Lecture 4: Protected Modules Architectures

Secure Compilation Seminar

Marco Patrignani

CISPA
HELMHOLTZ-ZENTRUM i. G.
Protected Modules Architecture (PMA)

- low-level isolation mechanism
Protected Modules Architecture (PMA)

- low-level isolation mechanism
- several research prototypes: Fides [SP], Sancus [Noo+13], Flicker [McC+08], TrustVisor [McC+10], Smart [smart]
Protected Modules Architecture (PMA)

- low-level isolation mechanism
- several research prototypes: Fides [SP], Sancus [Noo+13], Flicker [McC+08], TrustVisor [McC+10], Smart [smart]
- industrial prototype too: Intel SGX [McK+13]
Protected Modules Architecture (PMA)

- low-level isolation mechanism
- several research prototypes: Fides [SP], Sancus [Noo+13], Flicker [McC+08], TrustVisor [McC+10], Smart [smart]
- industrial prototype too: Intel SGX [McK+13]
- implemented via Hypervisor, Hardware, Software
Protected Modules Architecture (PMA)

• low-level isolation mechanism
• several research prototypes: Fides [SP], Sancus [Noo+13], Flicker [McC+08], TrustVisor [McC+10], Smart [smart]
• industrial prototype too: Intel SGX [McK+13]
• implemented via Hypervisor, Hardware, Software
What does PMA Provide?

Isolation:

• Code encapsulation
• Data encapsulation
• program counter-based access control
• precise API exposure via Entry Points
PMA in Action (1 Module)

- memory space

```
0x0001    call 0xb53
0x0002    movs r0 0x0b55

0x0b52    movs r0 0x0b55
0x0b53    call 0x0002
0x0b54    movs r0 0x0001
0x0b55    ...

0xab00    jmp 0xb53
0xab01    ...
```
PMA in Action (1 Module)

- memory space
- protected module = protected memory

```
0x0001    call 0xb53
0x0002    movs r0 0x0b55

0x0b52    movs r0 0x0b55
0x0b53    call 0x0002
0x0b54    movs r0 0x0001
0x0b55    ...
```

```
0xab00    jmp 0xb53
0xab01    ...
```
PMA in Action (1 Module)

- memory space
- protected module = protected memory
- split in code and data

```
0x0001  call 0xb53
0x0002  movs r0 0xb55

0xb52  movs r0 0xb55
0xb53  call 0x0002
0xb54  movs r0 0x0001
0xb55  ...

0xab00  jmp 0xb53
0xab01  ...
```
PMA in Action (1 Module)

- memory space
- protected module = protected memory
- split in code and data
- protected code is unrestricted

```
0x0001 call 0xb53
0x0002 movs r0 0x0b55

0x0b52 movs r0 0x0b55
0x0b53 call 0x0002
0x0b54 movs r0 0x0001
0x0b55 ...

0xab00 jmp 0xb53
0xab01 ...
```
PMA in Action (1 Module)

- memory space
- protected module = protected memory
- split in code and data
- protected code is unrestricted

```
0x0001    call 0xb53
0x0002    movs r0 0xb55
          ...
0xb52    movs r0 0xb55
0xb53    call 0x0002
0xb54    movs r0 0x0001
0xb55    ...
          ...
0xab00    jmp 0xb53
0xab01    ...
```
PMA in Action (1 Module)

- memory space
- protected module = protected memory
- split in code and data
- protected code is unrestricted
PMA in Action (1 Module)

- memory space
- protected module = protected memory
- split in code and data
- protected code is unrestricted
- unprotected code is restricted

\[
\begin{array}{c:c}
0x0001 & call 0xb53 \\
0x0002 & movs r0 0x0b55 \\
\vdots & \\
0xb52 & movs r0 0x0b55 \\
0xb53 & call 0x0002 \\
0xb54 & movs r0 0x0001 \\
0xb55 & \ldots \\
\vdots & \\
0xab00 & jmp 0xb53 \\
0xab01 & \ldots \\
\end{array}
\]
PMA in Action (1 Module)

- memory space
- protected module = protected memory
- split in code and data
- protected code is unrestricted
- unprotected code is restricted
PMA in Action (1 Module)

