Secure Address Space Protection in a Multi-hosts Environment

V1. Sep 22, 2012

Outline

- What kinds of hosts and devices in the PCIe network?
- What kinds of address spaces in the PCIe network?
- What components do we use to translate among addresses space?
- How does the each address space being translated?
Hosts and Devices

- **MH (Management Host)**
  - The host that extends its own PCI hierarchy into the PCIe switch fabric.

- **CH (Computer Host):** host behind a NT virtual side.
  - **VCH (Virtualized CH):** host with SW/HW supports for running virtual machines.
  - **NVCH (Non-virtualized CH):** host which runs single OS with IOMMU

- **Devices**
  - Normal PCI device
  - SR-IOV
Address Spaces in Hosts

- MH and NVCH
  - Physical address space
  - Virtual address space
  - Device virtual address space

- VCH
  - Host physical address space
  - Host virtual address space
  - Guest physical address space
  - Guest virtual address space
  - Device virtual address space
Address Space in Devices

- IOMMU provides each device its own device virtual address space.
- IOMMU translates a device’s virtual address into physical address by looking up the device’s corresponding page table.
- Host relies on IOMMU to isolate physical address range a device can access so that malicious device won’t access privileged memory.
## Translation Components

<table>
<thead>
<tr>
<th>Comp.</th>
<th>Identifier</th>
<th>input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>PT</td>
<td>Process ID</td>
<td>Host virtual addr.</td>
<td>Host phys addr</td>
</tr>
<tr>
<td>GPT</td>
<td>Process ID</td>
<td>Guest virtual addr.</td>
<td>Guest phys addr</td>
</tr>
<tr>
<td>EPT</td>
<td>Domain(VM) ID</td>
<td>Guest phys addr.</td>
<td>Host phys addr</td>
</tr>
<tr>
<td>IOMMU</td>
<td>Device ID</td>
<td>Dev virtual addr.</td>
<td>Host phys addr</td>
</tr>
<tr>
<td>LUT</td>
<td>Part of Device ID</td>
<td>Device ID</td>
<td>Device ID</td>
</tr>
<tr>
<td>NTB</td>
<td>BAR addresses</td>
<td>Physical Address at one side</td>
<td>Dev virtual addr. at the other side</td>
</tr>
</tbody>
</table>

PT: Page Table  
GPT: Guest Page Table  
EPT: Extended Page Table  
LUT: Look Up Table  
NTB: Non-Transparent Bridge
NTB Address Mapping

- NTB maps from <the primary side to the secondary side>

- Mapping: <addrA at prim. side to addrB at the secondary>
  - addrA = Dev1 BARn, addrB = {RAM | Dev2 BARn}
  - Dev1: {NT-Link | NT-Virtual}
  - Dev2: {any device on the other side}
  - Actually addrB could target any secondary side physical address

- One-way Translation:
  - Read/write at addrA == read/write addrB
  - Read/write at addrB does not translated to addrA

- Type
  - addrA = {physical addr.}
  - addrB = {device virtual addr.}
Address Translation to MH

1. CH’s CPU
   - CPU
   - GPT
   - PT
   - NTB
   - IOMMU

2. VCH VM’s CPU
   - CPU
   - GPT
   - EPT
   - NTB
   - IOMMU

3. CH’s device (P2P)
   - DEV
   - IOMMU
   - NTB
   - IOMMU

MH’s CPU
- CPU
- PT
- IOMMU

MH’s device
- DEV
- NTB
- IOMMU

MH’s Physical Address Space

hpa -> host physical addr.
hva -> host virtual addr.
gva -> guest virtual addr.
gpa -> guest physical addr.
dva -> device virtual addr.
Address Translation to CH

- **hpa** -> host physical addr.
- **hva** -> host virtual addr.
- **gva** -> guest virtual addr.
- **gpa** -> guest physical addr.
- **dva** -> device virtual addr.

**CH’s CPU**
- CPU
  - hva
  - PT

**CH’s device**
- DEV
  - gva
  - IOMMU
  - dva

**4. CH VM’s CPU**
- CPU
  - gva
  - GPT
  - gpa
  - EPT

**5. MH’s CPU**
- CPU
  - hva
  - PT
  - hpa
  - NTB
    - dva
    - IOMMU

**6. MH’s device (P2P)**
- DEV
  - hpa
  - NTB
    - dva
    - IOMMU

**CH’s Physical Address Space**
## Summary 1

<table>
<thead>
<tr>
<th>src \ dst</th>
<th>MH memory / device</th>
<th>CH memory / device</th>
</tr>
</thead>
<tbody>
<tr>
<td>MH CPU</td>
<td>MMU</td>
<td>MMU+NTB+IOMMU</td>
</tr>
<tr>
<td>MH Dev</td>
<td>IOMMU</td>
<td>IOMMU+NTB+IOMMU</td>
</tr>
<tr>
<td>CH CPU</td>
<td>MMU+NTB+IOMMU</td>
<td>MMU</td>
</tr>
<tr>
<td>CH Dev</td>
<td>IOMMU+NTB+IOMMU</td>
<td>IOMMU</td>
</tr>
<tr>
<td>CH VM</td>
<td>MMU+GPT+NTB+IOMMU</td>
<td>MMU+GPT</td>
</tr>
</tbody>
</table>

Device-to-device protection requires switch ACS and P2P support.
Inter-host Address Translation

