RT-DFI : Optimizing Data-Flow Integrity for Real-Time Systems

Nicolas Bellec* Guillaume Hiet† Simon Rokicki*
Frédéric Tronel† Isabelle Puaut*

*Univ Rennes, Inria, CNRS, IRISA, Rennes, France
†CentraleSupélec, Inria, Univ Rennes, CNRS, IRISA, Rennes, France

Real-time systems

System correctness depends on its response time

Real-time systems

System correctness depends on its response time

Real-time systems

System correctness depends on its response time

Real-time systems

System correctness depends on its response time

2-steps timing verification :

Real-time systems

System correctness depends on its response time

2-steps timing verification :

- Estimate the **Worst-Case Execution Time** (WCET) for each task

Real-time systems

System correctness depends on its response time

2-steps timing verification :

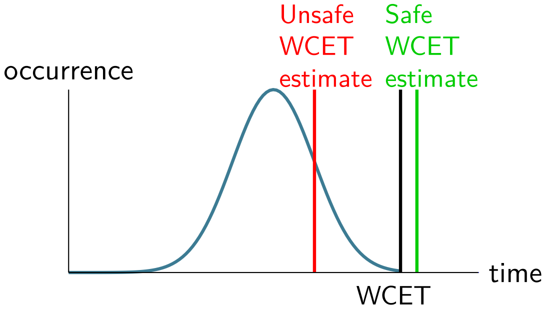
- Estimate the **Worst-Case Execution Time** (WCET) for each task
- Perform a **Schedulability Analysis**

WCET & Schedulability Analysis

WCET

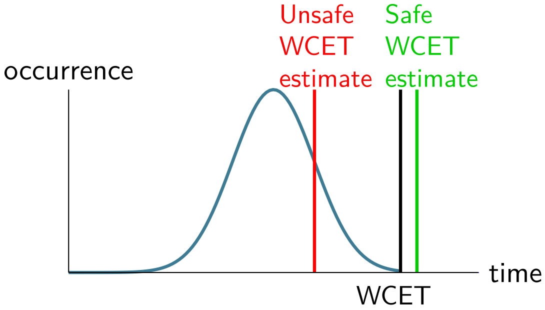
WCET & Schedulability Analysis

WCET



WCET & Schedulability Analysis

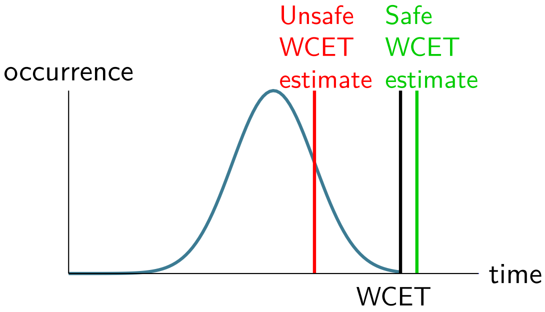
WCET



- Isolated tasks

WCET & Schedulability Analysis

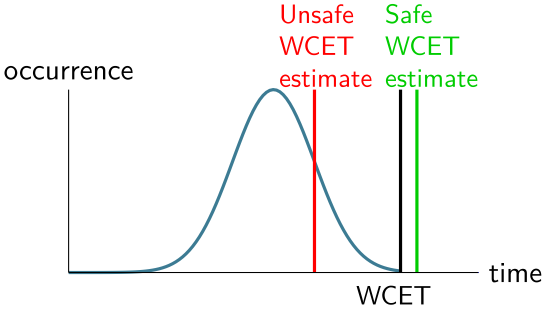
WCET



- Isolated tasks
- Estimated on the binary with hardware architecture information

WCET & Schedulability Analysis

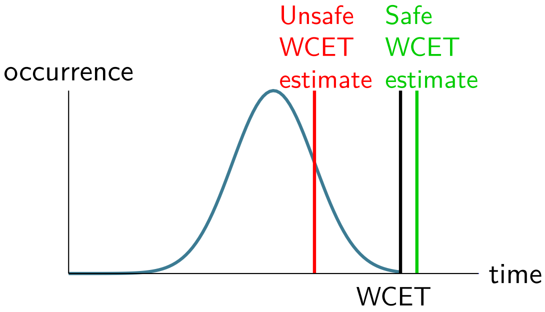
WCET



- Isolated tasks
- Estimated on the binary with hardware architecture information
- May require specific information (e.g. loopbounds)

WCET & Schedulability Analysis

WCET

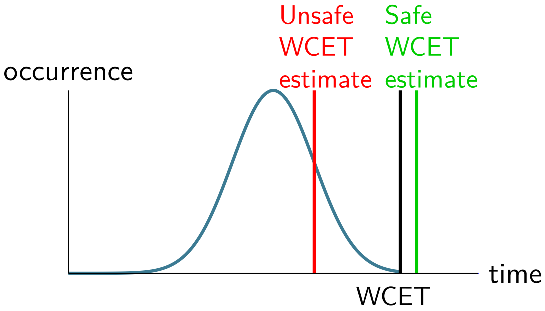


- Isolated tasks
- Estimated on the binary with hardware architecture information
- May require specific information (e.g. loopbounds)

Schedulability Analysis

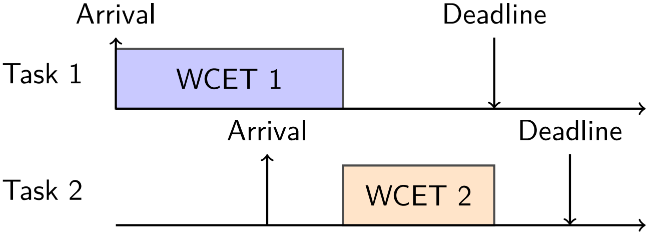
WCET & Schedulability Analysis

WCET



- Isolated tasks
- Estimated on the binary with hardware architecture information
- May require specific information (e.g. loopbounds)

Schedulability Analysis



Security for Real-Time Systems

Security for Real-Time Systems

- **Increased complexity and wireless communications**

Security for Real-Time Systems

- **Increased complexity and wireless communications**

Bluetooth, Wifi, IoT, ...[1]

[1] *Remote exploitation of an Unaltered Passenger Vehicle*, Valasek et Miller, BlackHat'15

Security for Real-Time Systems

- **Increased complexity and wireless communications**

Bluetooth, Wifi, IoT, ...[1]

- **Attackers break hypotheses**

[1] *Remote exploitation of an Unaltered Passenger Vehicle*, Valasek et Miller, BlackHat'15

Security for Real-Time Systems

- **Increased complexity and wireless communications**

Bluetooth, Wifi, IoT, ...[1]

- **Attackers break hypotheses**

Deadline miss ⟹

[1] *Remote exploitation of an Unaltered Passenger Vehicle*, Valasek et Miller, BlackHat'15

Security for Real-Time Systems

- **Increased complexity and wireless communications**

Bluetooth, Wifi, IoT, ...[1]

- **Attackers break hypotheses**

Deadline miss ⟹

WCET estimated **statically** to have an upper bound

[1] *Remote exploitation of an Unaltered Passenger Vehicle*, Valasek et Miller, BlackHat'15

Security for Real-Time Systems

- **Increased complexity and wireless communications**

Bluetooth, Wifi, IoT, ... [1]

- **Attackers break hypotheses**

Deadline miss $\Longrightarrow$

WCET estimated **statically** to have an upper bound
Can trade some overhead for more safety

[1] *Remote exploitation of an Unaltered Passenger Vehicle*, Valasek et Miller, BlackHat'15

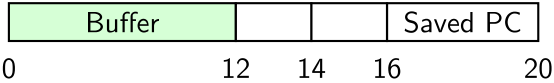
Real-Time Systems - Characteristics

Real-Time Systems - Characteristics

- **Unsafe low-level languages (C/C++)**

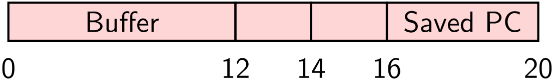
Real-Time Systems - Characteristics

- **Unsafe low-level languages (C/C++)**



Real-Time Systems - Characteristics

- **Unsafe low-level languages (C/C++)**



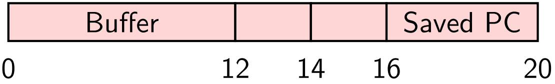
Buffer | | | Saved PC

0 12 14 16 20

Buffer Overflow

Real-Time Systems - Characteristics

- **Unsafe low-level languages (C/C++)**



Buffer Saved PC

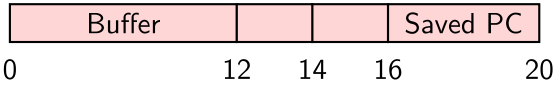
0 12 14 16 20

Buffer Overflow

Prone to Memory Corruption Attacks

Real-Time Systems - Characteristics

- **Unsafe low-level languages (C/C++)**



Buffer

Saved PC

0 12 14 16 20

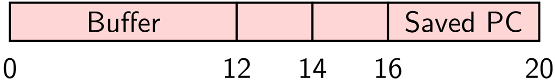
Buffer Overflow

Prone to Memory Corruption Attacks

- **Hard to update**

Real-Time Systems - Characteristics

- **Unsafe low-level languages (C/C++)**



Buffer

Saved PC

0 12 14 16 20

Buffer Overflow

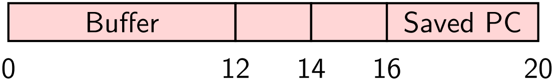
Prone to Memory Corruption Attacks

- **Hard to update**

Protection against unknown attacks

Real-Time Systems - Characteristics

- **Unsafe low-level languages (C/C++)**



Buffer			Saved PC

0 12 14 16 20

Buffer Overflow

Prone to Memory Corruption Attacks

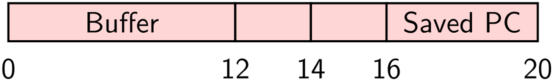
- **Hard to update**

Protection against unknown attacks

- **Predictable**

Real-Time Systems - Characteristics

- **Unsafe low-level languages (C/C++)**



Buffer Overflow

Prone to Memory Corruption Attacks

- **Hard to update**

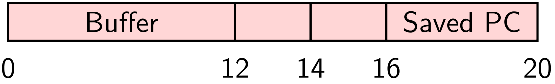
Protection against unknown attacks

- **Predictable**

Protection overhead **on the WCET** must be predictable statically

Real-Time Systems - Characteristics

- **Unsafe low-level languages (C/C++)**



Buffer

Saved PC

0 12 14 16 20

Buffer Overflow

Prone to Memory Corruption Attacks

- **Hard to update**

Protection against unknown attacks

- **Predictable**

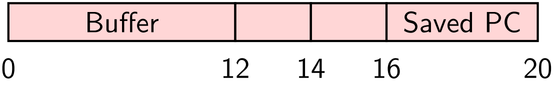
Protection overhead **on the WCET** must be predictable statically

Wanted

Robust and Predictable protection against Memory Corruption Attacks for Real-Time Systems

Real-Time Systems - Characteristics

- **Unsafe low-level languages (C/C++)**



Buffer

Saved PC

0 12 14 16 20

Buffer Overflow

Prone to Memory Corruption Attacks

- **Hard to update**

Protection against unknown attacks

- **Predictable**

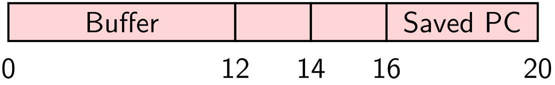
Protection overhead **on the WCET** must be predictable statically

Wanted

Robust and Predictable protection against Memory Corruption Attacks for Real-Time Systems

Real-Time Systems - Characteristics

- **Unsafe low-level languages (C/C++)**



Buffer Overflow

Prone to Memory Corruption Attacks

- **Hard to update**

Protection against unknown attacks

- **Predictable**

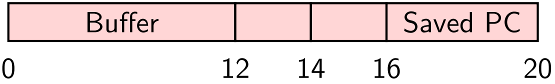
Protection overhead **on the WCET** must be predictable statically

Wanted

Robust and Predictable protection against Memory Corruption Attacks for Real-Time Systems

Real-Time Systems - Characteristics

- **Unsafe low-level languages (C/C++)**



Buffer

Saved PC

0 12 14 16 20

Buffer Overflow

Prone to Memory Corruption Attacks

- **Hard to update**

Protection against unknown attacks

- **Predictable**

Protection overhead **on the WCET** must be predictable statically

Wanted

Robust and Predictable protection against Memory Corruption Attacks for Real-Time Systems

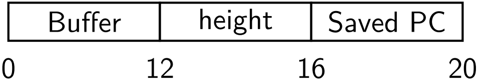
Attacker capacity

Attacker capacity

```c
char buffer[12];
char c = 255;
int height = getHeight();
for(int i = 0; c != 0; i++) {
  c = recv();
  buffer[i] = c;
}
if (height >= 10000)
  crash_plane()
```

