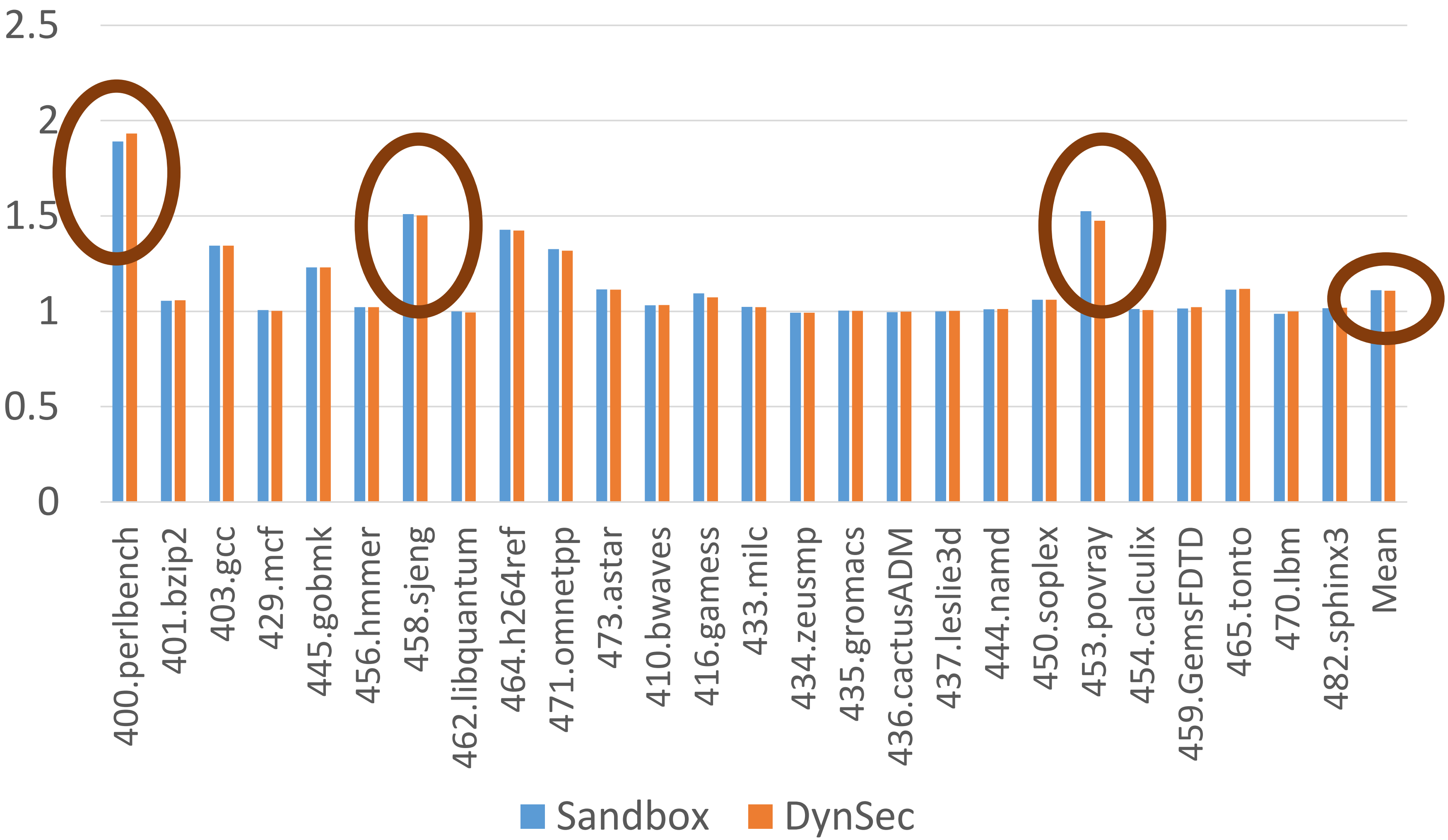
DynSec evaluated using SPEC CPU2006

- CPU: Intel Core2 Quad Q6600 @ 2.64GHz, 8GB RAM
- Ubuntu 11.04, Linux 2.6.38
- Used GCC 4.5.1 with –O2

