DynSec: On-the-fly Code Rewriting and Repair

Mathias Payer*, Boris Bluntschli, 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

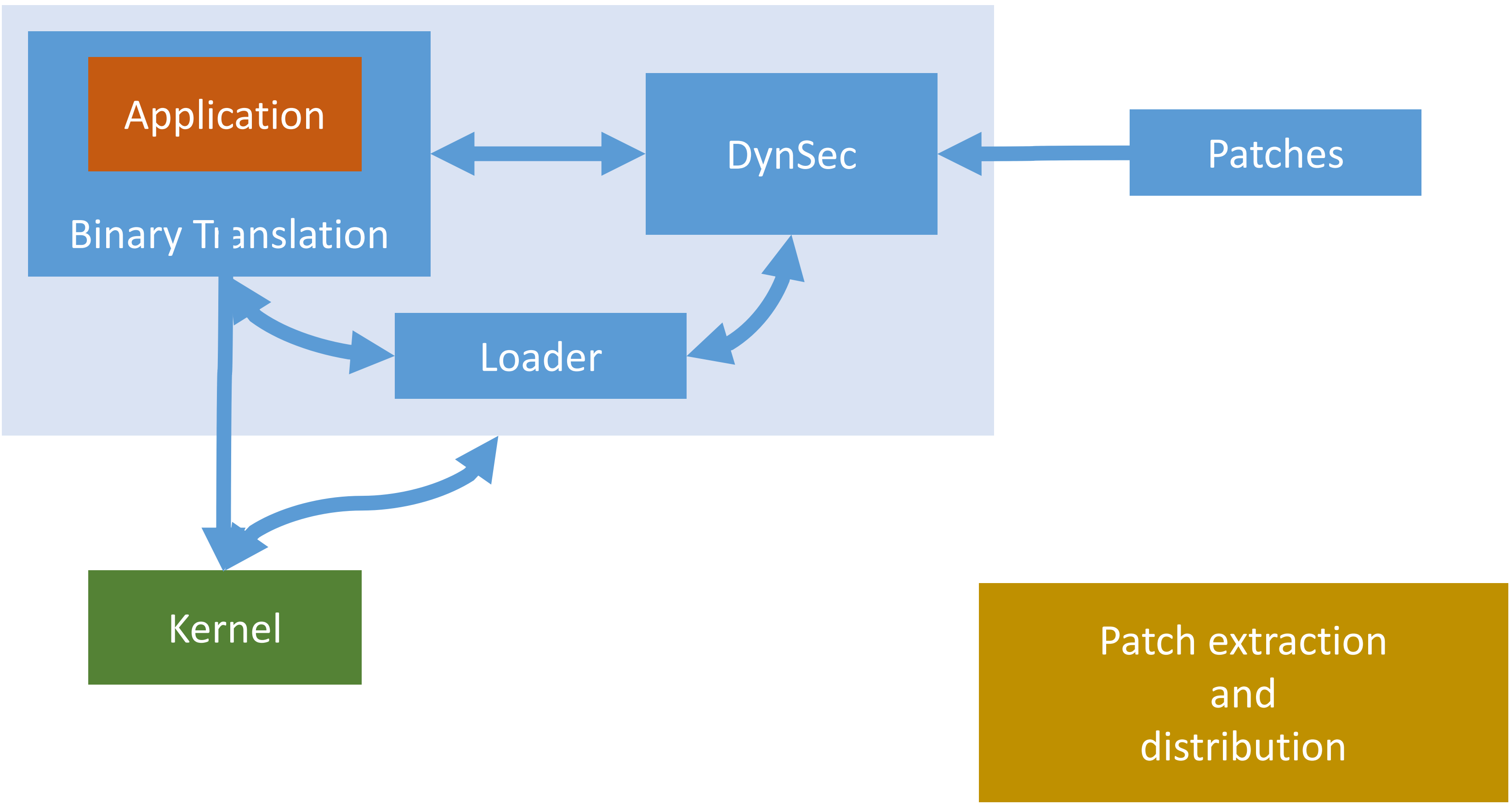