Attacker capacity

```
char buffer[12];
char c = 255;
int height = getHeight();
for(int i = 0; c != 0; i++) {
  c = recv();
  buffer[i] = c;
}
if (height >= 10000)
  crash_plane()
```

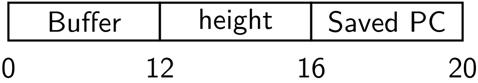


Attacker capacity

```
char buffer[12];
char c = 255;
int height = getHeight();
for(int i = 0; c != 0; i++) {
  c = recv();
  buffer[i] = c;
}
if (height >= 10000)
  crash_plane()
```

Input

'A'·12+'B'·4+'C'·4

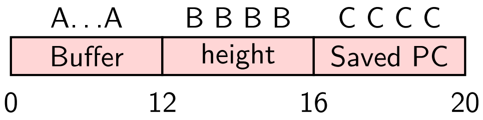


Attacker capacity

```
char buffer[12];
char c = 255;
int height = getHeight();
for(int i = 0; c != 0; i++) {
  c = recv();
  buffer[i] = c;
}
if (height >= 10000)
  crash_plane()
```

Input

'A'·12+'B'·4+'C'·4



A. . .A B B B B C C C C

Buffer height Saved PC

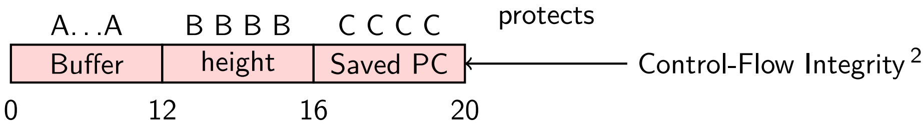
0 12 16 20

Attacker capacity

```
char buffer[12];
char c = 255;
int height = getHeight();
for(int i = 0; c != 0; i++) {
  c = recv();
  buffer[i] = c;
}
if (height >= 10000)
  crash_plane()
```

Input

'A'·12+'B'·4+'C'·4



A...A

B B B B

C C C C

Buffer

height

Saved PC

0

12

16

20

protects

Control-Flow Integrity[2]

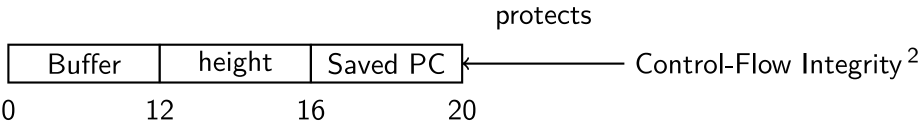
[2] Control-Flow Integrity for Real-Time Embedded Systems, *Walls et al.*, ECRTS'19

Attacker capacity

```c
char buffer[12];
char c = 255;
int height = getHeight();
for(int i = 0; c != 0; i++) {
  c = recv();
  buffer[i] = c;
}
if (height >= 10000)
  crash_plane()
```

Input

'A'·12+'B'·4



protects

Buffer	height	Saved PC

0 12 16 20

Control-Flow Integrity [2]

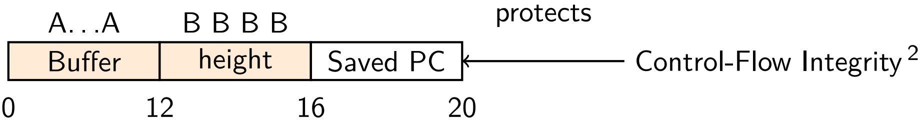
[2] Control-Flow Integrity for Real-Time Embedded Systems, *Walls et al.*, ECRTS'19

Attacker capacity

```
char buffer[12];
char c = 255;
int height = getHeight();
for(int i = 0; c != 0; i++) {
  c = recv();
  buffer[i] = c;
}
if (height >= 10000)
  crash_plane()
```

Input

'A'·12+'B'·4



[2] Control-Flow Integrity for Real-Time Embedded Systems, *Walls et al.*, ECRTS'19

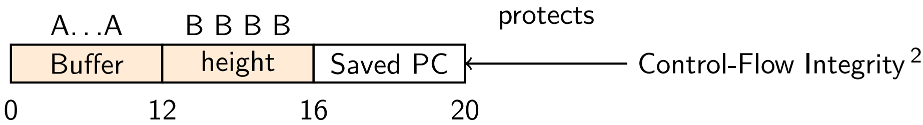
Attacker capacity

```
char buffer[12];
char c = 255;
int height = getHeight();
for(int i = 0; c != 0; i++) {
  c = recv();
  buffer[i] = c;
}
if (height >= 10000)
  crash_plane()
```

Input

'A'·12+'B'·4

Attacker controled ⚠



A. . .A

B B B B

Buffer

height

Saved PC

protects

Control-Flow Integrity [2]

0

12

16

20

[2] Control-Flow Integrity for Real-Time Embedded Systems, *Walls et al.*, ECRTS'19

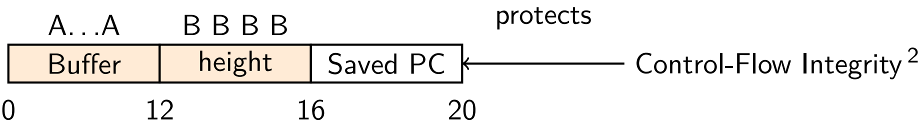
Attacker capacity

```
char buffer[12];
char c = 255;
int height = getHeight();
for(int i = 0; c != 0; i++) {
  c = recv();
  buffer[i] = c;
}
if (height >= 10000)
  crash_plane()
```

Input

'A'·12+'B'·4

Attacker controled ⚠



Control-Flow Integrity[2]

Data-Flow Attacks[3]

[2] Control-Flow Integrity for Real-Time Embedded Systems, *Walls et al.*, ECRTS'19

[3] Non-Control-Data Attacks Are Realistic Threats, *Chen et al.*, USENIX'05

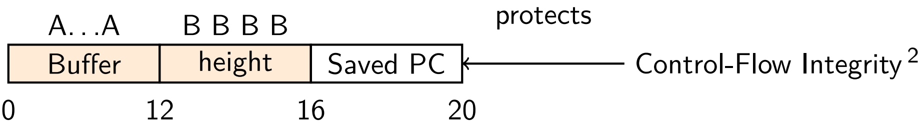
Attacker capacity

```
char buffer[12];
char c = 255;
int height = getHeight();
for(int i = 0; c != 0; i++) {
  c = recv();
  buffer[i] = c;
}
if (height >= 10000)
  crash_plane()
```

Input

'A'·12+'B'·4

Attacker controled ⚠



A...A B B B B

Buffer | height | Saved PC

0 12 16 20

protects

Control-Flow Integrity [2]

Data-Flow Attacks [3] $\Longrightarrow$ We want to protect **all memory operations**

[2] Control-Flow Integrity for Real-Time Embedded Systems, *Walls et al.*, ECRTS'19

[3] Non-Control-Data Attacks Are Realistic Threats, *Chen et al.*, USENIX'05

Our Work : Real-Time Data-Flow Integrity

[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

Our Work : Real-Time Data-Flow Integrity

Data-Flow Integrity[4] (DFI) :

[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

Our Work : Real-Time Data-Flow Integrity

Data-Flow Integrity[4] (DFI) : ✔ Robust

[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

Our Work : Real-Time Data-Flow Integrity

Data-Flow Integrity[4] (DFI) :

- ✔ Robust
- ✔ Static analysis **at compile time**

[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

Our Work : Real-Time Data-Flow Integrity

Data-Flow Integrity[4] (DFI) :

- ✔ Robust
- ✔ Static analysis **at compile time**
- ∼ WCET Overhead

[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

Our Work : Real-Time Data-Flow Integrity

Data-Flow Integrity[4] (DFI) :

✔ Robust

✔ Static analysis **at compile time**

∼ WCET Overhead

Goal

Reduce the overhead of DFI on WCET

[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

Data-Flow Integrity [4]

```
p <- r1 (store)
if (...)
  p <- r2 (store)
else
  x <- r3 (store)
p -> r4 (load)
```

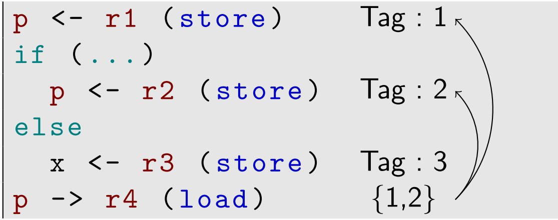
[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

Data-Flow Integrity[4]

```
p <- r1 (store)    Tag : 1
if (...)
  p <- r2 (store)  Tag : 2
else
  x <- r3 (store)  Tag : 3
p -> r4 (load)
```

[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

Data-Flow Integrity[4]

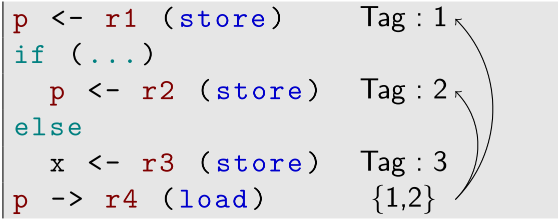


```
p <- r1 (store)     Tag : 1
if (...)
  p <- r2 (store)   Tag : 2
else
  x <- r3 (store)   Tag : 3
p -> r4 (load)       {1,2}
```

Static analysis
at compile time

[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

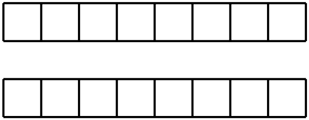
Data-Flow Integrity [4]



```
p <- r1 (store)     Tag : 1
if (...)
  p <- r2 (store)   Tag : 2
else
  x <- r3 (store)   Tag : 3
p -> r4 (load)       {1,2}
```

Static analysis
at compile time

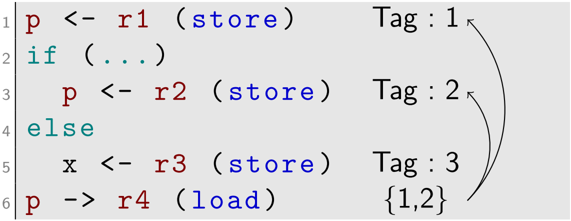
Memory



Runtime Definition Table

[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

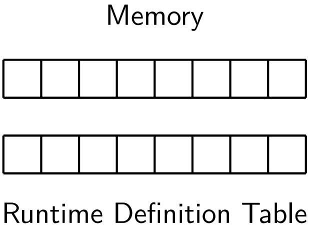
Data-Flow Integrity[4]



```
p <- r1 (store)      Tag : 1
if (...)
  p <- r2 (store)    Tag : 2
else
  x <- r3 (store)    Tag : 3
p -> r4 (load)        {1,2}
```

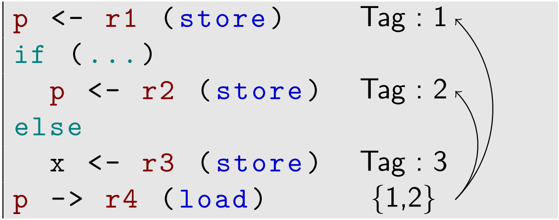
Static analysis
at compile time

p <- r2 (store)



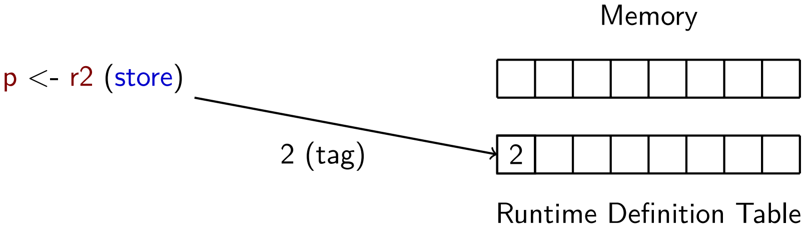
[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

Data-Flow Integrity [4]



```
p <- r1 (store)       Tag : 1
if (...)
  p <- r2 (store)     Tag : 2
else
  x <- r3 (store)     Tag : 3
p -> r4 (load)         {1,2}
```

Static analysis
at compile time

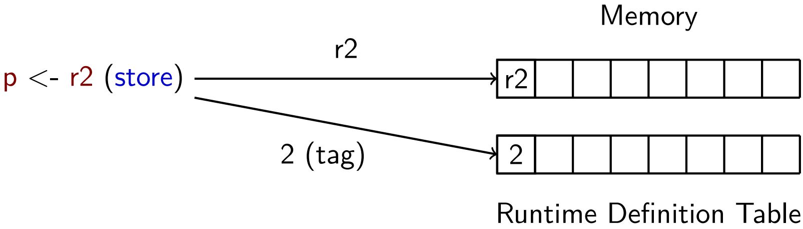


[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX ’06.