7. From CH1’s CPU to CH2’s RAM
8. From CH1’s Device to CH2’s RAM
9. From VM in CH1 to CH2’s RAM

MH’s Physical Address Space

H2’s Physical Address Space
## Summary 2

<table>
<thead>
<tr>
<th>src \ dst</th>
<th>CH Memory</th>
<th>Other CH memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>CH CPU</td>
<td>MMU</td>
<td>MMU + NTB + MH’s IOMMU + NTB + dst’s IOMMU</td>
</tr>
<tr>
<td>CH Dev</td>
<td>IOMMU</td>
<td>IOMMU + NTB + MH’s IOMMU + NTB + dst’s IOMMU</td>
</tr>
<tr>
<td>CH VM</td>
<td>MMU + GPT</td>
<td>MMU + GPT + NTB + MH’s IOMMU + NTB + dst’s IOMMU</td>
</tr>
</tbody>
</table>
## Case Description

<table>
<thead>
<tr>
<th>Case</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CH accesses MH’s memory,</td>
</tr>
<tr>
<td>2</td>
<td>VM on CH accesses MH’s memory,</td>
</tr>
<tr>
<td>3</td>
<td>CH’s device accesses MH’s memory</td>
</tr>
<tr>
<td>4</td>
<td>CH’s VM access CH’s memory</td>
</tr>
<tr>
<td>5</td>
<td>MH’s CPU accesses CH’s memory</td>
</tr>
<tr>
<td>6</td>
<td>MH’s device accesses CH’s memory</td>
</tr>
<tr>
<td>7</td>
<td>CH1’s CPU accesses CH2’s memory</td>
</tr>
<tr>
<td>8</td>
<td>CH1’s device accesses CH2’s memory</td>
</tr>
</tbody>
</table>
PCI device ID translation

MH's PCI ID Domain

1. Each host (MH + CHs) has its own device id domain.
2. LUT is responsible for ID translation from one host domain to another.
3. IOMMU depends on ID to setup its access table.
LUT + IOMMU based protection

- Prevent unauthorized dev from accessing any memory area in other CHs such as CH2

- Solution strategies:
  - LUT lookup fail or IOMMU lookup fail

- NTB translates dev’s ID space from MH to CH2 by LUT.

- Make sure
  - Either LUT has no entry for dev’s <bus:dev>
  - Or dev’s translated ID is forbidden in the CH2’s IOMMU table.
Examples
LUT translation example

- Each VF / NTB is uniquely assigned a DevID <BusNo : DevNo . FunNo>

- LUT entries consists of index : <BusNo, DevNo>
  - LUT translation:
    - <BusNo : DevNo. FunNo> translate to
    - <NTB’s BusNo : matching index in LUT. FunNo>

- Ex: VF1=<3:12.1>, VF5=<3:12.5>, NTB for VF5=<2:0.0>, LUT= 4 : <3:12>
  - VF1 after translation: <2:4.1>, forbidden in IOMMU
  - VF5 after translation: <2:4.5>, allow in IOMMU
Example 1  From CH2's CPU to MH's Device VF1's CSR

- MMU -> NTB -> IOMMU -> MH VF1's CSR
  - VF1's CSR: 0xF4000000, ID=[3:11.1]
  - MMU mapping 0xFF800000 -> 0xF6800000
  - NTB [2:0.0/4:0.0] mapping 0xF6800000 -> 0xFA800000
  - IOMMU: 0xFA800000 -> 0xF4000000

- ID translation
  - Setup [0:0] at LUT index 0
  - CH2's CPU ID = [0:0.0], ID translation: [0:0.0] -> [4:0.0]
  - setup IOMMU for [4:0.0]
Example 2: From VF1 to DMA CH2's memory 0x6800000

- VF1 -> IOMMU -> NTB -> IOMMU
  - VF1[3:11.2] writes 0xFB800000
  - IOMMU: 0xFB800000 -> 0xF8800000
  - NTB[2:0.0] maps 0xF8800000 -> 0xFA800000
  - IOMMU mapping 0xFA800000 -> 0x6800000

- ID translation
  - Setup first IOMMU for [3:11.2]
  - Setup entry [3:11] at LUT index 3
  - ID translation: [3:11.2] -> [2:3.2]
  - VF1's ID at CH2 = [2:3.2], setup second IOMMU for [2:3.2]

CH's Physical Address Space
Example 3: From CH1’s CPU to CH2’s memory 0x8000000

- Let CH1 write to CH2 NTB’s address mapping range
  - Same setup as example 1, but write to NTB instead of VF1
  - Setup NTB to map 0xF480000000 -> 0xF28000000
  - Setup IOMMU to map NTB’s ID for 0xF2800000000 -> 0xA8000000

- ID translation
  - Setup first IOMMU for [4:0.0] (as in ex1)
  - Setup entry [4:0] at right side LUT index 3
  - ID translation: [4:0.0] -> [2:3.0]
  - Setup second IOMMU for [2:3.0]
End
Translation Components

- **PT (Page Table):**
  - Host virtual address -> host physical address

- **GPT (Guest Page Table):**
  - Guest physical address -> Host physical address

- **EPT (Extended Page Table):**

- **IOMMU:**
  - Device virtual address -> Host physical address

- **NTB:**
  - Physical Address at one side of a host -> device virtual Address another side of a host

- **LUT:**
  - PCI device ID at one side of NTB -> PCI device ID at another side