DynSec in 2 Minutes

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

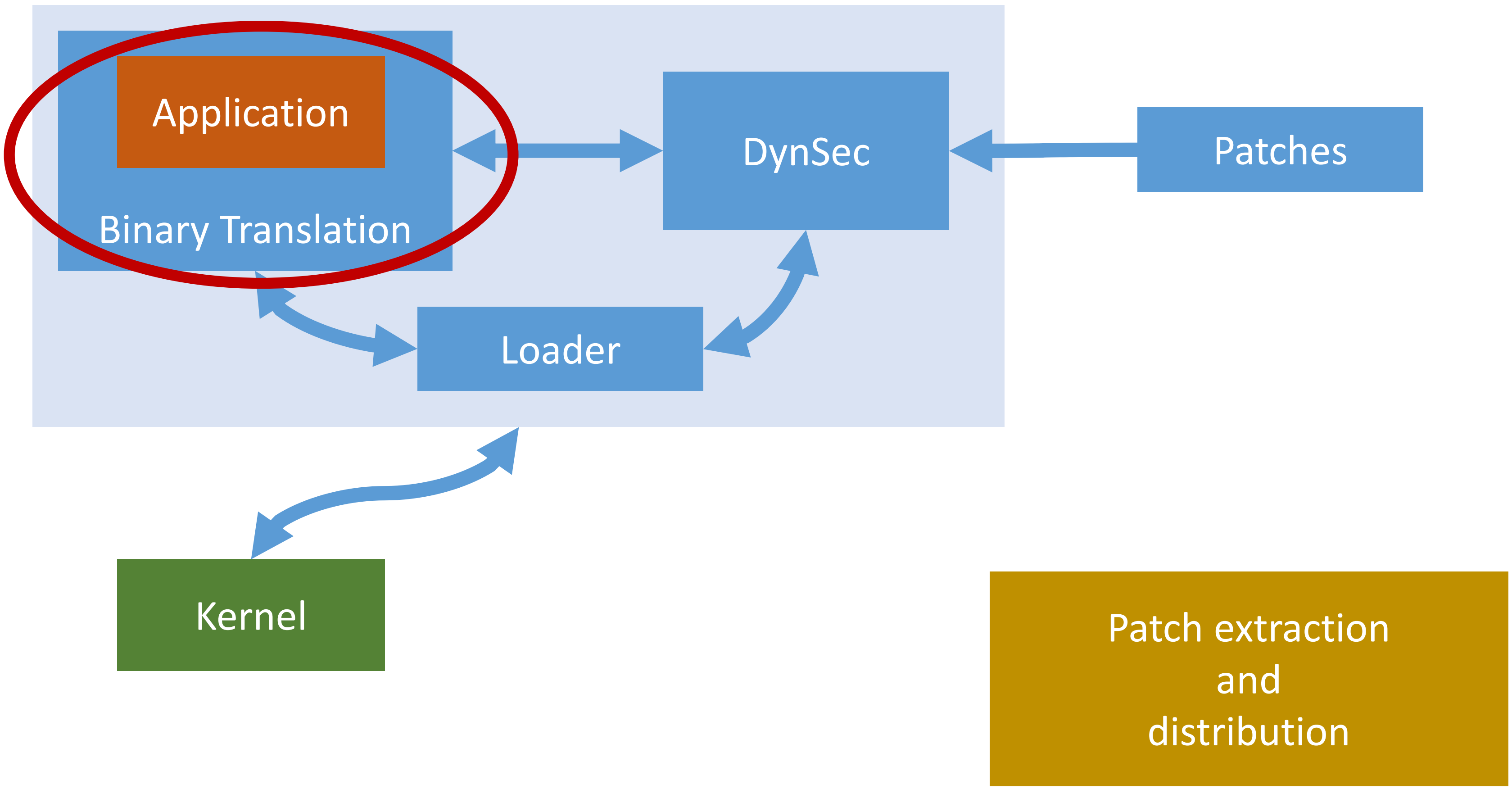
Binary Translation (BT) provides virtualization