Data-Flow Integrity[4]

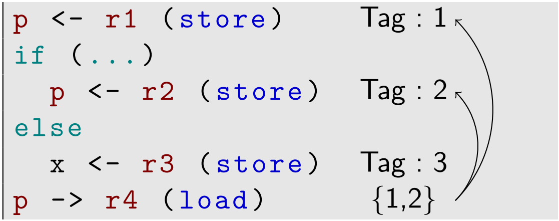
```
p <- r1 (store)      Tag : 1
if (...)
  p <- r2 (store)    Tag : 2
else
  x <- r3 (store)    Tag : 3
p -> r4 (load)        {1,2}
```

Static analysis
at compile time



[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

Data-Flow Integrity[4]



```
p <- r1 (store)      Tag : 1
if (...)
  p <- r2 (store)    Tag : 2
else
  x <- r3 (store)    Tag : 3
p -> r4 (load)        {1,2}
```

Static analysis
at compile time

p -> r4 (load)

Memory

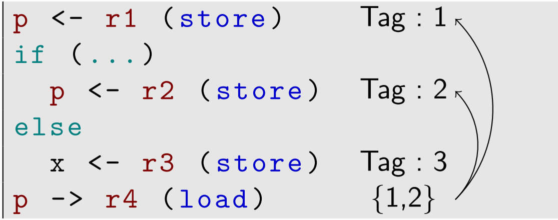
r2							

2							

Runtime Definition Table

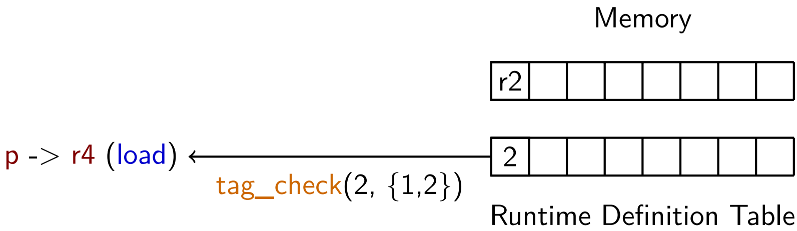
[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

Data-Flow Integrity[4]



```
p <- r1 (store)      Tag : 1
if (...)
  p <- r2 (store)    Tag : 2
else
  x <- r3 (store)    Tag : 3
p -> r4 (load)        {1,2}
```

Static analysis
at compile time

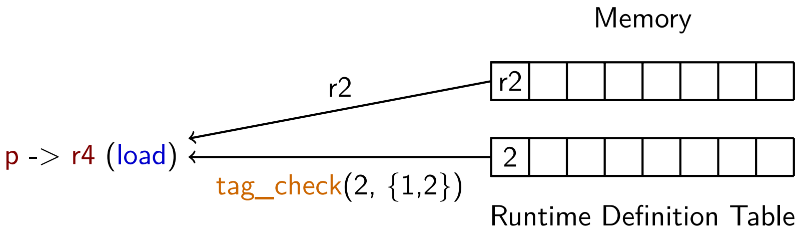


[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

Data-Flow Integrity[4]

```
p <- r1 (store)      Tag : 1
if (...)
  p <- r2 (store)    Tag : 2
else
  x <- r3 (store)    Tag : 3
p -> r4 (load)        {1,2}
```

Static analysis
at compile time



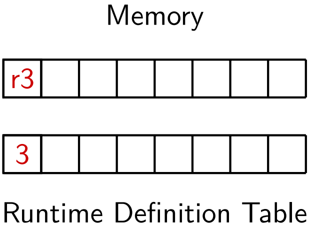
[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

Data-Flow Integrity [4]

```
p <- r1 (store)      Tag : 1
if (...)
  p <- r2 (store)    Tag : 2
else
  x <- r3 (store)    Tag : 3
p -> r4 (load)        {1,2}
```

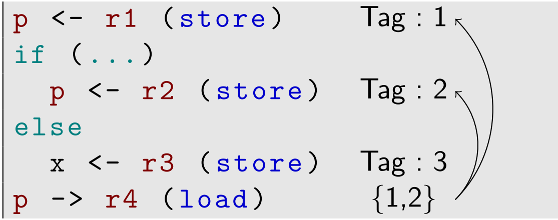
Static analysis
at compile time

p -> r4 (load)



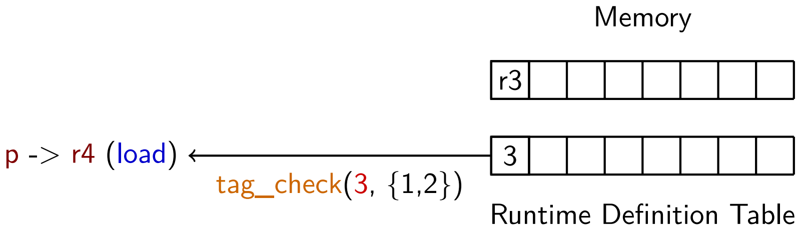
[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

Data-Flow Integrity [4]



```
p <- r1 (store)      Tag : 1
if (...)
  p <- r2 (store)    Tag : 2
else
  x <- r3 (store)    Tag : 3
p -> r4 (load)        {1,2}
```

Static analysis
at compile time



Memory

r3							

p -> r4 (load)

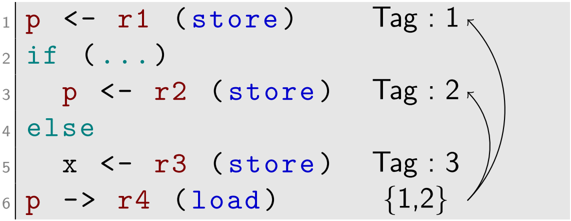
tag_check(3, {1,2})

3							

Runtime Definition Table

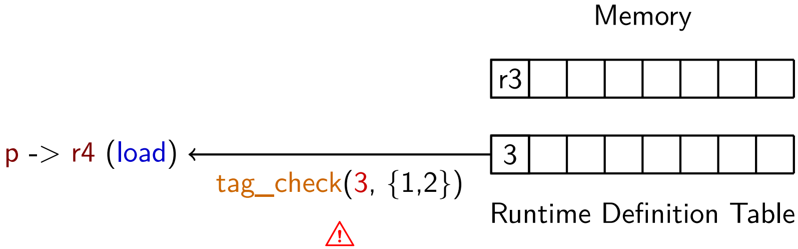
[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

Data-Flow Integrity [4]



```
p <- r1 (store)      Tag : 1
if (...)
  p <- r2 (store)    Tag : 2
else
  x <- r3 (store)    Tag : 3
p -> r4 (load)        {1,2}
```

Static analysis
at compile time



[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

DFI[4] - Tag check existing optimizations

```
check_tag(tag,{1,3,4,5})
```

[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

DFI[4] - Tag check existing optimizations

```
check_tag(tag,{1,3,4,5})
```

$\eqsim$

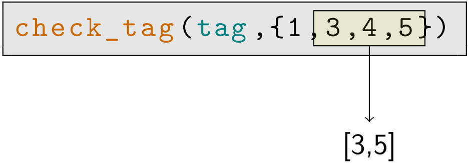
```
if tag == 1 or
   tag == 3 or
   tag == 4 or
   tag == 5:
  continue

raise Exception()
```

4 conditions

[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

DFI[4] - Tag check existing optimizations



```
check_tag(tag,{1,3,4,5})
```

[3,5]

$\simeq$

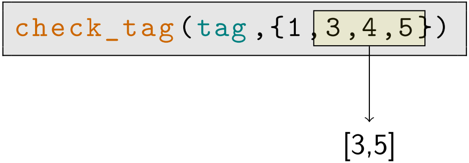
```
if tag == 1 or
   tag == 3 or
   tag == 4 or
   tag == 5:
  continue

raise Exception()
```

4 conditions

[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

DFI[4] - Tag check existing optimizations



```
check_tag(tag,{1,3,4,5})
```

[3,5]

=

```
if tag == 1 or
    3 <= tag <= 5:
  continue

raise Exception()
```

With intervals : 3 conditions

[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

DFI[4] - Check Tag Optimizations

[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

DFI[4] - Check Tag Optimizations

- Number of intervals

[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

DFI[4] - Check Tag Optimizations

- Number of intervals

```
if tag == 1 or
   3 <= tag <= 4 or
   6 <= tag <= 7:
  continue
```

[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

DFI[4] - Check Tag Optimizations

- Number of intervals

```
if tag == 1 or
    3 <= tag <= 4 or
    6 <= tag <= 7:
  continue
```

$\xrightarrow{7 \mapsto 5}$

[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

DFI[4] - Check Tag Optimizations

- Number of intervals

```
if tag == 1 or
    3 <= tag <= 4 or
    6 <= tag <= 7:
  continue
```

$$\xrightarrow{7 \mapsto 5}$$

```
if tag == 1 or
    3 <= tag <= 6:
  continue
```

[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

DFI[4] - Check Tag Optimizations

- Number of intervals ([4] uses a greedy algorithm)

```
if tag == 1 or
    3 <= tag <= 4 or
    6 <= tag <= 7:
  continue
```

$\xrightarrow{7 \mapsto 5}$

```
if tag == 1 or
    3 <= tag <= 6:
  continue
```

[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

DFI[4] - Check Tag Optimizations

- Number of intervals ([4] uses a greedy algorithm)

```
if tag == 1 or
    3 <= tag <= 4 or
    6 <= tag <= 7:
  continue
```

$$\xrightarrow{7 \mapsto 5}$$

```
if tag == 1 or
    3 <= tag <= 6:
  continue
```

- Interval order

[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

DFI[4] - Check Tag Optimizations

- Number of intervals ([4] uses a greedy algorithm)

```
if tag == 1 or
    3 <= tag <= 4 or
    6 <= tag <= 7:
  continue
```

$\xrightarrow{7 \mapsto 5}$

```
if tag == 1 or
    3 <= tag <= 6:
  continue
```

- Interval order

```
if tag == 1 or
    3 <= tag <= 4:
  continue
```

[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

DFI[4] - Check Tag Optimizations

- Number of intervals ([4] uses a greedy algorithm)

```
if tag == 1 or
   3 <= tag <= 4 or
   6 <= tag <= 7:
  continue
```

$\xrightarrow{7 \mapsto 5}$

```
if tag == 1 or
   3 <= tag <= 6:
  continue
```

- Interval order

```
if tag == 1 or
   3 <= tag <= 4:
  continue
```

```
if 3 <= tag <= 4 or
   tag == 1:
  continue
```

[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX ’06.

DFI[4] - Check Tag Optimizations

- Number of intervals ([4] uses a greedy algorithm)

```
if tag == 1 or
   3 <= tag <= 4 or
   6 <= tag <= 7:
  continue
```

$7 \mapsto 5$

```
if tag == 1 or
   3 <= tag <= 6:
  continue
```

- Interval order

```
if tag == 1 or
   3 <= tag <= 4:
  continue
```

```
if 3 <= tag <= 4 or
   tag == 1:
  continue
```

[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

DFI[4] - Check Tag Optimizations

- Number of intervals ([4] uses a greedy algorithm)

```
if tag == 1 or
    3 <= tag <= 4 or
    6 <= tag <= 7:
  continue
```

$7 \mapsto 5$

```
if tag == 1 or
    3 <= tag <= 6:
  continue
```

- Interval order (un-used by [4])

```
if tag == 1 or
    3 <= tag <= 4:
  continue
```

```
if 3 <= tag <= 4 or
    tag == 1:
  continue
```

[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

RT-DFI - Worst-Case Execution Path

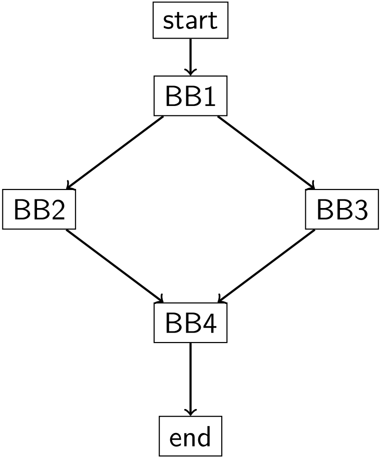
RT-DFI - Worst-Case Execution Path

WCET computed based on **Worst-Case Execution Path** (WCEP)

RT-DFI - Worst-Case Execution Path

WCET computed based on **Worst-Case Execution Path** (WCEP)

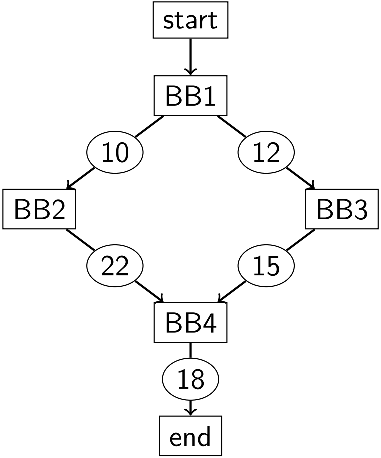
Control-Flow Graph



RT-DFI - Worst-Case Execution Path

WCET computed based on **Worst-Case Execution Path** (WCEP)