- memory space
- protected module = protected memory
- split in code and data
- protected code is unrestricted
- unprotected code is restricted

```
0x0001  call 0xb53
0x0002  movs r0 0x0b55
⋮
0x0b52  movs r0 0x0b55
0x0b53  call 0x0002
0x0b54  movs r0 0x0001
0x0b55  ...
⋮
0xab00  jmp 0xb53
0xab01  ...
```
PMA in Action (1 Module)

- memory space
- protected module = protected memory
- split in code and data
- protected code is unrestricted
- unprotected code is restricted
- entry points for communication (•)
PMA in Action (1 Module)

- memory space
- protected module = protected memory
- split in code and data
- protected code is unrestricted
- unprotected code is restricted
- entry points for communication (x)
## PMA Access Control Summary

Access Control Policy enforced based on the PC location

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Protected</th>
<th></th>
<th></th>
<th>Unprotected</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Entry Point</td>
<td>Code</td>
<td>Data</td>
<td></td>
</tr>
<tr>
<td>Protected</td>
<td>Protected</td>
<td>r x</td>
<td>r x</td>
<td>r w</td>
<td>r w x</td>
</tr>
<tr>
<td>Unprotected</td>
<td>Unprotected</td>
<td>x</td>
<td></td>
<td></td>
<td>r w x</td>
</tr>
</tbody>
</table>
PMA & Assembly Execution

: 10 jmp r₄
11 movi r₁ 14
12 jmp r₁
13 cmp r₁ r₂
14 jmp r₃

r₁ = 0 ; r₂ = 101 ; r₃ = 100 ; r₄ = 104 ; r₅ = 11

: 100 jmp r₂
101 movi r₁ 10
102 jmp r₁
103 sub r₁ r₂
104 jmp r₅

:
PMA & Assembly Execution

Protected Memory

```
10 jmp r₄
11 movi r₁ 14
12 jmp r₁
13 cmp r₁ r₂
14 jmp r₃
```

Unprotected Memory

```
100 jmp r₂
101 movi r₁ 10
102 jmp r₁
103 sub r₁ r₂
104 jmp r₅
```

\[ r₁ = 0 \; ; \; r₂ = 101 \; ; \; r₃ = 100 \; ; \]
\[ r₄ = 104 \; ; \; r₅ = 11 \]
### PMA & Assembly Execution

**Protected Memory**

- 10 jmp r₄
- 11 movi r₁ 14
- 12 jmp r₁
- 13 cmp r₁ r₂
- 14 jmp r₃

**Unprotected Memory**

- 100 jmp r₂
- 101 movi r₁ 10
- 102 jmp r₁
- 103 sub r₁ r₂
- 104 jmp r₅

**Register Values**

- r₁ = 0
- r₂ = 101
- r₃ = 100
- r₄ = 104
- r₅ = 11
### Protected Memory

- `10 jmp r_4`
- `11 movi r_1 14`
- `12 jmp r_1`
- `13 cmp r_1 r_2`
- `14 jmp r_3`

### Unprotected Memory

- `100 jmp r_2`
- `101 movi r_1 10`
- `102 jmp r_1`
- `103 sub r_1 r_2`
- `104 jmp r_5`

### Variables

- \( r_1 = 14 \)
- \( r_2 = 101 \)
- \( r_3 = 100 \)
- \( r_4 = 104 \)
- \( r_5 = 11 \)
PMA & Assembly Execution

Protected Memory

```
: 10 jmp r4
 11 movi r1 14
 12 jmp r1 r1=14
 13 cmp r1 r2
 14 jmp r3
: 100 jmp r2
```

Unprotected Memory

```
: 100 jmp r2
 101 movi r1 10
 102 jmp r1
 103 sub r1 r2
 104 jmp r5
: 
```

\[ r_1 = 14 \ ; \ r_2 = 101 \ ; \ r_3 = 100 \ ; \ r_4 = 104 \ ; \ r_5 = 11 \]
PMA & Assembly Execution

Protected Memory

```
:  
10  jmp r₄
11  movi r₁ 14
12  jmp r₁  r₁=14
13  cmp r₁ r₂
14  jmp r₃
:  
```