- Sandbox protects integrity
- Dynamic update mechanism protects availability

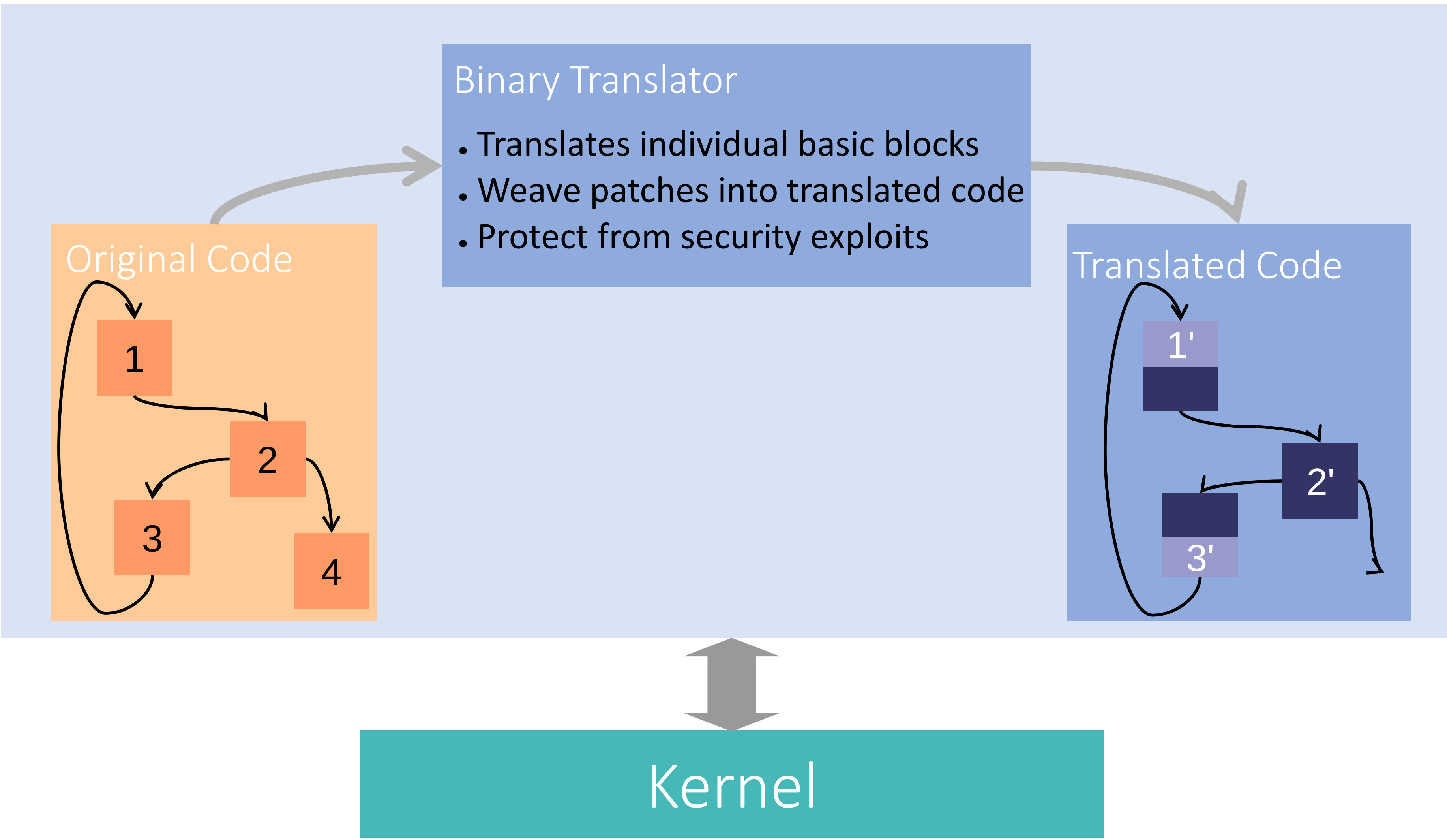
DynSec in 2 Minutes



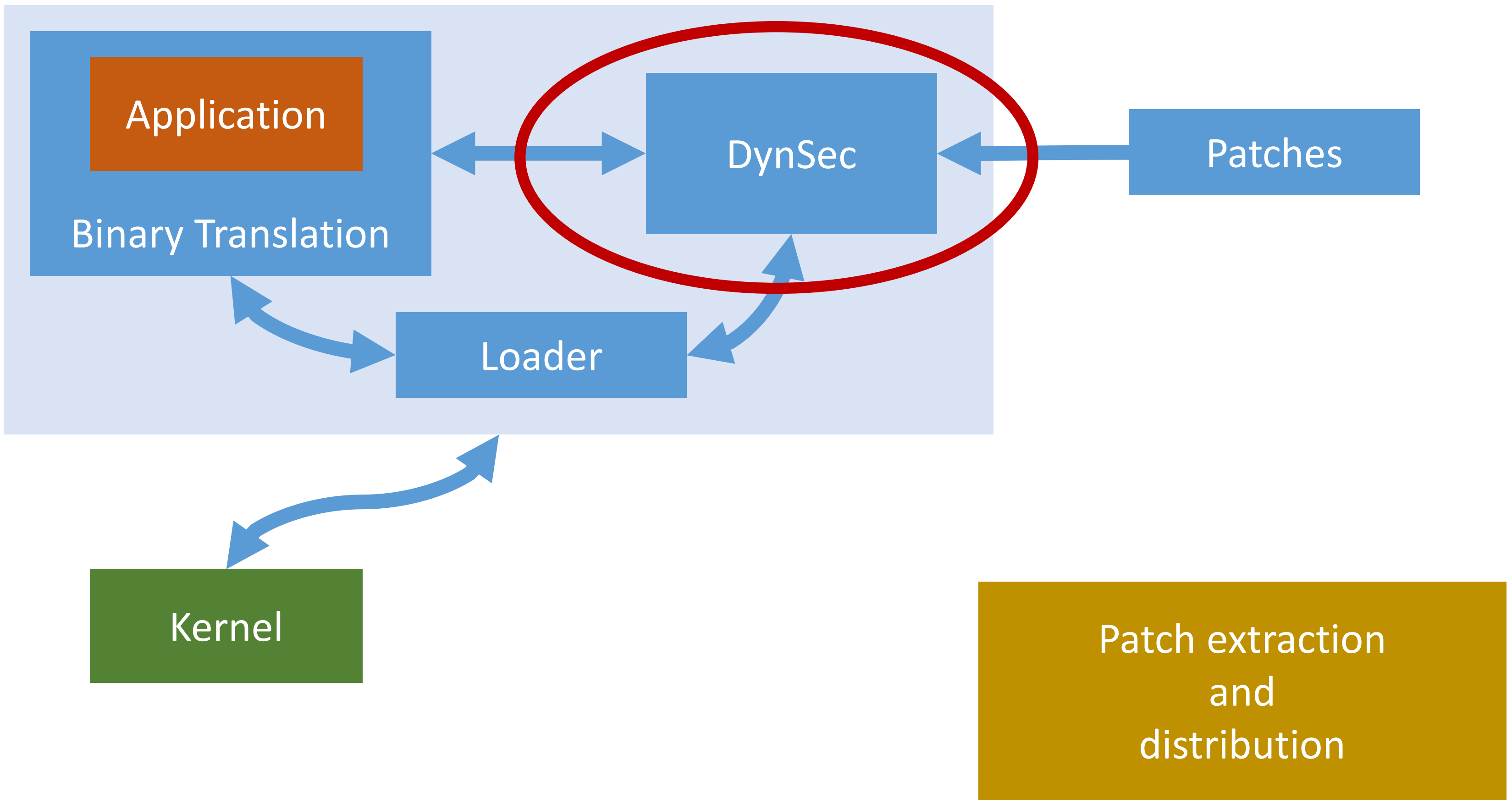
Outline


Application
Binary Translation
DynSec
Patches
Loader
Kernel
Patch extraction
and