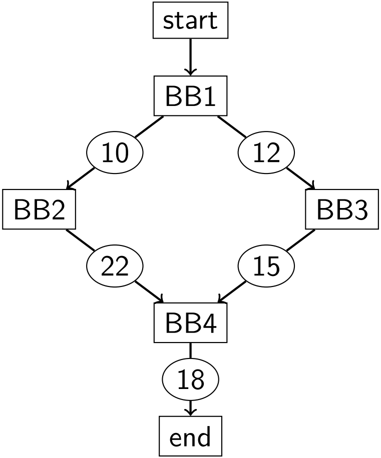
Control-Flow Graph with basic-block cost



RT-DFI - Worst-Case Execution Path

WCET computed based on **Worst-Case Execution Path** (WCEP)

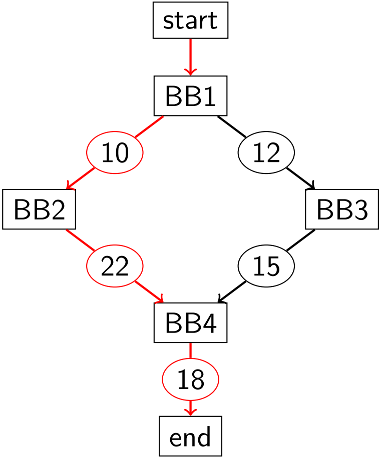
Control-Flow Graph with basic-block cost



RT-DFI - Worst-Case Execution Path

WCET computed based on **Worst-Case Execution Path** (WCEP)

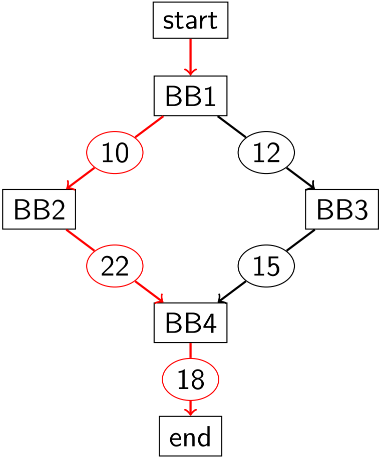
Control-Flow Graph
with basic-block cost



RT-DFI - Worst-Case Execution Path

WCET computed based on **Worst-Case Execution Path** (WCEP)

Control-Flow Graph with basic-block cost



Focus the optimization on the WCEP

RT-DFI - Value analysis

RT-DFI - Value analysis

WCET analysis uses a **Value analysis** (at the **binary level**)

RT-DFI - Value analysis

WCET analysis uses a **Value analysis** (at the **binary level**)
Provides additional information on the loads in the WCEP

RT-DFI - Value analysis

WCET analysis uses a **Value analysis** (at the **binary level**)
Provides additional information on the loads in the WCEP

Context

Information used to distinguish different uses of the same program location

RT-DFI - Value analysis

WCET analysis uses a **Value analysis** (at the **binary level**)
Provides additional information on the loads in the WCEP

Context

Information used to distinguish different uses of the same program location

```c
int f(int i) {
  return i;
}

x = f(1);

y = f(2);
```

RT-DFI - Value analysis

WCET analysis uses a **Value analysis** (at the **binary level**)
Provides additional information on the loads in the WCEP

Context

Information used to distinguish different uses of the same program location

Without context

```
int f(int i) {
  return i;
}

x = f(1); ⓐ

y = f(2); ⓑ
```

RT-DFI - Value analysis

WCET analysis uses a **Value analysis** (at the **binary level**)
Provides additional information on the loads in the WCEP

Context

Information used to distinguish different uses of the same program location

Without context

```
int f(int i) {
  return i;
}

x = f(1); ⓐ

y = f(2); ⓑ
```

RT-DFI - Value analysis

WCET analysis uses a **Value analysis** (at the **binary level**)
Provides additional information on the loads in the WCEP

Context

Information used to distinguish different uses of the same program location

Without context

```
int f(int i) {
  return i;  ← i ∈ {1}
}

x = f(1); a

y = f(2); b
```

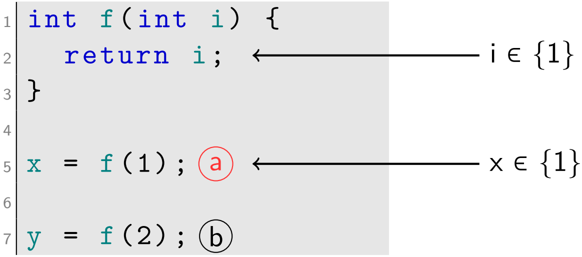
RT-DFI - Value analysis

WCET analysis uses a **Value analysis** (at the **binary level**)
Provides additional information on the loads in the WCEP

Context

Information used to distinguish different uses of the same program location

Without context



```
int f(int i) {
  return i;
}

x = f(1); ⓐ

y = f(2); ⓑ
```

$i \in \{1\}$

$x \in \{1\}$

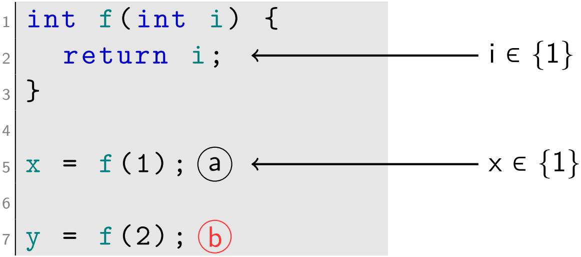
RT-DFI - Value analysis

WCET analysis uses a **Value analysis** (at the **binary level**)
Provides additional information on the loads in the WCEP

Context

Information used to distinguish different uses of the same program location

Without context



```
int f(int i) {
  return i;
}

x = f(1); ⓐ

y = f(2); ⓑ
```

$i \in \{1\}$

$x \in \{1\}$

RT-DFI - Value analysis

WCET analysis uses a **Value analysis** (at the **binary level**)
Provides additional information on the loads in the WCEP

Context

Information used to distinguish different uses of the same program location

Without context

```
int f(int i) {
  return i;     ←  i ∈ {1} ∪ {2} = {1,2}
}

x = f(1); ⓐ     ←  x ∈ {1}

y = f(2); ⓑ
```

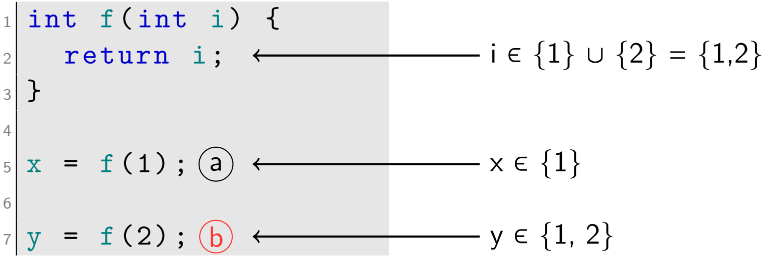
RT-DFI - Value analysis

WCET analysis uses a **Value analysis** (at the **binary level**)
Provides additional information on the loads in the WCEP

Context

Information used to distinguish different uses of the same program location

Without context



```
int f(int i) {
  return i;
}

x = f(1); ⓐ

y = f(2); ⓑ
```

$i \in \{1\} \cup \{2\} = \{1,2\}$

$x \in \{1\}$

$y \in \{1, 2\}$

RT-DFI - Value analysis

WCET analysis uses a **Value analysis** (at the **binary level**)

Provides additional information on the loads in the WCEP

Context

Information used to distinguish different uses of the same program location

With context

```
int f(int i) {
  return i;
}

x = f(1); ⓐ

y = f(2); ⓑ
```

RT-DFI - Value analysis

WCET analysis uses a **Value analysis** (at the **binary level**)
Provides additional information on the loads in the WCEP

Context

Information used to distinguish different uses of the same program location

With context

```
int f(int i) {
  return i;
}

x = f(1); ⓐ

y = f(2); ⓑ
```

RT-DFI - Value analysis

WCET analysis uses a **Value analysis** (at the **binary level**)
Provides additional information on the loads in the WCEP

Context

Information used to distinguish different uses of the same program location

With context

```
int f(int i) {
  return i;
}

x = f(1); ⓐ

y = f(2); ⓑ
```

$\longleftarrow$ $i \in \{(a \rightarrow)\ 1\}$

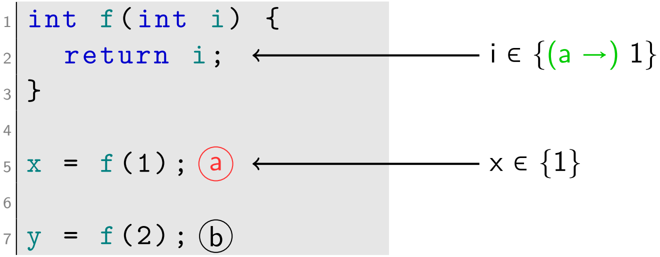
RT-DFI - Value analysis

WCET analysis uses a **Value analysis** (at the **binary level**)
Provides additional information on the loads in the WCEP

Context

Information used to distinguish different uses of the same program location

With context



```
int f(int i) {
  return i;
}

x = f(1); ⓐ

y = f(2); ⓑ
```

$i \in \{(a \rightarrow)\ 1\}$

$x \in \{1\}$

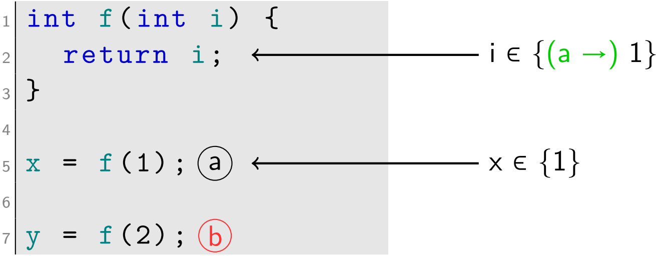
RT-DFI - Value analysis

WCET analysis uses a **Value analysis** (at the **binary level**)
Provides additional information on the loads in the WCEP

Context

Information used to distinguish different uses of the same program location

With context



```
int f(int i) {
  return i;
}

x = f(1); a

y = f(2); b
```

$i \in \{(a \rightarrow)\ 1\}$

$x \in \{1\}$

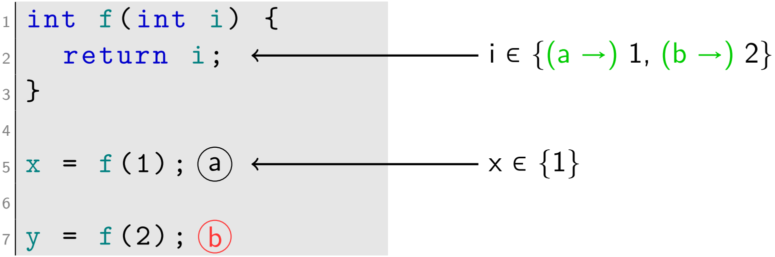
RT-DFI - Value analysis

WCET analysis uses a **Value analysis** (at the **binary level**)
Provides additional information on the loads in the WCEP

Context

Information used to distinguish different uses of the same program location

With context



```
int f(int i) {
  return i;
}

x = f(1); ⓐ

y = f(2); ⓑ
```

$i \in \{(a \rightarrow)\ 1,\ (b \rightarrow)\ 2\}$

$x \in \{1\}$

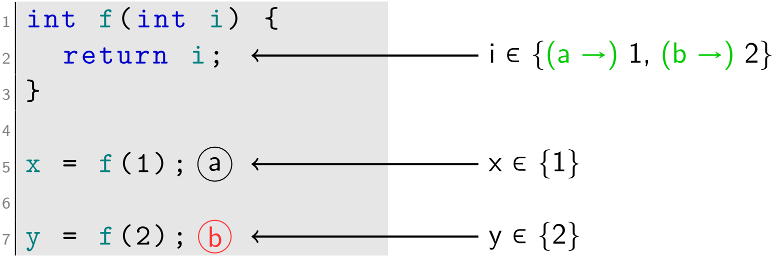
RT-DFI - Value analysis

WCET analysis uses a **Value analysis** (at the **binary level**)
Provides additional information on the loads in the WCEP

