Secure Compilation of Object-Oriented Components to Protected Module Architectures

Marco Patrignani ¹  Dave Clarke ¹  Frank Piessens ¹

¹iMinds-DistriNet, Dept. Computer Science, KU Leuven

10 December 2013
Outline

1. Protected Modules Architectures
2. Secure (Fully Abstract) Compilation
3. What Can Go Wrong With...
   - Dynamic Memory Allocation
   - Exceptions
   - Cross-package Inheritance
Protected Modules Architecture (PMA)

- assembly-level isolation mechanism
protected modules architectures
secure (fully abstract) compilation
what can go wrong with... 

Protected Modules Architecture (PMA)

- assembly-level isolation mechanism
- several research prototypes: Fides [SP12], Sancus [NAD+13], Flicker [MPP+08], TrustVisor [MLQ+10], Smart [EFPT12]
Protected Modules Architecture (PMA)

- assembly-level isolation mechanism
- several research prototypes: Fides [SP12], Sancus [NAD+13], Flicker [MPP+08], TrustVisor [MLQ+10], Smart [EFPT12]
- industrial prototype too: Intel SGX [MAB+13]
Protected Modules Architecture (PMA)

- assembly-level isolation mechanism
- several research prototypes: Fides [SP12], Sancus [NAD+13], Flicker [MPP+08], TrustVisor [MLQ+10], Smart [EFPT12]
- industrial prototype too: Intel SGX [MAB+13]
- implemented via Hypervisor, Hardware, Software
Protected Modules Architecture (PMA)

- assembly-level isolation mechanism
- several research prototypes: Fides [SP12], Sancus [NAD+13], Flicker [MPP+08], TrustVisor [MLQ+10], Smart [EFPT12]
- industrial prototype too: Intel SGX [MAB+13]
- implemented via Hypervisor, Hardware, Software

Let’s see an example of PMA in action
PMA in Action

memory space

```
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

- memory space
- protected module = protected memory

```
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

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

```
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

- 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

- 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

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

```
0x0001 call 0xb53
0x0002 movs r0 0xb55
...
0x0b52 movs r0 0xb55
0x0b53 call 0x0002
0x0b54 movs r0 0x0001
0x0b55 ...
... 
0xab00 jmp 0xb53
0xab01 ...
```
PMA in Action

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

<table>
<thead>
<tr>
<th>Address</th>
<th>Operation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0001</td>
<td>call</td>
<td>0xb53</td>
</tr>
<tr>
<td>0x0002</td>
<td>movs r0</td>
<td>0x0b55</td>
</tr>
<tr>
<td>0x0b52</td>
<td>movs r0</td>
<td>0x0b55</td>
</tr>
<tr>
<td>0x0b53</td>
<td>call</td>
<td>0x0002</td>
</tr>
<tr>
<td>0x0b54</td>
<td>movs r0</td>
<td>0x0001</td>
</tr>
<tr>
<td>0x0b55</td>
<td></td>
<td>...</td>
</tr>
<tr>
<td>0xab00</td>
<td>jmp</td>
<td>0xb53</td>
</tr>
<tr>
<td>0xab01</td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>

Marco Patrignani, Dave Clarke, Frank Piessens
PMA in Action

```
0x0001 call 0xb53
0x0002 movs r0 0x0b55
...
0x0b52 movs r0 0x0b55
0x0b53 call 0x0002
0x0b54 movs r0 0x0001
0x0b55 ...
```

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

Marco Patrignani, Dave Clarke, Frank Piessens
PMA in Action

- Entry points for communication:
  - We need only 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

- memory space
- protected module $\Rightarrow$ protected memory
- split in code and data
- protected code is unrestricted
- unprotected code is restricted
- entry points for communication (■)

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

0xb52  movs r0 0x0b55
0xb53  call 0x0002
0xb54  movs r0 0x0001
0xb55  ...

0xab00  jmp 0xb53
0xab01  ...
```
PMA in Action

- memory space
- protected module = protected memory
- split in code and data
- protected code is unrestricted
- unprotected code is restricted
- entry points for communication (■)

Marco Patrignani, Dave Clarke, Frank Piessens
PMA in Action

- memory space
- protected module = protected memory
- split in code and data
- protected code is unrestricted
- unprotected code is restricted
- entry points for communication (■)
- we need only 1 module

```
0x0001 call 0xb53
0x0002 movs r0 0x0b55
...
0xb52 movs r0 0x0b55
0xb53 call 0x0002
0xb54 movs r0 0x0001
0xb55 ...
...
0xab00 jmp 0xb53
0xab01 ...
```
Languages of the Compiler