distribution

Code Translation



Outline


Application
Binary Translation
DynSec
Patches
Loader
Kernel
Patch extraction
and
distribution

Patching Architecture

DynSec thread waits for incoming patches

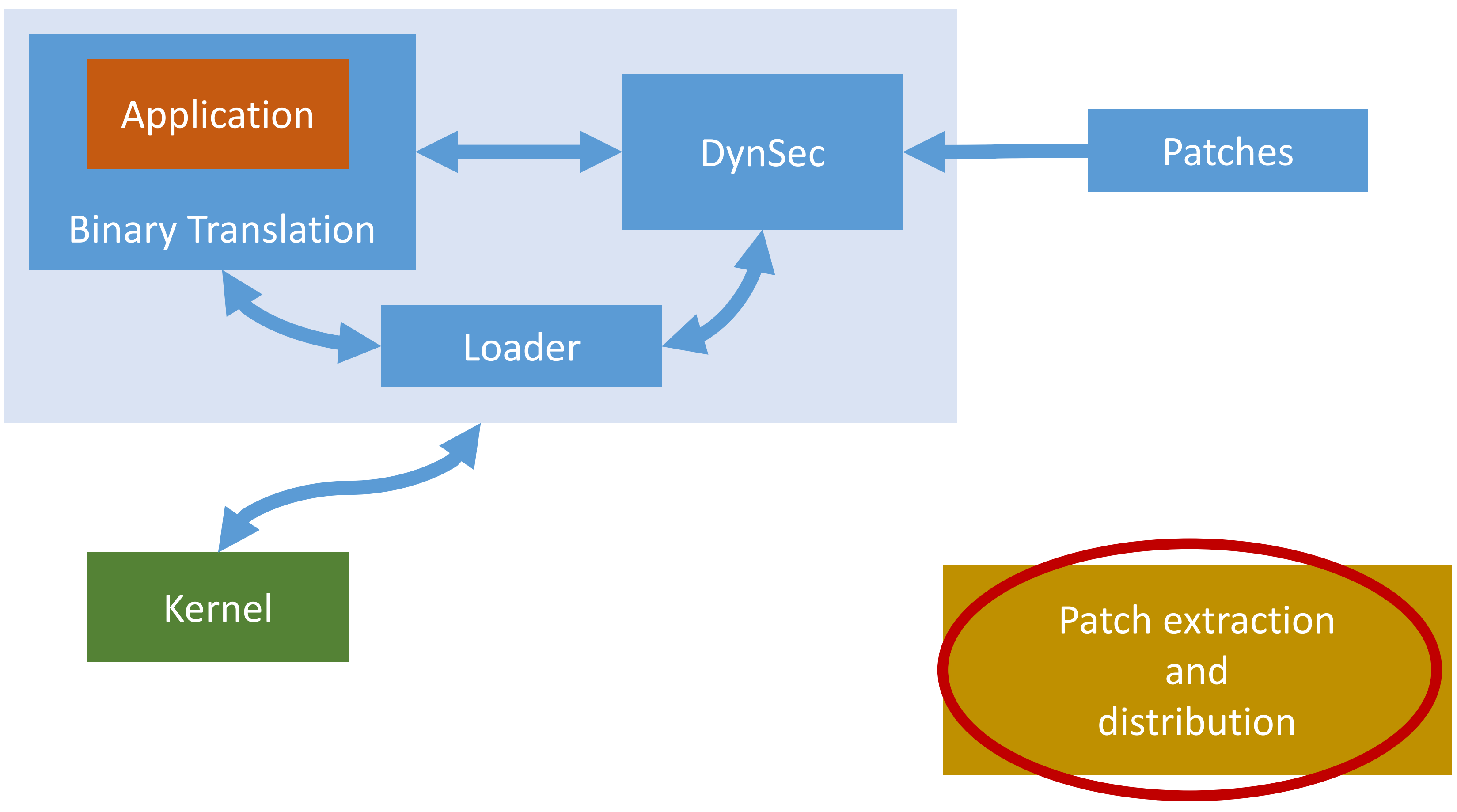
Patch application happens in 3 steps:

- 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

Outline



Application
Binary Translation
DynSec
Patches
Loader
Kernel
Patch extraction
and
distribution

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)

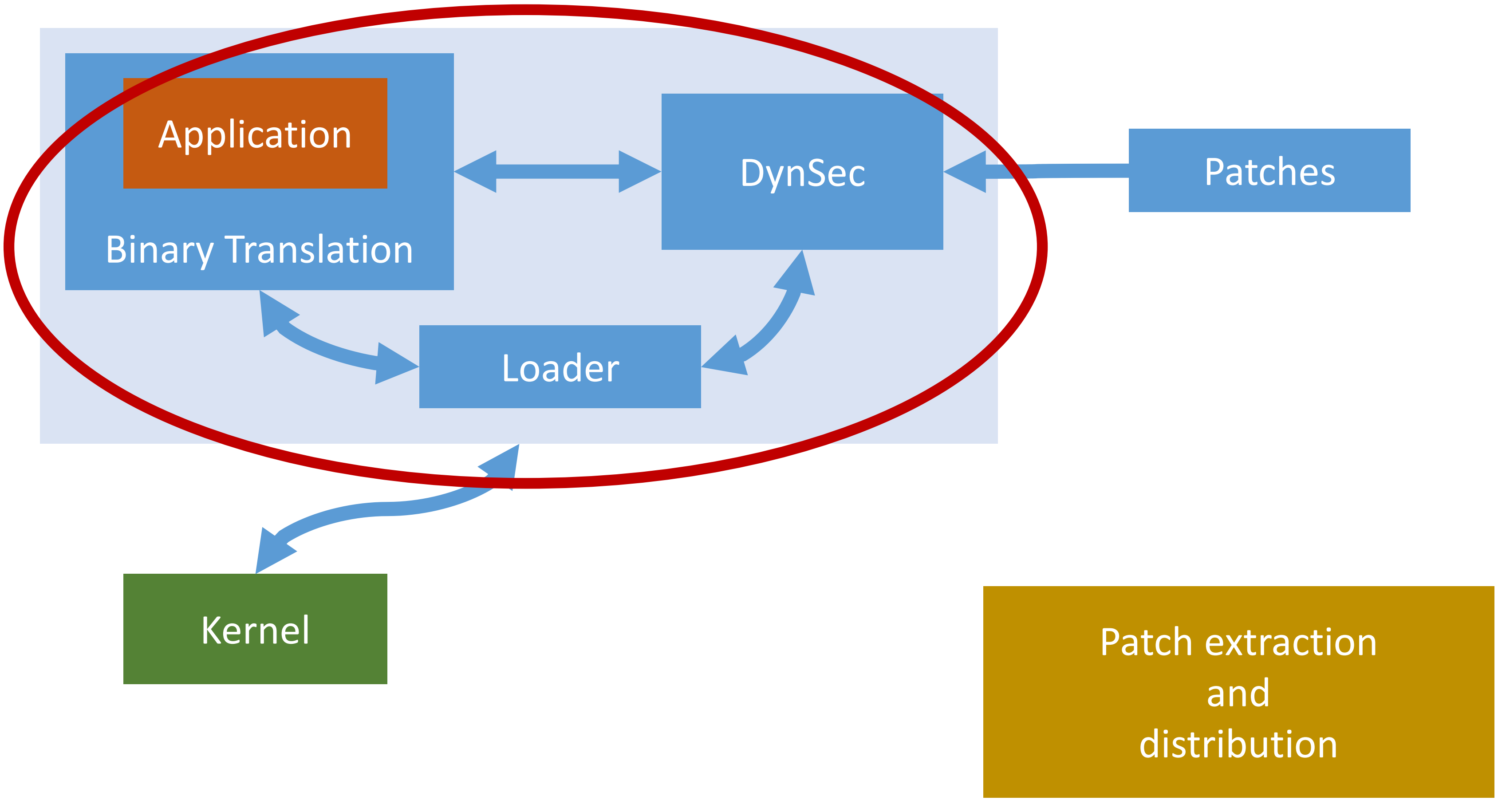
- Changed instructions are added to patch
- Check differences in static read-only data
- Manually ensure integrity of patch (no type changes, no data changes)