Context

Information used to distinguish different uses of the same program location

With context



```
int f(int i) {
  return i;
}

x = f(1); ⓐ

y = f(2); ⓑ
```

i ∈ {(a →) 1, (b →) 2}

x ∈ {1}

y ∈ {2}

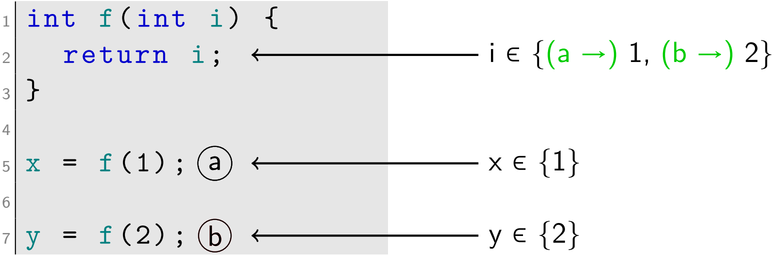
RT-DFI - Value analysis

WCET analysis uses a **Value analysis** (at the **binary level**)
Provides additional information on the loads in the WCEP

Context

Information used to distinguish different uses of the same program location

With context



```
int f(int i) {
   return i;
}

x = f(1); a

y = f(2); b
```

$i \in \{(a \rightarrow) 1, (b \rightarrow) 2\}$

$x \in \{1\}$

$y \in \{2\}$

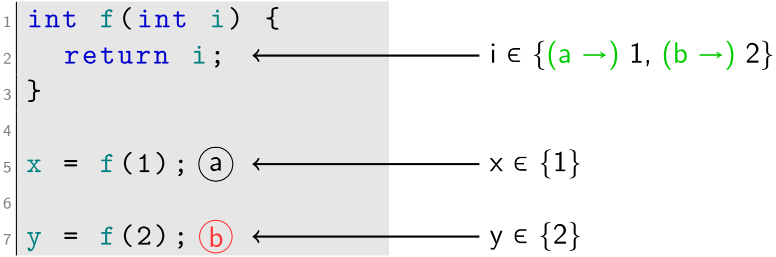
RT-DFI - Value analysis

WCET analysis uses a **Value analysis** (at the **binary level**)
Provides additional information on the loads in the WCEP

Context

Information used to distinguish different uses of the same program location

With context



```
int f(int i) {
  return i;
}

x = f(1); ⓐ

y = f(2); ⓑ
```

$i \in \{(a \rightarrow)\ 1,\ (b \rightarrow)\ 2\}$

$x \in \{1\}$

$y \in \{2\}$

Provide refined load information for the optimizations

RT-DFI - Combining Everything

RT-DFI - Combining Everything

```
if tag == 1 or
   3 <= tag <= 5:
  continue
```

RT-DFI - Combining Everything

Tags of the context

```
if tag == 1 or
   3 <= tag <= 5:
  continue
```

RT-DFI - Combining Everything

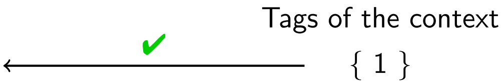
```
if tag == 1 or
    3 <= tag <= 5:
  continue
```

Tags of the context

$\{ 1 \}$

RT-DFI - Combining Everything

```
if tag == 1 or
    3 <= tag <= 5:
  continue
```



Tags of the context

{ 1 }

RT-DFI - Combining Everything

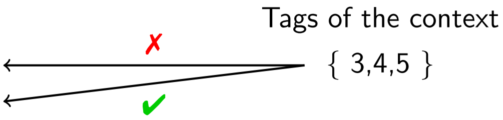
```
if tag == 1 or
   3 <= tag <= 5:
  continue
```

Tags of the context

{ 3,4,5 }

RT-DFI - Combining Everything

```
if tag == 1 or
    3 <= tag <= 5:
  continue
```



Tags of the context

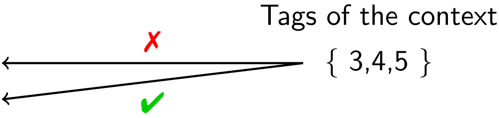
{ 3,4,5 }

✗

✔

RT-DFI - Combining Everything

```
if tag == 1 or
    3 <= tag <= 5:
  continue
```



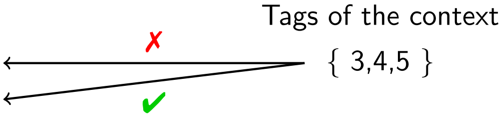
Tags of the context

{ 3,4,5 }

Possible optimizations

RT-DFI - Combining Everything

```
if tag == 1 or
    3 <= tag <= 5:
  continue
```



Tags of the context

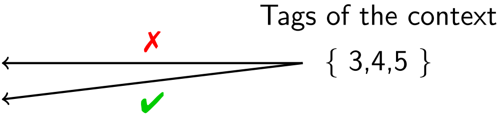
{ 3,4,5 }

Possible optimizations

```
if tag == 1 or
    3 <= tag <= 5:
  continue
```

RT-DFI - Combining Everything

```
if tag == 1 or
   3 <= tag <= 5:
  continue
```



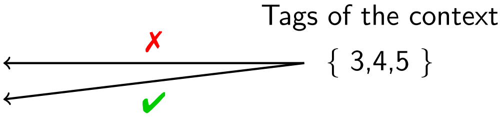
Possible optimizations

```
if tag == 1 or
   3 <= tag <= 5:
  continue
```

```
if 3 <= tag <= 5 or
   tag == 1:
  continue
```

RT-DFI - Combining Everything

```
if tag == 1 or
   3 <= tag <= 5:
  continue
```



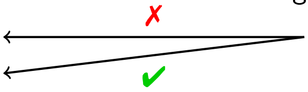
Possible optimizations

```
if tag == 1 or
   3 <= tag <= 5:
  continue
```

```
if 3 <= tag <= 5 or
   tag == 1:
  continue
```

RT-DFI - Combining Everything

```
if tag == 1 or
    3 <= tag <= 5:
  continue
```



Tags of the context

{ 3,4,5 }

Possible optimizations

```
if tag == 1 or
    3 <= tag <= 5:
  continue
```

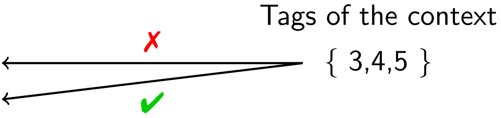
⤫

```
if 3 <= tag <= 5 or
    tag == 1:
  continue
```

Local

RT-DFI - Combining Everything

```
if tag == 1 or
   3 <= tag <= 5:
  continue
```



Tags of the context

{ 3,4,5 }

Possible optimizations

```
if tag == 1 or
   3 <= tag <= 5:
  continue
```

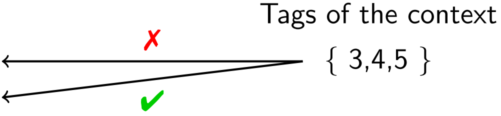
```
if 3 <= tag <= 5 or
   tag == 1:
  continue
```

Local

{ 1,[3,5] }

RT-DFI - Combining Everything

```
if tag == 1 or
    3 <= tag <= 5:
  continue
```



Tags of the context

{ 3,4,5 }

Possible optimizations

```
if tag == 1 or
    3 <= tag <= 5:
  continue
```

⤨

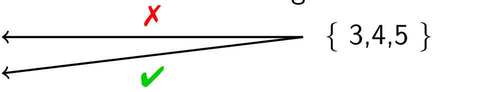
```
if 3 <= tag <= 5 or
    tag == 1:
  continue
```

Local

$\{ 1,[3,5] \} \xrightarrow{5 \mapsto 2}$

RT-DFI - Combining Everything

```
if tag == 1 or
    3 <= tag <= 5:
  continue
```



Tags of the context

{ 3,4,5 }

✗

✔

Possible optimizations

```
if tag == 1 or
    3 <= tag <= 5:
  continue
```

⤨

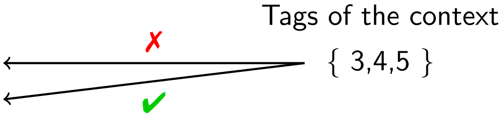
```
if 3 <= tag <= 5 or
    tag == 1:
  continue
```

Local

$$\{ 1,[3,5] \} \xrightarrow{5 \mapsto 2} \{ [1,4] \}$$

RT-DFI - Combining Everything

```
if tag == 1 or
    3 <= tag <= 5:
  continue
```



Possible optimizations

```
if tag == 1 or
    3 <= tag <= 5:
  continue
```

⤧

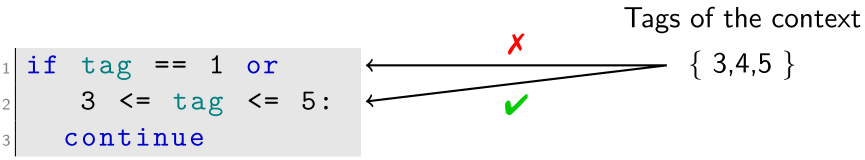
```
if 3 <= tag <= 5 or
    tag == 1:
  continue
```

Local

$$\{\ 1,[3,5]\ \} \xrightarrow{5 \mapsto 2} \{\ [1,4]\ \}$$

$$\{\ [4,5],7\ \} \xrightarrow{} \{\ 2,4,7\ \}$$ ⚠

RT-DFI - Combining Everything



Tags of the context

```
if tag == 1 or
   3 <= tag <= 5:
  continue
```

✗

✔

{ 3,4,5 }

Possible optimizations

```
if tag == 1 or
   3 <= tag <= 5:
  continue
```

⤨

```
if 3 <= tag <= 5 or
   tag == 1:
  continue
```

Local

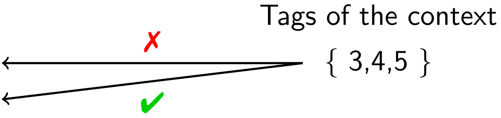
$$\{\ 1,[3,5]\ \} \xrightarrow{5 \mapsto 2} \{\ [1,4]\ \}$$

$$\{\ [4,5],7\ \} \longrightarrow \{\ 2,4,7\ \} \quad ⚠$$

Global

RT-DFI - Combining Everything

```
if tag == 1 or
    3 <= tag <= 5:
  continue
```



Possible optimizations

```
if tag == 1 or
    3 <= tag <= 5:
  continue
```

⤨

```
if 3 <= tag <= 5 or
    tag == 1:
  continue
```

Local

$$\{\ 1,[3,5]\ \} \xrightarrow{5 \mapsto 2} \{\ [1,4]\ \}$$

$$\{\ [4,5],7\ \} \longrightarrow \{\ 2,4,7\ \} \quad ⚠$$

Global

We use **Integer Linear Programming**
to optimize the **DFI** on the whole **WCEP**

Integer Linear Programming

Integer Linear Programming

Key Idea

Optimization of a linear function under linear constraints

Integer Linear Programming

Key Idea

Optimization of a linear function under linear constraints

Variables

Integer Linear Programming

Key Idea

Optimization of a linear function under linear constraints

Variables

Constraints

Integer Linear Programming

Key Idea

Optimization of a linear function under linear constraints

Variables

$V_{\mathcal{N}}$: Integer Variables

$v_1, \ldots, v_n \in V_{\mathcal{N}}$

Constraints

Integer Linear Programming

Key Idea

Optimization of a linear function under linear constraints

Variables

$V_{\mathcal{N}}$: Integer Variables

$v_1, \ldots, v_n \in V_{\mathcal{N}}$

Constraints

$$C_i : \sum_j a_{i,j} \cdot v_i \quad \underset{\uparrow}{\square} \quad b_i$$

$\in \{\leq, <, \geq, >, =\}$

Integer Linear Programming

Key Idea

Optimization of a linear function under linear constraints

Variables

$V_{\mathcal{N}}$: Integer Variables
$v_1, \ldots, v_n \in V_{\mathcal{N}}$

Constraints

$$C_i : \sum_j a_{i,j} \cdot v_i \quad \underset{\uparrow}{\square} \quad b_i$$

$\in \{\leq, <, \geq, >, =\}$

Goal

Find $v_1, \ldots, v_n$ maximizing a linear function (e.g. $v_1 + v_2 - 2 \cdot v_3$) under constraints C_i

Integer Linear Programming

Key Idea

Optimization of a linear function under linear constraints

Variables

$V_{\mathcal{N}}$: Integer Variables
$v_1, \ldots, v_n \in V_{\mathcal{N}}$

Constraints

$$C_i : \sum_j a_{i,j} \cdot v_i \;\; \underset{\uparrow}{\square} \;\; b_i$$

$\in \{\leq, <, \geq, >, =\}$

Goal

Find $v_1, \ldots, v_n$ maximizing a linear function (e.g. $v_1 + v_2 - 2 \cdot v_3$) under constraints C_i

- Current solvers are very efficient (CPLEX)

Integer Linear Programming

Key Idea

Optimization of a linear function under linear constraints

Variables

$V_{\mathcal{N}}$: Integer Variables
$v_1, \ldots, v_n \in V_{\mathcal{N}}$

Constraints

$$C_i : \sum_j a_{i,j} \cdot v_i \quad \underset{\in \{\leq, <, \geq, >, =\}}{\underset{\uparrow}{\square}} \quad b_i$$