Unprotected Memory

```
:  
100 jmp r₂
101 movi r₁ 10
102 jmp r₁
103 sub r₁ r₂
104 jmp r₅
:  
```

```
r₁ = 14 ; r₂ = 101 ; r₃ = 100 ;
r₄ = 104 ; r₅ = 11
```
## PMA & Assembly Execution

### Protected Memory

<table>
<thead>
<tr>
<th>PC</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>jmp $r_4</td>
</tr>
<tr>
<td>11</td>
<td>movi $r_1 14</td>
</tr>
<tr>
<td>12</td>
<td>jmp $r_1</td>
</tr>
<tr>
<td>13</td>
<td>cmp $r_1 $r_2</td>
</tr>
<tr>
<td>14</td>
<td>jmp $r_3</td>
</tr>
</tbody>
</table>

### Unprotected Memory

<table>
<thead>
<tr>
<th>PC</th>
<th>Instruction</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>jmp $r_2</td>
</tr>
<tr>
<td>101</td>
<td>movi $r_1 10</td>
</tr>
<tr>
<td>102</td>
<td>jmp $r_1</td>
</tr>
<tr>
<td>103</td>
<td>sub $r_1 $r_2</td>
</tr>
<tr>
<td>104</td>
<td>jmp $r_5</td>
</tr>
</tbody>
</table>

- $r_1 = 14$
- $r_2 = 101$
- $r_3 = 100$
- $r_4 = 104$
- $r_5 = 11$
PMA & Assembly Execution

Protected Memory

Unprotected Memory

\[
\begin{align*}
r_1 &= 14; \\
r_2 &= 101; \\
r_3 &= 100; \\
r_4 &= 104; \\
r_5 &= 11
\end{align*}
\]
PMA & Assembly Execution

Protected Memory

: 10 jmp r₄
   11 movi r₁ 14
   12 jmp r₁
   13 cmp r₁ r₂
   14 jmp r₃

:  

Unprotected Memory

: 100 jmp r₂
   101 movi r₁ 10
   102 jmp r₁
   103 sub r₁ r₂
   104 jmp r₅

:  

r₁ = 14 ; r₂ = 101 ; r₃ = 100 ; r₄ = 104 ; r₅ = 11
### PMA & Assembly Execution

#### Protected Memory

```assembly
10 jmp r_4
11 movi r_1 14
12 jmp r_1
13 cmp r_1 r_2
14 jmp r_3
```

#### Unprotected Memory

```assembly
100 jmp r_2
101 movi r_1 10
102 jmp r_1
103 sub r_1 r_2
104 jmp r_5
```

<table>
<thead>
<tr>
<th>r_1</th>
<th>r_2</th>
<th>r_3</th>
<th>r_4</th>
<th>r_5</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>101</td>
<td>100</td>
<td>104</td>
<td>11</td>
</tr>
</tbody>
</table>
PMA & Assembly Execution

Protected Memory

: 10 jmp r₄
11 movi r₁ 14
12 jmp r₁
13 cmp r₁ r₂
14 jmp r₃
: 

Unprotected Memory

: 100 jmp r₂ \( r₂=101 \)
101 movi r₁ 10
102 jmp r₁
103 sub r₁ r₂
104 jmp r₅
: 

\( r₁ = 10 \); \( r₂ = 101 \); \( r₃ = 100 \); \( r₄ = 104 \); \( r₅ = 11 \)
PMA & Assembly Execution

Protected Memory

```
10 jmp r4
11 movi r1 14
12 jmp r1
13 cmp r1 r2
14 jmp r3
```

Unprotected Memory

```
100 jmp r2
101 movi r1 10
102 jmp r1
103 sub r1 r2
104 jmp r5
```

\[ r_1 = 10 ; r_2 = 101 ; r_3 = 100 ; r_4 = 104 ; r_5 = 11 \]
PMA & Assembly Execution

Protected Memory

```
10 jmp r4
11 movi r1 14
12 jmp r1
13 cmp r1 r2
14 jmp r3
```

Unprotected Memory

```
100 jmp r2
101 movi r1 10
102 jmp r1
103 sub r1 r2
104 jmp r5
```

\[ r_1 = 10 \ ; \ r_2 = 101 \ ; \ r_3 = 100 \ ; \ r_4 = 104 \ ; \ r_5 = 11 \]
PMA & Assembly Execution