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

Outline



Application
Binary Translation
DynSec
Patches
Loader
Kernel
Patch extraction
and
distribution

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

Evaluation

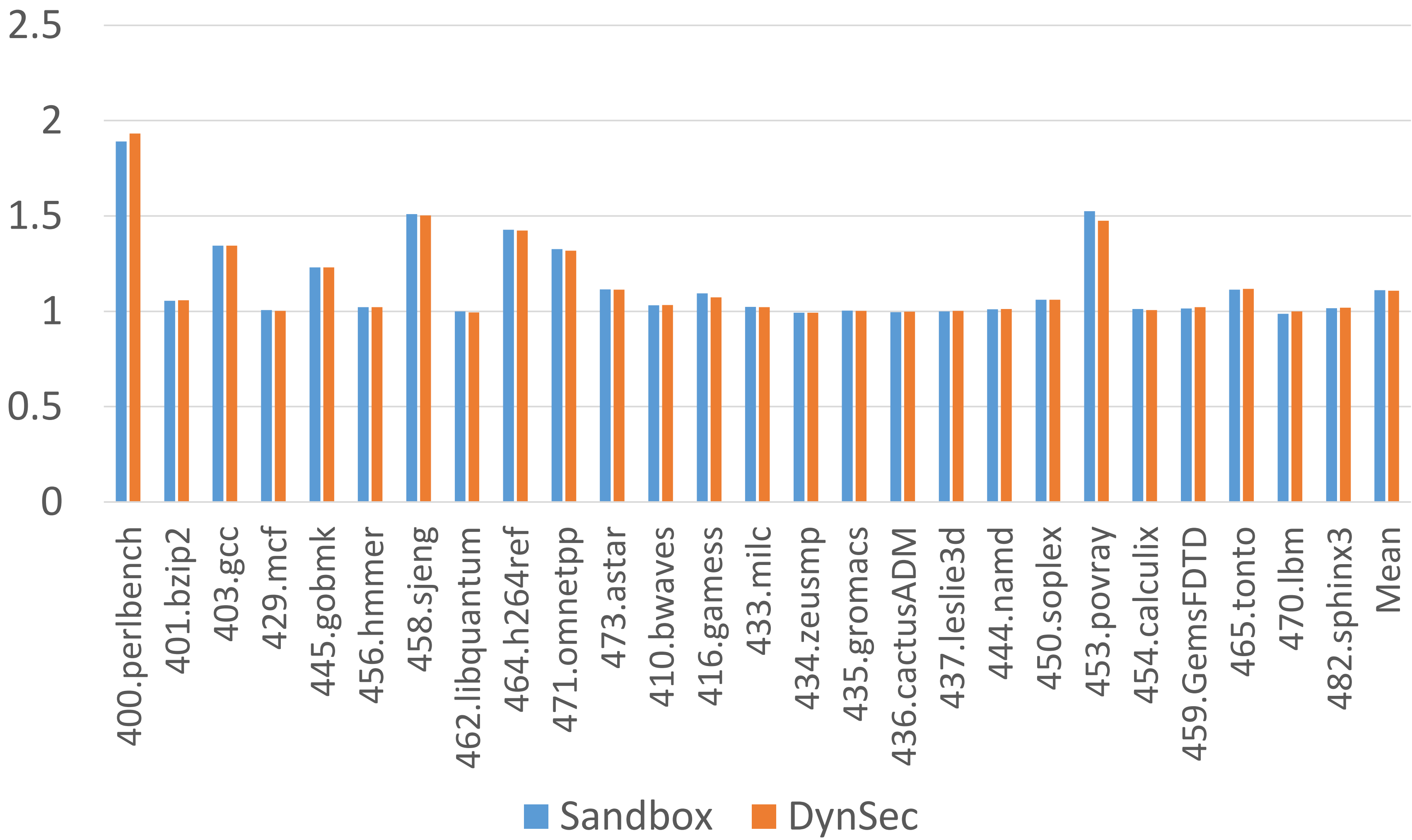