- target language: assembly
Languages of the Compiler

- target language: assembly
- source language: +/- Java
  jr [JR05]
Languages of the Compiler

- target language: assembly
- source language: +/- Java
  - component-based
  - private fields
  - programming to an interface
  - exceptions

```
package PI;

interface Account {
    public createAccount() : Foo;
}

extern extAccount : Account;

package PE;

class AccountClass implements PI.Account {
    AccountClass() { counter = 0; }
    public createAccount() : Account {
        return new PE.AccountClass();
    }

    private counter : Int;
}

object extAccount : AccountClass;
```
package PI;
interface Account {
  public createAccount() : Foo;
}
extern extAccount : Account;

package PE;
class AccountClass implements PI.Account {
  AccountClass() { counter = 0; }
  public createAccount() : Account {
    return new PE.AccountClass();
  }
  
  private counter : Int;
}
object extAccount : AccountClass;
Languages of the Compiler

Dynamic dispatch

v-tables

Secure stack

---

```
package PI;

interface Account {
    public createAccount() : Foo;
}

extern extAccount : Account;

package PE;

class AccountClass implements PI.Account {
    AccountClass() { counter = 0; }
    public createAccount() : Account {
        return new PE.AccountClass();
    }

    private counter : Int;
}

object extAccount : AccountClass;
```
Languages of the Compiler

- proxy to createAccount
- Dynamic dispatch
- v-tables
- Secure stack

```
package PI;
interface Account {
    public createAccount() : Foo;
}
extern extAccount : Account;

package PE;
class AccountClass implements PI.Account {
    AccountClass() { counter = 0; }
    public createAccount() : Account {
        return new PE.AccountClass();
    }
    private counter : Int;
}
object extAccount : AccountClass;
```
Languages of the Compiler

- proxy to `createAccount`
- `createAccount` body
- `constructor`
- Dynamic dispatch
- `v-tables`
- Secure stack
Languages of the Compiler

- Protected Modules Architectures
- Secure (Fully Abstract) Compilation
- What Can Go Wrong With...

Languages of the Compiler

- Proxy to createAccount
- createAccount body
- Constructor
- Dynamic dispatch
- V-tables
- Secure stack
- extAccount counter

Marco Patrignani, Dave Clarke, Frank Piessens
Secure Compilation, Informally

- Protect against low-level attackers

Source level

O1
O2

Ext 1
Ext 2

Distinction:
- Preservation of contextual equivalence
- Reflection of contextual equivalence (checks needed)
Secure Compilation, Informally

- Protect against low-level attackers

Source level:

```
O1 ← O1.createAccount() ➔
O2
```

Target level:

```
Ext 1 ➔
Ext 2
```

- Preservation of contextual equivalence
- Reflection of contextual equivalence (checks needed)

Marco Patrignani, Dave Clarke, Frank Piessens
Secure Compilation, Informally

- Protect against low-level attackers

- Preservation of contextual equivalence
- Reflection of contextual equivalence (checks needed)

Source level:
- O1
- O2
- O3

Target level:
- Ext 1
- Ext 2
Secure Compilation, Informally

- Protect against low-level attackers

```
return O3
```

```
O1
O2
O3

source level

⇒ Ext 1
⇒ Ext 2
```

```
Ext 1
Ext 2
```

Preservation of contextual equivalence

Reflection of contextual equivalence (checks needed)
Secure Compilation, Informally

- Protect against low-level attackers

Source level

O1
O2
O3

03.counter

Ext 1
Ext 2

Preservation of contextual equivalence

Reflection of contextual equivalence (checks needed)

Formally:

Marco Patrignani, Dave Clarke, Frank Piessens
Secure Compilation, Informally

- Protect against low-level attackers

03\_counter

Source level

O1
O2
O3

Target level

Ext 1
Ext 2
Secure Compilation, Informally

Secure Compilation

- Protect against low-level attackers

Source level:

O1
O2
O3

Target level:

Ext 1
Ext 2
Secure Compilation, Informally

- Protect against low-level attackers

Source level

O1
O2
O3

Target level

O1↓
O2↓

Ext 1
Ext 2

Ext 1↓
Ext 2↓
Secure Compilation, Informally

- Protect against low-level attackers

Source level

- O1
- O2
- O3

Target level

- O1↑.createAccount()
- O2↓
- O3↓

Ext 1
Ext 2
Ext 1↓
Ext 2↓
Secure Compilation, Informally

```
O1
O2
O3

⇓

Ext 1
Ext 2

⇒ Protect against low-level attackers

O1инф
O2инф
O3инф

\textbf{return } O3инф

⇒ Ext 1инф
⇒ Ext 2инф
```

- Protect against low-level attackers
- Preservation of contextual equivalence
- Reflection of contextual equivalence (checks needed)

Marco Patrignani, Dave Clarke, Frank Piessens
Secure Compilation, Informally

Source level

O1
O2
O3

Target level

O1$\downarrow$
O2$\downarrow$
O3$\downarrow$

Ext 1
Ext 2

Protect against low-level attackers

- Protect against low-level attackers
- Preservation of contextual equivalence
- Reflection of contextual equivalence (checks needed)

Formally:

Marco Patrignani, Dave Clarke, Frank Piessens
Secure Compilation, Informally

- Protect against low-level attackers

- Preservation of contextual equivalence
- Reflection of contextual equivalence (checks needed)

Protect against low-level attackers

Marco Patrignani, Dave Clarke, Frank Piessens
Secure Compilation, Informally

Source level:
- O1
- O2
- O3

Target level:
- O1
- O2
- O3

Ext 1
- Ext 1
- Ext 1

Ext 2
- Ext 2
- Ext 2

- Protect against low-level attackers
- Preservation of contextual equivalence
Secure Compilation, Informally

- Protect against low-level attackers
- Preservation of contextual equivalence
- Reflection of contextual equivalence (checks needed)
Secure Compilation, Informally

- Protect against low-level attackers
- Preservation of contextual equivalence
- Reflection of contextual equivalence (checks needed)

Formally:

\[ C_1 \simeq C_2 \iff C_1 \downarrow \simeq \downarrow C_2 \]
Things That Can Go Wrong

1. Protected Modules Architectures
2. Secure (Fully Abstract) Compilation
3. What Can Go Wrong With...
   - Dynamic Memory Allocation
   - Exceptions
   - Cross-package Inheritance
Dynamic Memory Allocation

Source level

O1
O2

Target level

O1↓
O2↓

Ext 1↓
Ext 2↓

Marco Patrignani, Dave Clarke, Frank Piessens
Dynamic Memory Allocation

Source level

O1
O2

Ext 1
Ext 2

01.createAccount()

Target level

O1
O2

Ext 1
Ext 2

Marco Patrignani, Dave Clarke, Frank Piessens
Dynamic Memory Allocation

Source level

O1
O2
O4
O3

Target level

O1↓
O2↓

Ext 1
Ext 2

Ext 1↓
Ext 2↓
Dynamic Memory Allocation

Source level

O1
O2
O4
O3

return 03

Ext 1
Ext 2

Target level

O1↓
O2↓

Ext 1↓
Ext 2↓
Dynamic Memory Allocation

Source level

O1
O2
O4
O3

Target level

01↓.createAccount()

Ext 1
Ext 2
Ext 1↓
Ext 2↓
Dynamic Memory Allocation

Source level

O1
O2
O4
O3

Target level

⇓

O1↓
O2↓
O4↓
O3↓

Ext 1
Ext 2

EXT 1

EXT 2

Object id guessing
map Oid to natural numbers
add Oid to map
lookup (O(1)) when number is received
dynamic typecheck for: current object
arguments
no need of extra information
Dynamic Memory Allocation

Source level

O1
O2
O4
O3

Target level

O1↓
O2↓
O4↓
O3↓

Ext 1
Ext 2

return O3↓

Ext 1↓
Ext 2↓

Marco Patrignani, Dave Clarke, Frank Piessens

Secure Compilation of OO Components to PMA

8/13
Dynamic Memory Allocation

Source level

O1
O2
O4
O3

Target level

0x001
0x005
0x009
0x00C

return 0x00C

Ext 1
Ext 2

Ext 1↓
Ext 2↓
Dynamic Memory Allocation

Source level

- O1
- O2
- O4
- O3

Target level

- 0x001
- 0x005
- 0x009
- 0x00C

Ext 1

Ext 2

0x009.createAccount()
Dynamic Memory Allocation

Source level:
- O1
- O2
- O4
- O3

Target level:
- 0x001
- 0x005
- 0x009
- 0x00C

Ext 1
- O4.createAccount()