Protected Memory

: 10 jmp r_4
11 movi r_1 14
12 jmp r_1
13 cmp r_1 r_2
14 jmp r_3
:  

Unprotected Memory

: 100 jmp r_2
101 movi r_1 10
102 jmp r_1  
103 sub r_1 r_2
104 jmp r_5
:  

r_1 = 10; r_2 = 101; r_3 = 100;  

r_4 = 104; r_5 = 11
### PMA & Assembly Execution

<table>
<thead>
<tr>
<th>Protected Memory</th>
<th>Unprotected Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>10 jmp rᵣ&lt;sub&gt;4&lt;/sub&gt;</td>
<td>100 jmp r&lt;sub&gt;2&lt;/sub&gt;</td>
</tr>
<tr>
<td>11 movi r&lt;sub&gt;1&lt;/sub&gt; 14</td>
<td>101 movi r&lt;sub&gt;1&lt;/sub&gt; 10</td>
</tr>
<tr>
<td>12 jmp r&lt;sub&gt;1&lt;/sub&gt;</td>
<td>102 jmp r&lt;sub&gt;1&lt;/sub&gt;</td>
</tr>
<tr>
<td>13 cmp r&lt;sub&gt;1&lt;/sub&gt; r&lt;sub&gt;2&lt;/sub&gt;</td>
<td>103 sub r&lt;sub&gt;1&lt;/sub&gt; r&lt;sub&gt;2&lt;/sub&gt;</td>
</tr>
<tr>
<td>14 jmp r&lt;sub&gt;3&lt;/sub&gt;</td>
<td>104 jmp r&lt;sub&gt;5&lt;/sub&gt;</td>
</tr>
<tr>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>

PC → r = 10

---

10 is an entry point

r<sub>1</sub> = 10 ; r<sub>2</sub> = 101 ; r<sub>3</sub> = 100 ; r<sub>4</sub> = 104 ; r<sub>5</sub> = 11
PMA & Assembly Execution

Protected Memory

: 10 jmp r_4
11 movi r_1 14
12 jmp r_1
13 cmp r_1 r_2
14 jmp r_3
:

Unprotected Memory

: 100 jmp r_2
101 movi r_1 10
102 jmp r_1
103 sub r_1 r_2
104 jmp r_5
:

r_1 = 10 ; r_2 = 101 ; r_3 = 100 ;
r_4 = 104 ; r_5 = 11
PMA & Assembly Execution

Protected Memory

```
10 jmp r₄  \[ r₄ = 104 \]
11 movi r₁ 14
12 jmp r₁
13 cmp r₁ r₂
14 jmp r₃
```

Unprotected Memory

```
100 jmp r₂
101 movi r₁ 10
102 jmp r₁
103 sub r₁ r₂
104 jmp r₅
```

\[ r₁ = 10 ; r₂ = 101 ; r₃ = 100 ; r₄ = 104 ; r₅ = 11 \]
### PMA & Assembly Execution

#### Protected Memory

<table>
<thead>
<tr>
<th>PC</th>
<th>Instruction</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>jmp r₄</td>
<td>r₄ = 104</td>
</tr>
<tr>
<td>11</td>
<td>movi r₁  14</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>jmp r₁</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>cmp r₁ r₂</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>jmp r₃</td>
<td></td>
</tr>
</tbody>
</table>

#### Unprotected Memory

<table>
<thead>
<tr>
<th>PC</th>
<th>Instruction</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>jmp r₂</td>
<td></td>
</tr>
<tr>
<td>101</td>
<td>movi r₁  10</td>
<td></td>
</tr>
<tr>
<td>102</td>
<td>jmp r₁</td>
<td></td>
</tr>
<tr>
<td>103</td>
<td>sub r₁ r₂</td>
<td></td>
</tr>
<tr>
<td>104</td>
<td>jmp r₅</td>
<td></td>
</tr>
</tbody>
</table>

r₁ = 10 ; r₂ = 101 ; r₃ = 100 ; r₄ = 104 ; r₅ = 11
PMA & Assembly Execution

Protected Memory

```
: 10 jmp r₄
 11 movi r₁ 14
 12 jmp r₁
 13 cmp r₁ r₂
 14 jmp r₃
:
```