DynSec evaluated using SPEC CPU2006

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

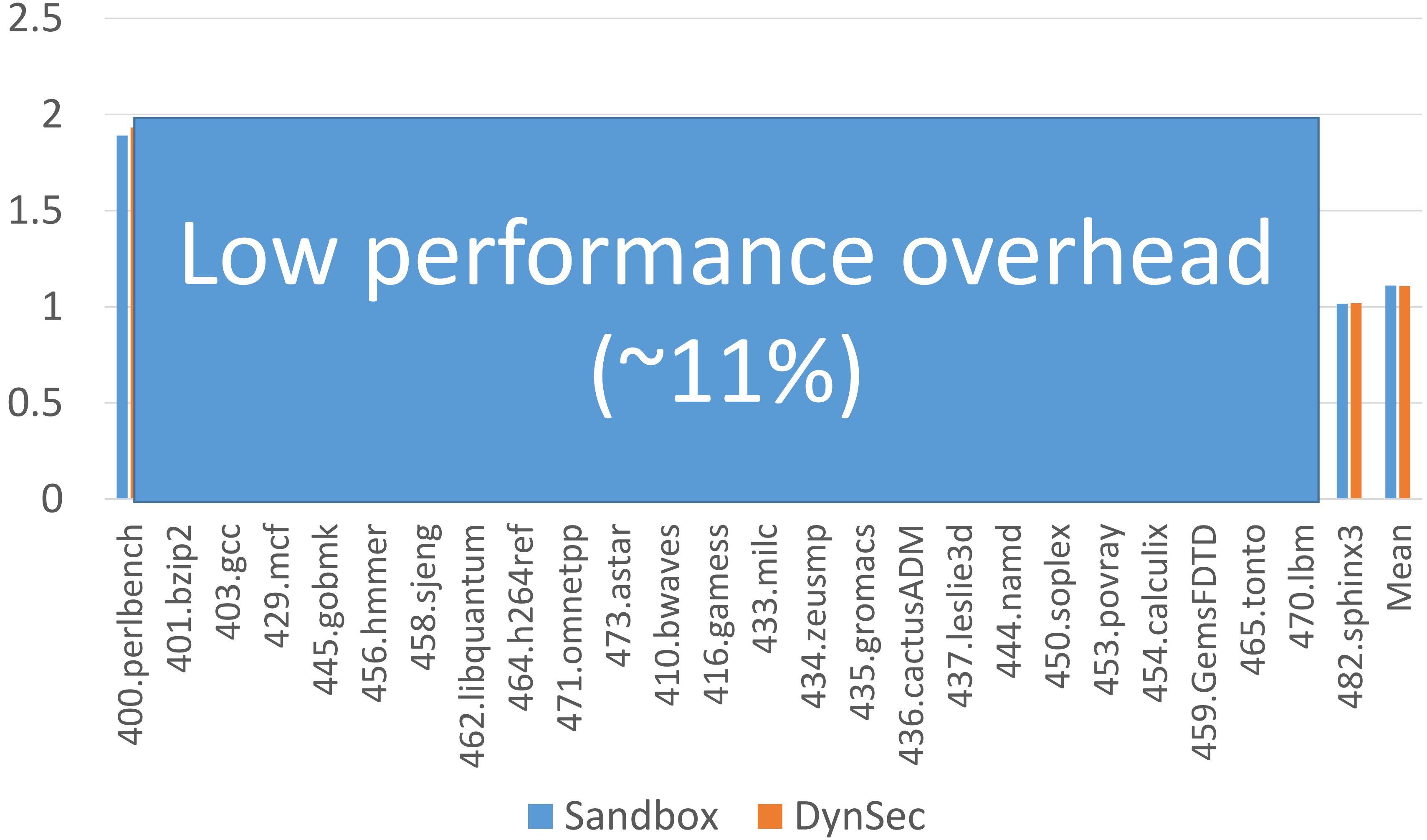
Three configurations

- Native
- Sandboxing (use TRuE w/ shadow stack and checks)
- DynSec (with one large patch)