Ext 2
- O3.createAccount()

O1
- 04.createAccount()
Dynamic Memory Allocation

Source level:

O1
O2
O4
O3

04.createAccount() → Ext 1
→ Ext 2

Target level:

0x001
0x005
0x009
0x00C

0x009.createAccount() → Ext 1
↓
↓
Ext 1
Ext 2

Object id guessing

Marco Patrignani, Dave Clarke, Frank Piessens
Dynamic Memory Allocation

Source level

O1
O2
O4
O3

Ext 1
Ext 2

Object id guessing
map Oid to natural numbers

Target level

0x001
0x005

Ext 1↓
Ext 2↓
Dynamic Memory Allocation

Source level

- O1
- O2
- O4
- O3

Ext 1
Ext 2

Target level

- 0x001 $\mapsto$ 1
- 0x005 $\mapsto$ 2

- Object id guessing
- map Oid to natural numbers

Marco Patrignani, Dave Clarke, Frank Piessens
**Dynamic Memory Allocation**

**Source level**
- O1
- O2
- O4
- O3

**Target level**
- 0x001 ➞ 1 ➔ Ext 1
- 0x005 ➞ 2 ➔ Ext 2

- 1.createAccount()

- Object id guessing
- map Oid to natural numbers
Dynamic Memory Allocation

Source level

O1
O2
O4
O3

Ext 1
Ext 2

Target level

0x001 ↦ 1
0x005 ↦ 2
0x009
0x00C

Ext 1↓
Ext 2↓

Object id guessing
map Oid to natural numbers

Marco Patrignani, Dave Clarke, Frank Piessens
Dynamic Memory Allocation

Source level

O1
O2
O4
O3

Ext 1
Ext 2

Target level

0x001 ↦ 1
0x005 ↦ 2
0x009
0x00C ↦ 3

Ext 1↓
Ext 2↓

- Object id guessing
- map Oid to natural numbers
- add Oid to map

Marco Patrignani, Dave Clarke, Frank Piessens
Dynamic Memory Allocation

Source level

O1
O2
O4
O3

Ext 1
Ext 2

Target level

0x001 \mapsto 1
0x005 \mapsto 2
0x009
0x00C \mapsto 3

Ext 1↓
return 3
Ext 2↓

- Object id guessing
- map Oid to natural numbers
- add Oid to map

Marco Patrignani, Dave Clarke, Frank Piessens
Dynamic Memory Allocation

- Object id guessing
- map Oid to natural numbers
- add Oid to map
- lookup (O(1)) when number is received

Source level:
- O1
- O2
- O4
- O3

Target level:
- 0x001 \(\mapsto\) 1
- 0x005 \(\mapsto\) 2
- 0x009
- 0x00C \(\mapsto\) 3
Dynamic Memory Allocation

Source level

O1 : Account
O2 : Pair
O3
O4

Target level

0x001 ↦ 1
0x005 ↦ 2
0x009
0x00C ↦ 3

Ext 1

Ext 2
- Object id guessing
- map Oid to natural numbers
- add Oid to map
- lookup (O(1)) when number is received

Ext 1↓
Ext 2↓
Dynamic Memory Allocation

Source level

O1: Account
O2: Pair
O4
O3

Target level

0x001 ↦ 1
0x005 ↦ 2
0x009
0x00C ↦ 3

02.createAccount()

Ext 1
Ext 2

- Object id guessing
- map Oid to natural numbers
- add Oid to map
- lookup (O(1)) when number is received

Ext 1↓
Ext 2↓
Dynamic Memory Allocation

Source level:

O1: Account
O2: Pair
O3
O4

Ext 1

Ext 2

Dynamic typecheck for: current object
arguments
no need of extra information

Target level:

0x001 ↦ 1
0x005 ↦ 2
0x009
0x00C ↦ 3

Object id guessing
map Oid to natural numbers
add Oid to map
lookup (O(1)) when number is received
Dynamic Memory Allocation

O1: Account
O2: Pair
O4
O3

0x001 \mapsto 1
0x005 \mapsto 2
0x009
0x00C \mapsto 3

2. createAccount()