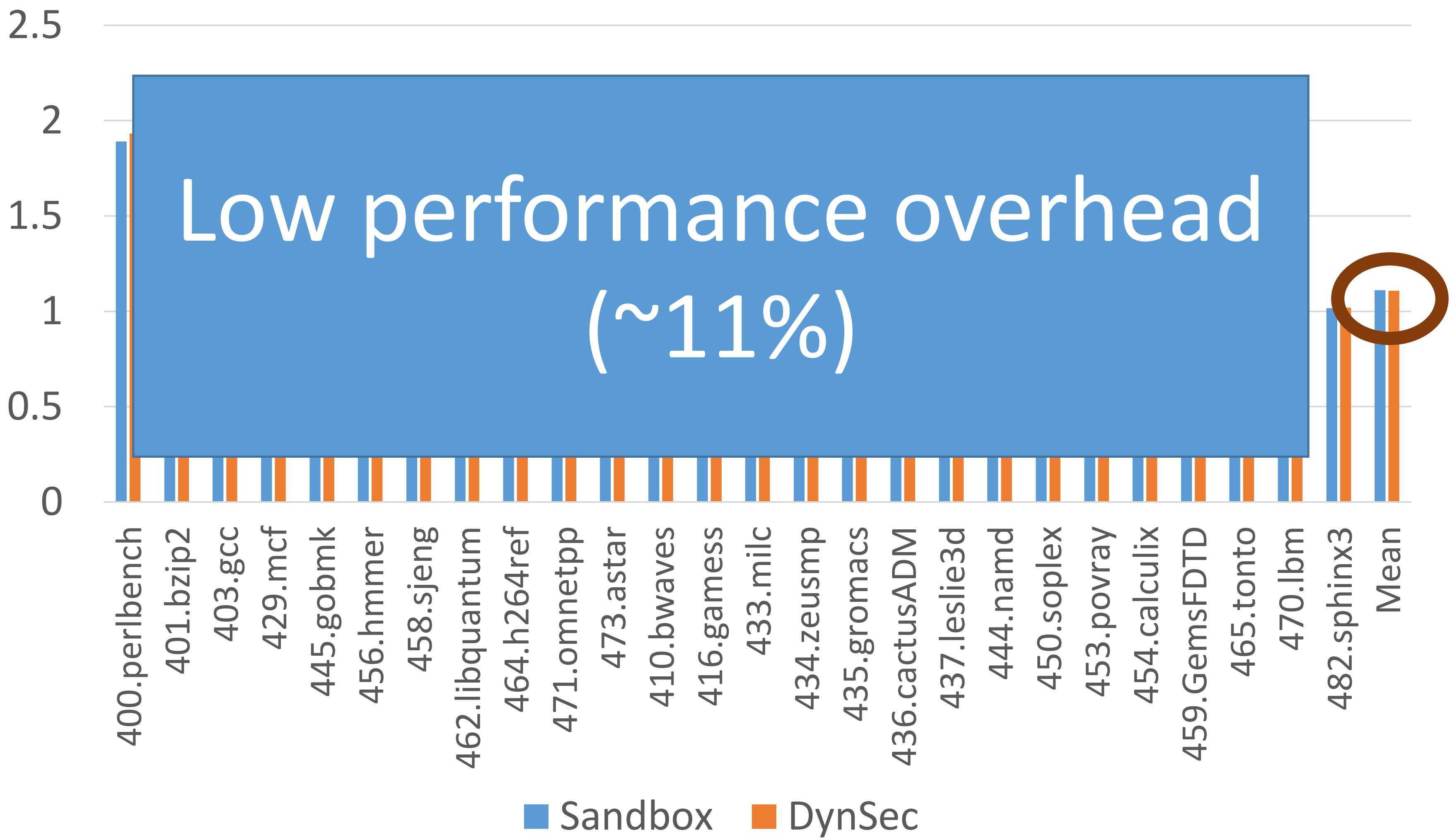
Benchmark configurations

- Native
- Sandboxing (use TRuE w/ shadow stack and checks)
- DynSec (with different patches)

