Outfoxing the Mammoths

Marek Olszewski
Saman Amarasinghe

MIT CSAIL
Commit Group
Deterministic Multithreading
[Vasudevan and Edwards]

Always-on Race Detection
[Hans Boehm]
Tackling these ambitious challenges:

Applications

Operating Systems

Architectures
Where we want to be
Greatest invention since the transistor:
DynamoRIO
Pin
Valgrind

Application-Top Boxes

OS-Top Boxes
Aikido VM

- Adds per-thread page protection support to threads running in single address space
- Duplicates shadow pages:

  - Makes it possible to cheaply detect which data each thread accesses
Good for systems that instrument accesses to shared data:
- Race Detectors
- Deterministic Multithreading
- STMs, TLS, etc...

Currently conservative:
- Instrument all instructions that *might* access shared memory

Instead:
- Dynamically detect which instructions *definitely* access shared *pages*, and instrument only those
Questions?

http://groups.csail.mit.edu/commit/

Aikido VM
Backup Slides
Originally inspired by:
Overshadow by Chen et al. ASPLOS ’08

Overshadow: A Virtualization-Based Approach to Retrofitting Protection in Commodity Operating Systems

Xiaoxin Chen  Tiel Garfinkel  E. Christopher Lewis  Pratap Subramoniam  Carl A. Waldspurger
VMware, Inc.  *Stanford University  Princeton University  MIT

Abstract

Traditionally, operating systems executed with security sensitive data are vulnerable to large and complex, and consequently, expensive protective countermeasures. To address this limitation, we introduce a virtualization-based system called Overshadow that protects the privacy and integrity of application data, even in the event of a total OS compromise. Overshadow provides protection even in the event of a total OS compromise, but the OS is an encrypted view. This allows the operating system to carry out the entire set of managing an application’s resources, without having to trust or modify them. Thus, Overshadow offers a fast line of defense for application data.

Overshadow builds on our earlier conclusions, a novel specification that provides different views of physical memory, depending on the context determining the check. This provides an efficient way to ensure that the virtualization layer is secure. By providing an abstraction of the virtual layer, it is possible to retrofit this protection into existing systems without changing the underlying Operating System architecture. This approach eliminates the need for virtualization layers, and allows for the protection of application data even in the event of a total OS compromise. In addition, this approach also provides some level of protection even in the event of a partial OS compromise, and can be integrated into existing systems without modifying the underlying architecture.

Unfortunately, the security provided by commodity operating systems is still inadequate. Trapped off-compromise models that rely on hardware support do not provide adequate protection against advanced threats. Therefore, we propose a new approach that provides protection even in the event of a total OS compromise. This approach is based on the concept of a virtualization layer, which provides an abstraction of the underlying physical hardware. This abstraction allows for the protection of application data even in the event of a total OS compromise. In addition, this approach also provides some level of protection even in the event of a partial OS compromise, and can be integrated into existing systems without modifying the underlying architecture.

In conclusion, Overshadow provides a fast line of defense for application data in commodity operating systems, even in the event of a total OS compromise. This approach is based on the concept of a virtualization layer, which provides an abstraction of the underlying physical hardware. This abstraction allows for the protection of application data even in the event of a total OS compromise. In addition, this approach also provides some level of protection even in the event of a partial OS compromise, and can be integrated into existing systems without modifying the underlying architecture.

References

PetraVM Jinx

Uncover Hidden Concurrency Bugs Quickly

Complex concurrency bugs can stay hidden in your code for thousands of hours of operation, only crashing your application when certain conditions align. Jinx makes hidden bugs appear in development and test, before your customers find them.

JINX

Uses a VM to transparently checkpoint and explore different interleavings of shared memory communication to try to trigger bugs.