Unprotected Memory

```
: 100 jmp r₂
 101 movi r₁ 10
 102 jmp r₁
 103 sub r₁ r₂
 104 jmp r₅
:
```

```
r₁ = 10 ; r₂ = 101 ; r₃ = 100 ; r₄ = 104 ; r₅ = 11
```
PMA & Assembly Execution

Protected Memory

```
10 jmp r4
11 movi r1 14
12 jmp r1
13 cmp r1 r2
14 jmp r3
```

Unprotected Memory

```
100 jmp r2
101 movi r1 10
102 jmp r1
103 sub r1 r2
104 jmp r5 \( r_5 = 11 \)
```

\( r_1 = 10; \ r_2 = 101; \ r_3 = 100; \)
\( r_4 = 104; \ r_5 = 11 \)
PMA & Assembly Execution

Protected Memory

:  
10 jmp $r_4$
11 movi $r_1$ 14
12 jmp $r_1$
13 cmp $r_1$ $r_2$
14 jmp $r_3$

:  

: 11 is not an entry point

Unprotected Memory

:  
100 jmp $r_2$
101 movi $r_1$ 10
102 jmp $r_1$
103 sub $r_1$ $r_2$
104 jmp $r_5$

:  

PC →

r_1 = 10 ; r_2 = 101 ; r_3 = 100 ; r_4 = 104 ; r_5 = 11

r_5 = 11

\text{is not an entry point}
class C1
public create() : C1{
    this.hide();
    return new C1();
}
private hide(): Unit{
    return null;
}
object obj : C1;
class C1
public create(): C1{
    this.hide();
    return new C1();
}
private hide(): Unit{
    return null;
}
object obj : C1;
PMA in Action (N Modules)

\[
\begin{align*}
0x0001 & \quad \text{call} \ 0xb53 \\
0x0002 & \quad \text{movs} \ r_0 \ 0xb55 \\
& \quad \vdots \\
0x0b52 & \quad \text{movs} \ r_0 \ 0xb55 \\
0x0b53 & \quad \text{call} \ 0x0002 \\
0x0b54 & \quad \text{movs} \ r_0 \ 0xeb54 \\
0x0b55 & \quad \ldots \\
& \quad \vdots \\
0xab00 & \quad \text{jmp} \ 0xb53 \\
& \quad \vdots \\
0xeb52 & \quad \text{movs} \ r_0 \ 0xeb54 \\
0xeb53 & \quad \text{call} \ 0xab02 \\
0xeb54 & \quad \ldots \\
& \quad \vdots 
\end{align*}
\]
PMA in Action (N Modules)

0x0001  call 0xb53
0x0002  movs r0 0x0b55

0x0b52  movs r0 0x0b55
0x0b53  call 0x0002
0x0b54  movs r0 0xeb54
0x0b55  ...

0xab00  jmp 0x0b53

0xeb52  movs r0 0xeb54
0xeb53  call 0xab02
0xeb54  ...

ID /one.osf
ID /two.osf
/eight.osf
PMA in Action (N Modules)

```
0x0001  call 0xb53
0x0002  movs r0 0x0b55
...
0x0b52  movs r0 0x0b55
0x0b53  call 0x0002
0x0b54  movs r0 0xcbe54
0x0b55  ...
...
0xab00  jmp 0x0b53
...
0xeb52  movs r0 0xebe54
0xeb53  call 0xab02
0xeb54  ...
```

ID 1

ID 2
PMA in Action (N Modules)

```
0x0001  call 0xb53
0x0002  movs r0 0x0b55

0x0b52  movs r0 0x0b55
0x0b53  call 0x0002
0x0b54  movs r0 0xeb54
0x0b55  ...

0xab00  jmp 0x0b53

0xeb52  movs r0 0xeb54
0xeb53  call 0xab02
0xeb54  ...
```

ID 1

ID 2
PMA in Action (N Modules)

0x0001  call 0xb53
0x0002  movs r_o 0x0b55

0x0b52  movs r_o 0x0b55
0x0b53  call 0x0002
0x0b54  movs r_o 0xeb54
0x0b55  ...