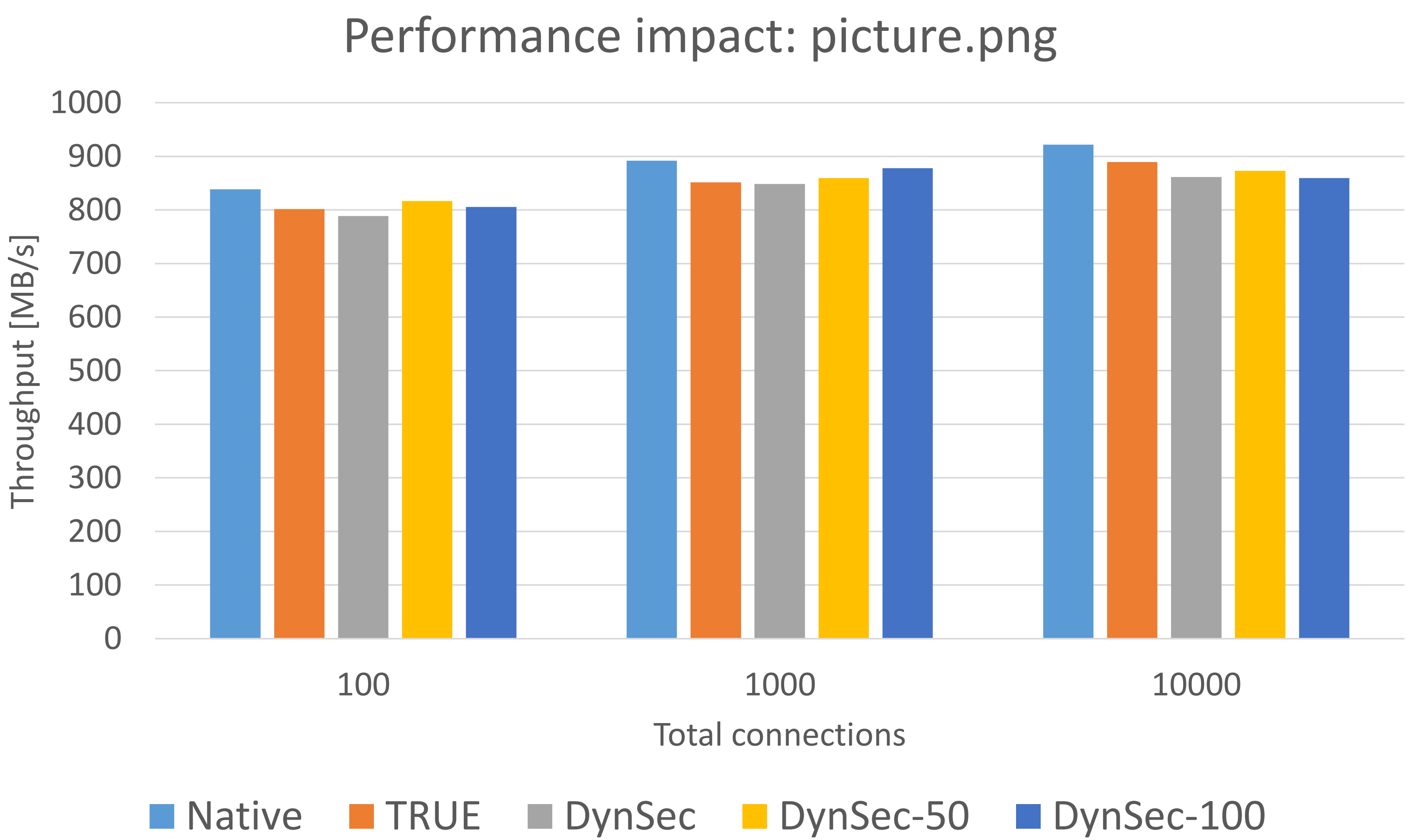
SPEC CPU2006: Performance