Ext 1
Ext 2

• Object id guessing
• map Oid to natural numbers
• add Oid to map
• lookup (O(1)) when number is received
## Dynamic Memory Allocation

### Source level

- O1: Account
- O2: Pair
- O4
- O3

### Target level

- Ext 1
- Ext 2
  - Object id guessing
  - map Oid to natural numbers
  - add Oid to map
  - lookup (O(1)) when number is received

#### Ext 1

- O001 $\mapsto$ 1
- O005 $\mapsto$ 2
- O009
- O00C $\mapsto$ 3

#### Ext 2

- O001 $\mapsto$ 1
- O005 $\mapsto$ 2
- O009
- O00C $\mapsto$ 3
Dynamic Memory Allocation

Source level

O1 : Account
O2 : Pair
O4
O3

Target level

0x001 ⟷ 1
0x005 ⟷ 2
0x009
0x00C ⟷ 3

2. createAccount()

Ext 1

Ext 2

- Object id guessing
- map Oid to natural numbers
- add Oid to map
- lookup (O(1)) when number is received
- dynamic typecheck for: current object arguments

Marco Patrignani, Dave Clarke, Frank Piessens
Secure Compilation of OO Components to PMA
8/13
Dynamic Memory Allocation

Source level

O1 : Account
O2 : Pair
O4
O3

Target level

0x001 ↦ 1
0x005 ↦ 2
  0x009
  0x00C ↦ 3

Ext 1
Ext 2
- Object id guessing
- map Oid to natural numbers
- add Oid to map
- lookup (O(1)) when number is received
- dynamic typecheck for: current object arguments
- no need of extra information

Marco Patrignani, Dave Clarke, Frank Piessens
Secure Compilation of OO Components to PMA
Exceptions

Stack

- $f_s$
- $g$
- $h_s$
- $i$
- $l_s$

throw e

Secure stack

- $f_s$
- $h_s$
- $l_s$

throw e

Insecure stack

- $g$
- $i$
Exceptions

Stack

$\text{fs}$

$g$

$h_s$

$i$

$l_s$

$1$

throw e

Secure stack

$\text{fs}$

$h_s$

$l_s$

throw e

Insecure stack

$g$

$i$

Marco Patrignani, Dave Clarke, Frank Piessens

Secure Compilation of OO Components to PMA
Exceptions

Stack

$f_s$

$g$

$h_s$

$i$

$l_s$

Stack

Secure stack

$f_s$

$h_s$

$l_s$

throw e

Insecure stack

$g$

$i$

throw e
Exceptions

Stack

$f_s$

$g$

$h_s$

$i$

$l_s$

throw e

Secure stack

$f_s$

$h_s$

$l_s$

throw e

Insecure stack

$g$

$i$

Marco Patrignani, Dave Clarke, Frank Piessens

Secure Compilation of OO Components to PMA
Exceptions

Stack:

1. $l_s$
2. $i$
3. $h_s$
4. $g$

throw e

Secure stack:

1. $l_s$
2. $h_s$
3. $f_s$

throw e

Insecure stack:

1. $g$
2. $i$

Marco Patrignani, Dave Clarke, Frank Piessens
Exceptions

Stack

1. $l_s$
2. $i$
3. $h_s$
4. $g$
5. $f_s$

throw e

Secure stack

1. $l_s$
2. $h_s$
3. $f_s$

Secure stack

throw e

Insecure stack

1. $i$
2. $g$

Secure Compilation of OO Components to PMA
Exceptions

Stack

5 \(f_s\)
4 \(g\)
3 \(h_s\)
2 \(i\)
1 \(l_s\)

throw e

Secure stack

\(f_s\)
\(h_s\)
\(l_s\)

throw e

Insecure stack

\(g\)
\(i\)
Exceptions

Stack:

5. \( f_s \)
4. \( g \)
3. \( h_s \)
2. \( i \)
1. \( l_s \)

throw e

Secure stack:

- 1. \( l_s \)
- 2. \( h_s \)
- 3. \( f_s \)

throw e

Insecure stack:

- 1. \( g \)
- 2. \( i \)
Exceptions

Stack

Secure stack

Insecure stack

Record passed exceptions

Marco Patrignani, Dave Clarke, Frank Piessens

Secure Compilation of OO Components to PMA
Exceptions

Secure stack

Insecure stack

Record passed exceptions
Check that exception could be thrown

Marco Patrignani, Dave Clarke, Frank Piessens
Secure Compilation of OO Components to PMA
Exceptions