Goal

Find $v_1, \ldots, v_n$ maximizing a linear function (e.g. $v_1 + v_2 - 2 \cdot v_3$) under constraints C_i

- Current solvers are very efficient (CPLEX)
- Used for industrial problems (e.g. WCET estimation)

Integer Linear Programming

Key Idea

Optimization of a linear function under linear constraints

Variables

$V_{\mathcal{N}}$: Integer Variables

$v_1, \ldots, v_n \in V_{\mathcal{N}}$

Constraints

$$C_i : \sum_j a_{i,j} \cdot v_i \quad \underset{\underset{\in \{\leq, <, \geq, >, =\}}{\uparrow}}{\square} \quad b_i$$

Goal

Find $v_1, \ldots, v_n$ maximizing a linear function (e.g. $v_1 + v_2 - 2 \cdot v_3$) under constraints C_i

- Current solvers are very efficient (CPLEX)
- Used for industrial problems (e.g. WCET estimation)

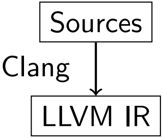
See paper for how the optimizations are modeled

RT-DFI - Implementation

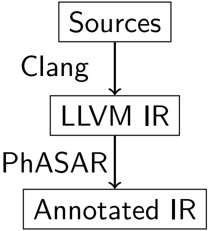
RT-DFI - Implementation

Sources

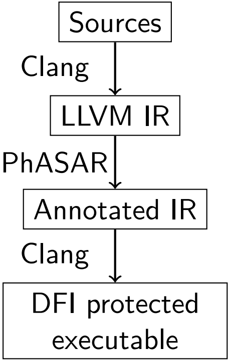
RT-DFI - Implementation



RT-DFI - Implementation



RT-DFI - Implementation


Sources

Clang

LLVM IR

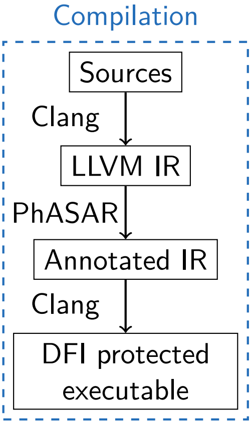
PhASAR

Annotated IR

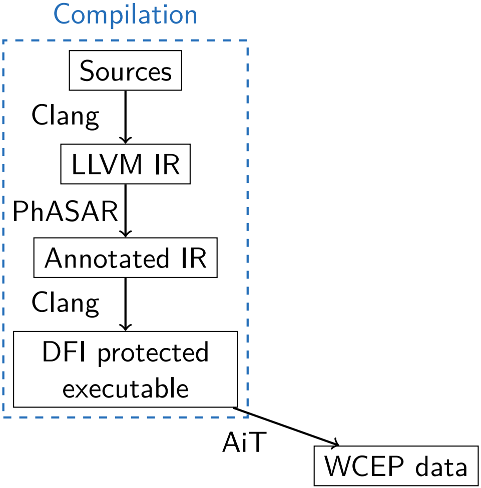
Clang

DFI protected executable

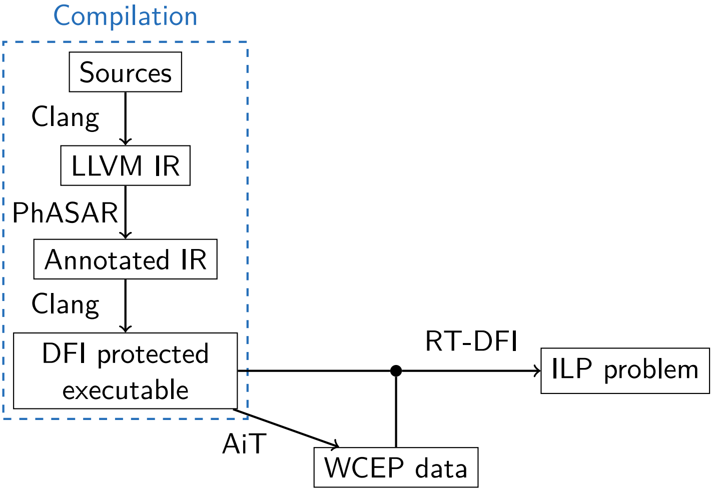
RT-DFI - Implementation



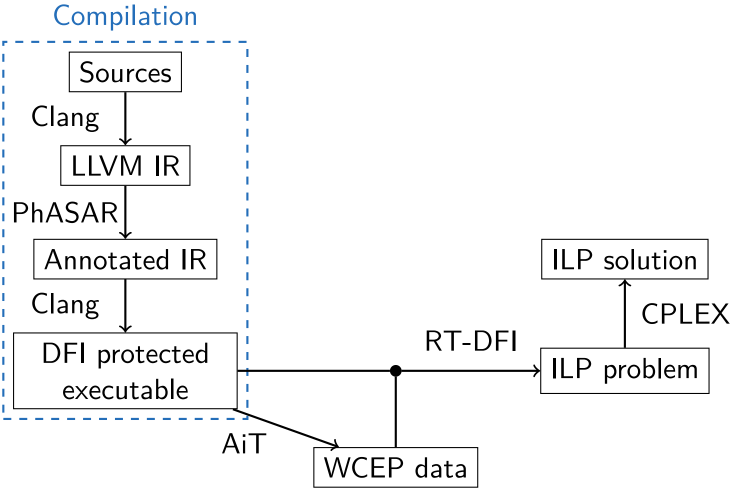
RT-DFI - Implementation



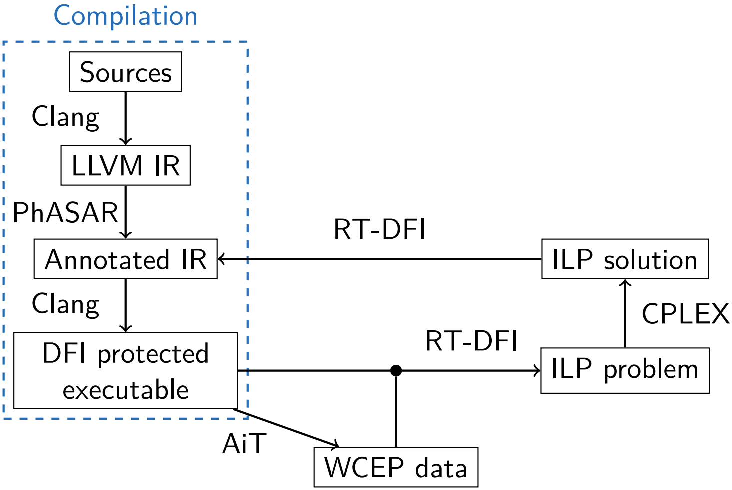
RT-DFI - Implementation



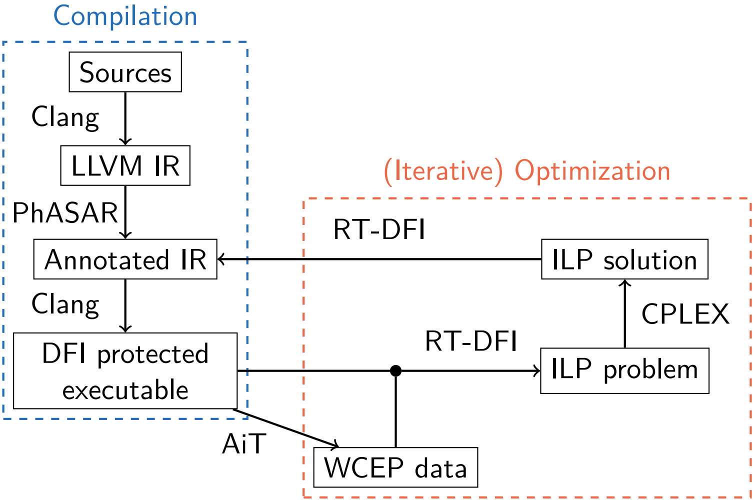
RT-DFI - Implementation



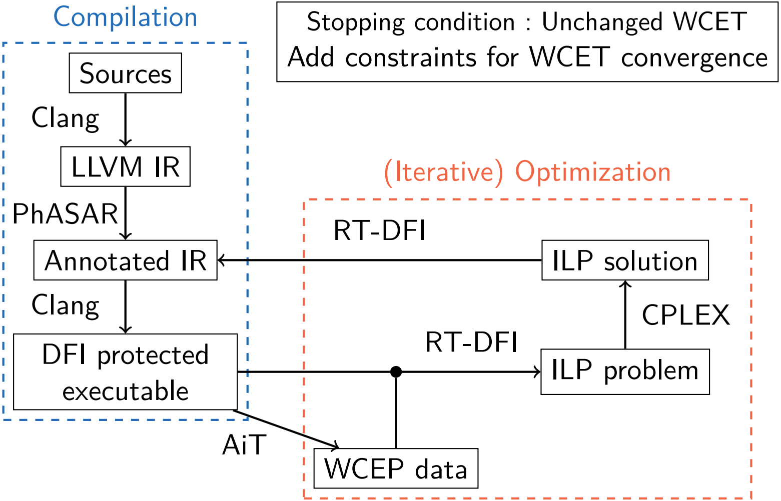
RT-DFI - Implementation



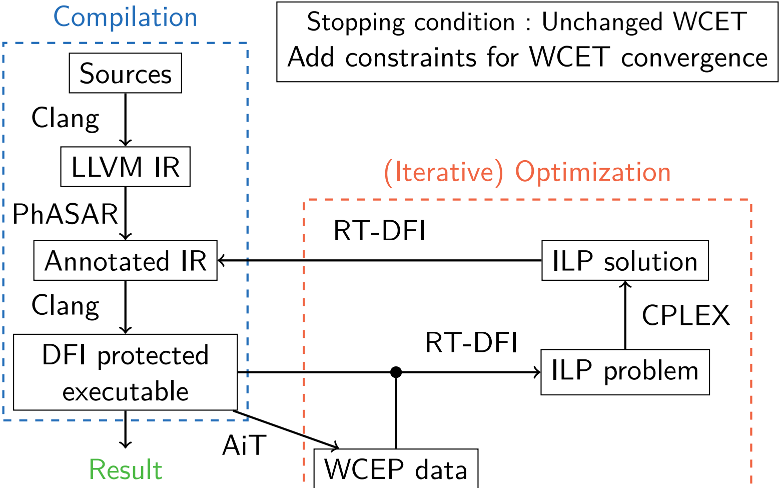
RT-DFI - Implementation



RT-DFI - Implementation



RT-DFI - Implementation

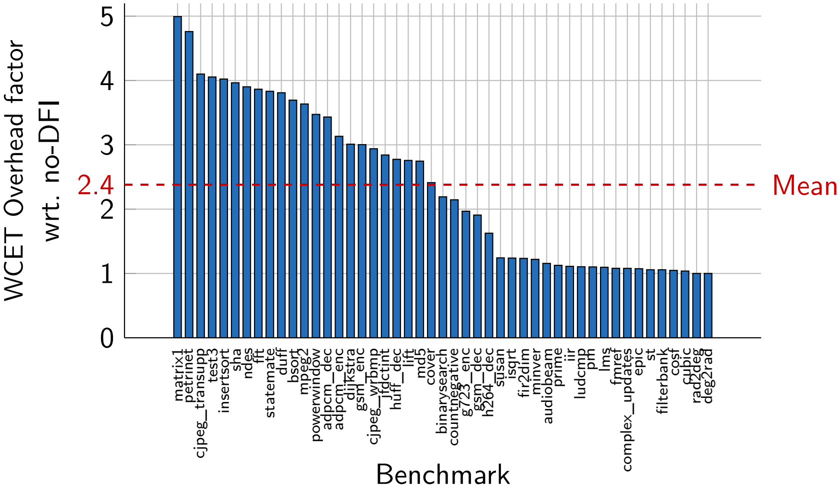


RT-DFI - Experimental Setup

- **Architecture** : RudolV (RISC-V processor)
- **WCET Estimator** : AiT (industrial standard)
- **ILP Solver** : CPLEX
- **Benchmark** : TacleBench (Real-time benchmarks, single task benchs)
- **Compilation flags** : -O1
- **Stopping condition** : Unchanged WCET
- **Baseline** : Overhead of DFI (as implemented in [4]) on estimated WCET

4. *Securing software by enforcing data-flow integrity*, Castro et al., USENIX '06.