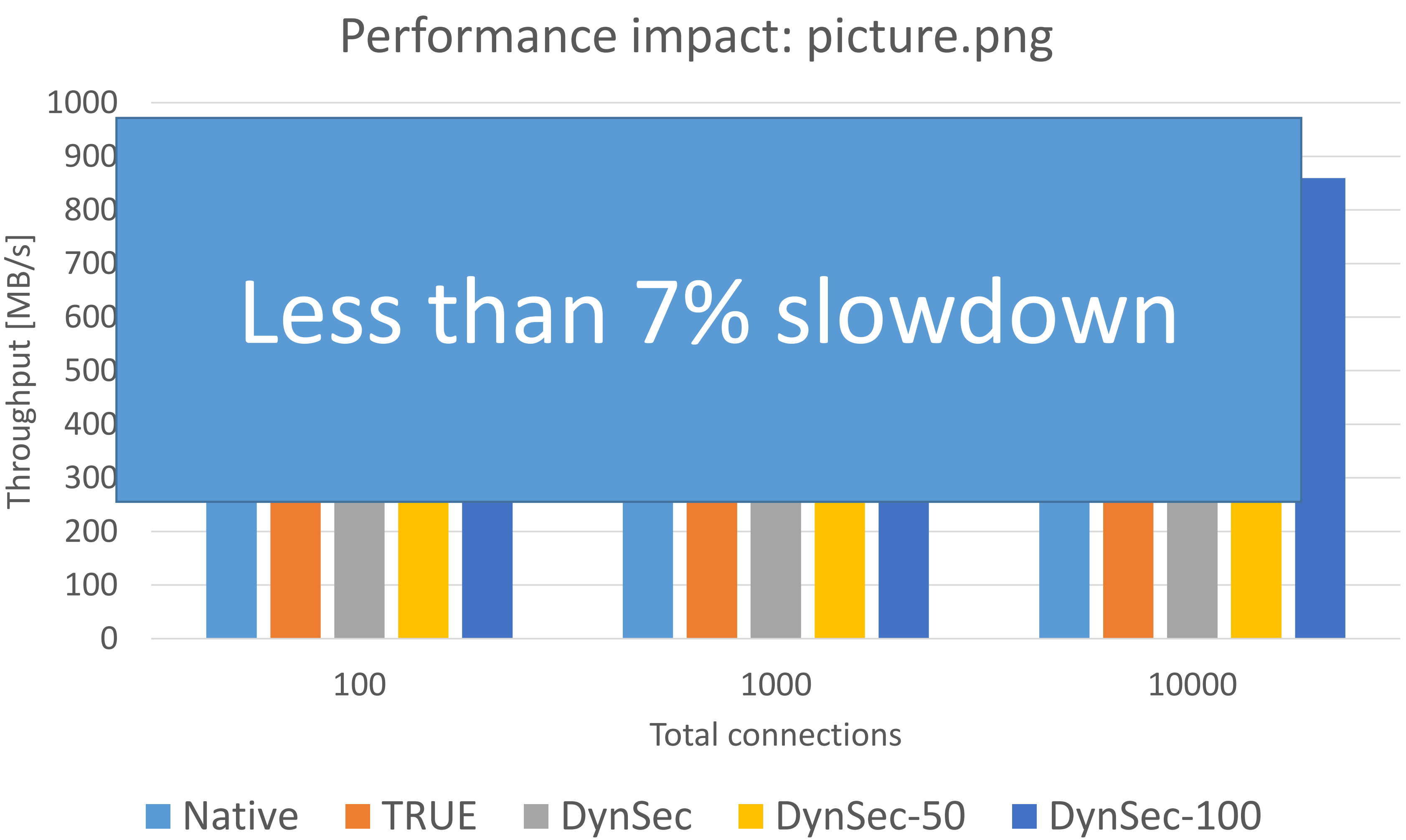
SPEC CPU2006: Performance



