Virtual machine-provided
Context sensitive page mappings

Nathan Rosenblum, Gregory Cooksey, Barton P. Miller
Computer Sciences Department
University of Wisconsin

{nater,cooksey,bart}@cs.wisc.edu
http://pages.cs.wisc.edu/~nater/
Context

single store (von Neumann) memory architecture
Context

single store (von Neumann) memory architecture
Context

single store (von Neumann) memory architecture

instruction fetch

process

data reference

memory

introspective (self-examining) code

hot/runtime patched code

self-modifying programs

dynamic code unpacking/decryption
What is context sensitivity?
What is context sensitivity?

kind of

memory reference

execution

data read/write
Context sensitivity has been useful

**Circumvention**

**Stealth**
Sparks, Butler. *Shadow Walker: Raising the bar for rootkit detection.* Black Hat Japan. 2005

**Protection**
A brief look ahead

What are context sensitive page mappings?

Why “virtual machine [monitor]-provided”?

Nitty-gritty details

Case study: self-checksumming codes

What else are CSPM good for?
Context sensitive page mappings

1. Memory reference

2. Context-free page access

physical memory
Context sensitive page mappings

1. Memory reference

2. Context-free page access

3. Data or instruction

Something weird happens
Why implement with VMMs?
Why implement with VMMs?

unmodified OS

user processes

guest os

VMM

hardware
Why implement with VMMs?

- **Hardware**
- **VMM** controls hardware
- **Protection**
- **Control of memory management hardware**

Diagram:
- Unmodified OS
- User processes
- Guest OS
- VMM controls

nathan rosenblum
Splitting paged memory on IA32

linear address

<table>
<thead>
<tr>
<th>PD index</th>
<th>PT index</th>
<th>offset</th>
</tr>
</thead>
</table>

fetch?

dtlb

itlb

mmu

physical memory

linear address

fetch?

dtlb

itlb

mmu

physical memory
Splitting paged memory on IA32

The diagram illustrates the process of splitting memory using a PD index, PT index, and offset to fetch memory. The paging caches are synchronized.

**Diagram Details:**
- Linear address
- PD index
- PT index
- Offset
- Fetch?
- DTLB
- ITLB
- MMU
- Physical memory

**Caption:**
- Paging caches synchronized
Splitting paged memory on IA32

linear address

<table>
<thead>
<tr>
<th>PD index</th>
<th>PT index</th>
<th>offset</th>
</tr>
</thead>
</table>

mmu

fetch?

paging caches synchronized

dtlb

itlb

physical memory

violating assumptions

nathan rosenblum
Xen hypervisor implementation

MMU Management

Guest OS

mmu update

Xen
Xen hypervisor implementation

MMU Management

Guest OS

mmu update

Xen

protect PTEs, maintain map
Xen hypervisor implementation

Page Fault Handling

memory reference

page fault

mmu

Xen
Page Fault Handling

memory reference

page fault

mmu

induce TLB split

Xen hypervisor implementation
Desynchronizing TLBs

Flush TLB on page fault, then:

physical memory

CS page

user addr space

VMM reserved
Desynchronizing TLBs

Flush TLB on page fault, then:

- CS page
- create new mapping
- physical memory

user addr space — VMM reserved —
Flush TLB on page fault, then:

DTLB loaded
Desynchronizing TLBs

Flush TLB on page fault, then:

1. create new mapping
2. read from this page
3. remap

Flush TLB on page fault, then:

DTLB loaded
Desynchronizing TLBs

Flush TLB on page fault, then:

1. create mapping
2. read from this page
3. remap
4. install trampoline code doesn’t pollute DTLB

Flush TLB on page fault, then:

DTLB loaded
Desynchronizing TLBs

Flush TLB on page fault, then:
- DTLB loaded
- ITLB loaded

1. Create a new mapping
2. Read from this page
3. Remap
4. Install trampoline code
5. Execute this page

 Doesn’t pollute DTLB

User addr space

- VMM reserved

Physical memory
Desynchronizing TLBs

Flush TLB on page fault, then:

- DTLB loaded
- ITLB loaded

Restore mappings and return

1. create new mapping
2. read from this page
3. remap
4. install {trampoline code}
5. execute this page
doesn’t pollute DTLB

user addr space — VMM reserved —

CS page — execute this page — physical memory
Self checksumming code

Assumes that bytes read and bytes executed are the same
Self checksumming code

Assumes that bytes read and bytes executed are the same

Terminate if modifications detected (tamper resistance)
VMM + user level system

guest VM

igor

target

page mappings

Xen
VMM + user level system

guest VM

evil assistant

igor

target

page mappings

Xen
VMM + user level system

copy & modify pages

guest VM

Xen

target

igor

evil assistant
VMM + user level system

Copy & modify pages → alert hypervisor

page mappings

Xen
VMM + user level system

1. Copy & modify pages
2. Alert hypervisor
3. Target continues blissfully unaware
How much does it cost?

Per MMU update
- checking/enforcing protection bits
- maintaining mapping structures

Per Page Fault
- checking for mapping
- inducing TLB split

- normal pages + 8%
- context sensitive pages + 11%

- normal pages + 13%
- context sensitive pages + 132%
How much does it cost?

Per MMU update
- checking/enforcing protection bits
- maintaining mapping structures

Per Page Fault
- checking for mapping
- inducing TLB split

normal pages + 8%
context sensitive pages + 11%

[relatively complex]

normal pages + 13%
context sensitive pages + 132%
Other uses

OS tamper resistance
  protecting system call tables

Shielded processes
  protecting processes from malicious privileged access