0xab00  jmp 0x0b53

0xeb52  movs r_o 0xeb54
0xeb53  call 0xab02
0xeb54  ...

ID /one.osf
ID /two.osf
PMA in Action (N Modules)

```
0x0001  call 0xb53
0x0002  movs r0 0x0b55

0x0b52  movs r0 0x0b55
0x0b53  call 0x0002
0x0b54  movs r0 0xeb54
0x0b55  ...

0xab00  jmp 0x0b53

0xeb52  movs r0 0xeb54
0xeb53  call 0xab02
0xeb54  ...
```

ID /one.osf
ID /two.osf

r/w/x
PMA in Action (N Modules)

0x0001: call 0xb53
0x0002: movs r0 0x0b55

0x0b52: movs r0 0xb55
0x0b53: call 0x0002
0x0b54: movs r0 0xeb54
0x0b55: ...

0xab00: jmp 0x0b53

ID 1

0xeb52: movs r0 0xeb54
0xeb53: call 0xab02
0xeb54: ...

ID 2

r/w/x
### PMA in Action (N Modules)

<table>
<thead>
<tr>
<th>Address</th>
<th>Instruction</th>
<th>Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0001</td>
<td>call 0xb53</td>
<td></td>
</tr>
<tr>
<td>0x0002</td>
<td>movs r0 0x0b55</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x0b52</td>
<td>movs r0 0x0b55</td>
<td></td>
</tr>
<tr>
<td>0x0b53</td>
<td>call 0x0002</td>
<td></td>
</tr>
<tr>
<td>0x0b54</td>
<td>movs r0 0xeb54</td>
<td></td>
</tr>
<tr>
<td>0x0b55</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xab00</td>
<td>jmp 0x0b53</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0xeb52</td>
<td>movs r0 0xeb54</td>
<td></td>
</tr>
<tr>
<td>0xeb53</td>
<td>call 0xab02</td>
<td></td>
</tr>
<tr>
<td>0xeb54</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

**ID 1**

**ID 2**
PMA in Action (N Modules)

```
0x0001  call 0xb53
0x0002  movs r0 0xb55

0x0b52  movs r0 0xb55
0x0b53  call 0x0002
0x0b54  movs r0 0xeb54
0x0b55  ...

0xab00  jmp 0x0b53

0xeb52  movs r0 0xeb54
0xeb53  call 0xab02
0xeb54  ...
```

ID /one.osf
ID /two.osf
ID /eight.osf
PMA in Action (N Modules)

```
0x0001  call 0xb53
0x0002  movs r0 0xb55

0x0b52  movs r0 0xb55
0x0b53  call 0x0002
0x0b54  movs r0 0xeb54
0x0b55  ...

0xab00  jmp 0x0b53

0xeb52  movs r0 0xeb54
0xeb53  call 0xab02
0xeb54  ...
```

ID 1

ID 2
PMA in Action (N Modules)

```
0x0001   call 0xb53
0x0002   movs r0 0xb55
  ...
0xb52    movs r0 0xb55
0xb53    call 0x0002
0xb54    movs r0 0xeb54
0xb55    ...
  ...
0xab00   jmp 0xb53
  ...
0xeb52   movs r0 0xeb54
0xeb53   call 0xab02
0xeb54   ...
  ...
```

ID 1

ID 2
PMA & Trust

- Coarse-grained trust domains
PMA & Trust

- Coarse-grained trust domains
- The same trust domain fits the same Module
• Coarse-grained trust domains
• The same trust domain fits the same Module
• May require libraries and code to be split among different Modules
Threat Models

Different implementations address different attacks:

- Application
- OS
- OS device interface
- HW-SW Interface
- Firmware
- Hardware
Formalising PMA

Language: assembly
Formalising PMA

Language: assembly

• Instruction list
Formalising PMA

Language: assembly

- Instruction list
- Memory
Formalising PMA

Language: assembly

- Instruction list
- Memory
- Registers file
Formalising PMA

Language: assembly

- Instruction list
- Memory
- Registers file
- Module sizes

Helper functions for the ACP
References