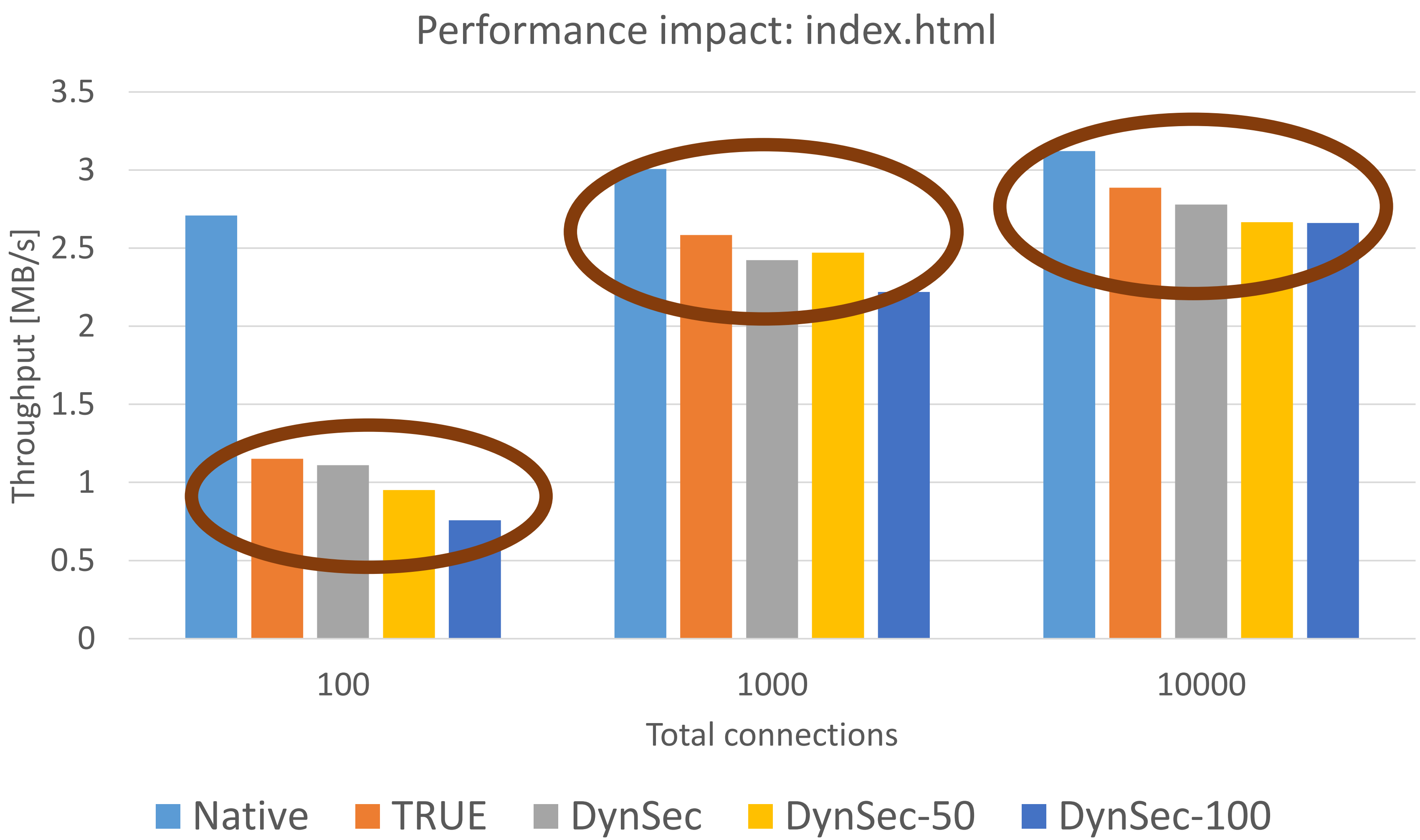
Apache: “large” files (~250kb)