Record passed exceptions
Check that exception could be thrown
Exceptions

Stack

- $f_s$
- $g$
- $h_s$
- $i$
- $l_s$

Throw e

Secure stack

- $f_s$
- $h_s$
- $l_s$

Record passed exceptions
Check that exception could be thrown

Insecure stack

- $g$
- $i$

Protected Modules Architectures
Secure (Fully Abstract) Compilation
What Can Go Wrong With...
Dynamic Memory Allocation
Exceptions
Cross-package Inheritance

Marco Patrignani, Dave Clarke, Frank Piessens
Secure Compilation of OO Components to PMA
Cross-package Inheritance

O1: JointAccount
O1.counter
O1.limit

Source level

```java
package PSUP;

class Account {
    public getBalance(): Int {
        ...
    }

    private counter : Int;
}

package PSUB;

class JuniorAccount extends PSUP.Account {
    public getBalance() : Int {
        return super.getBalance();
    }

    private limit : Int;
}
```

Marco Patrignani, Dave Clarke, Frank Piessens
Cross-package Inheritance

O1 : JointAccount
O1.counter
O1.limit

Target level

O1↓
O1↓.counter
O1↓.limit

Source level

package PSUP;
class Account {
    public getBalance(): Int {
        ...
    }
    private counter : Int;
}

class JuniorAccount extends PSUP.Account {
    public getBalance() : Int {
        return super.getBalance();
    }
    private limit : Int;
}

- PSUP protected PSUB no Marco Patrignani, Dave Clarke, Frank Piessens
Cross-package Inheritance

Source level

O1 : JointAccount
O1.counter
O1.limit

Target level

[Diagram]

```java
package PSUP;

class Account {
    public getBalance(): Int {
        ...
    }
    private counter : Int;
}

package PSUB;

class JuniorAccount extends PSUP.Account {
    public getBalance() : Int {
        return super.getBalance();
    }
    private limit : Int;
}
```

- PSUP protected PSUB

Marco Patrignani, Dave Clarke, Frank Piessens
Cross-package Inheritance

Source level

O1 : JointAccount
O1.counter
O1.limit

Target level

O1 ↓
O1↓.counter
O1↓.limit

```
package PSUP;

class Account {
    public getBalance():Int {
        ...
    }  
    private counter : Int;
}

package PSUB;

class JuniorAccount extends PSUP.Account {
    public getBalance() : Int {
        return super.getBalance();
    }  
    private limit : Int;
}
```

• PSUP protected PSUB no
Cross-package Inheritance

Source level

O1: JointAccount
O1.counter
O1.limit

Target level

O1↓
O1↓.counter
O1↓.limit

```java
package PSUP;
class Account {
    public getBalance(): Int { ... }
    private counter : Int;
}

class JuniorAccount extends PSUP.Account {
    public getBalance(): Int {
        return super.getBalance();
    }
    private limit : Int;
}
```

- PSUP protected PSUB no
Cross-package Inheritance

Source level

O1 : JointAccount
O1.counter
O1.limit

Target level

O1↓SUP
O1↓.counter
O1↓.limit

```
package PSUP;

class Account {
   public getBalance():Int {
      ...
   }
   private counter : Int;
}

package PSUB;

class JuniorAccount extends PSUP.Account {
   public getBalance() : Int {
      return super.getBalance();
   }
   private limit : Int;
}
```

- PSUP protected PSUB no
Cross-package Inheritance

Source level

O1: JointAccount
O1.counter
O1.limit

Target level

O1↓SUP
O1↓SUB

super

O1↓SUP
O1↓SUB

package PSUP;
class Account {
    public getBalance(): Int {
        ...
    }
    private counter : Int;
}

package PSUB;
class JuniorAccount extends PSUP.Account {
    public getBalance() : Int {
        return super.getBalance();
    }
    private limit : Int;
}

PSUP protected PSUB no...
Conclusion

- PMA allows the creation of secure (fully abstract) compilers
PMA allows the creation of secure (fully abstract) compilers
identified naïve mistakes
Conclusion

- PMA allows the creation of secure (fully abstract) compilers
- identified naïve mistakes
- proposed sound solution
Questions