SPEC CPU2006: Performance



SPEC CPU2006: Performance



CoreHTTP Security Study

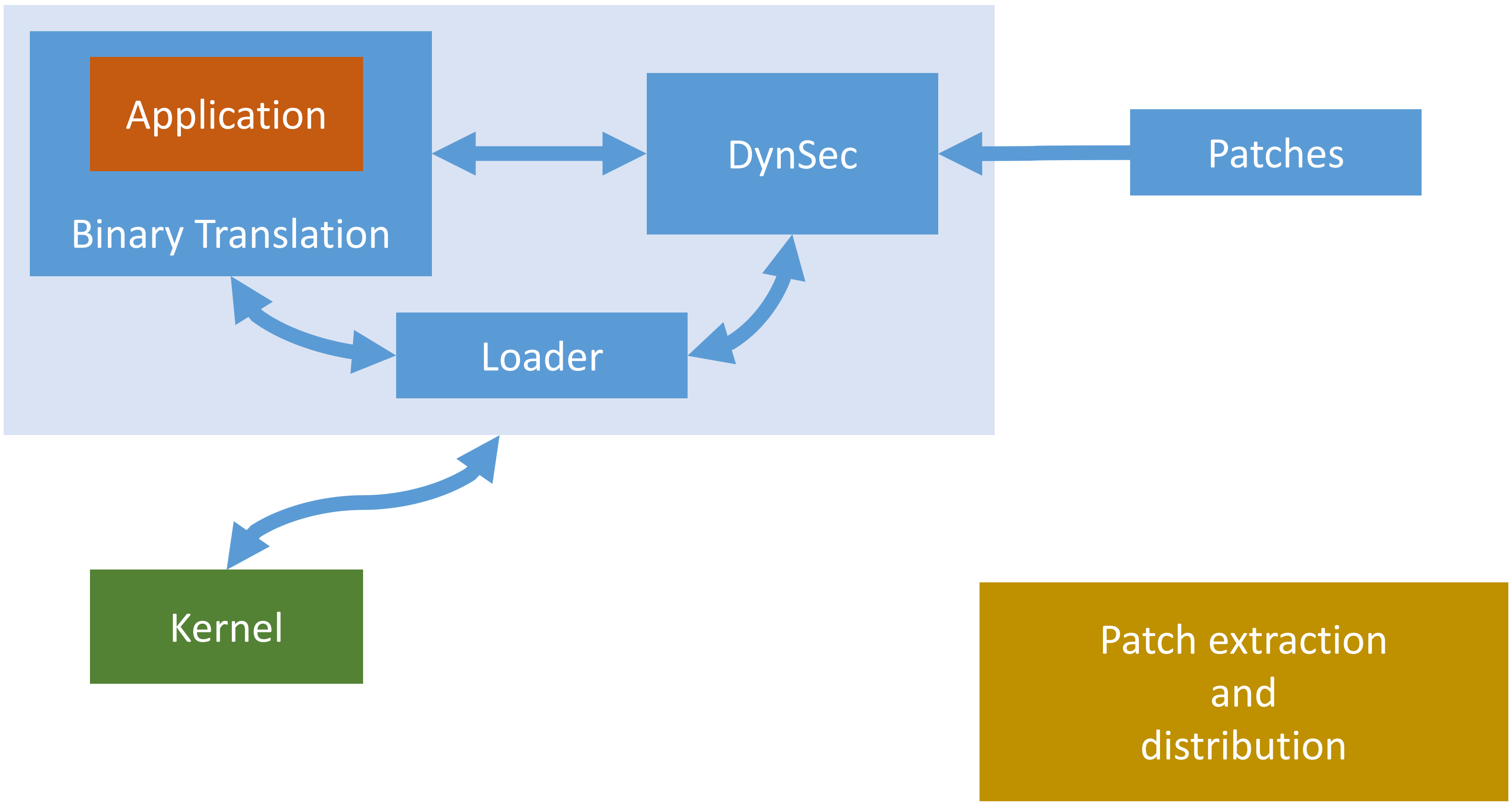
CoreHTTP is a simple web server with CGI support

We evaluate three security vulnerabilities

- CVE-2007-4060: missing input sanitation in `sscanf` (results in buffer overflow)
- CVE-2009-3586: off-by-one error in input sanitation (results in 1 byte buffer overflow)
- ExploitDB-10610: arbitrary command execution (`popen` is called with unescaped input string)

DynSec patches each vulnerability and protects CoreHTTP from exploitation

Outline


Application
Binary Translation
DynSec
Patches
Loader
Kernel
Patch extraction
and
distribution

Conclusion

DynSec offers on-the-fly code rewriting and repair for unmodified applications

Use virtualization (through Binary Translation) to combine power of two worlds:

- Sandbox protects integrity (control-flow protection)
- Dynamic update framework provides availability