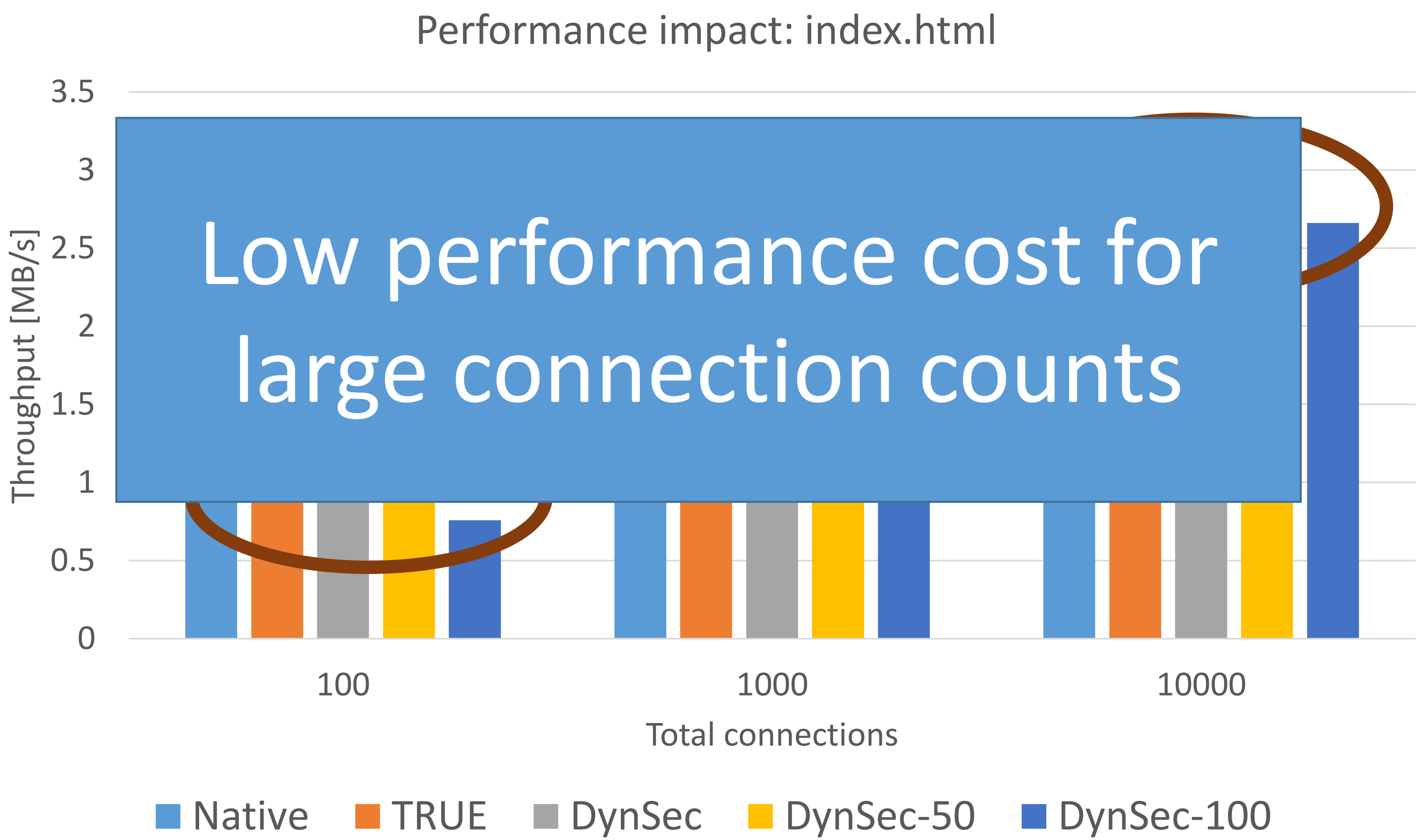
Apache: “large” files (~250kb)



Apache: small (tiny) files (~50b)



Apache: small (tiny) files (~50b)



Outline

Motivation

Patching architecture & distribution

Apache case-study

Evaluation

Conclusion

Conclusion

Virtualization enables on-the-fly code rewriting and repair for unmodified applications

- Sandbox protects integrity
- Patches provide availability

Study shows that protecting large, long-running, and modular applications like Apache is feasible

- High coverage: 45 of 49 Apache bugs patchable
- Low performance impact: 7% for Apache 2.2

Patching Architecture

DynSec thread waits for incoming patches

Patch application happens in 3 steps:

- Signal all application threads to stop
- Flush all code caches
- Restart application threads

Patch is applied indirectly when code is retranslated

- BT checks for every instruction if a patch is available

Patch Format

The focus of DynSec is on security patches

- Most security patches are only few lines of code
- Type changes and code refactoring out of scope

Patches are sets of changed instructions

Each patch may specify additional shared library for more heavyweight changes

Patch Extraction

Build patched application with current toolchain

Extract instruction differences between patched and unpatched version of the binary (per function)

- Changed instructions are added to patch
- Check differences in static read-only data
- Manually ensure integrity of patch (no type changes, no data changes)