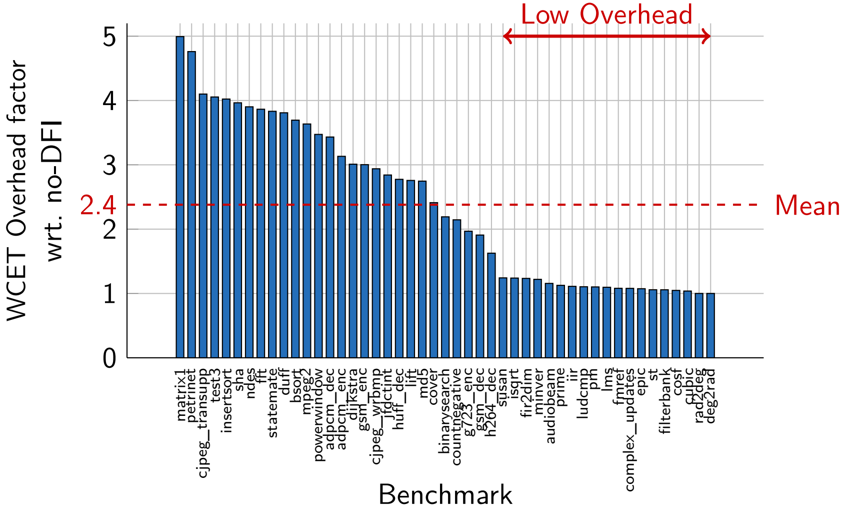
Baseline[4]



Strong guarantees at a cost

[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX ’06.

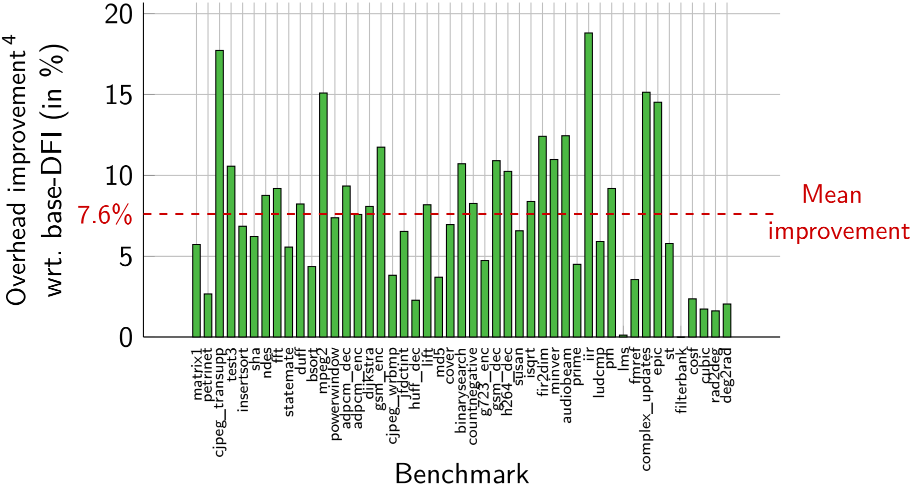
Baseline[4]



Strong guarantees at a cost

[4] *Securing software by enforcing data-flow integrity*, Castro et al., USENIX ’06.

RT-DFI - Improvement vs Baseline

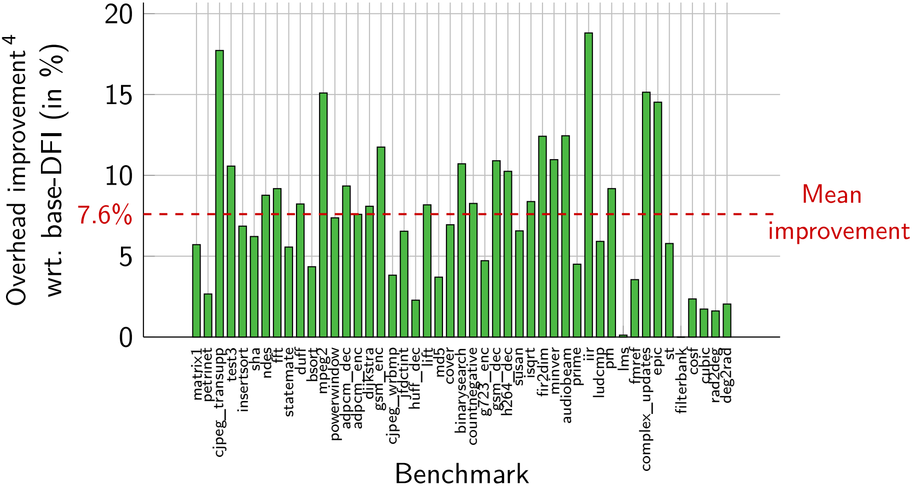


7.6%

Mean improvement

[4] Also containing improvement due to improved data-flow analysis

RT-DFI - Improvement vs Baseline



Note : No improvement past the first iteration

[4] Also containing improvement due to improved data-flow analysis

RT-DFI - Found Security Errors

RT-DFI - Found Security Errors

Data-flow errors are present in the benchmarks

RT-DFI - Found Security Errors

Data-flow errors are present in the benchmarks

Statically detected (aiT + DFI) :

RT-DFI - Found Security Errors

Data-flow errors are present in the benchmarks

Statically detected (aiT $+$ DFI) : rijndael_dec

RT-DFI - Found Security Errors

Data-flow errors are present in the benchmarks

Statically detected (aiT + DFI)	:	rijndael_dec
		rijndael_enc

RT-DFI - Found Security Errors

Data-flow errors are present in the benchmarks

Statically detected (aiT + DFI)	:	rijndael_dec rijndael_enc
Dynamically detected (executed with DFI)	:	

RT-DFI - Found Security Errors

Data-flow errors are present in the benchmarks

Statically detected (aiT + DFI)	:	rijndael_dec rijndael_enc
Dynamically detected (executed with DFI)	:	sha

Conclusion & Future Work

Conclusion & Future Work

Conclusion

Conclusion & Future Work

Conclusion

- Optimize DFI for the WCET (mean improvement : 7.6%)

Conclusion & Future Work

Conclusion

- Optimize DFI for the WCET (mean improvement : 7.6%)
- Iterative optimization is not efficient on the benchmarks

Conclusion & Future Work

Conclusion

- Optimize DFI for the WCET (mean improvement : 7.6%)
- Iterative optimization is not efficient on the benchmarks

Future Work

Conclusion & Future Work

Conclusion

- Optimize DFI for the WCET (mean improvement : 7.6%)
- Iterative optimization is not efficient on the benchmarks

Future Work

- Reducing tag address computation redundancy

Conclusion & Future Work

Conclusion

- Optimize DFI for the WCET (mean improvement : 7.6%)
- Iterative optimization is not efficient on the benchmarks

Future Work

- Reducing tag address computation redundancy
- Handling shared resources

Conclusion & Future Work

Conclusion

- Optimize DFI for the WCET (mean improvement : 7.6%)
- Iterative optimization is not efficient on the benchmarks

Future Work

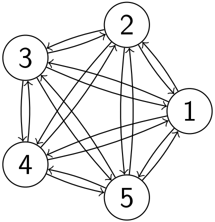
- Reducing tag address computation redundancy
- Handling shared resources
- WCET estimation for hardware-assisted DFI

Appendix - RT-DFI ILP key idea

Transform into graph :

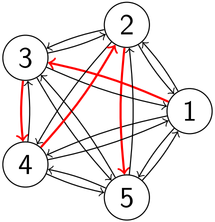
Appendix - RT-DFI ILP key idea

Transform into graph :



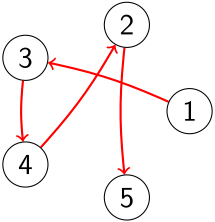
Appendix - RT-DFI ILP key idea

Transform into graph :



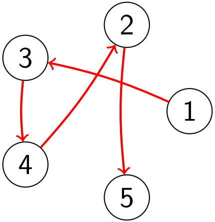
Appendix - RT-DFI ILP key idea

Transform into graph :



Appendix - RT-DFI ILP key idea

Transform into graph :

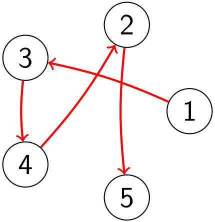


Tag representation

$$\mathscr{R} : \quad \begin{array}{c} 1 \mapsto 1 \\ 3 \mapsto 2 \\ 4 \mapsto 3 \\ 2 \mapsto 4 \\ 5 \mapsto 5 \end{array}$$

Appendix - RT-DFI ILP key idea

Transform into graph :



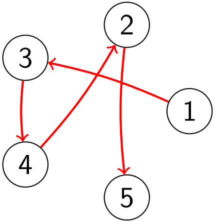
Tag representation

$$\mathscr{R} : \quad \begin{array}{l} 1 \mapsto 1 \\ 3 \mapsto 2 \\ 4 \mapsto 3 \\ 2 \mapsto 4 \\ 5 \mapsto 5 \end{array}$$

Interval order :

Appendix - RT-DFI ILP key idea

Transform into graph :



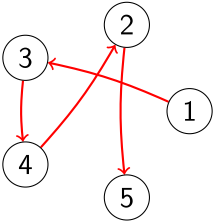
Tag representation

$$\mathscr{R} : \quad \begin{array}{c} 1 \mapsto 1 \\ 3 \mapsto 2 \\ 4 \mapsto 3 \\ 2 \mapsto 4 \\ 5 \mapsto 5 \end{array}$$

Interval order : $\phi\,(\,\mathrm{I}\,,\,\mathrm{t}\,)$

Appendix - RT-DFI ILP key idea

Transform into graph :



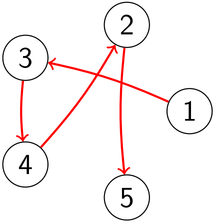
Tag representation

$$\mathscr{R} : \quad \begin{array}{c} 1 \mapsto 1 \\ 3 \mapsto 2 \\ 4 \mapsto 3 \\ 2 \mapsto 4 \\ 5 \mapsto 5 \end{array}$$

Interval order : $\phi(\mathtt{l}, \mathtt{t})$

Appendix - RT-DFI ILP key idea

Transform into graph :



Tag representation

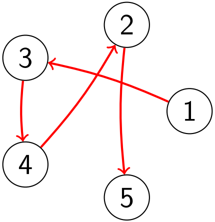
$$\mathscr{R} : \quad \begin{array}{l} 1 \mapsto 1 \\ 3 \mapsto 2 \\ 4 \mapsto 3 \\ 2 \mapsto 4 \\ 5 \mapsto 5 \end{array}$$

Interval order : $\phi(\mathsf{l}, \mathsf{t})$

order

Appendix - RT-DFI ILP key idea

Transform into graph :



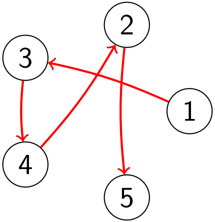
Tag representation

$$\mathscr{R} : \quad \begin{array}{c} 1 \mapsto 1 \\ 3 \mapsto 2 \\ 4 \mapsto 3 \\ 2 \mapsto 4 \\ 5 \mapsto 5 \end{array}$$

Interval order : $\underbrace{\phi}_{\text{order}}(\underbrace{\mathsf{l}}_{\text{load}}, \mathsf{t})$

Appendix - RT-DFI ILP key idea

Transform into graph :



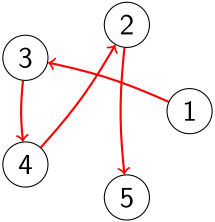
Tag representation

$$\mathscr{R} : \quad \begin{array}{c} 1 \mapsto 1 \\ 3 \mapsto 2 \\ 4 \mapsto 3 \\ 2 \mapsto 4 \\ 5 \mapsto 5 \end{array}$$

Interval order : $\underset{\text{order}}{\phi}(\underset{\text{load}}{\mathsf{l}}, \underset{\text{tag}}{\mathsf{t}})$

Appendix - RT-DFI ILP key idea

Transform into graph :



Tag representation

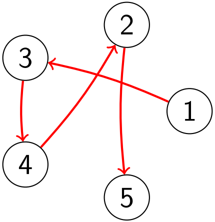
$$\mathscr{R}: \quad \begin{array}{l} 1 \mapsto 1 \\ 3 \mapsto 2 \\ 4 \mapsto 3 \\ 2 \mapsto 4 \\ 5 \mapsto 5 \end{array}$$

Interval order : $\underset{\text{order}}{\phi}(\underset{\text{load}}{\mathsf{l}}, \underset{\text{tag}}{\mathsf{t}})$

$$\mathscr{R}(t) \quad < \quad \mathscr{R}(t')$$

Appendix - RT-DFI ILP key idea

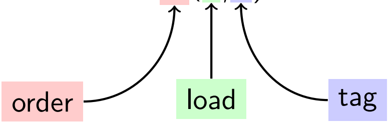
Transform into graph :



Tag representation

$$\mathscr{R}: \quad \begin{aligned} 1 &\mapsto 1 \\ 3 &\mapsto 2 \\ 4 &\mapsto 3 \\ 2 &\mapsto 4 \\ 5 &\mapsto 5 \end{aligned}$$

Interval order : $\phi(\text{l}, \text{t})$



order

load

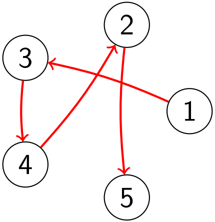
tag

$$\mathscr{R}(t) \quad < \quad \mathscr{R}(t')$$

$$\phi(l, t) \quad \neq \quad \phi(l, t')$$

Appendix - RT-DFI ILP key idea

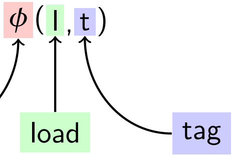
Transform into graph :



Tag representation

$$\mathscr{R} : \quad \begin{array}{l} 1 \mapsto 1 \\ 3 \mapsto 2 \\ 4 \mapsto 3 \\ 2 \mapsto 4 \\ 5 \mapsto 5 \end{array}$$

Interval order :



ϕ (l , t)

order

load

tag

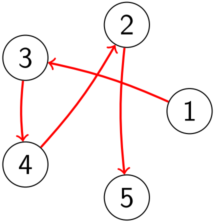
$$\mathscr{R}(t) \quad < \quad \mathscr{R}(t')$$

$$\boxed{\phi(l,t) \quad \neq \quad \phi(l,t')}$$

Appendix - RT-DFI ILP key idea

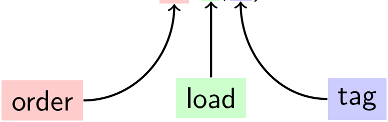
Tag representation

Transform into graph :



$$\mathscr{R} : \quad \begin{aligned} 1 &\mapsto 1 \\ 3 &\mapsto 2 \\ 4 &\mapsto 3 \\ 2 &\mapsto 4 \\ 5 &\mapsto 5 \end{aligned}$$

Interval order : ϕ (l , t)



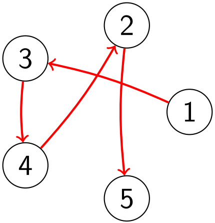
$$\mathscr{R}(t) \quad < \quad \mathscr{R}(t')$$

$$\updownarrow$$

$$\boxed{\phi(l,t) \quad \neq \quad \phi(l,t')}$$

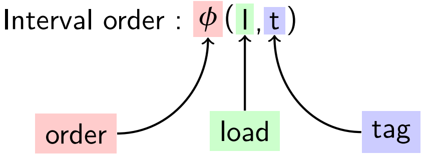
Appendix - RT-DFI ILP key idea

Transform into graph :



Tag representation

$$\mathscr{R} : \quad \begin{aligned} 1 &\mapsto 1 \\ 3 &\mapsto 2 \\ 4 &\mapsto 3 \\ 2 &\mapsto 4 \\ 5 &\mapsto 5 \end{aligned}$$



Interval order : ϕ (l , t)

order

load

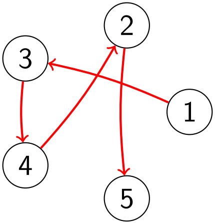
tag

$$\mathscr{R}(t) \quad < \quad \mathscr{R}(\,t''\,) \quad < \quad \mathscr{R}(t')$$

$$\boxed{\phi(l,t) \quad \neq \quad \phi(l,t')}$$

Appendix - RT-DFI ILP key idea

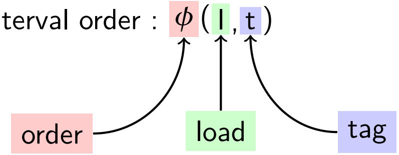
Transform into graph :



Tag representation

$$\mathscr{R} : \quad \begin{array}{l} 1 \mapsto 1 \\ 3 \mapsto 2 \\ 4 \mapsto 3 \\ 2 \mapsto 4 \\ 5 \mapsto 5 \end{array}$$

Interval order : $\phi(l, t)$



$$\begin{tikzcd}
 & t'' \not\in l \arrow[d] & \\
\mathscr{R}(t) \quad < & \mathscr{R}(t'') \arrow[d, leftrightarrow] & < \quad \mathscr{R}(t') \\
 & \boxed{\phi(l, t) \neq \phi(l, t')} &
\end{tikzcd}$$

Appendix - ILP for Tag Representation

$$\sum_{t,t' \in V} e_{t,t'} = Card(V) - 1 \tag{1}$$

$$\forall t \in V, entry_t = \sum_{t' \in T \setminus \{t\}} e_{t',t} \tag{2}$$

$$\forall t \in V, exit_t = \sum_{t' \in T \setminus \{t\}} e_{t,t'} \tag{3}$$

$$\forall t \in T, entry_t = 1 \tag{4}$$

$$\forall t \in T, exit_t = 1 \tag{5}$$

$$\forall t, t' \in V, (R_{t'} + 1) - Card(T) \cdot (1 - e_{t',t}) \leq R_t \tag{6}$$

$$\forall t, t' \in V, R_t \leq (R_{t'} + 1) + Card(T) \cdot (1 - e_{t',t}) \tag{7}$$

$$entry_{start} = 0, exit_{end} = 0, R_{start} = 0, R_{end} = Card(V) - 1 \tag{8}$$

Appendix - ILP for Interval Order

$$\forall t \in s_l, \Phi^+_{l,t} = \sum_{t' \in T} (e_{t,t'} \cdot \Phi_{l,t'})$$

$$\forall t \in V, \lambda^+_{l,t} = \sum_{t' \in s_l} e_{t,t'}$$

$$\forall t, t' \in s_l, \Lambda^+_{l,t,t'} = (R_t < R_{t'}) \cdot \lambda^+_{l,t} + (R_{t'} < R_t) \cdot \lambda^+_{l,t'}$$

$$\forall t, t' \in s_l, \Delta_{l,t,t'} = (\Phi_{l,t'} < \Phi_{l,t}) \cdot (\Phi_{l,t} - \Phi_{l,t'}) + (\Phi_{l,t} < \Phi_{l,t'}) \cdot (\Phi_{l,t'} - \Phi_{l,t})$$

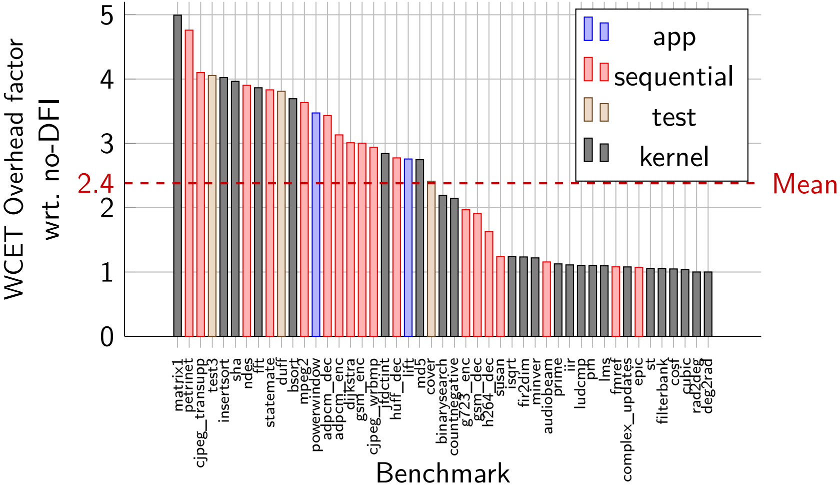
$$\forall t, t' \in s_l, \Delta^+_{l,t,t'} = (\Phi_{l,t'} < \Phi^+_{l,t}) \cdot (\Phi^+_{l,t} - \Phi_{l,t'}) + (\Phi^+_{l,t} < \Phi_{l,t'}) \cdot (\Phi_{l,t'} - \Phi^+_{l,t})$$

$$\forall t, t' \in s_l, \Gamma_{l,t,t'} = (R_t < R_{t'}) \cdot \lambda^+_{l,t} \cdot \Delta^+_{l,t,t'} + (R_{t'} < R_t) \cdot \lambda^+_{l,t'} \cdot \Delta^+_{l,t',t}$$

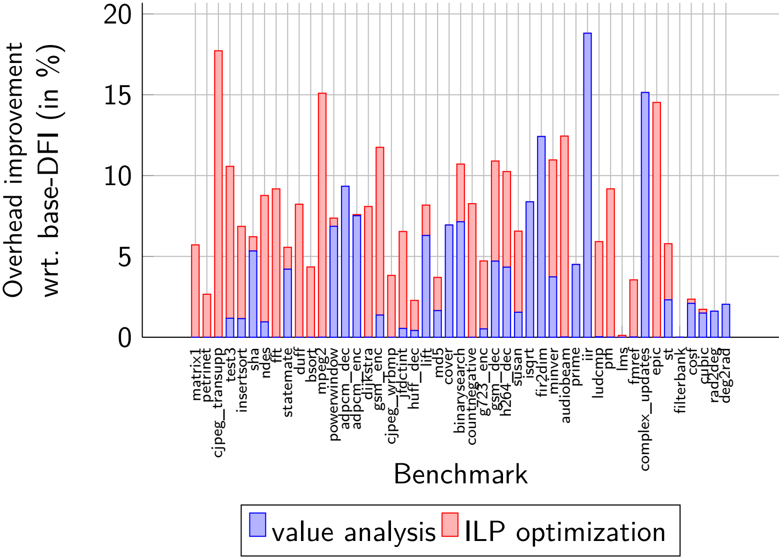
$$\forall t, t' \in s_l, \Delta_{l,t,t'} <= \Gamma_{l,t,t'} + (1 - \Lambda^+_{l,t,t'}) \cdot M$$

$$\forall t, t' \in s_l, \Delta_{l,t,t'} >= \Gamma_{l,t,t'} + (1 - \Lambda^+_{l,t,t'})$$

Appendix - Overhead per group



Appendix - Improvement per optimization



RT-DFI - Experimental time execution

Process part	Runtime (in avg.)
WCET Estimation	66%
Compilation	29%
ILP Solver	$<$ 40s (all)

Appendix - Program Instrumentation

Store

```
check_sandbox(&p)
tmp = kernel(&p)
store 1, tmp
store r1, &p
```

Load

```
tmp = kernel(&p)
load tag, tmp
check_tag(tag,{1,3})
load r1, &p
```

Appendix - Program Instrumentation

Store

```
check_sandbox(&p)
tmp = kernel(&p)
store 1, tmp
store r1, &p
```

Load

```
tmp = kernel(&p)
load tag, tmp
check_tag(tag,{1,3})
load r1, &p
```

check_sandbox : Ensure the store does not target RDT

Appendix - Program Instrumentation

Store

```
check_sandbox(&p)
tmp = kernel(&p)
store 1, tmp
store r1, &p
```

Load

```
tmp = kernel(&p)
load tag, tmp
check_tag(tag,{1,3})
load r1, &p
```

check_sandbox : Ensure the store does not target RDT

kernel : Compute the address of the tag in the RDT

Appendix - Program Instrumentation

Store

```
check_sandbox(&p)
tmp = kernel(&p)
store 1, tmp
store r1, &p
```

Load

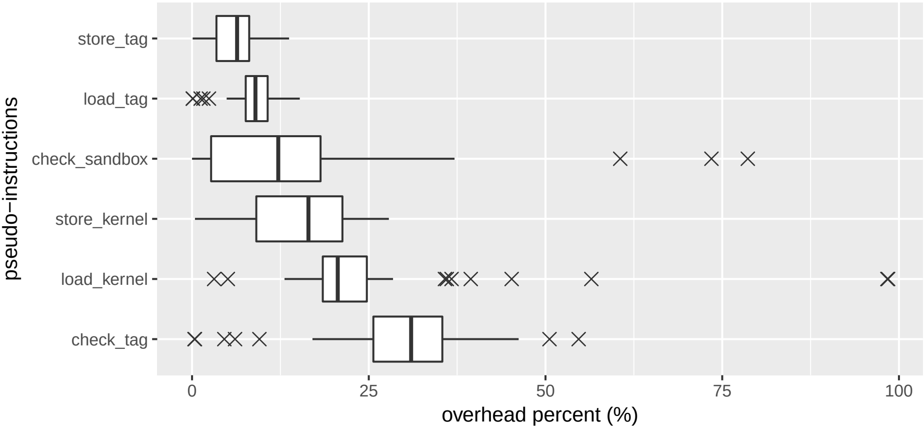
```
tmp = kernel(&p)
load tag, tmp
check_tag(tag,{1,3})
load r1, &p
```

check_sandbox : Ensure the store does not target RDT

kernel : Compute the address of the tag in the RDT

check_tag : Verifies that the loaded tag belongs to the valid tag set

DFI - Overhead partitioning


store_tag
load_tag
check_sandbox
store_kernel
load_kernel
check_tag
pseudo−instructions
overhead percent (%)